ВАДИМ ГОРЯЧКІН
ОЛЕНА КУРОП’ЯТНИК

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Навчально-практичний посібник

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ

В. М. Горячкін, О. С. Куроп‘ятник

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

НАВЧАЛЬНО-ПРАКТИЧНИЙ ПОСІБНИК
ДЛЯ ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОС БАКАЛАВР

ДНІПРО
2026

УДК 004.41:378.147.091.33(075.8)
Г 71

Авторський колектив:
Горячкін В. М., Куроп'ятник О. С.

Рекомендовано Радою якості освітньої діяльності УДУНТ
Протокол № 9 від 19.05.2026 р.

Г 71 **Горячкін, В. М.** Інженерія програмного забезпечення : навч.-практич. посіб. для підготовки кваліфікаційної роботи на здобуття ОС Бакалавр / В. М. Горячкін, О. С. Куроп'ятник ; Укр. держ. ун-т науки і технологій. – Електрон. вид. – Дніпро : УДУНТ, 2026. – 99 с.

ISBN 978-617-8665-07-4 (PDF)

Навчально-практичний посібник охоплює основні аспекти підготовки, оформлення та захисту кваліфікаційної роботи для здобуття освітнього ступеня бакалавра зі спеціальності F2 (121) «Інженерія програмного забезпечення». Наведено рекомендації щодо обрання теми роботи, формулювання мети й постановки задачі до кваліфікаційної роботи, надано практичні поради та приклади опису результатів збору та аналізу вимог, проектування програмного забезпечення, розроблення (алгоритмізації та кодування), його тестування та налагодження. Надано короткі рекомендації щодо опису дослідницької частини роботи, а також підготовки супровідних матеріалів.

Для здобувачів ОС Бакалавр, наукових керівників та консультантів з підготовки кваліфікаційної роботи.

Іл. 11, табл. 27, бібліогр. 16 назв.

УДК 004.41:378.147.091.33(075.8)

ISBN 978-617-8665-07-4 (PDF)
DOI 10.15802/978-617-8665-07-4

ЗМІСТ

ВСТУП

Розвиток глобального інформаційного суспільства характеризується тотальною цифровізацією всіх сфер людської життєдіяльності. Програмне забезпечення перестало бути просто інструментом автоматизації - воно стало фундаментом критичної інфраструктури, економіки, медицини, освіти, інших галузей. У цих умовах роль фахівця, здатного не просто створювати код, а системно проєктувати складне та надійне програмне забезпечення, стає визначальною.

Підготовка фахівців з інженерії програмного забезпечення передбачає здобуття знань та набуття практичних навичок для підтримки усіх стадій життєвого циклу програмного забезпечення, де у фокусі не лише процеси, а й гарантування їх якості та якості кінцевого продукту.

Кваліфікаційна робота є завершеною самостійно виконаною працею здобувача вищої освіти, яка виконується на завершальному етапі навчання.

Метою роботи є демонстрація здобувачем набутої кваліфікації за певним освітнім ступенем, що передбачено освітньою програмою підготовки та стандартом відповідної спеціальності. Призначення кваліфікаційної роботи полягає у демонстрації та подальшому об’єктивному визначенні ступеня сформованості передбачених освітньою програмою компетентностей та оцінюванні рівня досягнення програмних результатів навчання здобувачем вищої освіти. Зокрема, оцінюється здатність здобувача розробляти компоненти програмного забезпечення з використанням сучасних парадигм програмування, проєктувати бази даних, проводити верифікацію та валідацію створеного програмного продукту. Важливим аспектом є також оцінка розуміння принципів безпеки, масштабованості та зручності використання розроблених програмних систем.

Процес підготовки і захисту кваліфікаційної роботи вимагає від здобувача вміння працювати з великими масивами науково-технічної інформації та здатності дотримуватися жорстких графіків виконання етапів розробки. Особлива увага при виконанні роботи приділяється інженерній культурі: чистоті коду, якості документації, використанню систем контролю версій, автоматизованих засобів тестування тощо. Це дозволяє переконатися, що випускник готовий до роботи в команді та здатний підтримувати професійні стандарти розробки на всіх етапах життєвого циклу програмного забезпечення.

Освітній ступінь бакалавра та відповідна кваліфікація присвоюється на основі публічного захисту кваліфікаційної роботи. Публічний захист є

відповідальним моментом, де здобувач має продемонструвати не лише здобуті компетентності, а й високий рівень комунікативних навичок. Вміння лаконічно презентувати складні технічні рішення, аргументовано відповідати на критичні зауваження та відстоювати власну професійну позицію є невід'ємною ознакою кваліфікованого фахівця.

Метою посібника є ознайомлення здобувачів освітнього ступеня бакалавра за спеціальністю F2 (121) «Інженерія програмного забезпечення» з особливостями підготовки кваліфікаційної роботи за цією спеціальністю, вимогами, що висуваються до таких робіт, процедурою захисту та надання рекомендацій щодо структури та змісту роботи.

Матеріал посібника розділено на логічні блоки, що відповідають етапам виконання кваліфікаційної роботи.

В першому розділі описана типова тематика робіт, визначено ролі учасників процесу та наведені етапи виконання кваліфікаційної роботи, що допоможе здобувачу ефективно спланувати роботу над нею.

Другий розділ регламентує формальні вимоги до кваліфікаційної роботи. Дотримання стандартів оформлення пояснювальної записки є обов'язковою умовою допуску до захисту.

Третій розділ висвітлює методику виконання окремих частин кваліфікаційної роботи та містить рекомендації щодо оформлення відповідних розділів пояснювальної записки. Він охоплює всі ключові аспекти життєвого циклу програмного забезпечення: збір та аналіз вимог, проєктування архітектури, інтерфейсів, фізичної структури бази даних, розроблення програмного коду, тестування та налагодження розробленого програмного забезпечення. Окремо розглядаються питання вибору засобів розробки та проведення експериментальних досліджень.

В четвертому розділі містяться практичні поради щодо підготовки до захисту кваліфікаційної роботи. Наведено рекомендації щодо візуалізації отриманих результатів при підготовці доповіді та демонстраційних матеріалів, що дозволять найбільш повно розкрити виконану роботу.

Дотримання рекомендацій, викладених у цьому посібнику, сприятиме підвищенню якості кваліфікаційних робіт та допоможе студентам при їх підготовці.

1 ОСНОВНІ ПОЛОЖЕННЯ

1.1 Тематика кваліфікаційних робіт

Тема кваліфікаційної роботи має відображати суть і спосіб розв’язання комплексного практичного завдання інженерії програмного забезпечення.

Перелік тем кваліфікаційних робіт розробляє випускова кафедра. Студент має право запропонувати на розгляд випускової кафедри власну тему кваліфікаційної роботи.

Кваліфікаційна робота передбачає розв’язання спеціалізованого завдання або практичної задачі інженерії програмного забезпечення, що характеризуються комплексністю та/або невизначеністю умов, із застосуванням сучасних теорій та методів інформаційних технологій на всіх стадіях розроблення програмного забезпечення (ПЗ).

Основою для вибору теми кваліфікаційної роботи є вимоги освітньої програми та Стандарту вищої освіти зі спеціальності F2 (121) «Інженерія програмного забезпечення».

В основу теми кваліфікаційної роботи може бути покладено розроблення прикладного, інструментального програмного забезпечення та/або автоматизація окремих видів діяльності підприємств:

– автоматизовані робочі місця та системи;
– управління технологічними процесами;
– системи автоматизації прийняття рішень;
– складні науково-технічні розрахунки.

Назва теми повинна бути конкретною й містити процедуру діяльності та/або назву продукту, який має бути отриманим.

При виборі теми кваліфікаційної роботи обов'язково повинні братися до уваги такі характеристики ПЗ, як об'єм і складність. Об'єм звичайно оцінюють кількістю самостійно написаних операторів (команд) тексту програми. Оцінка складності в більшості випадків суб'єктивна і ґрунтується на визначенні співвідношення витрат праці на проєктування і кодування програми.

В рамках підготовки кваліфікаційної роботи розробка має передбачати елементи всіх стадій життєвого циклу програмного забезпечення. Допускається виключення щодо розгортання та введення в експлуатацію. Кожна стадія повинна демонструвати обґрунтоване використання існуючих та/або самостійно розроблених чи реалізованих методів та алгоритмів, в тому числі з аналізом їхньої ефективності.

Темою кваліфікаційної роботи не може бути розробка, що не передбачає проєктування багаторівневої архітектури ПЗ.

Кваліфікаційну роботу не може бути присвячено виключно вивченню існуючого програмного забезпечення та/або введення його в експлуатацію.

При виборі теми кваліфікаційної роботи не слід ставити задачу доведення розроблюваного ПЗ до рівня експлуатації.

Кваліфікаційна робота може бути комплексною (кафедральною, міжкафедральною, міжуніверситетською) і виконуватися декількома студентами. Для комплексних робіт призначається головний керівник і керівники частин (складових теми).

За складністю завдання кваліфікаційна робота має бути адекватною освітньому ступеню бакалавра та кваліфікації, що здобувається, та має показати:

– спеціальну підготовку здобувача;
– уміння при вирішенні професійних завдань:
  – ставити проблеми, формулювати мету й завдання дослідження, розробляти план робіт з урахуванням термінів виконання кваліфікаційної роботи та принципів таймменеджменту;
  – працювати з літературними джерелами та фактичним матеріалом;
  – обґрунтовувати власні узагальнення і висновки.

Теми кваліфікаційних робіт із вказанням прізвищ та імен студента та керівника затверджуються наказом ректора у встановлені терміни.

1.2 Права та обов’язки здобувача вищої освіти та керівника кваліфікаційної роботи

На період підготовки кваліфікаційної роботи кафедра призначає здобувачу наукового керівника. Призначення наукового керівника здійснюється, за можливості, з урахуванням наукових інтересів здобувача за принципом спадковості виконання курсових і науково-дослідних робіт з обраної теми.

Під час підготовки кваліфікаційної роботи бакалавра здобувач має право на:

– внесення своїх пропозицій щодо обрання теми кваліфікаційної роботи та наукового керівника, враховуючи власні напрацювання;
– прийняття самостійних рішень щодо вибору методів дослідження та аналізу, а також обробки первинних матеріалів та документів;
– безпечні і нешкідливі умови навчання;
– безоплатне користування бібліотеками, інформаційними фондами, навчальною, науковою базами університету;

– участь у науково-дослідних роботах, конференціях, симпозіумах, виставках, конкурсах, представлення своїх робіт для публікації тощо.

При виконанні кваліфікаційної роботи здобувач зобов'язаний:

– запропонувати або обрати і узгодити з керівником тему роботи, отримати завдання на кваліфікаційну роботу;
– самостійно виконувати кваліфікаційну роботу, ґрунтуючись на матеріалах виробничих практик, методичного та інформаційного забезпечення;
– використовувати інформаційні джерела з дотриманням принципів академічної доброчесності;
– не допускати проявів академічної недоброчесності, зокрема академічний плагіат, фабрикацію, фальсифікацію, у тому числі з використанням засобів генеративного штучного інтелекту;
– систематично відвідувати консультації керівника кваліфікаційної роботи і консультантів розділів, враховувати зауваження та виконувати методичні вказівки керівників;
– згідно з графіком звітувань інформувати керівника про хід виконання кваліфікаційної роботи, інформувати наукового керівника про можливі відхилення від затвердженого графіку виконання роботи;
– відповідно до запланованих термінів подати кваліфікаційну роботу на перевірку керівнику роботи та консультантам розділів;
– підготувати доповідь про основні результати кваліфікаційної роботи, відповіді на зауваження керівника кваліфікаційної роботи, консультантів розділів;
– відповідно до розкладу роботи екзаменаційної комісії з захисту кваліфікаційних робіт.

До основних функцій керівника належать:

– практична допомога здобувачу при виборі теми кваліфікаційної роботи і її деталізації;
– видача завдання на виконання кваліфікаційної роботи;
– надання рекомендації щодо вибору джерел науково-технічної та спеціальної інформації;
– проведення науково-методичних консультацій щодо розкриття змісту роботи, її оформлення та презентації;
– систематичний контроль за ходом виконання окремих етапів затвердженого календарного плану виконання роботи;
– надання відгуку керівника щодо оцінки якості виконання роботи за встановленими вимогами [1].

Основні права керівника:

– вимагати від студента дотримання календарного плану виконання кваліфікаційної роботи;
– висловлювати зауваження до кваліфікаційної роботи загалом або окремих її частин та повертати роботу студенту на додаткове опрацювання;
– вимагати виправлення помилок змістового характеру, а також невідповідності чинному правопису та стандартам оформлення програмної документації, вимогам до оформлення кваліфікаційної роботи;
– приймати рішення про невідповідність кваліфікаційної роботи встановленим вимогам з інформуванням про це студента та завідувача випускової кафедри.

1.3 Етапи виконання кваліфікаційної роботи, контрольні строки та заходи

Після оприлюднення наказу про затвердження тем кваліфікаційних робіт для здобуття освітнього ступеня бакалавра здобувач повинен отримати у керівника роботи завдання на виконання кваліфікаційної роботи. Завдання має містити вихідні дані до кваліфікаційної роботи, вимоги щодо змісту пояснювальної записки та демонстраційних матеріалів, календарний план виконання роботи. Хід виконання кваліфікаційної роботи контролюється відповідно до встановлених термінів.

Результати виконання кваліфікаційної роботи мають бути представленні у вигляді пояснювальної записки та розробленої комп’ютерної програми.

Пояснювальна записка повинна містити відомості, що є результатом самостійного виконання кваліфікаційної роботи, обґрунтуванням ухвалених рішень, описом процесу розроблення програмного забезпечення, аналізом результатів налагодження, тестування і виконання програм. В записці не слід надавати відомі визначення, описи загальновідомих методів (достатньо посилання на джерела), не слід описувати методологію розроблення програмного забезпечення.

Оформлена пояснювальна записка до її публічного захисту підлягає обов'язковій перевірці нормконтролером кафедри [1, п. 5.4] та на наявність плагіату відповідно до встановленого порядку [2].

Публічному захисту кваліфікаційної роботи передує попередній розгляд роботи на засіданні кафедри, що передбачає бесіду з автором роботи у присутності керівника. Метою такого розгляду є визначення рівня готовності роботи та ступеня її відповідності встановленим вимогам. На підставі результатів розгляду ухвалюється остаточне

рішення про допущення кваліфікаційної роботи до захисту. Дата попереднього розгляду встановлюється кафедрою.

Перед публічним захистом здобувач повинен отримати відгук керівника щодо кваліфікаційної роботи.

Публічний захист кваліфікаційної роботи проводиться на відкритому засіданні екзаменаційної комісії. Тривалість захисту кваліфікаційної роботи бакалавра не повинна перевищувати 30 хвилин [1, п. 6.2]. Захист передбачає доповідь автора та відповіді на питання екзаменаційної комісії.

У своїй доповіді здобувач повинен обґрунтувати атуальність кваліфікаційної роботи, доповісти про її об’єкт і предмет, мету та завдання роботи, основні результати, викласти висновки і пропозиції. Доповідь має супроводжуватися презентацією, демонстрацією роботи розробленого ПЗ та включати коментар ілюстративних матеріалів. Рекомендована тривалість доповіді - від 7 до 10 хвилин.

Оцінювання бакалаврської роботи спирається на таке:

– глибина аналізу сучасного стану задачі інженерії програмного забезпечення, що розглядається у роботі, враховуючи наявні результати, представлені у вітчизняних та закордонних джерелах;
– методи та засоби інформаційних технологій, використані в роботі;
– достовірність результатів і обґрунтованість висновків;
– структурованість, стиль, мова і технічна грамотність викладення матеріалу.

1.4 Матеріали, що подаються до захисту

До захисту здобувачем мають бути надані такі документи та матеріали:

– заповнена залікова книжка, що містить підписи здобувача;
– пояснювальна записка;
– відгук керівника щодо кваліфікаційної роботи;
– мультимедійна презентація з висвітленням основних положень та результатів роботи;
– демонстраційне відео роботи програми.

У додатках до пояснювальної записки має бути представлено: технічне завдання на розроблену програмне забезпечення, текст програми, керівництво користувача (для всіх принципово різних ролей у програмі), за наявності – копії наукових публікацій (наукових статей, тез доповідей на конференціях), копії авторських свідоцтв на програму. Для кожної наукової публікації слід навести титульну сторінку видання, зміст і сторінки з текстом публікації.

1.5 Завдання та запитання до розділу 1

1. Перелічіть основні вимоги до теми кваліфікаційної роботи на здобуття ОС Бакалавр.
2. Елементи яких стадій життєвого циклу програмного забезпечення має передбачити розробка в рамках підготовки кваліфікаційної роботи? Чи допустимі виключення у стадіях, якщо так, то які?
3. Чи може кваліфікаційна робота бути присвяченою виключно вивченню існуючого програмного забезпечення та/або введення його в експлуатацію?
4. Які права та обов'язки здобувача при написанні кваліфікаційної роботи?
5. Хто з учасників навчального процесу має право приймати рішення про невідповідність кваліфікаційної роботи встановленим вимогам?
6. Що має містити завдання на кваліфікаційну роботу?
7. Перелічіть документи та матеріали, що подаються під час захисту роботи разом з пояснювальною запискою.
8. На чому ґрунтується оцінювання бакалаврської роботи?
9. Назвіть основні функції керівника.
10. Назвіть основні права керівника.
11. Які обов’язки покладаються на здобувача при виконанні кваліфікаційної роботи?
12. Що має відображати презентація для захисту роботи?
13. Що має бути представлено у додатках до кваліфікаційної роботи?
14. Яким чином визначається кількість документів виду “Керівництво користувача” для програмного забезпечення, яке було розроблено в рамках виконання кваліфікаційної роботи?
15. Якою є рекомендована тривалість доповіді здобувача на захисті роботи?

2 ВИМОГИ ДО СТРУКТУРИ ТА ОФОРМЛЕННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

2.1 Вимоги до структурних складових пояснювальної записки

Пояснювальна записка повинна містити:

– титульний аркуш українською та англійською мовами (окремі аркуші);
– завдання;
– відомість кваліфікаційної роботи;
– реферат;
– зміст;
– перелік умовних позначень, символів, скорочень і термінів (за необхідності);
– вступ;
– основні розділи:
  – збір та аналіз вимог;
  – проєктування та архітектура;
  – розроблення програми;
  – тестування та налагодження;
  – підготовка та проведення експериментів (для тем, що мають дослідницький характер);
– висновки та рекомендації;
– перелік посилань;
– додатки.

Кожен з основних розділів має містити пункт висновків.

Обсяг основних розділів повинен складати від 30 сторінок формату А4. Рекомендований обсяг кожного розділу основної частини – від 10 сторінок формату А4. Обсяг першого розділу, присвяченого збору та аналізу вимог, не повинен перевищувати 20% від загального обсягу основної частини пояснювальної записки.

Титульний аркуш містить (у такій послідовності):

– назву міністерства, закладу вищої освіти, факультету та кафедри, де виконано кваліфікаційну роботу бакалавра;
– тему роботи;
– назву освітньої програми;
– шифр і назву спеціальності;

– ім’я та прізвище автора роботи та його підпис;
– підписи, імена та прізвища керівника та нормоконтролера;
– місто і рік складання пояснювальної записки.

На титульному аркуші здобувач засвідчує особистим підписом відсутність запозичень з праць інших авторів без відповідних посилань.

Приклад оформлення титульного аркуша наведено в додатку А.

Після титульного аркушу слід розмістити титульний аркуш кваліфікаційної роботи перекладений англійською мовою без підписів. Приклад оформлення перекладу титульного аркуша англійською мовою наведено в додатку Б.

Завдання на кваліфікаційну роботу (додаток В) містить:
– тему роботи;
– перелік вихідних даних;
– перелік питань, які підлягають опрацюванню в кожному розділі роботи;
– перелік супровідних матеріалів: складових графічної та електронної частини (за необхідності);
– календарний план виконання роботи.

Завдання підписують керівник роботи, виконавець - здобувач освіти та затверджує завідувач випускової кафедри.

У ***відомості кваліфікаційної роботи*** перераховуються всі матеріали, які надані до її захисту: пояснювальна записка із зазначенням кількості аркушів; презентація; відеоролик, який демонструє роботу програми; комп’ютерні програми із зазначенням носія, на якому вони подані, тощо.

Реферат має бути стислим та інформативним. Реферат розташовується безпосередньо перед змістом і повинен містити:
– інформацію про обсяг пояснювальної записки (кількість сторінок), кількість ілюстрацій, таблиць, додатків, кількість джерел згідно з переліком посилань (відомості наводять, включаючи дані додатків);
– текст реферату;
– перелік ключових слів.

Послідовність викладу тексту реферату:
– об'єкт і предмет розроблення;
– мета роботи;
– методи розв'язання задачі;
– отримані результати.

Обсяг реферату – до 500 слів. Реферат повинен займати не більше ніж одну сторінку формату А4.

Реферат завершується комплексом ключових слів і сталих термінологічних словосполучень від 5 до 15 слів (словосполучень), що з позицій інформаційного пошуку мають той обсяг смислового навантаження, який чітко корелюється з основним змістом

кваліфікаційної роботи. Ключові слова наводять у називному відмінку в рядок через кому. Приклади оформлення реферату наведено в додатку Г.

Правила оформлення змісту. Зміст розпочинають з нової сторінки.

До змісту включають назви (вступ; послідовно перелічені розділи, підрозділи; висновки; перелік посилань; додатки) і номери сторінок, які містять початок матеріалу.

У змісті повинні бути назви структурних елементів (розділів, підрозділів тощо) щонайменше трьох рівнів, кожен з яких повинен розташовуватися з відступом (як багаторівневі списки). Текст назв не повинен розташовуватися під цифрами номерів структурних елементів та сторінок. Назви розділів та пунктів у змісті пишуться у форматі речення: перша буква велика, далі малі [3].

Перелік умовних позначень, символів, скорочень і термінів. Якщо в роботі вжито маловідомі скорочення, специфічну термінологію, позначення тощо, то їх перелік подається у вигляді окремого списку, який розміщується з нової сторінки після ЗМІСТУ, перед ВСТУПОМ. Незалежно від цього при першій появі цих елементів у тексті кваліфікаційної роботи наводять їх розшифровку.

Вимоги до розділів пояснювальної записки. Текст пояснювальної записки викладають українською мовою. Матеріал поділяють на розділи згідно із завданням.

Кожен розділ може поділятися на пункти або на підрозділи та пункти. Пункти, якщо це необхідно, поділяють на підпункти. Кожен пункт і підпункт повинен містити закінчену інформацію.

Суть розділів пояснювальної записки – викладання відомостей про предмет розроблення. Розділи пояснювальної записки повинні бути об'єднані загальною метою та органічно пов'язані між собою, містити необхідні посилання на відповідні джерела та елементи записки.

Пояснювальна записка не повинна містити матеріалів описового характеру, дублювання, стереотипних розв’язків, які не впливають на суть та висвітлення результатів кваліфікаційної роботи.

Цифровий матеріал, що розміщується в пояснювальній записці, рекомендується оформлювати у вигляді таблиць для підвищення наглядності та читабельності даних.

У тексті пояснювальної записки не допускається застосовувати:

– звороти розмовної мови, техніцизми, професіоналізми;
– довільні словотворення;
– для одного й того самого поняття різні науково-технічні терміни, які близькі за значенням (синоніми), а також іноземні слова й терміни за наявності рівнозначних слів і термінів в українській мові;
– скорочення слів, окрім тих, що встановлені стандартами та орфографією.

Вимоги до висновків. Заключним етапом є формулювання висновків, які повинні відображати те нове й суттєве, що визначає основні результати виконаної роботи.

Висновки розташовують після викладу розділів кваліфікаційної роботи, починаючи з нової сторінки.

У висновках може бути наведено оцінку одержаних результатів відносно аналогів, висвітлено досягнутий ступінь новизни, практичне, наукове значення результатів, прогнозні припущення про подальший розвиток предмета розроблення.

Текст висновків може поділятися на пункти.

Вимоги до переліку посилань. Перелік джерел, на які є посилання в роботі, наводять з нової сторінки.

Бібліографічні джерела в переліку посилань подають у порядку, за яким вони вперше згадуються в тексті пояснювальної записки або в алфавітному порядку.

Посилання на бібліографічні джерела в тексті кваліфікаційної роботи слід подавати у квадратних дужках [ ].

Список використаних джерел складають відповідно до чинних стандартів з бібліотечної та видавничої справи [4] (приклад див. Додаток Е). У списку літератури повинні переважати новітні видання (не старші 5 років). Рекомендована кількість використаних джерел для кваліфікаційної роботи бакалавра – від 15.

Вимоги до додатків. У додатках до пояснювальної записки необхідно навести технічне завдання та керівництво користувача (їх може бути декілька, якщо програмою будуть користуватися різні фахівці, які вирішують свої специфічні задачі).

У вказаних документах можуть бути свої додатки, у яких подають матеріал, необхідний для їх повноти і не може бути послідовно розміщений в основній частині через великий обсяг.

Додатки позначають великими літерами української абетки, починаючи з А, за винятком літер Ґ, Є, З, І, Ї, Й, О, Ч, Ь.

Додатки подаються в порядку посилань на них у тексті.

2.2 Основні вимоги до оформлення роботи

Структурні елементи кваліфікаційної роботи «РЕФЕРАТ», «ЗМІСТ», «ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ», «ВСТУП», «ВИСНОВКИ», «РЕКОМЕНДАЦІЇ» не нумерують, а їх назви правлять за заголовки структурних елементів.

Кваліфікаційну роботу слід оформлювати машинним способом і роздруковувати з використанням друкувальних і графічних пристроїв на

аркушах білого паперу формату А4 з однієї сторони. У разі потреби допускається використання аркушів формату А3. Кваліфікаційну роботу слід переплітати у тверду обкладинку з палітурного картону.

Текст основного матеріалу кваліфікаційної роботи повинен мати шрифт Times New Roman, розмір 14, нарис звичайний, колір чорний. Міжрядковий інтервал тексту встановлюють рівним 1,5 пунктів. Вирівнювання тексту – за шириною.

Розміри лівого, верхнього та нижнього полів кваліфікаційної роботи мають становити 20 мм, розмір правого поля – 10 мм. Абзацний відступ у кваліфікаційній роботі повинен бути однаковим упродовж усього тексту й дорівнювати 15…17 мм. Абзаци основного тексту не слід відокремлювати один від одного відступом або вільним рядком.

Розділи й підрозділи кваліфікаційної роботи повинні мати заголовки. Заголовки структурних елементів кваліфікаційної роботи й заголовки розділів розташовують посередині рядка й друкують великими літерами без крапки в кінці, використовуючи шрифт Times New Roman, розмір 14, нарис напівжирний. Перенесення частини слова не допускається. Від попереднього тексту заголовок слід відокремлювати інтервалом 12 пунктів, від наступного – 6 пунктів. Заголовки підрозділів кваліфікаційної роботи слід виконувати малими літерами (перша літера – велика) і розміщувати з абзацу. Використовують шрифт Times New Roman, розмір 14, нарис звичайний. Перенесення частини слова не допускається. Крапку наприкінці не ставлять. Від попереднього тексту заголовки підрозділів відокремлюється інтервалом 6 пунктів, відступу від першого рядка наступного тексту немає.

Нумерація сторінок основної частини кваліфікаційної роботи наскрізна. Першою сторінкою вважається титульний аркуш. Номер сторінки слід проставляти арабськими цифрами в правому верхньому куті сторінки без крапки в кінці.

Титульні аркуші українською та англійськьою мовами, завдання, відомість кваліфікаційної роботи та реферат включають до загальної нумерації пояснювальної записки, але номери сторінок не вказують. Першим пронумерованим листом є «ЗМІСТ».

Кожний розділ слід починати з нової сторінки. Підрозділи нумерують арабськими цифрами в межах кожного розділу. Номер підрозділу складається з номера розділу й номера підрозділу, розділених крапкою. Наприкінці номера підрозділу крапка не ставиться.

Рисунки й таблиці, які розташовують на окремих сторінках, слід включати до загальної нумерації.

Ілюстрації (креслення, рисунки, графіки, схеми, діаграми, фотографії, зокрема копії з екрана монітора комп’ютера) розміщують відразу після посилання на них у тексті або на наступній сторінці. Ілюстрацію слід розміщати так, щоб її можна було розглядати без повороту

кваліфікаційної роботи. Якщо таке розміщення неможливе, то ілюстрацію необхідно розташовувати так, щоб для її розгляду достатньо було повернути пояснювальну записку на чверть оберту за рухом годинникової стрілки.

Ілюстрації повинні мати назву, яку слід розташовувати під ілюстрацією. У разі потреби під ілюстрацією можна розміщувати пояснювальні дані (підрисунковий текст). Ілюстрації слід позначати словом «Рисунок», яке разом із номером та підписом розміщують після пояснювальних даних, наприклад: «Рисунок 4.2 – Схема алгоритму». Слово «Рисунок», номер і підпис повинні бути поза ілюстрацією, мати шрифт Times New Roman, розмір 14, нарис звичайний, вирівнювання по центру, інтервал перед рисунком встановлювати 6 пунктів, після – 12 пунктів. Ілюстрації слід нумерувати послідовно в межах розділу арабськими цифрами. Номер ілюстрації складається із номера розділу й порядкового номера в межах розділу, розділених крапкою. На всі ілюстрації повинні бути посилання в тексті. При посиланні слід вказувати повний номер ілюстрації. Повторні посилання варто давати зі скороченими словами «дивись рисунок», наприклад: див. рис. 2.3.

Таблицю слід розташовувати безпосередньо після тексту, у якому вона згадується вперше, або на наступній сторінці. На всі таблиці мають бути посилання в тексті. Таблиці нумерують арабськими цифрами в межах розділу. Номер таблиці складається з номера розділу й порядкового номера таблиці в ньому, розділених крапкою. Таблиця повинна мати стислу змістовну назву, наприклад: «Таблиця 3.1 – Характеристики технічних засобів». Назву таблиці й заголовки стовпців слід починати з великої літери, підзаголовки – з малої, якщо вони складають одне речення із заголовком, і з великої, якщо вони самостійні. У кінці заголовків і підзаголовків таблиці крапка не ставиться. Заголовки й підзаголовки стовпців слід указувати в однині. У разі переносу частини таблиці на наступну сторінку заголовки стовпців таблиці слід повторювати й над нею з вирівнюванням по правому полю поміщати слова «Продовження таблиці» з указівкою номера. Заголовок таблиці повторювати непотрібно.

Формули та рівняння слід розміщувати безпосередньо після тексту, в якому вони згадуються, окремими рядками, з вирівнюванням по центру. До і після формули залишають один вільний рядок. Після формул слід ставити той розділовий знак, який необхідний, виходячи з побудови фрази. Розділові знаки ставлять безпосередньо за формулами до номера. Формули та рівняння нумерують за порядком у межах розділу. Номер формули або рівняння складається з номера розділу й порядкового номера формули або рівняння в межах розділу, розділених крапкою. Номер слід записувати в круглих дужках на рівні формули або рівняння у крайньому правому положенні на рядку. Посилаючись у тексті на

формулу або рівняння, потрібно зазначати в дужках повний номер. Переносити формулу чи рівняння на наступний рядок допускається тільки на знаках виконання операцій, повторюючи знак операції на початку наступного рядка. Знак множення при цьому записують у вигляді «×».

Пояснення позначень і числових коефіцієнтів, які входять до формули чи рівняння, слід наводити безпосередньо під формулою в тій послідовності, у якій вони вказані у формулі або рівнянні. Пояснення значення кожного символу й числового коефіцієнта потрібно подавати з нового рядка. Перший рядок пояснення слід починати словом «де» без двокрапки.

Цифри у формулах повинні мати прямий шрифт. Літерні позначення величин (символи), для яких застосовуються букви латинського алфавіту, повинні мати курсивний шрифт. Скорочені математичні терміни (наприклад, sin, arcsin, lg, lim, const, max) у формулах та рівняннях потрібно записувати прямим шрифтом. Скорочені найменування фізичних і технічних одиниць виміру, метричних мір і похідних від них, які позначені буквами українського алфавіту, слід наводити курсивним шрифтом без крапок. Скорочення в індексах повинні мати прямий шрифт.

Розмірність одного й того самого параметра в межах кваліфікаційної роботи повинна бути однаковою. Математичні знаки слід застосовувати лише у формулах. У тексті їх пишуть словами. Не слід вживати математичні знаки без цифр. Позначення одиниць належить застосовувати тільки після числового значення розмірів та в заголовках граф і найменуваннях рядків таблиць. У тексті (без числового значення розмірів) слід наводити повне найменування одиниць. Не дозволяється застосовувати різні позначення однакових фізичних величин.

Примітки потрібно наводити, якщо необхідні пояснення до змісту тексту, таблиць або графічного матеріалу. Примітку слід розташовувати безпосередньо після матеріалу, якого вона стосується, друкувати шрифтом Times New Roman, розміром 12, нарис звичайний, вирівнювання за шириною з абзацним відступом, інтервал перед приміткою – 6 пунктів, після – 12 пунктів. Слово «Примітка» слід набирати розрідженим шрифтом із розрядкою 1 пункт.

2.3 Завдання та запитання до розділу 2

1. Назвіть вимоги до структурних складових пояснювальної записки.
2. Опишіть правила оформлення змісту та розділів пояснювальної записки.
3. Назвіть вимоги до надання посилань на ілюстрації, таблиці та формули (рівняння).

4. Вкажіть вимоги до списку використаних джерел та посилань на його елементи в тексті пояснювальної записки
5. Як утворюються номери розділів та підрозділів пояснювальної записки?
6. За якими правилами слід оформлювати рисунки у тексті пояснювальної записки?
7. Розкажіть про правила запису формул у тексті документа.
8. Наведіть приклади запису формули.
9. Які існують правила іменування додатків?
10. Які існують обмеження та на розмір кожного з основних розділів пояснювальної записки?
11. Яким є рекомендований обсяг розділів основної частини пояснювальної записки?
12. Складіть перелік елементів відомості кваліфікаційної роботи.
13. Назвіть складові тексту реферату до кваліфікаційної роботи.

3 ВИМОГИ ДО ЗМІСТУ РОЗДІЛІВ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

3.1 Вступ

Вступ (не більше ніж 5 сторінок) розкриває сутність задачі, що вирішується, її значущість тощо.

Вступ повинен містити:

– актуальність роботи;

– мету роботи;

– експлуатаційне призначення розробки.

Актуальність роботи – обов'язкова вимога до будь-якої роботи. У цій частині вступу через критичний аналіз та порівняння з відомими розв'язаннями проблеми обґрунтовують актуальність та доцільність роботи. Обґрунтування актуальності обраної теми роботи характеризує професійну підготовленість здобувача.

Висвітлення актуальності повинно бути небагатослівним. Достатньо коротко викласти:

– сутність та значення проблематики задачі, що вирішується у роботі;

– доцільність роботи, її відмінність порівняно з відомими розв'язаннями проблеми.

Мета роботи повинна відображати напрям кваліфікаціної роботи та вказувати на очікуваний кінцевий результат. Не слід формулювати мету як «Дослідження …», «Вивчення…», тому що таким чином визначається процес досягнення мети, а не сама мета.

Експлуатаційне призначення повинно відображати практичне застосування розробленого ПЗ, а саме для кого вирішувалася наведена задача, визначає нематеріальну вигоду, яку отримаю кінцевий користувач в результатів впровадження та використання розробленого ПЗ

3.2 Збір та аналіз вимог

Метою збору та аналізу вимог є розгорнуте визначення функціональних та нефункціональних вимог, що очікуються в ПЗ, яке буде реалізовано.

Розділ пояснювальної записки, присвячений збору та аналізу вимог, має містити коротку інформацію щодо обраних джерел вимог та методів їх обробки, основну інформацію, отриману з джерел, яка є корисною для подальшої розробки ПЗ, та її аналіз. Останнім пунктом перед висновками

цього розділу має бути постановка задачі, що містить перелік завдань, які слід виконати для досягнення мети, а також функціональне призначення розроблюваного ПЗ.

3.2.1 Класифікація вимог та їх джерела

Вимоги до програмного забезпечення – це перелік властивостей та функцій ПЗ, яке буде розроблено або знаходиться у розробці, а також даних щодо його якості та надійності.

Вимоги за характером поділяються на:

1) функціональні – до поведінки системи;
2) нефункціональні – до характеру поведінки системи:
   - бізнес-правила – визначають обмеження, що витікають з предметної області;
   - системні вимоги – вимоги до програмного та апаратного забезпечення, що має бути надано для роботи програми, яка розробляється;
   - атрибути якості;
   - зовнішні системи та інтерфейси.

В ході виконання кваліфікаційної роботи слід сформувати вимоги до розроблюваного ПЗ. Для кращого розуміння предметної області та потреб потенційних користувачів перед формуванням вимог доцільно обрати джерела, вивчення яких нададуть корисну інформацію.

Джерелами вимог є:

- документація (законодавство, стандарти, нормативна література тощо);
- бізнес-процеси (вимоги, що обумовленні процесами виробництва, наданням послуг і таке інше);
- люди (очікування та бачення користувачів системи, замовників та інших зацікавлених сторін);
- існуючі повні та часткові аналоги.

Слід критично оцінити доступність джерел та обрати ті, доступ до яких можна отримати протягом часу, передбаченого календарним планом виконання кваліфікаціної роботи. При обранні джерел також слід виходити з їх інформативності (обирати джерела з низьким вмістом загальновідомих положень), актуальності та достовірності.

При роботі з документацією перевагу слід надати офіційним (якщо мова йде про нормативні документи) та науковим (статтям, монографіям, матеріалам науково-практичних конфренецій тощо) виданням.

При роботі з бізнес-процесами корисним є моделювання процесів «as-is» (як є зараз) та «to-be» (як буде після впровадження вашого ПЗ), використовуючи нотації Business Process Model and Notation (BPMN) [5, с. 23 - 53] або UML Activity Diagrams [6]. Це допоможе чітко виділити

автоматизовані функції та уникнути розривів у логіці, коли програма не відповідає реальному ланцюжку дій користувача.

При роботі з людьми слід чітко розділити ролі: замовник (наприклад, керівник роботи) та кінцевий користувач. Доречним є проведення короткого інтерв'ю або бесіди з замовником, анкетування – з кінцевими користувачами. Це допоможе виявити «приховані» очікування, оскільки те, що здається очевидним здобувачу-розробнику, може бути критичним бар'єром для реального користувача.

При роботі з аналогами слід провести їх оцінювання за обраними критеріями. Слід не лише перерахувати їхні назви, а й акцентувати увагу на недоліках, які може бути усунено для підкреслення практичної цінності роботи.

3.2.2 Методи збору вимог

Методи збору вимог – це сукупність технік, що застосовуються для отримання максимально повної інформації про розроблювану систему з обраного джерела. У пояснювальній записці необхідно не просто перерахувати існуючі методи, а вказати, які саме з них були застосовано та яких результатів вдалося досягти за їх допомогою.

Під час підготовки кваліфікаційної роботи зазвичай використовують такі методи збору вимог як анкетування, огляд аналогів та літератури.

Спілкування з зацікавленими сторонами.

Зацікавлені сторони – це особи чи організації, які мають дійсний інтерес до системи. Вони можуть впливати на неї прямо чи опосередковано.

У рамках кваліфікаційної роботи зацікавленими сторонами є: розробник – здобувач освіти, керівник роботи, потенційні кінцеві користувачі. Останні визначаються відповідно до теми роботи.

Для збору вимог проводять опитування зацікавлених сторін. Для цього можуть бути проведені інтерв'ю, бесіди, анкетування тощо. Анкетування є найбільш універсальним способом опитування, який дозволяє охопити значну кількість респондентів та мінімізувати витрати часу на організацію та обробку результатів.

При розробці прикладного ПЗ до такого опитування здобувач може запрошувати широке коло людей: своїх знайомих, інших здобувачів спеціальності та викладачів випускової кафедри, факультету, університету тощо.

Анкета – це структурована послідовність питань, що передбачає отримання відповідей на певний перелік питань у короткій формі. Для збільшення лояльності респондентів, підвищення їх довіри до коректності подальшої обробки отриманих даних рекомендовано на початку анкети розмітити вступ з інформацією про мету опитування та те, хто його проводить. Анкетування варто проводито анонімно.

Основними видами питань у анкеті є: відкриті, закриті, напівзакриті.

Відкрите питання передбачає відповідь у вигляді одного словосполучення чи речення. Закриті питання надають можливість обрати одну чи декілька відповідей з переліку. Напівзакриті питання передбачають перелік відповідей та поле «інше», в якому респондент може дати словесну відповідь. Анкетування спрощує процес опитування та обробки результатів, але обмежує думку опитуваного рядом запропонованих відповідей. Наявність варіанту типу «інше» без розшифрування зменшує ясність відповіді, з полем для розшифрування – ускладнює обробку відповіді.

До анекети може бути включено блоки питань про:

– соціально-демографічні показники: питання щодо віку користувача, навичок, освіти, наявності особливих потреб тощо, якщо відповідні дані мають вплив на особливості розроблення інтерефейсу користувача, додавання спеціальних функцій ПЗ;
– досвід: питання, які визначають чи має користувач досвід роботи з аналогами та якими саме - є корисним для переймання окремих рішень та є підказкою, які аналоги слід розглянути в рамках етапу збору вимог;
– функціональні вимоги: питання щодо необідних функцій ПЗ, формату вхідних та вихідних даних;
– нефункціональні вимоги, у тому числі вимоги до інтерфейсу користувача та порядку використання ПЗ.

При формуванні альтернатив відповідей до питань слід уникати понять, пов’язаних з інтуїцією, простотою без пояснення їх змісту, уникати назв відтінків кольорів (при з’ясуванні вимог до інтерфейсу користувача).

В якості інструменту для проведення анкетування рекомендується використовувати Google або Microsoft Forms.

Розглянемо приклад збору вимог до застосунку «Фітнес-трекер». Функціональне призначення застосунку полягає у накопиченні даних про показники фізично активності людини та веденні відповідної статистики.

Як джерело вимог розглянемо зацікавлених сторін, а саме – кінцевих користувачів застосунку. Для збільшення охоплення респондентів виконаємо анкетування за допомогою Google Forms.

Розробимо перелік запитань для форми.

Для даного застосунку відсутні обмеження користувачів за віком, статтю та іншими демографічними показниками. Також їхні значення не впливають на наявність потенційних функцій у застосунку та нефункціональні вимоги до нього, тому анкета не міститиме демографічного блоку.

Розглянемо блок питань щодо функціональних вимог (табл. 3.1)

Таблиця 3.1

Запитання для збору функціональних вимог

№ з/п	Тип питання	Текст питання	Варіанти відповіді
1.	Закрите, просте альтернативне	Чи маєте досвід користування фітнес-трекером?	Так Ні
2.	Відкрите	Досвід користування якими фітнес-трекерами маєте?	
3.	Закрите, з вибіркової підмножиною	Які показники фізичної активності Ви бажаєте відслідковувати?	Кількість кроків. Пройдена дистанція. Спалені калорії. Час активності (хвилини).
4.	Закрите, з вибіркової підмножиною	Які серцево-судинні показники Ви бажаєте відслідковувати?	Пульс. Веріабельність серцевого ритму (показник стресу та готовності до навантажень). Рівень кисню в крові. Артеріальний тиск.
5.	Закрите, просте альтернативне	Чи бажаєте відслідковувати перебіг Вашого сну?	Так. Ні.
6.	Закрите, з вибіркової підмножиною	Які додаткові показники Ви б хотіли відслідковувати?	Температура тіла. Рівень шуму навколо. Перепади висот під час руху. Жодних додаткових.
7.	Закрите, просте альтернативне	Чи необхідна функція експорту даних у окремі файли?	Так. Ні.
8.	Закрите, зі шкалою Лайкерта	Для збільшення мотивації до фізичної активності слід внести у застосунок елементи гейміфікаціїї	Цілком не погоджуюсь. Скоріше не погоджуюсь. Не можу сказати. Скоріше погоджуюсь. Цілком погоджуюсь.

Відповіді на запитання 1 та 2 необхідні для того, аби розробник ПЗ мав змогу ознайомитися з аналогами, які є популярними серед

респондентів. Це стане додатковим джерелом для збору та аналізу вимог для розроблюваного ПЗ. Запитання 2 стає доступне респондентові лише при позитивній відповіді на питання 1.

Розробимо перелік запитань щодо нефункціональних вимог (табл. 3.2)

Таблиця 3.2

Запитання для збору нефункціональних вимог

№ з/п	Тип питання	Текст питання	Варіанти відповіді
1	2	3	4
1.	Напівзакрите, з декількома варіантами відповідей	Чи потрібна у застосунка додаткова мова інтерфейсу (крім української)?	Англійська. Німецька. Іспанська. Китайська. Інша (вказати).
2.	Закрите, просте альтернативне	Чи потрібні у застосунку аудіо-підказки?	Так. Ні.
3.	Закрите, з використанням шкали важливості	Чи важливо для Вас, щоб аудіо-підказки були озвучені саме обраною мовою?	Дуже важливо. Бажано. Не важливо.
4.	Закрите, з декількома варіантами відповідей	Яка система вимірювання є зручною для відображення фізичних показників Вашої активності?	Метрична система (метри, кілометри, кілограми). Імперська система (фути, милі, фунти) Можливість налаштувати формат для кожного показника окремо
5.	Закрите, просте альтернативне	Чи важлива для Вас підтримка вибору світлої/темної теми інтерфейсу?	Так. Ні.
6.	Напівзакрите, з вибірковою підмножиною	Які пристрої ви плануєте підключати до фітнес-трекера?	Смарт-годинник. Нагрудний пульсометр. Розумні ваги. Бездротові навушники. Фітнес-браслет. Інше (вказати).

1	2	3	4
7.	Закрите, з декількома варіантами відповіді	Який максимальний час очікування після натискання кнопки «Старт тренування» ви вважаєте допустимим?	До 1 секунди. До 3 секунд. До 5 секунд.
8.	Закрите, просте альтернативне	Чи плануєте ви використовувати додаток спільно з іншими сервісами, такими як Apple Health, Google Fit?	Так. Ні.
9.	Закрите, просте альтернативне	Чи є для вас прийнятним використання хмарного сховища для історії тренувань, що потребує інтернет-з'єднання?	Так. Ні.
10.	Відкрите	Як система має реагувати на критично низький заряд батареї (менше 5%) під час активного запису тренування?	
11.	Закрите, просте альтернативне	Якому типу оновлення застосунку Ви віддаєте перевагу?	Автоматичному. Ручному.
12.	Закрите, з декількома варіантами відповіді	Як часто ви готові оновлювати додаток для отримання нових функцій та покращення безпеки?	Щотижня. Один раз на місяць. Два рази на місяць.

Запитання 12 стає доступне респондентові лише при відповіді "Ручному" на питання 11.

Огляд аналогів — це процес детального вивчення та аналізу дослідження програмних продуктів, які вирішують ті самі або схожі завдання (повністю або частково) в обраній предметній області. Цей етап є важливим для аналізу вимог, оскільки дозволяє виконати оцінювання переваг та недоліків існуючих рішень.

Огляд аналогів дозволяє:

- виявити недоліки рішень, прийнятих в аналогах (наприклад, складність інтерфейсу користувача, відсутність необхідного функціоналу тощо);
- визначити основні вимоги до розроблюваного ПЗ;

– обгрунтувати вибір прийнятих рішень на основі існуючого досвіду користувачів;
– визначити конкурентні переваги розроблюваного ПЗ.

Процес аналізу аналогів включає:

1) вибір найбільш релевантних продуктів (це можуть бути як повні аналоги, так і системи з частковим збігом функцій);
2) визначення критеріїв порівняння (критерії мають бути важливим для кінцевого користувача або замовника системи і повинні базуватися на функціональних та нефункціональних вимогах, таких як повнота необхідного функціоналу, зручність інтерфейсу користувача, швидкодія тощо);
3) виконання порівняльного аналізу обраних аналогів за кожним з критеріїв.

Результати огляду аналогів доцільно представити як їх порівняння між собою за сформовними критеріями, наведеними у вигляді таблиці (табл. 3.3), в якій в першому стовпці вказується назва критерію, за яким здійснюється порівняння, а в наступних - результати аналізу за даним критерієм кожного з обраних аналогів.

Таблиця 3.3

Приклад наведення результатів аналізу аналогів

Критерій порівняння	Аналог 1	Аналог 2	...	Аналог N
Критерій 1				
Критерій 2				
...				
Критерій M				

Аналіз аналогів повинен закінчуватися висновками щодо того, чому розглянуті аналоги не повністю задовольняють потреби зацікавлених сторін.

Огляд літератури передбачає теоретичне дослідження предметної області та технологій розробки ПЗ. Здобувач має продемонструвати вміння працювати з літературними джерелами, критично оцінювати публікації та виділяти ключові аспекти, необхідні для розробки.

В процесі огляду літератури може бути розглянуто:

– наукові та фахові публікації (книги, статті, монографії, матеріали конференцій);
– нормативну документацію та стандарти, такі як ДСТУ, ISO/IEC (зокрема щодо життєвого циклу та якості ПЗ), галузеві стандарти;
– технічну документацію (офіційні керівництва до мов програмування, програмних засобів, фреймворків, бібліотек, СКБД, документацію до API сторонніх сервісів тощо).

У пояснювальній записці не слід просто перераховувати назви літературних джерел. Необхідно надати стислу інформацію про методи, моделі та засоби, що використовуються для вирішеня поставлених або аналогічних завдань у предметній області, а також їх аналіз з зазначенням переваг та недоліків. Всі літературні джерела повинні бути належним чином оформлені у списку використаних джерел та мати відповідні посилання в тексті роботи.

Для кращого розуміння різниці між формальним переліком джерел та якісним аналітичним оглядом, наведемо приклади невдалого та вдалого підходів до огляду літературних джерел.

Невдалий приклад огляду літератури:

«У книзі Мартіна Р. «Чиста архітектура» [5] описано принципи побудови програмних систем. Також я розглянув статтю про мікросервіси [8], де автор каже, що це сучасно. У документації до фреймворку React [12] написано, як створювати компоненти. Усі ці джерела корисні для моєї роботи та допомогли мені в розробці».

Наведений текст не містить аналізу. Автор просто констатує факт існування джерел, не пояснюючи, які саме ідеї були запозичені, як вони корелюють між собою та як саме вони були використані для розв'язання задачі, поставленої у кваліфікаційній роботі.

Вдалий приклад:

«Аналіз сучасних підходів до побудови архітектури високонавантажених систем, проведений на основі праць [5, 6] дозволяє виділити два панівних підходи, які є найбільш часто застосованими: монолітний та мікросервісний. Мікросервісна архітектура забезпечує кращу масштабованість [8], проте, згідно з технічними рекомендаціями розробників фреймворку Spring [14], вона суттєво підвищує складність розгортання та моніторингу системи.

Порівняльний аналіз, наведений у таблиці 1.1, показує, що для проєктів із обмеженими обчислювальними ресурсами, яким є об’єкт цієї роботи, оптимальним є використання «чистої архітектури» з чітким розділенням рівнів доступу до даних та бізнес-логіки. Це підтверджується також стандартами ISO/IEC 25010 щодо підтримуваності програмного забезпечення».

У цьому прикладі виконано порівняння різних точок зору, виділено переваги та недоліки, наведено посилання на стандарти та на основі проаналізованих джерел обґрунтовано вибір підходу, використаного здобувачем.

3.2.3 Формулювання постановки задачі

Формулювання постановки задачі є важливим етапом, яке важливим для формування технічного завдання на розробку ПЗ. Це логічний перехід від загальної мети проєкту до конкретного плану дій. Текст має бути лаконічним, технічно грамотним і позбавленим абстрактних описів, зосереджуючись на тому, що саме буде створено.

Починати цей пункт пояснювальної записки варто з функціонального призначення ПЗ. Тут необхідно чітко визначити роль системи: які бізнес-процеси вона автоматизує, для якої цільової аудиторії призначена та в якому середовищі функціонуватиме. Замість загальних фраз варто використовувати термінологію предметної області (наприклад, «автоматизація обліку складських залишків» або «моніторинг мережевого трафіку в реальному часі»), щоб одразу окреслити практичну цінність розробки.

Далі постановка задачі розгортається у вигляді деталізованого переліку завдань, які необхідно розв’язати для досягнення мети.

Завдання можуть бути розподілено на блоки проєктування, реалізації, тестування і налагодження ПЗ.

У блоці проєктуванні слід вказати на необхідність розробки архітектури ПЗ на рівні статичної моделі класів та/або компонентів системи, структури бази даних, проектування інтерфейсів користувача (UI/UX) та побудови логіки взаємодії компонентів системи. Важливо акцентувати увагу на специфічних інженерних аспектах, зокрема забезпечення масштабованості, безпеки даних або інтеграції зі сторонніми сервісами через API.

Блок реалізації передбачає опис завдань розроблення алгоритмів, вибору парадигми та мови програмування, середовища розроблення програми та інших технічних засобів.

У блоці тестування та налагодження ПЗ доцільно привести завдання з проєктування та реалізації тестових випадків, виправлення виявленних дефектів. Тестові випадки мають забезпечувати перевірку ПЗ на відповідність сформульованим функціональним вимогам. Це може бути юніт-тестування, навантажувальне тестування або проведення експериментальних досліджень (наприклад, порівняння швидкодії алгоритмів чи точності моделей класифікації тексту).

Завершувати постановку задачі рекомендується коротким описом очікуваного результату як цілісного програмного продукту із супровідною документацією, що розв'язує конкретну прикладну чи науково-технічну проблему. Такий структурний підхід дозволяє чітко оцінити обсяг виконаної роботи та відповідність кваліфікаційному рівню бакалавра.

3.3 Проєктування та архітектура

Проєктування програмного забезпечення складається з двох частин: зовнішнього та внутрішнього. Під зовнішнім проєктуванням розуміють процес опису функцій проєкту й очікуваної поведінки продукту, що розробляється, з погляду зовнішнього по відношенню до нього спостерігача-користувача. Мета процесу - визначення взаємодій майбутнього програмного продукту із зовнішнім середовищем (зазвичай з користувачем) без конкретизації його внутрішньої структури (будови).

Зовнішнє проєктування ПЗ виражається у формі зовнішніх специфікацій, призначених для широкої аудиторії, включаючи користувача (для перевірки та схвалення), авторів документації для користувача, всіх програмістів, що беруть участь у проєкті, а також усіх тих, хто займатиметься тестуванням продукту.

Результатом має бути документ з логічною організацією змісту.

В кваліфікаційній роботі слід виконати зовнішнє проєктування, визначивши функціональні вимоги та вимоги до вхідних і вихідних даних. Якщо планується використання визначених методів для розв'язання поставлених задач, то доречним в даному пункті вказати їх назви, надати посилання на відповідні літературні джерела.

Для вхідних та вихідні даних слід зазначити їх найменування, суть, формат, в тому числі область допустимих значень.

У цьому розділі пояснювальної записки може бути наведено опис зовнішнього інформаційного середовища, в якому буде експлуатуватися ПЗ. Тоді слід визначити всі канали вводу-виводу та інформаційні об’єкти, з якими воно взаємодіє (перелік програм, що необхідні для функціонування проєктованої).

Специфікацію функціональних вимог слід представити у вигляді діаграми прецедентів [6]. На діаграмі графічно показана сукупність прецедентів (можливих варіантів використання програми) та суб’єктів (акторів), а також відношень між ними.

При побудові діаграми прецеднтів слід відштовхуватися від такого: кожній виділеній функціональній вимозі має відповідати основний (з’єднаний з актором асоціацією) або додатковий (приєднаний за допомогою зв’зяку розширення або включення) прецедент.

При наявності кількох суб’єктів, що мають різні ролі користувачів системи (користувачів з різними задачами), доцільним є побудова окремих діаграм для кожного суб’єкта.

Внутрішнє проєктування полягає у визначенні складових системи (програми), яка розробляється, їх структури та поведінки. В рамках даного проєктування може бути визначено:

– статичну архітектуру системи: модулі, їх структуру, відповідальність та взаємозв’язок;

– поведінку системи: взаємодію модулів системи з уточненням повідомлень, які передаються, життєвого циклу створених об’єктів, станів об’єктів;
– топологію системи (фізичне розміщення системи та її частин);
– структуру (логічну схема) баз та сховищ даних;
– особливості інтерфейсу користувача: набір апаратних та програмних засобів, які будуть використовуватися для взаємодії з програмою, сценарій та структура діалогу, візуальне атрибути інтерфейсу тощо.

3.3.1 Проєктування архітектури системи

Під архітектурою системи розуміють набір модулів та їх зв’язок між собою. Виходячи з позицій об’єктно-орієнтованого програмування (ООП), модулем може бути клас та інтерфейс. З точки зору інших парадигм програмування модулями можуть виступати функції, набори функцій, агенти тощо.

Для специфікації архітектури рекомендовано застосування UML - уніфікованої мови моделювання [7]. Моделювання засобами UML є процесом спуску від абстрактної моделі до логічної та фізичної. Задача моделюється за допомогою набору діаграм, кожна з яких представляє собою певну проекцію системи.

Для опису статичної структури системи доцільно використовувати діаграму класів. Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування [6].

Об’єктно-орієнтовне програмування базується на представленні програми у вигляді сукупності об’єктів, які взаємодіють між собою. Ключовим поняттями є клас, об’єкт, атрибут, метод.

Клас – це опис множини об’єктів з однаковим набором атрибутів, операцій, зав’язків та семантикою. У програмуванні клас – це структурований тип даних, який створений користувачем, являє собою сукупність атрибутів (даних) та методів їх обробки. Атрибути – це змінні, які входять до класу та описують його суть. Методи (іноді – операції) – функції для роботи з об’єктами. Об’єкт – це екземпляр класу (створюється після запуску програми і ініціалізації атрибутів класу).

Діаграму класів можна розглядати з таких точок зору:
– концептуальної – представлення понять досліджуваної предметної області, що розглядаються незалежно від мови програмування;
– специфікації – розглядається програмна система на рівні інтерфейсів (опису задач), без реалізації;
– реалізації – розглядаються класи на рівні реалізації конкретної мови.

Моделювання системи можна розбити на кілька етапів, а саме:

1. Моделювання словника системи:

– ідентифікація сутностей, якими оперують користувачі або розробники для опису проблеми і її рішення;
– ідентифікація обов’язків кожної абстракції;
– представлення атрибутів і операцій, необхідних для виконання обов'язків кожного сутності-класу.

2. Моделювання розподілу обов’язків у системі:
   – ідентифікація множини класів, які працюють спільно для досягнення деякого поведінки;
   – ідентифікація набору обов’язків для кожного з класів;
   – розгляд множини класів як єдиного цілого: розчленування (занадто велика частка обов'язків), об'єднання (дрібні з тривіальними обов'язками), перерозподіл (для кожної абстракції оптимальний набір).
   – розгляд способів взаємодії і, якщо потрібно, перерозподіл обов’язків.
3. Моделювання непрограмних сутностей:
   – моделювання абстрагованих сутностей у вигляді класів;
   – моделювання апаратної частини ПЗ як нового вузлу з метою можливого розширення її структури.
4. Моделювання примітивних типів. Моделювання сутності, що представляє абстракцію, у вигляді класу або перерахування із зображенням у нотації класу з відповідним стереотипом. Для специфікації діапазону значень, асоційованого з типом, використовуються обмеження.
5. Моделювання простих залежностей. Найбільш поширений вид залежності – це зв’язок класу, який використовує інший клас в якості параметра своєї операції (залежність спрямована на клас, який використовується в якості параметра).
6. Моделювання одиночного наслідування:
   – знаходження в наборі класів обов’язків, атрибутів та операцій, які є загальними для декількох класів;
   – винесення загальних компонентів в інший або новий клас;
   – встановлення узагальнюючих зв’язків.
7. Моделювання структурних зв’язків (асоціативних).

Заключним результатом моделювання системи є діаграма класів та специфікація її елементів. Основними елементами діаграми класів є позначення класів та зв’язків між ними.

Найбільш важливими зв’язками, які використовуються в діаграмах класів [6], є асоціація, залежність, реалізація та узагальнення.

Окремим випадком асоціації є агрегація та композиція, що моделюють відношення “ціле-частина” (різниця полягає у життєвих циклах об’єктів відношення).

Приклад проєктування. Виконаємо проєктування програми для дослідження алгоритмічної складності алгоритмів сортування. Функціональне призначення програми полягає у визначенні залежності обчислювальної складності алгоритму сортування від обсягу даних та визначення асимптотичної складності алгоритму.

Функціональні вимоги:

– завдання мінімальної та максимальної кількості елементів у масиві;
– завдання кроку збільшення кількості елементів масиву;
– заповнення масиву генератором випадкових чисел на заданому діапазоні;
– завдання кількості повторень експерименту при незмінній кількості елементів масиву;
– обчислення часової складності алгоритму шляхом безпосереднього підрахунку кількості операцій для сортування масиву з n елементів;
– обрання алгоритму сортування: злиттям, за розрядами, вставкою, вибором, гнома, гребінцем, коктейльне;
– проведення експерименту з визначення складності алгоритму за наведеної схемою;
– збереження результатів експерименту у файл;
– побудова графіку залежності складності алгоритму від розміру масиву за даними експерименту;
– для графіку залежності складності алгоритму побудова лінії тренду та виведення її рівняння для визначення асимптотичної складності алгоритму.

Схема експерименту:

1. Створити масив з n елементів.
2. Заповнити масив випадковими числами.
3. Відсортувати масив та обчислити кількість виконаних операцій.
4. Повторити countOneSize разів пункти 2-3 та обчислити середню оцінку (середня кількість операцій при сортуванні), m не менше countOneSize .
5. Зберегти значення, отримане в п. 4.
6. Збільшити n на заданий крок та повторимо пункти 1 – 5.
7. Повторити весь експеримент (п. 1 – 6) доки n менше або дорівнює заданій кількості елементів.

Вхідні дані:

– minSize, maxSize - мінімальна та максимальна кількість елементів масиву - цілі додатні числа, тип даних int;
– step - крок збільшення кількості елементів масиву - ціле додатне число в межах від 1 до (maxSize-minSize);
– countOneSize - кількість повторень експерименту без зміни розміру масиву - ціле число, не менше 64;

– обраний метод сортування - ціле число, що позначає номер алгоритму: 1 – злиттям, 2 – за розрядами, 3 – вставкою, 4 – вибором, 5 – гнома, 6 – гребінцем, 7 – коктейльне.

Вихідні дані:

– файл txt-формату, який має назву, що відповідає номеру обраного алгоритму, та містить рядки. Кожен рядок містить середню кількість операцій, необхідну для сортування масиву розміром minSize <= n <= maxSize;

– графік функції зростання алгоритмічної складності, лінія тренду та її рівняння на екранній формі. Ось абсцис - кількість елементів масиву, ординат - кількість операцій.

Специфікуємо функціональні вимоги у вигляді прецедентів (рис. 3.1). Наведена діаграма прецедентів візуалізує функціональні вимоги до системи аналізу складності алгоритмів сортування, визначаючи межі взаємодії між користувачем та програмним продуктом. Актором виступає Користувач, який ініціює ключові сценарії роботи: налаштування параметрів, проведення експерименту, візуалізацію та збереження даних. Структура діаграми побудована таким чином, щоб продемонструвати всі етапи аналізу склданості алгоритму: від підготовки вхідних даних до отримання асимптотичних характеристик алгоритму.

Особливу увагу в моделі приділено використанню зв'язків «include» (включення), які дозволяють деталізувати складні операції. Наприклад, базовий прецедент «Задати параметри експерименту» обов’язково включає специфікацію діапазону масивів (min/max), кроку ітерації та кількості повторів. Це вказує на те, що виконання основного завдання неможливе без визначення цих конкретних параметрів. Аналогічно, прецедент «Провести експеримент» декомпозується на процеси генерації випадкових даних та безпосереднього обчислення складності, що відображає внутрішню механіку роботи обчислювального модуля.

Зв'язок «extend» (розширення) на діаграмі застосований для опису додаткового функціоналу аналізу. Прецедент «Побудувати лінію тренду та вивести її рівняння» розширює базову функцію побудови графіка. Така архітектурна побудова означає, що система дозволяє не просто візуалізувати дані, а й проводити їх математичну апроксимацію за запитом користувача для визначення асимптотичної складності. Це підкреслює гнучкість інтерфейсу: користувач може зупинитися на візуальному аналізі або продовжити його до отримання аналітичної моделі.

У пояснювальній записці до кваліфікаціної роботи опис діаграми має демонструвати розуміння здобувачем основних сценаріїв роботи з прогармою та ієрархічні зв’язки між додатковими. Чітке розмежування між обов’язковими та опціональними сценаріями через механізми UML є ознакою якісного проєктування вимог та готовності архітектури до подальшої програмної реалізації.

Рис. 3.1. Діаграма прецедентів

Виконаємо структурне проєктування системи відповідно етапів, описаних раніше.

Етап 1 – моделювання словника системи.

Ідентифіковані сутності: експеримент, масив, генератор випадкових чисел, алгоритм, результати експерименту, графік, рівняння, робота з файлом. Додатково зазначимо інтерфейс користувача у вигляді екранної форми.

Ідентифіковані обов’язки:

– експеримент - багаторазове виконання обчислення складності алгоритму та формування ряду середніх значень складності алгоритму для заданого розміру масиву;
– масив - збереження кількості та елементів ряду, який буде сортуватися;
– генератор випадкових чисел - генерація числової величини в заданому діапазоні;
– алгоритм - реалізація всіх алгоритмів сортування, передбачених зовнішніми специфікаціями, та обчислення кількості операцій, які виконуються при сортуванні масиву;
– результати експерименту - збереження числових рядів, що вказують на середню кількість операцій при роботі кожного з алгоритмів сортування, ініціація збереження результатів у файл;
– робота з файлом - збереження результатів експерименту у файл;
– графік - побудова за результатами експерименту графіку залежності складності алгоритму від кількості елементів масиву, побудова для нього лінії тренду та пошук її рівняння;
– рівняння - визначення функції, що описує залежність кількості операцій від розміру масиву на основі результатів експерименту;
– форма - введення параметрів експерименту, запуск та перегляд результатів, подання команди на збереження результатів.

Атрибути та операції, необхідні для виконання обов'язків кожної сутності-класу наведемо у табл. 3.4.

Таблиця 3.4

Сутності, атрибути та методи

Сутність	Атрибути	Методи
1	2	3
експеримент	minSize, maxSize - мінімальна та максимальна кількість елементів масиву - цілі додатні числа, тип даних int; step - крок збільшення кількості елементів масиву - ціле додатне число в межах від 1 до (maxSize-minSize);	запустити

1	2	3
	countOneSize - кількість повторень експерименту без зміни розміру масиву - ціле число, не менше 64; sortMethod - обраний метод сортування: ціле число, що позначає номер алгоритму.	
масив	кількість елементів елементи	створити, заповнити
генератор випадкових чисел		генерувати число
алгоритм	масив лічильник кількості операцій при сортуванні	сортувати злиттям, сортувати за розрядами, сортувати вставкою, сортувати вибором, сортувати гномом, сортувати гребінцем, сортувати коктейлем
результати експери–менту	числовий ряд (кількість операцій залежно від кількості елементів масиву), назва алгоритму, кортеж параметрів експерименту	записати дані ряду, отримати назву алгоритму, отримати параметр експерименту за номером, записати результати в файл
робота з файлом		зберегти результати в файл
графік		скласти рівняння, побудувати графік лінії тренду, побудувати графік за точками
рівняння		скласти рівняння
форма	поля для введення вхідних даних поле виведення рівняння лінії тренду поле побудови графіку кнопки для запуску експерименту, побудови графіку, збереження результатів	запустити експеримент, побудувати графіки, зберегти в файл

Етап 2 – моделювання розподілу обов’язків у системі.

Множини класів, які працюють спільно для досягнення деякого поведінки:

– форма, експеримент – введення параметрів для експерименту та його запуск, перегляд результатів;
– форма, графік – відображення кривих залежності складності алгоритмів від кількості елементів масиву;
– експеримент, алгоритм - обчислення складності алгоритму;
– алгоритм, масив - сортування масиву та обчислення складності для окремого алгоритму та набору даних;
– експеримент, результати експерименту – збереження даних експерименту;
– результати експерименту, графік – побудова графіків за даними експерименту, визначення рівняння лінії тренду;
– масив, генератор випадкових чисел – заповнення масиву.

Набір обов'язків класів ідентифікується, виходячи з атрибутів та методів. Додатково зазначимо, що клас «експеримент» реалізує схему багаторазового виконання обчислень складності, описану у функціональних вимогах.

«Генератор випадкових чисел» в подальшому може бути представлено стандартизованим класом деякої мови програмування. Наприклад, клас Random з пакету java.util.Random.

Для сутності «графік» виконаємо розподіл обов'язків на два класи: «рівняння» та «побудова графіку», оскільки пошук рівняння є частиною рівня логіки програми, а побудова графіка є частиною інтерфейсу користувача, що покликана на візуалізацію результатів експерименту.

Етап 3. Моделювання непрограмних сутностей для даної системи виконувати не потрібно, оскільки особливості апаратної частини не використовується в роботі системи, всі інші сутності вже представлено у вигляді класів.

Етапи 4-7. Моделювання примітивних типів, простих залежностей, наслідування та структурних зв'язків.

Виокремлення примітивних типів у вишляді перелічень та окремих класів у цьому проекті не передбачено.Визначення типів атрибутів сутностей виконаємо на етапі побудови діаграми класів.

Наслідування в даному проєкті відсутнє, оскільки всі сутності унікальні. У них відсутні спільні атрибути та/або методо, які можна було б винести у базовий клас.

Результат моделювання різних видів зв'язків подано у табл. 3.5.

Слід зазначити, що усі зв'язки є односторонніми. Наприклад, якщо деякий клас А має зв'язок з класом В, то це не означає, що клас В пов'язано з А.

Таблиця 3.5

Структурні зв’язки

Клас, який зв'язується	Клас, з яким зв’язуються	Тип зв’язку
Форма	Експеримент	Асоціація
	Графік	Асоціація
Експеримент	Масив	Залежність
	Алгоритм	Асоціація
	Результати експерименту	Асоціація
Алгоритм	Масив	Залежність
Масив	Генератор випадкових чисел	Залежність
Графік	Результати експерименту	Залежність
	Рівняння	Залежність
Результати екперименту	Робота з файлом	Залежність

Представимо заключний результат моделювання у вигляді діаграми класів (рис. 3.2).Діаграма відображає архітектуру системи, побудовану з чітким розділенням відповідальності між компонентами. Центральний клас Experiment виступає в ролі контролера, який акумулює параметри дослідження та координує взаємодію між генератором даних (Array, Random), набором методів сортування (Algorithm) та модулем збереження результатів (Results). Виокремлення класів Form для інтерфейсу та Graphic для візуалізації свідчить про дотримання принципу розділення інтересів, де логіка представлення не змішується з обчислювальною логікою.

З точки зору принципів SOLID, структура демонструє високий рівень відповідності принципу єдиної відповідальності (SRP): кожен клас має вузькоспеціалізовану роль. Зокрема, клас Algorithm інкапсулює лише логіку сортування та лічильник операцій, а клас Results відповідає виключно за структурування та персистентність отриманих даних. Використання окремого класу Random для заповнення масиву реалізує гнучкість системи, дозволяючи за потреби змінювати стратегію генерації даних без модифікації логіки самого масиву чи алгоритмів сортування.

На представленій діаграмі відсутня більша частина конструкторів, сетерів та гетерів. Сетери, гетери та конструктори є кодом, який не несе інтелектуального навантаження щодо алгоритмічної суті розробки. Видалення цих елементів дозволяє сфокусувати увагу на унікальних методах (наприклад, run(), makeEquation(), rankSort()), які визначають функціональність застосунку.

Діаграма, перевантажена десятками службових методів, стає візуально важкою для сприйняття. Тому допускається приховування другорядних членів класу для підтримки високого рівня абстракції. Крім

того, наявність приватних полів на діаграмі вже передбачає існування механізмів доступу до них.

Відповідно до вищесказаного, діаграма відображає концептуальну модель системи та ключові інтерфейси взаємодії. Службові методи були опущені для підвищення читабельності схеми та акцентування уваги на прикладній логіці дослідження складності алгоритмів, яке виконує застосунок.

Уточнення складу методів класу може бути виконано при проєктуванні динаміки системи.

Класи можуть бути специфіковані у вигляді CRC-карток (Class - Responsibility- Collaboration) (табл. 3.6).

Таблиця 3.6

Загальний вид CRC-картки

Базовий клас	Похідні класи (нащадки)
Перелік всіх класів та інтерфейсів, від яких унаслідується даний	Перелік всіх класів, які є похідними від даного
Обов’язки	Зв’язки
Опис призначення класу	Перелік класів, з якими взаємодіють об’єкти даного

У табл. 3.7 наведемо приклад картки для одного з класів рис. 3.2.

Таблиця 3.7

CRC-картка для класу “Експеримент”

Базовий клас	Похідні класи (нащадки)
відсутній	відсутні
Обов’язки	Зв’язки
Багаторазовий запуск сортування з метою отримання числового ряду середньої кількості операцій, необхідної для сортування масиву розміром в заданому діапазоні	Масив, алгоритм, результати експерименту

При проєктуванні класів слід дотримуватися SOLID-принципів [8, 9 с. 62-86] та за потреби використовувати патерни проєктування [10]. При використанні патернів для кожного слід вказати його назву та яку проблему в розробці здобувача він вирішує. Доречним буде акцентувати увагу на перевагах редакції проєкту ПЗ з патернами порівняно з їх відсутністю.

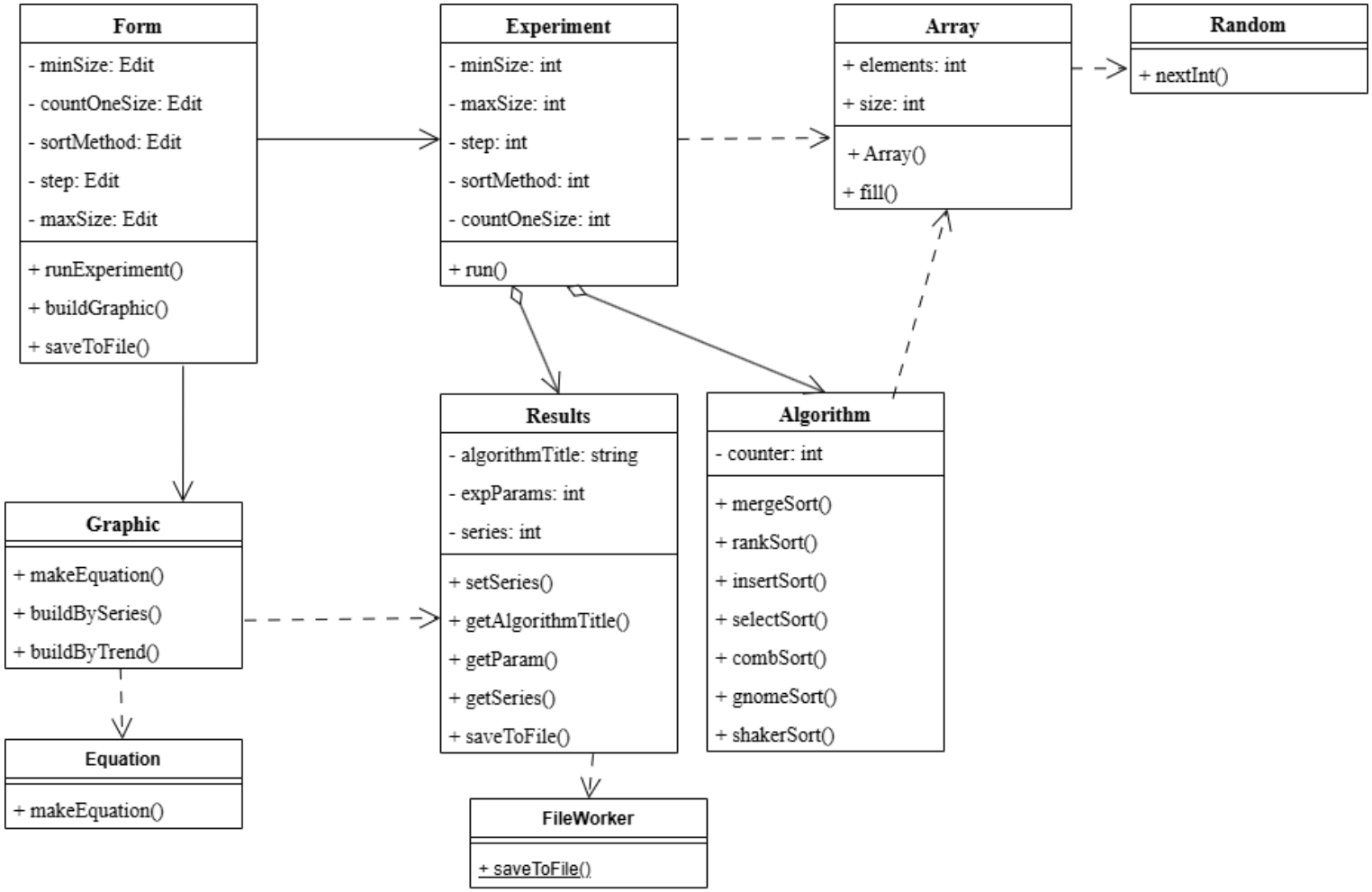


Рис. 3.2. Діаграма класів

3.3.2 Проєктування інтерфейсу користувача

Інтерфейс користувача (ІК) – це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії складають діалоги. Під діалогом розуміють регламентований (обмежений деякими правилами) обмін інформацією між людиною і комп'ютером, який здійснюється в реальному масштабі часу і спрямований на спільне вирішення конкретного завдання: обмін інформацією та координація дій. Кожен діалог складається з окремих процесів введення-виведення, які фізично за допомогою монітору, клавіатури, миші і інших маніпуляторів забезпечують зв'язок користувача і комп'ютера.

При проєктуванні ІК особливу увагу слід приділити сценарію, структурі діалогу та візуальних атрибутів. До останніх належать:

– взаємне розташування і розмір відображуваних об'єктів;
– кольорова палітра;
– засоби залучення уваги користувача (анімація, звук, виділення об’єктів).

Розробка сценарію передбачає опис станів та переходів, в яких перебуває система та користувач в рамках вирішення конкретного завдання. Розробка та тестування сценарію діалогу дозволяють уникати складності у виборі операцій, станів програми без виходу та відсутності необхідних даних при вже виконаному переході до деякої операції.

Сценарій діалогу може бути представлено у словесному вигляді, проте така форма може містити неоднозначності. На противагу їй доцільним є використання певних формалізмів: графа діалогу, автоматів, мереж Петрі, UML-діаграм прецедентів, діяльності та станів.

Наведемо приклад сценарію діалогу для системи визначення складності алгоритмів сортування (рис. 3.3), описаної в п. 3.3.1. В якості засобу формалізації використаємо діаграму станів. На діаграмі стани, позначені кругом та кругом в колі, є початковим та заключним станами відповідно. Вони позначають початок та завершення роботи з програмою. Вихід з програми можливий з будь-якого стану, проте, зважаючи на функціональне призначення програми, перехід у заключний стан на діаграмі представлено в одному екземплярі. Прямокутниками з заокругленими кутами позначені стани користувача.

Подана діаграма станів деталізує циклічний характер взаємодії користувача з інтелектуальною системою. Процес починається з етапу конфігурації, де стани «Введення значень параметрів експерименту» та «Вибір алгоритму» пов'язані двостороннім переходом. Це дозволяє гнучко змінювати умови тестування (наприклад, розмір масиву або тип розподілу даних) залежно від обраного методу сортування ще до моменту виконання обчислень. Перехід до стану «Запуск експерименту» є точкою

фіксації вхідних даних, після якої система переходить у фазу очікування результатів.

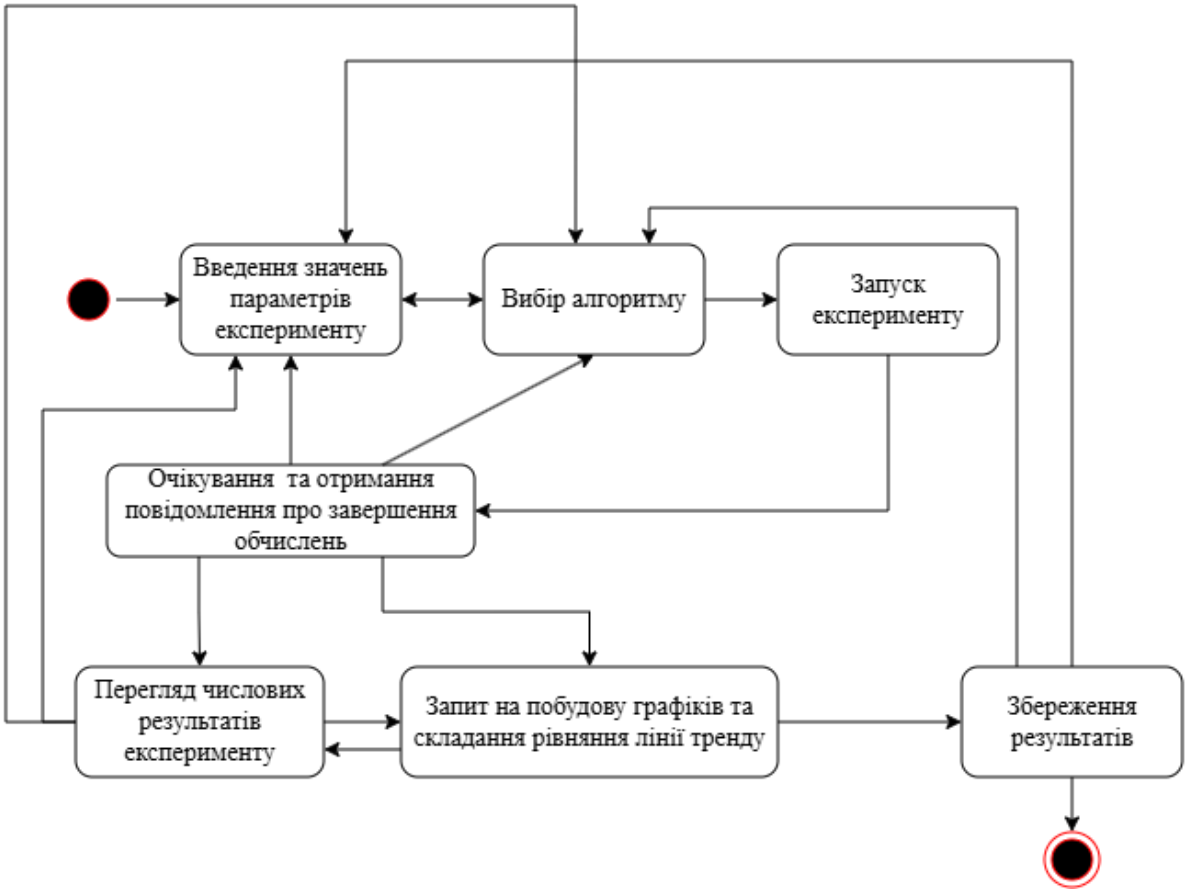


Рис. 3.3. Діаграма станів користувача програми

Аналітичний блок сценарію активується після отримання повідомлення про завершення обчислень. Діаграма демонструє розгалужену логіку обробки результатів: користувач має можливість перемикатися між табличним представленням («Перегляд числових результатів») та графічною візуалізацією з апроксимацією даних лінією тренду. Важливою інженерною особливістю є наявність зворотних зв’язків зі станів аналізу до етапу введення параметрів та вибору алгоритму. Це реалізує ітеративний підхід до дослідження складності, дозволяючи проводити серію порівняльних експериментів без перезапуску всього застосунку.

Заключний етап сценарію передбачає перехід до стану «Збереження результатів», що забезпечує персистентність отриманих даних (експорт у файл *.txt для подальшого використання у звітності чи наступного експорту до Google Sheets або аналогів для візуалізації результатів різними видами графіків). Логічне завершення роботи з програмою відбувається через перехід у фінальний стан, що гарантує коректне вивільнення ресурсів та завершення сеансу користувача. Такий опис

сценарію у формі діаграми станів дозволяє чітко спроектувати інтерфейс користувача, мінімізуючи ризик виникнення «глухих кутів» у навігації застосунку.

Структура діалогу визначає спосіб надання команд від користувача для програми. Основними структурами діалогу є питання-відповідь, командна мова, структура на основі меню, на основі форм.

У пояснювальній записці слід вказати обрану структуру та обґрунтування вибору.

Для системи обчислення часової складності алгоритмів, описаної вище, зручною є структура діалогу на основі екранної форми, оскільки користувач має ввести стандартизований набір даних, а наявність графік потребує використання графічного ІК (GUI).

Після визначення сценарію та структури діалогу слід перейти до проєктування форм, меню, іконок або інших об’єктів, які входять до складу інтерфейсу. На даному етапі слід виконати такі дії:

1. Визначити склад інформації, яка повинна з'являтися на екрані.
2. Вибрати формату представлення інформації.
3. Визначити взаємне розташування даних (або об'єктів) на екрані - створити ескізи з дотриманням принципів розміщення інформації (наведено нижче). Ескізи виконуються у масштабі, з низькою деталізацією та фокусом уваги на функціональності. Кольорова гама та призначення об’єктів вказується додатково, як пояснення до рисунку.
4. Вибрати засоби привернення уваги користувача.
5. Розробити макет розміщення даних на екрані з високою деталізацією елементів інтерфейсу з акцентом на естетиці та бренді.
6. Оцінити ефективність розміщення інформації, визначивши кількість операцій, необхідних для виконання поставлених задач (кількість натискань кнопок миші та/або клавіатури) та час, необхідний на опанування основних принципів роботи з програмою, яка розробляється.

Загальні принципи розташування інформації на екрані повинні забезпечити для користувача:

– можливість перегляду екрану в логічній послідовності;
– простоту вибору потрібної інформації;
– можливість ідентифікації пов'язаних груп інформації;
– розрізнення виняткових ситуацій (повідомлень про помилки або попереджень);
– можливість визначити, яку дію з боку користувача потрібно (і чи потрібне взагалі) для продовження виконання завдання.

Обов’язковим є визначення переліку повідомлень для користувача. Він може бути поданий в форматі «текст – адресат – ситуація – рекомендовані дії».

При проєктуванні ІК додатково можна надати повний перелік пристроїв, що будуть використовуватись при роботі програми, можливі варіанти для дублювання виконання необхідних операцій.

Для складних багато функціональних систем слід визначити засоби адаптації системи і засоби підтримки користувача, а саме: спливаючі підказки, рядок стану, майстер (або декілька), довідник (багаторівнева допомога), засоби навчання в разі необхідності.

3.3.3 Проєктування динаміки системи

Динаміка об‘єктно-орієнтованих систем проєктується на основі аналізу застосування системи (Use Case Diagram) та первинної класифікації. Аналізуючи кожен варіант застосування, треба визначити, об'єкти яких класів повинні створюватися, бути активними та взаємодіяти між собою. При цьому визначаються конкретні методи класів, які необхідні для реалізації того чи іншого варіанту використання програми.

Результати проєктування динаміки системи можуть бути представлені рядом UML-діаграм [6, 7]:

– діяльності (Activity Diagram);
– кооперації/комунікації (Collaboration/ Communication Diagram);
– послідовності (Sequence Diagram);
– станів (Statechart Diagram).

Кожна з перелічених діаграм фокусується на різних аспектах системи: діаграма діяльності описує логіку бізнес-процесів або алгоритмів, діаграма станів – життєвий цикл окремого об’єкта, а діаграми послідовності та кооперації зосереджені на обміні повідомленнями між об’єктами.

Діаграма діяльності є найкращим інструментом для моделювання складних алгоритмів або бізнес-логіки прецедентів (Use Cases). Вона візуалізує паралельні потоки та розгалуження, що робить її незамінною в розділі проєктування для опису того, як дані проходять через систему від входу до виходу. У проекті ПЗ її варто використовувати для деталізації складних функцій, які неможливо однозначно зрозуміти з текстового опису.

Діаграма послідовності фокусується на часовій упорядкованості взаємодій. Її слід використосувати для опису сценаріїв роботи, де важливо показати, як об’єкти (наприклад, Контролер, Сервіс, База даних) викликають методи один одного в певному порядку.

Діаграма кооперації за змістом ідентична діаграмі послідовності, але акцентує увагу не на часі, а на структурних зв'язках між об'єктами. Вона показує «архітектурну карту» взаємодії. Якщо у проєкті важливо підкреслити статичне розташування об’єктів та їхні канали зв'язку, слід використати саме таку діаграму.

Діаграма станів описує поведінку одного складного об’єкта, чий стан змінюється під впливом зовнішніх подій (наприклад, замовлення, яке переходить зі стану «Нове» в «Оплачене»). Це незамінний інструмент для систем, що працюють у реальному часі, або для опису логіки користувацького інтерфейсу.

Побудова даних діаграм для проєкту є опціанальною, оскільки несе додаткову інформацію. Однак, вони дозволяють доповнити діаграму класів новими компонентами:

- класами, якщо при моделюванні поведінки в ході логічних міркувань з’являється об’єкт, проте в відсутній клас для його опису;
- методами, коли для передачі повідомлення відсутня функція в класі, яка може виконати дану дію;
- зв’язками, за умови, якщо при моделюванні взаємодії логічною є передача повідомлень, проте зв'язки між відповідними класами відсутні (немає стрілок відношень).

Розглянемо ***приклад побудови діаграми діяльності***. На діаграмі представимо дії для реалізації прецеденту побудови графіка (рис. 3.4). Ініціатором прецеденту – сценарію дій – є користувач як зовнішній об’єкт системи.

Діаграма діяльності чітко ілюструє розподіл відповідальності між компонентами системи завдяки використанню «доріжок» (swimlanes). Кожен логічний блок системи має свою зону відповідальності, що дозволяє наочно продемонструвати потік керування та даних. Зокрема, на діаграмі видно, як запит користувача проходить через об’єкт Form до компонента Graphic, який стає центральним вузлом прийняття рішень.

Оскільки графік може бути побудований за точками або за знайденою залежністю, то діаграма містить позначку ромба, що вказує на варіативність дій. Рішення про те, будувати графік на основі сирих експериментальних даних («За рядом») чи на основі математичної моделі («Equation»), визначає подальший шлях виконання процесу. Такий підхід дозволяє врахувати варіативність сценаріїв у межах одного прецеденту, що є обов'язковою вимогою до якісно оформленої конструкторської документації бакалаврської роботи.

В реалізації прецеденту беруть два об’єкти, які відпочатку не показані на доріжках. Ряд чисел є масивом даних та частиною об’єкта класа Results. Функція є новим об’єктом, що описує характер (рівняння) знайденої залежності складності алгоритму від кількості елементів масиву.

Завершення сценарію візуалізацією результату та поверненням контролю користувачеві підкреслює цілісність описуваного бізнес-процесу та його готовність до програмної імплементації.

Наведемо ***приклад діаграми послідовності*** (рис. 3.5) для прецеденту «Провести експеримент” (див. п. 3.3.1 рис. 3.1)».

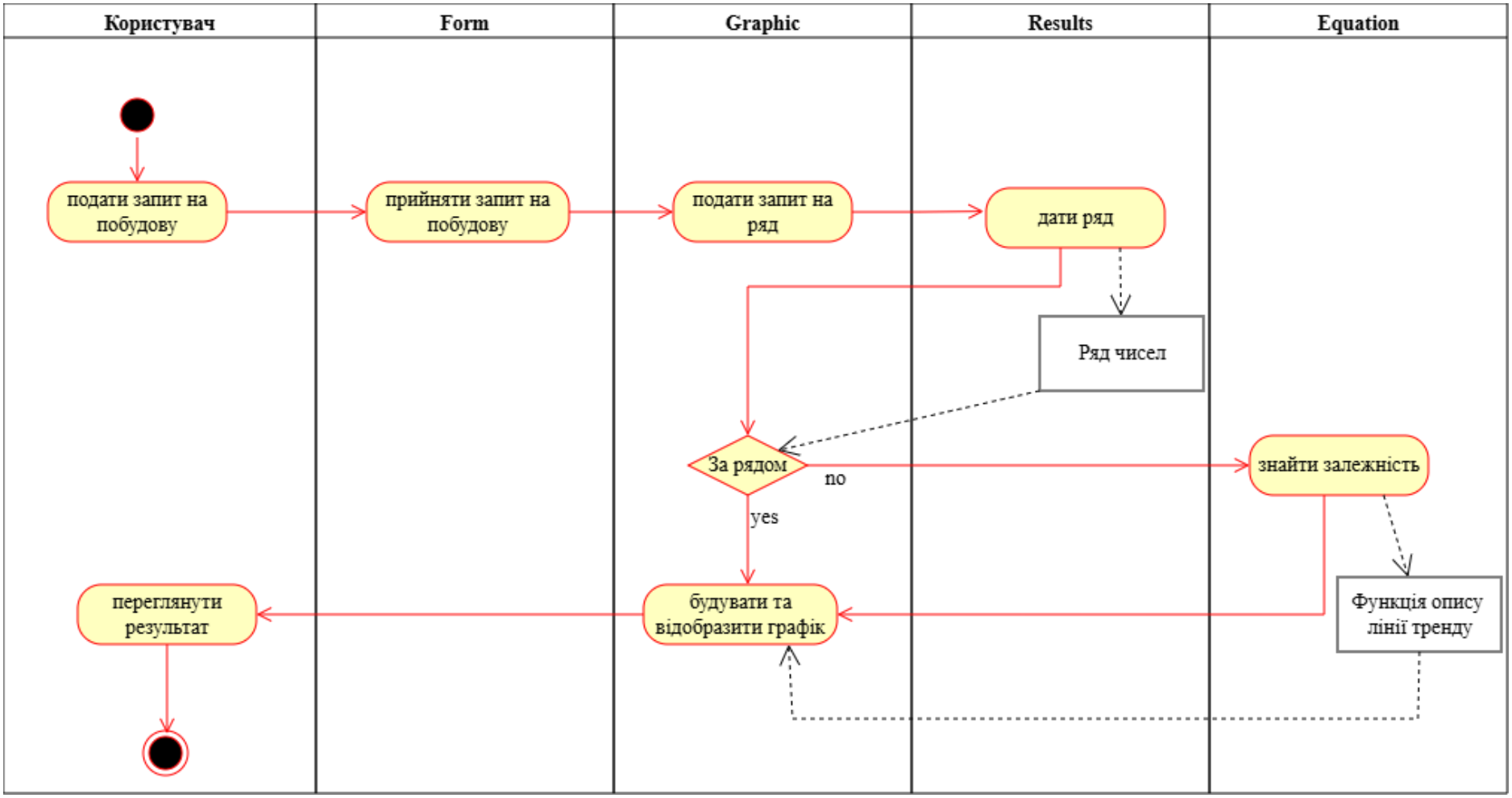


Рис. 3.4 Діаграма діяльності

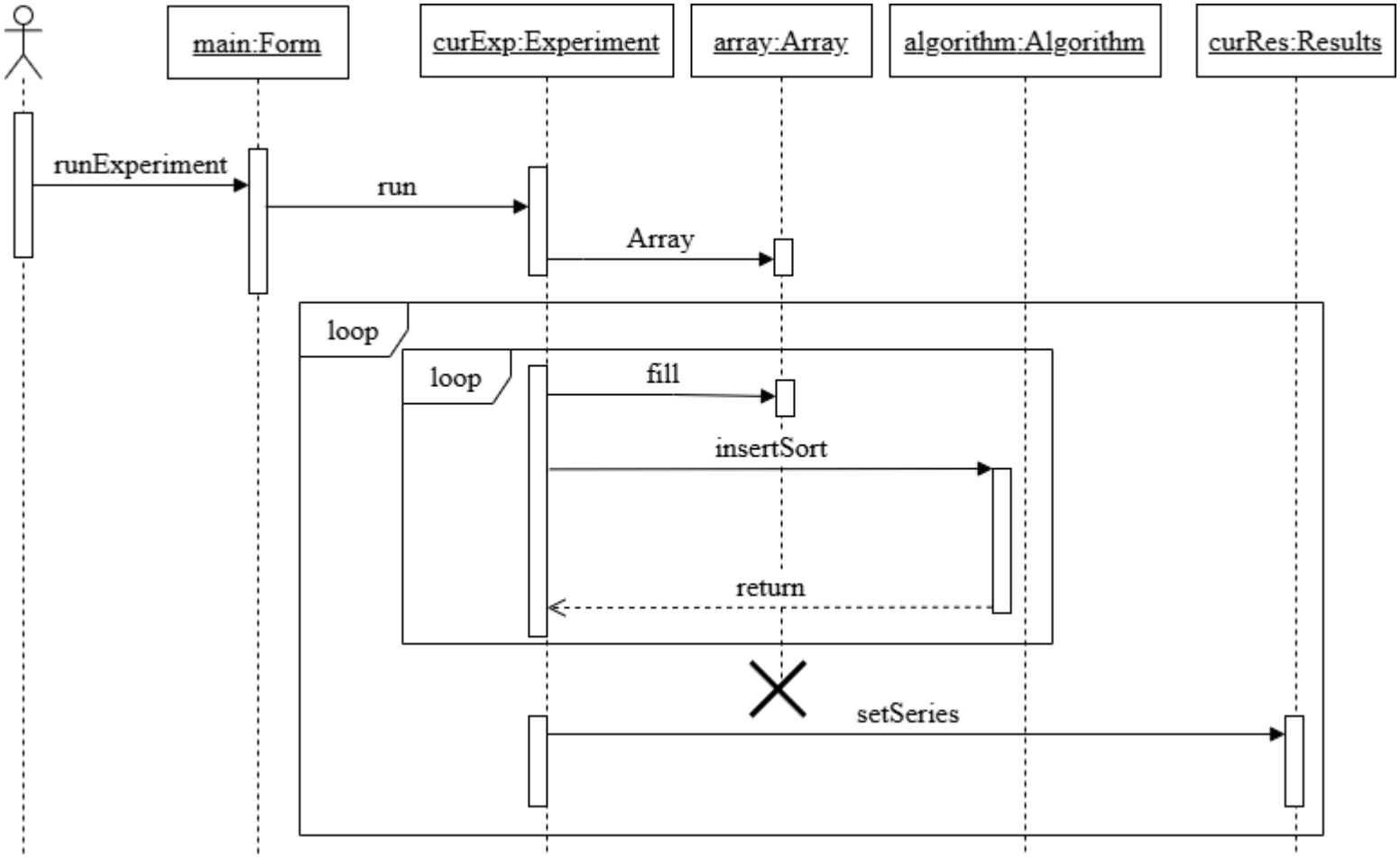


Рис. 3.5. Діаграма послідовностей

Розглянутий прецедент реалізує багаторазове виконання сортування масиву різного розміру для визначення обчислювальної складності алгоритму. У якості алгоритму обрано сортування вставками.

На представленій діаграмі (рис. 3.5) використано позначку циклу «loop», що передбачає багаторазове виконання методів, які в нього включено.

Діаграма станів представляє динамічну поведінку сутностей на основі специфікації їх реакції на сприйняття деяких конкретних подій. Діаграма станів за своєю суттю є графом спеціального виду, який представляє певний автомат.

Діаграма станів може застосовуватися для моделювання внутрішніх та зовнішніх об’єктів системи. Так до зовнішніх можна віднести користувача. Діграма станів на рис. 3.3 ілюструє стани користувача системи для обчислення складності алгоритмів сортування. Як приклад моделювання внутрішнього об’єкта розглянемо стани екземпляру класу Array (рис. 3.6).

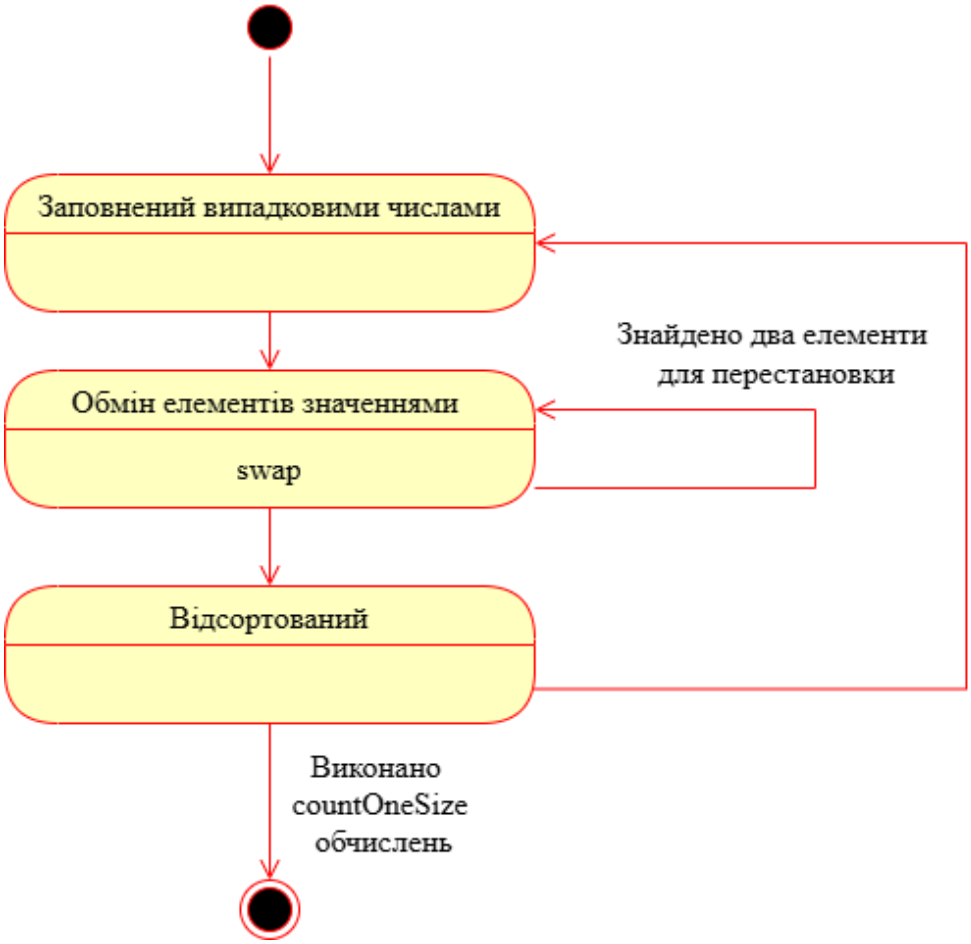


Рис. 3.6. Діаграма станів

Один і той самий масив використовується кілька разів для отримання середнього числа операцій при сортуванні масиву заданого розміру - цим

пояснюється повернення до стана заповненості випадковими числами. Оскільки сортування масиву - процес циклічний, то відповідний стан обміну елементів має на діаграмі дугу-петлю.

3.3.4 Проєктування системи на фізичному рівні

Для моделювання системи на фізичному рівні доцільним є використання UML-діаграм компонентів, або артефактів (Component Diagram) та розгортання (Deployment Diagram). Розглянемо особливості їх побудови.

Артефакт – це фізична частина системи, яка існує на рівні платформи реалізації. Прикладами артефактів є вихідні (source) файли, скрипти, виконувані файли, таблиці баз даних, документи текстових редакторів, які створюються в ході роботи ПЗ, та поштові повідомлення тощо [11].

Діаграми артефактів застосовуються для статичного представлення реалізації системи і, зазвичай, містять артефакти і зв'язки між ними (залежності, узагальнення, асоціації та реалізації), а також примітки та обмеження.

Класи і артефакти є класифікаторами, але між ними є істотна різниця:

– класи – логічні абстракції, а артефакти – фізичні сутності і можуть розташовуватися на вузлах, класи – ні;
– артефакти представляють фізичну упаковку біт на платформі реалізації;
– класи можуть мати атрибути і операції. Артефакти можуть реалізовувати класи і методи, але самі не мають атрибутів і операцій.

UML визначає кілька стандартних стереотипів для артефактів, включаючи «source» та «executable», які можуть бути додатково спеціалізовані за потреби [10].

Моделювання артефактів допомагає візуалізувати, уточнювати, конструювати і документувати прийняті рішення щодо фізичної організації системи. Його важливість зростає, якщо потрібно контролювати версії і керувати конфігурацією частин системи.

При розростанні моделі концептуально або семантично близькі артефакти моделюються за допомогою пакетів.

Для великих систем, які розміщуються на кількох комп'ютерах, може знадобитися моделювання способу розміщення артефактів із зазначенням вузлів, на яких вони знаходяться.

Розглянемо приклад діаграми артефактів для системи визначення обчислювальної складності алгоритмів (рис. 3.7).

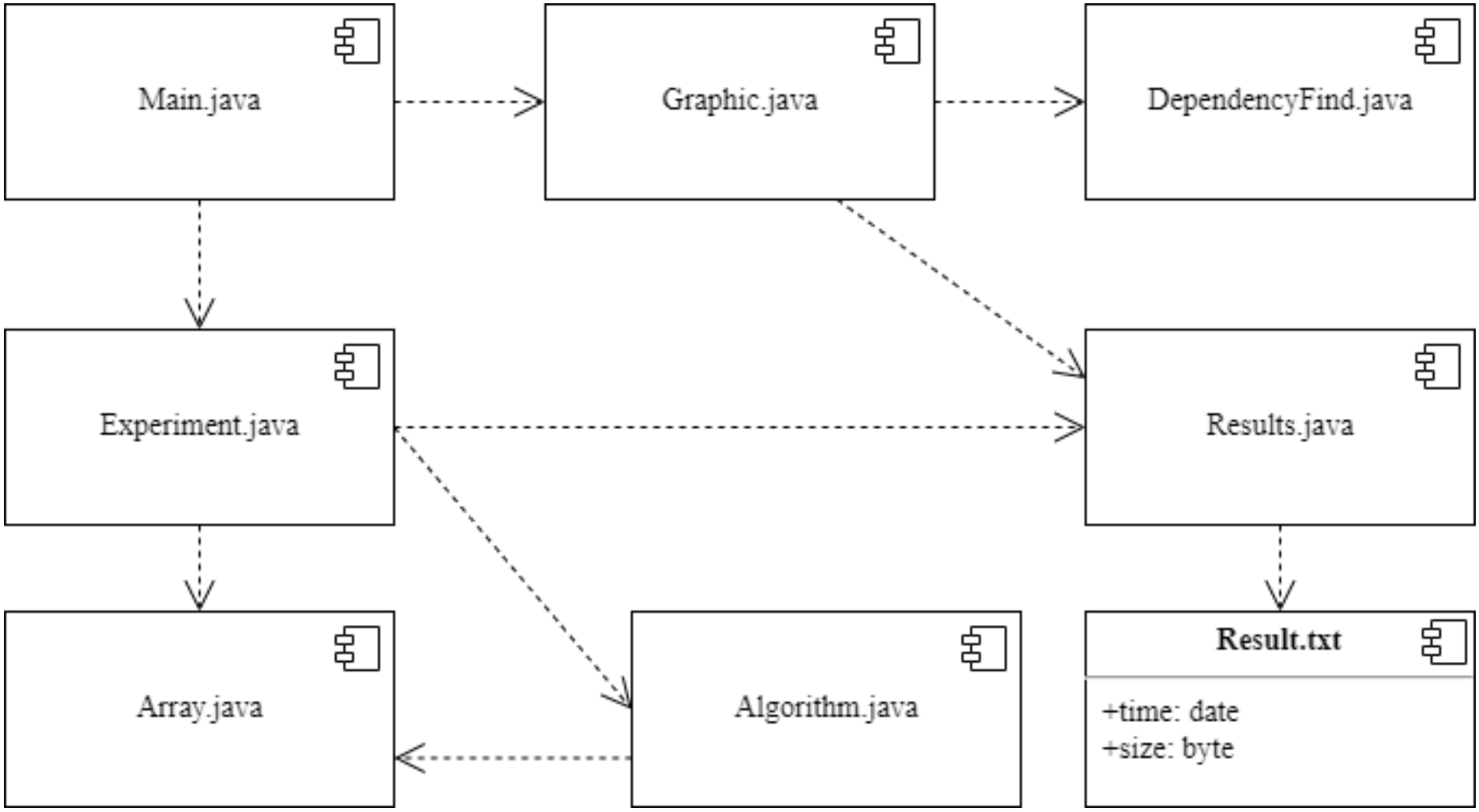


Рис. 3.7. Діаграма артефактів (компонентів)

Передбачаються, що система буде реалізована мовою java. Компонент Main буде реалізовувати створення форми, тому на нього покладено реалізацію елементу Form діаграми класів (рис. 3.2). Всі інші назви компонентів збігаються з відповідними назвами класів. Після компіляції буде створено однойменні компоненти *.class. Компонент Result.txt є артефактом процесу виконання - файл, в який зберігаються результати роботи програми.

Артефактом розміщення для даної системи буде jar–файл, який можна створити засобами IDE на основі class-файлів. java-компоненти, наведені на діаграмі, є артефактами робочих продуктів. Щодо стереотипів, то до Executable віднесемо jar–файл; Library відсутні; File - всі java-файли; Document - Result.txt.

Для визначення топології системи та фізичного розміщення артефактів доцільним є використання діаграми розгортання.

Діаграма розгортання (розміщення) застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів по окремих вузлах системи [7]. Крім того, діаграма розгортання показує наявність фізичних з'єднань – маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи.

Цілі побудови діаграми розгортання такі:

– визначити розподіл компонентів системи по її фізичним вузлам;
– показати фізичні зв'язки між усіма вузлами реалізації системи на етапі її виконання;
– виявити вузькі місця системи і зміни конфігурацію її топологію для досягнення необхідної продуктивності.

Діаграма розгортання (розміщення) є корисною для клієнт-серверних систем, у тому числі з клієнтами, які мають різні ролі, а також при наявності спільного мережевого обладнання, наприклад, принтерів.

Для системи визначення обчислювальної складності алгоритму, яка проєктувалася в попередніх пунктах, діаграма розгортання не буде інформативною, оскільки система не є розподіленою. Зовнішній вигляд діаграми в даному випадку зводиться до одного вузла-процесора, що позначається кубом.

3.3.5 Проєктування бази даних

Проєктування бази даних (БД) полягає у визначенні моделі даних - способів і засобів представлення предметної області. Класичними моделями є: мережева, ієрархічна та реляційна. Остання є найбільш поширеною, вона може бути перетворена на інші. Зручним засобом представлення проєкту БД є модель «сутність-зв’язок» [11], що характерна, в першу чергу, для реляційних БД.

Модель «сутність-зв'язок» візуалізується у вигляді схеми (ERD або ER-діаграми). Вона є різновидом блок-схеми, де показано, як різні «сутності» (люди, об'єкти, концепції тощо) пов'язані між собою всередині системи.

При ER-моделюванні бази даних можуть бути розглянуті на трьох рівнях:

– концептуальному: схема найвищого рівня з мінімальною кількістю подробиць. Перевага цього підходу полягає у можливості відобразити загальну структуру моделі;
– логічному: схема містить більш детальну інформацію, ніж концептуальна модель. На цьому рівні визначаються докладніші операційні та транзакційні сутності. Логічна модель не залежить від технології, в якій вона застосовуватиметься;
– фізичному: має бути достатньо технічних подробиць упорядкування і застосування самої бази даних.

Графічне позначення елементів моделі залежить від нотації. Найпоширенішими є нотація Чена та «вороняча лапка» [12].

На етапі моделювання доцільним є приведення БД до нормальних форм [12, 13]. Процес перетворення відношень бази даних до виду, що відповідає нормальним формам, називається нормалізацією.

Загальне призначення процесу нормалізації полягає у такому:

– виключення деяких типів надмірності;
– усунення деяких аномалій оновлення;
– розробка проєкту бази даних, який є досить «якісним» уявленням реального світу, інтуїтивно зрозумілий і може бути хорошим підґрунтям для подальшого розширення;
– спрощення процедури застосування необхідних обмежень цілісності.

Розглянемо приклад проєктування БД, що є частиною застосунку, який реалізує електронний посібник. Для роботи з ним має бути реалізовано такі ролі для користувачів: студент, викладач, адміністратор. Функціональні вимоги специфікуємо за допомогою діаграми варіантів використання.

Застосунок повинен працювати з трьома рівнями доступу до інформації: студент, викладач, адміністратор.

На рівні “студент” передбачено такі можливості:

– реєстрація у програмному застосунку;
– перегляд лекційного матеріалу;
– перегляд матеріалів для практичних робіт;
– проходження тестування;
– можливість пропустити питання при проходженні тестування.

На рівні “викладач”, крім вище зазначених, передбачено такі можливості:

– перегляд результатів тестування студентів з можливістю їх видалення;
– додавання, видалення та редагування лекційних матеріалів, матеріалів для практичних робіт, тестів;
– можливість роздрукування детальних результатів проходження тестування окремого студенту;
– встановлення обмеження часу проходження тестування;
– встановлення випадкового порядку завдань у тесті.

На рівні “адміністратор” надано усі можливості попередніх рівнів, а також можливість введення даних про користувачів програми та надання їм певних рівнів доступу.

Виконаємо специфікацію вимог за допомогою UML-діаграми прецеднітв (рис. 3.8). Звернемо увагу на відношення між акторами: узагальнення позначає спадкування дій студента викладачем, викладача – адміністратором. Використання такого позначення дозволяє уникнути дублювання інформації про можливі варіанти використання застосунку. Зокрема, актор «Викладач» автоматично отримує доступ до всіх функцій перегляду контенту та тестування, що доступні «Студенту», а «Адміністратор» володіє повним набором прав обох попередніх ролей, додаючи до них функції управління користувачами.

Також на діаграмі використано відношення узагальнення між прецедентами, яке вказує на можливість зведення дій до «базової». Така форма представлення вказана для акцентування уваги на тому, що узагальнені дії хоча й подібні, але мають свої особливості. Наприклад, прецеденти «Додати лекційні матеріали» та «Додати тест» є специфічними формами абстрактної дії «Додати елемент посібника», що дозволяє в майбутньому уніфікувати програмні інтерфейси для роботи з різними типами контенту.

Функціональні можливості системи на діаграмі можна розділити на кілька ключових блоків:

Блок взаємодії з контентом охоплює базові потреби користувачів (реєстрація, перегляд теоретичних та практичних матеріалів).

Блок контролю знань фокусується на процесі тестування. Використання пунктирних ліній із відкритими стрілками, які позначають додаткові сценарії (наприклад, можливість пропустити питання під час тестування або встановлення випадкового порядку питань викладачем) дозволяє відобразити варіативність виконання прецедентів залежно від умов.

Блок керування та моніторингу відображає інструментарій для подальшого аналізу результатів (перегляд, друк та видалення результатів тестування) та адміністративні функції (додавання користувачів і налаштування їхніх прав доступу).

Рис. 3.8. Діаграма прецедентів для застосунку «Електронний посібник»

Блок редагування деталізує процеси оновлення та актуалізації матеріалів посібника, що є критично важливим для підтримки системи у робочому стані.

Така деталізація архітектури прецедентів дозволяє на етапі проєктування чітко розмежувати зони відповідальності в системі та визначити повний обсяг функціональних вимог до майбутнього програмного продукту.

Процес проектування бази даних для застосунку «Електронний посібник» починається з побудови концептуальної ER-діаграми, яка дозволяє абстрагуватися від технічних обмежень конкретних СКБД і зосередитися на сутностях предметної області та їхніх атрибутах. На цьому етапі визначаються ключові об'єкти. Така візуалізація є критично важливою для інженерії ПЗ, оскільки вона фіксує бізнес-логіку зберігання даних та забезпечує фундамент для подальшої розробки.

Побудуємо ER-модель бази даних на концептуальному рівні (рис. 3.9). На діаграмі наведено основні сутності з їх атрибутами та зв’язками між сутностями.

Діаграма відображає структуру бази даних для застосунку «Електронний посібник», побудовану за модульним принципом. В основі архітектури лежить сутність «Модуль», яка структурує навчальний контент за номером та темою. Кожен модуль виступає контейнером для трьох основних типів матеріалів: теоретичних лекцій, практичних завдань та контрольних тестів. Всі ці елементи мають спільні атрибути, такі як тема та посилання на відповідний файл, що забезпечує логічний зв'язок між різними формами навчання в межах одного розділу.

Окремий блок системи присвячений взаємодії користувача з контрольними заходами. Сутність «Тест» містить інформацію про кількість питань і пов’язана із сутністю «Тестування», яка фіксує результати проходження: ПІБ студента, дату складання тесту та отриману оцінку. Це дозволяє системі зберігати історію успішності та динаміку навчання для кожного конкретного випадку.

Кінцевою ланкою схеми є сутність «Користувач», яка відповідає за автентифікацію та розмежування прав доступу. Окрім персональних даних (ПІБ та пароль), вона містить номер групи та рівень доступу, що критично важливо для поділу функціоналу між студентами (перегляд матеріалів, проходження тестів) та адміністраторами чи викладачами (видалення або друк результатів тестування). Схема демонструє чітку ієрархічну залежність від загального контенту до індивідуальних результатів контролю.

Наступним кроком є перехід від концептуальної моделі до логічної схеми БД. У пояснювальній записці важливо продемонструвати ітераційний шлях: від високорівневого опису зв'язків на ER-діаграмі до деталізованої логічної структури. Такий підхід підтверджує системність проектування та гарантує, що розроблена база даних буде ефективно підтримувати функціональні вимоги за стосунку.

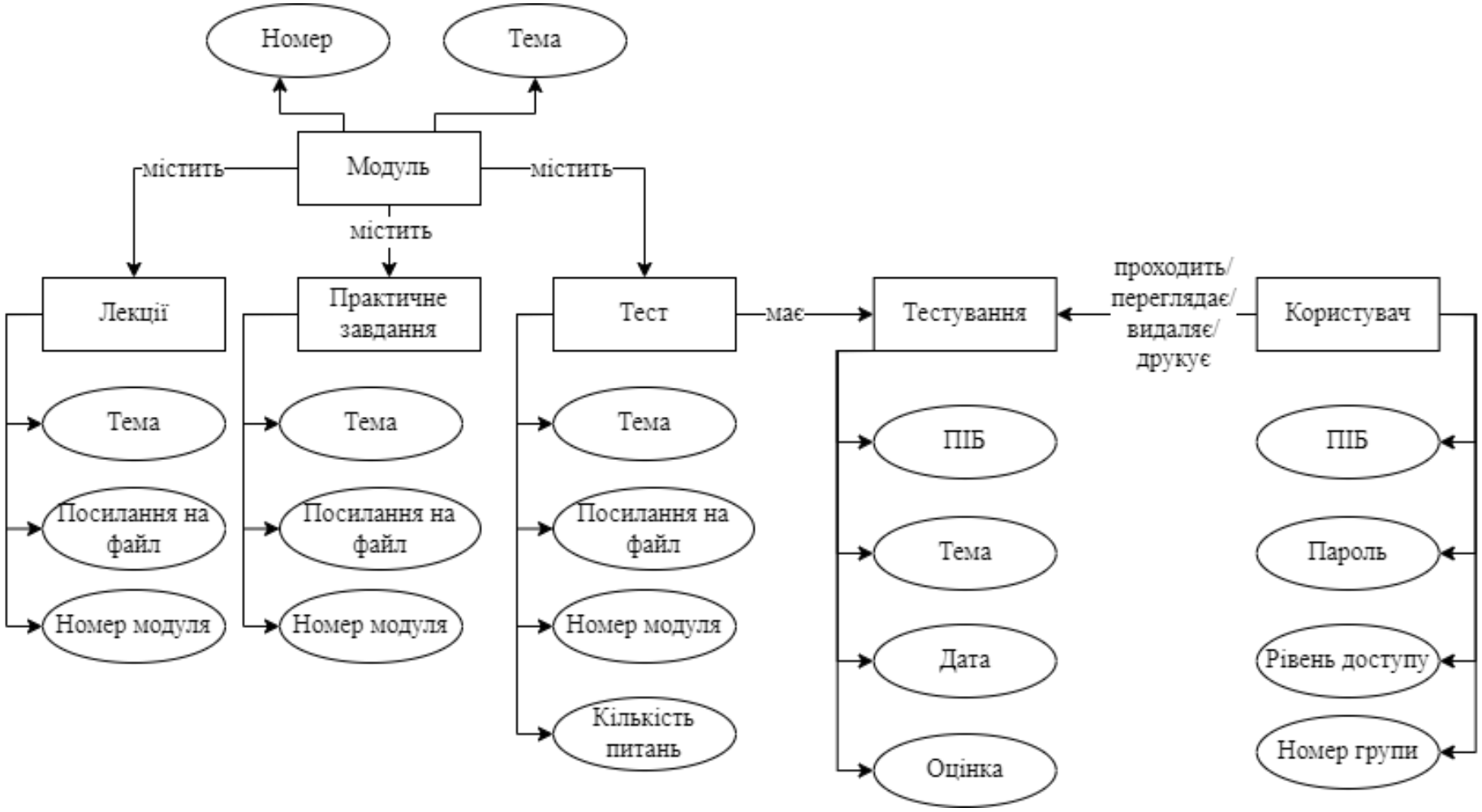


Рис. 3.9. ER-діаграма концептуального рівня

З метою реалізації БД була побудована логічна схема (ER-модель на логічному рівні рис. 3.10) та виконана нормалізація. Логічна схема бази даних, представлена на рисунку, є результатом трансформації концептуальної ER-моделі у реляційну структуру. Схему виконано у нотації «вороняча лапка» (Crow's Foot), яка наочно відображає потужність зв'язків між сутностями. Основними вузлами системи є таблиці Module, User та Testing, що забезпечують збереження навчального контенту, персоналізацію доступу та фіксацію результатів навчання. Використання цієї нотації дозволяє чітко ідентифікувати зв'язки типу «один-до-багатьох» (1:N), де, наприклад, один навчальний модуль може містити декілька лекцій, практичних завдань або тестів, що позначено розгалуженням лінії («лапкою») на стороні підлеглих таблиць.

Архітектура БД демонструє високий рівень нормалізації. Зокрема, сутність Testing виступає як сполучна ланка між користувачем та тестами, реалізуючи зв'язок «багато-до-багатьох» через проміжну таблицю. Деталізація результатів винесена в окрему таблицю detailed_results, що дозволяє зберігати відповіді на кожне конкретне питання без дублювання загальної інформації про сесію тестування. Аналогічно, довідник відповідей Directory_of_answers відокремлений від структури тесту, що полегшує редагування контенту без зміни логіки самого іспиту

В програмі передбачено використання таблиць в MySQL Workbench, на основі концептуальної схеми були розроблені таблиці бази даних (таблиці 3.8 – 3.16). Порівняно з з концептуальною схемою було:

– збільшено кількість таблиць;
– впорядковано атрибути;
– створено первинні та зовнішні ключі;
– вилучено складні зв’язки та проведена нормалізація до третьої нормальної форми.

Схема зв’язків таблиць зображена на рис. 3.10.

На цьому кроці проєктування бази закінчено і можна переходити до реалізації запитів, форм, звітів тощо.

Таблиця 3.8

Таблиця «User»

Назва поля	Опис поля	Тип поля	Ключ
User_code	Унікальний код користувача	Int	+
Full_name	ПІБ, також виконує роль логіну	Varchar(45)	-
Group_number	Номер групи	Varchar(45)	-
Password	Пароль	Varchar(45)	-

.

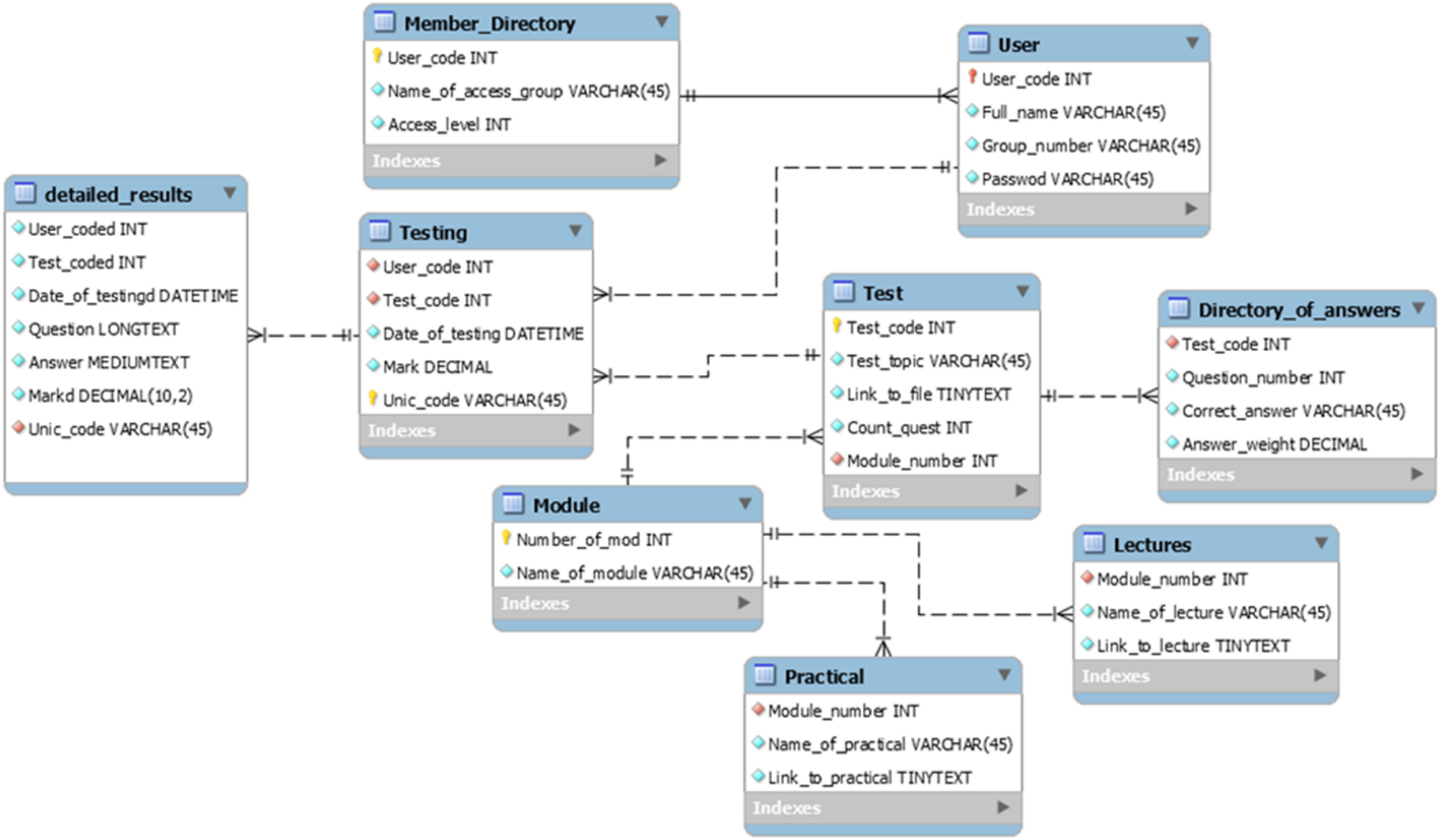


Рис. 3.10. Логічна схема бази даних

Таблиця 3.9

Таблиця «Member_Directory»

Назва поля	Опис поля	Тип поля	Ключ
User_code	Унікальний код користувача	Int	+
Name_of_access_group	Назва групи доступу користувача	Varchar(45)	-
Access_level	Рівень доступу користувача	Int	-

Таблиця 3.10

Таблиця «Module»

Назва поля	Опис поля	Тип поля	Ключ
Number_of_mod	Унікальний номер модуля	Int	+
Name_of_module	Назва модуля	Varchar(45)	-

Таблиця 3.11

Таблиця «Practical»

Назва поля	Опис поля	Тип поля	Ключ
Module_numberP	Номер модулю до якого відноситься дане практична робота	Int	-
Name_of_practical	Назва практичної роботи	Varchar(45)	-
Link_to_practical	Посилання на файл-завдання	Tinytext	-

Таблиця 3.12

Таблиця «Lectures»

Назва поля	Опис поля	Тип поля	Ключ
Module_numberL	Номер модулю до якого відноситься даний лекційний матеріал	Int	-
Name_of_lecture	Назва лекційного матеріалу	Varchar(45)	-
Link_to_lecture	Посилання на файл-лекцію	Tinytext	-

Таблиця 3.13

Таблиця «Test»

Назва поля	Опис поля	Тип поля	Ключ
Test_code	Унікальний код тесту	Int	+
Test_topic	Тема тесту	Varchar(45)	-
Link_to_file	Посилання на файл з питаннями	Tinytext	-
Count_quest	Кількість питань які будуть задіяні у тестуванні	Int	-
Module_number	Номер модулю до якого відноситься даний тест	Int	-

Таблиця 3.14

Таблиця «Directory_of_answers»

Назва поля	Опис поля	Тип поля	Ключ
Test_code	Унікальний код тесту	Int	+
Question_number	Унікальний номер питання у тесті	Int	+
Correct_answer	Правильний варіант відповіді на питання	Varchar(45)	-
Answer_weight	Вага правильної відповіді	Decimal	-

Таблиця 3.15

Таблиця «Testing»

Назва поля	Опис поля	Тип поля	Ключ
User_codet	Унікальний код користувача	Int	-
Test_codet	Унікальний код тесту	Int	-
Date_of_testing	Дата та час проходження тестування	Datetime	-
Mark	Підсумкова оцінка	Decimal	-
Unic_code	Унікальний код пройденого тесту	Varchar(45)	+

Таблиця 3.16

Таблиця «Detailed_results»

Назва поля	Опис поля	Тип поля	Ключ
User_coded	Унікальний код користувача	Int	-
Test_coded	Унікальний код тесту	Int	-
Date_of_testingd	Дата та час проходження тестування	Datetime	-
Question	Унікальний номер питання у тесті	Longtext	-
Answer	Відповідь користувача	Mediumtext	-
Markd	Оцінка за питання	Decimal	-
Unic_code	Унікальний код пройденого тесту	Varchar(45)	-

3.4 Розроблення програми

У цьому розділі пояснювальної записки слід висвітлити основні моменти щодо програмної реалізації, а саме: вибір мови програмування, бібліотек, фреймворків, методів вирішення задач та конструювання алгоритмів. Можливе наведення фрагментів програми, що демонструють особливості реалізації алгоритму, який було сконструйовано.

Доцільним є наведення опису методів та алгоритмів розв'язання задач у словесній (зокрема з використанням методу покрокової деталізації або аналізу повідомлень), графічній (у вигляді блок-схем, діаграм Нассі-Шнайдермана) або комбінованій формі.

У разі застосування методу покрокової деталізації доцільним є приведення структурних схем програми. Можливе наведення структурної схеми поєднання програмних сутностей при висхідному програмуванні.

3.4.1 Критерії вибору засобів розробки

При обґрунтуванні вибору інструментарію, що використовується в процесі розроблення (мови програмування, системи керування базами даних, фреймворки, бібліотеки та інші допоміжні засоби), необхідно чітко вказати аргументи на користь обраних рішень. Вибір не може базуватися лише на особистих уподобаннях або популярності технології. Він має логічно слідувати з виконаного раніше аналізу вимог та визначених архітектурних особливостей системи.

При обґрунтуванні вибору доцільно орієнтуватися на такі критерії:

– відповідність функціональним завданням, наявність готових модулів або специфічних бібліотек для реалізації ключового функціоналу;
– продуктивність, надійність та кросплатформеність обраних засобів;
– наявність вичерпної офіційної документації, активної спільноти для розв'язання технічних проблем, регулярний випуск оновлень безпеки;
– ліцензійна політика та вартість користування обраними засобами.

У пояснювальній записці потрібно надати стислий порівняльний аналіз альтернативних варіантів. Результатом такого аналізу має стати чітка аргументація, чому саме обрана комбінація засобів є найбільш раціональною для розробки конкретного ПЗ, враховуючи баланс між швидкістю розроблення, якістю і надійністю кінцевого продукту.

3.4.2 Розроблення і представлення алгоритмів

Для спрощення задачі, яку слід розв'язати за допомогою програмного застосунку, може бути виконано її декомпозицію. Універсальними

методами декомпозиції є методи покрокової деталізації та аналізу повідомлень.

Суть ***методу покрокової деталізації*** полягає у розробці алгоритму зверху вниз. На першому кроці формулюється задача в цілому, визначаються формати вхідних та вихідних даних, складається якомога повніша специфікацію. На другому виділяються основні під задачі. На наступних кроках ці під задачі розбиваються ще на кілька підзадач. Розбиття виконується доти, доки програміст не матиме повного уявлення як писати програмний код.

Покрокова деталізація може бути представлена в формі:

– словесній – текстовому форматі у вигляді нумерованого багаторівневого списку;

– графічній – деревом (рис. 3.11);

– комбінованій – каскадною таблицею (табл. 3.17).

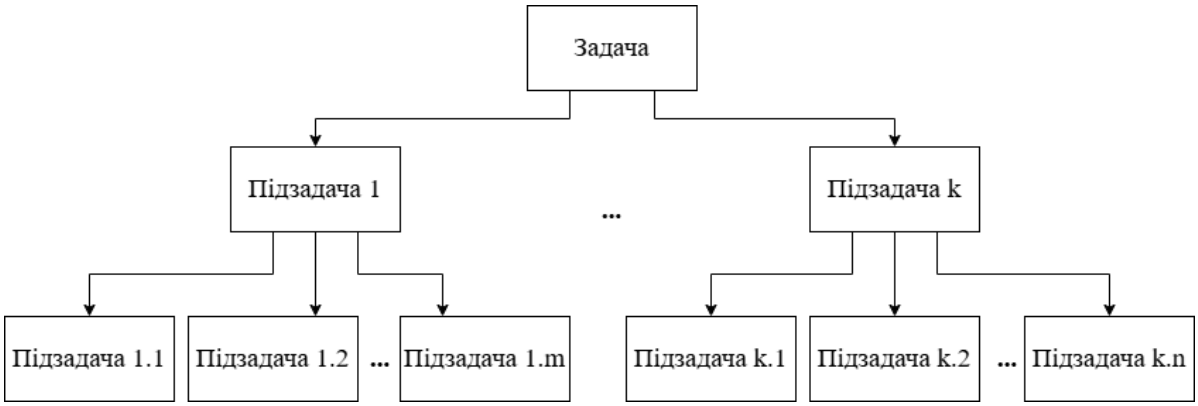


Рис. 3.11. Дерево покрокової деталізації

Таблиця 3.17

Каскадна форма представлення покрокової деталізації

Основна задача	Підзадача 1	Підзадача 1.1
		Підзадача 1.2
		…
		Підзадача 1.n
	Підзадача 2	Підзадача 2.1
		Підзадача 2.2
		…
		Підзадача 2.m
	…	…
	Підзаача k	Підзадача k.1
		Підзадача k.2
		…
		Підзадача k.p

При виборі форми представлення покрокової деталізації слід виходити з принципу наочності та складності самого алгоритму. Словесна форма зручна для швидкого опису лінійних процесів та ті, що мають невелику варіативність (розгалуженість), оскільки легко інтегрується в текст документації (пояснювальної записки), проте вона втрачає ефективність при описі розгалужених структур, де рівні вкладеності стають занадто глибокими для текстового сприйняття.

Графічна форма у вигляді дерева є найбільш репрезентативною для демонстрації архітектурної цілісності системи. Вона дозволяє швидко оцінити масштаб декомпозиції та ієрархічні зв'язки між модулями. Проте може бути більш складною в реалізації, порівняно з іншими.

Каскадна таблиця поєднує в собі переваги обох методів: вона зберігає чітку ієрархію дерева, але при цьому дозволяє додати стислі технічні примітки до кожної підзадачі, що робить її доречною при складанні технічних специфікацій.

У пояснювальній записці до кваліфікаційної роботи рекомендовано використовувати графічне дерево для візуалізації загальної архітектури програми, каскадні таблиці - для деталізації логіки роботи найбільш критичних або складних модулів.

Метод покрокової деталізації безпосередньо корелює з модульним принципом розробки. Кожна виділена підзадача стає основою для окремого модуля або функції, що сприяє високій зв’язності та низькій зачепленості (low coupling) компонентів системи. Це не лише полегшує процес написання коду, але й суттєво спрощує подальше тестування та супровід програмного продукту, оскільки кожна гілка дерева декомпозиції може бути перевірена автономно.

Процес деталізації вважається завершеним, коли кожна кінцева підзадача стає елементарною, тобто такою, що може бути реалізована базовими конструкціями структурного програмування (послідовність, розгалуження, цикл).

Застосування цього методу дозволяє ефективно розподілити завдання під час групової розробки або при плануванні етапів власної роботи. Визначення основних підзадач на ранніх стадіях дає змогу оцінити трудомісткість кожного модуля та встановити критичні точки розробки.

Метод аналізу повідомлень у контексті структурного програмування та потокової декомпозиції (Data Flow Analysis) передбачає просування згори до низу. Він описує підхід, де алгоритм розглядається як конвеєр трансформації даних. Інформація представляється в трьох частинах: витік, перетворення, стік, де витік дає специфікацію вхідних даних, стік – вихідних, перетворення описують дії з обробки даних.

Для опису алгоритму за методом аналізу повідомлень (витік – перетворення – стік) найкраще підходить трирівнева структура. Вона

дозволяє поступово переходити від абстрактного розуміння задачі до конкретної реалізації, зберігаючи логіку потоків даних.

Загальна схема опису алгоритму.

1. Рівень контексту (зовнішній інтерфейс). На цьому етапі алгоритм розглядається як «чорна скринька». Визначаються межі системи, її вихідні та вихідні потоки даних.

Для вхідного потоку (вхідних повідомлень) наводиться опис фізичних даних, які надходять до алгоритму (файли, введення з клавіатури, сигнали датчиків). Визначення вихідного потоку (вихідних повідомлень) передбачає опис очікуваного результату (звіту, оновленого масиву даних, команди на пристрій тощо).

2. Рівень логічної декомпозиції (тріада) – це основний етап аналізу повідомлень, де задача розбивається на три функціональні частини (табл. 3.18).

3. Рівень процедурної деталізації (покроковий опис). На цьому рівні кожен з компонентів тріади розписується за допомогою класичних методів (наприклад, блок-схем чи псевдокоду).

На третьому рівні виконується деталізація витоку у вигляді опису циклу читання, умови переривання при помилках введення. Для перетворення описується основний цикл обробки, математичні формули, логічні розгалуження. При деталізації стоку визначають процедуру формування звіту або передачі даних далі.

Таблиця 3.18

Елементи логічної декомпозиції

Компонент	Призначення	Результат
Витік	Отримання даних, перевірка їх на валідність та перетворення у внутрішній (логічний) формат.	Очищений та структурований потік даних.
Перетворення	Виконання основних обчислювальних операцій, реалізація бізнес-логіки або математичних розрахунків.	Оброблені дані, готові до виведення.
Стік	Форматування результату, підготовка повідомлення для користувача або іншої системи.	Завершений вихідний продукт.

Перевагами цього методу є визначеність даних на кожному етапі та легка організація зв’язування модулів за даними. Метод є корисним для вирішення завдання відділення інтерфейсу користувача від логіки системи.

Наведемо шаблон розроблення алгоритму методом аналізу повідомлень.

1. Загальна характеристика задачі.

Назва алгоритму. Наприклад, сортування масиву методом бульбашки.

Призначення – короткий опис, яку проблему вирішує алгоритм.

Вхідні дані: тип даних, структура, діапазон значень.

Вихідні дані: формат результату.

2. Структурна декомпозиція. На основі аналізу потоків повідомлень алгоритм поділяється на три функціональні блоки:

2.1. Витік – підготовка даних – відповідає за отримання зовнішніх повідомлень та їх переведення у внутрішнє представлення.

Джерело: консоль, файл, сенсор, генератор випадкових чисел тощо.

Механізм обробки: опис перевірки на валідність, приведення типів.

Результат блоку: структура даних, готова до обробки.

2.2. Перетворення – логіка обробки – ядро алгоритму, де відбувається основна трансформація даних. Не залежить від способів введення та виведення.

Основна операція. Наприклад, порівняння та перестановка елементів масиву.

Математична модель/умови: формули, логічні правила.

Результат блоку. Наприклад, оброблений масив, розраховане значення тощо.

2.3. Стік – формування відповіді. Перетворює результат обробки у вихідне повідомлення для споживача.

Отримувач. Наприклад, екран, файл, база даних, інший програмний модуль.

Формат представлення. Наприклад: таблиця, графік, JSON-об’єкт.

3. Покрокова деталізація

```
    Алгоритм [Назва]
ПОЧАТОК
  // 1. БЛОК ВИТОКУ
  Зчитати дані з [Джерело];
  ЯКЩО (дані некоректні) ТО
    Вивести повідомлення про помилку;
    ЗАВЕРШИТИ;
  КІНЕЦЬ ЯКЩО;

  // 2. БЛОК ПЕРЕТВОРЕННЯ
  ДЛЯ кожного елемента В [Дані] ВИКОНУВАТИ
    [Логіка обробки];
  КІНЕЦЬ ЦИКЛУ;

  // 3. БЛОК СТОКУ
  Сформувати вихідне повідомлення;
```

Передати [Результат] до [Отримувач];
КІНЕЦЬ

Для перевірки коректності застосування методу після заповнення шаблону доцільно дати відповіді на такі питання:

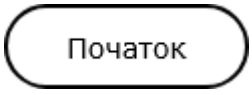
– чи може блок Перетворення працювати, якщо змінити джерело даних у блоці Витік?
– які повідомлення про помилки генеруються на кожному етапі?
– чи є чітка межа, де вхідне повідомлення стає логічною структурою даних?

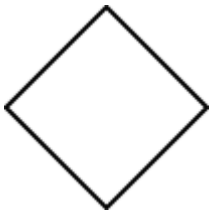
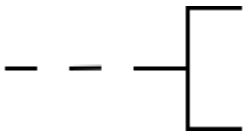
Для представлення алгоритму в цілому та/або його частини, в тому числі виділених за допомогою декомпозиції задачі, може бути використано такі графічні представлення, як блок-схема та діаграма Нассі-Шнайдермана.

Блок-схема – опис структури алгоритму за допомогою геометричних фігур з лініями-зв'язками, що показують порядок виконання окремих інструкцій. У блок-схемі кожному типу дій (введенню початкових даних, обчисленню значень виразів, закінченню обробки і т. п.) відповідає блок, представлений стандартизованою геометричною фігурою – символом (табл. 3.19). Інформація у блоках записується природною мовою або математичними формулами. Блоки з'єднуються лініями переходів, які можуть мати на кінці стрілки. Лінії переходів визначають черговість виконання дій. Якщо вони не мають зламів, то стрілки можна не зображати. Лінію переходу зазвичай підводять до середини блока.

Таблиця 3.19

Основні види блоків та їх призначення

Символ	Смислове навантаження
1	2
	Початок алгоритму
	Кінець алгоритму
	Блок введення/виведення даних
	Обробка даних (обчислення)

1	2
	Рішення, що має один вхід та декілька альтернативних виходів, тільки один з яких може бути активований після обчислення умов, визначених усередині символу. Результати обчислення умови (наприклад, TRUE або FALSE, цілочисленні значення тощо) записують поруч із лініями відповідних шляхів
	Заздалегідь визначений процес, що складається з однієї або кількох операцій чи кроків програми, які визначені в іншому місці (у підпрограмі, модулі)
	Циклічний процес, де кількість циклів відома заздалегідь або визначається кроком лічильника
	Коментар
	З’єднувач блоків

Діаграми Нассі-Шнейдермана (N-S-діаграми) – це схеми, що ілюструють передачу керування в алгоритмі за допомогою вкладених один в одного блоків. N-S-діаграми більш компактні за рахунок відсутності явної вказівки ліній переходу за керуванням. Кожен блок має форму прямокутника і може бути вписаний у будь-який внутрішній прямокутник будь-якого іншого блоку (табл. 3.20). Однією з ключових переваг діаграм Нассі-Шнейдермана є їхня відповідність принципам структурного програмування.

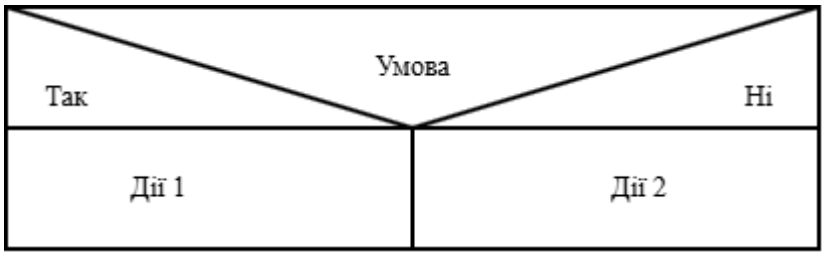
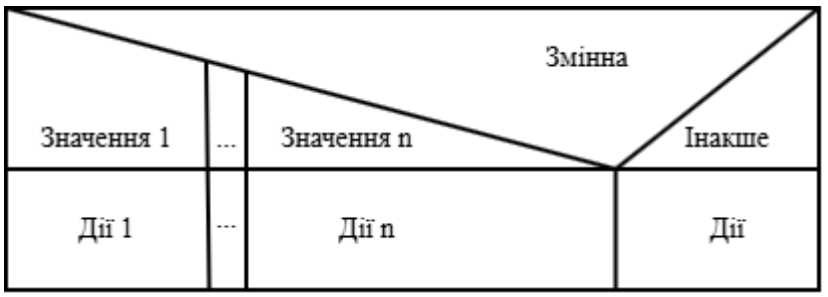
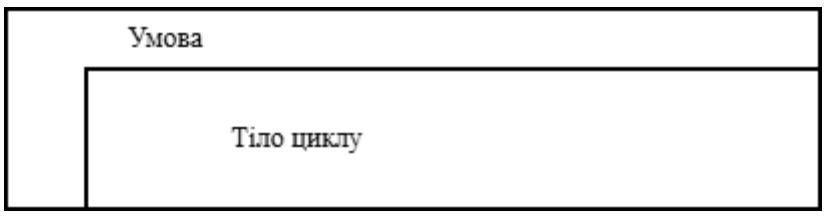
Вибір конкретного підходу до візуалізації алгоритму залежить від мети опису та складності логіки. Для високорівневого представлення загальних процесів доцільно використовувати блок-схеми. Їхня головна перевага - наочність напрямку потоку управління, що робить їх зручними для швидкого ознайомлення з логікою роботи програми. Проте при проєктуванні складних систем блок-схеми часто стають громіздкими, а велика кількість ліній переходів («спагеті-логіка») може приховати структуру алгоритму, що ускладнює його аналіз та подальшу реалізацію.

Діаграми Нассі-Шнейдермана є професійним інструментом, що найкраще підходить для документування складної бізнес-логіки в межах парадигми структурного програмування (завдяки жорсткій ієрархічній вкладеності блоків). Використання цього підходу у кваліфікаційній роботі

унеможливлює створення неструктурованого коду та привчає до проєктування логічно завершених і читабельних програмних модулів.

Таблиця 3.20

Елементи діаграми та їхнє призначення

Елемент	Смислове навантаження
	Лінійна послідовність дій
	Розгалуження
	Вибір за значенням
	Цикл з постумовою
	Цикл з передумовою

В табл. 3.21 наведено зіставлення особливостей даних підходів.

Правильно обраний метод візуалізації дозволяє виявити логічні суперечності ще на етапі проєктування, що значно зменшує кількість помилок у вихідному тексті програми.

Таблиця 3.21

Порівняльна характеристика засобів візуалізації алгоритмів

Критерій порівняння	Блок-схема	Діаграма Нассі-Шнейдермана
Візуальна наочність	Висока для лінійних процесів; напрямок виконання чітко вказаний стрілками	Висока для ієрархічних структур; фокусує увагу на вкладеності логічних блоків
Структурна цілісність	Допускає довільні переходи	Жорстко дотримується принципів структурного програмування; унеможливлює неструктуровані переходи
Компактність	Займає значну площу; при ускладненні алгоритму стає громіздкою та важкою для читання	Максимально економить простір на сторінці за рахунок відсутності зовнішніх ліній-зв’язків
Масштабованість	Низька; додавання нової умови часто вимагає повного перекомпонування всієї схеми	Вища; дозволяє легко додавати вкладені рівні логіки без порушення загальних меж діаграми
Сфера застосування	Опис загальних бізнес-процесів, презентаційні матеріали	Професійне проєктування складної бізнес-логіки програмних модулів та функцій

3.4.3 Вимоги до текстів програм

Питання якості є актуальним протягом усіх стадій життєвого циклу програмного забезпечення. Про якість вихідного коду можна судити за такими параметрами:

- читабельність;
- легкість у тестуванні та налагодженні, модифікації;
- економне використання ресурсів: пам'яті, процесора, дискового простору;
- відсутність зауважень компілятора;
- відсутність блоків, які підлягають рефакторингу: невикористовуваних змінних, недосяжних блоків коду, дублювання коду тощо, а також надмірних коментарів.

До текстів програм, які створюються в ході виконання кваліфікаційної роботи висуваються вимоги до:

– форматування тексту програми;
– вмісту програмного коду;
– стиль іменування програмних сутностей;
– наявність коментарів.

Правильне форматування програми значно полегшую читання коду, дозволяє візуально оцінити складність і логічну структуру програми, особливості використаних рішень: керуючі структури, умови тощо. До форматування висуваються такі вимоги:

– в тексті програми використовуються однакові відступи;
– в прототипах функцій вказуються імена аргументів;
– у виразах використовуються пробіли та круглі дужки (у логічних);
– в рядку міститься по одному оператору, за винятком складених операторів, наприклад, циклу for, тернараного оператору.

До вмісту програмного коду висуваються такі вимоги:

– функція (метод) повинна мати лише одне призначення, клас - єдину відповідальність;
– відсутніх повторних обчислень за межами циклу;
– в програмі не використовуються глобальні змінні;
– коротший блок умовного оператора if/else розміщений першим в операторі;
– змінні ініцалізовано;
– в програмі присутня перевірка значень, які повертають функції;
– в програмі немає витоку динамічної пам’яті;
– способи передачі аргументів до функцій обґрунтовані;
– для класів дотримано принципу інкапсуляції, його порушення має обгрунтування, пов’язане з вимогами до розроблюваної програми.

Дотримання стилю іменування програмних сутностей не лише підвищує читабельність, а й полегшує розуміння коду. До стилю можна висунути такі вимоги:

– імена ідентифікаторів відображають їх призначення, не містять абревіатур, окрім загальноприйнятих, написані англійською мовою;
– присутній стиль іменування кожного виду програмних стуностей: типів даних, класів, інтерфейсів та їх членів, змінних та констант, функцій тощо.

Коментарі покликані на спрощення розуміння основних ідей, реалізованих у коді, та призначення програмних сутностей. При цьому їх кількість має бути мінімальною та достатньою для розуміння коду. Доречним є розміщення коментарів до:

– функцій (методів) у вигляді регламентованих специфікацій (у форматі, передбаченому IDE);

– модулів/пакетів щодо їх призначення;
– основних керуючих конструкцій (при наявності складної логіки, в тому числі виходічи з методу покрокової деталізації).

Не слід розміщувати коментарі, що:
– підтверджують код, а не дають пояснення до нього;
– дублюють іменування ідентифікаторів (пояснюють те, що зрозуміло з назви).

Текст програми оформлюється у стандартизованому виді документу «Текст програми» та додається до тексту пояснювальної записки як додаток. Не слід наводити тексти окремих класів чи модулів, що не є напрацюваннями здобувача, зокрема вміст бібліотек, фреймворків тощо. Не є обов'яковим наведення тексту програми, згенерованого при розобці елементів інтерфейсу користувача за допомогою компонентів відповідної панелі інструментів IDE.

3.5 Тестування та налагодження

У цьому розділі пояснювальної записки доцільним є демонстрування процесів тестування та налагодження програмного забезпечення. При виборі програмних сутностей (функцій, методів класів) для тестування слід спиратися на основні прецеденти, описані в діаграмі варіантів використання та обирати підмножину методів класу (функцій модулю), що реалізують прецеденти. Перевагу слід надати сутностям, на які покладено основний функціонал ПЗ та які реалізують нові або модифіковані алгоритми.

Однією з головних складових процесу тестування є розроблення тестів (тестових випадків). Для цього може бути застосовано методи білої та чорної скриньки [14, 15]. До перших відносяться:
– покриття операторів;
– покриття рішень;
– покриття умов;
– покриття умов та рішень;
– комбінаторного покриття умов.

Вибір методу залежить від критичності коду, наявних ресурсів та вимог до якості. Також слід брати до уваги структуру тексту програми: наявності керуючих конструкцій коду, їх вкладень, використання складених умов.

Метод покриття операторів слід обрати для базової перевірки коду, де логіка лінійна і проста. Це найслабший метод, він легко пропускає помилки в логічних розгалуженнях, тому застосовується як мінімально допустимий для гарантування якості ПЗ.

Метод покриття рішень застосовується для бізнес-застосунків, які вирішують типові завдання. Метод є більш надійним, ніж покриття операторів.

Метод покриття умов доцільно застосовувати при наявності складених умов. Цей метод рекомендовано у випадках, коли важливо перевірити кожен компонент умови незалежно один від одного. Але повне покриття умов не гарантує повне покриття рішень.

Метод покриття умов та рішень доцільно застосовувати для тестування систем з підвищеними вимогами до надійності, зокрема в області медицини, авіоніки, складних фінансових систем.

Метод комбінаторного покриття умов є ефективним лише за наявності складених умов у керуючих конструкціях. Рекомендовано застосовувати лише для найвідповідальніших ділянок коду, оскільки потребує 2^n тестів, при n компонентів складеної умови.

До методів чорної скриньки належать:

– еквівалентне розбиття;

– аналіз граничних значень (умов);

– припущення про помилку;

– функціональні діаграми;

– перехід за станами.

Метод еквівалентного розбиття є ефективним, якщо вхідні дані мають великі діапазони (наприклад, вік, сума переказу, кількість товару). Використання цього методу робить процес складання тестів більш раціональним, значно зменшуючи їх кількість.

Аналіз граничних значень слід застосовувати завжди, коли є числові межі, у тому числі обмеження на довжину рядка.

Метою застосування методу припущення про помилку є виявлення нестандартних дефектів, які не вписуються в логічні моделі (наприклад, введення літер у поле для цифр або натискання кнопки «Оплатити» десять разів поспіль). Тому його доцільно застосувати наприкінці циклу тестування або коли систему потрібно перевірити «на міцність» (Stress/Monkey testing).

Для уникнення плутанини у складних бізнес-правилах, що супроводжуються отриманням результату на основі комбінації кількох умов, доцільно використовувати метод функціональних діаграм.

Метод переходу за станами доречно застосувати при тестуванні систем із чіткою послідовністю дій з метою перевірки можливості користувача оминути певні кроки послідовності.

Рекомендовано для тестування використовувати поєднання методів білою та чорної скриньок.

У пояснювальній записці слід навести перелік обраних методів тестування, сутностей ПЗ для тестування та обгрунтування такого вибору. Для кожної сутності необхідно навести дефінітну специфікацію, яка

включає опис призначення (функціоналу), вхідних і вихідних даних з зазаченням назв параметрів, їх змісту, діапазону допустимих значень. Для методів білої скриньки слід навести програмний код, який буде протестовано. Його можна розмістити у додатку “Текст програми”, а в пояснювальній записці дати посилання на відповідний додаток та його пункт. Після цього необхідно подати опис тестів як наборів значень вхідних та вихідних даних програмної сутності, яка тестується, для разового її виконання (виконання програми). Кожен тест повинен мати унікальний ідентифікатор.

Після приведення тестів у тексті поояснювальнї записки слід навести демонстрацію відповіності тестів обраним методам.

Для демонстрації відповідповідності методам білої скриньки слід навести таблиці покриття (див. приклади табл. 3.22 – 3.23). Якщо у таблицях використано умовні позначення, наприклад, умова 1, рішення 2, або використано позначення операторів (нумерація), то перед таблицею слід дати пояснення щодо їх змісту. У прикладах зірочками позначено покриття відповідної конструкції коду (оператора, умови, рішення), їх розміщення має ілюстративний характер.

Таблиця 3.22

Покриття операторів методу [Назва методу]

Ідентифікатор теста / Номер оператора	T1	...	Tn
1	*		
2	*		*
...			*
m			*

Таблиця 3.23

Покриття умов методу [Назва методу]

Умова / Іденти–фікатор тесту	Умова 1		...		Умова n	
	true	false			true	false
T1	*	*				
...			*			
Tk				*	*	*

Таблиці покриття рішень та умов і рішень за структурою є аналогічними до таблиці покритт умов. Відмінність полягає у позначеннях конструкцій коду, які покриваються.

При комбінаторному покритті умов слід позначити виконання кожної з комбінацій. Якщо деяка комбінація недосяжна, то слід це відзначити та пояснити причини такого явища. У табл. 3.24 наведено структуру покриття для умови, що складається з двох частин (наприклад, A==0 && B!=0). Як було зазначено раніше, кількість комбінацій залежить від кількості умов.

Для демонстрації відповідності тестів методам чорної скриньки слід показати, якому елементу (класу еквівалентності, граничній умові, припущенню тощо) відповідає кожен з тестів.

Приклад оформелення відповідності тестів за методом еквівалентного розбиття подано у табл. 3.25, 3.26.

Таблиця 3.24

Комбінаторне покриття умов методу [Назва методу]

	Запис умови			
Комбінація / Ідентифікатор тесту	true-true	true-false	false-true	false-false

Таблиця 3.25

Класи еквівалентності для методу [Назва методу]

Вхідні умови	Правильні класи	Неправильні класи
Опис 1 (що позначає змінна 1, відносно значень якої виділяються класи)	Опис правильного класу (номер класу)	Опис неправильного класу (номер класу)
...	...	...
Опис n (що позначає змінна n, відносно значень якої виділяються класи)	Опис правильного класу (номер класу)	Опис неправильного класу (номер класу)

Таблиця 3.26

Тести за методом еквівалентних розбиттів для [Назва методу]

Номер класу, що покривається	Ідентифікатор тесту	Значення вхідних параметрів	Значення вихідних параметрів

При використанні методу аналізу граничних значень слід перелічити всі границі та вказати ідентифікатори тестів, які їх покривають за принципом: значення границі, по одному значенню з обох боків від границі.

Застостосовуючи метод припущення про помилку усі припушення слід пронемуревати. Далі для кожного припущення вказати ідентифікатор тесту, який відповідає припущенню.

Для методу функціональних діаграм слід навести нумеревані переліки всіх причин та наслідків, привести функціональну діаграму, що містить позначки номерів з перелік. Далі для кожної допустимої комбінації причин вказати ідентифікатори тестів, які їх задовольняють.

При використанні методу переходу за станами слід навести діаграму станів, ідентифікувати на ній допустимі маршрути переходів та для кожного такого маршруту зазначити ідентифікатор тесту, який дозволяє виконати прохід за маршрутом.

Після розроблення тестів, наведення їх опису та відповідності обраним методам, слід перейти до виконання тестів. Результати виконання тестів слід навести у пояснювальній записці у вигляді переліку ідентифікаторів тестів з помітками “пройдений/непройдений”.

Якщо тест непройдений, необхідно виконати його аналіз. При виявленні невідповідності очікуваних результатів роботи сутності, яка тестується, з отриманими, слід перевірити коректність тесту по відношенню до специфікації сутності. Якщо тест не відповідає специфікації, необхідно внести корективи у тест та повторити його виконання. Якщо тест складено коректно, слід перейти до налагодження програми.

Для ефективного налагодження рекомендується слідувати таким принципам:

– перевірка помилки на стійкість (чи завжди вона проявляється);
– реєстрація власних помилок;
– починати перевірку з більш простих гіпотез;
– нічого не приймати на віру: все необхідно перевіряти за допомогою тестових прикладів;
– вироблення системного підходу.

Для налагодження програм і функцій може бути застосовано такі засоби та інструменти:

– комплекс засобів, передбачений IDE: покрокове виконання, точки зупину з різними логічними умовами, перегляд значень змінних, асемблерного коду;
– включення коду в програму – метод вставок: вставка друку для налагодження (вивід значень змінних та текстових повідомлень), зняття дампа пам'яті та подальший їх аналіз;
– тестові драйвери і заглушки;
– спеціальні програми для налагодження.

Результати налагодження програми слід навести у формі, представленій у табл. 3.27. Опис помилки передбачає визначення у чому полягає розбіжність тестованої програмної сутності зі специфікацією. Опис умов містить набори вхідних даних, за яких сталася помилка. Способи усунення включають одну або декілька гіпотез з приводу можливих дій для усунення помилки.

Таблиця 3.27

Протокол налагодження програми

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення

3.6 Підготовка та проведення експериментів

Якщо функціональне призначення розробки полягає у наданні інструментарію для дослідження, то доцільним є включення до основної частини пояснювальної записки розділу про проведення експериментів.

Цей розділ має складатися з опису мети експерименту, контрольних показників (метрик), схеми процесу проведення експерименту, експериментальної бази та отриманих результатів.

Опис контрольних показників (метрик) повинен містити визначення критеріїв, за якими оцінюється ефективність розробленого ПЗ. Наприклад, якщо це ПЗ для аналізу даних, доцільно використовувати такі показники, як точність, повнота або час виконання операцій. Необхідно пояснити фізичний чи математичний зміст кожного показника у контексті поставленої задачі. Чітка система показників дозволяє перевести суб'єктивні враження від роботи програми у площину об'єктивних цифрових даних.

Схема процесу проведення експерименту має регламентувати доступні сценарій дій (операцій) при проведенні експерименту. Тут зазначається в якому порядку та як вимірюється кожен з показників.

Опис експериментальної бази повинен містити опис даних, на яких проводиться експеримент: кількість (наприклад, кількість записів у таблиці, кількість файлів), розмір (байти, КБайти...), характер (що саме містять дані, чи є вони однорідними, структурованими, впорядкованими тощо).

Якщо на значення вимірювального показника має вплив апаратне забезпечення, на якому виконується експеримент, слід навести його характеристики. Зокрема може бути наведено інформацію про тип та розмір оперативної пам’яті, частоту та кількість ядер центрального процесора, розмір кешу різних рівнів, розмір відеокарти тощо - в залежності від особливостей обчислювальних процесів, для яких проводиться експеримент.

Отримані результати варто візуалізувати та інтерпретувати. Статистичні дані найкраще сприймаються у вигляді таблиць, графіків або діаграм, які обов'язково мають бути прокоментовані. Тип діаграми та графіку доречно обирати відповідно до типів даних (впорядковані, кількісні, категорійні) та їх характеру [16]. Недостатньо просто вставити графік — потрібно пояснити аномалії, тренди або характерні точки вигину. Здобувач має проаналізувати, чому програма повелася саме так у конкретних умовах, і чи підтвердилися початкові гіпотези. Візуалізація допомагає швидко вловити суть досягнутого наукового чи практичного результату.

Завершити цей розділ слід лаконічними висновками, які резюмують результати експериментів. Потрібно чітко зазначити, чи вдалося досягти поставленої мети дослідження за допомогою розробленого інструментарію. Також варто вказати на обмеження розробки (якщо вони були виявлені) та окреслити потенційні напрямки подальших досліджень. Це продемонструє критичне мислення здобувача та глибоке розуміння предметної області.

3.7 Висновки та рекомендації

Розділ «Висновки та рекомендації» є логічним завершенням кваліфікаційної роботи, де підбиваються підсумки проведеного дослідження та розробки. Основна мета цього розділу — продемонструвати, що поставлені завдання виконані, а мета роботи досягнута. Висновки мають базуватися на результатах кожного розділу: від аналізу вимог та архітектурного проєктування до етапів тестування готового програмного засобу. Важливо уникати загальних фраз, натомість

зосередитися на конкретних технічних та функціональних досягненнях проєкту. Посилання на інших авторів, їх цитування, а також наведення загальновідомих фактів у цьому розділі не припустиме.

У першій частині висновків необхідно надати стислий аналіз отриманих результатів. Варто зазначити, які саме UML-діаграми стали основою для побудови архітектури. За наявності у роботі бази даних слід вказати, як спроєктована база забезпечує цілісність і швидкість обробки інформації. Доречно акцентувати увагу на виборі технологічного стеку (мови програмування, фреймворків), обґрунтувавши їхню ефективність для реалізації розроблених алгоритмів. Окремо виділяються результати тестування, які підтверджують працездатність програми та її відповідність вихідним вимогам замовника або специфікації.

Особливу увагу слід приділити порівняльному аналізу розробленого ПЗ з існуючими аналогами. Необхідно чітко сформулювати конкурентні переваги розробки. Серед них може бути вища продуктивність, інтуїтивно зрозумілий інтерфейс, нижчі системні вимоги, наявність специфічного функціоналу або краща адаптація під потреби конкретної галузі чи групи користувачів. Аргументація переваг має бути підкріплена даними, отриманими під час аналізу вимог та безпосередньої розробки, що підкреслює практичну цінність роботи.

Заключна частина розділу присвячується рекомендаціям щодо впровадження та окресленню перспектив подальшого розвитку проєкту. Доцільно вказати вектори майбутньої модернізації: наприклад, масштабування архітектури, додавання нових модулів, інтеграція з іншими системами або перехід на хмарні технології. Такі рекомендації свідчать про глибоке розуміння автором життєвого циклу ПЗ та його здатність бачити потенціал власної розробки у довгостроковій перспективі.

Типові помилки у висновках до кваліфікаційної роботи. Найпоширенішою помилкою є невідповідність висновків тому, що було задекларовано у постановці задачі. Кожне завдання, поставлене на початку роботи, має отримати своє відображення у підсумках (наприклад: «проаналізовано вимоги», «спроєктовано архітектуру», «реалізовано алгоритм»).

Слід уникати використання висловів без конкретики, наприклад, «програма працює добре» або «код написаний якісно» . Краще вказувати технічні параметри: «програма забезпечує обробку запитів за ... мс», «реалізовано підтримку ... одночасних користувачів» або «база даних приведено до третьої нормальної форми».

Висновки не повинні бути вставкою тексту з попередніх частин роботи. Замість опису того, як було зроблено, слід написати що це дало в результаті для цілісності системи.

Якщо у висновках не згадано про унікальність чи ефективність власного рішення порівняно з іншими (наприклад, за критеріями вартості, швидкодії чи інтерфейсу), робота втрачає свою практичну вагу.

Наведені рекомендації щодо майбутніх напрямів роботи мають бути логічним кроком її розвитку.

3.8 Завдання та запитання до розділу 3

1. Надайте рекомендації щодо змісту вступу пояснювальної записки.
2. Опишіть структуру реферату.
3. Поясність відмінності між функціональними та нефункціональними вимогами до ПЗ.
4. Перелічіть джерела вимог до ПЗ. Для кожного з джерел наведіть приклади.
5. Які існують методи збору вимог до ПЗ?
6. Який зміст має бути у пункті “Постановка задачі” пояснювальної записки?
7. Визначте основні задачі зовнішнього проєктування.
8. Визначте основні задачі внутрішнього проєктування.
9. Які існують UML-засоби для моделювання структури системи?
10. Які засоби UML можуть бути застосовані для моделювання поведінки системи?
11. В яких випадках доцільним є побудова діаграми розгортання?
12. У чому полягає суть нормалізації бази даних?
13. У чому полягає суть ER-моделювання? Перелічіть основі елементи ER-моделі.
14. Перелічіть основні форми представлення алгоритму. Визначте переваги та недоліки кожної з них.
15. Поясніть суть методу покрокової деталізації. Назвіть форми представлення алгоритму, розробленого за цим методом.
16. Дайте рекомендації щодо використання різних форм представлення алгоритму, розробленого за методом покрокової деталізації.
17. Поясніть суть методу аналізу повідомлень. Визначте його ключові відмінності від методу покрокової деталізації.
18. Назвіть показники, за якими може бути оцінено якість тексту програми.
19. Які існують вимоги до форматування та вмісту програмного коду?
20. Які існують вимоги до іменування програмних сутностей? Наведіть приклади іменування, що відповідають та не відповідають цим вимогам.
21. У чому полягає суть процесу тестування програми?

22. Поясніть відмінність між методами тестування білою та чорною скринькою?
23. Перелічіть методи тестування білою скринькою.
24. Перелічіть методи тестування чорною скринькою.
25. У чому полягає суть процесу налагодження програми?
26. Назвіть принципи налагодження програм.
27. Які існують засоби налагодження програм у сучасних IDE? Наведіть приклади, спираючись на одне з середовищ розробки.
28. Яку основну інформацію слід навести у розділі “Висновки та рекомендації”?
29. Яка основна мета викладення інформації розділу “Висновки та рекомендації”?
30. Сформулюйте особливості викладення рекомендацій щодо впровадження та окресленню перспектив подальшого розвитку проєкту.

4 СУПРОВІДНІ МАТЕРІАЛИ

Супровідні матеріали включають у себе мультимедійну прентацію та демонстраційне відео роботи програми. Вони використовуються при захисті кваліфікаційної роботи під час доповіді здобувача щодо основних отриманих результатів.

4.1 Структура доповіді та презентації

Структура доповіді та презентації має логічно повторювати розділи пояснювальної записки, проте з акцентом на власні досягнення здобувача та практичну цінність. Рекомендовано підготувати 12–15 слайдів: від обґрунтування актуальності та аналізу предметної області до демонстрації інтерфейсу та результатів тестування. Особливу увагу в доповіді варто приділити етапу проєктування (UML-діаграмам) та опису ключових алгоритмів, оскільки це демонструє кваліфікацію здобувача.

У частині, що стосується розробки, важливо не просто перелічити технології (Python, React, PostgreSQL тощо), а пояснити, чому саме цей стек є оптимальним для вирішення поставлених завдань. Не рекомндується наводити слайди з кодом. Доречними є високорівневі схеми архітектури або фрагменти критично важливих алгоритмів.

Презентація має містити демонстрацію роботи програми - короткий відеозапис, що ілюструє виконання основних функцій розробленого ПЗ.

Розділ тестування у доповіді має підтверджувати надійність системи. У презентації слід зазначити методи тестування, відомості щодо покриття ПЗ тестами.

На завершальних слайдах варто акцентувати увагу на перевагах перед існуючими рішеннями: зручність інтерфейсу, швидкодія, розв’язання специфічної проблеми, яка раніше не була автоматизована, тощо.

Висновки в доповіді мають бути лаконічними: що було зроблено, які результати отримано та як проєкт можна розвивати далі. Пам’ятайте, що презентація — це візуальна підтримка, тому уникайте великих масивів тексту на слайдах; використовуйте графіки, схеми та діаграми, які легше сприймаються аудиторією під час вашої промови.

4.2 Оформлення презентації

Основою якісного дизайну презентації є лаконічність. Надмірне нагромадження елементів створює візуальний шум, що заважає аудиторії сприймати головні тези. Для удосконалення презентацій варто дотримуватися певних правил [16].

Мінімізація брендування та обрамлення: логотипи доречні на титульному слайді, але на робочих сторінках вони лише відволікають від змісту. Також слід уникати зайвих рамок навколо ілюстрацій. Якщо фон зображення контрастує з фоном слайда, краще змінити колір фону або розтягнути картинку на весь екран.

Зрозумілість візуалізації: слід відмовитися від об'ємних 3D-графіків для простих даних, оскільки вони викривлюють сприйняття показників. Не слід накладати графіки на фонові зображення, оскільки це заважає зчитуванню інформації. Головне правило: якщо елемент можна видалити без втрати сенсу – його треба видалити.

Читабельність: зважаючи на особливості проєкторів та відстань до глядачів, слід використовувати великі кеглі. Для заголовків – від 24 пт, для основного тексту – не менше 18 пт.

Пріоритет візуального над текстовим: слайд має містити лише те, що складно пояснити словами (схеми, фото, діаграми). Не варто дублювати на екрані весь текст промови.

Презентація повинна містити титульний слайд з темою кваліфікаційної роботи та коротку інформацію про автора: ім’я здобувача у формі, в якій можливе звернення до нього, назву (номер) академічної групи.

Кожен слайд повинен мати унікальний заголовок, що описує основну ідею цього слайду. Всі заголовки повинні мати єдине форматування, у кінці заголовку крапка не ставиться. Кожен слайд має передавати одну думку.

Зображення на слайдах мають бути підписані (в окремих випадках заголовок слайду може слугувати таким підписом). Навколо всіх об’єктів на слайдах варто залишати поля, бажано однакової ширини.

На кожному слайді, крім титульного, слід розмістити його номер, аби потім можна було на нього посилатися при обговоренні доповіді.

4.3 Зміст демонстраційного відео

Демонстраційне відео є ключовим супровідним матеріалом, що підтверджує працездатність розробленого ПЗ та реалізацію заявлених у постановці задачі функцій. Відеоролик має бути лаконічним (рекомендована тривалість – до 3 хвилин) і фокусуватися виключно на

головних сценаріях використання ПЗ. Відео не повинно містити звукових доріжок, музики чи записаних коментарів. Здобувачу слід заздалегідь підготувати усний супровід, який він озвучуватиме під час захисту синхронно з відеорядом.

Поради щодо змісту та структури відео. Починати слід із головного екрана та продемонструвати виконання 3–4 ключових операцій (наприклад, створення запису, обробка даних, генерація звіту). Не слід акцентувати увагу на етапах реєстрації та авторизації користувачів.

Для демонстрації слід використовувати заздалегідь підготовлені тестові дані, які виглядають природно. Варто уникати порожніх таблиць або тестових значень виду «111», «asdf», «тест» тощо.

Слід обов'язково показати кінцевий результат роботи алгоритму (зміну статусу, появу файлу, візуалізацію на графіку), щоб було видно завершеність кожного процесу.

Для запису відео слід використовувати доступне ПЗ, яке дозволяє захопити вказане вікно програми. Записувати слід лише вікно програми або конкретну область, щоб уникнути потрапляння в кадр зайвих елементів робочого столу, сповіщень чи панелі завдань.

Відео має бути чітким, щоб текст та елементи інтерфейсу користувача були розбірливими при відтворенні через проєктор. Слід налаштувати відображення курсора миші (за наявності), зокрема додати підсвічування кліків для спрощення відстеження переміщень та взаємодії з інтерфейсом.

До типових помилок, яких слід уникати, відносяться затягнутість, низька динаміка, дрібні шрифти.

Не слід показувати процеси, які тривають довго (інсталяція ПЗ, тривале завантаження даних, реєстрація з підтвердженням поштою). Такі моменти варто вирізати або пришвидшити при монтажі. Паузи у відео, де нічого не відбувається, створюють враження «зависання» програми. Відео має бути динамічним і чітко слідувати за розповіддю. Якщо інтерфейс програми містить багато дрібних деталей, під час запису краще збільшити масштаб (DPI) операційної системи, щоб елементи були помітними на великому екрані.

4.4 Завдання та запитання до розділу 4

1. Які матеріали є супровідними до пояснювальною записки?
2. Які структурні елементи мають містити доповідь та презентація для захисту кваліфікаційної роботи?
3. Назвіть основні правила оформлення презентації.
4. Чому на робочих слайдах презентації рекомендується уникати наскрізного брендування (логотипів на кожній сторінці) та зайвого обрамлення ілюстрацій?

5. Які обов’язкові елементи (крім контенту) мають бути присутні на кожному слайді презентації для полегшення подальшого обговорення доповіді?
6. На чому має фокусуватися демонстраційне відео?
7. Які дані для роботи ПЗ слід використовувати при записі демонстраційного відео?
8. Яка рекомендована тривалість відеоролика і як саме має відбуватися його коментування під час процедури захисту (зважаючи на вимоги до звукових доріжок)?
9. Назвіть основні помилки при створенні демонстраційного відео.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Положення про виконання кваліфікаційної роботи в Українському державному університеті науки і технологій. Наказом від 06.12.2023 № 81. Дніпро : УДУНТ, 2023. 47 с. URL: https://ust.edu.ua/wp-content/uploads/2024/06/polozhennya-pro-vykonannya-kvalifikaczijnoyi-roboty-v-udunt.pdf (дата звернення: 29.03.2026).
2. Положення про порядок перевірки академічних текстів на плагіат в Українському державному університеті науки і технологій. Наказ від 30.01.2025 № 18. Дніпро : УДУНТ, 2025. 25 с URL: https://ust.edu.ua/wp-content/uploads/2025/02/polozhennya-pro-poryadok-perevirky-akademichnyh-tekstiv-na-plagiat-v-udunt_compressed.pdf (дата звернення: 29.03.2026).
3. *ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки.* На заміну ДСТУ 3008-95 ; чинний від 2017-07-01. Вид. офіц. Київ : УкрНДНЦ, 2017. 31 с. (дата звернення: 30.03.2026).
4. *ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання*. Чинний від 2016-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.
5. Пістунов І. М. Моделювання бізнес процесів : навч. посіб. Дніпро : НТУ «Дніпровська політехніка», 2021. 130 с. URL: http://pistunovi.inf.ua/MOD_BIZ_IIPOU.pdf (дата звернення: 17.03.2026).
6. Rumbaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual. 2nd ed. Boston : Addison-Wesley Professional, 2005. 721 p.
7. Unified Modeling Language Specification. Version 2.5.1. *Object Management Group*. 2017. URL: https://www.omg.org/spec/UML/2.5.1/About-UML (date of access: 20.03.2026)
8. Все, що ви хотіли знати про принципи SOLID. Ч. I: SRP. *DOU*. 2025. URL: https://dou.ua/forums/topic/52394/ (дата звернення: 20.03.2026).
9. Martin R. C. Clean Architecture. A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2018. 364 p.
10. Патерни проектування. *Refactoring. Guru*. URL: https://refactoring.guru/uk/design-patterns (дата звернення: 20.03.2026).
11. Остапченко К. Б. Бази даних. Комп‘ютерний практикум : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2022. 151 с. URL: https://ela.kpi.ua/items/3387c2aa-332c-4a57-897f-98acd45cef01 (дата звернення: 29.03.2026).

12. Харів Н. О. Бази даних та інформаційні системи : навч. посіб. Рівне : НУВГП, 2018. 127 с.
13. Демиденко М. А. Введення в сучасні бази даних : навч. посіб. Дніпро : НТУ «Дніпровська політехніка», 2020. 38 с.
14. Авраменко А. С., Авраменко В. С., Косенюк Г. В. Тестування програмного забезпечення : навч. посіб. Черкаси : ЧНУ імені Богдана Хмельницького, 2017. 284 с.
15. Якість та тестування інформаційних систем : навч. посіб. / Золотухіна О. А., Негоденко О. В., Резник С. Ю., Разіна С. Я. Київ : ННІІТ ДУТ, 2020. 128 с.
16. Основи документування та представлення проєктів : навч.-метод. рек. до лаб. робіт / упоряд. О. С. Куроп‘ятник. Дніпро : УДУНТ, 2024. 36 с.

Додаток А

Форма титульного аркуша кваліфікаційної роботи

Міністерство освіти і науки України
Український державний університет науки і технологій
Факультет “Комп'ютерних технологій і систем”
Кафедра “Комп’ютерні інформаційні технології”

Пояснювальна записка

до кваліфікаційної роботи
ОС Бакалавр

на тему _____
_____

за освітньою програмою Інженерія програмного забезпечення
зі спеціальності F2 (121) Інженерія програмного забезпечення

Виконав: студент групи _____ _____ /_____/
(номер групи) (підпис студента) (Ім’я ПРІЗВИЩЕ)

Керівник _____ /_____/
(підпис) (посада, Ім’я ПРІЗВИЩЕ)

Нормоконтролер _____ /_____/
(підпис) (посада, Ім’я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис студента)

Дніпро - 20___

Додаток Б

Форма титульного аркуша кваліфікаційної роботи англійською мовою

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies
Faculty «Computer technology and systems»
Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: _____

_____

according to educational curriculum Software engineering
in the Speciality F2 (121) Software engineering

Done by the student of the group _____ /_____/
(group number) (name, surname)

Scientific Supervisor /_____/
(position, name, surname)

Normative controller /_____/
(position, name, surname)

Dnipro – 20___

Додаток В

Форма завдання на кваліфікаційну роботу

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерних технологій і систем»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: перший (бакалаврський)
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: F2 (121) Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
«Комп'ютерні інформаційні технології»

_____ /_____/
(підпис) (Ім'я ПРІЗВИЩЕ)

«_____» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу ОС Бакалавр
студенту _____
(Прізвище, Ім'я, По батькові)

1 Тема роботи: _____
_____
Керівник роботи _____
(Прізвище, Ім'я, По батькові, науковий ступінь, вчене звання)

затверджені наказом від «___» _____ 20__ р. № _____.
2 Строк подання студентом роботи «___» _____ 20__ р.
3 Вихідні дані до роботи _____
_____
4 Зміст пояснювальної записки (перелік питань, які потрібно опрацювати)
_____
_____
_____
5 Перелік демонстраційного матеріалу _____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Зміст роботи (розділу)	Строк виконання етапів роботи	Примітка
1			
2			
3			
4			
5			
6			
7	Подання кваліфікаційної роботи до кафедри		
8	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії		

Студент _____ /_____/
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ /_____/
(підпис) (Ім'я ПРІЗВИЩЕ)

Додаток Г

Приклади оформлення реферату кваліфікаційної роботи

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на здобуття ОС Бакалавр: 80 с., 10 рис., 8 табл., 4 додатки, 20 джерел.

Об’єкт розробки – процес автоматизації складського обліку на підприємстві.

Предмет розробки – вебзастосунок для моніторингу залишків товарів.

Мета роботи – підвищення ефективності керування складськими запасами шляхом впровадження вебзастосунку для моніторингу товарних залишків у режимі реального часу.

Методи розв'язання задачі: аналіз аналогів та огляд літератури, опитування зацікавлених сторін для формування вимог до програмного засобу; структурне моделювання та моделювання динаміки системи засобами UML; методи тестування «чорного» та «білого» ящика.

У роботі обґрунтовано актуальність автоматизації складських процесів. Проведено збір та аналіз вимог зацікавлених сторін, виконано огляд існуючих програмних рішень. Здійснено зовнішнє і внутрішнє проєктування системи, включаючи формалізацію задачі, опис вхідних і вихідних даних, побудову ескізів екранних форм інтерфейсу користувача. Методом покрокової деталізації побудовано алгоритми вирішення основних задач. Розроблено вебзастосунок для моніторингу товарних залишків. Проведено тестування та налагодження вебзастосунку, проаналізовано вплив можливих помилок на стабільність його роботи.

Результати роботи може бути впроваджено на підприємствах для автоматизації обліку товарів, що дозволить мінімізувати помилки ручного введення та оптимізувати логістичні процеси.

Ключові слова: АВТОМАТИЗАЦІЯ СКЛАДСЬКОГО ОБЛІКУ, ВЕБЗАСТОСУНОК, МОНІТОРИНГ ЗАЛИШКІВ ТОВАРІВ, ЗБІР ВИМОГ, ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ, ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОЄКТУВАННЯ, АЛГОРИТМ, ТЕСТУВАННЯ, НАЛАГОДЖЕННЯ.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на здобуття ОС Бакалавр: 97 с., 13 рис., 4 табл., 3 додатки, 55 джерел.

Об’єктом розробки є процеси пошуку та систематизації наукових публікацій відповідно до визначеної тематики.

Предметом розробки є інструментарій для автоматизованого вибору та верифікації кодів УДК, що присвоюються науковим працям.

Мета роботи полягає у створенні програмного продукту, призначеного для підбору та контролю правильності УДК-шифрів наукових праць.

Методи розв’язання задачі. Під час проєктування і реалізації системи було обрано об’єктно-орієнтовану методологію із застосуванням мови моделювання UML та розробкою на Python. Процеси лінгвістичного аналізу реалізовані через алгоритми токенізації, очищення від стоп-слів, морфологічної розмітки та виділення іменованих сутностей на базі інструментів бібліотеки spaCy.

Отримані результати. Створено настільну програму з інтерфейсом командного рядка (CLI) для ідентифікації та перевірки УДК-кодів в англомовних матеріалах. Рішення дозволяє оптимізувати класифікацію масштабних фондів літератури: дані, отримані внаслідок ручної обробки, слугують базою для навчання моделі, яка в подальшому формує автоматичні рекомендації. У заключній частині роботи сформульовано напрями для подальшої модернізації та вдосконалення функціонала програми.

Ключові слова: PYTHON, SPACY, МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, УДК, CLI, ІТЕРАЦІЙНА РОЗРОБКА, АВТОМАТИЗАЦІЯ.

Додаток Д

Форма відгуку керівника роботи

Міністерство освіти і науки України
Український державний університет науки і технологій
Відгук керівника
кваліфікаційної роботи бакалавра

Студент групи _____ _____
(номер групи) (Прізвище, Ім'я, По батькові)

Тема випускної роботи: _____
_____

1. Якісні відмінності кваліфікаційної роботи: _____
_____
_____
_____
_____
_____

2. Зауваження: _____
_____
_____
_____

3. Висновок щодо дотримання академічної доброчесності: _____
_____
_____

Комплексна оцінка кваліфікаційної роботи: _____
_____
_____
_____

Керівник: _____ _____ _____
(посада) (Ім'я, ПРІЗВИЩЕ)

Дата: _____

Додаток Е

Приклади оформлення бібліографічного опису в списку джерел згідно з вимогами ДСТУ 8302:2015

Характеристика джерела	Приклад оформлення
1	2
Книги одного, двох або трьох авторів	1. Мартін Р. Чиста архітектура. Мистецтво розроблення програмного забезпечення / пер. з англ. С. Світличний. Київ: Фабула, 2019. 368 с. 2. Галісєєв Г. В. Системне програмування. Київ: Університет «Україна», 2019. 113 с. 3. Єременко В. С., Защепкіна Н. М. Цифрові вимірювальні прилади: комп'ютерний лабораторний практикум. Київ: Книжкове вид-во НАУ, 2020. 168 с. 4. Шаховська Н. Б., Камінський Р. М., Вовк О. Б. Системи штучного інтелекту: навч. посіб. Львів: Видавництво Львівської політехніки, 2018. 392 с.
Книги чотирьох чи більше авторів авторів	1. Комп'ютерна графіка: навч.-метод. посіб. / Сухарькова О. І. та інш. Харків: НУЦЗУ, 2024. 139 с. 2. Advanced measurement technologies in engineering / R. T. Anderson, M. K. Peterson, S. L. Thompson et al. Chicago : Professional Publishing, 2023. 412 p.
Книга без зазначення автора (з редактором тощо)	Енциклопедія інформаційно-вимірювальних технологій / за ред. В. С. Єременка. Київ : КПІ ім. Ігоря Сікорського, 2023. 567 с.
Статті з журналів	1. Скалозуб, В. В., Горячкін В. М., Мурашов О. В. Комплексні моделі впорядкування мультипослідовностей із нечіткими параметрами. Наука та прогрес транспорту. 2021. № 2 (92). С. 50–64. DOI: 10.15802/stp2021/237291. 2. Куроп'ятник О. С. Моделювання сценарію діалогу для системи виявлення запозичень на основі кольорової мережі Петрі. Системні технології. Дніпро, 2022. Т. 1. № 138. С. 36–47. DOI: 10.34185/1562-9945-1-138-2022-04. 3. Shynkarenko, V., Letuchyi, O., Chyhir, R. Constructive-synthesizing modeling of fractal crystal lattices In: IEEE 18th International Conference on Computer Science and Information Technologies (CSIT). IEEE, 2023. p. 1-4. DOI: 10.1109/CSIT61576.2023.10324251 4. LazyLLM: Dynamic Token Pruning for Efficient Long Context LLM Inference / Q. Fu, M. Cho, T. Merth et al. 2024.

1	2
	P. 1–12. arXiv:2407.14057.
Матеріали конференцій	1. Танасієнко Д. О., Андрющенко В. О. Обробка даних та генерування рекомендацій в маркетинговому тестуванні. Інформаційні технології в металургії та машинобудуванні – ITMM'2025: тези доп. Міжнародної наук.-техн. конф. (м. Дніпро, 23-24 березня 2025 р.). Дніпро, 2025. С. 411–416. DOI: 10.34185/1991-7848.itmm.2025.01.073. 2. Трегуб І. О., Іванов О. П. Дослідження часової ефективності відображення елементів графічного інтерфейсу під .NET. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості і освіті: Тези XIX Міжнародної науково-практичної конференції (Дніпро, 18-19 грудня 2025 р.). Дніпро: УДУНТ, 2025. С. 95.
Законодавчі документи	1. Конституція України : Закон від 28.06.1996 № 254к/96-ВР : станом на 01.01.2024. URL: https://zakon.rada.gov.ua/laws/show/254к/96-вр (дата звернення: 25.02.2026). 2. Про електронні документи та електронний документообіг: Закон України від 22.05.2003 № 851-IV : станом на 16.08.2024. URL: https://zakon.rada.gov.ua/laws/show/851-15 (дата звернення: 29.03.2026).
Стандарти	1. ДСТУ ISO/IEC 12207:2008. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Київ, 2008. 108 с. 2. ДСТУ ISO 27001:2023. Інформаційні технології. Методи забезпечення безпеки. Системи управління інформаційною безпекою. Вимоги. Київ, 2023. 35 с.
Електронні ресурси	*Стаття з вебсайту* 1. Патерни проектування. URL: https://refactoring.guru/uk/design-patterns (дата звернення: 20.03.2026). 2. Code Health – How easy is your code to maintain and evolve? CodeScene. URL: https://codescene.io/docs/guides/technical/code-health.html (дата звернення: 30.03.2026). *Документація мови/фреймворку* 1. React Documentation. URL: https://react.dev (дата звернення: 30.03.2026). 2. Collections Framework Overview. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html (дата звернення: 30.03.2026)

Навчальне видання

Горячкін Вадим Миколайович, Куроп'ятник Олена Сергіївна

ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Навчально-практичний посібник
для підготовки кваліфікаційної роботи
на здобуття ос бакалавр

Електронне видання

Комп'ютерна верстка О. С. Куроп'ятник
Дизайн обкладинки О. С. Куроп'ятник

Експертний висновок склав д-р техн. наук, проф. В. І. Шинкаренко

Зареєстровано НМВ УДУНТ (№ 1.868 від 01.05.2026)

Формат 60х84 $^{1/16}$. Ум. друк. арк. 5,74. Обл.-вид. арк. 5,82.
Зам. № 51

Видавець: Український державний університет науки і технологій.
вул. Лазаряна, 2, ауд. 2216, ауд. 263 (наукова бібліотека)
м. Дніпро, 49010.
Свідоцтво суб'єкта видавничої справи ДК № 7709 від 14.12.2022