Міністерство освіти і науки України
Український державний університет науки і технологій

Комп’ютерні технології і системи

Комп’ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістр

на тему: Стратегія розробки ПЗ колективом слабо мотивованих виконавців

за освітньою програмою «Інженерія програмного забезпечення»

зі спеціальності: «121 Інженерія програмного забезпечення»

Виконав: студент групи: ПЗ2121

Микита НУШТАЄВ

Керівник: професор Віктор ШИНКАРЕНКО

Нормоконтролер: доцент Світлана ВОЛКОВА

Консультанти:

Техніко-економічні розрахунки доцент Микола ГНЕННИЙ

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies
Computer technologies and system
Computer information technology

Explanatory Note

To Master`s Thesis
master

on the topic: Strategy for the development of software by a team of weakly motivated workers

according to educational curriculum «Software engineering»

in the Speciality: «121 Software engineering»

Done by the student of the group: ПЗ2121

Mykyta Nushtaiev

Scientific Supervisor: professor Viktor Shynkarenko

Normative controller: associate professor Svitlana Volkova

Supervisors:

Technical and economic calculations associate professor Mykola Hnenny

Dnipro – 2022

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: комп'ютерні технології і системи
Кафедра: комп'ютерні інформаційні технології
Рівень вищої освіти: бакалавр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: 121 - Інженерія програмного забезпечення

(шифр та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри _____

_____ Вадим ГОРЯЧКІН

(підпис) (Ім'я ПРІЗВИЩЕ)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу магістр

(ступінь вищої освіти)

студенту Нуштаєва Микити Юрійовича

(Прізвище, Ім'я По батькові)

1. Тема роботи: Стратегія розробки ПЗ колективом слабо мотивованих виконавців

Керівник роботи: Шинкаренко Віктор Іванович, доктор технічних наук, професор

(Прізвище, Ім'я, По батькові, науковий ступінь, вчене звання)

затверджені наказом від "11"02 2022 р. № 173ст

2. Строк подання студентом роботи: 22.12.2022 р.

3. Вихідні дані до роботи: файли у форматі .docx в яких представлено склад команди та дані поради які дозволяють покращити роботу створеної команди

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):

4.1 Аналітична частина: провести аналіз предметної сфери, аналогів та поставити задачу

4.2 Основна частина: розробити систему для підвищення мотивації студентів, розробити додаток для визначення мотивації та додаток для розподілу студентів на групи, провести експерименти які визначать працездатність розробленої системи

4.3 Економічна частина: визначити витрати на проектування програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): ескізи інтерфейсу, алгоритм роботи додатку, діаграма класів

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Завдання видав:	Завдання прийняв:
Техніко-економічні розрахунки	доц. Гненний М. В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	25.07.22 - 30.07.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	01.08.22 - 22.08.22	
3	Постановка задачі, технічне завдання	23.08.22 - 04.09.22	
4	Створення тестової програми	05.09.22 - 10.11.22	
5	Перші тестування	11.11.22 - 29.11.22	30%
6	Аналіз результатів	30.11.22 - 05.12.22	
7	Розрахунок економічних показників	06.12.22 - 07.12.22	60%
8	Оформлення пояснювальної записки	08.12.22 - 19.12.22	100%
9	Демонстраційні матеріали	20.12.22 - 24.12.22	
10	Подання кваліфікаційної роботи до кафедри	25.12.22	
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	27.12.22	

Студент _____ Микита НУШТАЄВ

Керівник роботи _____ Віктор ШИНКАРЕНКО

Зміст

ВСТУП

Актуальність роботи. Проблема визначення мотивації конкретної людини існує давно. Дослідження у цьому напрямку проводили багато людей однак навіть після усіх досліджень немає як одного тесту для того щоб дізнатися рівень мотивації людини так і якоїсь універсальної методології задля підвищення рівня мотивації.

Вмотивованість людини дуже сильно впливає на якість та швидкість виконання роботи чи отримання знань. Проте багато людей не звертають на цю проблему належної уваги.

Актуальність роботи полягає у тому щоб допомогти групі студентів більш якісно виконувати групові завдання бо у програмістський сфері роботи командна робота є дуже важливою, адже одна людина не може створити великий проект чи робить его занадто довго. Пошук та вирішення проблем щодо мотивації групи студентів завжди є актуальною темою, адже люди можуть добитися разом набагато більшого аніж поодинці.

Створення програмного додатку який полегшить виявлення рівня мотивації та дає поради щодо її підвищення є вкрай необхідним, бо великі фірми не завжди вважають доцільним втратити на це свої ресурси через те, що вважають що їх працівники достатньо вмотивовані грошима або якимось іншими матеріальними нагородами чи статусними привілеями. На мотивацію студентів же взагалі звертають мало уваги коли турбуються про їх мотивацію адже вони самі пішли вчитися, той повинні бути вмотивовані вчити усе що їм дають.

Мета й завдання дослідження. Мета роботи полягає у дослідженні рівня мотивації групи студентів, пошуку основних проблем які знижують мотивацію та створення програмного продукту який опитує студента і видає поради щодо підвищення рівня вмотивованості як для одного студента так і для команди.

Завдання:

- розробити методику яка дозволяє оцінити рівень вмотивованості студентів;

- розробити програму яка підвищує рівень мотивації студентів.

Методи дослідження. Методи дослідження роботи на тему «Стратегія розробки ПЗ різноманітним колективом мало вмотивованих виконавців» викладені таким чином.

Для встановлення роботи здатності створеної системи для оцінки мотивації студентів проводилися експерименти. Була проведена серія тестів над групою людей мотивація яких була відома. Результати були порівнянні із заявленими. Також ж було виявлено чи збігається результат тестів із загальним результатом тестування.

Для перевірки працездатності розробленої системи мотивації студентів було проведено два тестування над командою, до і після виданих їй порад. Перше тестування виявило рівень мотивації та згуртованість групи до надання їй порад, а другий тестування було проведено після виконання групою завдання з використанням порад.

Наукова новизна. Наукова новизна цієї роботи полягає у створенні унікальної методології дослідження мотивації студента, що включає у себе дослідження різних сторін життя та особистості студента.

Також вперше створена методологія для підняття мотивації групи студентів під час виконання командної роботи.

Практичне значення. Отримані під час дослідження дані дозволяють знайти головні проблеми низької мотивації студентів на які можна вплинути.

Розроблений програмний продукт може використовуватись викладачами задля більш гарного розбиття групи студентів на команди під час виконання групової роботи та дослідження проблем вже існуючих команд.

Апробація результатів дослідження та публікації.

Виконано доповідь на XV International Conference «Modern information and communication technologies on a transport, in industry and education» for the 100th anniversary of Professor Yevgen Mironovich Shafit.

1. АНАЛІЗ ПРОБЛЕМИ РІЗНОЇ МОТИВАЦІЇ ГРУПИ СТУДЕНТІВ ПРИ ВИКОНАННІ ГРУПОВОГО ПРОЕКТУ

1.1 Аналіз предметної сфери

Мотивація під час навчання є однією з головних проблем для викладачів: не усі студенти готови однаково гарно працювати. Ця проблема посилюється під час надання студентам групових завдань, адже з’являться недоліки мотивації – зниження сприятливого впливу мотивації на студентів, зниження самої мотивації до ефективної роботи. До них будуть і спроби деяких членів команди перекласти більшу частину роботи на плечі більш старанних студентів, що підриває довгострокову вмотивованість старанних студентів до подібної роботи, і те, що відсутність інструментів перевірки вкладу кожної окремої людини змушує багатьох працювати не на повну силу, і неминучі тертя всередині групи, які часто перетікають у конфлікти, та багато інших факторів. Мотивація до групової роботи – серйозна тема для досліджень, і над цією темою працює багато видатних учених.

Центр педагогічної майстерності Eberly[1] вважає, що організація групової роботи має витрати і студентам – учасників команди, й у викладачів. Ці витрати являють собою додаткову роботу, яка необхідна для ефективної роботи групи, так і додатковий час, який витрачається на організацію роботи.

Більше того, є дослідники, які послідовно висловлюються проти групових завдань, принаймні на рівні бакалаврських програм. Наприклад, Пол Хеллер дотримується саме такої думки. Він вважає, що у груповій роботі навіть сильні студенти не мають можливості проявити себе, або, навпаки, змушені брати на себе левову частку роботи. Проте, приходячи в реальне бізнес-середовище, навіть ті студенти, які не мають навичок групової роботи, дуже швидко їх набувають і чудово працюють у командах. Команди на робочому місці принципово відрізняються від студентської групи, що працює над загальним завданням, і, отже, зникають багато проблем та недоліків, притаманні командним завданням в університеті.

Окремо треба відмітити ключові відміни від командної роботи у бізнес-середі[2]:

- немає одного з ключових аспектів мотивації: грошового;
- різний інтелектуальний рівень студентів на відміну від більш професіональної команди де робітники підбирались під конкретну задачу чи хоча б напрям;
- т. я. на кону стоїть кар’єрний ріст і заробітна плата то оцінка вкладу кожного працівника буде ретельно перевірятися начальниками.

Однак групова робота все ще має багато плюсів:

- підвищується навчальна та пізнавальна мотивація;
- знижується рівень тривожності учнів, страху виявитися неуспішним, некомпетентним у вирішенні якихось завдань;
- у групі вища навченість, ефективність засвоєння та актуалізації знань;
- поліпшується психологічний клімат у групі.

Сумуючи усе вищесказане можна побачити що робота у команді дуже поширена і проблеми зв’язані з нею досліджуються увесь час та шукають шляхи їх вирішення, проте основна робота проводиться орієнтуючись на проблеми груп робітників ігноруючи проблемі груп учнів, які хоча і схожі, проте мають ряд ключових відмінностей.

1.2 Дослідження предметної галузі

Основний сенс мотивації це спонукати людину до дії. Для того щоб людина вмотивувалася робити якусь справу її треба зацікавити метою роботи, а для цього потрібно описати подальші перспективи її використання. В основі мотивації дії лежить психологія суб'єкту. Усі психологічні теорії та методи, які лежать в основі система мотивації, націлені на те, щоб побувати у співробітника намір якісно виконувати роботу. Основною ціллю при формуванні системи

мотивації дії є формування у людини зацікавленості в своїй роботі яку вона виконує, досягти такої мети: людина повинна сама хотіти виконати поставлене завдання, тобто сформувати у людини розуміння того що виконання цього завдання перш за все потрібно їй самій, а не людям навколо неї.

На практиці мотивація це не одна конкретне дія, а комплекс мер. Більшість теорій мотивації покладений в основу практик мотивації спирається на психологічні поняття потреба та очікування.

Потреба - внутрішній стан психологічного чи функціонального відчуття недостатності чогось[3]. Потреби по-різному виявляються залежно від ситуаційних чинників. Потреби розрізняють:

- за сферами діяльності: потреби праці, пізнання, спілкування, відпочинку;
- по об'єкту потреб: матеріальні, духовні, етичні, естетичні та інші потреби;
- по функціональній ролі: домінуючі/другорядні, центральні/периферичні, стійкі/ситуативні потреби;
- за суб'єктом потреб: групові, індивідуальні, колективні, громадські.

Очікування - подія, яка розглядається як найбільш вірогідна в ситуації невизначеності та передчуття чогось, також більш менш реалістичне припущення щодо майбутньої події. Також очікуванням називається перебування у стані націленості уваги на передбачувану в майбутньому подію[4].

На початку 20 століття почали формуватися перші системи мотивації людей. Тоді посилювалась конкуренція в економічних відносинах і з'явилася потреба чим завгодно підвищити продуктивність праці.

Для того щоб вирішити цю задачу психологи почали вивчати психотипи працівників і створювати концепції які б дозволяли максимально задіяти потенціал співробітників задля досягнення цілей компанії.

Результатом цих досліджень стали три групи теорій, вони пояснюють виникнення потреб і шукають різні варіанти використання цих знань щоб підвищити у працівників цікавість до труду.

Перша група це змістовні теорії мотивації, вони намагаються знайти серед потреб людини такі які зумовлювали відношення до праці. Тобто вони використовують для мотивації потреби до праці у людей[5].

Найвідомішою теорією є піраміда потреб Маслоу[6]. У її основі лежать фізіологічні потреби людини (наприклад потреба у їжі сні) а на самому високому рівні розвинені потреби особистості, наприклад самореалізація. Маслоу вважав що коли людина закриває потребу у нижньому різні вона переходить до більш високого і на цьому будуються багато теорії мотивації. Наприклад якщо зарплатня людини перекриває її потреби (вона може задовільнити свої фізіологічні потреби та фізіологічні потреби своєї сім'ї) то за для мотивації цієї людини запропонувати їм премію чи підвищення окладу буде недостатньо, треба намагатися зацікавити цю людину завданнями які вона отримує.

Далі йде двофакторна теорія мотивації Фредеріка Герцберга[7].

Теорія виходить з потреб людини. На його прохання, 200 інженерів та бухгалтерів однієї великої фірми описали ситуації, коли їхня робота приносила їм особливе задоволення і коли вона особливо їм не подобалася. В результаті експериментів Герцберг дійшов висновку, що існують дві основні категорії факторів оцінки ступеня задоволеності від виконаної роботи: фактори, що утримують на роботі, та фактори, що мотивують до роботи[].

Чинники, які утримують роботі (гігієнічні чинники) — адміністративна політика підприємства, умови праці, величина зарплати, міжособистісні стосунки з начальниками, колегами, підлеглими.

Фактори, що мотивують до роботи (мотиватори) - досягнення, визнання заслуг, відповідальність, можливості кар'єрного зростання.

Гігієнічні чинники пов'язані з середовищем, у якому виконується робота. За теорією Герцберга, відсутність чи нестача гігієнічних факторів призводить до

незадоволеності людини своєю роботою. Але, якщо вони представлені в достатньому обсязі, самі собою вони задоволення не викликають і не здатні мотивувати людину до необхідних дій.

Відсутність мотиваторів, а вони пов'язані з характером і суттю самої роботи, не веде до незадоволення людей роботою, проте їхня присутність належною мірою викликає задоволення і мотивує працівників до необхідних дій та підвищення ефективності.

Слід звернути увагу, що Герцберг зробив парадоксальний висновок у тому, що вести перестав бути мотивуючим чинником. З іншого боку, вона є мотиватором лише до певного моменту.

Друга група це процесуальні теорії мотивації, не заперечуючи існування і впливом геть людини її потреб, вказують те що, що поведінка людей формується як під їх впливом, а як і визначається: сприйняттям, очікуваннями, що з конкретної ситуацією, і можливими наслідками обраного ними типу поведінки. Вони описують динаміку взаємодії різних мотивів та досліджують, як спонукається і спрямовується поведінка людини.

Теорія очікувань грунтується на тому, що наявність інтенсивної потреби ніяк не вважається єдиною важливою умовою мотивації людини на досягнення певної мети. Вперше було викладено канадським психологом Віктором Врумом 1964-го року у роботі «Праця і мотивація»[8].

Теоретично виділяють три основних моменти мотивації людини[9]:

- очікування;
- сприяння;
- валентність.

Очікування, згідно з теорією, очікування є уявленням людини про те, що витрачені ним зусилля приведуть до очікуваного та бажаного результату.

Сприяння, Врум писав про те, що під сприянням розуміється надія людини на винагороду залежно від результатів.

Валентність є передбачуваною мірою відносного задоволення або незадоволення, яка виникає внаслідок отримання певної винагороди.

Мотивація, згідно з теорією, є складовою функцією всіх трьох компонентів. З цього випливає, що вона буде високою тоді, коли всі складові високі. У випадку, якщо один з цих трьох компонентів дорівнює нулю, загальний рівень мотивації також дорівнює нулю. Якщо робітник вірить, що його зусилля приведуть до результату, який буде винагороджено, мотивації не буде, якщо валентність очікуваної винагороди дорівнює нулю.

У той же час мотивація не рівноцінна результатам роботи. Ця теорія визнає, що мотивація одна із кількох важливих параметрів, визначальних результат. Зокрема, теорія передбачає, що навички та здібності роблять великий внесок у результат трудової діяльності: деякі люди більш пристосовані до роботи, ніж інші, завдяки властивим їм індивідуальним рисам, умінням і талантам.

Сприйняття ролі людиною також впливає результат роботи. Доки розбіжності у визначенні службових обов'язків, може страждати продуктивність.

Теорія справедливості[10], або рівності фокусується на прагненні співробітників підтримувати рівність між вкладом, який вони привносять у роботу, та результатами, які вони отримують від неї, порівняно з вкладами, що сприймаються, та результатами інших. Ситуація вважається справедливою лише в тому випадку, якщо співвідношення докладених зусиль і очікуваної винагороди є рівним порівняно з людиною, що порівнюється. Теорія була розроблена Джоном Стейсі Адамсом у 1963 році на підставі результатів досліджень, проведених ним у компанії General Electric[11].

Моде́ль По́ртера-Ло́улера — комплексна теорія мотивації, що містить елементи попередніх теорій. На думку авторів, мотивація є одночасно функцією потреб, очікувань і сприйняття працівниками справедливої винагороди. В моделі Портера-Лоулера фігурує 5 основних ситуаційних факторів:

- витрачені працівником зусилля;

- сприйняття;
- отримані результати;
- винагородження;
- ступінь задоволення.

Відповідно до моделі Портера-Лоулера[12]:

- рівень зусиль, що витрачаються (3) залежить від цінності винагороди (1) і від впевненості в наявності зв'язку між витратами зусиль і винагородою (2);
- на результати, досягнуті працівником (6), впливають три фактори: витрачені зусилля (3), здібності і характерні особливості людини (4), а також від усвідомлення нею своєї ролі в процесі праці (5);
- досягнення необхідного рівня результативності (6) може призвести до внутрішньої винагороди (7а), тобто відчуття задоволеності роботою, компетентності, самоповаги, і зовнішньої винагороди (7б) — похвала керівника, премія, просування за службою тощо;
- пунктирна лінія між результатами і винагородженням, що сприймається як справедливе (8) виходить з теорії справедливості і показує, що люди мають власну оцінку ступеня справедливості винагороди;
- задоволення є результатом зовнішнього і внутрішнього винагородження з урахуванням їх справедливості;

3 група це теорії мотивації засновані на відношенні людини до праці. Теорії близькі до процесуальних але припускають більш високу свідомість суб'єкта мотивації. Приклади таких теорій це теорія «X» та «Y» Дугласа Макгрегора та теорія «Z» Вільяма Оучі.

Теорія X і Теорія Y - теорії Дугласа Макгрегора про мотивацію людей і поведінку в управлінні.

Теорія X припускає, що працівники спочатку ліниві і будуть по можливості уникати роботи. Тому працівники повинні бути під пильним наглядом, для чого розробляються комплексні системи контролю.

Менеджер з теорії X, як правило, вважає, що все має закінчуватися покладанням відповідальності на кого-небудь. Він вважає, що всі передбачувані працівники шукають вигоди для себе. Як правило, такі керівники вважають, що єдина мета зацікавленості працівників у роботі – це гроші. У більшості випадків вони звинувачують насамперед людину, не ставлячи питання про те, що, можливо, звинувачувати треба систему, стратегію чи відсутність підготовки.

Теорія Y передбачає, що працівники можуть бути амбітними, мати внутрішні стимули, прагнути взяти на себе більше відповідальності, здійснювати самоконтроль та самоврядування. Вважається, що співробітники одержують задоволення від своїх обов'язків, пов'язаних як з розумовою, так і фізичною працею.

Менеджер Теорії Y вважає, що за сприятливих умов більшість людей хочуть працювати добре і, що робоча сила має резерв невикористовуваних творчих здібностей. Вони вірять, що задоволення від хорошого виконання своєї роботи саме собою є потужним стимулом. Менеджер Теорії Y намагатиметься усунути перешкоди, що заважають працівникам повністю реалізувати себе[13].

Теорія Оучі (теорія "Z") - модель управління, що поєднує в собі досвід американського та японського менеджменту[14].

Основні положення теорії Оучі:

- довготривале наймання: це положення У. Оучі запозичив у японської моделі менеджменту. У великих японських фірмах працівники наймаються довічно. Подібна політика фірми підтримує у працівників відчуття, що фірма бере за них відповідальність і стає другою домівкою;
- прийняття рішень з урахуванням консенсусу, групове прийняття рішень, у своїй ініціатива грає певну роль;

- відповідальність за результати рішень є колективною, як і в американській системі. Тобто, хоча рішення і приймається на основі консенсусу, озвучує його менеджер, оберігаючи тим самим індивідуальність своїх співробітників;
- просування кар'єрними сходами повільне, безпосередньо залежить від кількості років, які співробітник пропрацював на компанію, а також його власного віку;
- механізм контролю є неформальним і не має чітких та ясних правил, як у японській традиції, проте інструменти здійснення контролю суворо визначені;
- увага до працівника: Оучі адаптував японську традицію сприйняття компанії як другого будинку.

Таким чином, ідеальна модель, по Оучі, є поєднанням прагнення людини до знаходження в колективі, а також його індивідуалістичні риси [2, 3, 4].

1.2 Огляд програмних аналогів

Хоча тема мотивації персоналу та учнів дуже важлива і постійно досліджується, однак додатків ціллю яких би це було майже немає. Розділити аналоги можна на дві категорії: програми потрібні для організації роботи однаково вмотивованої групи або додатки що потрібні для мотивації однієї людини. Був виділений ряд критеріїв до програми та проведений пошук аналогів, їх порівняння наведено у таблиці 1.

ToDoist — веб-сервіс та повноцінний додаток для управління персональними та робочими завданнями. Існує три версії програми: дві для особистого користування (Free-to-pay та Premium), одна для бізнесу (Business)[15].

ToDoist офіційно підтримує 13 платформ - iOS, Android, Mac, Windows та інші. Дані зберігаються за допомогою онлайн-синхронізації та резервного копіювання.

У 2015-2016 році програма пережила редизайн. У рамках редизайну команда працювала над зовнішнім виглядом продукту та підкоригувала нутрощі. Розробники використовували у розробці алгоритми машинного навчання. Так ToDoist навчився пропонувати користувачеві цікаві завдання, а також визначати відповідний для їхнього виконання день.

З усіх розглянутих аналогів має найбільше інструментів для підвищення мотивації.

Функціонал:

- швидке додавання завдань, проектів;
- терміни виконання завдань, що повторюються;
- повне відображення окремого завдання;
- створення власних розділів, підрозділів;
- розміщення пріоритетів для завдань;
- делегування завдань працівникам;
- додавання коментарів до завдань, проектів;
- отримання очок «Карми». Окуляри «Карми» допомагають відстежувати прогрес та продуктивність;
- візуалізація продуктивності;
- перегляд вже виконаних завдань за весь час.

Pyrus – сервіс для організації робочих процесів, у якому впроваджено бізнес-месенджер, інструменти для управління завданнями та узгодження заявок. У системі можна делегувати завдання і контролювати їх виконання. Завдяки отриманим звітам дається оцінка ефективності роботи команди та результатів за кількістю заявок, вносяться зміни до поточних бізнес-процесів[16].

Функціонал:

- постановка задач із можливістю додавання пов'язаних завдань для обговорення етапів виконання;
- форми для організації бізнес-процесів – узгодження документів, заявки на витрати, збирання звернень та запитів клієнтів через зовнішні форми;
- спільна робота з файлами – прикріплення документів до завдань та форм, додавання посилань на файл;
- пошук завдань з різних атрибутів – учасників обговорення, ключових слів, назв документів, товарів тощо;
- звіти – автоматичний збір статистики з проектів та бізнес-процесів.
- готові інтеграції з Google Apps, Drive, Dropbox, Box, Active Directory, 1С: Підприємство;
- Pyrus API для розробників.

Доступ до сервісу можливий через веб-версію, мобільні програми для iOS та Android, через програму Pyrus Sync, яка інтегрується з хмарними сервісами.

MeisterTask - інструмент для управління завданнями та проектами. За допомогою MeisterTask команди можуть налагодити планування завдань, їх виконання та отримати дані щодо ефективності роботи. Інструмент підійде для маленьких та середніх команд, що працюють за проектним підходом, маркетологів, HR-фахівців, рекламників, розробників та інших відділів. Управляти проектами можна через мобільний та десктопний додаток, і в браузері[17].

MeisterTask дозволяє створювати необмежену кількість проектів та завдань. Проект є канбан-дошкою, на якій розміщуються завдання. Колонки на дошці можна перейменовувати та додавати нові. Для фокусування уваги на завдання керівники можуть обмежити кількість відкритих завдань на одного співробітника. Для економії кроки, що повторюються, при роботі з завданнями можна автоматизувати, а також створити регулярні завдання. У завдань є

призначені особи та спостерігачі, до них можна додавати поля користувача, помічати їх тегами, встановлювати терміни виконання, прикріплювати файли. У кожного користувача є дашборд, де він може переглядати призначені на нього завдання, створювати особисті завдання та додавати звіти користувача.

Для планування у керівників є і співробітників MeisterTask є діаграма Ганта, на якій можна переглядати завантаженість роботи. Також є можливість аналізувати роботу за допомогою деталізованих звітів, функції відстеження часу. Проекти для команд створюють адміністратори, вони можуть керувати доступом та ролями, налаштовувати спільні повідомлення та створювати групи користувачів.

Функціонал:

- ведення проектів на канбан-дошці;
- перегляд завантаженості на діаграмі Ганта;
- настроювані проекти та поля для завдань;
- фільтри та інтерактивний пошук;
- відстеження часу працівників;
- коментарі до завдань та повідомлення;
- імпорт даних з Wunderlist, Asana та Trello;
- інтеграція з хмарними сервісами;
- особистий дашборд, що настроюється.

Loop – це програма з відкритим вихідним кодом для відстеження прогресу у розвитку звичок. Loop набагато простіше, ніж решта програм для цих цілей, завдяки полегшеному інтерфейсу та більш зрозумілій технікою роботи. Програма виглядає як календар, куди ви можете додавати навички та вибирати час роботи над ними. При додаванні мети ви можете вибрати їй назву, колір, уподобання при відстеженні, дні повторення та нагадування. У разі потреби ви зможете відредагувати ці налаштування у будь-який час[18].

Після того, як ви створите звичку, вона додасться на головний екран програми. Також там показуватиметься основна інформація щодо неї за останні п'ять днів. Звідси можна легко контролювати все, пов'язане з вашими завданнями, простим вибором потрібної дати. Натискання на ціль на головному екрані призведе до показу докладніших даних: силу навички, історія звикання, найкращі результати та частоту звички.

У версії 1.8 з'явилася нова гістограма, що показує кількість повторень, що виконуються щотижня, місяць чи рік. Тепер виконання навичок у нерегулярні робочі дні більше не порушує прогрес завдання. Важливо, що Loop не має можливості реєстрації облікового запису для подальшої синхронізації даних, але ви можете завантажити всі дані у файлі формату CSV і завантажити їх назад, якщо це знадобиться.

Таблиця 1.1 - Порівняння функціональних можливостей

Функціональність	Наявність у програмі			
	Loop	ToDoist	Pyrus	MeisterTask
Терміни виконання завдань	+	+	+	+
Створення розділів, підрозділів	-	+	+	+
Розміщення пріоритетів для завдань	-	+	-	-
Додавання коментарів до завдань, проектів	+	+	+	+
Візуалізація продуктивності	+	+/-	+/-	+/-

Продовження таблиці 1.1

Інтеграція з хмарними сервісами	-	-	+/-	+
Форми для організації бізнес-процесів	-	-	+	-
Звіти	+	+	+/-	+/-
Готові інтеграції				
Інструменти підвищення вмотивованості виконавців проекту	+	+/-	-	-
Наявність декількох ролей	-	+/-	+/-	+/-
Можливість бачити прогрес інших учасників	-	-	-	-
Історія роботи	+	+	+	+
Нагадування про виконання завдання	+	-	-	-
Система балів	-	+	-	-

У таблиці. 1.1 використано такі умовні позначення: «+» означає, що функціональна можливість наявна, «+/–» – наявна частково, «–» – відсутня.

У таблиці 1.2 наведено опис переваг та недоліків оглянутих мною програмних аналогів.

Таблиця 1.2 – Опис переваг та недоліків аналогів

Назва програми	Переваги	Недоліки
ToDoist	Однією з основних переваг є наявність системи продуктивності та наявність навчального контенту такого як шаблони, статті та поради в середині додатку; можливість інтегрувати інші сервіси; підходить для роботи групи, має усі необхідні інструменти для планування завдання.	Є проблеми з організацією підзадач та підпроектів, незручно додавати їх під час роботи та правильно виставляти ролі на них; нема вбудованого календаря, що значно поскладнює розподіл задач за часом ті вимушує використовувати додаткові додатки; складності з переносом завдань; не якісна технічна реалізація, наявність багів та видно незавершеність програми.
Pyrus	Має можливість налагодити інтеграцію з багатьма іншими сервісами; зручна система звітів щодо виконаних завдань; сортування завдань по різним атрибутам; форму для організації бізнес-процесів де присутні як і шаблони так і можливість самому їх створювати; можливість робити об’яви які побачать усі учасники проекту; сортування завдань за їх пріоритетом; наявність календаря з візуалізацією справ; аналітика щодо роботи групи чи окремої людини	Дуже лаконічна програма, не має ряду функціоналу на відміну від інших; нема діаграми Ганту, замало статистики щодо як завдань які виконуються так і загальних результатів роботи над проектом

Продовження таблиці 1.2

MeisterTask	Можливість підключатися до інших подібних додатків та розширювати їх функціонал; функціонал дозволяє легко додавати нові задачі та підзадачі; відстеження часу що тратиться на задачі;	Відсутню діаграма Ганта, немає ніякого виводу звітів чи інформації у вигляді графіків, діаграм чи схем; налаштування додатку має у більшості косметичний характер аніж практичний
Loop	Проста у освоєні, реалізує найпопулярніщі та прості способи прогамної мотивації людей.	Є програмою для одного користувача, нема підтримки групових завдань.

1.3 Призначення та сфера застосування

Програмний продукт призначений проводити тестування людей та виявляти їх рівень мотивації навчання і визначати ролі які вони займають в команді. Друга частина потрібна для розбиття групи людей на низку команд с описом їх слабких сторін та шляхами їх вирішення.

Сферою застосування розробленого програмного продукту можуть бути всі галузі, де потрібно покращити чи створити команди та навчальні галузі де треба дізнатися рівень мотивації студентів.

1.4 Постановка задачі

Слід виділити два незалежні модулі програмного комплексу: тестування людини та розбиття групи людей на команди.

Програмний додаток повинен володіти наступним функціоналом:

- проводити тестування людей і визначати їх рівень мотивації та ролі для роботи у команді;
- розбивати групу людей на команди, виявляти слабкості команд та пропанувати шляхи вирішення.

Вхідні дані:

- варіанти відповідей від користувача;
- номер групи студентів з якою треба розбити на команди.

Вихідні дані:

- для модулю першої програми – файл формату .json з результатами тестування людини;
- для другого модулю – файли формату .docx з складами команд, їх проблемами та шляхами покращення роботи команд.

Висновки до 1 розділу

Аналіз предметної області допоміг виявити основні методології мотивації які використовуються при створенні системи мотивації. За їх допомогою можна зрозуміти чому слід приділяти увагу при побудові таких систем. Погляд аналоги в показав що серед існуючого програмного забезпечення нема прямих аналогів додатку що розробляється.

Стало зрозуміло що результатом роботи буде система мотивації яка допоможе викладачам покращити мотивацію команди студентів. Програмне забезпечення яке буде розроблено під час дипломної роботи слугує помічником для виявлення рівня мотивації студентів та повинно розподілити групу студентів на команди та дати цим командам поради які допомогли їм виправити свої недоліки.

2. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНИХ СИСТЕМ ДОСЛІДЖЕННЯ МОТИВАЦІЇ СТУДЕНТІВ ТА СИСТЕМИ МОТИВАЦІЇ СТУДЕНТІВ

2.1 Опис системи яка досліджується

Дослідження було вирішено поділити на декілька частин:

Спочатку був проведений опит для аналізу проблем та пошук факторів які тормозили виконання лабораторних робіт під час навчання упродовж декількох років.

Однією з основних проблем при виконанні будь-якої лабораторної чи практичної роботи під час навчання є не зрозумілість студенту навіщо потрібно робити цю роботу. Студент може бути незнайомий з необхідністю вивчати той чи матеріал для покращення своїх знань та необхідність цих навичок задля роботи за спеціальністю.

Робота у команді дуже складна річ. Звичайно, якщо робити якийсь проект разом з іншими людьми то можна зробити набагато більший обсяг роботи та швидше аніж якщо усе робити одному, тому робота у команді є фундаментальним навиком для будь-якої людини. Однак якщо неправильно влаштовувати групову роботу то ефект буде зворотній, тобто усі люди у команді працюючи окремо змогли б зробити більше аніж працюючи разом. Це на мою думку одна з найбільших проблем під час роботи у групах. Задля їх вирішення треба дослідити принципи формування груп за методологією Белбіна[19].

Вирішення проблеми роботи у команді, а саме формування гарної команди має бути досягнуто за допомогою методології Белбіна. Отримані групи мають мати у собі таких виконавців у яких будуть різні ролі і навіть якщо у команді не будуть представлені усі необхідні задля гарної роботи команди ролі, то можна буде сказати чого саме не вистачає команді та доповнити відсутність тих чи інших ролей спеціальними діями.

Даний метод повинен покращити мотивацію студентів через те що їм буде цікаво попрацювати за цією методологією та отримані під час виконання завдання навички допоможуть їм під час їх подальшої праці. Адже, можливо, що

під час роботи над реальним проектом їх команда теж буде мати деякі труднощі з організацією і надбаний досвід допоможе подолати ці проблеми.

Після визначення проблеми, та шляху було почато два дослідження: аналіз мотивації людей у групі та пошук причин які безпосередньо впливають на загальну мотивацію; пошук людей які не мають мотивація та хист працювати у групі, тобто люди які із вірогідністю 99 відсотків будуть заважати роботі команди та із-за них команда потерпить крах.

Друге дослідження це розділення групи на команди та пошук слабких сторін окремо взятих команд. Дослідження повинно показати підвищиться чи ні мотивація команди студентів якщо що вказати їм на їх слабкі та сильні сторони та показати як можна виправити недоліки.

Далі наведено методи якими досліджувались команді навички та мотивація студентів а також метод за допомогою якого можна розділити групу студентів на команди та за якими ознаками.

- оцінка мотивації досягнення;
- вивчення мотивації навчання у вузі;
- дослідження соціальної та пізнавальних мотивацій;
- мотиви вибору професії;
- задоволеність учбовою діяльністю;
- оцінка екстравертності;
- особливості людини як командного гравця.

2.2 Опис тестів які лягли у систему визначення мотивації студентів

2.2.1 Оцінка мотивації досягнення

Мотивація досягнення виявляється у прагненні до поліпшення результатів, наполегливості у досягненні своїх цілей чи незадоволеністю досягнутим, бажанням зробити краще та досягти свого будь-якою ціною. Багато досліджень показали що ця мотивація є однією з головних при отриманні досягнень та успіху у життєдіяльності[20].

Результат тесту показує те, який в людини рівень мотивації досягнення: низький, середній чи високий. Також ці пункти можна розбити на декілька підпунктів для більш точного встановлення рівня мотивації але через деяку похибку у відповідях було вирішено зупинитися лише на цих трьох основних шкалах[21].

2.2.2 Вивчення мотивації навчання у вузі

Наступне дослідження це методика дослідження мотивації навчання у ВУЗі Т. І. Ільїної [22].

Тестування побудоване на цій методиці вираховує 3 шкали мотивації навчання в ВУЗі. Перша це придбання знань, друга - опанування професії та третя це отримання диплому[23].

Переважання мотивів по першим двом шкалам говорить про гарний вибір студентам професії та задоволеність нею, що каже нам про високу мотивацію до навчання за обраним напрямком.

Висока третя шкала, отримання диплома, сама по собі не є поганою, проте якщо буде переважати вона, то це каже про мотивацію студента лише до отримання диплома що не є гарною мотивацією та у студента у майбутньому вона може впасти, навіть якщо зараз є нормальною чи високою.

Якщо усі три шкали малі це каже про вкрай низьку мотивацію студента

Шкала оцінюються так: перша від нуля до 4.2 це погана мотивація, від 4.3 до 8.4 це середня мотивація і від 8.5 до 12.6 це висока мотивація. Друга та третя шкали оцінюються так від 0 до 3 низька, від 4 до 7 середня та від 8 до 10 висока.

2.2.3 Дослідження соціальної та пізнавальних мотивацій

Далі тест на мотивацію мову діяльності розглядання рівнів і типів мотивації[24]. Цей тест був розроблений Домбровською І.С, на основі А.К. Макаров, Л.І. Божович.[25] Божович поділяє мотиви на дві групи а Макаров

розбиває кожну з цих груп ще на три. Перша група це пізнавальні мотиви а друга це соціальні мотиви.

В свою чергу пізнавальні мотиви діляться на такі підгрупи:

- широкі пізнавальні мотиви що мають під собою мотивацію студентів на оволодіння новими знаннями;
- навчально пізнавальні мотиві які показують мотивацію на засвоєнні способів здобуття знань;
- мотиви самоосвіти які показують спрямованість студентів на самостійне вдосконалення методів здобуття знань.

Друга велика група мотивів це соціальні мотиви і вони розділяються на такі підгрупи:

- широкі соціальні мотиви: мотивація яка показує прагнення здобувати знання щоб бути корисним батьківщині чи суспільству та бажання виконати свій обов'язок вчитися та в почутті відповідальності. Ця під шкала указує на мотиви усвідомлення студентом своєї соціальної необхідності а також до цієї підгрупи можна віднести бажання добре підготуватись до обраної професії;
- вузькі соціальні мотиви, ще їх називають позиційні мотиви, полягають у прагненні зайняти певну позицію чи місце у відносинах з оточуючими, а саме отримати від них схвалення, заслужити у них авторитет і т. д.;
- мотиви соціального співробітництва, які полягають у тому, що студент не тільки хоче спілкуватися та взаємодії з іншими людьми а також прагне аналізувати ти та усвідомлювати різні способи співробітництва та взаємовідносин з іншими людьми та постійно вдосконалювати ці навички. Цей мотив є одним з важливіших при самовихованні та самовдосконалення особистості[26].

Цей тест дозволяє оцінити як власну мотивації студента яка полягає у самовдосконаленні, так і дізнатися вплив на нього оточуючих та силу цього впливу на вибір професії с чого можна побачити бажання навчатися[27].

2.2.4 Мотиви вибору професії

Методика розроблена Р. В. Овчаровой.[28] Цей тест дозволяє визначити провідний тип мотивації у студента під час вибору професії. Мотивація вибору професії поділяється на такі типи: внутрішні та зовнішні. Внутрішні поділяються на індивідуальну значимі та соціально значимі, а зовнішні на позитивні та негативні мотиви.

Внутрішні мотиви вибору професії це задоволення від роботи, її суспільна значимість, її особиста значимість, задоволення від творчої роботи, а також можливість спілкування та взаємодії з іншими людьми. Внутрішня мотивація дуже важлива адже виникає з потреб людини і при її наявності людина працює самостійно та отримує задоволення від виконання роботи, на неї не треба використовувати додаткову стимуляцію.

Зовнішня мотивація у свій час це заробіток, прагнення престижу, страх перед невдачею чи страх осуду та все інше. Зовнішні мотиви можна поділити на позитивні та негативні. Позитивні мотиви це матеріальне стимулювання, можливість кар'єрного росту, схвалення від колег та престиж. Це стимули заради досягнення/підвищення яких людина готова докласти зусиль[29].

До негативних мотивів у свою чергу ставлять покарання, критику, засудження та інші санкції. Дослідження показують що переважання внутрішніх мотивів ефективніше за переважання зовнішніх для продуктивності та задоволення від праці. Проте наявність тільки високих позитивних зовнішніх мотивів не дає високого задоволення.

2.2.5 Задоволеність учбовою діяльністю

І останній тест: задоволеності навчальної діяльності від Л. В. Міщенко.[30] Основна шкала загальна задоволеність навчальну діяльність діяльністю поділяються на 6:

- задоволення змістом навчального процесу;
- задоволення виховним процесом;
- задоволення обраною професією;
- задоволення взаємодії з однокурсниками;
- шкала задоволення взаємодії з викладачами та адміністрацією;
- шкала задоволеності побутом бюджетом дозвілля та здоров'ям.

Можна побачити що деякі пункти у тестах перетинаються, це допомагає більш точному визначенню цих параметрів. Одним з найголовніших та найчастішим пунктом що зустрічається пов’язане з задоволеність у та правильністю обраної професії та задоволенням від навчання чи його ефективністю, що в першу чергу впливає на мотивацію студента.

2.2.6 Оцінка екстравертності

Для оцінки екстравертності був використаний тест Юнгу[31]. Результат цього тесту буде встановлення інтровертного, екстравертного чи амбівертного типу особистості людини. Це показує наскільки людині приемно/легко працювати у команді, однак це не як не зв’язано з хистом людини до командної роботи.

2.2.7 Особливості людини як командного гравця

Даний пункт визначається за допомогою 3 простих тестів. Вони показують вміння людини працювати у команді, та його конфліктність. Треба проводи усі ці 3 тесту разом задля більш коректного результату.

2.3 Методіка Белбіна для створення та покращення команд

Головна теза методу Белбіна це: недосконалі люди можуть створити досконалу команду.

Ціль цього тестування виявити сильні сторони учасників команди. Людина повинна не намагатися стати універсальна а покращувати свої сильні сторони, тоді зібрав у команду необхідних людей які компенсують недоліки один одного з'являється дуже сильна команда, яка може вирішити всі задачі та має мало труднощів. Звичайно таку команду можливо створити не завжди, адже у фірмах може не вистачати необхідних людей для формування ідеальних команд, а в університеті ми обмежені тими людьми які є в нашій групі. Через це все завдяки цьому методу можна також дізнатися слабкі сторони команди і прийняти якісь міри задля нівелювання цих недоліків (наприклад використовувати ті чи інший методики, додаткові програми чи залучати інших людей за для вирішення критичних питань).

Далі я розповім про методику Белбіна: як проходить тестування, які ролі є у командах і проведу експеримент над не ідеальною командою яка вже працювала разом і порівняю результат їх праці до введення моєї методи і після.

Правила проходження тесту: тест складається з 7 окремих блоків з 8 питань чи тверджень, з якими ви можете погодитись чи не погодитись. На кожен блок Ви маєте 10 балів. Надавати бали можна не більше, ніж 3-м або 4-м твердженням у блоці. Якщо Ви погоджуєтесь з будь-яким твердженням на всі 100%, Ви можете віддати йому всі 10 балів. При цьому одному реченню можна присвоїти мінімум 2 бали. Перевірте, щоб сума всіх балів по кожному блоку не перевищувала 10[32]. Питання які ставляться до опитуємих представлені на таблицях 2.1 – 2.7

Таблиця 2.1 - Блок 1, що я можу запропонувати команді?

10	*Я думаю, що в змозі швидко сприймати та використовувати нові можливості. Я легко кооперуюсь з людьми різних типів*
11	*Мій головний актив – продукувати нові ідеї.*
12	*Я здатен залучати людей, які, на мою думку, можуть зробити великий вклад в досягнення командних цілей.*
13	*Моя особиста здібність – ефективно доводити справу до кінця.*
14	*Я не уявляю собі тимчасового зниження своєї популярності, навіть якщо це приведе до збільшення прибутків.*
15	*Зазвичай я відчуваю, що реалістичне і що дієздатне.*
16	*Я здатен запропонувати вагомі аргументи на користь іншої лінії дій, не провокуючи при цьому упередженого ставлення.*
17	*Я думаю, що в змозі швидко сприймати та використовувати нові можливості. Я легко кооперуюсь з людьми різних типів*

Таблиця 2.2 - Блок 2, що характеризує мене як члена команди?

20	*Я почуваю себе некомфортно на зборах та нарадах, навіть якщо вони чітко структуровані і продумано організовані.*
21	*Я схильний покладатися на людей, які добре вміють аргументувати свою точку зору ще до того, коли вона була всебічно обговорена.*
22	*Коли група дискутує з приводу нових ідей, я можу занадто багато говорити. Моє особисте ставлення заважає мені підтримувати колег з ентузіазмом.*

Продовження таблиці 2.2

23	*Коли потрібно зробити будь-яку справу, деякі люди вважають, що я дію агресивно і авторитарно.*
24	*Я вагаюсь брати на себе лідерську роль, можливо тому, що занадто чуттєвий до почуттів та настроїв групи.*
25	*У мене є схильність настільки захоплюватися власними справами, що я забуваю про те, що відбувається довкола.*
26	*Мої колеги вважають, що я надмірно переймаюсь незначними деталями та боюсь ризику, що справа буде зроблена марно.*
27	*Я почуваю себе некомфортно на зборах та нарадах, навіть якщо вони чітко структуровані і продумано організовані.*

Таблиця 2.3 - Блок 3, коли я працюю з іншими над проектом

30	*Я можу добре впливати на інших людей, при цьому не здійснюючи сильного тиску.*
31	*Моє „шосте чуття" підказує і застерігає мене від помилок та інцидентів, які інколи трапляються через неуважність.*
32	*В ім'я досягнення головних цілей, я готовий прискорювати події, не витрачаючи часу на обговорення.*
33	*Від мене завжди можна чекати чогось оригінального.*
34	*Я завжди готовий підтримати хорошу пропозицію, яка принесе вигоду всім*
35	*. Я постійно відслідковую останні ідеї та новітні досягнення.*

Продовження таблиці 2.3

36	*Я думаю, що мої здібності до суджень та оцінок можуть бути значним внеском у прийняття правильних рішень*
37	*На мене завжди можна покластися на завершальному етапі роботи.*

Таблиця 2.4 - Блок 4, моє відношення та інтерес до групової роботи

40	Я щиро бажаю знати своїх колег краще.
41	Я не боюся ні оскаржувати точку зору іншої людини, ні залишитися в меншості
42	Зазвичай, я можу довести неконкурентоспроможність невдалої пропозиції.
43	Я вважаю, що я здатен добре виконати будь-яку функцію заради виконання спільного блага.
44	Часто я уникаю очевидних рішень і приходжу замість цього до несподіваного вирішення проблеми.
45	Все, що я роблю, я прагну доводити до досконалості.
46	Я готовий використовувати контакти та зв’язки поза групою.
47	Я не відчуваю труднощів у прийнятті рішень, хоча я завжди відкритий різним точкам зору.

Таблиця 2.5 - Блок 5, я одержую задоволення від роботи, тому що…

50	*Мені подобається аналізувати ситуації та оцінювати можливі напрями діяльності.*
51	*Мені цікаво знаходити практичні шляхи вирішення проблеми.*

Продовження таблиці 2.5

52	*Мені приємно відчувати, що я допомагаю створенню хороших відносин на роботі.*
53	*Часто маю сильний вплив на рішення, що приймається.*
54	*Я маю відкриті, привітні відносини з людьми, які можуть запропонувати щось нове.*
55	*Я можу переконувати людей в необхідності певної лінії дій.*
56	*Я почуваю себе добре вдома, коли я можу приділити максимум уваги завданню.*
57	*Я люблю працювати з будь-чим, що стимулює мою уяву.*

Таблиця 2.6 - Блок 6, коли завдання складне та незнайоме

60	*Я відкладаю справу на певний час та розмірковую над проблемою.*
61	*Я готовий співпрацювати з людьми, які більш позитивно і з більшим ентузіазмом відносяться до проблеми.*
62	*Я намагаюсь зробити завдання простішим, підшукуючи в групі людей, які можуть взяти на себе вирішення частини проблеми.*
63	*Моє вроджене почуття часу дозволяє мені витримувати терміни виконання завдання.*
64	*Я вважаю, мені вдається зберегти ясність думки і спокій.*
65	*Навіть під тиском зовнішніх обставин, я не відступлю від мети.*
66	*Якщо я відчуваю, що група не прогресує, я готовий взяти лідерські обов'язки на себе.*

Продовження таблиці 2.6

67	*Я б розпочав дискусію з метою стимулювання появи нових думок, які сприяли б вирішенню проблеми.*

Таблиця 2.7 - Блок 7, проблеми, що виникають при роботі в групах

70	*Я схильний висловлювати свою нетерпимість по відношенню до людей, які стоять на шляху розвитку прогресу (заважають).*
71	*Інші можуть критикувати мене за те, що я занадто аналітичний і не підключаю інтуїцію.*
72	*Моє бажання впевнитися в тому, що робота виконується з високим якісним рівнем, може інколи приводити до затримки.*
73	*Мені швидко все набридає, і я покладаюсь на те, що хтось із групи стимулює мій інтерес.*
74	*Мені важко приступити до вирішення завдання, не маючи чіткої цілі.*
75	*Інколи мені важко пояснити та описати проблему в комплексі.*
76	*Я знаю, що вимагаю від інших того, що сам не можу виконати.*
77	*Я вагаюсь висловлювати власну думку, коли я знаходжусь в очевидній опозиції до більшості.*

Обробка результатів: задля підрахунку результату треба перенести свої бали з кожного блоку в Таблицю 2.8.

Таблиця 2.8 – Підрахунок результатів тестування

Блок №	РЕАЛІЗАТОР	КООРДИНАТОР	ТВОРЕЦЬ	ГЕНЕРАТОР ІДЕЙ	ДОСЛІДНИК	ЕКСПЕРТ	ДИПЛОМАТ	ФАХІВЦЬ
1	16	13	15	12	10	17	11	14
2	20	21	24	26	22	23	25	27
3	37	30	32	33	35	36	34	31
4	43	47	41	44	46	42	40	45
5	51	55	53	57	54	50	52	56
6	65	62	66	60	67	64	61	63
7	74	76	70	75	73	71	77	72
Підсумок:								

В результаті можна побачити яка(які) характеристика(ки) домінують у людини. Опис ролей буде наведено нижче.

Реалізаторам властиві практичний здоровий глузд і хороше відчуття самоконтролю і дисципліни. Вони люблять важку роботу і подолання проблем в системному режимі. Більшою мірою Реалізатори є типовими особами, чия вірність і інтерес співпадають з цінностями компанії. Вони менш сконцентровані на переслідуванні власних інтересів. Проте, їм може не вистачати спонтанності і вони можуть проявляти жорсткість і непохитність.

Реалізатори дуже корисні для компанії завдяки своїй надійності і старанності. Вони добиваються успіху, тому що дуже працездатні і можуть чітко визначити те, що здійснимо і має відношення до справи. Говорять, що багато

виконавців роблять тільки ту роботу, яку хочуть робити і нехтують завданнями, які знаходять неприємними. Реалізатори, навпаки, робитимуть те, що необхідно справі. Хороші Реалізатори часто просуваються до високих посадових позицій в управлінні завдяки своїм хорошим організаторським здібностям і компетентності в рішенні всіх важливих питань.

Відмінною рисою Координаторів є здатність примушувати інших працювати над розподіленими цілями. Зрілий, досвідчений і упевнений, Координатор охоче роздає доручення. У міжособових відносинах вони швидко розкривають індивідуальні схильності і таланти і мудро їх використовують для досягнення мети команди. Вони не обов'язково найрозумніші члени команди, це люди з широким світоглядом і досвідом, що користуються загальною повагою команди.

Координатори добре себе проявляють, знаходячись на чолі команди людей з різними навиками і характерами. Вони краще працюють спільно з колегами рівними по рангу або позиції, чим із співробітниками нижчих рівнів. Їх девізом може бути «консультація з контролем». Вони вірять, що проблему можна вирішити мирним шляхом. У деяких компаніях Координатори можуть вступати в конфлікти через різницю в поглядах з Творцями.

Творці це люди з високим рівнем мотивації, невичерпною енергією і великим прагненням до творчих та професійних звершень. Зазвичай, це яскраво виражені екстраверти, що володіють сильною напористістю. Їм подобається кидати виклик іншим, їх мета – перемога. Їм подобається вести інших і підштовхувати до дій. Якщо виникають перепони, вони швидко знаходять обхідні шляхи. Свавільні і уперті, упевнені і напористі, вони мають схильність емоційно відповідати на будь-яку форму розчарування або краху планів. Цілеспрямовані, такі, що люблять посперечатися. Але їм часто не вистачає простого людського розуміння. Їх роль сама конкурентна в команді.

Творці, зазвичай, стають хорошими керівниками, завдяки тому, що уміють генерувати дії і успішно працювати під тиском. Вони уміють легко

надихати команду, і дуже корисні в групах з різними поглядами, оскільки здатні приборкати пристрасті. Творці здатні підніматися над проблемами будь-якого роду, продовжуючи лідирувати, не зважаючи на них. Вони можуть легко провести необхідні зміни і не відмовляються від нестандартних рішень. Відповідаючи назві, вони намагаються нав'язувати групі деякі зразки або форми поведінки і діяльності. Вони є найефективнішими членами команди, здатними гарантувати позитивні дії.

Генератори ідей є новаторами і винахідниками, можуть бути дуже креативними. Вони сіють зерно і ідеї, з яких проростають більшість розробок і проектів. Зазвичай вони вважають за краще працювати самостійно, відокремившись від інших членів команди, використовуючи свою уяву і часто слідуючи нетрадиційним шляхом. Мають схильність бути інтровертами і сильно реагують як на критику, так і на похвалу. Часто їх ідеї мають радикальний характер, і їм не вистачає практичних зусиль. Вони незалежні, розумні і оригінальні, але можуть бути слабкими в спілкуванні з людьми іншого рівня або напряму.

Основна функція Генераторів ідей – створення нових пропозицій і вирішення складних комплексних проблем. Вони дуже необхідні на початкових стадіях проектів або коли проект знаходиться під загрозою зриву. Вони зазвичай є засновниками компаній або організаторами нових виробництв. Проте, велика кількість Генераторів ідей в одній компанії може привести до контрпродуктивності, оскільки вони мають тенденцію проводити час, укріплюючи свої власні ідеї і вступаючи один з одним в конфлікт.

Дослідники - часто ентузіасти і яскраві екстраверти. Вони уміють спілкуватися з людьми в компанії і за її межами. Вони народжені для ведення переговорів, дослідження нових можливостей і налагодження контактів. Хоча і не будучи генераторами оригінальних ідей, вони дуже легко підхоплюють ідеї інших і розвивають їх. Вони дуже легко розпізнають, що є в наявності і що ще можна зробити. Їх зазвичай дуже тепло приймають в команді завдяки їх

відкритій натурі. Вони завжди відкриті і допитливі, готові знайти можливості у всьому новому. Але, якщо вони не стимулюються іншими, їх ентузіазм швидко знижується.

Дослідники дуже добре реагують і відповідають на нові ідеї і розробки, можуть знайти ресурси і поза групою. Вони самі відповідні люди для установки зовнішніх контактів і проведення подальших переговорів. Вони уміють самостійно думати, отримуючи інформацію від інших.

Експерти дуже серйозні і передбачливі люди з природженим імунітетом проти надмірного ентузіазму. Повільні в ухваленні рішення, віддають перевагу добре все обдумати. Вони здатні критично мислити. Вони уміють бути проникливими в думках, беручи до уваги всі чинники. Експерти рідко помиляються.

Експерти найбільш підходять для аналізу проблем і оцінки ідей і пропозицій. Вони добре уміють зважувати всі «за і проти» запропонованих варіантів. В порівнянні з іншими, Експерти здаються черствими, занудними і надмірно критичними. Деякі дивуються, як їм вдається стати керівниками. Проте багато Експертів займають стратегічні пости і досягають успіху на посадах вищого рангу. Дуже рідко успіх або зрив справи залежить від ухвалення квапливих рішень. Це ідеальна «сфера» для Експертів, людей, які рідко помиляються і, врешті-решт, виграють.

Дипломати це люди, що користуються найбільшою підтримкою команди. Вони дуже ввічливі і товариські. Вони уміють бути гнучкими і адаптуватися до будь-якої ситуації і різних людей. Дипломати дуже дипломатичні і сприйнятливі. Вони уміють слухати інших і співпереживати, дуже популярні в команді. У роботі вони покладаються на чутливість, але можуть зіткнутися зі складністю при ухваленні рішень в термінових і невідкладних ситуаціях.

Роль Дипломатів полягає в запобіганні міжособових проблем, що з'являються в команді, і тому це дозволяє ефективно працювати всім її членам. Уникаючи проблем, вони йтимуть довгою дорогою, ради того, щоб обійти їх

стороною. Вони не часто стають керівниками, тим більше, якщо їх безпосередній начальник підкоряється Творцеві. Це створює клімат, в якому дипломатія і сприйнятливість людей цього типу є справжньою знахідкою для команди, особливо при управлінському стилі, де конфлікти можуть виникати і повинні штучно присікатися. Такі люди як керівники не представляють загрозу ні для кого і тому завжди бажані для підлеглих. Дипломати служать свого роду «мастилом» для команди, а люди в такій обстановці співпрацюють краще.

Виконавці володіють величезною здатністю доводити справу до завершення і звертати увагу на деталі. Вони ніколи не починають те, що не можуть довести до кінця. Вони мотивуються внутрішніми переживаннями, хоча часто зовні виглядають спокійними і незворушними. Представники цього типу часто є інтровертами. Їм зазвичай не потрібне стимулювання із зовні. Вони не терплять випадковостей. Не схильні до делегування, вважають за краще виконувати завдання самостійно.

Виконавці є незамінними в ситуаціях, коли завдання вимагають сильної сконцентрованості і високого рівня акуратності. Вони несуть відчуття терміновості і невідкладності в команду і добре проводять різні мітинги. Добре справляються з управлінням, завдяки своєму прагненню до вищих стандартів, своєї акуратності, точності, уваги до деталей і уміння завершувати розпочату справу.

Фахівці це особи, які пишаються придбаними технічними навиками і уміннями у вузькій сфері. Їх пріоритетами є надання професійних послуг, сприяння і просування в своїй сфері діяльності. Проявляючи професіоналізм у своєму предметі, вони рідко цікавляться справами інших. Можливо, вони стануть експертами, слідуючи своїм стандартам і працюючи над вузьким колом специфічних проблем. Взагалі, небагато людей, беззастережно відданих своїй справі і прагнучих стати першокласними фахівцями.

Фахівці грають свою специфічну роль в команді завдяки своїм рідкісним навикам, на яких і базується сервіс або виробництво компанії. Будучи

керівниками, вони користуються пошаною, оскільки знають набагато більше про свій предмет, чим хто-небудь ще і зазвичай вимушені ухвалювати рішення, спираючись на свій глибокий досвід.

Перевірка вмотивованості групи проводиться за допомогою тесту І. Д. Ладанова. Тестування було проведено до введення системи і після.

Інструкція до тесту: тест містить 25 факторів (позитивних та негативних), що дозволяють оцінити ступінь сформованості групової мотивації. Треба уважно оцінити ці фактори та вибрати відповідний бал, що характеризує стан мотивації у групі, членом якої ви є. Треба, щоб це завдання виконали всі члени вашої групи та було отримано середній результат оцінок.

Таблиця 2.9 – Тест І. Д. Ладанова

№	Переважаючі фактори	Оцінка в балах	Переважаючі фактори
1	Високий рівень згуртованості групи	7 6 5 4 3 2 1	Низький рівень згуртованості групи
2	Висока активність членів групи	7 6 5 4 3 2 1	Низька активність членів групи
3	Нормальні міжособистісні стосунки у групі	7 6 5 4 3 2 1	Погані міжособистісні стосунки у групі
4	Відсутність конфліктів у групі	7 6 5 4 3 2 1	Наявність конфліктів у групі
5	Високий рівень групової сумісності	7 6 5 4 3 2 1	Низький рівень групової сумісності
6	Особистісне осмислення організаційних цілей та їх прийняття	7 6 5 4 3 2 1	Неприйняття працівниками організаційних цілей
7	Визнання авторитету керівника	7 6 5 4 3 2 1	Члени групи не приймаю авторитету керівника
8	Повага до компетентності керівника	7 6 5 4 3 2 1	Члени групи не віддають належної компетентності керівника

Продовження таблиці 2.9

9	Визнання лідерських якостей керівника	7 6 5 4 3 2 1	Члени групи не вважають свого керівника лідером
10	Наявність довірчих відносин членів групи з керівником	7 6 5 4 3 2 1	Відсутність довірчих відносин членів групи з керівником
11	Участь членів групи у процесі прийняття рішення	7 6 5 4 3 2 1	Неприйняття членами групи участі в обговоренні та прийнятті рішення
12	Є умови для вираження творчого потенціалу членів групи	7 6 5 4 3 2 1	Немає умов для вираження творчого потенціалу членів групи
13	Прагнення прийняти відповідальність членами групи за роботу	7 6 5 4 3 2 1	Відсутність прагнення у членів групи приймати відповідальність за виконувану роботу
14	Наявність хорошого психологічного клімату групи	7 6 5 4 3 2 1	Наявність поганого психологічного клімату групи
15	Високий рівень контролю за діями кожного члена групи	7 6 5 4 3 2 1	Низький рівень контролю за діями кожного члена групи
16	Наявність активної життєвої позиції усередині групи	7 6 5 4 3 2 1	Відсутність активної життєвої позиції усередині групи
17	Прагнення самореалізації у членів групи	7 6 5 4 3 2 1	Відсутність прагнення самореалізації в членів групи
18	Високий ступінь узгодженості дій у членів групи	7 6 5 4 3 2 1	Слабкий ступінь узгодженості дій у членів групи
19	Сформованість загальногрупових цінностей	7 6 5 4 3 2 1	Відсутність загальногрупових цінностей
20	Відсутність стресів усередині групи	7 6 5 4 3 2 1	Наявність стресів

Закінчення таблиці 2.9

21	Бажання працювати у групі	7 6 5 4 3 2 1	Прагнення членів групи працювати індивідуально
22	Позитивне ставлення керівника до своїх підлеглих	7 6 5 4 3 2 1	Негативне ставлення керівника до членів робочої групи
23	Позитивне ставлення членів групи до свого керівника	7 6 5 4 3 2 1	Негативне ставлення членів групи до свого керівника
24	Прийняття моральних норм поведінки усередині групи	7 6 5 4 3 2 1	Відсутність моральних норм поведінки усередині групи
25	Вміння виявляти самостійність у вирішенні поставлених завдань членами групи	7 6 5 4 3 2 1	Відсутність прагнення самостійно вирішувати поставлені завдання

Висновки до розділу 2

Підготовка до експерименту являє собою пошук та поєднання необхідних методологій для виявлення мотивації студента та пошуку параметрів за якими можна розбити студентів на групи. Буде проведено два експерименти, перший повинен показати як справляється з завданням система виявлення рівня мотивації та рівня командних навичок студентів, а другий експеримент повинен показати чи працює створена система на прикладі команди студентів.

Після проведення експерименту можна буде казати про те які тести є реально діючими а які треба вилучити чи замінити. Методика тестування Белбіна була досліджена та повинна дати гарний результат. Однак до неї не було створено ПЗ яке полегшили використання цієї методики. Також був розроблена комплексна крапка які додаток дає студентам при оцінки їх команди за допомогою методології Белбіну.

3 Проектування й розробка інструментального забезпечення для дослідження ефективності стохастичних алгоритмів

3.1. Вибір інструментальних засобів розробки

Для розробки ПЗ необхідно:

- вибрати мову програмування;
- вибрати середовище розробки із вбудованим налагоджувачем;
- обрати систему контроля версій.

3.1.1. Вибір мови програмування

Оскільки для розробки ПЗ необхідно надати тести, то треба розглянути мову, яка володіє зручним інструментом для графічного виведення інформації. Для розробки ПЗ потрібна мова з підтримкою об'єктно-орієнтованої методології програмування, що дозволяє реалізовувати більш гнучкі програмні рішення. Ще одним критерієм вибору мови є його поширеність, яка забезпечує широкий набір ресурсів для пошуку додаткової інформації у разі виникнення помилок або пошуку найкращих програмних та архітектурних рішень. Як об'єкти дослідження були обрані мови програмування, що задовольняють переліченим вимогам, а саме: C++ [33], C [34], Java [35], C # [36] та Python [37].

Мову програмування C++ розроблено на початку 80-х років співробітником Bell Laboratories Б'єрном Страуструп [33]. Мова C++ іноді називають покращеною мовою C, тому що вона забезпечує перевантаження функцій, контроль типів та ряд інших можливостей. Але головна відмінність C++ у тому, що він додає до об'єктної орієнтованості. C++ був стандартизований 1998 року Міжнародною організацією стандартизації під номером 1488:1998 — Мова програмування C++.

Стандартна бібліотека мови C++ містить поширені контейнери та алгоритми, засоби введення-виведення, регулярні висловлювання, багатопоточність та інші можливості. У C++ поєднуються властивості як високорівневих, і низькорівневих мов.

Синтаксис C++ успадкований від мови C. Серед нових можливостей можна виділити простори імен, класи (поліморфізм, інкапсуляція, успадкування, віртуальні функції, функції-члени, абстрактні класи та конструктори тощо), навантаження операторів, шаблони, обробку винятків та багато іншого.

C++ широко використовується при розробці різного програмного забезпечення, а також є однією з найвідоміших та затребуваних мов програмування.

Мова програмування C була розроблена Кеном Томпсоном та Деннісом Рітчі на початку 70-х років [33]. Початкова реалізація була для створення операційної системи UNIX. Потім, мова C була перенесена на багато інших платформ і існує незалежно від них. Програми, написані З, близькі за швидкістю виконання до мови асемблера.

Мова C широко застосовується для створення системного програмного забезпечення, і є потужним засобом для вирішення важких завдань та створення складних програмних рішень. Програми, написані мовою C, переносяться будь-яку платформу або з мінімальними змінами, або зовсім без них.

Мова C вклала істотний внесок у розвиток програмного забезпечення.

Синтаксис цієї мови було закладено основою таких мов програмування як C++, PHP, Java, C# та інших.

Мова Java є об'єктно-орієнтованою мовою програмування. Java почав розроблятися компанією Sun Microsystem в 1991, а офіційний випуск відбувся 23 травня 1995 [34]. Спочатку ця мова називалася Oak і призначалася для використання в побутовій електроніці, але надалі мову перейменували на Java і почали використовувати для написання додатків та серверних програмних рішень.

Java-програми спочатку переводяться в байт-код, цей байт-код необхідний для запуску на віртуальній Java машині (JVM - Java Virtual Machine). JVM обробляє байт-код та передає інструкції обладнання. Тим самим пропадає необхідність у складанні програми під кожну платформу окремо, головне, щоб

існувала JVM для відповідної платформи. Також можна відзначити гнучку систему безпеки: будь-які операції, що перевищують встановлені повноваження програми (такі як несанкціонований доступ до даних або з'єднання з іншим комп'ютером), викликають її переривання. Однак, головним недоліком концепції віртуальної машини вважають зниження продуктивності.

C# - об'єктно-орієнтована мова програмування. C# був розроблений в 2000. Спочатку C# був частиною MS .Net Framework, проте пізніше мова стандартизувався як ISO/IEC 23270 [35].

C# відноситься до сім'ї мов із C-подібним синтаксисом, він близький до C++ та Java. Для зниження можливості внесення помилок у код, C# введений механізм складання сміття, який звільняє програміста від необхідності ручного очищення пам'яті. У C# відсутні глобальні змінні, множинне успадкування (замість нього допускається множинне успадкування інтерфейсів). Натомість, C# існує підтримка поліморфізму, є статична типізація, підтримується навантаження операторів, узагальнені типи, ітератори, анонімні функції, винятки та інші можливості.

У C# заборонено пряме маніпулювання пам'яттю, але платформа Microsoft .NET надає натомість багатофункціональний збирач сміття та велику кількість типів даних. Перетворення типів у мові C# відрізняються від перетворень у мові C++, це виражається, наприклад, у цьому, що це перетворення мають бути лише явними. До того ж, усі перетворення типів можуть бути лише безпечними (неявні перетворення чи покажчики у вигляді цілих змінних не допускаються).

Python – мова програмування високого рівня, що широко застосовується як інтерпретована об'єктно-орієнтована мова [36].

Python підтримує динамічну типізацію, автоматичне керування пам'яттю та структурами даних високого рівня: словниками (хеш-таблицями), списками, кортежами. Серед основних можливостей Python можна виділити класи, успадкування, поліморфізм, інкапсуляція, модулі, навантаження операторів,

статичні методи та класи, обробку винятків та багатопотокові обчислення. Синтаксис у мови Python простий і виразний. Python підтримує та одночасно поєднує три підходи до написання коду: процедурний, об'єктно-орієнтований та функціональний.

Python має багатофункціональну стандартну бібліотеку. Велика кількість модулів, що працюють з різними операційними системами, робить мову кросплатформною. Крім цього, Python вміє підтримувати регулярні вирази, різні кодування тексту, містить мультимедійні засоби, працює з архівами, має функціонал для написання юніт-тестів та ін.

Для підсумку результатів дослідження була створена таблиця 3.1

Порівняльний аналіз мов програмування C++, C, Java, C# та Python

Таблиця 3.1 – порівняння мов програмування

Характеристики	Мова програмування				
	C++	C	Java	C#	Python
Підтримка об’єктно – орієнтованого підходу	+	-	+	+	+
Можливість компілювати в машинний код	+	+	-	-	+
Обробка виключень	+	-	+	+	+
Статична типізация	+	+	+	-	-
Кросплатформеність	+	+	+	+	+
Досвід використування	+	+	-	+	-

3.1.2 Опис використаних бібліотечних засобів

System.Windows.Forms[37] це простір імен, який містить класи для створення програм Windows, які дозволяють найбільш ефективно використовувати розширені можливості інтерфейсу користувача, доступні в операційній системі Microsoft Windows.

Ця бібліотека має є дуже багато різних класів структур інтерфейсів для графічного виводу інформації. За допомогою неї був створений графічний інтерфейс програми. Ця бібліотека дуже зручна коли треба швидко створити простий інтерфейс для додатку але вона мала підходить для більш складних програм та інтерфейсів.

System.IO [38] це простір імен який містить типи, що дозволяють читати і записувати файли та потоки даних, а також типи для базової підтримки файлів і папок.

Ця бібліотека дозволяє є дуже легко та швидко читати та записувати файли і потоки даних. Вона знадобиться для використання наступної бібліотеки.

Newtonsoft.Json [39] бібліотека яка застосовується для того щоб читати файли формату json у у програмі написаній на c#.

JSON (англ. JavaScript Object Notation) - текстовий формат обміну даними, що базується на JavaScript. Але при цьому формат незалежний від JS може використовуватися в будь-якій мові програмування.

Бібліотека дозволяє серіалізувати та десеріалізувати любий об'єкт. Її функції:

- створення JSON;
- аналіз JSON;
- зміна JSON;
- надсилання запитів до JSON.

Також вона є високопродуктивною, на 50% швидше ніж DataContractJsonSerializer. Проста у використанні та підтримує XML. За її допомогою можна конвертувати JSON в XML. Ця бібліотека поширюється повністю безкоштовно для комерційного використання.

Microsoft.Office.Interop.Word[40] бібліотека яка дає інструменти для роботи з Microsoft Word. Функціонал бібліотеки дозволяє з'єднуватися з файлами Microsoft Word та проводити пошук та редагування тексту в них.

3.1.3. Вибір середовища розробки

C# є однією з найпопулярніших мов програмування. Через це для нього існує велика кількість інтегрованих серед розробки. Вони усі мають великий але різний функціонал. Для порівняння були обрані такі інтегровані середи розробки MS Visual Studio [41], JetBrains Clion [42] и Eclipse SDK [44]. Ці середи є найбільш популярні та розповсюдженні.

Microsoft Visual Studio є інтегрованим середовищем, призначеним для розробки ПЗ, що володіє рядом додаткових інструментальних засобів [41].

Продукти MS Visual Studio дозволяють розробляти консольні програми, програми з графічним інтерфейсом, веб-сайти, веб-програми та веб-служби для платформ.

MS Visual Studio має інструменти для редагування вихідного коду та для його рефакторингу. Відлагодник який присутній у MS має декілька рівнів роботи, а саме рівень налагодження вихідного коду та рівень налагодження машинного коду. У наявності присутні форми графічного інтерфейсу та інструментальні средства для роботи з ними, а також зручний редактор для створення додатків з интерфейсами. Редактор класів та веб редактор теж присутні.

У цій середі зручно реалізована підтримка можливості створення та підключення плагінів для розширення функціоналу. MS Visual Studio підтримує

різні системи контролю версій та має багато інструментів для редагування та проектування коду на предметно орієнтованих мовах програмування

JetBrains CLion - розумне інтегроване середовище розробки програмного забезпечення, призначене для розробки додатків мовами C і C++ [42]. JetBrains Clion дозволяє розробляти програмне забезпечення на платформах Windows, OS X та Linux.

JetBrains CLion є багатофункціональною середою розробки, вона включає у себе шаблони готового коду відлагодник інтерфейс для роботи з системи контролю версій(Subversion, Git, GitHub, Mercurial, CVS, Perforce и TFS). CLion виконує аналіз коду та може автодоповнювати і автоформатувати код, виконувати його аналіз виводячи результат на екран, де показані місця з потенціальними помилками та пропонує способи їх виправлення. Також підтримує систему зборки кроссплатформених проектів. JetBrains CLion має багато плагинів для розширення функціональності та різні рефакторинги коду.

Eclipse SDK – вільне інтегроване середовище розробки програмного забезпечення з відкритим вихідним кодом [43]. Eclipse SDK має велику розгалужену систему плагінів (доповнень), яка забезпечує середовище роботи з такими мовами програмування, як C#, C++, PHP, Perl, Python, Ruby, Ada або COBOL.

За допомогою Eclipse можна створювати кросплатформене програмне забезпечення для Microsoft Windows, Mac OS X, Linux дистрибутивів та Solaris. Дана середа постачається із Eclipse Java development tools (JDT), що має у собі компілятор Java. Разом з усіма інструментами які використовує програма утворюється система Eclipse Rich Client Platform. Eclipse SDK надає програму Java IDE з усім необхідним набором інструментів для розробки додатків високої якості.

В результаті вивчення розглянутих засобів масової інформації для програмування в C# було створено порівняльну таблицю 3.2.

Таблиця 3.2 порівняння середовищ розробки

Параметри	MS Visual Studio	JetBrain Clion	Exlipse SDK
Редактор інтерфейсу	+	-	+
Вбудований профільник	+	-	-
Вбудований налагуджувач	+	+	+
Підтримка C# 11	+	+	+
Наявність ліцензії	+	-	+
Опит використання	+	-	-

Для розробки ПО було обрано середовище розробки Microsoft Visual Studio. Середовище розробки Microsoft Visual Studio було обрано через те, що вона має низку необхідних для даної розробки функцій, таких як підтримка профілювання коду мовою програмування C# та наявністю систем для легшої взаємодії з іншими продуктами Microsoft.

3.1.4 Вибір системи управління версіями

Система управління версіями - це програмне забезпечення, для полегшення управління зміною інформації [44]. Призначення систем управління версіями полягає в зберіганні різних версій одного файлу, виправлення змін між версіями цього файлу та можливості повернутися до попередніх версій, якщо це необхідно.

Системи управління версіями часто використовуються програмістами з метою поетапного зберігання версій розробленого програмного забезпечення.

Як об'єкти дослідження, системи управління версіями CVS [45], Subversion [47], GIT [48] та Mercurial [49], як найбільш функціональні та загальні.

Це програмний продукт із родини системи контролю версій. CVS застосовує технологію клієнт-сервер, вони можуть з'єднуватись або по сеті, або бути на одній машині. Принцип роботи: усі версії програми та усі зміни зберігаються у репозиторію серверу. Коли клієнт приєднується до сервера він перевіряє відмінності своїй версії від версії сервера і, якщо є відмінності, вони завантажуються в локальний репозиторій. Якщо при завантаженні змін виникають конфлікти то користувачу треба їх вирішити, після чого усі зміни вносяться до проекту. Потім зміни завантажуються до репозиторію сервера. CVS це система яка може керувати паралельними версіями, тобто вона підтримує керування декількома проектами одночасно та можливість повернутися до попередніх версій цих проектів.

Основні переваги цієї системи керування версіями:

- можливість одночасної роботи над одним проектом декількох клієнтів;
- можливість керування усім проектом;
- наявність зручного графічного інтерфейсу.

Система CVS дуже поширена та предзавантажена на багатьох операційних системах GNU/Linux. У цій системі при завантаженні файлів з серверів надаються лише зміни а не весь файл.

Недоліки системи:

- якщо перемістити чи перейменувати файл або директорію у якій знаходиться проект то це призведе до втрати усіх змін зв'язаних з ними;
- вести паралельні ділки одного проекту досить складно;
- кожна зміна бінарного файлу потребує збереження всій версії файлу;
- передача відредагованих файлів на сервер від клієнта здійснюється передачею усього файлу.

Основним недоліком цієї системи є те що її активна розробка вже завершена. І хоча існують більш сучасні системи контролю версій які не мають недоліків CVS ця система все ще є однією з найрозповсюдженіших. Наприклад вона широко використовуються для невеликих проектів які не мають паралельних версій через те що такий проекти доводиться періодично об'єднувати.

Subversion є вільною системою керування версіями. Офіційний випуск був проведений компанією CollabNet у 2004 [47]. Subversion містить усі важливі функції CVS, але в ній зменшено ряд недоліків CVS (зокрема, перейменування чи переміщення файлів та каталогів, робота з двійковими файлами).

Ця система базується на технології клієнт-сервер і має схожий принцип роботи із CVS: спочатку клієнти завантажують зміни із репозиторія та з'єднують їх з проектом, під час чого конфлікти локальних та серверних змін вирішуються розробником.

Основні переваги системи:

- команди схожі на команди у CVS;
- наявність графічного інтерфейсу;
- зручна робота з консолі;

- можливість побачити всю історію змін файлів та каталогів після зміни їх імен;
- підтримуються у багатьох середах розробки;
- можливість копіювати репозиторій;
- два типи репозиторій: база даних чи звичайний набір файлів;
- зручний механізм створення гілок проекту.

Основні недоліки системи:

- повна копія сховища зберігається на локальному комп'ютері у прихованих файлах і займає достатньо великий об'єм пам'яті;
- проблеми під час обробки паралельної зміни імені файлу різними користувачами;
- складність повного видалення файлу та інформації по цьому файлу з репозиторію.

Subversion дуже поширена серед розробників. Використовуються під час розробки як відкритого програмного забезпечення так і закритих корпоративних проектів. Subversion зручна у використанні через те що має великий набір інструментів для керування версіями проекту.

CVS і SUBVERSION це системи управління що відносяться до централізованих систем контролю версій і є додатком типу клієнт-сервер. Особливістю цього є те що репозиторій проекту існує тільки в одному екземплярі який лежить на сервері. Доступ до цього репозиторія йде через спеціальний клієнтський додаток. На рисунку 3.1 представлена архітектура централізованої системи контролю версій.

Рисунок. 3.1 Архітектура централізованої системи контролю версій

Git - розподілена система управління версіями, метою створення якої було керування процесом розробки ядра Linux [48]. Git - це гнучка, розподілена система контролю версій, яка дає багато можливостей розробникам програмних продуктів, для яких важливе управління історією змін.

Git дозволяє обмінюватися збереженими даними з іншими користувачами у репозиторії. Існує центральний репозиторій, з ним з'єднуються декілька розробників, кожному розробнику дається локальна копія усі історії розробки. Вони можуть завантажувати оновлення з центрального репозиторій вносити правки і завантажувати їх у центральний репозиторій.

Git дає можливість вести роботу над проектом паралельно у декількох гілках адже забезпечує швидке розділення і злиття версії: вони можуть легко об'єднатися, видалитися, змінитися чи розростатися у нові гілки проекту.

Основні переваги системи управління версіями Git:

- система дуже надійна під час порівняння ревізії і перевірки коректності даних. Це забезпечує алгоритм хешифрування SHA 1;
- висока продуктивність;
- системи розгалуження та злиття гілок дуже гнучкі;
- усі розробники мають локальний репозиторій з повною інформацією щодо змін у проекті;
- наявність графічних оболонок;

- можливість створювати контрольні точки. В будь-який момент часу можна швидко відновити дані адже у цих точках повністю збережені дані щодо стану проекту;
- універсальний мережевий доступ за допомогою HTTP, FTP, RSYNC, SSH та інших протоколів;
- наявність гарної документації;
- можливість інтегрування з іншими системами контроля версій;
- система дозволяє налаштовувати її під себе та створювати користувацький інтерфейси адже є дуже гнучкою.

Основні недоліки системи керування версіями ГІТ:

- система орієнтована на Unix-подібні операційні системи;
- є мала вирогідність спадання хеш коду різних за змістом ревізій;
- під час зміни не зв'язаних один з одним файлів у великому проекті зміни зберігаються дуже довго адже зміни в проекті відстежується повністю;
- довга синхронізація з іншими розробниками при створенні репозиторія.

Git зручна, гнучка та потужна система контролю версій. Він підходить багатьом користувачам. Розробники постійно покращують його та прибирають недоліки не створюючи проблем користувачам.

Mercurial є багатоплатформною та розподіленою системою контролю версій. Mercurial була створена для спрощення та прискорення роботи з великими репозиторіями коду [49]. Створена Меттом Макалом паралельно із системою Git.

Mercurial написана на мові програмування пайтон а у місцях де треба було підвищити продуктивність на сі. ідентифікація ревізій відбувається за

допомогою алгоритма херширування SHA1, так само як і в Git. Також є можливість надання індивідуального номеру. Є можливість злиття та створення гілок проекту. Взаємодію між клієнтами забезпечують протоколи HTTP, HTTPS чи SSH.

Типовий процес роботи у Mercurial виглядає так:

- створення на локальному комп'ютері репозиторія;
- виконання додавання зміни чи видалення файлів у репозиторій;
- фіксація змін;
- повторення другого та третього року;
- синхронізація змін з іншими репозиторіями.

Відмінністю Mercurial є те що вся робота на відміну від інших систем контролю версій проходить у локальному репозиторій II лише за необхідністю тю робиться відправка результатів роботи у інші репозиторії.

Переваги даної системи управління версіями:

- швидка обробка даних;
- робота з декількома гілками проекту;
- можливість інтеграції з іншими системами контролю версій.

Основні недоліки M перекладається з недоліками Git, наприклад можливість що співпадають в шкоди різних ревізій і орієнтованість на роботу в консолі.

Інтерфейс, набір команд та можливість імпортування репозиторіїв з інших систем роблять перехід на Mercurial простим. Є достойним конкурентом Git через свою надійність та швидкість роботи.

Git і Mercurial це системи управління версіями що відносяться до розподілених систем контроля версій. Репозиторій та його копії локально зберігаються у кожного розробника що користуються даними системами. В них

можна виділити центральний репозиторій куди відправляються усі зміни з локальних репозиторіїв та з яким вони синхронізуються.

Після внесення змін локальну копію користувач відправляє її на сервер.

На рисуноку 3.2 представлена архітектура розподіленої системи контролю версій.

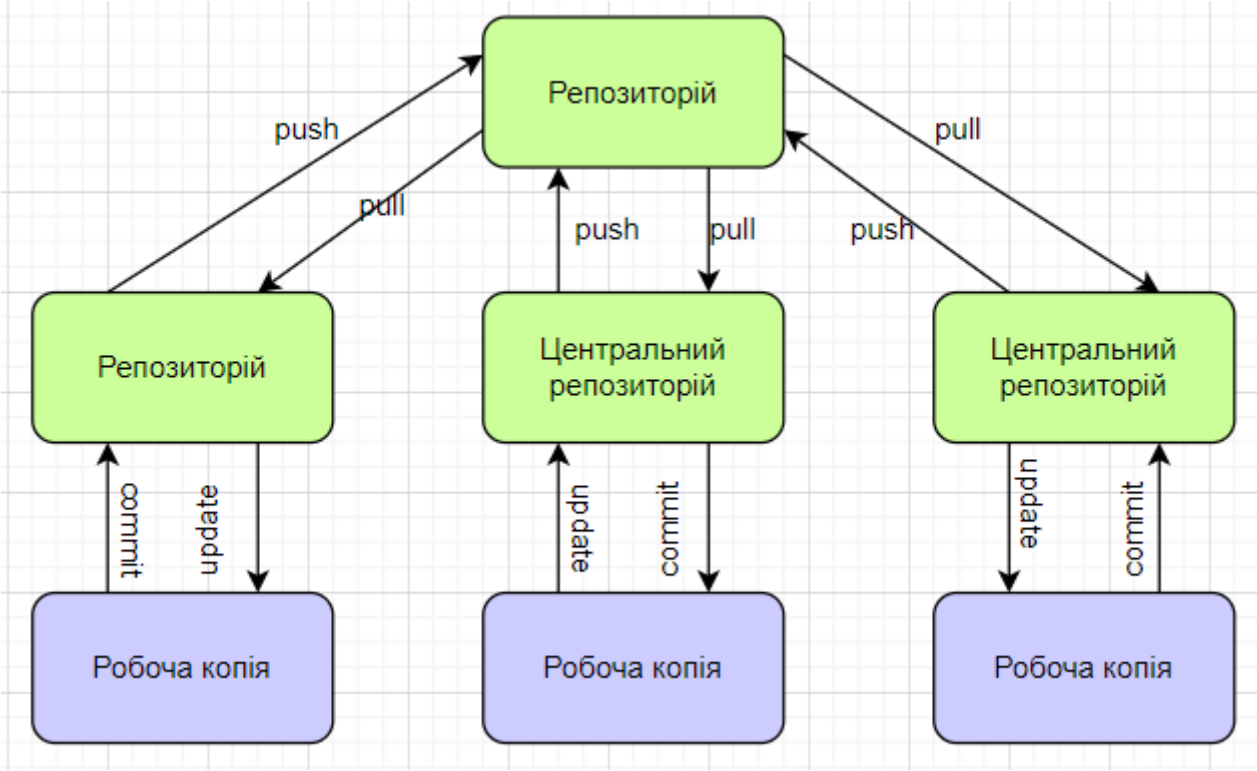


Рисунок. 3.2 Архітектура розподіленої системи контролю версій

3.2 Внутрішнє проектування

3.2.1 Опис функціональних характеристик

Розробка системи аналізу і підвищення мотивації студентів груп

Програмний комплекс «MotivationAnalysis&Enhancement» виконує аналіз і підвищує мотивацію студентів груп. Він написаний на мові C#, для зберігання файлів з варіантами відповідей використовуються формат файлу .json. Зберігання даних тестування виконується базою даних Microsoft SQL server.

Програма має два модулі. Перший модуль (назва: «Personal tests») необхідний для тестування студентів. Він включає в себе 8 різних тестів. Результати обраховується та зберігаються у базі даних. Результат представляє собою число 1, 2 чи 3. 1 означає що студент не підходить для командної роботи, 2 означає що студент підходить для командної роботи, 3 означає що у студента висока мотивація та хист до командної роботи.

Окремо іде тест за методикою тестування Белбіна. Він виявляє ролі людини у команді. Результатом тесту є від однієї до трьох явних ролей у людини. Всі ролі відображають сильні сторони людини та при наявності усіх ролей у команді вона вважається гарною, тобто такою що зможе гарно виконати поставлене завдання, маючи людей які можуть виконати всі завдання та будуть мати при цьому мінімум проблем.

Другий модуль (назва: «Teams division») обчислює можливі команди. Він приймає до себе перелік студентів певної групи. Потім усі студенти що мають результат тесту на мотивацію та командну роботу 1 видаляються, вони не рекомендовані до командної роботи, їм запропоновану видати окремі завдання.

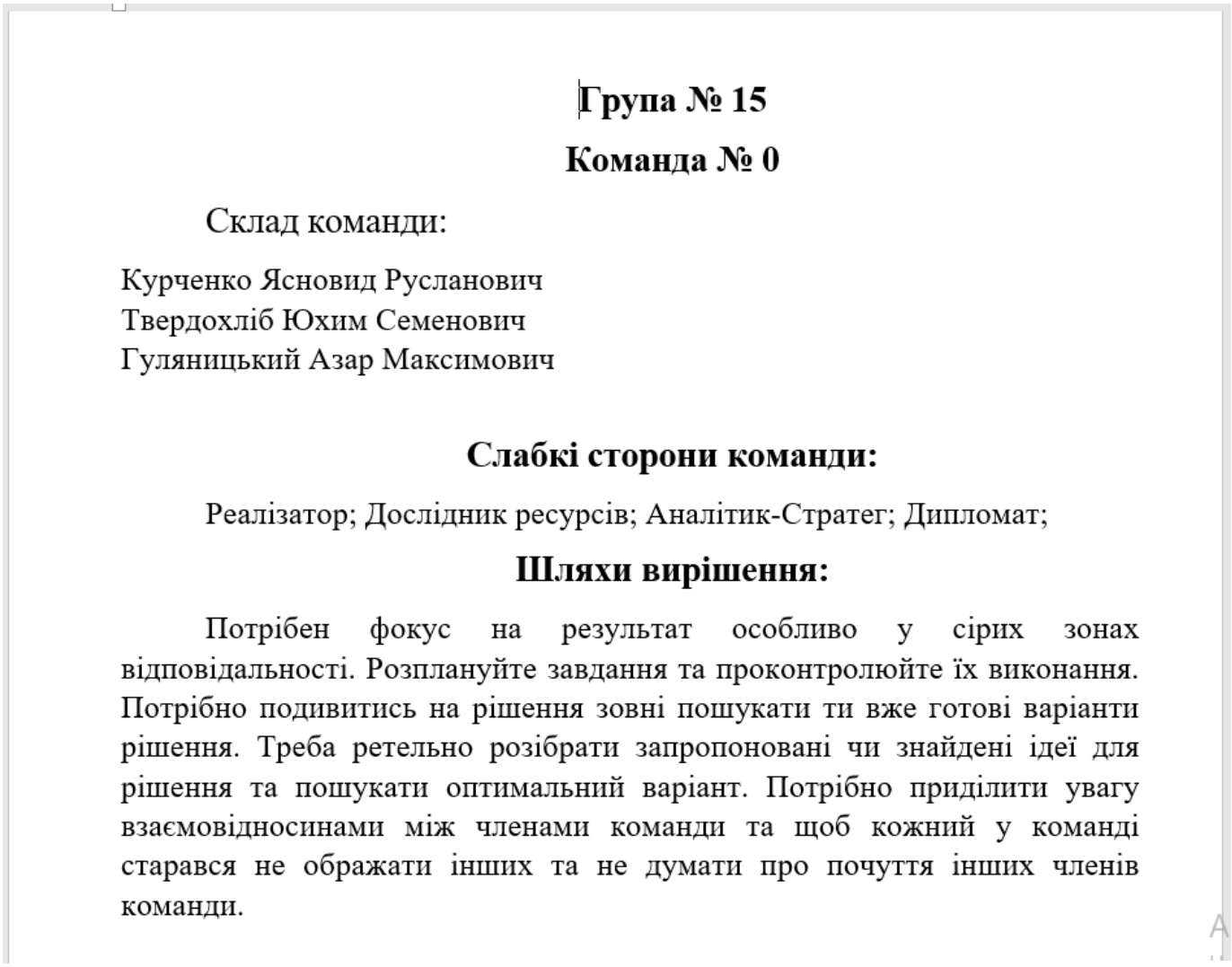
Всі інші студенти потрапляють до програми та розбиваються на команди. Існує велика можливість що якщо ми візьмемо групу студентів і розіб’ємо на команди то ми фізично не зможемо зробити команди які б мали усі 8 ролей, адже зібрати групу людей яка б підходила за рівнем вмінь та мала б усі ролі з трудом можуть навіть на фірмах спеціально відбираючи людей. Тому програма розділяє людей на команди так, щоб команди мали максимальну кількість різних ролей. При цьому кількість людей у команді може бути 3 або 4 чоловіка. Такий розмір команди є оптимальним як у міжнародній практиці так і для студентських груп.

Після цього аналізуються створені команди та виявляються проблеми в них. Проблеми це відсутність тих чи інших ролей. Далі програма дає поради для кожної команди на що їм треба звернути увагу чи за допомогою яких способів

можна покращити їх команди. Поради повинні допомогти людям закрити чи максимально нівелювати недостачу ролей.

В результаті списки створених команд з переліком недоліків та порад формуються у файл в форматі .docx (ці файли використовує програма Microsoft Word). На рисунку 3.3 представлено приклад роботи програми.

Це зроблено задля того щоб викладач міг легко скинути всі файли студентам і сам зручні списки команд.



Група № 15

Команда № 0

Склад команди:

Курченко Ясновид Русланович
Твердохліб Юхим Семенович
Гуляницький Азар Максимович

Слабкі сторони команди:

Реалізатор; Дослідник ресурсів; Аналітик-Стратег; Дипломат;

Шляхи вирішення:

Потрібен фокус на результат особливо у сірих зонах відповідальності. Розплануйте завдання та проконтролюйте їх виконання. Потрібно подивитись на рішення зовні пошукати ти вже готові варіанти рішення. Треба ретельно розібрати запропоновані чи знайдені ідеї для рішення та пошукати оптимальний варіант. Потрібно приділити увагу взаємовідносинами між членами команди та щоб кожний у команді старався не ображати інших та не думати про почуття інших членів команди.

Рисунок. 3.3 Приклад результату роботи програми

Програмний додаток повинен володіти наступним функціоналом:

- провести тестування на рівень мотивації до навчання усіх студентів в групі;
- провести тестування на рівень командної роботи усіх студентів в групі;
- визначити ролі у команді для кожного студента в групі;
- виявити студентів які погано працюють в групах;
- розбитий групу студентів на декілька команд;
- виявити слабкості команд;
- дати поради щодо вирішення проблем у командах.

3.2.2 Вхідні дані

Необхідні на вході програми дані наступні:

- результати тестів студентів;
- результати тесту Белбіна;
- номер групи яку треба розбити на команди.

3.2.3 Вихідні дані

Файл формату .docx у якому му написано:

- номер команди;
- з якої групи команда;
- перелік студентів команди;
- проблеми команди;
- шляхи вирішення проблем команди.

Кількість файлів дорівнює кількості команд.

Personal tests - компонентів системи який виконує тестування студентів. Підраховує результати тестування. Потім сумує всі результати та враховує є здатність людини до командної роботи та його мотивацію. Результат записується в файл.

Teams division – компонентів системи який вибирає з бази даних студентів за обраним параметром (в моєму випадку номером групи). Після цього розбиває групу на команди розміром 3 або 4 людини. Людям з низькою мотивацією буде запропоновано дати окреме завдання. Виводить результат розбиття на групи у файл.

3.2.4 Проектування архітектури системи

В цьому розділі наведений опис розроблених методів та їх взаємодії між собою. Було прийнято рішення розбите завдання на дві окремі програми: програма що проводить тестування студентів та програма що розділяє групу на команди.

Діаграми класів представлені на рисунку 3.4 та рисунку 3.5:

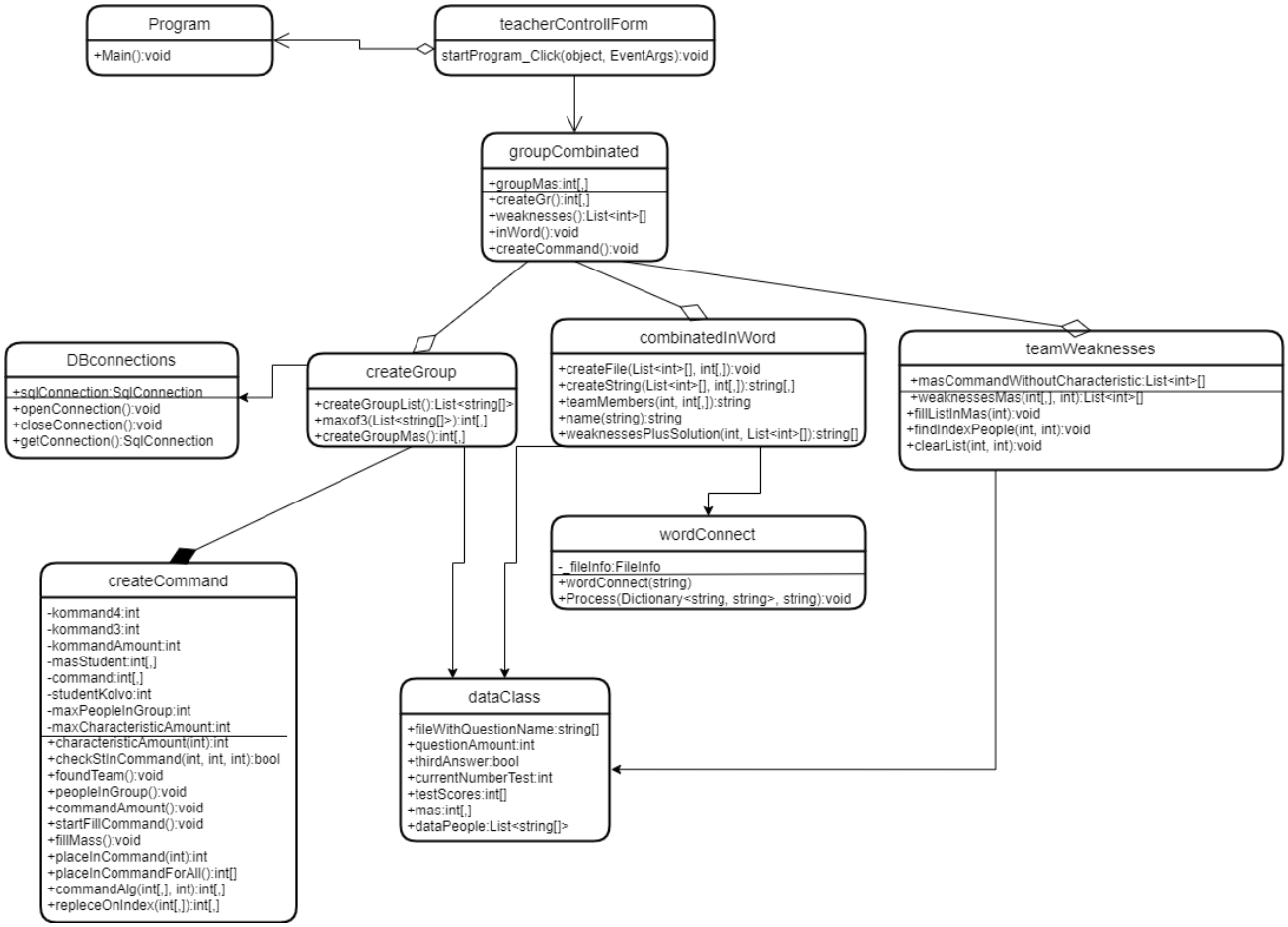


Рисунок 3.4 Діаграма класів для модуля Personal tests

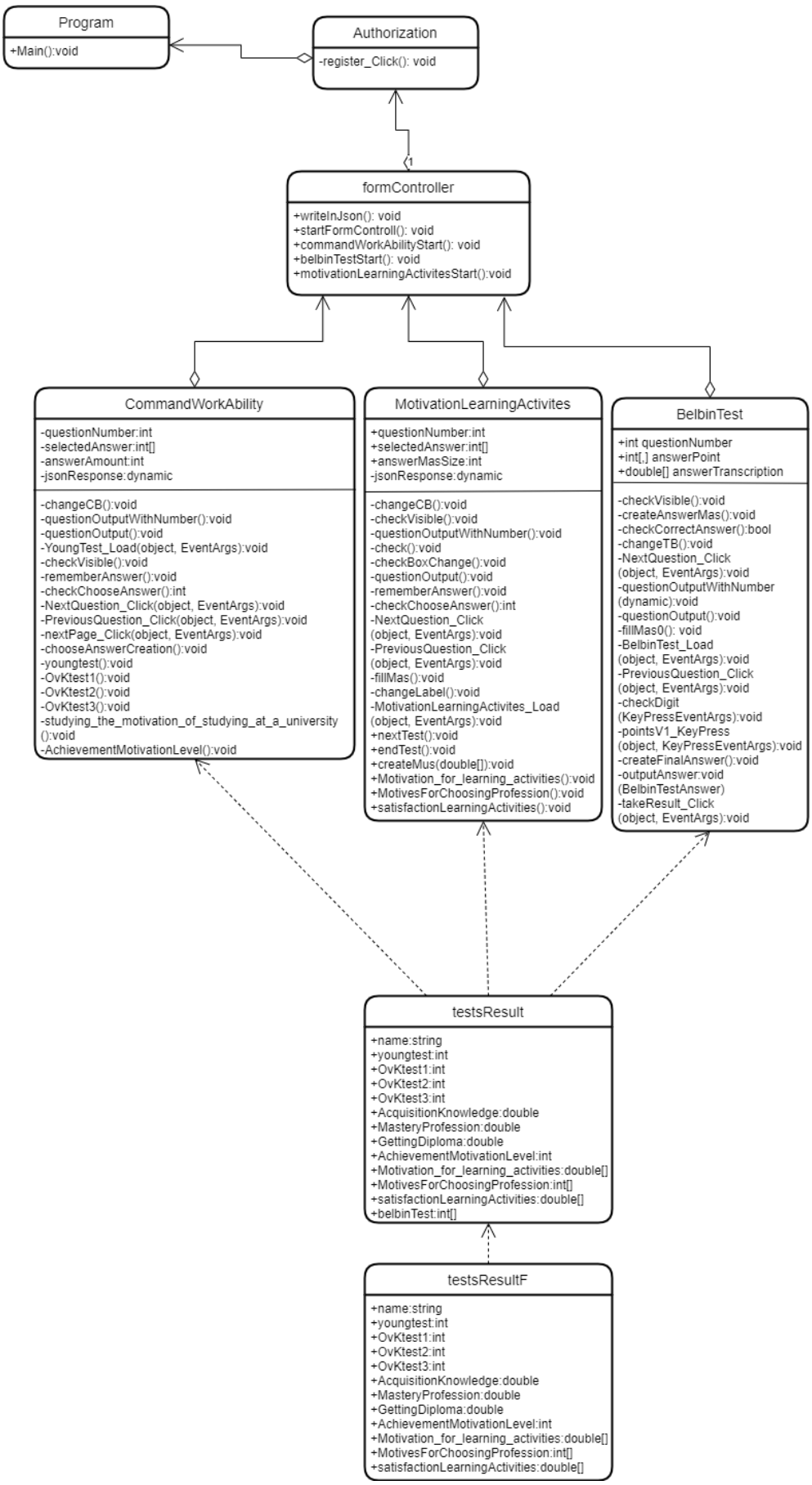


Рисунок 3.5 Діаграма класів для модуля Personal tests

Нижче приведено опис класів програми.

Authorization.cs - клас необхідний для внесення даних щодо випробуваного. Він має у собі графічні форми в яку користувач вводити таку інформацію: своє ім'я та свою групу.

CommandWorkAbility.cs - клас який створює графічне представлення для тестування студентів. Він поєднує у собі перелік тестів. Слугує для таких тестів: тест Юнгу, тести на командну роботу, мотивація до навчання в вузі, мотивація досягнення. Створює форму на яку виводиться питання, варіанти відповідей на запитання та декількох кнопок які дозволяють пересуватися по тесту (перейти до наступного, повернутися до попереднього питання, завершити тест і перейти до наступного тесту). Різні тести мають різну кількість відповідей тому цей клас також адаптує форму під кожний тест. При поверненні до попереднього питання відповідь яку користувач до цього ввів збережеться. Неможливо перейти до наступного питання якщо нема відповіді на поточне. Результати тестування відправляються у клас calculetedResult.cs. Питання та варіанти відповідей записані у .json файлі.

MotivationLearningActivites.cs - другий клас для графічного представлення тесту. Він слугує для праці з такими тестами: мотивація учбової діяльності, мотивація вибору професії, задоволення від навчання. Цей клас також адаптується під кожний тест. Він створюємо графічну форму на яку виводить таку інформацію: питання, варіанти відповідей, інструкцію яка пояснює як проходити тест, кнопки для навігації по тексту (перейти до наступного питання, повернутися до попереднього питання, перейти до наступного тесту). При повертанні до попереднього питання відповідь яку користувач до цього ввів збережеться. Неможливо перейти до наступного питання якщо нема відповіді на поточне. Результати тестування відправляються у клас calculetedResult.cs. Питання та варіанти відповідей записані у .json файлі.

BelbinTest.cs - клас необхідний для графічного представлення тест Белбіна. Тест Белбіна сильно відрізняється від попередніх тестів тому його реалізацію було винесено окремо в цей клас.

Program.cs - клас який першим завантажується при запуску програми. У ньому прописані базові налаштування та викликається клас типу форма який завантажується першим

teacherControllForm.cs - клас необхідний для другого модуля. Представляє собою графічну форму яка приймає номер групи студентів з якої треба вибрати та розподілити на команди. Викликає клас formController.cs та передає йому номер групи.

calculetedResult.cs - клас необхідний для підрахунку результатів тестування. Цей клас має метод для кожного тесту і вираховує результати тестування. Після цього він окремо обчислює комбінацію різних параметрів мотивації та результату різних тестів на командну роботу. Результатом є число 1, 2 чи 3. Це число показує вміння студента щодо командної роботи та його вмотивованість. 1 це поганих хист до командної роботи та погана вмотивованість, 2 означає що студенти підходить для командної роботи але в нього середні результати вмотивованості та хисту до командної роботи (мається на увазі комбінація усіх цих параметрів, тобто можлива ситуація коли у студента середній хист до командної роботи та середня вмотивованість, чи низький хист та висока вмотивованість чи низька вмотивованість але великий хист). 3 означає що у студента великий хист до командної роботи та велика вмотивованість. Окремо вираховується результат тесту Белбіна. В результаті створюється масив даних з характеристиками щодо ролі у командній роботі для студенту.

DBconnections.cs - клас у якому прописані методи та задано шлях до бази даних. Він викликається іншими класами коли їм потрібно відкрити чи закрити з’єднання до бази даних.

DBwrite.cs - клас який потрібний для запису даних у базу даних. Він викликаються іншими класами та має у собі спеціальні методи які змінюють чи вносять в базу даних ти чи інші дані. Це дані про студента та результати його тестування. Цей клас викликає DBconnections.cs задля відкриття з’єднання до бази даних.

formController.cs - це клас який відповідає за виклик форм з тестами. У ньому збережені методи які створюють тий чи інший тест передаючи необхідні дані в клас графічного представлення тестів.

combinatedInWord.cs - цей клас записує результати розділення групи на команди у файл формату .docx.

createCommand.cs - клас який розбиває групу на команди, після чого викликається клас wordConnect.cs для з'єднання з файлом формату .docx. Викликає combinatedInWord.cs для виводу результату у файл.

createGroup.cs - потрібен для створення структури даних яка відображає групу студентів у яких є характеристики у вигляді їх ролей у команді. Після створення цієї структури викликає клас createCommand.cs.

dataClass.cs - клас у якому зберігаються дані.

teamWeaknesses.cs - клас який оцінює створені команди та шукаю у них недоліки, після чого надає варіанти для їх вирішення.

testResultF.cs - клас для збереження даних після тестування.

testsResult.cs - клас для збереження даних після тестування.

wordConnect.cs - клас для з'єднання з файлом формату docx.

3.3 Алгоритм роботи програми

3.3.1 Опис усіх частин програми

Так як програма має дві підпрограми то про кожну буде розказано окремо.

Програмний модуль Personal tests призначений для студентів. Спочатку по черзі викликаються методи з класу formController які запускають тестування (викликають форми з графічним інтерфейсом для тестування передаючи їм дані з інформацією про те який з тестів вони повинні запустити. Користувач має пройти усі тести після запуску додатку. Він не може перепройти чи пройти лише якийсь один тест. Це зроблено через те що всі усі тести видають достовірний результат лише у випадку коли їх проходять разом і одночасно, адже с плином часу результат тестування може змінитися. Через це не можна використовувати результати тесту яким більше ніж півроку, тобто студент повинен проходити даний тест кожен семестр.

Після проходження студентом всіх тестів викликаються класи для обробки результатів і запису їх у базу даних. Після цього програма один завершує свою роботу.

Програмний модуль Teams division призначений для викладачів. Робота з ним починається коли викладач вводить номер групи яку треба розбити на команди. Після цього з бази даних беруться усі студенти які причетні до введеної раніше групи. Всі вони формуються в одну групу у спеціальну структуру: списком масивів. Список являє собою перелік студентів. Кожному студенту відповідає масив з його даними. Дані які представлені в масиві:

- 8 полей зберігають у собі дані про схильність до ролей у студента;
- поле з ID студента;
- поле з ім'ям студента;
- номер групи студента.

Тести на здатність до командної роботи (це 3 тести 2 з яких вираховують чи буде людина гарна у командній роботі і 1 показує конфліктність людини) дають результат у вигляді трьох значень: 1 це поганий результат, тобто людина або не підходить для командної роботи або конфліктна, 2 означає що людина

підходить для командної роботи та не є сильно конфліктною і 3 означає що людина дуже гарно підходить для командної роботи та дуже комунікабельна контактна людина. Результуюча оцінка за цими тестами вираховуються наступним чином: результати складаються та діляться на 3 (кількість тестів), фінальний результат округляється до цілого за математичними правилами.

Тест для вивчення мотивації навчання у ВНЗ Т. І. Ільїной має наступну модель підрахунку результату: якщо у студента превалюють шкали отримання знань та оволодіння професією то за кожну з них він отримує 1 бал, якщо усі три шкали високі то студент отримує три бали. Коли у студента превалюють лише отримання диплому він отримує 0 балів, а коли не превалює жодна віднімається 1 бал.

Далі я оцінюю мотивацію досягнення студента, це ще можна назвати мотивацією до покращення результатів та наполегливості в досягненні своїх цілей. Вона поділяється на низьку середню та високу. Відповідно нараховуються бали: 1 за низький рівень, 2 за середній та 3 за високий.

Наступний це мотивація учбової діяльності за авторством Домбровської І. С. За результатами цього тесту можна дізнатися багато чого про направленість мотивації студента, однак мене цікавить лише кінцевий результат який показує рівень мотивації студента. Він також поділяється на низький середній і високий і в результаті нараховуються теж 1, 2 чи 3 бали.

Тест мотиви вибору професії Р. В. Овчарова вираховують мотиви навчання студенту: це внутрішній індивідуально значення мотиви, внутрішній соціально значимі мотиви, зовнішні позитивні мотиви та зовнішні негативні мотиви. Перші три види мотивів є позитивними, якщо вони переважають для студента чи є високими студент отримує по одному балу за кожну та -1 бал якщо що головне для нього зовнішні негативні мотиви.

Завершує це великий тест за який в студенти може отримати ти від одного до шести балів. Результатами тесту може бути вкрай низька задоволеність

життям та навчання за що студент отримує 1 бал до мотивації, за мотивацію трохи нижче та трохи вище середнього студент отримає 3/4 бали відповідно і за високу мотивацію студент отримує 6 балів. Такі оцінки стоять адже отримати вкрай високу мотивацію дуже важко а вкрай низька мотивація показує що у студента є ряд проблем у житті чи у взаємовідносинах у ВНЗ. Цей тест має найбільший вплив на фінальний результат адже він є одним з найбільших і найточніших з усіх.

Тест Белбіна має незвичайну структуру. Він складається з 7 блоків по 8 питань. У кожному блоці користувач повинен розподілити 10 балів по цим 8 питанням, в залежності від того наскільки він згоден з даним твердженням. Якщо студент залишає 0 то це означає що він зовсім не згоден якщо ставить 10 то це максимально згоден. Він може розділити ці бали між декількома питаннями, але не можна ставити 1 бал, тобто або 0 балів або 2 і вище. Після проходження усіх 7 блоків користувач натискає кнопку завершити тест та його результати зберігаються. Підрахунок переважних ролей у людини виконується за спеціальною таблицею (привести Таблицю). Переважно вважається роль яка набрала більше ніж 10 балів, або якщо таких нема то будь-яка максимальна. Якщо у людини більше 10 балів по декількох ролях то беруться три ролі до яких є найбільша здатність. За умовою тесту у людини може бути від однієї до трьох ролей у команді.

На таблиці 3.1 представлено способ розрахунку результатів тесту Белбіна.

Таблиця 3.1 – способ підрахунку результатів тесту Белбіна

Блок №	РЕАЛІЗАТОР	КООРДИНАТОР	ТВОРЕЦЬ	ГЕНЕРАТОР ІДЕЙ	ДОСЛІДНИК	ЕКСПЕРТ	ДИПЛОМАТ	ФАХІВЦЬ
1	16	13	15	12	10	17	11	14
2	20	21	24	26	22	23	25	27
3	37	30	32	33	35	36	34	31
4	43	47	41	44	46	42	40	45
5	51	55	53	57	54	50	52	56
6	65	62	66	60	67	64	61	63
7	74	76	70	75	73	71	77	72
Підсумок:								

Результати опитування записуються до файлу формату .json, а потім заносяться до бази даних. Такий формат було обрано через те що при проходженні опитування не було можливості напряму контактувати з людьми яких описують тому ці люди проходили тестування на своєму комп'ютері та відправляли результат у вигляді файлу. Потім цей результат заноситься до бази даних і наступна програма яка формує команди вже брала результат з бази даних.

3.3.2 Алгоритм роботи додатку для створення груп.

Користувач вводить номер групи яку треба розбити на команди та починає роботу с програмою.

Створюється структура даних яка зберігає в собі інформацію про дану вибірку студентів. Це масив даних який зберігає ID студенту та його ролі у команді. Після цього викликається клас який розбиває групу на команди.

Робота алгоритму описана нижче.

Спочатку, використовуючи формули 3.1 – 3.3 розраховується загальна кількість команд та скільки команд має 3 або 4 чоловіка.

$$C_4 = S - \left\lfloor \frac{S}{3} \right\rfloor * 3 , \tag{3.1}$$

$$C_3 = \left\lfloor \frac{S}{3} \right\rfloor - C_4 , \tag{3.2}$$

$$C = \left\lfloor \frac{S}{3} \right\rfloor , \tag{3.3}$$

де S – загальна кількість людей, C_3 – кількість команд з 3 людей, C_4 – кількість команд з 4 людей, C - загальна кількість команд.

Далі береться кількість людей рівна кількості команд та записуються у різні команди, наприклад якщо в нас 3 команди то ми беремо з переліку людей, який надано, три людини та записуємо кожну з них в команду в якої ще нікого нема. Це перші люди і нам немає ніякої різниці які в них ролі адже алгоритм намагаються зробити команди з максимальною кількістю ролей і при цьому розподілити рівномірно людей.

Після цього ми беремо людину яка ще не знаходиться в жодній команді. Процес знаходження команди для людини є наступним:

- якщо у людини є три переважні ролі то кожна команда перевіряється наявність схожих ролей. Якщо хоча б одна роль в команді та у цієї людини співпадають то людина вважається непідходящою для цієї команди і перевіряється таким чином для наступної команди;
- якщо у людини є лише дві ролі або людина з трьома ролями не підійшла в жодну команду то схожа на попередній крок перевірка

проводиться для цих студентів, але людина підходить для команди у випадку коли в неї є дві унікальні ролі, тобто для людини з трьома ролями вважається нормальним збіг однієї ролі;

- передостанній крок, сюди приходять люди які мають одну роль чи мають дві або три ролі але не знайшли команду під час попередніх кроків алгоритму. На цьому кроці якщо у людини є хоча б одна унікальна роль тобто хоча б одна роль не представлена у команді вона потрапляє до команди. Якщо після цього кроку алгоритму людина не потрапила в жодну команду вона записується до окремого масиву людей яких нема в командах;
- усі хто не потрапили в команди розподіляються до свобод них місць у командах, тобто доповнюючи команди щоб в них було 3 або 4 людини, згідно кількості таких команд прорахованої вище.

Після створення команд вони перевіряються на слабкість. Тобто йде пошук ролей не представлених у командах. Якщо у команді нема деяких ролей то для кожної команди створюються список не представлених у ній ролей.

І завершує це запис у файл формату .docx. Спочатку записується номер групи до якої належить команда. Далі записується номер команди він дорівнює індексу у масиві, де зберігається інформація про команди. Наступним кроком записується перелік людей представлених у команді. І наприкінці записується перелік які не представлені у команді. Остання частину файлу де записано способи покращення команди формується виходячи з недоліків команди, наприклад якщо у команді немає контролеру команді буде запропоновано вести облік щодо того хто що в команді робить та контролювати цей процес усім разом. Результуючий и файли дуже легко сприймаються людиною та їх можна поширити швидко серед студентів.

Вибір файлу для запису результуючого результату. Критерії такого фалу:

- файл повинен бути поширеним, щоб його могла продивитися максимальна кількість людей;
- файл повинен бути візуально комфортним для сприйняття інформації.

Файл .docx (створюється під час роботи програми Microsoft Word) для результуючої відповіді було обрано через його популярність. Адже це найпоширеніша програма для створення текстових документів і перелік іншим програм можуть працювати з чим форматом (наприклад Open Office, Docx Viewer, Docx Reader, WPS Office і т.д.). Було вирішено не використовувати файли формату .txt через те що вони складуть сприймання інформації. Через це файл формату .docx є ідеальним адже він відкривається у максимальної кількості людей при цьому є зручним у сприйманні.

Створення такого файлу робиться наступним чином. У файлах програми належить створений шаблон файлу з мітками. Програма проводить пошук по меіткам та заміняє їх на визначений напередодні текст. Шаблон файлу наведено на рисунку 3.6.

Група № {groupN}

Команда № {teamN}

Склад команди:

{teamMembers}

Слабкі сторони команди:

{teamWeaknesses}

Шляхи вирішення:

{Problem1} {Problem2} {Problem3} {Problem4} {Problem5} {Problem6} {Problem7} {Problem8}

Рисунок 3.6 Шаблон результуючего файлу

3.4 Тестування

3.4.1 Опис методів тестування

Тестування застосовується для визначення відповідності предмета випробування заданим специфікаціям. До завдань тестування не належить визначення причин невідповідності заданим вимогам (специфікаціям). Тестування — один з розділів діагностики.

Тестування застосовується в техніці, медицині, психіатрії, освіті для визначення придатності об'єкта тестування для виконання тих чи інших функцій. Якість тестування і достовірність його результатів значною мірою залежить від методів тестування та складу тестів.

Процес тестування включає:

- подачу тестового набору;
- визначення реакції об'єкта тестування на тестовий набір;
- оцінку реакції і висновки.

Тестовий набір складається з окремих тестів і розробляється таким чином, щоб забезпечити повне або значне покриття множини ймовірних впливів на об'єкт тестування. Цим, також, визначається складність розробки як окремих тестів, так і тестових наборів.

При розробці додатку для тестування використовувались наступні види і методи тестування:

- модульне тестування;
- інтеграційне тестування;
- системне тестування;
- тестування методом білої скриньки;
- тестування методом чорної скриньки.

3.4.2 Модульне, інтеграційне і системне тестування

Модульне тестування[50] виконується окремо для невеликих елементів системи. Воно проводиться зазвичай одразу після розробки елементу програмного продукту. Це тестування перевіряє функціонал кожного з проектованого компонента на відповідність вимогам.

Інтеграційне тестування[51] проводиться вже для груп елементів або підсистем. Його ще іноді називають комплексним тестуванням. Це тестування необхідне для перевірки коректності роботи взаємодія окремих програмних модулів між собою. Це тестування виконується тільки після того як всі елементи були окремо протестовані та влагодження. Після інтеграційного тестування всі модулі з'єднуються в єдину програму.

Системне тестування[52] виконуються для усієї програми. Це останній етап тестування після модульного та інтеграційного. На цьому етапі перевіряють відповідність роботи усієї програми згідно до заданих значень.

Для тестування моєї програми я спочатку протестував усі модулі окремо після чого перевірив взаємодії між модулями, а у кінці протестував усю програму на коректність заданим вимогам.

Тестування для першої програми було провести дуже легко та швидко адже користувач міг ввести лише малу кількість помилкових значень і цей блок не мав у собі складних обчислень.

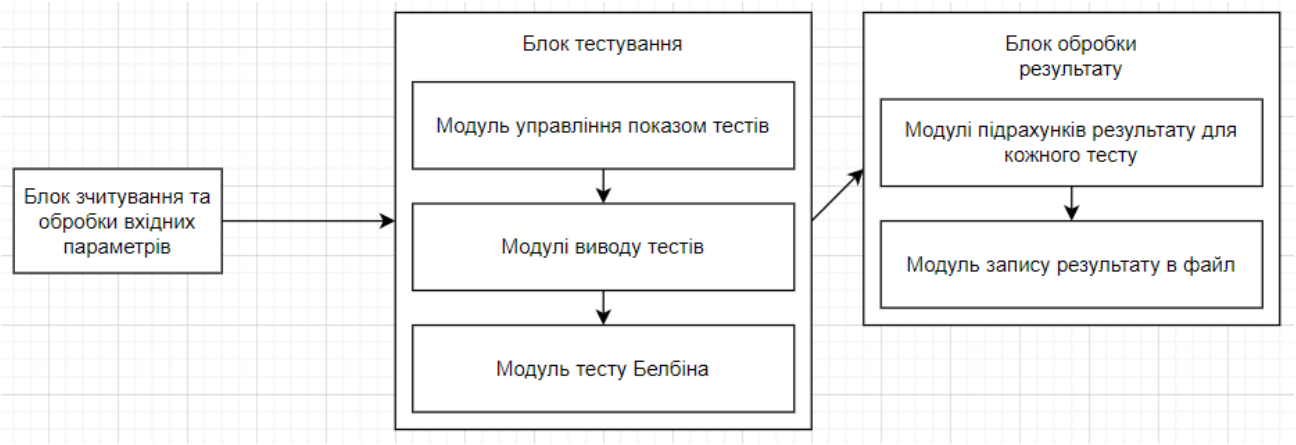


Рисунок 3.7 Комплекс блоків та додатків для програми тестування студентів

Спочатку було проведено тестування та влагодження кожного окремого модуля програми, перелік модулів представлено на малюнку. Під час наступного кроку проводилось інтеграційне тестування: перевіряла взаємодія між модулями програми, зв'язки між модулями представлений на малюнку. Останнім етапом виконувалась системне тестування завданням якого було перевірити функціонування програми ми на відповідність умовам, комплекс блоків в додатків проведено на рисунках 3.7 та 3.8.

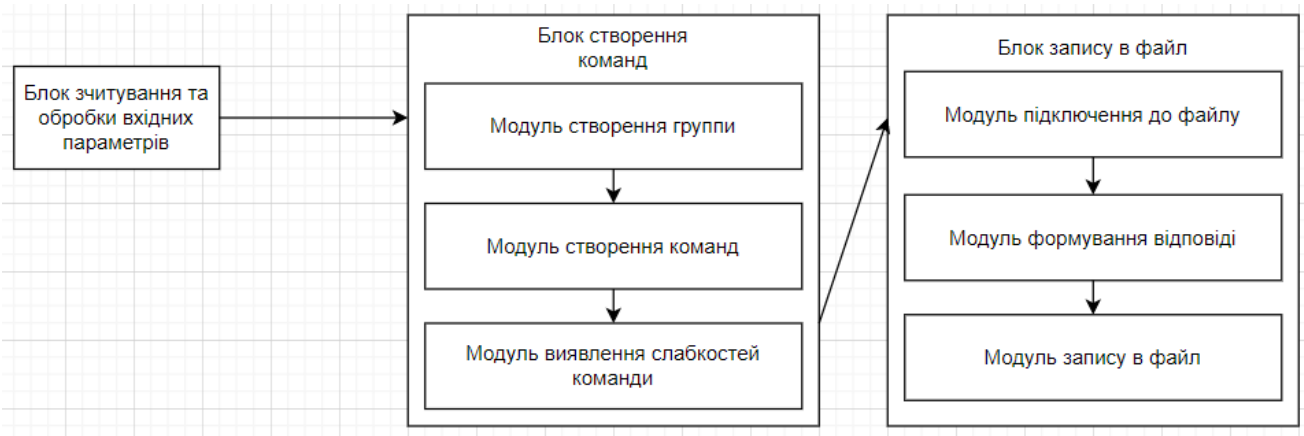


Рисунок 3.8 Комплекс блоків та додатків для програми розподілу студентів на команди

Для першого додатку було протестована такі кейси:

- перехід до наступного питання не обравши відповідь на поточне;
- вибір не коректного місце для зберігання результуючого файлу;
- тестування на показ всіх необхідних тестів;
- в тесті Белбіну тестування на можливість воду не числових даних або одиниці;
- тестування коректності підрахунків результатів тестів;
- тестування коректного переходу до наступного тесту.

Для другого додатку було протестовані ряд кейсів наведених нижче.

- Тестування існуючого номеру групи.

- Тестування на коректність створеного масиву групи:
  - перевірка на унікальність усіх студентів;
  - перевірка на коректність взятих даних;
  - перевірка на коректність даних щодо ролей.
- Тестування розбиття на команди:
  - перевірка створення кількості команд;
  - перевірка коректного розбиття на групи: людина може потрапити в групу тільки якщо задовольняють умовам на поточному кроці алгоритму;
  - перевірка отриманих груп: кожен студент зустрічається хоча б один і не більше одного разу на всі групи;
  - недоліки яка має група не вирішує жоден з наявних у ній студентів.
- Тестування створеного файлу:
  - файл створюються з унікальним ім'ям яке складається з поточної дати поточного часу і номеру команди;
  - файл створюються чи при його створенні виникає помилка;
  - перевірка коректність і даних занесених у файл.

3.4.3 Тестування чорної скриньки

Метод тестування чорної скриньки[53] використовуються для аналізу функціоналу програмного забезпечення, а саме для виявлення невірно виконуємого чи відсутнього функціоналу ПЗ. Під час цього тестування перевіряючий знаходиться у ролі користувача і не використовує ніякі знання про внутрішню структуру перевіряємої системи. Ця методологія дозволяє перевірити коректність виконання програм.

Я використовував п'ять прийомів тестування[54].

Еквівалентне розбиття. Ідея тестування по цьому методу розбиття класів еквівалентності у тому, щуп виключити набір вхідних даних які змушують з тему вести себе однаково і давати однакові результат при тестуванні програми. Оскільки метою тестування є виявлення дефектів успішно тестових сценарій той, який виявляє дефект. Цей процес тестування включає ідентифікацію набору даних якому введення які дають однаковий результат при виконанні програми і класифікують їх як набір еквівалентних даних.

Аналіз граничних значень[55]. Граничні умови ситуацій які виникають на вищих та нижчих границях вхідних класів еквівалентності.

Алгоритм такий:

- визначити класи еквівалентності (попередній прийом);
- визначити межі кожного класу еквівалентності;
- створити тест кейси для кожного граничного значення вибираючи по одній точці безпосередньо на кордоні вища та нижча за кордон.

Аналіз причинно-наслідкових зав’язків. Такий вид тестування має на увазі пошук багатьох причин та наслідків, причина це окрема вхідна умова чи клас еквівалентності. На основі аналізу тематичного змісту специфікації будуються таблиця істинності в якій послідовно перевіряють всі можливі комбінації причин і знаходять наслідки для кожної комбінації.

Припущення про помилку. Основна ідея методу полягає в тому, щоб скласти перелік, який перераховує можливі помилки та ситуації в яких ці помилки могли виявитися. Потім по цьому списку складають тести.

3.4.4 Тестування білою скринькою

Тестування білої скриньки[56] зосереджена на процедурних деталях програмного забезпечення тому його структура тісно пов'язано з вихідним кодом. Інженер випробувач обирає різні вхідні значення щоб перевірити кожен

із можливих потоків виконання програми, щоб переконатися що повертаються правильні вихідні значення.

Хоча тестування методом білою скриньки застосовуються на різних рівнях: модульну інтеграційного та системному, зазвичай воно застосовується до програмних модулів. Її завдання перевіряти потоки виконання всередині кожного модулю, але також ж цей мати може застосовуватись для тестування зав’язків між модулями та великих підсистем і фінальної системи.

Основні критерії покриття коду:

- покриття операторів;
- покриття умов;
- покриття шляхів;
- покриття функцій;
- покриття вхід/вихід;
- покриття значень параметрів.

3.5 Розробка інтерфейсу

Під час розробки програми дуже важливою частиною є інтерфейс, особливо коли користувачі програми звичайні люди а не спеціально навчені оператори.

Для мого додатку необхідно було створити п'ять інтерфейсів:

- інтерфейс авторизації студенту;
- інтерфейс для викладача;
- три інтерфейси для роботи з різними тестами.

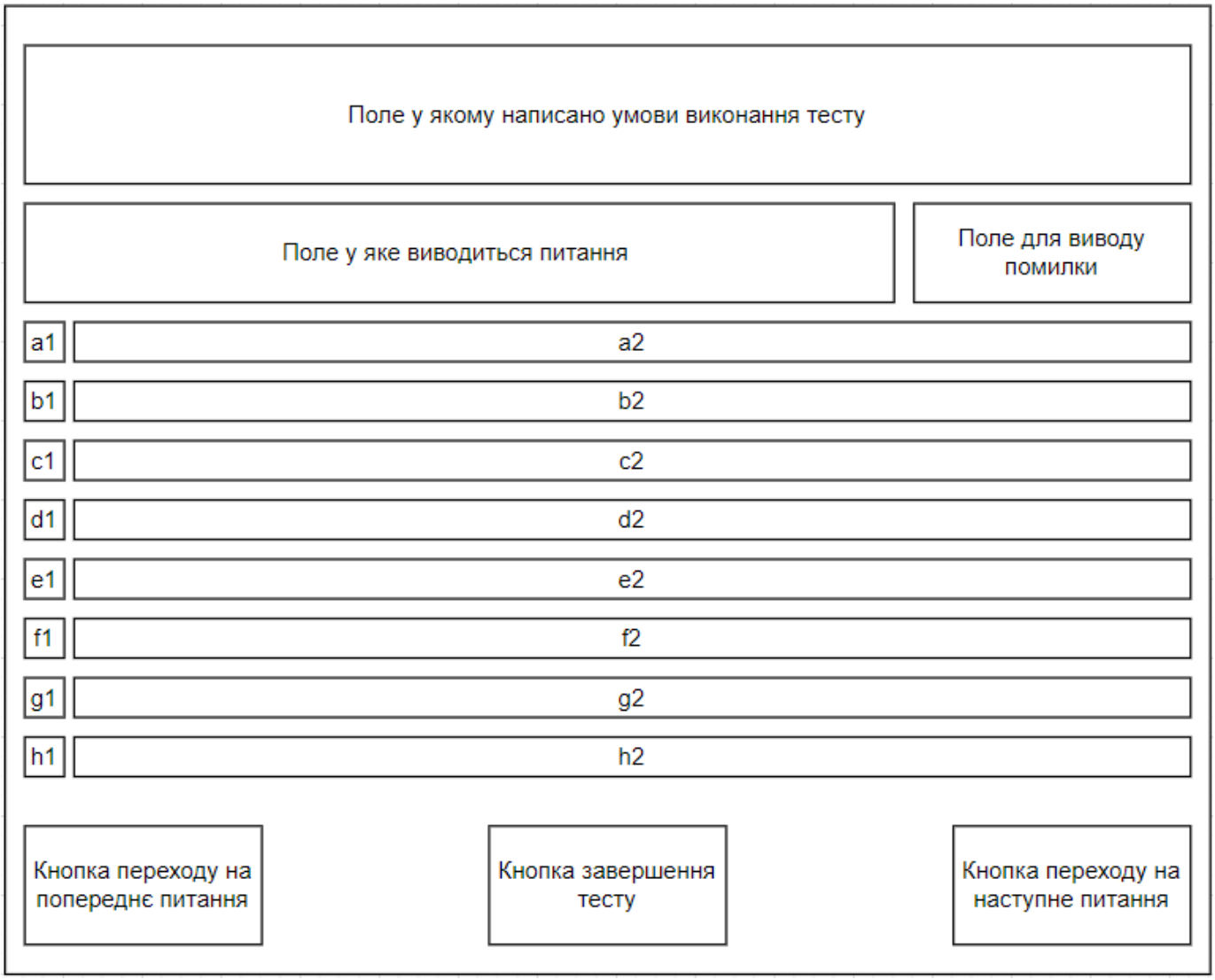


Рисунок 3.9 Форма для проведення тесту Белбіна

Рисунок 3.9 має перелік елементів про які розказано далі. Елементи a1 – h1 це поля для вводу балів згідно умови тесту. Елементи a2 – h2 це поля для виводу тверджень з якими користувач повинен погодитися чи ні згідно умови тесту.

Кнопка переходу на наступне питання неактивна на останньому питанні, кнопка переходу на попереднє питання неактивна коли користувач знаходиться тільки на 1 питанні. Кнопка завершення тесту з’являється тільки на останньому питанні.

Весь текст на формі чорний окрім тексту помилки (помилка означає що користувач неправильно заповнює поля для вводу балів). Для переходу на наступне питання користувач повинен повністю відповісти на поточне.

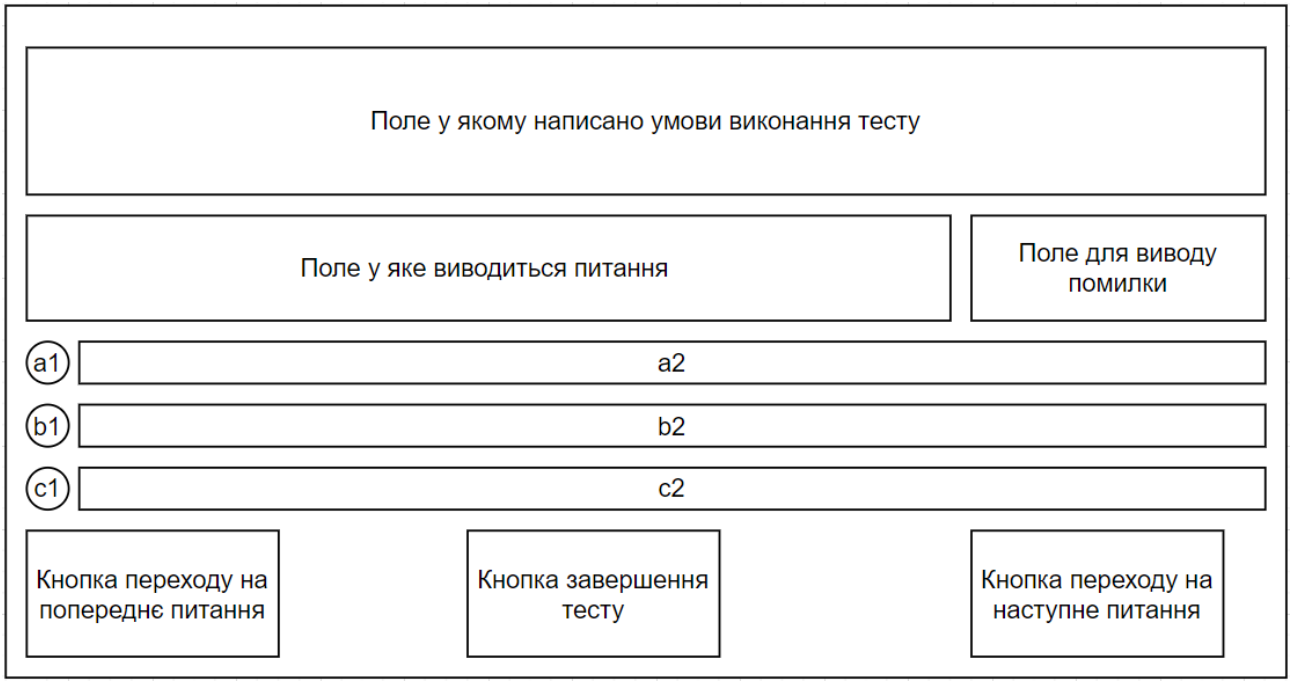


Рисунок 3.10 Форма для проходження тестів

Рисунок 3.10 має перелік елементів про які розказано далі. a1 – h1 це елементи виду check box (це елемент графічного інтерфейсу користувача, що дозволяє користувачу зробити множинний вибір з декількох варіантів). Елементи a2 – h2 це поля для виводу варіантів відповідей. Обраний check box напроти варіанту відповіді сигналізує про те що користувач обрав цей варіант. Одночасно може бути обраний лише 1 варіант

Кнопка переходу на наступне питання неактивна на останньому питанні, кнопка переходу на попереднє питання неактивна коли користувач знаходиться тільки на 1 питанні. Кнопка завершення тесту з’являється тільки на останньому питанні.

Весь текст на формі чорний окрім тексту помилки (помилка означає що користувач неправильно заповнює поля для вводу балів). Для переходу на наступне питання користувач повинен повністю відповісти на поточне.

Форма масштабується під різні тести шляхом зміни кількості варіантів відповідей.

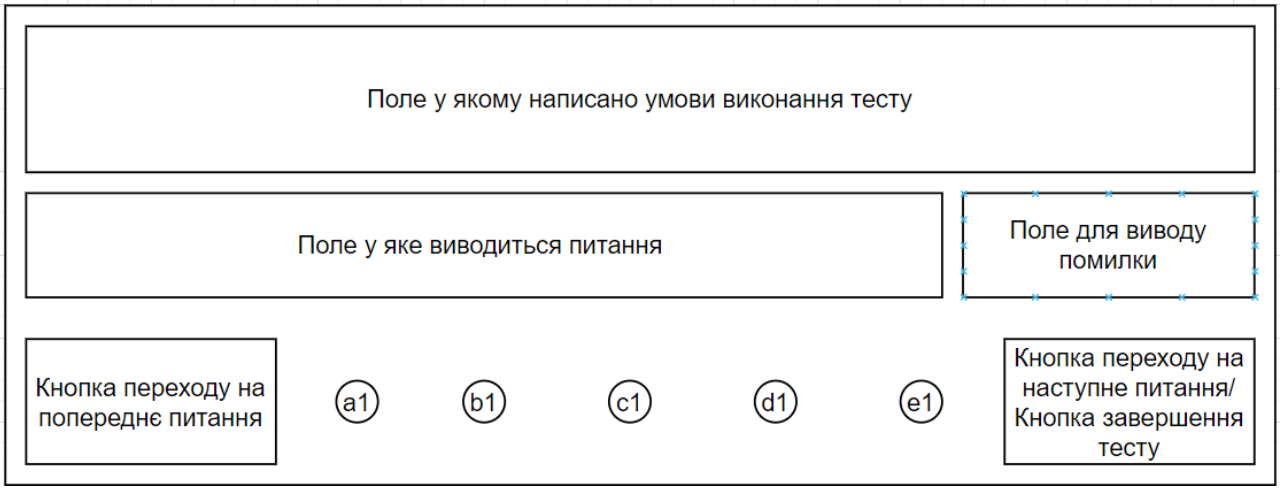


Рисунок 3.11 Форма для проходження тестів

Рисунок 3.10 має перелік елементів про які розказано далі. a1 – e1 це елементи виду check box (це елемент графічного інтерфейсу користувача, що дозволяє користувачу зробити множинний вибір з декількох варіантів[]).

Кнопка переходу на наступне питання неактивна на останньому питанні, кнопка переходу на попереднє питання неактивна коли користувач знаходиться тільки на 1 питанні. Кнопка завершення тесту з’являється тільки на останньому питанні.

Весь текст на формі чорний окрім тексту помилки (помилка означає що користувач неправильно заповнює поля для вводу балів). Для переходу на наступне питання користувач повинен повністю відповісти на поточне.

Форма масштабується під різні тести шляхом зміни кількості варіантів відповідей та їх значенням.

Рисунок 3.12 Форма для авторизації студента

Перед проходженням тестів необхідно ввести своє ім'я та номер групи. Ескіз форми наведено на рисунку 3.12.

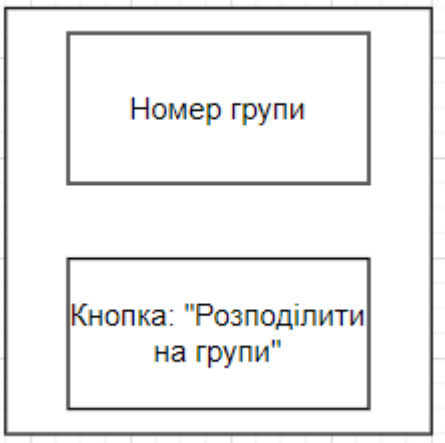


Рисунок 3.13 Форма для програми яка розподіляє студентів по командам

Для розбиття групи на команди треба ввести номер цієї групи. Ескіз форми наведено на рисунку 3.13.

4. ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ВИЗНАЧЕННЯ РІВНЯ МОТИВАЦІЇ СТУДЕНТІВ ТА ЕФЕКТИВНОСТІ СТВОРЕННОЇ СИСТЕМИ МОТИВАЦІЇ ГРУП СТУДЕНТІВ

4.1 Дослідження ефективності обраних тестів при виявленні мотивації студента до навчання

4.1.1 Опис методології проведення першого експерименту

Щоб розробити стратегію для групи з різною мотивацією виконавців треба спочатку дізнатися їх мотивацією. Для цього я застосовую комплексне тестування студентів.

Задля виявлення справжньої мотивації студентів та зниженні вплива похибок які виникнуть при малому обсязі питань через те що у деяких питаннях опитувальний може навмисно чи ненавмисно брехати (наприклад неправильно зрозуміє питання або не буде мати достовірну інформацію щодо нього або відповість прикрашаючи себе).

Під час аналізу видів мотивацій були виявлені наступні напрямки для оцінки у студентів:

- оцінка мотивації досягнення;
- вивчення мотивації навчання у вузі;
- дослідження соціальної та пізнавальних мотивацій;
- мотиви вибору професії;
- задоволеність учбовою діяльністю;
- оцінка екстравертності;
- особливості людини як командного гравця.

Після даного аналізу можна буде сказати чи рекомендується людина до командної роботи, чи краще не брати цю людину до команди взагалі, а дати їй окреме завдання.

4.1.2 Опис використаного програмно - апаратного середовища

Для дослідження було використане програмно-апаратний середовище власної розробки наведене у попередньому розділі.

Дослідження за допомогою цього середовища проводилися у декілька етапів: спочатку було проведено тестування групи студентів для виявлення їх мотивації та потенціалу командної роботи, потім була обрана команда у членів якої було виявлено їхній риси характеру та запропоновані методи задля покращення взаємодії учасників команди.-

Останнім етапом дослідження було порівняння отриманих результатів задоволення від командної роботи і виконана робота чи ні з результатами які були прогнозовані.

4.1.3 Проведення експерименту

Було проведено опитування 12 людей що навчаються або закінчили навчання не більш ніж 2 роки назад. Результати тестів були проаналізовані.

4.1.4 Оцінка результатів експерименту

Експеримент показав чи спростував залежності у мотивації студентів. Загальна мотивація до навчання у людей представлена на рисунку 4.1

Значення на рисунку 4.1 по Y(вертикалі): 0 – мотивація відсутня, 14-мотивація максимально висока. Значення на рисунку 4.1 по X(горизонталі): номер студента.

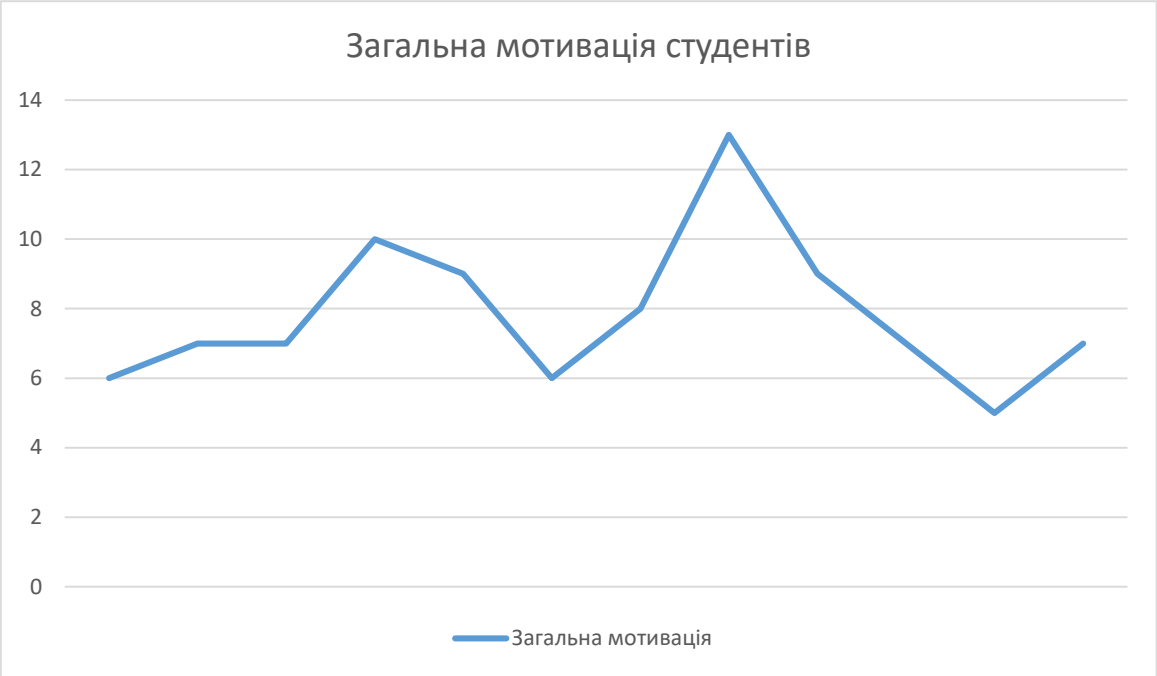


Рисунок. 4.1 Графік загальної мотивації студентів.

Дослідження залежності інтроверсності та командних навичок людини на рисунку 4.2 по Y(0 – людина інтроверт/низькі командні навички, 1 – людина екстраверт, середні командні навички), та номер людини по X:

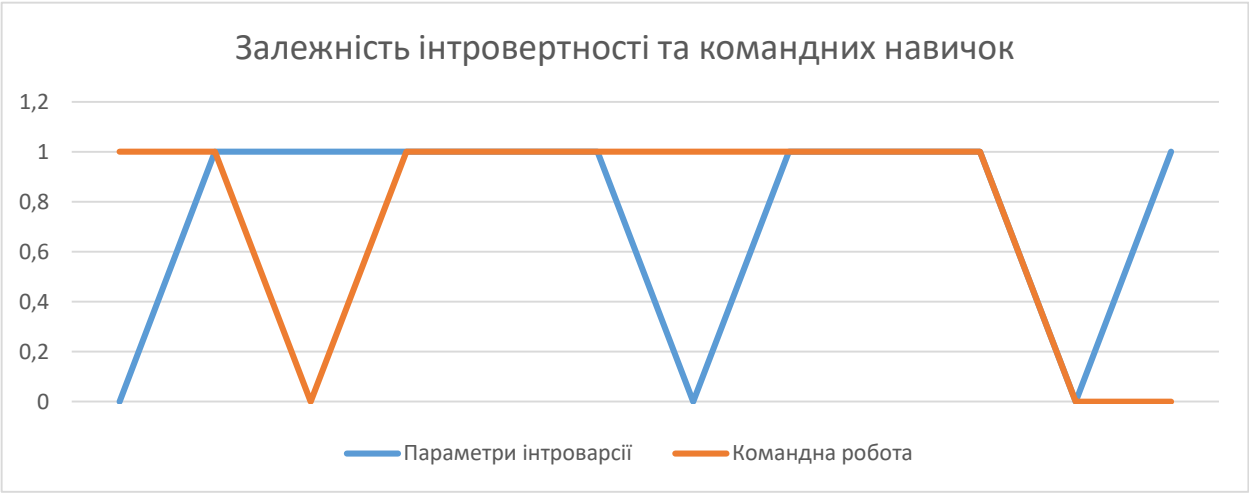


Рисунок. 4.2 Графік залежності інтровертності та командних навичок

На графіку можна побачити що нема прямої залежності між тим що людина є інтровертом і немає навичок для командної роботи, тобто ці величини не зв’язані.

Мотивація досягнення відображає мотивацію людини до навчання. Значення на рисунку 4.3 по Y(вертикалі): 0 – мотивації досягнення немає, 1-мотивація досягнення є, 0 – мотивація відсутня, 14-мотивація максимально висока. Значення на рисунку 4.3 по X(горизонталі): номер студента.

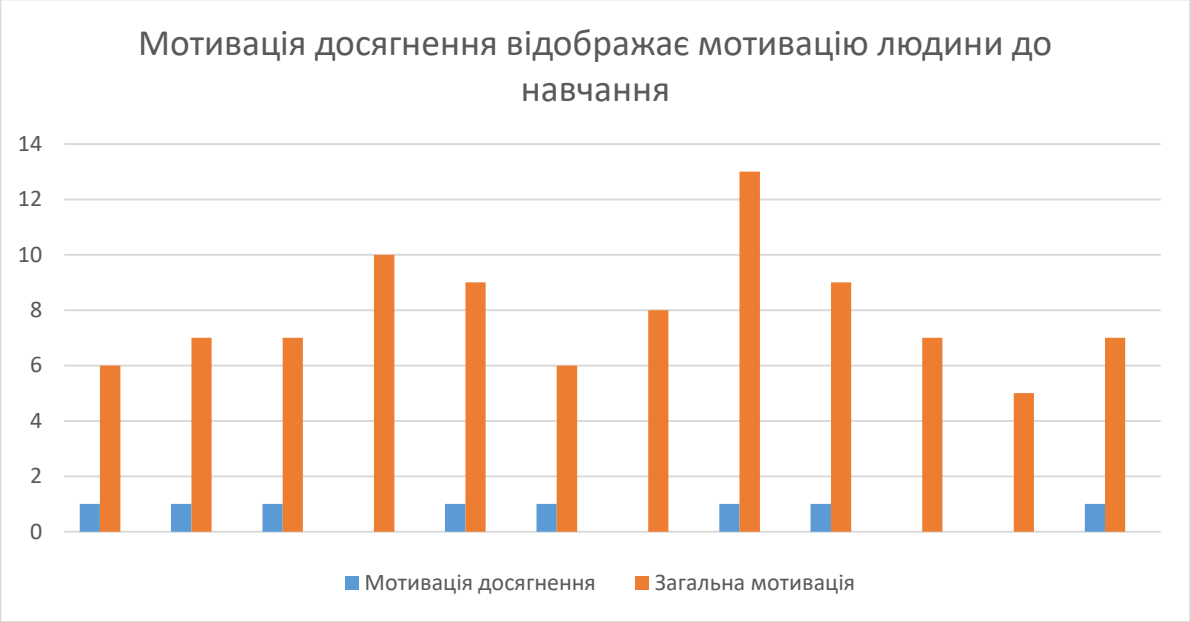


Рисунок. 4.3 Графік залежності мотивації досягнення та загальної мотивації студента.

Мотивація досягнень лише частково співпадає із загальною мотивацією, зв’язку майже нема.

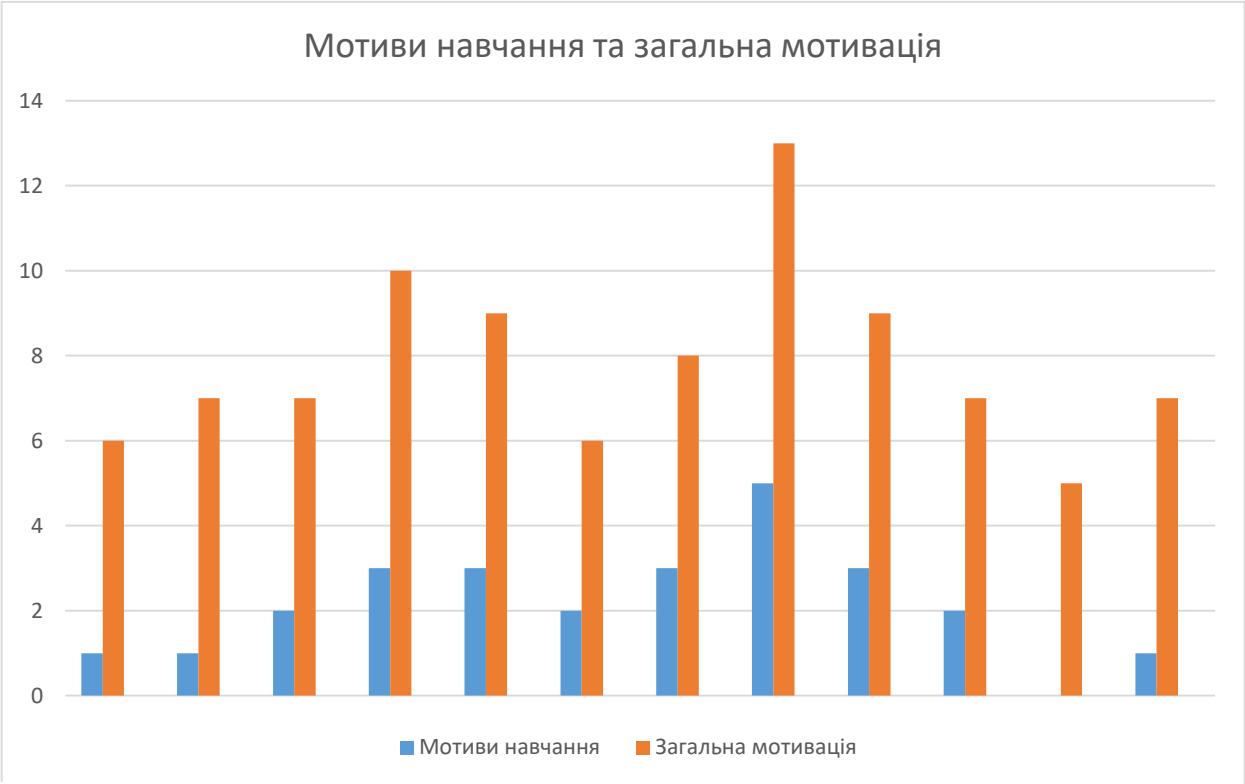


Рисунок. 4.4 Графік залежності мотивів навчання та загальної мотивації студента.

Значення на рисунку 4.4 по Y(вертикалі): 0 – мотивація навчання дуже низька, 6- мотивація навчання висока є, 0 – мотивація відсутня, 14-мотивація максимально висока. Значення на рисунку 4.4 по X(горизонталі): номер студента.

Мотивація навчання співпадають із загальною мотивацією, із чого можна сказати, що ті студенті хто мають більше мотивів для навчання у вузі більш вмотивовані.

Задоволеність учбовим процесом та загальна мотивація представлена на рисунку 4.5.

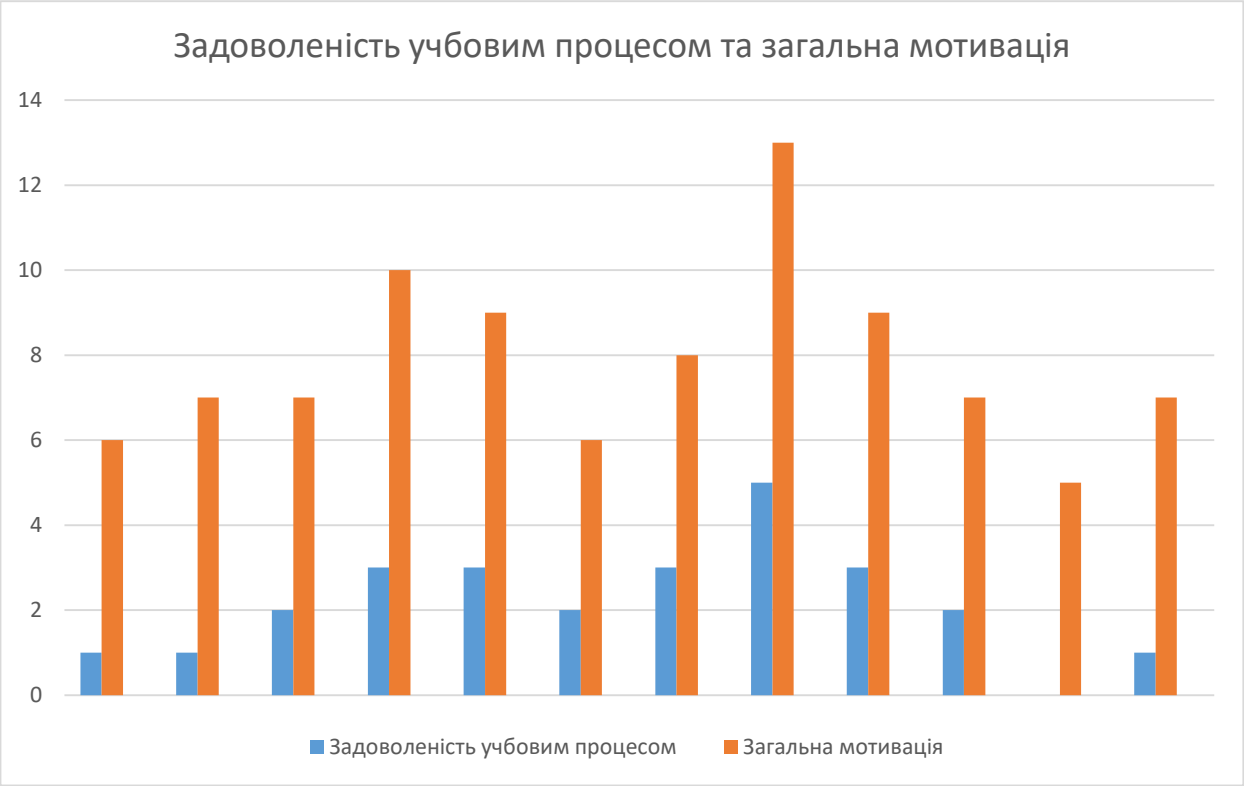


Рисунок. 4.5 Графік залежності задоволеності учбовим процесом та загальної мотивації студента.

Значення на рисунку 4.5 по Y(вертикалі): 0 – задоволеності учбовим процесом нема, 6- задоволеність учбовим процесом максимальна, 0 – мотивація відсутня, 14-мотивація максимально висока. Значення на рисунку 4.5 по X(горизонталі): номер студента.

Задоволеність навчальним процесом також напряму впливає на загальну мотивацію.

Пошук поганих та гарних студентів для командної роботипроводиться наступним чином. Студенти з високою мотивацією та гарні у командній роботи є тими хто майже не потребує додаткової мотивації і навпаки студенті з низькою мотивацією треба виключити з роботи. Висока чи низька мотивація вираховується виходячи з середньої мотивації групи. На рисунку 4.6 представлено графік на якому зображено порівняння загальної мотивації студентів, хисту до командної роботи та їх зв’язку.

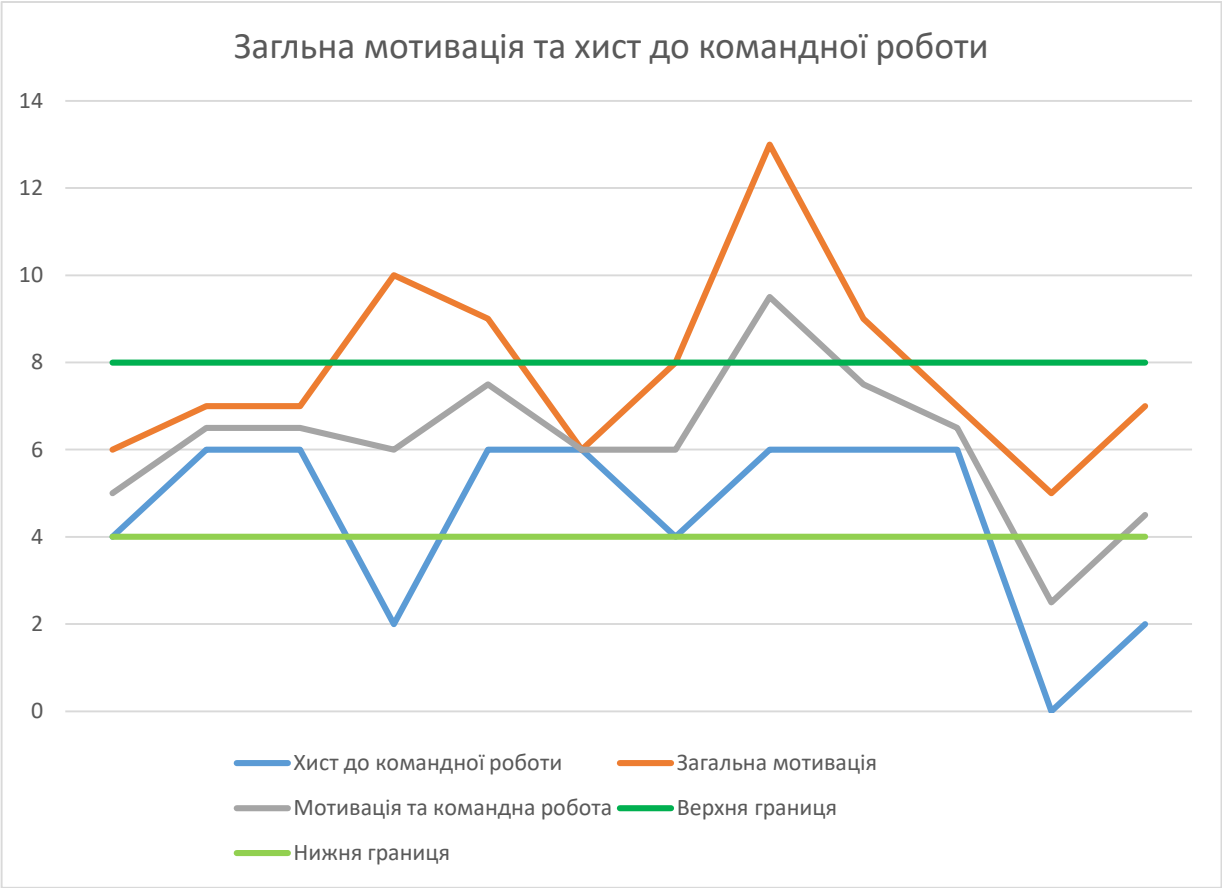


Рисунок. 4.6 Графік мотивації студентів до навчання

4.2 Оцінка зміни рівня мотивації команди студентів при застосування ними системи мотивації команди

4.2.1 Опис методології проведення другого експерименту

Другий експеримент проводився над командою із 4 людей. Були виявлена слабкість цієї команди та запропоновані шляхи подолання цієї слабкості. Для визначення можливого результату виконання завдання та задоволеності учасників від процесу було досліджено виконання цією командою роботи у їх звичайному темпі. Порівнювалась командна мотивація виконавців до і після виконання ними роботи з розробленою мною системою мотивації.

4.2.2 Опис використаного програмно - апаратного середовища

Для дослідження було використане програмно-апаратний середовище власної розробки наведене у попередньому розділі.

Останнім етапом дослідження було порівняння отриманих результатів задоволення від командної роботи і виконана робота чи ні з результатами які були прогнозовані.

4.2.3 Проведення експерименту

У експерименту було 3 етапи. Перший етап повинен був виявити групову мотивацію команди яка вже працювала разом. Така умова була через те що неможливо провести заміри на групі яка ще не працювала, адже можна блок дати ти лише якісь експертні оцінки які мало на чому базується. Дійсно гарно провести експерименти над людьми які працювали разом і яким заважали недоліки їхньої команди, після чого сказати їм про їх недоліки та запропонувати шляхи вирішення. Така команда була знайдена і нижче наведено результати на початок експерименту, та опис їхньої команди.

Було виявлено такі негативні моменти:

- команда погано виконує завдання в строк;
- у команди проблеми з генерацію ідеї.

Було виявлено такі позитивні моменти:

- учасники команди досить швидко разом виконують окремі частини завдання які отримують;
- якщо ідея вже придумано проблем з її обробкою і реалізацією мало;
- проблем з лідером нема.

Команда має такий склад:

- учасник А: реалізатор, шейпер;
- учасник Б: реалізатор;
- учасник В: аналітик-стратег, координатор;
- учасник Г: душа команди, аналітик-стратег.

Можна побачити що в цій команді нема генератора ідей, контролера та дослідника. Ось які рекомендації дії дала моя програма: “Команді не вистачає креативних нових ідей треба проводити мозковий штурм задля пошуку рішення завдання. Потрібно подивитись на рішення зовні пошукати ти вже готові варіанти рішення. Потрібно ввести чек-листи та адміністративний контроль за завершенням задач”.

Як ми бачимо з складу ролей в команді вони не повинні мати проблем з лідером, адже вони мають і координатора і шейпера, а коли в команді є люди з цими двома ролями проблем з керуванням групи зазвичай небагато. Через те що в групі нема генератора ідей та дослідника у команди є проблеми з генерації нових ідей, адже люди які повинні придумувати ідеї чи шукати їх зовні відсутні. Однак через те що в команді є аналітики-стратеги та реалізатори в ситуації коли ідея є команда легко її обробляє та реалізує.

Проблеми команди з менеджментом часу присутні через те що в команді нема контролера який би слідкував за іншими учасниками команди і вчасно повідомляв їх проте що вони не встигають робити ту чи іншу частину роботи та їм треба поквапитись чи сказав бо їм що вони неправильно розподіляють завдання у часі

Після того як стало зрозуміло що команди є ті чи інші недоліки наступна робота команди була виконана з використанням порад які дала команда.

При отриманні завданні учасники провели мозковий штурм за для генерації нових ідей та провели пошук аналогів в інтернеті. Це значно прискорило проблемний для команди етап, а через наявність аж двох аналітиків-стратегів вони досить швидко змогли проаналізувати усі варіанти які виникли та сформулювати ідею свого рішення.

Подивившись на другу пораду команда вирішила використати для планування часу при роботі над завданням діаграму Ганта. Це допомогло команді більш пильно слідкувати за часом та розуміти коли їм треба ба робити

чи інші частини завдання щоб вкластися у строк. Через те що команди не було проблем з соціальними ролями та виконавцями процес роботи над програмою йшов так як вони звикли, адже у ньому не повинно було бути проблем за результатом роботи програми (за результатами опитування команди ми бачимо що в них і до цього не було проблем з цією частиною роботи).

4.2.4 Оцінка результатів експерименту

Для ілюстрації покращення чи погіршення стану команди було використано тест "Діагностика групової мотивації" від І. Д. Ладанова.

Результати опитування на початку експерименту (до пошуку проблем у команді):

- учасник А: 118 балів;
- учасник Б: 134 бали;
- учасник В: 153 бали;
- учасник Г: 123 бали.

Середня по команді оцінка це (118 + 134 + 153 + 123) / 4 = 132. Це нормальний результат, який означає що група достатньо мотивована на досягнення успіху у діяльності.

Результати опитування після роботи із використанням порад:

- учасник А: 120 балів;
- учасник Б: 146 бали;
- учасник В: 171 бали;
- учасник Г: 122 бали.

Середня по команді оцінка це (120 + 146 + 171 + 122)/4 = 139,75. Це нормальний результат, який означає що група достатньо мотивована на досягнення успіху у діяльності.

Ми бачимо, що рівень мотивації команди підвисився, через що можна казати про успішність застосованого методу.

Висновок: команда змогла покращити свої результати через розуміння наявних у ній проблем та завдяки їх вирішенню. Командний рівень мотивації також підріс. Проблеми які були озвучені учасниками команди підтвердилися після використання програми та аналізу результатів її роботи. Експеримент можна вважати вдалим.

Висновок до 4 розділу

Перше дослідження дозволило побачити як ті чи інші тести пов'язані з фінальним результатом да між собою. Результати тестів на інтроверсію та екстраверсію і блоку тестів які показують навички командної роботи виявились слабо зв'язаними між собою. Це покаже що тест Юнга не є реальним показником в питанні можливості людини працювати у команді. Мотивація досягнення в усіх опитуваннях була не вище середнього рівня, через це неможливо реально сказати як сильно вона впливає на фінальний результат. Мотиви навчання впливають на пряму та загальну мотивацію, тобто чим їх більше тим вище загальна мотивація. Серед опитуваннях велика кількість різних результатів. Фінальний тест який перевіряє задоволення швом процесу показав свою важливість адже усі результати напряму зв'язані з ним, тобто у людей з високою мотивацією у цій рубриці висока задоволеність службовим процесом а у людей з низькою мотивацію ця задоволеність теж низька.

В результаті серед запрошених було виявлено одну людину яка мала занадто низький рівень мотивації та навички командної роботи, таку людину буде запропоновано не брати у команду.

Другий експеримент показав гарну якість розробленої системи мотивації. Вже існуючих група змогла покращити результат скориставшись порадами які дала їй програма.

Результуючий усе вищесказане можна казати про те що тест Юнга можна вилучити з переліку тестів, однак розроблена система мотивації є робочою.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Дослідження трудовитрат при групових роботах [Ел. ресурс]. Доступ: https://www.cmu.edu/teaching/technology/index.html [Дата звернення 11.09. 22]
2. Успешная команда – успешный бизнес [Ел. ресурс]. Доступ: https://kachestvo.pro/kachestvo-upravleniya/proektnoe-upravlenie/uspeshnaya-komanda-uspeshnyy-biznes/ [Дата звернення 11.09.22]
3. Потребности [Ел. ресурс]. Доступ: https://web.archive.org/web/-20090501232752/http://www.glossary.ru/cgi-bin/gl_sch2.cgi?RPuywlhtuxyo [Дата звернення 11.09. 22]
4. Ожидание [Ел. ресурс]. Доступ: https://ru.wikipedia.org/wiki/Ожидание [Дата звернення 12.09. 22]
5. Теория и практика мотивации персонала [Ел. ресурс]. Доступ https://searchinform.ru/kontrol-sotrudnikov/motivatsiya-personala/teoriya-i-praktika-motivatsii-personala/ [Дата звернення 13.09. 22]
6. Піраміда Маслоу https://termin.in.ua/piramida-maslou/ [Дата звернення 13.09. 22]
7. Двухфакторная теория мотивации Герцберга https://ru.wikipedia.org/wiki/-Двухфакторная_теория_мотивации_Герцберга
8. *Виктор Врум.* Труд и мотивация. — 1964. — 331 с.
9. Теория ожиданий [Ел. ресурс]. Доступ: https://ru.wikipedia.org/wiki/Теория_ожиданий#cite_note-:0-2 [Дата звернення 14.09. 22]
10. Теория справедливости Дж. С. Адамса [Ел. ресурс]. Доступ: https://ru.wikipedia.org/wiki/Теория_справедливости_Дж._С._Адамса [Дата звернення 14.09. 22]
11. Adams J.S.. Inequality in social exchange // Advanced Experimental Psychology, 1965. 62: 335—343.

12. Теорія мотивації Портера-Лоулера [Ел. ресурс]. Доступ: https://uk.wikipedia.org/wiki/Теорія_мотивації_Портера-Лоулера [Дата звернення 15.09. 22]
13. Парадокс Мак-Грегора [Ел. ресурс]. Доступ: https://www.talent-management.com.ua/5009-paradoks-mak-gregora-samoe-tragicheskoe-nedorazumenie-v-istorii-raboty-i-organizatsij/ [Дата звернення 15.09. 22]
14. Теория Z [Ел. ресурс]. Доступ: https://ru.wikipedia.org/wiki/Теория_Z [Дата звернення 15.09. 22]
15. ToDoist [Ел. ресурс]. Доступ: https://todoist.com/ru [Дата звернення 16.09. 22]
16. Pyrus [Ел. ресурс]. Доступ: https://pyrus.com/ru [Дата звернення 16.09. 22]
17. MeisterTusk [Ел. ресурс]. Доступ: https://www.meistertask.com/ru?utm_source=google&utm_medium=cpc&utm_campaign=World_ru_Brand_brand&utm_content=homepage&utm_term=meistertask&utm_campaign=World_ru_Brand&utm_source=adwords&utm_medium=ppc&hsa_acc=8361758703&hsa_cam=1033738504&hsa_grp=49791905254&hsa_ad=361345248171&hsa_src=g&hsa_tgt=aud-416360927900:kwd-110394787007&hsa_kw=meistertask&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAiAkfucBhBBEiwAFjbkr2rddPW2aM9SNV144yBo5sDqyLbPGRKKwWxeZe42mpbReqUL5KulJhoCLpoQAvD_BwE [Дата звернення 16.09. 22]
18. Loop [Ел. ресурс]. Доступ: https://play.google.com/store/apps/details?id=org.isoron.uhabits&hl=ru&gl=US&pli=1 [Дата звернення 16.09. 22]
19. The Nine Belbin Team Roles [Ел. ресурс]. Доступ: https://www.belbin.com/about/belbin-team-roles [Дата звернення 17.09. 22]
20. Шкала оценки потребности в достижении [Ел. ресурс]. Доступ: http://testoteka.narod.ru/ms/1/19.html [Дата звернення 17.09. 22]
21. Карелин А. Большая энциклопедия психологических тестов. - М.: Эксмо, 2007. - 416 с. (с. 20-21)

22. Методика изучения мотивации обучения в вузе Т.И. Ильиной [Ел. ресурс]. Доступ: http://testoteka.narod.ru/ms/1/05.html [Дата звернення 17.09. 22]
23. Тест Т. И. Ильиной [Ел. ресурс]. Доступ: http://www.psychometrica.ru/index.php?hid=50&met_info=200 [Дата звернення 18.09. 22]
24. Мотивация учебной деятельности: уровни и типы [Ел. ресурс]. Доступ: http://testoteka.narod.ru/ms/1/21.html [Дата звернення 18.09. 22]
25. Герасимова А.С. Теория учебной мотивации в отечественной психологии М.: Эксмо, 2007. - 416 с. (с. 20-21)
26. Маркова А.К. Формирование мотивации учения в школьном возрасте. – М., 1983
27. Рожков М.И. Методика социализированности личности 1994. —55с.
28. «Мотивы выбора профессии» [Ел. ресурс]. Доступ: http://testoteka.narod.ru/ms/1/18.html [Дата звернення 18.09. 22]
29. Врублевская М.М., Зыкова О.В. Профориентационная работа в школе: Методические рекомендации. - Магнитогорск: МаГУ, 2004. - 80 с. (с. 7 - 9)
30. До проблеми діагностики ставлення студентів до навчальної діяльності [Ел. ресурс]. Доступ: https://psyjournals.ru/files/28868/vestnik_psyobr_2007_3_Mizchenko.pdf [Дата звернення 19.09. 22]
31. Тест Юнга Ел. ресурс]. Доступ: https://experimental-psychic.ru/test-associacij-yunga/ [Дата звернення 19.09. 22]
32. Командний тест Белбіна [Ел. ресурс]. Доступ: http://tests.puet.edu.ua/files/test2-2.pdf [Дата звернення 19.09. 22]
33. C++ reference. URL: http://en.cppreference.com/w/cpp [Дата звернення 19.09. 22]
34. Документация Java. URL: https://docs.oracle.com/javase/8/ http://java.net/ [Дата звернення 20.09. 22]
35. C# Reference. URL: https://msdn.microsoft.com/en/library/618ayhy6.aspx [Дата звернення 20.09. 22]

36. Документация Python. URL: https://www.python.org/doc/ [Дата звернення 21.09. 22]
37. [Ел. ресурс]. Доступ: https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms?view=windowsdesktop-7.0 [Дата звернення 21.09. 22]
38. [Ел. ресурс]. Доступ: https://learn.microsoft.com/ru-ru/dotnet/api/system.io?view=net-7.0 [Дата звернення 21.09. 22]
39. [Ел. ресурс]. Доступ: https://www.newtonsoft.com/json [Дата звернення 22.09. 22]
40. [Ел. ресурс]. Доступ: https://learn.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word?view=word-pia [Дата звернення 22.09. 22]
41. Документация MS Visual Studio. URL: https://msdn.microsoft.com/library/1461a.aspx [Дата звернення 22.09. 22]
42. Официальный сайт JetBrains Clion. URL: https://www.jetbrains.com/clion/ [Дата звернення 23.09. 22]
43. Официальный сайт Eclipse SDK. URL: https://eclipse.org/sdk/ [Дата звернення 23.09. 22]
44. Новое поколение систем контроля версий [Электронный ресурс] - Доступ к ст.: http://www.techinfo.net.ru/docs/Version_Control_Systems.html. [Дата звернення 23.09. 22]
45. Jim Blandy. Version Control with CVS. [Электронная книга]. URL: http://citforum.ru/programming/digest/cvsintrorus.shtml. [Дата звернення 23.09. 22]
46. Available CVS Alternatives [Электронный ресурс] - Доступ к ст.: http://betterscm.berlios.de/aegis/. [Дата звернення 24.09. 22]
47. Бен Коллинз-Сассман, Брайан У. Фитцпатрик, К. Майкл Пилато. Управление версиями в Subversion [Электронная книга]. URL: http://kharchuk.ru/svn.html [Дата звернення 13.09. 22]
48. Чакон С. Штрауб Б. "Git для профессионального программиста". – Питер, 2016 – 496с.

49. Bryan O'Sullivan. Mercurial: The Definitive Guide. — O'Reilly Media, Inc., 2009. — 288 с.
50. Модульне тестування [Ел. ресурс]. Доступ: https://qalight.ua/baza-znaniy/modulne-testuvannya/ [Дата звернення 24.09. 22]
51. Інтеграційне тестування [Ел. ресурс]. Доступ: https://qalight.ua/baza-znaniy/integratsijne-testuvannya/ [Дата звернення 24.09. 22]
52. Системне тестування [Ел. ресурс]. Доступ: https://qalight.ua/baza-znaniy/sistemne-testuvannya/ [Дата звернення 25.09. 22]
53. Black-box testing [Ел. ресурс]. Доступ: https://en.wikipedia.org/wiki/Black-box_testing [Дата звернення 25.09. 22]
54. Тестирование черным ящиком [Ел. ресурс]. Доступ: https://habr.com/ru/post/462837/ [Дата звернення 25.09. 22]
55. Техника анализа граничных значений [Ел. ресурс]. Доступ: https://qaevolution.ru/testovaya-dokumentaciya/test-dizajn/texnika-analiza-granichnyx-znachenij/ [Дата звернення 25.09. 22]
56. Тестирование Белого ящика [Ел. ресурс]. Доступ: https://ru.wikipedia.org/wiki/Тестирование_белого_ящика [Дата звернення 25.09. 22]

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Б.Є. Боднар

СТРАТЕГІЯ РОЗРОБКИ ПЗ РІЗНОМАНІТНИМ КОЛЕКТИВОМ МАЛО МОТИВОВАНИХ ВИКОНАВЦІВ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01268-01-ЛЗ

Завідувач кафедри КІТ
доц. В. М. Горячкін
_____

Керівник розробки
проф. В.І. Шинкаренко
_____

Виконавець
студент групи ПЗ2121
М. Ю. Нуштаєв
_____

Нормоконтролер
доц. С. А. Волкова
_____

2022

ЗАТВЕРДЖЕНО
1116130.01268-01

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Технічне завдання

1116130.01268-01

Аркушів 26

2022

АНОТАЦІЯ

Документ 1116130.01268-01 «Система дослідження та підвишення мотивації студентів. Технічне завдання» входить до складу програмної документації дипломного проекту.

У документі представлене призначення та область застосування програмного продукту, основні вимоги, стадії та терміни виконання проекту, техніко-економічні показники, що пред’являються до програмного продукту.

ЗМІСТ

ВСТУП

Система дослідження та підвишення мотивації студентів «MotivationAnalysis&Enhancement» являє собою програмний комплекс для виміру рівня мотивації групи студентів та розбиття їх на команди. Система доступна як додаток і може функціонувати в операційній системі Windows.

Через те що кожен студент у групі має різний рівень мотивації і різні навички для командної роботи неможливо створити оптимальні команди просто розбивши людей на групи. Необхідно було створити систему яка б дозволяла досліджувати рівень мотивації студентів групи, відділяти тих хто має настільки низький рівень мотивації що буде заважати роботі команди, та розбитих студентів, що залишились, на команди, які б мали мінімум проблем у роботі, а також визначити можливі проблеми та надати способи їх вирішення.

Причиною замовлення продукту стала відсутність прямих аналогів на українському ринку, а також неможливість адаптації існуючих аналогів через відсутність відкритого доступу до коду.

Сферою застосування розробленого програмного продукту можуть бути всі видавничі та наукові галузі, де потрібно визначати рівень мотивація людей на навчання та ділити їх на групи.

1 ПІДСТАВА ДО РОЗРОБКИ

Підставою для розробки є наказ ректора українського державного університету науки і технологій Пшінька О.М. №173ст від 11.01.2022 р. «Про призначення наукових керівників та затвердження тем дипломних проектів» факультету «Комп’ютерні технології та системи» за спеціальністю 121 «Інженерія програмного забезпечення» по кафедрі «Комп’ютерні інформаційні технології».

Тема проекту «Стратегія розробки ПЗ колективом слабо мотивованих виконавців», керівник дипломного проекту проф. Шинкаренко В. І.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Програмний продукт призначений проводити тестування людей та виявляти їх рівень мотивації навчання і визначати ролі які вони займають в команді. Друга частина потрібна для розбиття групи людей на низку команд с описом їх слабких сторін та шляхами їх вирішення.

2.2 Експлуатаційне призначення

Експлуатаційне призначення програмного засобу:

- використання у ВУЗах для визначення рівня мотивації студента;
- застосовування в навчальних та робочих сферах для створення команд с груп людей та виявлення недоліків команд.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Слід виділити два незалежні модулі програмного комплексу: тестування людини та розбиття групи людей на команди.

Програмний додаток повинен володіти наступним функціоналом:

- проводити тестування людей і визначати їх рівень мотивації та ролі для роботи у команді;
- розбивати групу людей на команди, виявляти слабкості команд та пропанувати шляхи вирішення.

Вхідні дані:

- для модулю першої програми варіанти відповідей від користувача;
- для модулю другої програми номер групи студентів з якою треба розбити на команди.

Вихідні дані:

- для модулю першої програми – файл формату .json з результатами тестування людини;
- для модулю другої програми – файли формату .docx з складами команд, їх проблемами та шляхами покращення роботи команд.

3.2 Вимоги до надійності

Список вимог до надійності програмного засобу в цьому розділі включає лише пункти, що стосуються програмного забезпечення, збої засобів обчислювальної техніки до нього не входять.

Вимоги до надійності програмного продукту наступні:

- наявність архівної копії тексту програми та файлів словнику на зовнішньому носії інформації;
- не більше ніж одна помилка на тисячу операторів;
- контроль коректності вхідної інформації, яка вводиться користувачем під час роботи з програмним продуктом.

3.3 Умови експлуатації

Даний програмний продукт може використовуватись в умовах, які відповідають вимогам.

Для коректного функціонування програмного продукту необхідно дотримання наступних вимог:

- програмний комплекс повинен використовуватись в приміщеннях, призначених для роботи ЕОМ у наступних кліматичних умовах: температура – 21-25 °С, відносна вологість повітря 40-60%;
- кваліфікація персоналу повинна бути на рівні користувача ЕОМ, що має досвід роботи з операційною системою Windows та пройшов підготовку за керівництвом користувача.

3.4 Вимоги до складу та параметрів технічних засобів

Мінімальна конфігурація комп’ютеру для забезпечення роботи програмного продукту:

- комп’ютер з тактовою частотою процесора 2 ГГц і розрядністю 64-біти;
- ОЗП об’ємом не менше 4 ГБ;

- вільний дисковий простір 500 МБ;
- наявність CD/DVD приводу, USB роз’єму для встановлення ПЗ;
- монітор з роздільною здатністю екрану 1024x768 та більше;
- стандартні клавіатура та маніпулятор «миша».

3.5 Вимоги до інформаційної та програмної сумісності

Для функціонування програмного продукту необхідна Windows 7 або більш пізня версія.

3.6 Вимоги до маркування та упаковки

Програма може зберігатися на постійних запам’ятовуючих носіях (CD/DVD-диски, флеш-пристрої). Упаковка продукту повинна забезпечувати захист від механічних пошкоджень та мати маркування, зображене на рис. 3.1:

“MotivationAnalysis&Enhancement”, 1.0

Нуштаєв Микита Юрійович

кафедра КІТ, ДНУЗТ 2022

Рис. 3.1. Штамп продукту

3.7 Вимоги до транспортування та зберігання

Умови транспортування та зберігання повинні забезпечувати захист носія від фізичних пошкоджень і відповідати вимогам .

Термін зберігання програмного продукту необмежений та залежить лише від умов зберігання та встановленого замовником періоду експлуатації.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна включати наступні документи: технічне завдання та робочий проект у складі:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача. Структуризації ;
- керівництво користувача зі встановлення авторства

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів (ГОСТ 19.101-77).

5 ВИЗНАЧЕННЯ ВИТРАТ НА ПРОЕКТУВАННЯ ПРОГРАМИ

Розрахунок витрат на розробку та впровадження програмного продукту

Економічна частина є логічним завершенням дипломного проекту. Для кожного проекту треба дати техніку економічне обґрунтування для того щоб зрозуміти доцільність розробки і впровадження проекту та його прибутковість.

5.1 Трудомісткість розробки програмного продукту

Трудомісткість — показник, що характеризує витрати робочого часу на виробництво певної споживної вартості або на виконання конкретної технологічної операції.[1]

Етапи створення програмного продукту: збір інформації, проектування бази даних, розробка алгоритму проекту, розробка програмного продукту, влагодження програмного продукту, тестування програмного продукту та розробка опису програмного продукту для користувачів чи операторів

Етап збір інформації зайняв 92 години. За цей час було поставлене завдання та проведено його аналіз у результаті якого були зібрані відповідні вхідні дані.

Етап проектування бази даних зайняв 35 годин. Була обрана модель даних та її елементи.

Етап розробка алгоритму проекту тривав 35 годин в результаті чого була створена структура програми, на основі якої розроблявся сам програмний продукт.

Етап розробка програмного продукту самий довгий, він тривав 370 годин. За цей час була написана основна програма, створені необхідні для супроводу файли та налаштовані програми шо обслуговують основну.

Етап влагодження програмного продукту проходив під час усієї розробки продукту та у процесі тестування після завершення написання програми. Процес влагодження зайняв 26 годин.

Під час етапу тестування програмного продукту було проведене інтенсивне тестування продукту задля пошуку і виправлення можливих помилок. На це витрачено 18 годин.

Розробка опису програмного продукту для користувачів чи операторів заключний етап створення програмного продукту, під час нього була створена необхідна документація для використання і керування програмою. Він тривав 10 годин.

Часові затрати на дані етапи наведені в таблиці 5.1:

Таблиця 5.1 Трудомісткість розробки програмного продукту

Назва етапу	Одиниця виміру	Трудомісткість
збір інформації	людино-години	92
проектування бази даних	людино-години	35
розробка алгоритму проекту	людино-години	35
розробка програмного продукту	людино-години	370
влагодження програмного продукту	людино-години	26
тестування програмного продукту	людино-години	18
розробка опису програмного продукту для користувачів чи операторів	людино-години	10
Разом:	людино-години	586

5.2 Розрахунок оплати праці за розробку програмного продукту

Програму розробляє програміст. Оплата труда була розрахована виходячи з трудомісткості розробки програмного продукту. Була взята середня по ринку година тарифна ставка.

Розрахунок годинної тарифної стави проводився на основі даних з сайту DOU[2].

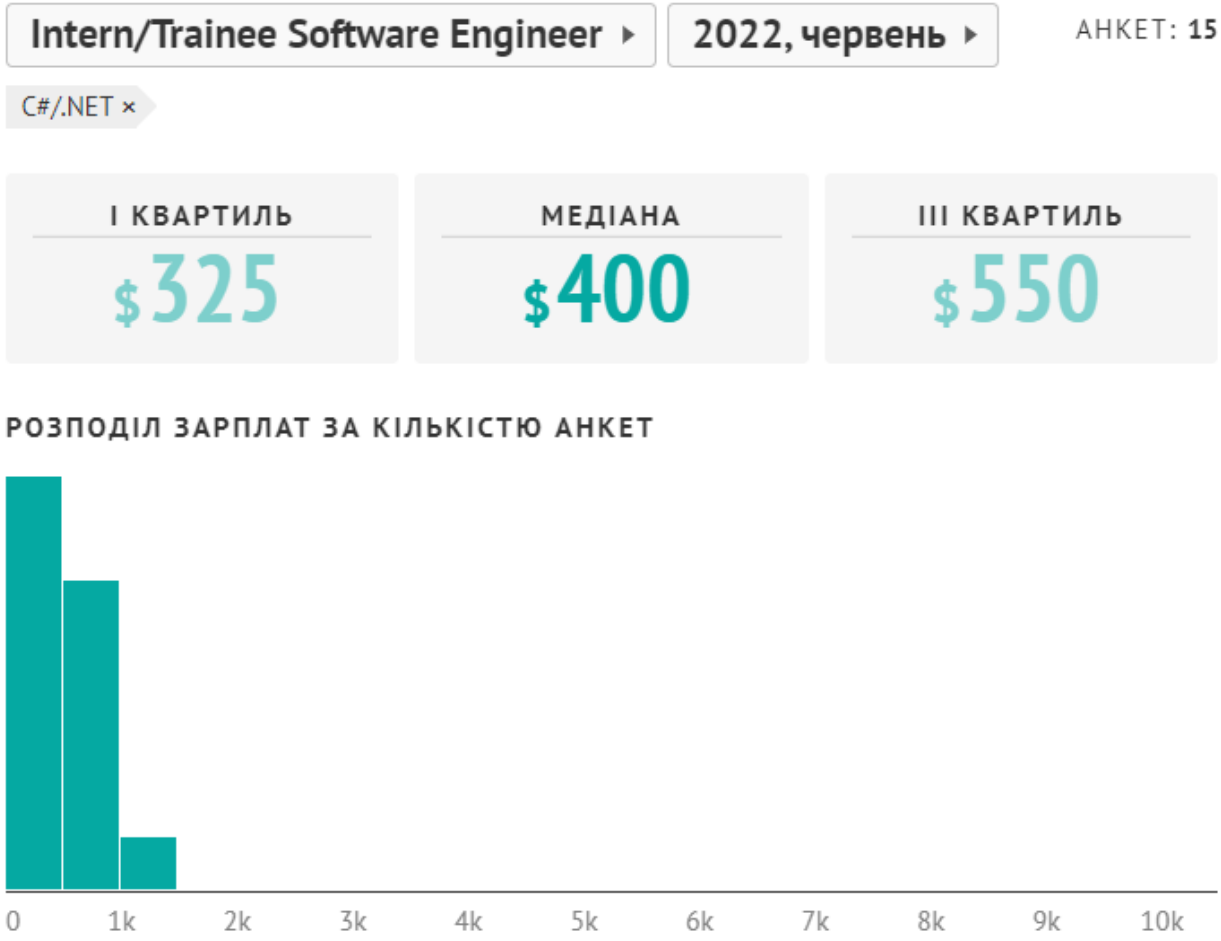


Рисунок 5.1 Розподіл місячних зарплат для програміста рівня junior

Середня зарплата для розробника на мові c# складає 400 доларів на місяць. За курсом Приват Банку на 08.12.2022 курс долара 37.44 [3]. У гривнях середня зарплатня складає 14976 гривень на місяць.

Формула розрахунку погодинної оплати 5.1.

$$ПО = \frac{ЗП}{КРД*КРГД} \quad (5.1)$$

ПО – погодинна оплата;

ЗП – заробітна плата за місяць;

КРД – кількість робочих днів;

КРГД – кількість робочих годин у день.

Дані про КРД були узяті з сайту buhoblik.org.ua [4] і становлять 257 днів на рік, тобто 21 робочий день у місяці.

У програмістів прийнятий стандартний 8 – годинний робочий день.

$$\text{ПО} = \frac{35568}{21*8} = 89$$

Тобто погодинна оплата програміста складає 89 гривні.

Зарплатня по тарифу (Зт) вираховується по формулі 5.2.

$$З_т = V_{ПП} * t_{СТ}\ , \quad (5.2)$$

де $V_{ПП}$ – трудомісткість розробки програмного продукту людино – годин; $t_{СТ}$ – годинна тарифна ставка оператора, грн.

В даному випадку зарплатня складає:

$$З_т = 89 * 586 = 52154 \text{ грн.}$$

Преміальні виплати ($З_{ПРЕМ}$) вираховуються по формулі 5.3.

$$З_{ПРЕМ} = З_Т * К_{ПРЕМ}\ , \quad (5.3)$$

де $З_Т$ – зарплата по тарифу, грн; $К_{ПРЕМ}$ - коефіцієнт (%) преміювання.

$К_{ПРЕМ}$ дорівнює 19% згідно з даними статті сайту хабр [5].

Рисунок 5.2 Частка премії у річному доході за спеціалізацією

Премія буде складати:

$$З_{ПРЕМ} = 52154 * 0{,}19 = 9909 \text{ грн}$$

Додаткова заробітна плата ($З_{ДОП}$) розраховується за формолую 5.4

$$З_{ДОП} = \frac{(З_Т + З_{ПРЕМ}) * К_{ДОП}}{100}, \quad (5.4)$$

де $К_{ДОП}$ - % додаткової заробітної плати (10%).

Сума додаткової заробітної складає:

$$З_{ДОП} = \frac{(52154 + 9909) * 10}{100} = 6206{,}3 \text{ грн}$$

Загальна сума заробітної плати вираховується за допомогою формули 5.5.

$$З = З_{ПРЕМ} + З_Т + З_{ДОП}. \quad (5.5)$$

Сума загальної заробітної плати для даного програмного продукту становить:

$$З = 52154 + 9909 + 6206{,}3 = 68269{,}3 \text{ грн}$$

Відрахування на соціальні потреби вираховуються за допомогою формули 5.6.

Це різні обов'язкові відрахування підприємств до позабюджетних соціальних фондів. До них відносяться відрахування до пенсійного фонду, до

фонду соціального страхування, до фонду зайнятості та до фондів обов'язкового медичного страхування.

$$О = \frac{З * \%\ в\ ЄСВ}{100}, \quad (5.6)$$

ЄСВ - єдиний соціальний внесок на загальнообов'язкове державне соціальне страхування (скор. ЄСВ) — консолідований страховий внесок в Україні, збирання якого здійснюється в системі загальнообов'язкового державного страхування в обов'язковому порядку та на регулярній основі. Його розмір складає 22% [6].

В моєму випадку розмір відчислень сягає:

$$О = \frac{68269{,}3 * 22}{100} = 15019{,}25 \text{ грн}$$

Накладні витрати. Це витрати які не пов’язані безпосередньо з технологічним процесом виготовлення продукції, витрати, понесені на організацію, обслуговування виробництва, реалізацію продукції та управління (наприклад витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше.)

Вони визначаються в процентах (30 – 40%) від суми прямих (формула 5.7).

$$С_{накл} = \frac{(З+О) * 35\%}{100\%}, \quad (5.7)$$

у моєму випадку сума накладних витрат буде сягати:

$$С_{накл} = \frac{(68269{,}3 + 15019{,}25) * 35}{100} = 29151 \text{ грн.}$$

5.3 Розрахунок капітальних вкладень

Необхідно додатково розрахувати витрати на придбання комп’ютера, складових частин для нього, програмного забезпечення.

Перечень необхідного обладнання з цінами на нього для розробко ПЗ наведено в Таблиці 5.2

Таблиця 5.2 Матеріальні витрати

Статі витрат	Кількість шт.	Ціна за одиницю, грн	Усього, грн
Персональний комп’ютер Acer Aspire 3 A315-34 (NX.HE3EU.040)[7]	1	20 260	20 260
Комп’ютерна миша	1	500	500
Разом			20760

Модель комп’ютера обиралась ґрунтуючись на необхідних системних вимогах використовуваного ПЗ. Комп’ютерна миша необхідна для підвищення зручності та швидкості роботи з персональним комп’ютером.

Крім цього існують витрати на експлуатацію придбаної техніки. Це щорічні кошти які визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру.

Перелік експлуатаційних витрат:

- вартість витратних матеріалів;
- витрати на ремонт;
- оренда приміщення;
- додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;
- амортизаційні витрати на персональний комп'ютер і програмне забезпечення;
- витрати на електроенергію

Вартість витратних матеріалів буле сягати приблизно 10% від вартості комп’ютеру. Термін експлуатації комп’ютеру 5 років. Вирахуємо за цими даними вартість витратних матеріалів використовуючи формулу 5.8

$$С_{вм} = В_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%} \quad (5.8)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 20260 \cdot \frac{73}{5 \cdot 365} \cdot \frac{10}{100} = 81 \text{ грн.}$$

Згідно статистики витрати на запчастини для комп'ютера складають 10% від його вартості за термін його експлуатації, для розрахунку використовують формулу 5.9.

$$С_з = В_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (5.9)$$

У моєму випадку сума буде така:

$$C_з = 20260 \cdot \frac{73}{5 \cdot 365} \cdot \frac{10}{100} = 81 \text{ грн.}$$

Оренда офісу на 1 людину коштує 3120 гривень на місяць. [8]

Тривалість розробки 73 дні, тобто 2 місяці 13 днів. Т.я. офіс орендується на місяць то витрати на нього будуть складати 9360 грн.

Додаткові витрати входять у витрати на знайдений офіс, тобто витрати за 3 і 4 пунктами сумарно 9360 грн.

Розрахунок амортизаційних витрат виконується по формулі 5.10

$$А = \frac{В_О * Н_А}{100} \text{ грн}, \quad (5.10)$$

$В_О$ – вартість обладнання,

$Н_А$ – норма амортизації.

Норма амортизації дорівнює вартості комп'ютера за N років експлуатації. [9] Це дорівнює терміну морального старіння обчислювальної техніки. Тобто амортизація комп'ютера дорівнює 100% від вартості комп'ютера

за 3 роки. Для комп'ютерної миші такі ж саме данні, тобто вартість амортизації комп'ютерної миші за 3 роки складе 100% від її вартості.

За цими даними розрахуємо норму амортизації за період часу витраченого на створення дипломного додатку використовуючи формулу 5.11.

$$Н_А = \frac{Д_р}{Д_{макс}} * 100\%, \quad (5.11)$$

де Др= кількість днів витрачена на створення продукту,

$Д_{макс}$ = кількість днів за які норма амортизації складе 100%

Для мого випадку $Н_А$ буде дорівнювати:

$$Н_А = \frac{73}{1095} * 100 = 6{,}7\%$$

Витрати на амортизацію для комп'ютеру:

$$А = \frac{20\ 260 * 6{,}7}{100} = 1357{,}42 \text{ грн}$$

Витрати на амортизацію для комп'ютерної миші:

$$А = \frac{500 * 6{,}7}{100} = 33{,}5 \text{ грн.}$$

У Таблиці 5.3 наведені розміри амортизаційних відрахувань.

Таблиця 5.3. Амортизаційні відрахування

Найменування обладнання	Вартість обладнання, грн.	Норма амортизації %	Сума амортизації, грн.
Персональний комп'ютер Acer Aspire 3 A315-34 (NX.HE3EU.040)	20 260	6,7	1357,42
Комп'ютерна миша	500	6,7	33,5
Разом			

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.4.

Таблиця 5.4 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	13800	http://mtsoft.kiev.ua/product/windows-10-professional	377,2
Visual Studio 2019 Community	0	https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes	0
Разом:	13800		377,2

Енергетичні витрати складаються із витрат на освітлення приміщення, де проводилася розробка програмного продукту, а також витрат енергії на роботу комп'ютера.

Електрична потужність обладнання, що використовується 150 ватт в годину.[10] Електрична потужність ламп для освітлення приміщення 0,016 кВт, необхідна кількість: 4 шт. [11]

Сумарна кількість витраченої енергії під час створення проекту розраховується по формулі 5.12.

$$С_є = (Є_к + Є_л * К) * Г_с, \quad (5.12)$$

$Є_к$ - Електрична потужність обладнання,

$Є_л$ - Електрична потужність ламп

К – кількість ламп

$Г_с$ – загальна кількість годин праці

В моєму випадку загальна кількість витраченої енергії дорівнює:

$$С_є = (150 + 16 * 4) * 586 = 125404 \text{ Вт}$$

1 МВт енергії коштує 821,57 грн. [12] Енергетичні витрати за проект дорівнюють(125404/1000000)*821,57 = 103,03 грн.

Розрахунки витрат на увесь проект

В таблиці 5.5 наведено усі експлуатаційні витрати на ПК і ПЗ.

Таблиця 5.5 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	103,03
Вартість витратних матеріалів	81
Витрати на ремонт	81
Амортизація персонального комп’ютера	1390,92
Амортизація програмного забезпечення	377
Оренда приміщення	9360
Єлектроенергія	180,58
Разом	11573,53

Таким чином, витрати на створення програмного продукту можно порахувати по формулі 5.13.

$$C_{\text{розробки}} = З + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}} + C_{\text{комп}} \quad (5.13)$$

$$C_{\text{розробки}} = 22500 + 4950 + 9607 + 14051 = 51108 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.6.

Таблиця 5.6 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	68269,3
Відрахування на соціальні потреби	15019,25
Накладні витрати	29151
Експлуатаційні витрати	11573,53
Витрати на комп’ютер	36328,12
Всього	160341,2

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 160341,2 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програмного продукту представлені у табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

Стадії розробки	Етапи розробки	Терміни виконання
1. Технічне завдання (ТЗ)	Функціональне та експлуатаційне призначення	23.09.22 – 24.09.22
	Вхідні та вихідні дані	24.09.22 – 27.09.22
	Вимоги до програмної документації	27.09.22 – 29.09.22
	Техніко-економічні показники	29.09.22 – 01.10.22
	Узгодження та затвердження ТЗ	01.10.22 – 04.10.22
2. Робочий проект	Розробка та програмування функціоналу програми	05.10.22 – 19.11.22
	Розробка і реалізація інтерфейсу користувача	19.11.22 – 26.11.22
	Налагодження програми	26.11.22 – 01.12.22
	Розробка, узгодження та затвердження програмної документації відповідно до вимог ГОСТ 19.101-77	01.12.22 – 03.12.22
	Проведення випробувань і корегування програми та документації за результатами випробувань	03.12.22 – 07.12.22
3. Впровадження	Підготовка і передача програми та програмної документації замовнику	08.12.22 – 10.12.22

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмному продукті та його специфікації. Контроль виконання роботи забезпечується головним керівником розробки проф. Шинкаренко В. І.

Прийом програмного продукту здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1 Трудомісткість [Ел. ресурс]. Доступ: https://uk.wikipedia.org/wiki/Трудомісткість [Дата звернення 06.12.22]

2 Зарплатні програмістів [Ел. ресурс]. Доступ: https://jobs.dou.ua/salaries/?period=2022-06&position=Intern/Trainee%20SE [Дата звернення 06.12.22]

3 Курс долара [Ел. ресурс]. Доступ: https://privatbank.ua/ru [Дата звернення 06.12.22]

4 Робочі дні у 2022 році [Ел. ресурс]. Доступ: https://www.buhoblik.org.ua/kadry-zarplata/vremya/1664-rabochie-dni.html [Дата звернення 06.12.22]

5 Премии, льготы и бонусы в IT: результаты исследования Хабр Карьеры [Ел. ресурс]. Доступ: https://habr.com/ru/company/habr_career/blog/507258/ [Дата звернення 06.12.22]

6 Единый социальный взнос [Ел. ресурс]. Доступ: https://index.minfin.com.ua/labour/social/ [Дата звернення 06.12.22]

7 Ноутбук [Ел. ресурс]. Доступ: https://rozetka.com.ua/ua/360461643/p360461643/ [Дата звернення 06.12.22]

8 Об’ява про сдачу квартири [Ел. ресурс]. Доступ: https://www.olx.ua/d/uk/obyavlenie/sdam-ofis-3-km-ot-tsentra-ulitsa-karuna-75-IDOAr39.html [Дата звернення 06.12.22]

9 Срок амортизації комп’ютера [Ел. ресурс]. Доступ: https://spmag.ru/articles/srok-amortizacii-kompyutera [Дата звернення 06.12.22]

10 Сколько электроэнергии расходует компьютер? [Ел. ресурс]. Доступ: https://mobileplanet.ua/blog/skolko-elektroenergii-rashoduet-kompyuter-308 [Дата звернення 06.12.22]

11 Електрична потужність ламп [Ел. ресурс]. Доступ: https://www.sima-land.ru/o-kompanii/novosti-kompanii/4240/ [Дата звернення 06.12.22]

12 Тарифы на электроэнергию для предприятий [Ел. ресурс]. Доступ: https://index.minfin.com.ua/tariff/electric/prom/ [Дата звернення 06.12.22]

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор українського державного університету науки і технологій
В.А.Радкевич

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01268-01-ЛЗ

Завідувач кафедри КІТ
доц. В. М. Горячкін
_____

Керівник розробки
проф. В. І. Шинкаренко
_____

Виконавець
студент групи ПЗ2121
М. Ю. Нушаєв
_____

Нормоконтролер
доц. С. А. Волкова
_____

2022

ЗАТВЕРДЖЕНО
1116130.01268-01

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Специфікація

1116130.01268-01

Аркушів 2

2022

Позначення	Найменування	Примітка
Документація		
1116130.01268-01-ЛЗ	Лист затвердження	
1116130.01268-01 13 01-ЛЗ	Опис програми	
1116130.01268-01 І3 01	Керівництво користувача	
1116130.01268-01 12 01	Текст програми	

ЗАТВЕРДЖЕНО
1116130.01268-01-ЛЗ

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Опис програми

1116130.01268-01 13 01

Аркушів 15

2022

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Розроблений додаток складається з двох частин. Перша частина дозволяє протестувати студентів у групі та дізнатися і їх рівень мотивації до навчання і їх командні навички. Програма виводить результат в файл де описано розшифровані результати всіх тестів та фінальна оцінка мотивації студента. Друга частина слугують для розбиття людей на команди. Спочатку за результатами попереднього тестування усі люди які мають низьку мотивацію і погано працюють у командах будуть виключені з цього розбиття, адже є висока імовірність того, що вони не будуть гарно виконувати завдання, тим самим сильно погіршить моральний дух команди і мотивацію інших студентів. Далі програма розбиває студентів що залишилися на команди наступним чином: спочатку формуються групи студентів по 3 або 4 людини, цей формат команд є оптимальним як для студентської роботи(учнів не є доцільним розбивати на великі команди, адже лідеру їх команди буде тяжко робити свої обов’язки) так і у міжнародній практиці при створенні професіональних команд для вирішення різних задач(доцільним є створення команд від 3 до 7 чоловік згідно дослідження Белбіну). Пріоритетом при створенні команд є найбільша різноманітність ролей у командах, що дозволить уникнути максимальної кількості проблем. Після цього додаток аналізує проблеми які можуть виникнути у створених командах спираючись на відсутні у цих командах ролі. Результатом роботи є файл у якому наведено перелік студентів у команді ролі яких нема у команді та наведено низка порад для покращення командної роботи.

Програма функціонує в середовищі операційної системи Windows починаючи з версії 7 та вище. Програма розроблена на мові C# у середовищі Visual Studio 2019.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмний продукт призначений проводити тестування людей та виявляти їх рівень мотивації навчання і визначати ролі які вони займають в команді. Друга частина потрібна для розбиття групи людей на низку команд с описом їх слабких сторін та шляхами їх вирішення.

Сферою застосування розробленого програмного продукту можуть бути всі галузі, де потрібно покращити чи створити команди та навчальні галузі де треба дізнатися рівень мотивації студентів.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1. Алгоритм програми

На рис. 3.1 представлено алгоритм роботи програми для тестування, на рис. 3.2 алгоритм програми для розбиття на команди.

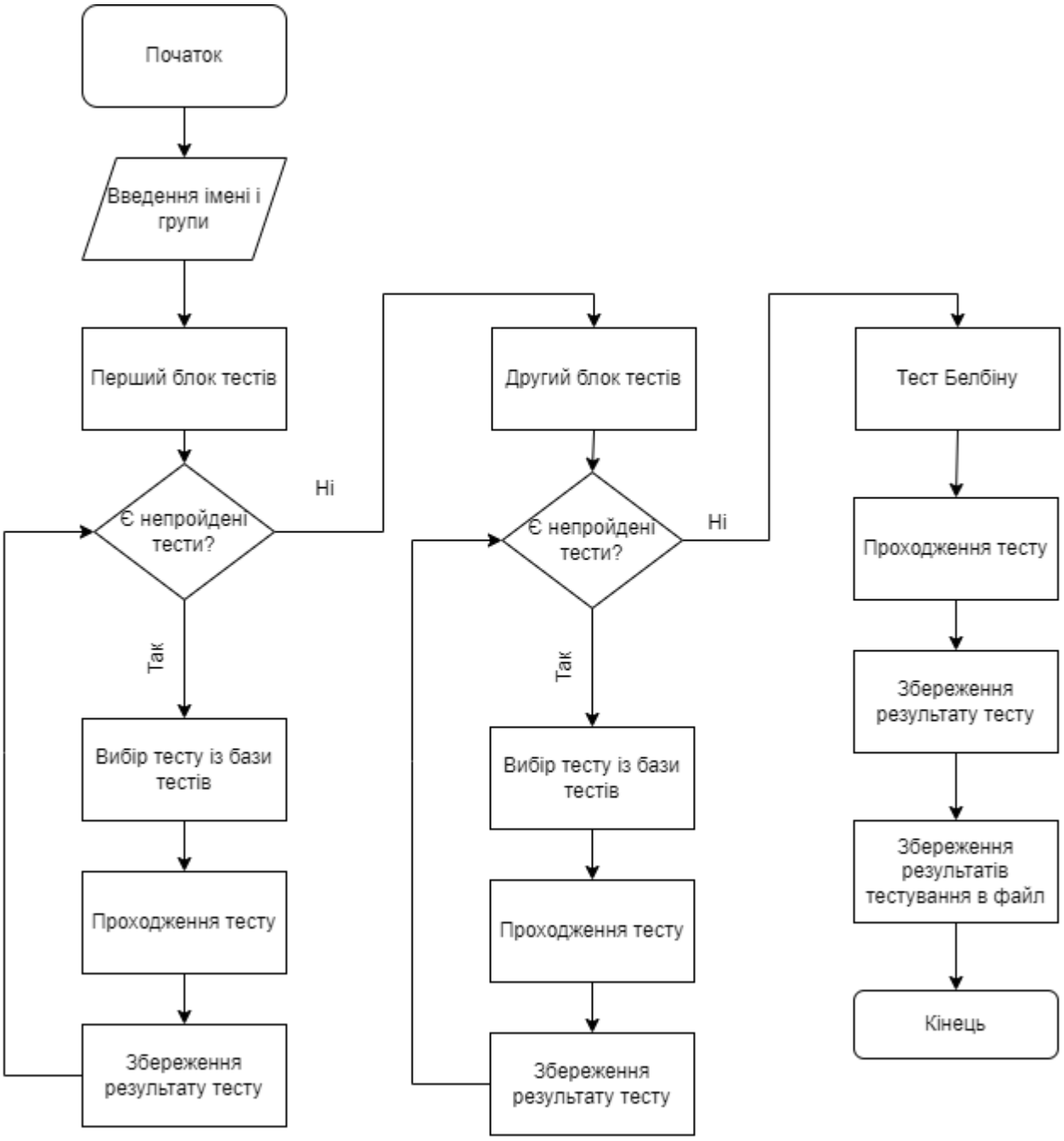


Рис. 3.1 Алгоритм програми для тестування студентів

Рис. 3.2 Алгоритм програми яка розбиває групу людей на команди

3.2 Використані методи

Newtonsoft.Json – спеціальна бібліотека з методами які дозволяють працювати з файлами формату JSON.

Microsoft.Office.Interop.Word - спеціальна бібліотека з методами які дозволяють працювати з файлами формату .docx.

3.3 Модулі програми

Personal tests - відповідає за проведення тестування людей.

Teams division – відповідає за створення і аналіз створених команд.

3.4 Зв’язки програми з іншими програмами

Зв’язків з іншими програмами немає.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Технічні засоби, які використовуються при роботі програми:

- ПК, під управлінням 64-розрядної ОС Windows;
- процесор з тактовою частотою не нижче 2 ГГц;
- оперативна пам’ять не менше 4 ГБ;
- простір на жорсткому диску не менше 500 ГБ;
- маніпулятор миша;
- клавіатура;
- наявність CD/DVD приводу, USB роз’єму або LAN-адаптеру для завантаження необхідного ПЗ в ОС;
- монітор з роздільною здатністю екрану 1024x768 та більше, з підтримкою не менш ніж 256 кольорів.

5 ВИКЛИК І ЗАВАНТАЖЕННЯ

Перед використанням програми необхідно виконати наступні дії:

- виконайте запуск виконавчого файлу програми Tests.exe для студентів;
- виконайте запуск виконавчого файлу програми TeamsDivision.exe.

Після проходження тесту треба занести інформацію з отриманого файлу в БД.

Після формування команд з’являться файли з інформацією для кожної команди.

6 ВХІДНІ ДАНІ

Вхідними даними програми, що розробляється, є:

- варіанти відповідей від користувача;
- номер групи студентів з якою треба розбити на команди.

7 ВИХІДНІ ДАНІ

Вихідними даними програми, що розробляється, є:

- для модулю першої програми – файл формату .json з результатами тестування людини;
- для другого модулю – файли формату .docx з складами команд, їх проблемами та шляхами покращення роботи команд.

8 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

8.1. Опис станів програми

Після запуску програми «Personal tests» (1) (Tests.exe) на екрані з’являється стартовий екран де користувач повинен ввести своє ім’я і номер групи. Далі він перейде до проходження тестів. У кінці роботи додатку користувач вибирає місце куди зберігає файл з результатами свого тестування.

Після запуску програми «Teams division» (2) (TeamsDivision.exe) на екрані з’являється стартовий екран де користувач повинен ввести номер групи и натиснути кнопку сформувати групи. Після цього він може працювати з отриманими файлами.

Стани програми «Personal tests» наведено у табл. 8.1, TeamsDivision.exe у табл. 8.2.

Таблиця 8.1. Стани програми

№ стану	Назва стану	Опис стану	Рекомендовані дії
1	Запуск додатку	Додаток запускається	Очікування стартового меню
2	Вводячи особистих даних	Користувач вводить необхідні данні	Вводити данні с клавіатури та миші
3	Проходження тестів	Користувач проходить тести вводячи відповідь	Вибрати одну із запропонованих відповідей чи ввести самому
4	Збереження результату	Користувач вводить шлях до папки куди треба зберегти результат тесту	Користувач вводить шлях до папки чи вибирає її з переліку

Таблиця 8.2. Стани програми

№ стану	Назва стану	Опис стану	Рекомендовані дії
1	Запуск додатку	Додаток запускається	Очікування стартового меню
2	Введення номеру групи	Користувач вводить номер команди	Вводити команди с клавіатури та миші
3	Формування команд	Додаток формує файли формату .docx з інформацією про команди	Очікувати коли файли створяться

8.2 Опис переходів між станами програми

Опис переходів між станами програми представлено на рисунку 8.1.

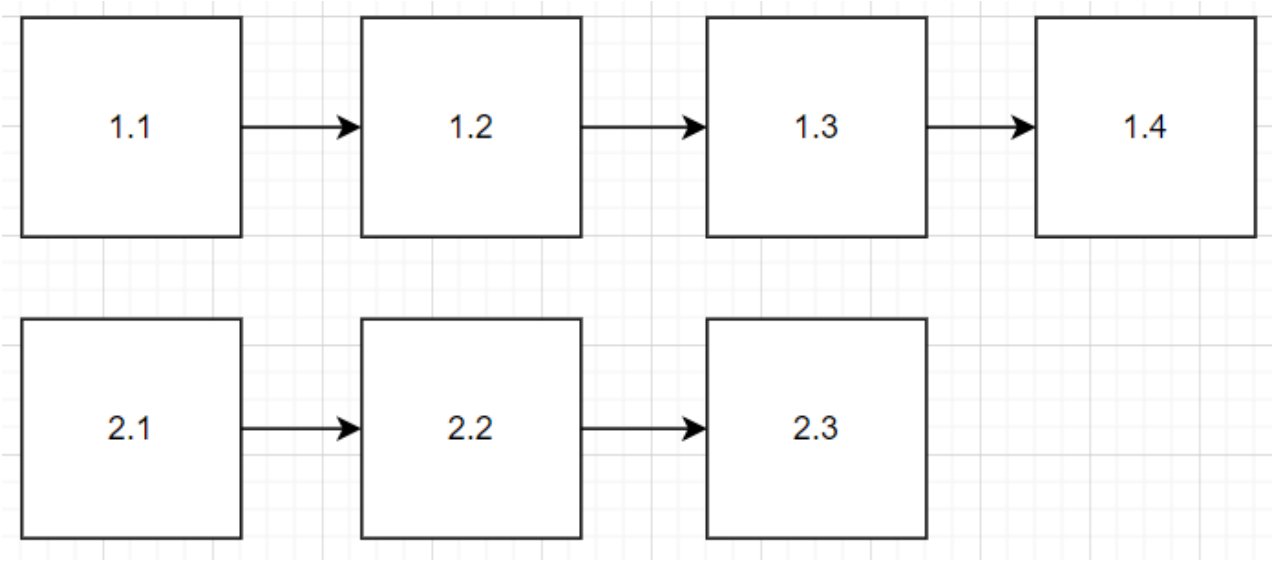


Рисунок. 8.1. Схема переходів між станами програми

8.3 Опис керування діалогом

Діалог між користувачем та додатком відбувається за допомогою інтерфейсу, а саме користувач взаємодіє за допомогою кнопок.

8.4 Формування екранів

Формування екранів додатку відбувається під час діалогу між користувачем та інтерфейсом користувача.

Для завершення роботи з програмою користувач повинен закрити додаток натиснувши на кнопку “  ”.

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Перед початком роботи користувач повинен впевнитись в безпечності користуванням ПК або ноутбуком.

Для роботи з програмою «Personal tests» користувач повинен ввести своє ім’я та номер групи після чого натиснути кнопку ”Почати тестування”. Під час проходження тесту користувач повинен вибрати варіант відповіді з якою він згоден. Під час деяких тестів інструкція до їх проходження буде зображена на екрані. Після відповіді на всі питання з’явиться кнопка “Завершити тест” після чого користувач або перейде до наступного тесту, або, якщо тест останній до меню вибору місця куди треба зберегти файл з відповідями.

Для роботи з програмою «Teams division» користувач повинен ввести номер групи та натиснути кнопку “Створити групи”. Після цього створиться низка файлів кожен з яких буде мати інформацію про одну групу.

10 ПОВІДОМЛЕННЯ

В табл. 10.1 представлені повідомлення користувачу, що можуть з’явитися у процесі роботи програми.

Таблиця 10.1 – Повідомлення, що з’являються в процесі роботи програми

Текст повідомлення	Кому призначене	Опис ситуації	Рекомендовані дії
Ви не вибрали відповідь!	Користувач	Користувач намагається перейти до наступного питання не обравши відповідь на попереднє	Обрати відповідь
Сума усіх відповідей не дорівнює 10 чи десь стоїть 1!	Користувач	Користувач неправильно відповів у тесті Белбіна	Рекомендовано перечитати правила тесту та правильно ввести відповідь
Ви не ввели номер групи!	Користувач	Користувач не ввів номер групи яку треба розбити на команди	Треба ввести номер групи

ЗАТВЕРДЖЕНО
1116130.01268-01-ЛЗ

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Керівництво користувача.

1116130.01268-01 І3 01

Аркушів 7

2022

ЗМІСТ

ВСТУП

Програмний комплекс «MotivationAnalysis&Enhancement» виконує аналіз і підвищує мотивацію студентів груп. Він написаний на мові C#, для зберігання файлів з варіантами відповідей використовуються формат файлу .json. Зберігання даних тестування виконується базою даних Microsoft SQL server.

Для використання програми, користувачу необхідно вміти працювати в операційній системі Windows, мати навички користування мишею і клавіатурою, розуміти принцип роботи текстових редакторів для роботи с результуючими файлами.

Перед роботою необхідно ознайомитись з наступними документами:

- опис програми;
- керівництво користувача.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Програмний комплекс призначений для полегшення процесу визначення рівня навчальної мотивації студентів та для розбиття групи на команду та аналізу і вирішення проблем утворених команд.

Для виконання програми необхідна наявність ПК або ноутбуку, на якому встановлена Windows 7 або новіша версія. Конфігурація ПК для функціонування програмного продукту повинна бути наступна:

- ПК, під управлінням 64-розрядної ОС Windows;
- процесор з тактовою частотою не нижче 2 ГГц;
- оперативна пам’ять не менше 6 ГБ;
- простір на жорсткому диску не менше 1 ГБ;
- маніпулятор миша;
- клавіатура;
- наявність CD/DVD приводу, USB роз’єму або LAN-адаптеру для завантаження необхідного ПЗ в ОС;
- монітор з роздільною здатністю екрану 1024x768 та більше, з підтримкою не менш ніж 256 кольорів.

Для користування програмою користувач повинен вміти працювати в операційній системі сімейства Windows та з редактором текстів Microsoft Word.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Склад і зміст дистрибутивного носія даних

Дистрибутив програмного продукту містить у собі:

- програмний додаток Tests.exe;
- програмний додаток TeamsDivision.exe.

2.2 Порядок завантаження даних і програм

Після запуску програмного додатку Tests.exe необхідно відправити людині яка формує базу даних. Перед стартом роботи TeamsDivision.exe треба вести необхідні дані в відповідну БД.

2.3 Порядок перевірки працездатності

Для перевірки працездатності програми необхідно провести тестовий запуск програми з наступним переліком дій:

- запустити Tests.exe, пройти усі запропоновані тести та вибрати папку куди зберегти результат;
- запустити TeamsDivision.exe;
- після закінчення виконання програми оцінити створені, під час роботи програм, файли.

3 ОПИС ОПЕРАЦІЇ

Далі приведено список операцій та послідовність дій для їх виконання.

3.1 Функція управління запуском тестів

Функція виконує черговий запуск тестів так, щоб людина пройшла всі тести один і тільки один раз. Вона починає свою роботу після того як студент завантажить додаток Tests.exe веде своє ім'я та номер групи і натиснути кнопку почати тестування.

Після закінчення роботи функції буде сформований файл з результатом тестування.

3.2 Функція розбиття на команди

Функція розбиття на команди має у собі спеціальні алгоритм який дозволяє створити групи людей такий набір команд які були максимальну ефективність.

Для запуску цієї функції треба запустити додаток TeamsDivision.exe. Після того як користувач введе номер групи а програма сформує групу студентів почнеться розбиття їх на команди. Після закінчення роботи функції вона виключи функцію яка сформує файл формату .docx для виводу результату роботи програми.

4. АВАРІЙНІ СИТУАЦІЇ

При повторюваних відмовах технічних засобів, які унеможливлюють роботу з програмою, необхідно звернутись до системного адміністратора.

У випадку одиничного збою треба виконати перезавантаження програмного засобу або системи. При повторних збоях звертатися до системного адміністратора.

У разі виявлення помилок у даних необхідно пересвідчитися у правильності алгоритму користування програмою, перевірити вхідні дані, після чого перезапустити програму або систему, якщо проблема була підтверджена.

У разі виявлення некоректного виводу зображення на електронний вимкнути персональний комп’ютер та від’єднати його та периферійні пристрої від мережі, перевірити контакт кабелю дисплею від комп’ютеру до монітору. Якщо проблема не була усунута, викликати адміністратора.

У разі відмови магнітних або твердотільних носіїв треба встановити програму з диску на справний магнітний або твердотільних носій.

У випадках виявлення несанкціонованого втручання в дані одразу повідомити про це системного адміністратора.

В інших аварійних ситуаціях повідомити системного адміністратора про аварійну ситуацію.

ЗАТВЕРДЖЕНО

1116130.01268-01 12 01-ЛЗ

СИСТЕМА ДОСЛІДЖЕННЯ ТА ПІДВИШЕННЯ МОТИВАЦІЇ СТУДЕНТІВ

Текст програми

1116130.01268-01 12 01

Аркушів 26

2022

Зміст

1 СТРУКТУРА ПРОГРАМИ

Комплекс програмних засобів «Motivation Analysis & Enhancement» складається з двох модулів:

- Personal tests – відповідає за проведння тестування;
- Teams division – відповідає за розділ групи на команди та формування файлу.

Модуль Personal tests складається з таких файлів написаних на мові C#, як:

Authorization.cs - клас необхідний для внесення даних щодо випробуваного. Він має у собі графічні форми в яку користувач вводити таку інформацію: своє ім'я та свою групу.

CommandWorkAbility.cs - клас який створює графічне представлення для тестування студентів. Він поєднує у собі перелік тестів. Слугує для таких тестів: тест Юнгу, тести на командну роботу, мотивація до навчання в вузі, мотивація досягнення. Створює форму на яку виводиться питання, варіанти відповідей на запитання та декількох кнопок які дозволяють пересуватися по тесту (перейти до наступного, повернутися до попереднього питання, завершити тест і перейти до наступного тесту).

MotivationLearningActivites.cs - другий клас для графічного представлення тесту. Він слугує для праці з такими тестами: мотивація учбової діяльності, мотивація вибору професії, задоволення від навчання. Цей клас також адаптується під кожний тест.

BelbinTest.cs - клас необхідний для графічного представлення тест Белбіна. Тест Белбіна сильно відрізняється від попередніх тестів тому його реалізацію було винесено окремо в цей клас.

calculetedResult.cs - клас необхідний для підрахунку результатів тестування. В результаті створюється масив даних з характеристиками щодо ролі у командній роботі для студенту.

formController.cs - це клас який відповідає за виклик форм з тестами. У ньому збережені методи які створюють тий чи інший тест передаючи необхідні дані в клас графічного представлення тестів.

testResultF.cs - клас для збереження даних після тестування.

testsResult.cs - клас для збереження даних після тестування.

Модуль Teams division складається з наступних файлів:

teacherControllForm.cs - клас необхідний для другого модуля. Представляє собою графічну форму яка приймає номер групи студентів з якої треба вибрати та розподілити на команди. Викликає клас formController.cs та передає йому номер групи.

wordConnect.cs - клас для з'єднання з файлом формату docx.

combinatedInWord.cs - цей клас записує результати розділення групи на команди у файл формату .docx.

createCommand.cs - клас який розбиває групу на команди, після чого викликається клас wordConnect.cs для з'єднання з файлом формату .docx. Викликає combinatedInWord.cs для виводу результату у файл.

createGroup.cs - потрібен для створення структури даних яка відображає групу студентів у яких є характеристики у вигляді їх ролей у команді. Після створення цієї структури викликає клас createCommand.cs.

dataClass.cs - клас у якому зберігаються дані.

teamWeaknesses.cs - клас який оцінює створені команди та шукаю у них недоліки, після чого надає варіанти для їх вирішення.

DBconnections.cs - клас у якому прописані методи та задано шлях до бази даних. Він викликається іншими класами коли їм потрібно відкрити чи закрити з’єднання до бази даних.

DBwrite.cs - клас який потрібний для запису даних у базу даних. Він викликаються іншими класами та має у собі спеціальні методи які змінюють чи вносять в базу даних ти чи інші дані. Це дані про студента та результати його тестування. Цей клас викликає DBconnections.cs задля відкриття з’єднання до бази даних.

2 ТЕКСТ ПРОГРАМИ

Модуль Authorization.cs

```
    public partial class Authorization : Form
    {
        public Authorization()
        {
            InitializeComponent();
        }
        private                          void
registrationB_Click(object sender, EventArgs
e)
        {
            if  (nameUser.Text  !=  ""  ||
userGroup.Text != "")
            {
                testsResult.name          =
nameUser.Text;
                testsResult.group = 0;
                this.Hide();
                formController
formController = new formController();

formController.startFormControll();
            }
            else
            {
                errorLable.Text  =  "Введіть
необхідну інформацію про себе!";
            }
        }
```

Модуль testsResult.cs

```
class testsResult
    {
        public static string name { get; set;
}
        public static int group { get; set;
}
        public  static  int  youngtest  {  get;
set; }
        public  static  int  OvKtest1  {  get;
set; }
        public  static  int  OvKtest2  {  get;
set; }
        public  static  int  OvKtest3  {  get;
set; }
        public         static         double
AcquisitionKnowledge { get; set; }
        public         static         double
MasteryProfession { get; set; }
        public static double GettingDiploma
{ get; set; }
        public          static          int
AchievementMotivationLevel { get; set; }
        public        static        double[]
Motivation_for_learning_activities  {  get;
set; }
        public          static          int[]
MotivesForChoosingProfession { get; set; }
        public        static        double[]
satisfactionLearningActivities { get; set; }
        public  static  int  []  belbinTest  {
get; set; }
    }
```

Модуль formController.cs

```
class formController
    {
        public void writeInJson()
        {
            string path= "D:";
            using  (var  path_dialog  =  new
FolderBrowserDialog())
            {
                if (path_dialog.ShowDialog()
== System.Windows.Forms.DialogResult.OK)
                {
                    path                    =
path_dialog.SelectedPath;
                };
            }
            string name = "TestRezult.txt";
            testResultF  finalResult  =  new
testResultF();
            string           json           =
JsonConvert.SerializeObject(finalResult);
            using (StreamWriter writer = new
StreamWriter(path+"\\"+name, false))
            {
                writer.WriteLineAsync(json);
            }
        }
        public void startFormControll()
        {
            commandWorkAbilityStart();

motivationLearningActivitesStart();
            writeInJson();
            belbinTestStart();
            Environment.Exit(0);
        }
        public                              void
commandWorkAbilityStart()
        {
            string[]  listOfQuestionFile  =
new     string[]     {     "youngtest.json",
```

```
"OvKtest1.json",            "OvKtest2.json",
"OvKtest3.json",
"studying_the_motivation_of_studying_at_a_un
iversity.json",
"NeedForAchievementScale.json" };
            dataClass.currentNumberTest = 0;
            dataClass.fileWithQuestionName =
listOfQuestionFile;
            YoungTest f = new YoungTest();
            f.ShowDialog();
        }
        public void belbinTestStart()
        {
            BelbinTest f = new BelbinTest();
            f.ShowDialog();
        }
        public                              void
motivationLearningActivitesStart()
        {
            string[]   listOfQuestionFile   =
new                string[]                {
"Motivation_for_learning_activities.json",
"MotivesForChoosingProfession.json",
"satisfactionLearningActivities.json" };
            dataClass.currentNumberTest = 0;
            dataClass.fileWithQuestionName =
listOfQuestionFile;
            MotivationLearningActivites  f  =
new MotivationLearningActivites();
            f.ShowDialog();
        }
}
```

Модуль dataClass.cs

```
class dataClass
    {
        public  static  string  group  {  get;
set; }
        public        static        string[]
fileWithQuestionName { get; set; }
        public  static  int  questionAmount  {
get; set; }
        public static bool thirdAnswer { get;
set; }
        public  static  int  currentNumberTest
{ get; set; }
        public static int[] testScores { get;
set; }
        public static int[,] mas { get; set;
}
        public     static     List<string[]>
dataPeople { get; set; }
    }
```

Модуль BelbinTest.cs

```
public partial class BelbinTest : Form
    {
        int questionNumber;
        int[,] answerPoint = new int[7,8];
        double[]  answerTranscription  =  new
double[8];
        public BelbinTest()
        {
            InitializeComponent();
        }

        private void checkVisible()
        {
            if(questionNumber!=7           &&
questionNumber!=1)
            {
                NextQuestion.Visible = true;
                PreviousQuestion.Visible   =
true;
            }
            if (questionNumber == 7)
            {
                NextQuestion.Visible       =
false;
            }
            if (questionNumber == 1)
            {
                PreviousQuestion.Visible   =
false;
            }
        }
        private void createAnswerMas()
        {
            foreach      (var      pb      in
this.Controls.OfType<TextBox>())
            {
                if(pb.Text==""){  pb.Text  =
"0"; }
            }
            answerPoint[questionNumber  -  1,
0] = Convert.ToInt32(pointsV1.Text);
            answerPoint[questionNumber  -  1,
1] = Convert.ToInt32(pointsV2.Text);
            answerPoint[questionNumber  -  1,
2] = Convert.ToInt32(pointsV3.Text);
            answerPoint[questionNumber  -  1,
3] = Convert.ToInt32(pointsV4.Text);
            answerPoint[questionNumber  -  1,
4] = Convert.ToInt32(pointsV5.Text);
            answerPoint[questionNumber  -  1,
5] = Convert.ToInt32(pointsV6.Text);
            answerPoint[questionNumber  -  1,
6] = Convert.ToInt32(pointsV7.Text);
            answerPoint[questionNumber  -  1,
7] = Convert.ToInt32(pointsV8.Text);
        }
        private bool checkCorrectAnswer()
        {
            createAnswerMas();
            int answerSum=0;
```

```
            for (int i = 0; i < 8; i++)
            {
                answerSum                   +=
answerPoint[questionNumber - 1, i];

if(answerPoint[questionNumber - 1, i] == 1)
{ return false; }
            }
            if (answerSum == 10)
            {
                return true;
            }
            else { return false; }

        }
        private void changeTB()
        {
            int a = 7;
            foreach      (var      pb      in
this.Controls.OfType<TextBox>())
            {
                pb.Text                     =
answerPoint[questionNumber       -       1,
a].ToString();
                a--;
            }
        }
        private                          void
NextQuestion_Click(object  sender,  EventArgs
e)
        {
            if (checkCorrectAnswer())
            {
                currentPoints.Text = "";
                questionNumber++;
                questionOutput();
                checkVisible();
                changeTB();
            }
            else
            {  currentPoints.Text  =  "Сумма
усіх відповідей не дорівнює 10 чи десь стоїть
1"; }
        }
        private                          void
questionOutputWithNumber(dynamic
jsonResponse)
        {
            questionName.Text               =
jsonResponse.block[questionNumber-1][0];
            Option1.Text                    =
jsonResponse.block[questionNumber - 1][1];
            Option2.Text                    =
jsonResponse.block[questionNumber - 1][2];
            Option3.Text                    =
jsonResponse.block[questionNumber - 1][3];
            Option4.Text                    =
jsonResponse.block[questionNumber - 1][4];
            Option5.Text                    =
jsonResponse.block[questionNumber - 1][5];
            Option6.Text                    =
jsonResponse.block[questionNumber - 1][6];
            Option7.Text                    =
jsonResponse.block[questionNumber - 1][7];
            Option8.Text                    =
jsonResponse.block[questionNumber - 1][8];
        }
        private void questionOutput()
        {
            using  (StreamReader  r  =  new
StreamReader("belbinTest.json"))
            {
                string json = r.ReadToEnd();
                dynamic    jsonResponse    =
JsonConvert.DeserializeObject(json);

questionOutputWithNumber(jsonResponse);
            }
        }
        private void fillMas0()
        {
            for (int i = 0; i < 7; i++)
            {
                for (int j = 0; j < 8; j++)
                {
                    answerPoint[i, j] = 0;
                }
            }
        }
        private  void  BelbinTest_Load(object
sender, EventArgs e)
        {
            questionNumber = 1;
            PreviousQuestion.Visible        =
false;
            fillMas0();
            questionOutput();
        }
        private                          void
PreviousQuestion_Click(object         sender,
EventArgs e)
        {
            if (checkCorrectAnswer())
            {
                currentPoints.Text = "";
                questionNumber--;
                questionOutput();
                checkVisible();
                changeTB();
            }
            else
            {  currentPoints.Text  =  "Сумма
усіх відповідей не дорівнює 10"; }
        }
        private                          void
checkDigit(KeyPressEventArgs e)
        {
```

```
            if     (Char.IsDigit(e.KeyChar))
return;
            else
                e.Handled = true;
        }
        private                          void
pointsV1_KeyPress(object            sender,
KeyPressEventArgs e)
        {
            checkDigit(e);
        }
        private void createFinalAnswer()
        {
            answerTranscription[0]         =
answerPoint[0,  6]  +  answerPoint[1,  0]  +
answerPoint[2,  7]  +  answerPoint[3,  3]  +
answerPoint[4,  1]  +  answerPoint[5,  5]  +
answerPoint[6, 4];
            answerTranscription[1]         =
answerPoint[0,  3]  +  answerPoint[1,  1]  +
answerPoint[2,  0]  +  answerPoint[3,  7]  +
answerPoint[4,  5]  +  answerPoint[5,  2]  +
answerPoint[6, 6];
            answerTranscription[2]         =
answerPoint[0,  5]  +  answerPoint[1,  4]  +
answerPoint[2,  2]  +  answerPoint[3,  1]  +
answerPoint[4,  3]  +  answerPoint[5,  6]  +
answerPoint[6, 0];
            answerTranscription[3]         =
answerPoint[0,  2]  +  answerPoint[1,  6]  +
answerPoint[2,  3]  +  answerPoint[3,  4]  +
answerPoint[4,  7]  +  answerPoint[5,  0]  +
answerPoint[6, 5];
            answerTranscription[4]         =
answerPoint[0,  0]  +  answerPoint[1,  2]  +
answerPoint[2,  5]  +  answerPoint[3,  6]  +
answerPoint[4,  4]  +  answerPoint[5,  7]  +
answerPoint[6, 3];
            answerTranscription[5]         =
answerPoint[0,  7]  +  answerPoint[1,  3]  +
answerPoint[2,  6]  +  answerPoint[3,  2]  +
answerPoint[4,  0]  +  answerPoint[5,  4]  +
answerPoint[6, 1];
            answerTranscription[6]         =
answerPoint[0,  1]  +  answerPoint[1,  5]  +
answerPoint[2,  4]  +  answerPoint[3,  0]  +
answerPoint[4,  2]  +  answerPoint[5,  1]  +
answerPoint[6, 7];
            answerTranscription[7]         =
answerPoint[0,  4]  +  answerPoint[1,  7]  +
answerPoint[2,  1]  +  answerPoint[3,  5]  +
answerPoint[4,  6]  +  answerPoint[5,  3]  +
answerPoint[6, 2];
        }
        private                          void
outputAnswer(BelbinTestAnswer f)
        {
            f.implementer.Text             =
Math.Round(answerTranscription[0]  /  70  *
100).ToString() + "%";
            f.coordinator.Text             =
Math.Round(answerTranscription[1]  /  70  *
100).ToString() + "%";
            f.creator.Text                 =
Math.Round(answerTranscription[2]  /  70  *
100).ToString() + "%";
            f.ideaGenerator.Text           =
Math.Round(answerTranscription[3]  /  70  *
100).ToString() + "%";
            f.researcher.Text              =
Math.Round(answerTranscription[4]  /  70  *
100).ToString() + "%";
            f.expert.Text                  =
Math.Round(answerTranscription[5]  /  70  *
100).ToString() + "%";
            f.diplomat.Text                =
Math.Round(answerTranscription[6]  /  70  *
100).ToString() + "%";
            f.executor.Text                =
Math.Round(answerTranscription[7]  /  70  *
100).ToString() + "%";


        }
        private void takeResult_Click(object
sender, EventArgs e)
        {
            if (checkCorrectAnswer())
            {
                createFinalAnswer();
                BelbinTestAnswer  f  =  new
BelbinTestAnswer();
                outputAnswer(f);
                f.ShowDialog();
            }
            else
            {  currentPoints.Text  =  "Сумма
усіх відповідей не дорівнює 10 чи десь стоїть
1!"; }

        }
    }
```

Модуль YoungTest.cs

```
public partial class YoungTest : Form
    {
        public YoungTest()
        {
            InitializeComponent();
        }
        private int questionNumber;
        private int[] selectedAnswer;
        private int answerAmount;
        private dynamic jsonResponse;
        private void changeCB()
```

```
        {
            if
(selectedAnswer[questionNumber - 1] == 0)
            {
                //answerA.Checked = true;
                answerA.Checked = false;
                answerB.Checked = false;
                answerC.Checked = false;
            }
            if
(selectedAnswer[questionNumber - 1] == 1)
            { answerA.Checked = true; }
            else
            {
                if
(selectedAnswer[questionNumber - 1] == 2)
                { answerB.Checked = true; }
                else
                {
                    if
(selectedAnswer[questionNumber - 1] == 3)
                    {    answerC.Checked    =
true; }
                }
            }
        }
        private                          void
questionOutputWithNumber()
        {
            questionName.Text              =
jsonResponse.block[questionNumber - 1][0];
            answerA.Text                   =
jsonResponse.block[questionNumber - 1][1];
            answerB.Text                   =
jsonResponse.block[questionNumber - 1][2];
            answerAmount = 2;
            if   (dataClass.thirdAnswer   ==
true)
            {
                answerC.Text               =
jsonResponse.block[questionNumber - 1][3];
                answerAmount = 3;
            }
        }
        private void questionOutput()
        {
            using  (StreamReader  r  =  new
StreamReader(dataClass.fileWithQuestionName[
dataClass.currentNumberTest]))
            {
                string json = r.ReadToEnd();
                jsonResponse               =
JsonConvert.DeserializeObject(json);
                selectedAnswer     =     new
int[jsonResponse.questionAmount];
                dataClass.thirdAnswer      =
jsonResponse.thirdAnswer;
                dataClass.questionAmount   =
jsonResponse.questionAmount;
                questionOutputWithNumber();
            }
        }
        private  void  YoungTest_Load(object
sender, EventArgs e)
        {
            questionNumber = 1;
            PreviousQuestion.Visible       =
false;
            questionOutput();
            checkVisible();

        }
        private void checkVisible()
        {
            errorLabel.Text = "";
            if   (dataClass.thirdAnswer   ==
true)
            { answerC.Visible = true; }
            if      (questionNumber      !=
dataClass.questionAmount  &&  questionNumber
!= 1)
            {
                NextQuestion.Visible = true;
                PreviousQuestion.Visible   =
true;
            }
            if      (questionNumber      ==
dataClass.questionAmount)
            {
                NextQuestion.Visible       =
false;
            }
            if (questionNumber == 1)
            {
                PreviousQuestion.Visible   =
false;
            }
            if      (questionNumber      ==
dataClass.questionAmount)
            { nextPage.Visible = true; }
        }
        private void rememberAnswer()
        {
            if (answerA.Checked == true)
            {

selectedAnswer[questionNumber - 1] = 1;
            }
            if (answerB.Checked == true)
            {

selectedAnswer[questionNumber - 1] = 2;
            }
            if (answerC.Checked == true)
            {

selectedAnswer[questionNumber - 1] = 3;
            }
```

```
        }
        private int checkChooseAnswer()
        {
            int a = 0;
            foreach (var pb in
this.Controls.OfType<RadioButton>())
            {
                if (pb.Checked == false) {
a++; }
            }
            return a;
        }
        private void
NextQuestion_Click(object sender, EventArgs
e)
        {
            if (checkChooseAnswer() != 3)
            {
            rememberAnswer();
            questionNumber++;
            questionOutputWithNumber();
            checkVisible();
            changeCB();
            }
            else
            {
                errorLabel.Text = "Ви не
вибрали відповідь!";
            }
        }
        private void
PreviousQuestion_Click(object sender,
EventArgs e)
        {
            rememberAnswer();
            questionNumber--;
            questionOutputWithNumber();
            checkVisible();
            changeCB();
        }
        private void nextPage_Click(object
sender, EventArgs e)
        {
            rememberAnswer();
            chooseAnswerCreation();
            dataClass.currentNumberTest++;
            if (dataClass.currentNumberTest
< dataClass.fileWithQuestionName.Length)
            {
                this.Hide();
                YoungTest f = new
YoungTest();
                f.ShowDialog();
            }
            else {
                this.Hide(); }
        }
        private void chooseAnswerCreation()
        {
            string a =
jsonResponse.calculationMethodName;
            switch (a)
            {
                case "youngtest":
                    youngtest();
                    break;
                case "OvKtest1":
                    OvKtest1();
                    break;
                case "OvKtest2":
                    OvKtest2();
                    break;
                case "OvKtest3":
                    OvKtest3();
                    break;
                case
"studying_the_motivation_of_studying_at_a_un
iversity":

studying_the_motivation_of_studying_at_a_uni
versity();
                    break;
                case
"NeedForAchievementScale":

AchievementMotivationLevel();
                    break;
            }
        }
        private void youngtest()
        {
            int[] answerMas = new int[20] {
2, 1, 2, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1,
1, 1, 1, 2, 1 };
            int rez = 0;
            for (int i = 0; i <
selectedAnswer.Length; i++)
            {
                if (selectedAnswer[i] ==
answerMas[i])
                {
                    rez++;
                }
            }
            testsResult.youngtest = rez;
        }
        private void OvKtest1()
        {
            int rez = 0;
            for (int i = 0; i <
selectedAnswer.Length; i++)
            {
                rez += selectedAnswer[i];
            }
            testsResult.OvKtest1 = rez * 5;
        }
        private void OvKtest2()
        {
```

```
            int rez = 0;
            for (int i = 0; i < selectedAnswer.Length; i++)
            {
                switch (selectedAnswer[i])
                {
                    case 1:
                        rez += 3;
                        break;
                    case 3:
                        rez += 6;
                        break;
                }
            }
            testsResult.OvKtest2 = rez;
        }
        private void OvKtest3()
        {
            int rez = 0;
            for (int i = 0; i < selectedAnswer.Length; i++)
            {
                switch (selectedAnswer[i])
                {
                    case 1:
                        rez += 2;
                        break;
                    case 2:
                        rez += 1;
                        break;
                }
            }
            testsResult.OvKtest3 = rez;
        }
        private void studying_the_motivation_of_studying_at_a_university()
        {
            double AcquisitionKnowledge=0, MasteryProfession=0, GettingDiploma=0;
            if (selectedAnswer[0]==1) { AcquisitionKnowledge += 3.6; }
            if (selectedAnswer[1] == 1) { MasteryProfession += 1; }
            if (selectedAnswer[2] == 2) { GettingDiploma += 3.5; }
            if (selectedAnswer[3] == 1) { AcquisitionKnowledge += 3.6; }
            if (selectedAnswer[4] == 1) { GettingDiploma += 2.5; }
            if (selectedAnswer[5] == 1) { AcquisitionKnowledge += 2.4; }
            if (selectedAnswer[6] == 2) { AcquisitionKnowledge += 1.2; }
            if (selectedAnswer[7] == 1) { MasteryProfession += 2; }
            if (selectedAnswer[8] == 1) { MasteryProfession += 2; }
            if (selectedAnswer[9] == 1) { GettingDiploma += 1.5; }
            if (selectedAnswer[10] == 1) { GettingDiploma += 1.5; }
            if (selectedAnswer[11] == 2) { AcquisitionKnowledge += 1.8; }
            if (selectedAnswer[12] == 1) { MasteryProfession += 3; }
            if (selectedAnswer[13] == 1) { GettingDiploma += 1; }
            if (selectedAnswer[14] == 1) { MasteryProfession += 1; }
            if (selectedAnswer[15] == 1) { MasteryProfession += 1; }

            testsResult.AcquisitionKnowledge = AcquisitionKnowledge;
            testsResult.MasteryProfession = MasteryProfession;
            testsResult.GettingDiploma = GettingDiploma;
        }
        private void AchievementMotivationLevel()
        {
            int[] answerMas = new int[22] {2, 1, 2, 2, 2, 1,1,1,2,3,2,2,2,1,2,1,2,1,1,2,1,1};
            int rez = 0;
            for (int i = 0; i < selectedAnswer.Length; i++)
            {
                if (selectedAnswer[i] == answerMas[i])
                {
                    rez++;
                }
            }

            testsResult.AchievementMotivationLevel = rez;
        }
    }
```

Модуль

MotivationLearningActivites.cs

```
public partial class MotivationLearningActivites : Form
    {
        public MotivationLearningActivites()
        {
            InitializeComponent();
        }
        int questionNumber = 1;
        int[] selectedAnswer;
        int answerMasSize;
```

```
        dynamic jsonResponse;
        private void changeCB()
        {
            if
(selectedAnswer[questionNumber - 1] == -1)
            {
                points4.Checked = false;
                points3.Checked = false;
                //points3.Checked = true;
                points2.Checked = false;
                points1.Checked = false;
                points0.Checked = false;

            }


            if
(selectedAnswer[questionNumber   -   1]   ==
dataClass.testScores[3])
            { points3.Checked = true; }
            else
            {
                if
(selectedAnswer[questionNumber   -   1]   ==
dataClass.testScores[2])
                { points2.Checked = true; }
                else
                {
                    if
(selectedAnswer[questionNumber   -   1]   ==
dataClass.testScores[1])
                    {   points1.Checked   =
true; }
                    else
                    {
                        if
(selectedAnswer[questionNumber   -   1]   ==
dataClass.testScores[0])
                        {  points0.Checked  =
true; }
                        else
                        {

if(dataClass.testScores.Length>4)
                            if
(selectedAnswer[questionNumber   -   1]   ==
dataClass.testScores[4])
                            {
points4.Checked = true; }
                        }
                    }
                }
            }
        }
        private void checkVisible()
        {
            errorLabel.Text = "";
            if      (questionNumber      !=
selectedAnswer.Length && questionNumber != 1)
            {
                NextQuestion.Visible = true;
                PreviousQuestion.Visible   =
true;
                NextQuestion.Text          =
"Наступне питання";
            }
            if      (questionNumber      ==
selectedAnswer.Length)
            {
                NextQuestion.Text          =
"Наступний тест";
            }
            if (questionNumber == 1)
            {
                PreviousQuestion.Visible   =
false;
            }

        }
        private                            void
questionOutputWithNumber()
        {
            questionText.Text              =
jsonResponse.block[questionNumber - 1];
        }
        private void check()
        {
            int help = answerMasSize;
            foreach     (var     pb     in
this.Controls.OfType<RadioButton>())
            {
                if (help == 0)
                { pb.Visible = false; }
                else
                { help--; }
            }
        }
        private void checkBoxChange()
        {

if(dataClass.testScores.Length<5)
            {
                points4.Visible = false;
            }
            else
            {
                points4.Text               =
dataClass.testScores[4].ToString();
            }
            points3.Text                   =
dataClass.testScores[3].ToString();
            points2.Text                   =
dataClass.testScores[2].ToString();
            points1.Text                   =
dataClass.testScores[1].ToString();
            points0.Text                   =
dataClass.testScores[0].ToString();
        }
```

```
        private void questionOutput()
        {
            using (StreamReader r = new
StreamReader(dataClass.fileWithQuestionName[
dataClass.currentNumberTest]))
            {
                string json = r.ReadToEnd();
                jsonResponse =
JsonConvert.DeserializeObject(json);
                selectedAnswer = new
int[jsonResponse.questionAmount];
                dataClass.testScores =
(int[])jsonResponse.scoresMas.ToObject(typeo
f(int[]));
                answerMasSize =
dataClass.testScores.Length;
                checkBoxChange();
                questionOutputWithNumber();
            }
        }
        private void rememberAnswer()
        {
            if (points4.Checked ==
true&&answerMasSize>=5)
            {

selectedAnswer[questionNumber - 1] =
dataClass.testScores[4];
            }
            if (points3.Checked == true)
            {

selectedAnswer[questionNumber - 1] =
dataClass.testScores[3];
            }
            if (points2.Checked == true)
            {

selectedAnswer[questionNumber - 1] =
dataClass.testScores[2];
            }
            if (points1.Checked == true)
            {

selectedAnswer[questionNumber - 1] =
dataClass.testScores[1];
            }
            if (points0.Checked == true)
            {

selectedAnswer[questionNumber - 1] =
dataClass.testScores[0];
            }
        }
        private int checkChooseAnswer()
        {
            int a = 0;
            foreach (var pb in
this.Controls.OfType<RadioButton>())
            {
                if (pb.Checked == false) {
a++; }
            }
            return a;
        }
        private void
NextQuestion_Click(object sender, EventArgs
e)
        {
            if (checkChooseAnswer() != 5)
            {
            if (questionNumber ==
selectedAnswer.Length)
            {
                rememberAnswer();
                endTest();
                nextTest();
            }
            else
            {
                rememberAnswer();
                questionNumber++;
                questionOutputWithNumber();
                checkVisible();
                changeCB();
            }
            }
            else
            {
                errorLabel.Text = "Ви не
вибрали відповідь!";
            }
        }
        private void
PreviousQuestion_Click(object sender,
EventArgs e)
        {
            rememberAnswer();
            questionNumber--;
            questionOutputWithNumber();
            checkVisible();
            changeCB();
        }
        private void fillMas()
        {
            for(int i=0; i<
selectedAnswer.Length; i++)
                selectedAnswer[i] = -1;
        }
        public void changeLabel()
        {
            if (dataClass.currentNumberTest
== 1)
            {
                label1.Text = " Прочитайте та
оцініть, наскільки кожне з них вплинуло на
ваш вибір професії. Відповіді можуть бути 5
видів: «дуже сильно вплинуло» – 5 балів;
```

```
«сильно вплинуло» – 4 бали; «середньо вплинуло» – 3 бали; «слабко вплинуло» – 2 бали; «не вплинуло» -1 бал.";
            }
            if  (dataClass.currentNumberTest == 2)
            {
                label1.Text = " Прочитайте кожне висловлювання та висловіть свою думку стосовно досліджуваних предметів, простовив проти номера висловлювання відповідну вам відповідь, використовуйте для цього вказані в дужках позначення:«вірно» (4); «мабуть, вірно» (3); «мабуть, не так» (2); «не так» (1)";
            }
        }
        private                          void MotivationLearningActivites_Load(object sender, EventArgs e)
        {
            PreviousQuestion.Visible       = false;
            questionOutput();
            fillMas();
            changeCB();
            changeLabel();
        }
        public void nextTest()
        {
            rememberAnswer();
            dataClass.currentNumberTest++;
            if  (dataClass.currentNumberTest < dataClass.fileWithQuestionName.Length)
            {
                this.Hide();
                MotivationLearningActivites f = new MotivationLearningActivites();
                f.ShowDialog();
            }
            else {
                this.Hide(); }
        }
        public void endTest()
        {
            if  (dataClass.currentNumberTest == 0)
            {

Motivation_for_learning_activities();
            }
            if  (dataClass.currentNumberTest == 1)
            {

MotivesForChoosingProfession();
            }
            if  (dataClass.currentNumberTest == 2)
            {

satisfactionLearningActivities();
            }
        }
        public void createMus(double[] mas)
        {
            int a = 0;
            for (int i = 0; i < 5; i++)
            {
                for (int j = 0; j < 6; j++)
                {
                    mas[j]                += selectedAnswer[a];
                    a++;
                }
            }
            for (int i = 0; i < mas.Length; i++)
            {
                mas[i] = mas[i] / 5;
            }
        }
        public                           void Motivation_for_learning_activities()
        {
            double[] mas = new double[6];
            createMus(mas);
            double    cognitive,    social, broadCognitive,     narrowOrProperCognitive, selfdevelopmentOrPersonal, motivesForForcedLearning,      narrowSocial, cooperationOrSocialityKnowledge;
            cognitive = (mas[0] + mas[1] + mas[2])/3;
            social  =  (mas[3]  +  mas[4]  + mas[5])/3;
            broadCognitive = mas[0];
            narrowOrProperCognitive        = mas[1];
            selfdevelopmentOrPersonal      = mas[2];
            motivesForForcedLearning       = mas[3];
            narrowSocial = mas[4];
            cooperationOrSocialityKnowledge = mas[5];
            double[] rezultMas = new double[] {   cognitive,   social,   broadCognitive, narrowOrProperCognitive, selfdevelopmentOrPersonal, motivesForForcedLearning,      narrowSocial, cooperationOrSocialityKnowledge };

testsResult.Motivation_for_learning_activities = rezultMas;
        }
        public                           void MotivesForChoosingProfession()
```

```
        {
            int         individual         =
selectedAnswer[0]+        selectedAnswer[4]+
selectedAnswer[7]+       selectedAnswer[14]+
selectedAnswer[19],
                social  =  selectedAnswer[2]+
selectedAnswer[6]+       selectedAnswer[11]+
selectedAnswer[13]+ selectedAnswer[16],
                positive                    =
selectedAnswer[3]+        selectedAnswer[8]+
selectedAnswer[9]+       selectedAnswer[15]+
selectedAnswer[18],
                negative                    =
selectedAnswer[1]+        selectedAnswer[5]+
selectedAnswer[10]+      selectedAnswer[12]+
selectedAnswer[17];

            int[]
MotivesForChoosingProfession  =  new  int[]  {
individual, social, positive, negative };

testsResult.MotivesForChoosingProfession   =
MotivesForChoosingProfession;

        }
        public                              void
satisfactionLearningActivities()
        {
            double educationalActivity = 0,
                studyingProccess = 0,
                educationalProcess = 0,
                chosenProfession = 0,
                relationshipsWithClassmates
= 0,
                interactionWithTeachers = 0,
                life = 0;

            for(int
i=0;i<selectedAnswer.Length;i++)
            {
                educationalActivity       +=
selectedAnswer[i];
            }
            for (int i = 0; i < 15; i++)
            {
                studyingProccess          +=
selectedAnswer[i];
            }
            for (int i = 15; i < 11+15; i++)
            {
                educationalProcess        +=
selectedAnswer[i];
            }
            for (int i = 26; i < 11+26; i++)
            {
                chosenProfession          +=
selectedAnswer[i];
            }
            for (int i = 37; i < 11+37; i++)
            {
                relationshipsWithClassmates
+= selectedAnswer[i];
            }
            for (int i = 48; i < 11+48; i++)
            {
                interactionWithTeachers   +=
selectedAnswer[i];
            }
            for (int i = 59; i < 11+59; i++)
            {
                life += selectedAnswer[i];
            }
            double[] answerMas = new double[]
{educationalActivity/70,
                studyingProccess/15,
                educationalProcess/11,
                chosenProfession/11,

relationshipsWithClassmates/11,
                interactionWithTeachers/11,
                life/11,
        };

testsResult.satisfactionLearningActivities =
answerMas;
        }
    }
```

Модуль calculetedResult.cs

```
class calculetedResult
    {
        private dynamic jsonResponse;
        private string[] answerTekst;
        public      void      readFile(string
fileName)
        {
            using  (StreamReader  r  =  new
StreamReader(fileName))
            {
                string json = r.ReadToEnd();
                jsonResponse                =
JsonConvert.DeserializeObject(json);
            }
        }
        public void youngR()
        {
            if  (jsonResponse.youngtestL  <=
7)
            { answerTekst[0] = "Інтроверт";
}
            if (jsonResponse.youngtestL > 7
&& jsonResponse.youngtestL <= 13)
            { answerTekst[0] = "Амбіверт"; }
```

```
            if  (jsonResponse.youngtestL  >
13)
            { answerTekst[0] = "Екстраверт";
}
        }
        public void commandWork()
        {
            int  rezT1,  commandUnSkill=0,
commandSkill = 0;
            int a = jsonResponse.OvKtest1L;
            rezT1 = (11-(a / 5 - 1)) * 5;
            if(rezT1<20)
            { commandUnSkill++; }
            if (jsonResponse.OvKtest2L<25) {
commandUnSkill++; }
            if  (jsonResponse.OvKtest3L<10||
jsonResponse.OvKtest3L>25)                {
commandUnSkill++; }
            if(commandUnSkill>=2)
            {
                answerTekst[1]  =  "Поганий
командний гравець";
            }
            else   {   answerTekst[1]   =
"Підходить для командної роботи"; }
            if (rezT1 >= 45)
            { commandSkill++; }
            if  (jsonResponse.OvKtest2L  >=
40) { commandSkill++; }
            if (jsonResponse.OvKtest3L > 15
||   jsonResponse.OvKtest3L   <   25)   {
commandSkill++; }
            if (commandSkill == 3)
            {
                answerTekst[1]  =  "Гарный
командний гравець";
            }
        }
        public void WuzMotivation()
        {
            int motivationAll = 0;
            if
(jsonResponse.AcquisitionKnowledgeL >= 8.4)
            {
                answerTekst[2]  +=  "Націлен
на здобування знань ";
                motivationAll += 1;
            }
            if
(jsonResponse.MasteryProfessionL >= 7)
            {
                answerTekst[2]  +=  "Отримає
велике задоволення від обранної професії ";
                motivationAll += 1;
            }
            if (jsonResponse.GettingDiplomaL
>= 7)
            {
                answerTekst[2]  +=  "Націлен
на отримання диплому ";
                motivationAll += 1;
            }
            if
(jsonResponse.AcquisitionKnowledgeL        <
jsonResponse.GettingDiplomaL              &&
jsonResponse.MasteryProfessionL            <
jsonResponse.GettingDiplomaL)
            {
                if (motivationAll == 1)
                {     answerTekst[2]     +=
"Цікавить тільке отримання диплому "; }
                if (motivationAll == 0)
                { answerTekst[2] += "Нічого
не цікавить "; }
            }
        }
        public                            void
AchievementMotivationLevel()
        {
            if
(jsonResponse.AchievementMotivationLevelL  <
12)
            {
                answerTekst[3]  +=  "Низька
мотивація досягнення";
            }
            if
(jsonResponse.AchievementMotivationLevelL >=
12&&
jsonResponse.AchievementMotivationLevelL<16)
            {
                answerTekst[3]  +=  "Середня
мотивація досягнення";
            }
            if
(jsonResponse.AchievementMotivationLevelL >=
16)
            {
                answerTekst[3]  +=  "Висока
мотивація досягнення";
            }
        }
        public                            void
Motivation_for_learning_activities()
        {
            int[]        answerMas        =
(int[])jsonResponse.Motivation_for_learning_
activitiesL.ToObject(typeof(int[]));
            if (answerMas[0] > answerMas[1]
&& answerMas[0] != answerMas[1])
            {
                answerTekst[4] += "Переважає
пізнавальна мотивація ";
            }
            else   {   answerTekst[4]   +=
"Переважає соціальна мотивація "; }
```

```
            int mLevel = (answerMas[0] + answerMas[1]) / 2;
            if (mLevel > 3)
            {
                answerTekst[4] += "Висока мотивація до навчання ";
            }
            if (mLevel >= 2&& mLevel <= 3)
            {
                answerTekst[4] += "Середня мотивація до навчання ";
            }
            if (mLevel < 2)
            {
                answerTekst[4] += "Низька мотивація до навчання ";
            }
        }
        public void MotivesForChoosingProfession()
        {
            int[] answerMas = (int[])jsonResponse.MotivesForChoosingProfessionL.ToObject(typeof(int[]));
            int mLevel = (answerMas[0] + answerMas[1]) / (answerMas[2] + answerMas[3]);
            if (mLevel == 1)
            {
                answerTekst[5] += "Переважають внутрішні мотиви ";
            }
            else
            {
                if (answerMas[3] > answerMas[2])
                {
                    answerTekst[5] += "Переважають зовнішні мотиви(негативні) ";
                }
                else
                {
                    answerTekst[5] += "Переважають зовнішні мотиви ";
                }
            }
        }
        public void satisfactionLearningActivities()
        {
            int mLevel = jsonResponse.satisfactionLearningActivitiesL[0];
            if (mLevel>1&& mLevel <= 1.5)
            {
                answerTekst[6] +="Велика незадоволеність під час навчального процесу ";
            }
            if (mLevel > 1.5 && mLevel <= 2.5)
            {
                answerTekst[6] += "Незадоволеність під час навчального процесу ";
            }
            if (mLevel > 2.5 && mLevel <= 3.5)
            {
                answerTekst[6] += "Учбова діяльність у мережах норми ";
            }
            if (mLevel > 3.5 && mLevel <= 4)
            {
                answerTekst[6] += "Задоволення учбовою діяльністю ";
            }

        }
        public async void calculeted(int textMasSize)
        {
            string path = "note1.txt";
            File.Delete(path);
            using (StreamWriter writer = new StreamWriter(path, true))
            {
                await writer.WriteLineAsync("Результат:");
                for (int i = 1; i <= textMasSize; i++)
                {
                    answerTekst = new string[7];
                    readFile("testResult\\" + i.ToString() + ".json");
                    youngR();
                    commandWork();
                    WuzMotivation();

                    AchievementMotivationLevel();

                    Motivation_for_learning_activities();

                    MotivesForChoosingProfession();

                    satisfactionLearningActivities();
                    await writer.WriteAsync(i.ToString()+".)        "+ answerTekst[0]+" ### "+ answerTekst[1]+ " ### " + answerTekst[2]+ " ### " + answerTekst[3] + " ### " + answerTekst[4] + " ### " + answerTekst[5] + " ### " + answerTekst[6] + "\n");
                }
            }
        }
    }
```

Модуль testResultF.cs

```
class testResultF
    {
        public string nameL { get; set; }
        public int youngtestL { get; set; }
        public int OvKtest1L { get; set; }
        public int OvKtest2L { get; set; }
        public int OvKtest3L { get; set; }
        public double AcquisitionKnowledgeL
{ get; set; }
        public double MasteryProfessionL {
get; set; }
        public double GettingDiplomaL { get;
set; }
        public int
AchievementMotivationLevelL { get; set; }
        public double[]
Motivation_for_learning_activitiesL { get;
set; }
        public int[]
MotivesForChoosingProfessionL { get; set; }
        public double[]
satisfactionLearningActivitiesL { get; set;
}

        public testResultF()
        {
            nameL = testsResult.name;
            youngtestL =
testsResult.youngtest;
            OvKtest1L =
testsResult.OvKtest1;
            OvKtest2L =
testsResult.OvKtest2;
            OvKtest3L =
testsResult.OvKtest3;
            AcquisitionKnowledgeL =
testsResult.AcquisitionKnowledge;
            MasteryProfessionL =
testsResult.MasteryProfession;
            GettingDiplomaL =
testsResult.GettingDiploma;
            AchievementMotivationLevelL =
testsResult.AchievementMotivationLevel;

Motivation_for_learning_activitiesL =
testsResult.Motivation_for_learning_activiti
es;
            MotivesForChoosingProfessionL =
testsResult.MotivesForChoosingProfession;
            satisfactionLearningActivitiesL
=
testsResult.satisfactionLearningActivities;
        }
    }
```

Модуль teacherControllForm.cs

```
public partial class teacherControllForm :
Form
    {
        public teacherControllForm()
        {
            InitializeComponent();
        }

        private void
startProgram_Click(object sender, EventArgs
e)
        {
            if (groupNumber.Text!="")
            {
                dataClass.group =
groupNumber.Text;
                groupCombinated
groupCombinated = new groupCombinated();

groupCombinated.createCommand();
                errorLable.Text =  "Файли
створені!";
            }
            else
            {
                errorLable.Text = "Введіть
номер групи!";
            }

        }
    }
```

Модуль teamWeaknesses.cs

```
class teamWeaknesses
    {
        List<int>[]
masCommandWithoutCharacteristic;
        public List<int>[]
weaknessesMas(int[,] groupMas, int command)
        {
            weaknesses(groupMas);
            return
masCommandWithoutCharacteristic;
        }
        public void weaknesses(int[,]
groupMas)
        {
            masCommandWithoutCharacteristic
= new List<int>[groupMas.GetLength(0)];

fillListInMas(groupMas.GetLength(0));
```

```
            for(int                    i=0;
i<groupMas.GetLength(0); i++)
            {
                for(int              j=0;j<
groupMas.GetLength(1); j++)
                {

findIndexPeople(groupMas[i,j], i);
                }
            }
        }
        public     void     fillListInMas(int
lenght)
        {
            for(int i=0; i<lenght; i++)
            {
                for(int j=1; j<9;j++)
                {
                    if (j == 1)
                    {
masCommandWithoutCharacteristic[i]   =   new
List<int>(); }

masCommandWithoutCharacteristic[i].Add(j);
                }
            }
        }
        public    void    findIndexPeople(int
numberPeople, int group)
        {
            for(int                    i=0;
i<dataClass.mas.GetLength(0);i++ )
            {

if(dataClass.mas[i,3]==numberPeople)
                {
                    clearList(i, group);
                    break;
                }
            }
        }
        public void clearList(int people, int
group)
        {
            for(int i=0; i<3;i++)
            {
                if (dataClass.mas[people, i]
!= 0)
                {

masCommandWithoutCharacteristic[group].

RemoveAt(masCommandWithoutCharacteristic[gro
up].

IndexOf(dataClass.mas[people, i]));
                }
            }
        }
```

Модуль wordConnect.cs

```
class wordConnect
    {
        private FileInfo _fileInfo;
        public wordConnect(string fileName)
        {
            if (File.Exists(fileName))
            {
                _fileInfo       =       new
FileInfo(fileName);
            }
            else      {      throw      new
ArgumentException("File not found"); }
        }
        internal                          void
Process(Dictionary<string,   string>   items,
string name)
        {
            Word.Application app = null;
                app           =           new
Word.Application();
                Object         file         =
_fileInfo.FullName;
                Object       missing       =
Type.Missing;

                app.Documents.Open(file);

                foreach (var item in items)
                {
                    Word.Find     find     =
app.Selection.Find;
                    find.Text = item.Key;
                    find.Replacement.Text   =
item.Value;

                    Object      wrap      =
Word.WdFindWrap.wdFindContinue;
                    Object     replace     =
Word.WdReplace.wdReplaceAll;

                    find.Execute(FindText:
Type.Missing,
                        MatchCase: false,
                        MatchWholeWord:
false,
                        MatchWildcards:
false,
                        MatchSoundsLike:
missing,
                        MatchAllWordForms:
false,
                        Forward: true,
                        Wrap: wrap,
    }
```

```
                        Format: false,
                        ReplaceWith:
missing, Replace: replace);
                }

                Object     newFileName     =
Path.Combine(_fileInfo.DirectoryName,DateTim
e.Now.ToString("yyyyMMdd HHmmss "+name));

app.ActiveDocument.SaveAs2(newFileName);
                app.ActiveDocument.Close();
                app.Quit();

        }
    }
```

Модуль groupCombinated.cs

```
class groupCombinated
    {
        int[,] groupMas;
        public int[,] createGr()
        {
            createGroup  createGroup  =  new
createGroup();
            groupMas                        =
createGroup.createGroupMas();
            return groupMas;
        }
        public List<int>[] weaknesses()
        {
            teamWeaknesses  teamWeaknesses  =
new teamWeaknesses();
            List<int>[]     weaknesses      =
teamWeaknesses.weaknessesMas(groupMas, 15);
            return weaknesses;
        }
        public void inWord()
        {
            combinatedInWord   combinated   =
new combinatedInWord();

combinated.createFile(weaknesses(),
groupMas);
        }
        public void createCommand()
        {
            groupMas = createGr();
            inWord();
        }
    }
```

Модуль combinatedInWord.cs

```
class combinatedInWord
    {
        public  void  createFile(List<int>[]
weaknesses, int[,] groupMas)
        {
            string      sampleFileName      =
"sampleWordAnswer.docx";
            var       helper       =       new
wordConnect(sampleFileName);
            string[,]      information      =
createString(weaknesses, groupMas);

            for   (int   i   =   0;   i   <
groupMas.GetLength(0); i++)
            {
                var      items      =      new
Dictionary<string, string>
                {
                    {            "{groupN}",
information[i,0]},
                    {             "{teamN}",
information[i,1]},
                    {       "{teamMembers}",
information[i,2]},
                    {    "{teamWeaknesses}",
information[i,3]},
                    //{  "{teamWeaknesses}",
""},
                   //     {     "{ProblemS}",
"aye"},
                    {           "{Problem1}",
information[i,4]},
                    {           "{Problem2}",
information[i,5]},
                    {           "{Problem3}",
information[i,6]},
                    {           "{Problem4}",
information[i,7]},
                    {           "{Problem5}",
information[i,8]},
                    {           "{Problem6}",
information[i,9]},
                    {           "{Problem7}",
information[i,10]},
                    {           "{Problem8}",
information[i,11]},
                };
                helper.Process(items,
i.ToString());
            }

        }
        public                      string[,]
createString(List<int>[]  weaknesses,  int[,]
groupMas)
        {
            string       group,       team,
teamMembers1, teamWeaknesses, solution;
            group                           =
dataClass.dataPeople[0][12];
```

```
            string[,] information = new
string[groupMas.GetLength(0), 12];
            string[] weakPlusSolution = new
string[2];
            for (int i = 0; i <
groupMas.GetLength(0); i++)
            {
                information[i, 0] = group;
                information[i, 1] =
i.ToString();
                information[i, 2] =
teamMembers(i, groupMas);
                weakPlusSolution =
weaknessesPlusSolution(i, weaknesses);
                information[i, 3] =
weakPlusSolution[0];
                information[i, 4] =
weakPlusSolution[1];
                information[i, 5] =
weakPlusSolution[2];
                information[i, 6] =
weakPlusSolution[3];
                information[i, 7] =
weakPlusSolution[4];
                information[i, 8] =
weakPlusSolution[5];
                information[i, 9] =
weakPlusSolution[6];
                information[i, 10] =
weakPlusSolution[7];
                information[i, 11] =
weakPlusSolution[8];
            }
            return information;
        }
        public string teamMembers(int
teamNumber, int[,] groupMas)
        {
            string rezult = "";
            for (int i = 0; i <
groupMas.GetLength(1); i++)
            {
                rezult +=
name(groupMas[teamNumber, i].ToString())+
"\r";
            }
            return rezult;
        }
        public string name(string number)
        {
            for (int i = 0; i <
dataClass.dataPeople.Count; i++)
            {
                if
(dataClass.dataPeople[i][10] == number)
                {
                    return
dataClass.dataPeople[i][9] + "\n";
                }
            }
            return "";
        }
        public string[]
weaknessesPlusSolution(int teamNumber,
List<int>[] weaknesses)
        {
            string[] result = new string[9];
            for(int i=0;
i<result.Length;i++)
            {
                result[i] = "";
            }
            foreach (int i in
weaknesses[teamNumber])
            {
                switch (i)
                {
                    case 1:
                        result[0] +=
"Реалізатор; ";
                        result[1] +=
"Потрібен фокус на результат особливо у сірих
зонах відповідальності. Розплануйте завдання
та проконтролюйте їх виконання.";
                        break;
                    case 2:
                        result[0] +=
"Координатор; ";
                        result[2] += "Вам
потрібно обрати лідера вашої команди або
приймати рішення разом. Також можливо
залучити зовнішнє керівництво. ";
                        break;
                    case 3:
                        result[0] +=
"Шейпер; ";
                        result[3] += "Вам
потрібно чітко прописувати цілі та
пріорітети, а також якщо з'являються труднощі
разом їх долати, а не ігнорувати їх.";
                        break;
                    case 4:
                        result[0] +=
"Генератор ідей; ";
                        result[4] +=
"Команді не вистачає креативних нових ідей
треба проводити мозковий штурм задля пошуку
рішення завдання.";
                        break;
                    case 5:
                        result[0] +=
"Дослідник ресурсів; ";
                        result[5] +=
"Потрібно подивитись на рішення зовні
пошукати ти вже готові варіанти рішення.";
                        break;
                    case 6:
```

```
                        result[0]        += "Аналітик-Стратег; ";
                        result[6] += "Треба ретельно розібрати запропоновані чи знайдені ідеї для рішення та пошукати оптимальний варіант.";
                        break;
                    case 7:
                        result[0]        += "Дипломат; ";
                        result[7]        += "Потрібно приділити увагу взаємовідносинами між членами команди та щоб кожний у команді старався не ображати інших та не думати про почуття інших членів команди.";
                        break;
                    case 8:
                        result[0]        += "Педант; ";
                        result[8]        += "Потрібно ввести чек-листи та адміністративний контроль за завершенням задач.";
                        break;
                    default:
                        result[0] += "";
                        break;
                }
            }
            return result;
        }
    }
```

Модуль createCommand.cs

```
class createCommand
    {
        private int kommand4, kommand3, kommandAmount;
        private int[,] masStudent, command;
        private int studentKolvo, maxPeopleInGroup, maxCharacteristicAmount;
        public int characteristicAmount(int Nstudent)
        {
            int characteristicAmount = 0;
            for (int i = 0; i < maxCharacteristicAmount; i++)
            {
                if (masStudent[Nstudent, i] != 0)
                {
                    characteristicAmount += 1;
                }
            }
            return characteristicAmount;
        }
        public bool checkStInCommand(int Nstudent, int Ncommand, int characteristic)
        {
            int check = 0, characteristicAmountN = characteristicAmount(Nstudent);
            for (int i = 0; i < maxCharacteristicAmount; i++)
            {
                for (int r = 0; r < maxPeopleInGroup; r++)
                {
                    for (int j = 0; j < maxCharacteristicAmount; j++)
                    {
                        if (command[Ncommand, r] != -1)
                        {
                            if (masStudent[Nstudent, i] == masStudent[command[Ncommand, r], j] && masStudent[Nstudent, i] != 0)
                            {
                                check += 1;
                            }
                        }
                    }
                }
            }
            if (check > characteristicAmountN - characteristic)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
        public void peopleInGroup()
        {
            if (kommand4 == 0)
            { maxPeopleInGroup = 3; }
            else { maxPeopleInGroup = 4; }
        }
        public void commandAmount()
        {
            kommand4 = studentKolvo - (studentKolvo / 3) * 3;
            kommand3 = studentKolvo / 3 - kommand4;
            kommandAmount = studentKolvo / 3;
        }
        public void startFillCommand()
        {
            for (int j = 0; j < kommandAmount; j++)
            {
                command[j, 0] = j;
```

```
            }
        }
        public void fillMass()
        {
            for (int i = 0; i < kommandAmount; i++)
            {
                for (int j = 0; j < maxPeopleInGroup; j++)
                {
                    command[i, j] = -1;
                }
            }
        }
        public int placeInCommand(int nCommand)
        {
            for (int i = 0; i < maxPeopleInGroup; i++)
            {
                if (command[nCommand, i] == -1)
                {
                    if (nCommand <= kommand4 || i + 1 != maxPeopleInGroup)
                    { return i; }
                }
            }
            return -1;
        }
        public int[] placeInCommandForAll()
        {
            int[] rez = new int[2];
            for (int i = 0; i < kommandAmount; i++)
            {
                for (int j = 0; j < maxPeopleInGroup; j++)
                {
                    if (command[i, j] == -1)
                    {
                        if (i <= kommand4 || j != maxPeopleInGroup)
                        {
                            rez[0] = i;
                            rez[1] = j;
                            return rez;
                        }
                    }
                }
            }
            return rez;
        }
        public int[,] commandAlg(int[,] masStudent1, int studentColvo1)
        {
            studentKolvo = studentColvo1;
            maxCharacteristicAmount = 3;
            masStudent = masStudent1;
            commandAmount();
            peopleInGroup();
            command = new int[kommandAmount, maxPeopleInGroup];
            fillMass();
            int characteristic;
            int placeInGroup;
            int[] withoutCommand = new int[studentKolvo - kommandAmount];
            int counterWithoutCommand = 0;
            bool inCommand;
            startFillCommand();
            for (int i = kommandAmount; i < studentKolvo; i++)
            {
                inCommand = false;
                characteristic = characteristicAmount(i);
                do
                {
                    for (int j = 0; j < kommandAmount; j++)
                    {
                        if (checkStInCommand(i, j, characteristic))
                        {
                            placeInGroup = placeInCommand(j);
                            if (placeInGroup != -1)
                            {
                                command[j, placeInGroup] = i;
                                inCommand = true;
                                characteristic = 0;
                                j = kommandAmount;
                            }
                        }
                    }
                    characteristic--;
                }
                while (characteristic > 0);
                if (!inCommand)
                {
                    withoutCommand[counterWithoutCommand] = i;
                    counterWithoutCommand++;
                }
            }
            int i1 = 0;
            int[] clearPozition;
            while (withoutCommand[i1] != 0)
            {
                clearPozition = placeInCommandForAll();
```

```
                command[clearPozition[0],
clearPozition[1]] = withoutCommand[i1];
                i1++;
            }

            return
repleceOnIndex(masStudent1);
        }
        public int[,] repleceOnIndex(int[,]
masStudent1)
        {
            int[,] indexMas = command;
            for (int i = 0; i <
kommandAmount; i++)
            {
                for (int j = 0; j <
maxPeopleInGroup; j++)
                {
                    indexMas[i,     j]     =
masStudent1[command[i,j], 3];
                }
            }
            return indexMas;
        }
    }
```

Модуль createGroup.cs

```
class createGroup
    {
        public                List<string[]>
createGroupList()
        {
            DBconnections    con    =    new
DBconnections();
            con.openConnection();
            string  sql  =  "SELECT  *  FROM
[dbo].[BelbinTestResult]                JOIN
[dbo].[student]
ON[dbo].[BelbinTestResult].[idUser]        =
[dbo].[student].[id]
WHERE[dbo].[student].[groupId]             =
"+dataClass.group;
            SqlCommand   sqlCommand   =   new
SqlCommand(sql, con.sqlConnection);
            SqlDataReader      reader      =
sqlCommand.ExecuteReader();
            List<string[]> dataPeople = new
List<string[]>();
            while (reader.Read())
            {
                dataPeople.Add(new
string[13]);
                dataPeople[dataPeople.Count
- 1][0] = reader[0].ToString();
                dataPeople[dataPeople.Count
- 1][1] = reader[1].ToString();
                dataPeople[dataPeople.Count
- 1][2] = reader[2].ToString();
                dataPeople[dataPeople.Count
- 1][3] = reader[3].ToString();
                dataPeople[dataPeople.Count
- 1][4] = reader[4].ToString();
                dataPeople[dataPeople.Count
- 1][5] = reader[5].ToString();
                dataPeople[dataPeople.Count
- 1][6] = reader[6].ToString();
                dataPeople[dataPeople.Count
- 1][7] = reader[7].ToString();
                dataPeople[dataPeople.Count
- 1][8] = reader[8].ToString();
                dataPeople[dataPeople.Count
- 1][9] = reader[9].ToString();
                dataPeople[dataPeople.Count
- 1][10] = reader[10].ToString();
                dataPeople[dataPeople.Count
- 1][11] = reader[11].ToString();
                dataPeople[dataPeople.Count
- 1][12] = reader[12].ToString();
            }
            reader.Close();
            con.closeConnection();
            return dataPeople;

        }
        public int[,] maxof3(List<string[]>
dataPeople)
        {
            int[,]      mas      =      new
int[dataPeople.Count, 4];
            int counter = 0, h1 = 0, h2 = 0,
h3 = 0;
            int countOfCharacteristic = 8;
            foreach    (string[]    s    in
dataPeople)
            {
                mas[counter,      3]      =
Int32.Parse(s[8]);
                h1 = 0; h2 = 0; h3 = 0;
                for  (int  i  =  0;  i  <
countOfCharacteristic; i++)
                {
                    if (Int32.Parse(s[i]) >=
10)
                    {
                        if       (h1       <
Int32.Parse(s[i]))
                        {
                            h1             =
Int32.Parse(s[i]);
                            mas[counter,  0]
= i + 1;
                        }
                        else
                        {
                            if     (h2     <
Int32.Parse(s[i]))
                            {
```

```
                            h2          =
Int32.Parse(s[i]);
                            mas[counter,
1] = i + 1;
                        }
                        else
                        {
                            if   (h3   <
Int32.Parse(s[i]))
                            {
                                h3     =
Int32.Parse(s[i]);
mas[counter, 2] = i + 1;
                            }
                        }
                    }
                }
            }
            counter++;
        }
        dataClass.mas = mas;
        return mas;
    }
    public int[,] createGroupMas()
    {
        List<string[]>   dataPeople   =
createGroupList();
        dataClass.dataPeople          =
dataPeople;
        int[,] mas;
        mas = maxof3(dataPeople);
        createCommand     a     =     new
createCommand();
       return          a.commandAlg(mas,
dataPeople.Count);
    }
}
```

Модуль DBconnections.cs

```
  class DBconnections
    {
        public SqlConnection sqlConnection =
new   SqlConnection(@"Data   Source=DESKTOP-
A7D0F6T\MSSQLSERVER01;              Initial
Catalog=diplomUser;             Integrated
Security=True");

        public void openConnection()
        {
            if    (sqlConnection.State    ==
System.Data.ConnectionState.Closed)
            {
                sqlConnection.Open();
            }
        }

        public void closeConnection()
        {
            if    (sqlConnection.State    ==
System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }

        public SqlConnection getConnection()
        {
            return sqlConnection;
        }
    }
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МІНІСТЕРСТВО ІНФРАСТРУКТУРИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ

ТЕЗИ
XV-ої Міжнародної науково-практичної конференції «СУЧАСНІ ІНФОРМАЦІЙНІ І КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ»
16-17 грудня 2021

Тезисы
XV-й Международной научно-практической конференции «СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ, В ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ»
16-17 декабря 2021

ABSTRACTS
of the XV-th International Conference «MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
16-17, December, 2021

Сучасні проблеми при мотивації працівників

Нуштаєв М. Ю., Шинкаренко В. І.,
Український державний університет науки і технологій, Україна

Працівники це головний актив компанії, без них неможливо виконати жодну роботу. Але без належної мотивації неможливо наладити ефективну працю і знайти працівників, що будуть якнайдовше залишатися з вами та з якими не буде проблем. Треба зауважити що одна з найголовніших проблем при поганій мотивації це великий плин кадрів, що понегативно позначиться на проекті над яким йде робота.

Спочатку треба зрозуміти на що впливає мотивація. По-перше це бажання працівника залишитися у фірмі. Це один з найголовніших факторів адже швидко замінити важливого працівника неможливо і кожна така заміна коштує значних втрат для компанії.

По-друге це ефективність роботи працівників. Мотивовані працівники будуть намагатися зробити роботу якомога швидше та краще, а інакше з'явиться проблема з якістю результатів роботи, що значно зашкодить як фірмі так і іншим співробітникам, а також погіршити мікроклімат у колективі.

Мотивацію треба розділяти за деякими ознаками: матеріальна чи не матеріальна, винагорода чи покарання. Основна складність полягає у тому, що неможливо використовувати лише один вид мотивації, а треба міксувати їх між собою.

Важливо пам'ятати що при нагороді чи покаранні працівник обов'язково повинен знати за що конкретно було це видано. Адже без чіткого зрозуміння що зробив працівник добре чи погано ні він ні його колеги не зрозуміють як покращити свою роботу чи які помилки не треба допускати. Також слід зазначити, що винагорода чи покарання повинні бути видані невідкладно, лише тоді вони будуть мати максимальний ефект та працівник краще зрозуміє причини які призвели до цього.

Умови праці сильно впливають на моральний стан працівників. Якщо у організації реалізуються пільги та гарантії соціального захисту встановлені законом чи перевищують це то для працівника мотивацією буде лише сам факт роботи у цій фірмі.

Працівника можна винагородити як і за допомогою грошей так і без неї. Однак треба одразу зауважити, що нематеріальна мотивація не є достатньо ефективною і її слід використовувати лише у комплекті з матеріальною. Наприклад можна видати працівнику премію, підвищити заробітну плату, або надати вихідний чи відпустку. Це лише найпростіші приклади, до більш неочевидних можна віднести оплату навчання співробітнику, допомогу з придбання житла та інше.

Мотивація у вигляді покарання є дешевою або навіть несе прибуток, однак є менш ефективною за винагороду. Також її переваження погіршує настрій у працівників та сприяє їх звільненню. Проте нетреба ігнорувати її взагалі, адже якщо працівники побачать, що за їх похибки немає покарання та продовжать їх робити. Спочатку завжди треба поспілкуватися з працівником і зрозуміти причину його помилки і тільки після цього назначати покарання, але частіше за все достатньо лише розмови. Покарання у матеріальному вигляді застосовуйте лише тоді коли дії працівника принесли фінансові втрати.

У підсумку можна зазначити що мотивація була різна і дуже багатогранна. На ефективність роботи людей у фірмі впливає велика кількість факторів які треба брати до уваги. Проте фірми що мають виважену мотивацію працівників показують неймовірні результати та рідше мають проблеми з термінами робіт та низьку текучість кадрів одночасно маючи гарних людей на більшості посад.

Гарною практикою є формування вмотивованості працівників за допомогою кваліфікованих психологів, які спираючись на особисті риси характеру працівників можуть значно підвищити його моральний, психологічний стан, що сприяє підвищенню мотивації.