Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка програмних засобів аналізу методів захисту JavaScript фреймворків»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1811»

_____ /Ілля ОЛІЙНИК/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Вадим АНДРЮЩЕНКО/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

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

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of software for analyzing methods of protecting JavaScript frameworks»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1811: /Illia OLIINYK/

Scientific Supervisor: /Vadym ANDRIUSHCHENKO/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

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

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра
студенту Олійнику Іллі Олександровичу
1. Тема роботи: «Розробка програмних засобів аналізу методів захисту JavaScript фреймворків»
Керівник роботи: Андрющенко Вадим Олександрович, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Вступ
Збір вимог до програмного забезпечення
Огляд програмних аналогів
JavaScript фреймфорки
Базовий захист JavaScript фреймфорків
Зовнішнє проєктування
Внутрішнє проєктування
Тестування методом «білої скриньки»
Тестування методом «чорної скриньки»
Налагодження програми
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
Презентація
Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі	05.01.2022	
2	Огляд літератури та аналіз аналогів	24.01.2022	
3	Розробка структур вхідних і вихідних даних	02.02.2022	
4	Визначення вимог до програми. Вибір та обґрунтування мови програмування	10.02.2022	
5	Узгодження та затвердження ТЗ	18.02.2022	
6	Розробка та програмування логіки програми	01.03.2022	
7	Розробка і реалізація інтерфейсу користувача	20.03.2022	
8	Відлагодження програми	24.03.2022	
9	Подання кваліфікаційної роботи до кафедри	10.06.2022	
10	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	21.06.2022	

Студент _____ (підпис) Ілля ОЛІЙНИК (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ (підпис) доц. Вадим АНДРЮЩЕНКО (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 7 розділів:
– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;
– збір вимог до програмного забезпечення – у цьому розділі описуються аналоги програми та література по даній предметній області. Складається з 7 сторінок;
– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, наводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 14 сторінок;
– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами «чорного» та «білого» ящика. Також аналіз помилок, їх вплив на систему та вирішення проблеми. Складається з 17 сторінок;
– висновки. Складається з 2 сторінок;
– список літератури – включає в себе список використаної літератури. Складає 1 сторінку;
– додатки – містить технічне завдання і робочий проект. Складає 55 сторінок.
Кількість таблиць: 9 штук.
Кількість рисунків: 19 штук.
Ключові слова: фреймворк, React, XSS атака, Cypress.

ЗМІСТ

Вступ……8
1. Збір вимог до програмного забезпечення……9
1.1. Огляд програмних аналогів……...9
1.2. Огляд літератури……9
1.2.1. JavaScript фреймфорки……..9
1.2.2. Базовий захист JavaScript фреймфорків……12
2. Проєктування……16
2.1. Зовнішнє проєктування……16
2.1.1. Вхідні дані……16
2.1.2. Вихідні дані……..16
2.1.3. Функціональне призначення……..16
2.1.4. Експлуатаційне призначення……..16
2.1.5 Функціональні вимоги……16
2.1.6. Специфікація функціональних вимог……18
2.2. Внутрішнє проєктування……22
2.2.1. Проєктування архітектури системи……...22
2.2.1.1. Моделювання словника системи……..…22
2.2.1.2. Моделювання примітивних типів, простих залежностей, наслідування та структурних зв’язків……....23
2.2.2. Проєктування інтерфейсу користувача……24
2.2.3. Проєктування бази даних……...26
3. Тестування та налагодження……30
3.1. Тестування методом «білої скриньки»……...30
3.2. Тестування методом «чорної скриньки»……37
3.2.1. Специфікація функції……..37
3.2.2. Розробка тестів та демонстрація їх відповідності методам тестування……37

3.2.2.1. Класи еквівалентності……..37
3.2.2.2. Тестування методом граничних умов…….40
3.2.2.3. Тестування методом припущення про помилку……40
3.2.2.4. Тести за методом еквівалентних розбиттів……40
3.2.2.5. Тести за методом припущення про помилку…….42
3.2.2.6. Unit-тестування…….44
3.3 Налагодження програми……..46
Висновки……....47
Література……..…….49
Додатки……...50

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Фреймворк — це каркас веб-застосунку. Він зобов'язує програміста будувати архітектуру програми відповідно до певної логіки. Зазвичай середовище надає можливість для подій, сховищ і з'єднань даних.[2]

CSS framework — це набір або декілька файлів з готовим кодом, який підключається до сторінки в головному розділі, після чого може використовуватися структура цієї структури.

ReactJS — це бібліотека, а не фреймворк і доводить свою ефективність в динамічних додатках з високим трафіком (пропускною здатністю) як приклад Facebook і Instagram. Це найшвидше зростаюча платформа JS, сьогодні існує близько 1000 авторів Github.[2]

XSS — це вразливість веб-безпеки, яка дозволяє зловмиснику скомпрометувати взаємодію користувачів із уразливою програмою.[3] Це дозволяє зловмиснику обійти ту саму політику походження, яка призначена для відокремлення різних веб-сайтів один від одного. Уразливості міжсайтових сценаріїв зазвичай дозволяють зловмиснику маскуватися під користувача-жертву, виконувати будь-які дії, які може виконати користувач, і отримати доступ до будь-яких даних користувача. Якщо користувач-жертва має привілейований доступ до програми, то зловмисник може отримати повний контроль над усіма функціями та даними програми.

ВСТУП

Ми живемо в той час, коли безперервний розвиток інтернету та інформаційних технологій в цілому зробили так, що навіть маленька компанія, не кажучи вже про великі корпорації, державні установи, має свій веб застосунок.

Фреймворк — це каркас веб-застосунку. Він зобов'язує програміста будувати архітектуру програми відповідно до певної логіки. Зазвичай середовище надає можливість для подій, сховищ і з'єднань даних. Як і машина, рама забезпечує готовий каркас, кузов і двигун.[4] Можна додавати, знімати або змінювати певні деталі за умови, що автомобіль залишається в робочому стані. Каркаси знаходяться на більш високому рівні абстракції в порівнянні з бібліотекою і дозволять не потрібні зусилля з розробки приблизно 80% додатків. Але слід пам'ятати:

• Останні 20% можуть викликати значні труднощі через обмеження, що накладаються рамками.

• Оновлення структури може бути зумовлене складністю, а іноді й неможливою.

• Основний код і поняття рідко відповідають вимогам творців у їх первісному вигляді. Завжди треба знаходити "кращій" спосіб зробити щось.

Більшість фреймворків пишуться за допомогою JavaScript. Сьогодні більше 90 відсотків веб-застосунків створенні на базі саме JavaScript фреймворків.

Незважаючи на гарний базовий захист у всіх сучасних фреймворках, все одно є лазівки, за допомогою яких хакери отримують доступ до важливих даних та інформації про користувачів, банківські рахунки, тощо. Саме через це ця робота є актуальною на сьогодні, бо кожен розробник під час написання вебзастосунку за допомогою JavaScript фреймворку може покращити базовий 9 захист, який йому надає фреймворк. Результатом цієї роботи і є такий метод, та його порівняння з базовими методами захисту інших фреймворків.

Мета роботи - оцінка захисту JavaScript фреймворків, які використовуються під час розробки веб-застосунків, побудова свого методу захисту для одного із фреймворків, та порівняльний аналіз написаного методу з наявними.

Завданням роботи є розробка методу захисту одного із JavaScript фреймворків та його порівняльний аналіз із базовими методами. Об’єктом дослідження є сучасні JavaScript фреймворки.

Предметом дослідження є методи захисту сучасних JavaScript фреймворків.

1. ЗБІР ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Огляд програмних аналогів

Перед розробкою програмного продукту було проведено дослідження ринку технологій та пошук програмних аналогів. У ході дослідження програмних аналогів не було знайдено, що робить проект унікальним.

1.2. Огляд літератури

1.2.1. JavaScript фреймфорки

Інструменти спрощують процес програмування. Наприклад, використовуйте препроцесор Sass замість чистого CSS, оскільки він надає можливість використання циклів, функцій, локальних змінних і багато іншого. Браузери не розуміють синтаксис Sass / SCSS, тому код перекладається на CSS.

Структура CSS-Bootstrap Bootstrap є найпопулярнішим фреймворком з готовими стилями і скриптами, для яких потрібно перезаписати необхідні класи і атрибути елементів HTML. Завдяки цій структурі легко створювати мобільні проекти за допомогою

Bootstrap, що дозволяє легко налаштовувати будь-яку сторінку та відображати її на будь-якому пристрої.

CSS framework - це набір або декілька файлів з готовим кодом, який підключається до сторінки в головному розділі, після чого може використовуватися структура цієї структури.[5]

Bootstrap значно спрощує компонування. Перш за все, рамки використовують різні браузери і варіанти адаптації, які є основними питаннями, яким повинен звернутися програміст. Bootstrap робить його дуже легко реалізувати.

По-друге, структура ідеально підходить для колективної роботи. Розміщення початкового завантаження з правильними навичками та розумінням займає в 3-5 разів швидше, а однорідність коду дозволяє кожному внести зміни. За відсутності рамки кожен розробник може мати свій власний стиль, а інша людина буде витрачати час на вивчення його коду.

Перевагою цієї структури є наявність великої громади і наявність хорошої документації. Через цю перевагу Bootstrap з'являвся шаблони, де вже витончений дизайн всіх основних елементів. На основі цих шаблонів можна внести лише невеликі зміни на веб-сайті.

Помилки завантаження

1. Код у бібліотеці написаний для кожного випадку, у випадку дизайну більше, ніж потрібно.

2. Дизайн шаблону. Цю проблему можна легко вирішити, оскільки вона буде мати місце лише тоді, коли ви використовуєте тільки готові компоненти скелета і нічого не робите.

Будівельні компоненти.

Bootstrap є всеосяжною платформою, має багато встановлених компонентів, все, що може знадобитися для створення типових сайтів, таких як розкривні меню, кнопки, закладки, індикатори стану, трикутник, букви, назви тощо. додається до документів HTML.

Рамка містить знаковий шрифт, що полегшує роботу з іконками, тому вам не потрібно завантажувати ці піктограми у вікні перегляду зображень. За замовчуванням Bootstrap має близько 200 значків.

Початок роботи з кадрами.

Ви, звичайно, повинні почати відвідування офіційного сайту getbootstrap.com. Сторінка "Перші кроки" або "Перші кроки" дозволяє завантажувати кадр одним із способів.

Крім того, ви можете приєднати необхідні файли через CDN, необхідно зареєструватися на головній сторінці і зв'язати ці файли з архівом cdn.

Щоб налаштувати структуру, використовуйте лише деякі з його компонентів, перейдіть до http://getbootstrap.com/customize. Він повинен буде вибрати компоненти, які він буде містити. За допомогою такого підходу ви можете позбутися непотрібного коду і зберегти тільки те, що вам потрібно. Налаштування дуже довге, і ви можете встановити багато параметрів, наприклад кольори за замовчуванням для різних компонентів тощо.

JavaScript framework.

JavaScript фреймворки – це інструменти для побудови динамічних веб/мобільних/настільних додатків на Javascript. Розробники використовують js фреймворки там, де неможливо/складно/довго виконувати завдання звичайними засобами. У переважній більшості випадків, фреймворки використовуються для написання, так званих, Single Page Applications.[4]

Односторінковий веб-застосунок – це веб-застосунок чи веб-сайт, який вміщується на одній сторінці. В односторінковому веб-застосунку весь необхідний код - HTML, JavaScript, та CSS - завантажується разом із сторінкою або динамічно довантажується за потребою, зазвичай, у відповідь на дії користувача. Сторінка не оновлюється і не переправляє користувача на іншу сторінку. Взаємодія з з астосунком часто здійснюється через динамічний зв'язок з веб-сервером.

За допомогою JavaScript можуть бути розроблені як повні веб-сайти, так і функціональні модулі (онлайн-інструменти). Зазвичай, повні кадри більше підходять для першого завдання, а для інших рекомендується використовувати більш світлі кадри або бібліотеки.

Рамки забезпечують прозору структуру програми та реалізуються за допомогою програми Найчастіше використовуються MVC (Model-ViewController), Model-View-Presenter (MVP) і Model-View-ViewModel (MVVM).

Переваги побудови програми в рамках JS:

• ефективність – проекти, розроблені протягом багатьох місяців з сотнями рядків коду, тепер можуть бути реалізовані набагато швидше з добре підготовленими готовими шаблонами та функціями. Код стає значно меншим і

чистішим, що позитивно впливає на швидкість розвитку, а також на підтримку і усунення помилок програмного коду.

• безпека - найкращі структури JavaScript мають фірмову систему безпеки і підтримуються великою спільнотою, члени та користувачі якої виступають лише тестерами;

• витрати - більшість кадрів відкриті і безкоштовні. Оскільки вони допомагають програмістам швидше розвивати власні рішення, остаточна ціна веб-програми буде нижчою. Ви можете легко реалізувати SPA (Single Page Application). Наявність структури означає модульність програми, що полегшує роботу багатьох програмістів одночасно;

• можливість швидко створити мобільний або комп'ютерний мультиплатформовий додаток з веб-версії за допомогою PhoneGap або Apache Cordova;

Існує багато програм на js-фреймворках, і цей сегмент швидко зростає.

Angular.

Angular.js часто називають MVW (Model-View-Whatever) фреймворком, в основні переваги для нових і середніх компаній включають: швидке кодування, швидке тестування всіх частин програми та двосторонні канали передачі даних (зміни, які миттєво відображаються у користувачеві) Інтерфейси), В даний час найпопулярнішою односторонньої прикладною рамкою для односторінкових додатків (додатків з одним SPA-сайтом) і є найбільша спільнота програмістів.

Angular2 має великий список функцій, які дозволяють розробляти все, починаючи від Інтернету, до настільних комп'ютерів і мобільних додатків. Рамка побудована на Microsoft TypeScript[6]. Angular2 включає в себе архітектуру на основі компонентів, вдосконалену DI (ін'єкцію залежностей), ефективні служби журналів, взаємодії компонентів і багато іншого.

Обидві версії Angular є найкращим вибором для високоякісних корпоративних додатків або середовищ програмування, перш ніж читати код.

ReactJS.

Це бібліотека, а не фреймворк і доводить свою ефективність в динамічних додатках з високим трафіком (пропускною здатністю) як приклад Facebook і Instagram. Це найшвидше зростаюча платформа JS, сьогодні існує близько 1000 авторів Github.

У MVC (Model-View-Controller) React.js працює як "V" і може бути легко інтегрована в будь-яку архітектуру. Використання віртуального дерева DOM забезпечує більш високу продуктивність у порівнянні з кутом 1.x. Респонденти можуть бути створені та повторно використані в інших програмах або навіть відправлені для загального користування.[7]

Відповідь важко вивчити, але це полегшує та спрощує створення додатків. Ідеально підходить для складних, високопродуктивних програмних рішень.

Vue.js.

Vue 2.0 був введений в 2016 році, ввів найкраще в Ember, React і Angular. Vue.js пропонує двонаправлене підключення до даних (як у AngularJS), візуалізацію на стороні сервера (як у Angular2 та ReactJS), Vue-cli (інструмент

лісів) та додаткову підтримку JSX. Автори стверджують, що Vue2 є одним з найшвидших вільних мислителів.

Vue.js буде найкращим вибором для швидкої розробки багатоплатформних додатків. Це може бути міцною основою для ефективних односторінкових додатків (SPA) та реальних рішень, де продуктивність важливіша, ніж хороша організація коду або структура програми.[7]

Ember.Js.

У 2015 році Ember була визнана найкращою структурою JS, залишивши React та AngularJS. В даний час у нього є велика спільнота розробників, регулярні оновлення та широко використовувані кращі практики в Java Script.

Ember має двостороннє підключення до даних, як у AngularJS, все ще синхронізуючи вигляд і модель. Використання Fastboot.js забезпечує швидку візуалізацію (перерахунок) дерева DOM на стороні сервера, покращуючи представлення складних інтерфейсів користувача. Emberjs широко використовується в складних багатофункціональних вебдодатках і веб-сайтах. Провідні користувачі включають Chipotle, Blue Apron, Nordstrom, Kickstarter, LinkedIn, Netflix та багато іншого. Більш того, найпростіше навчитися і має багато навчальних посібників та онлайн-каталогів.

Meteor.js.

Meteor є одним з найпопулярніших JavaScript фреймів, який має багато функцій для створення бекенда і переднього коду відтворення, управління базами даних і бізнес-логіки. З моменту його випуску в 2012 році його екосистема різко зросла. Ця повнофункціональна платформа дозволяє швидко створювати веб- та мобільні додатки. З точки зору продуктивності, всі зміни в базі даних негайно передаються користувальницькому інтерфейсу і назад без значних втрат часу через відмінності в мові або у відповідь на серверний час.

Meteor.js охоплює всі етапи циклу розробки програмного забезпечення і використовується для створення веб-додатків реального часу, таких як Mazda, IKEA, Honeywell і багато інших.

1.2.2. Базовий захист JavaScript фреймворків.

Інтернет-додатки постійно піддаються різним загрозам. Більше того, найсерйознішою загрозою є те, що вони є хакерами та вірусами. Перший може отримати доступ до конфіденційної інформації, розміщення серверів, хакерських сайтів і заміни їх контенту, а також порушити трафік сервера через розподілені атаки (DDoS атаки). Однак віруси, які заражають веб-сервери, перетворять їх на інфекції розсади. Крім того, вони значно сповільнюють його роботу, а також займають інтернет-канал. На перший погляд, ці загрози, здається, в основному різні. Насправді це не зовсім так. Виявляється, багато вірусів, особливо інтернетхробаків, використовуються для поширення прогалин у програмному забезпеченні. Хакери також віддають перевагу атакам, націленим на відомі отвори в програмному забезпеченні. І в цьому немає нічого особливого.

Використовуючи слабкі місця безпеки, обидва вони отримують відносно легкий доступ до віддаленого комп'ютера, навіть якщо він добре захищений.

Практично у кожній програмі є прогалини. І оскільки його вихідний код більший, то більше він може знайти "отвори". Розрив дуже легко пояснити. Людина не машина, він може помилятися. Існує навіть спеціальне правило програмування, яке визначає, скільки помилок експерт може допустити при написанні декількох рядків коду. Крім того, пам'ятайте, що велике програмне забезпечення написано не однією людиною, а цілою групою. Часто виникають помилки при створенні модулів, створених різними програмістами. Крім того, наявність вразливості не завжди залежить від якості написаного програмного забезпечення.

З точки зору уразливості в інтернет-додатках, переважна більшість людей відразу згадує "дірки" у своєму програмному забезпеченні. Це стосується самих серверних програм, таких як Apache, Microsoft Internet Information Server тощо. У цьому немає нічого дивного. Проте, програмне забезпечення досить обширне і складне, тому "дірка" в ній завжди доступна. Крім того, ми не можемо забувати, що сучасний веб-сервер не можна уявити без багатьох інших функцій, таких як мови програмування, такі як Perl, PHP тощо. Все це можливо за допомогою встановлення додаткового програмного забезпечення в веб-додатку. Він також може містити отвори безпеки.

В даний час сайти інформаційної безпеки постійно повідомляють про нові помилки в програмному забезпеченні веб-додатків. Цей процес включає захист даних і хакерів. А інша, яка виявляє нову "дірку", може заспокоїти її і спробувати використати її для своїх власних цілей. Але вірно протилежне. Хакер намагається говорити про нові вразливості для всіх, включаючи розробників програмного забезпечення.

Головною особливістю виробничих прогалин є їх залучення до конкретних версій програмного забезпечення. Справа в тому, що "дірки" часто не зустрічаються в ряді веб-додатків, а лише в деяких їх виданнях. Однак слід зазначити, що чим популярнішим є програмне забезпечення, тим більше шансів знайти нові уразливості.

І це не залежить від якості написаного програмного забезпечення. Це просто розглядається рядом фахівців, тому ймовірність виявлення «дірок» є відносно високою.

З метою захисту від типу програмного забезпечення, що оцінюється, уразливості можуть бути лише одним способом - своєчасне встановлення всіх виробників розробило оновлення та виправлення. Справа в тому, що програмісти регулярно дізнаються про офіційні оновлення своїх продуктів. Коли виявлено критичне отвір безпеки, патч швидко звільняється. Якщо прогалини, виявлені знову, є більш теоретичними, ніж реальна загроза, масові оновлення звільняються, коли вони накопичуються.

Проблеми з безпекою можуть виникати через неправильні налаштування програмного забезпечення веб-застосунку.

Можливо, не секрет, що безпека будь-якої речі залежить від того, як ви її використовуєте. Те ж саме можна сказати і про веб-додаток. Багато що залежить від конфігурації програмного забезпечення. Загалом, переважна більшість вебдодатків мають досить великий набір параметрів, які охоплюють практично всі аспекти їхнього бізнесу. Безпека, отже, значною мірою залежить від адміністраторів, що беруть участь у їхніх послугах. Але ми не можемо забувати, що адміністратори - це люди. А це означає, що вони можуть помилятися через свою неуважність, відсутність кваліфікації або з будь-якої іншої причини. І ці помилки можуть відкрити шлях до хакерів або вірусних веб-серверів.

Правильні налаштування не допоможуть вам встановити виправлення. Насправді, при оновленні програмного забезпечення його конфігурація не змінюється. Це означає, що існує ймовірність помилки в системі захисту після встановлення патча. Головною загрозою для типу "дірок" вважається тому складність їх знаходження. Єдиний спосіб надійно захистити від таких розривів - використання спеціальних сканерів безпеки. Ці програми, використовуючи спеціальні методи, досліджують захист веб-серверів і знаходять всі потенційно небезпечні сайти.

Веб-скрипти також можуть містити дірки в захисті.

Сучасні інтернет-програми та супутнє програмне забезпечення часто створюють специфічну основу для реалізації програм, написаних користувачем. Звичайно, це сценарії, які працюють на більшості сучасних веб-сайтів. Справа в тому, що більшість мов веб-програмування є серверними. Це означає, що сценарії, написані на них, запускаються безпосередньо на сервері, і тільки результати їх роботи (в даному випадку відвідувача сайту) надсилаються на комп'ютер користувача.

Це дуже серйозна небезпека. Справа в тому, що сценарії веб-сайту не завжди розробляються дійсно хорошими фахівцями. Багато інтернет-проектів використовують безкоштовне або самостійне програмне забезпечення. Звичайно, він може також містити пробіли. Деякі з них можуть бути дуже серйозними, дозволяючи зловмисникам отримати несанкціонований доступ до самого сервера. Крім того, слід враховувати, що деякі сценарії виконуються з підвищеними привілеями. Тому прогалини в них можуть бути гарним інструментом для хакерів.

Ви можете знайти "скрипти" у скриптах, використовуючи сканери безпеки. Тому кожен власник веб-сервера дійсно піклується про безпеку свого сайту, він повинен регулярно перевіряти його.[8] І слово "звичайний" у попередньому реченні не було випадковим. Справа в тому, що хакери постійно придумують нові способи атаки на віддалену атаку. Крім того, в оригінальному програмному забезпеченні постійно виявляються нові "діри", які, у поєднанні зі скриптами, які раніше вважалися безпечними, становлять реальну загрозу.

Покращення інформаційної безпеки при використанні новітніх інформаційних технологій є однією з стратегічних цілей розвитку інформаційного суспільства в Україні.

Суть інформаційної безпеки полягає в захисті інформаційного середовища, особистості, суспільства і держави від навмисних і ненавмисних загроз і впливів. Тому явище спаму можна вважати одним з джерел його загроз.

Зокрема, загрози для технологій спаму інформаційної безпеки полягають у наступному: складність інформаційних систем і ресурсів через непотрібний пошук інформації, витрачання багато часу на аналіз спаму і психічний дискомфорт, пов'язаний з подальшим видаленням багатьох повідомлень, ризик видалення необхідного спаму, що може призвести до різні види неприємних наслідків, висока ймовірність реалізації різних підроблених програм, які можуть бути жертвами як приватного, так і юридичного CSSR, одержувач оплачує спам або роботу провайдера, який отримує небажані листи з поштового сервера.

Небажана пошта існує тому, що існуючі умови для її існування є законними, зокрема відсутність обов'язкового закону, що забороняє його розповсюдження та заборону економічної діяльності. Включення спаму як суб'єкта обміну інформацією також дає можливість розрізняти його суб'єктів: клієнта, творців (дистриб'юторів) і споживачів.

Правовими (правовими) методами боротьби зі спамом можуть бути: прийняття положення про заборону спаму, створення спеціального підрозділу для виявлення та переслідування спамерів на основі служби захисту інформації, що надає постачальникам поштових послуг певні обов'язки та дозволи на фільтрування пошти.

Існують подібні системи:

• CleanTalk: захист від завантаження та спаму в коментарях, анти-спам-онлайн форум без captcha.

• Анти-спам ACP: інструмент пошуку IP, список спаму, перевірка профілю, підозрілі виправлення користувачів.

Висновки: стандартом сьогодні є односторінкові веб-застосунки. Вони створюються за допомогою фреймворків. Захищеність веб-застосунків – це відкрите питання, для дослідження якого потрібно більше детально розглянути основні вразливості та обрати декілька з них для аналізу використовуваних ними лазівок. Інтернет-додатки постійно піддаються різним загрозам. Більше того, найсерйознішою загрозою є те, що вони є хакерами та вірусами. Перший може отримати доступ до конфіденційної інформації, розміщення серверів, хакерських сайтів і заміни їх контенту, а також порушити трафік сервера через розподілені атаки (DDoS атаки). Однак віруси, які заражають веб-сервери, перетворять їх на інфекції розсади. Крім того, вони значно сповільнюють його роботу, а також займають інтернет-канал. На перший погляд, ці загрози, здається, в основному різні. Насправді це не зовсім так. Виявляється, багато вірусів, особливо інтернетхробаків, використовуються для поширення прогалин у програмному забезпеченні.

2. ЗОВНІШНЄ І ВНУТРІШНЄ ПРОЕКТУВАННЯ

2.1. Зовнішнє проектування

2.1.1. Вхідні дані:

Вхідними даними для розробки програмного засобу аналізу методів захисту React фреймворку є: веб-додаток “Company”. Веб-додаток “Company” має наступні вхідні дані:

- робітник (вакансія, ФІП, фото);
- проект (назва, посилання, фото);
- вакансія (назва, вимоги);
- форма (коментар, вид проекту, бюджет проекту);

2.1.2.Вихідні дані:

Вихідними даними веб-додатку “Company” для звичайного користувача є:

- список робітників;
- список проектів;
- список вакансій;

Вихідними даними веб-додатку “Company” для адміністратора є:

- список робітників;
- список проектів;
- список вакансій;
- список форм;

Вихідними даними програмного засобу аналізу методів захисту React фреймворку є: результат методів захисту React фреймворку.

2.1.3.Функціональне призначення

Функціональне призначення – програмний продукт, веб-додаток, використовує браузер для доступу до інтернету та синхронізації даних на всіх пристроях. Наглядно демонструє захист JavaScript фреймворків від популярних видів атак.

Існує 11 видів операцій:

- “add employee”;
- “edit employee”;
- “delete employee”;
- “add project”;
- “edit project”;
- “delete project”;
- “add vacancy”;
- “edit vacancy”;
- “delete vacancy”;

- “submit form”;

Існує два режими, режим адміністратора, який має права на редагування сайту, та режим глядача, який має право на перегляд інформації. На основі веб-додатку були створені авто тести для аналізу методів захисту React. Написані авто тести перевіряють веб-додаток на вразливість до введення стороннього коду, від спаму повідомлень робота та від спаму повідомлень людини.

2.1.4.Експлуатаційне призначення

За допомогою програмного продукту виконується тестування та аналіз методів захисту функцій сайту. Що дає змогу більш детально дослідити методи захисту JavaScript фреймворків, як вбудовані у фреймворк, так і самописні методи захисту.

2.1.5.Функціональні вимоги

Програма представляє собою веб сайт компанії, на якому можно міняти наповнення, через панель адміністратора.

Щоб зайти до панелі адміністратора, треба авторизуватися ввести свій логін та пароль.

Пісня входу адміністратор потрапляє в кабінет адміністратора, в якому може змінити або відстежувати вхідні дані сайту.

Кнопка “Back” повертає адміністратора на головну сторінку сайту де він можу спостерігати свої зміни.

У вкладці “Projects” адміністратор може змінювати наповнення секції “Projects” на сайті.

У вкладці “Team” адміністратор може змінювати наповнення секції “Team” на сайті.

У вкладці “Vacancies” адміністратор може змінювати наповнення секції “Vacancies” на сайті.

У вкладці “Requested_project” адміністратор може аналізувати дані, які зберігаються у формі “Create your project”

У вкладці “Requested_vacancy” адміністратор може аналізувати дані, які зберігаються у формі “Career”

Створення проекту.

Кожен “Project” (проект) створюється за допомогою конструктора, у якому заповнюються поля «Title» - назва проекту, «Link» - веб посилання на сторінку проекту, «Description» - опис проекта, «Image» - головне фото проекту. Кожен створений проект можна редагувати або видалити. Також всі поля валідуються для забезпечення захисту від стороннього коду.

Кожен “Team” (робітник) створюється за допомогою конструктора, у якому заповнюються поля «Name» - ім’я робітника, «Position» - позиція робітника у компанії, «Image» - фото робітника. Кожного створеного

робітника можна редагувати або видалити. Також всі поля валідуються для забезпечення захисту від стороннього коду.

Кожна “Vacancy” (вакансія) створюється за допомогою конструктора, у якому заповнюються поля «Title» - назва проекту, «Position» - позиція вакансії, «Location» - країна або місто компанії, «Description» - опис вакансії, перелік “Tasks” завдань, кожне завдання можно редагувати або видалити. Кожну створену вакансію можна редагувати або видалити. Також всі поля валідуються для забезпечення захисту від стороннього коду.

Тестування веб-додатку відбувається у двох режимах. В першому, автоматизований алгоритм (бот), перевіряє форми від введення стороннього коду, від спаму повідомлень робота та від спаму повідомлень людини.

Захист від введення стороннього коду здійснюється завдяки перевірки валідації полів на спроможність вводу спеціальних символів, які використовуються у написанні js скрипта. Авто тести заповнюють всі можливі поля тестовим “шкідливий” кодом
“” або “javascript:alert(Взламано!');” для посилань,
та перевіряє валідацію полів, та неспроможність відправити форму без вірних значень.

Захист від спаму повідомлень користувачем (людиною) здійснюється завдяки вимиканню кнопки “Send (відправити)” на 5000ms (5 секунд), після проходження 5 секунд кнопка стає знов активною. Також щоб запобігти спаму, було встановлено розширення “recaptcha v2” від “Google”, це дозволяє встановити поле, яке потрібно вмикати після кожної відправки. Це поле після кожних 3-4 відправки форми запитує спеціальні питання, наприклад: “Виберіть світлофори” або “Виберіть пішохідний перехід”, та відображає модальне віконце із 9 фотокартками, серед них треба вибрати фотокартки які відповідають даним вимогам.

Захист від спаму повідомлень автоматизованим алгоритмом (ботом) здійснюється завдяки вимиканню кнопки “Send (відправити)” на 5000ms (5 секунд), після проходження 5 секунд кнопка стає знов активною. Але бот автоматично може видаляти атрибут “disabled” у кнопки, тим самим автоматично її активовувати. Щоб запобігти можливості відправляти форму раніше ніж кожні 5 секунд було встановлено розширення “recaptcha v2” від “Google”, це дозволяє встановити поле, яке потрібно вмикати після кожної відправки. Це поле після кожних 3-4 відправки форми запитує спеціальні питання, наприклад: “Виберіть світлофори” або “Виберіть пішохідний перехід”, та відображає модальне віконце із 9 фотокартками, серед них треба вибрати фотокартки які відповідають даним вимогам. Так як ці фотокартки кожен раз різні, і запити на їх вибір теж рандомні, звичайний бот не спроможний вибрати правильні відповіді. Виходячи з цього не зможе відправити форму.

2.1.6.Специфікація функціональних вимог.

Специфікація функціональних вимог для користувача або адміністратора представлена у вигляді діаграми прецедентів (рис.1.1).

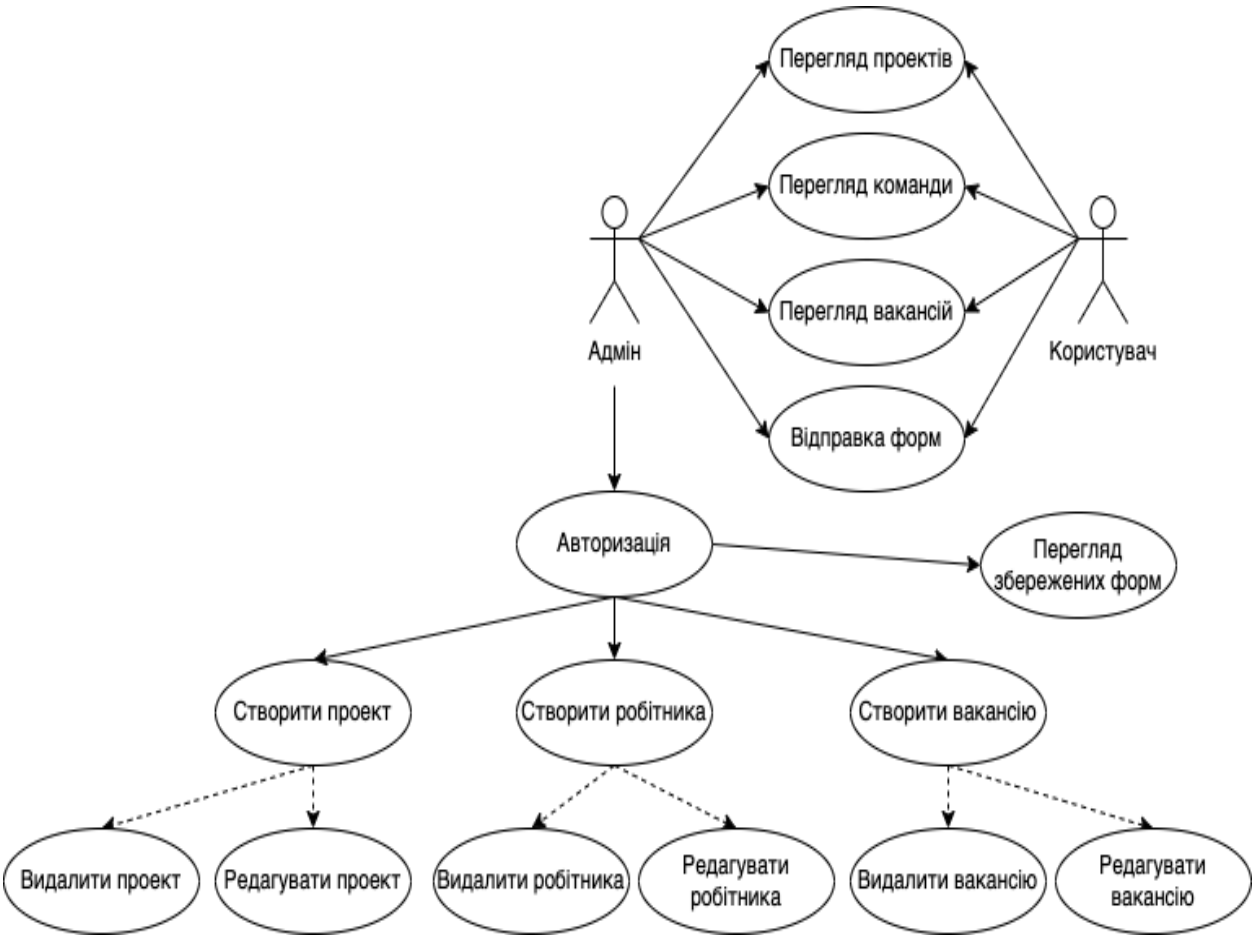

Рисунок 2.1 — Діаграма прецедентів для користувача або адміністратора.

Специфікація функціональних вимог для авто тесту для тестування форми замовлення проекту представлена у вигляді діаграми прецедентів (рис.2.2).

Специфікація функціональних вимог для авто тесту для тестування форми відгуку на вакансію представлена у вигляді діаграми прецедентів (рис.2.3).

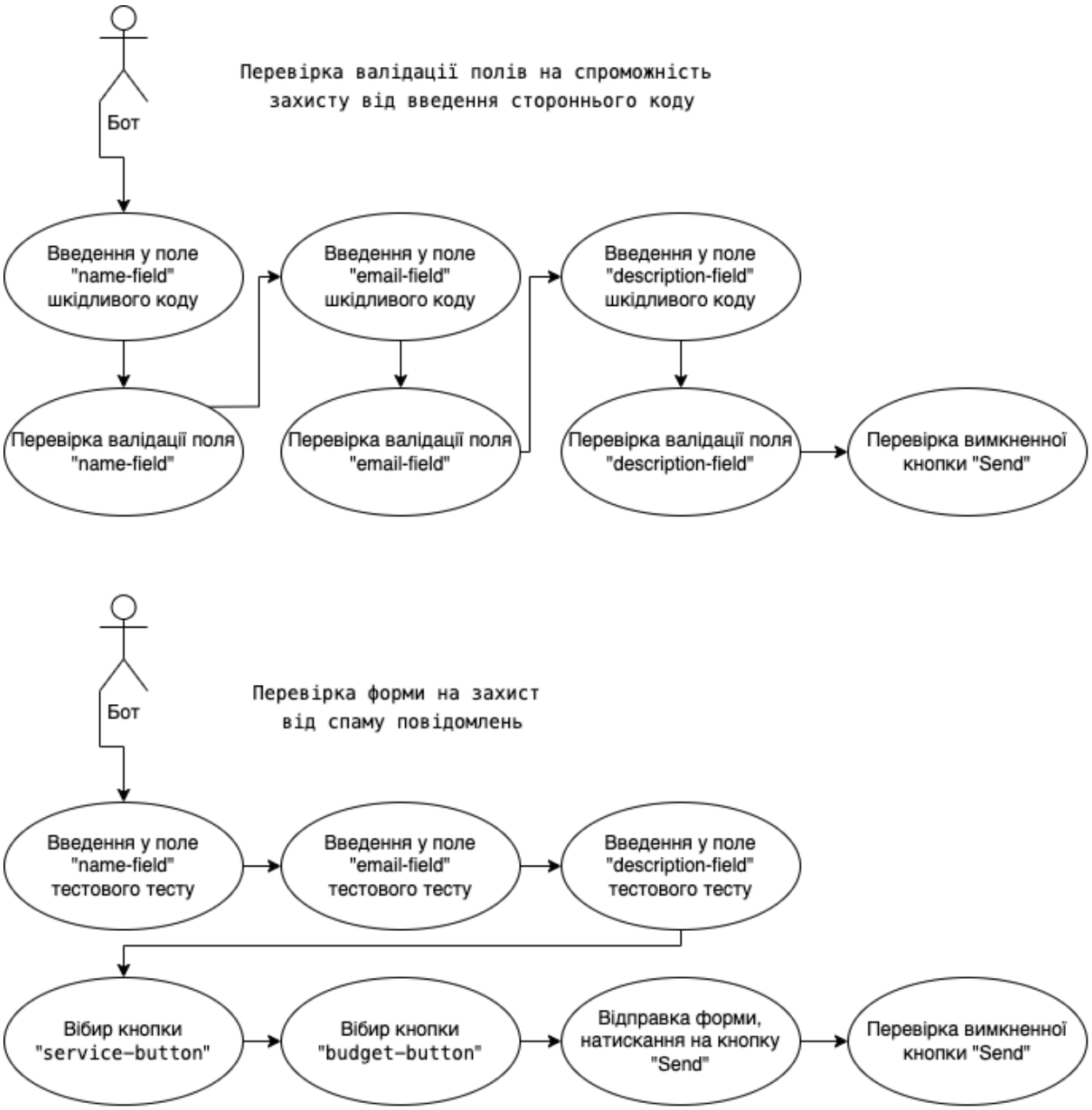

Рисунок 2.2 — Діаграма прецедентів для авто тесту.

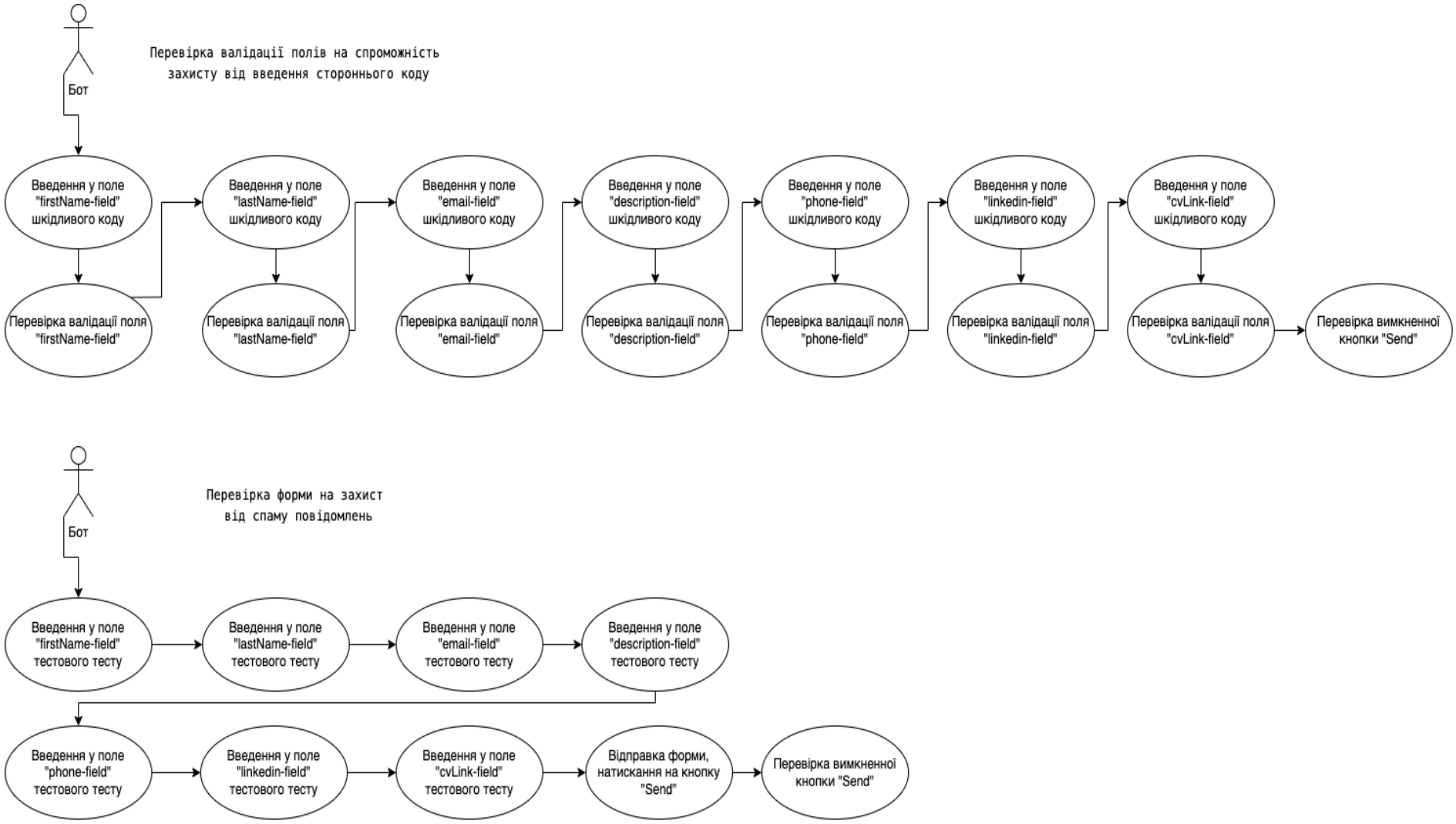

Рисунок 2.3 — Діаграма прецедентів для авто тесту для тестування форми відгуку на вакансію

2.2. Внутрішнє проектування

2.2.1. Проектування архітектури системи

2.2.1.1. Моделювання словника системи

Ідентифіковані сутності: проект, робітник, вакансія, вимога вакансії, замовлений проект, відгук на вакансію.

Ідентифіковані обов’язки:

Проект – відображення інформації проекта, із полями title, link, description, image.

Робітник – відображення інформації робітника, із полями name, position, image.

Вакансія – відображення інформації вакансії, із полями title, location, description, position, task.

Вимога вакансії – відображення інформації вимоги вакансії у вакансії, із полями text.

Замовлений проект – відображення інформації про замовлений проект, із полями budget, description, email, id, name, service.

Відгук на вакансію – відображення інформації про відгук на вакансію, із полями cvLink, description, email, firstName, id, lastName, linkedin, phone.

Атрибути та операції, необхідні для виконання обов'язків кожної сутності-класу наведено у табл. 2.1.

Таблиця 2.1. Атрибути та операції, необхідні для виконання обов'язків кожної сутності-класу.

Сутність	Атрибути	Методи
1	2	3
Проект	title link description image id	відобразити дані створити видалити редагувати
Робітник	name position image id	відобразити дані створити видалити редагувати

Закінчення табл.2.1.

1	2	3
Вакансія	title location description position task id	відобразити дані створити видалити редагувати
Вимога вакансії	text id	відобразити дані створити видалити редагувати
Замовлений проект	budget description email id name service	відобразити дані створити
Відгук на вакансію	cvLink description email firstName id lastName linkedin phone	відобразити дані створити

2.2.1.2. Моделювання примітивних типів, простих залежностей, наслідування та структурних зв’язків.

Визначення примітивних типів виконаємо на етапі побудови діаграми компонентів. Результат моделювання різних видів зв’язків подано у табл. 2.2.

Таблиця 2.2. Моделювання залежностей

Компонент, який зв'язується	Клас, з яким зв’язуються	Тип зв’язку
App	AppRouter	Асоціація
AppRouter	AdminPanel	Асоціація
	Vacancies	Залежність
	Team	Залежність
	Projects	Залежність
AdminPanel	AdminVacancyCard	Залежність
	AdminProjectCard	Залежність
	AdminTeamCard	Залежність

Відношення між компонентами (функціями), представлені у вигляді діаграми компонентів (рис.2.4).

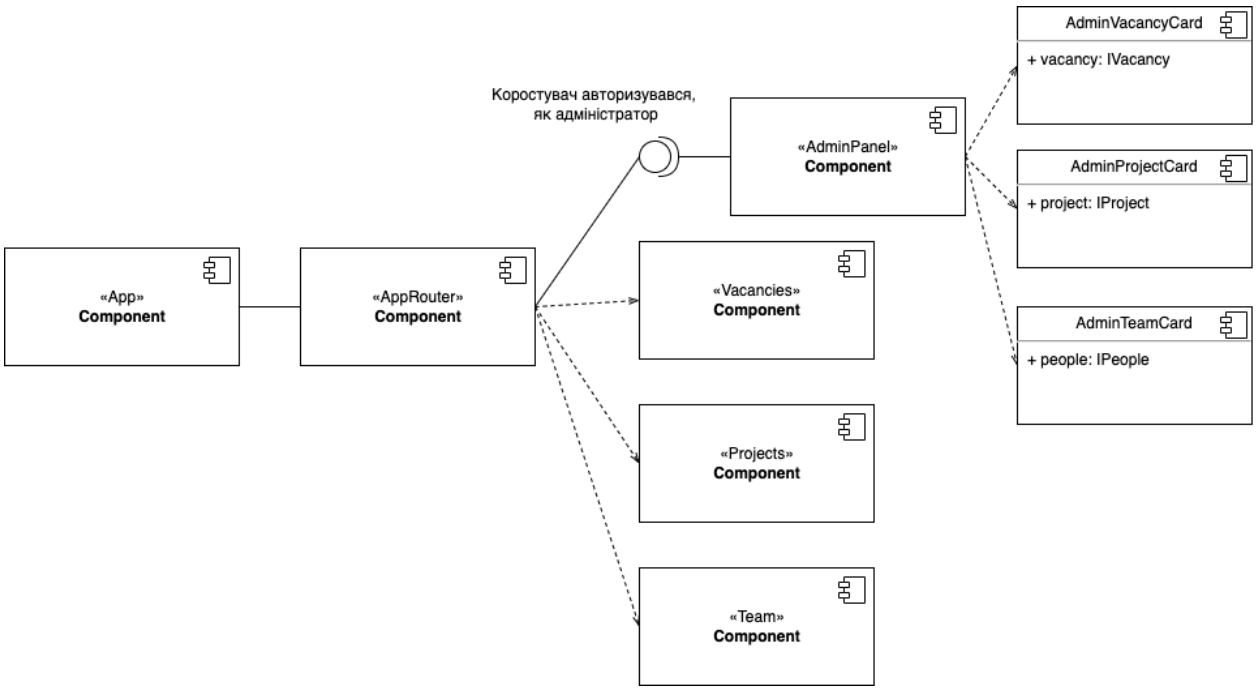

Рисунок 2.4 — Діаграма компонентів.

2.2.2. 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).

Під сценарієм будемо розуміти 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.

В даному випадку інтерфейс користувача буде відображатися на веб сайті, в режимі перегляду користувач може переглядати інформацію компанії, на головній сторінці, команду (Team), проекти (Projects), вільні вакансії (Vacancies). Або на окремих сторінках проєктів (Projects), вакансії (Vacancies).

Користувач такой може залишити свій відгук на вакансію або замовити проєкт у компанії.

Щоб потрапити в режим адміністратора, треба пройти авторизацію, через форму, в якій треба ввести свій логін та пароль.

У кабінеті адміністратора, користувач може змінити наповнення сайту, команду (Team), проекти (Projects), вільні вакансії (Vacancies) та переглянути інформацію про відгуки на вакансії або замовлені проекти.

Сценарій діалогу для системи «Company» для звичайного користувача наведено на діаграмі станів (рис.2.5).

Сценарій діалогу для системи «Company» для адміністратора наведено на діаграмі станів (рис.2.6).

2.2.3. Проєктування бази даних

У даному проєкті використовується платформа Firebase для доступу до бази даних.

Рисунок 2.5 — Діаграма станів для користувача.

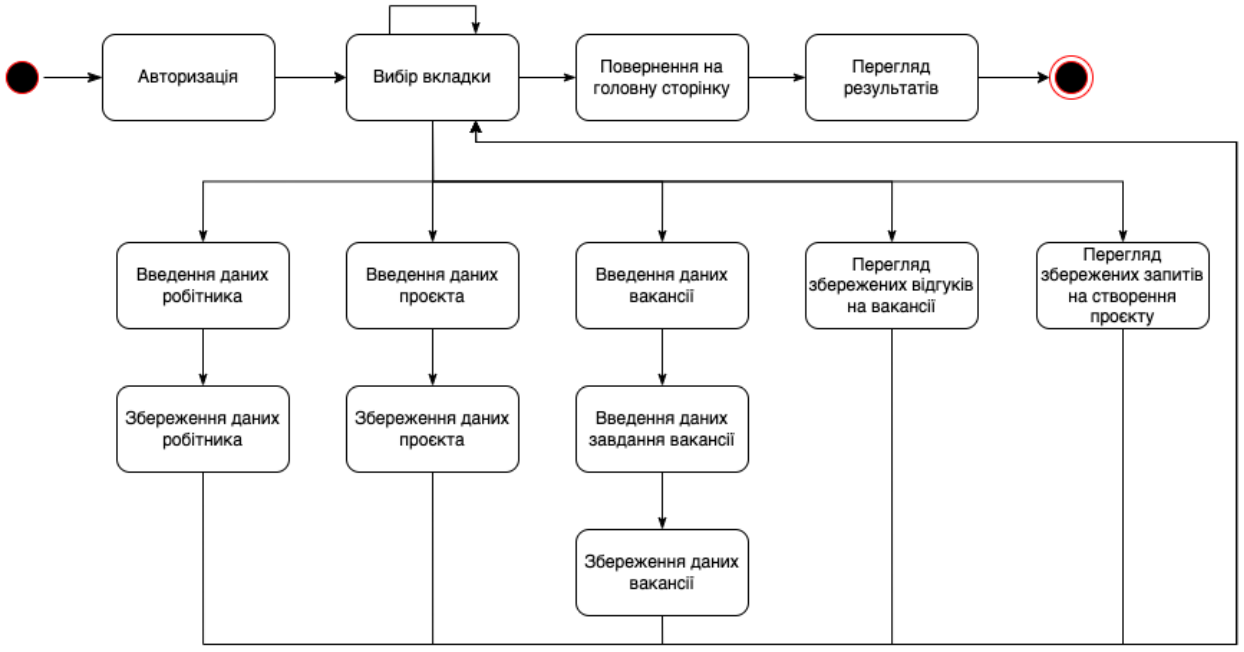

Рисунок 2.6 — Діаграма станів для адміністратора.

Firebase – це платформа розробки веб-додатків з великим функціоналом. Головне достоїнство платформи у цьому, що вона дозволяє розробнику не відволікатися створення бекенда, тобто прихованої від користувача програмної частини проекту, наприклад, серверного коду. І це спрощує і прискорює створення додатків, дає можливість повністю зосередитися саме на UX/UI, тобто, на інтерфейсі користувача.[9]

Firebase - це одне з BaaS-рішень (Backend as a Service), яке дає розробнику безліч можливостей.

Це і сервер, і база даних, і хостинг, аутентифікація в одній платформі. Так, Firebase Realtime Database надає розробникам API, який синхронізує дані програми між клієнтами та зберігає їх у хмарному сховищі. Програма підключається до бази даних через WebSocket, яка відповідає за синхронізацію даних протягом усього сеансу.

Також Firebase виступає як сховища файлів. Firebase Storage забезпечує надійне завантаження та вивантаження файлів для програми. Хмарне зберігання файлів відео, аудіо або іншого типу підтримується Google Cloud Storage. Вміст хмарного сховища надійно захищений системою безпеки.

Створювати систему аутентифікації щоразу з нуля досить затратно, причому ці витрати найчастіше не виправдані. Справиться з більшістю проблем дозволяє система аутентифікації Firebase Auth, в якій можлива автентифікація користувача програми пароля та електронної пошти. Підтримує Firebase Auth також відкритий протокол авторизації OAuth 2.0, використовуваний Google, Twitter, Facebook.

Система аутентифікації Firebase інтегрується безпосередньо до бази даних. Статичні файли програми розміщуються на хостингу Firebase. Підтримується хостинг файлів JavaScript, HTML, CSS та інших. Через Cloud Functions реалізовано динамічну підтримку Node.js. Передача файлів здійснюється через мережу доставки контенту за допомогою захищених протоколів SSL і HTTPS.

Використання платформи дозволяє скоротити час, необхідний для розробки, а також скоротити час завантаження програми та підвищити охоплення. Переваги платформи Firebase у додатку:

- висока швидкість роботи програми;
- надійна інфраструктура – відсутність збоїв у роботі;
- зручна статистика, що дозволяє отримувати дані про дії користувачів та підтримувати зворотний зв'язок;
- кросплатформеність – програма створена один раз і налаштована для роботи з усіма операційними системами;
- проста масштабованість – зростання кількості користувачів не потребує змін у серверному коді.

Використання платформи Firebase та її окремих функцій дозволяють зосередитись виключно на тому, як додаток буде виглядати та наскільки зручним буде його використання.

У Firebase було створено Realtime Database, яка працює через WebSocket. Створено сутності Projects, Vacancies, Team, Requested_projects, Requested_vacancies (рис.2.7).

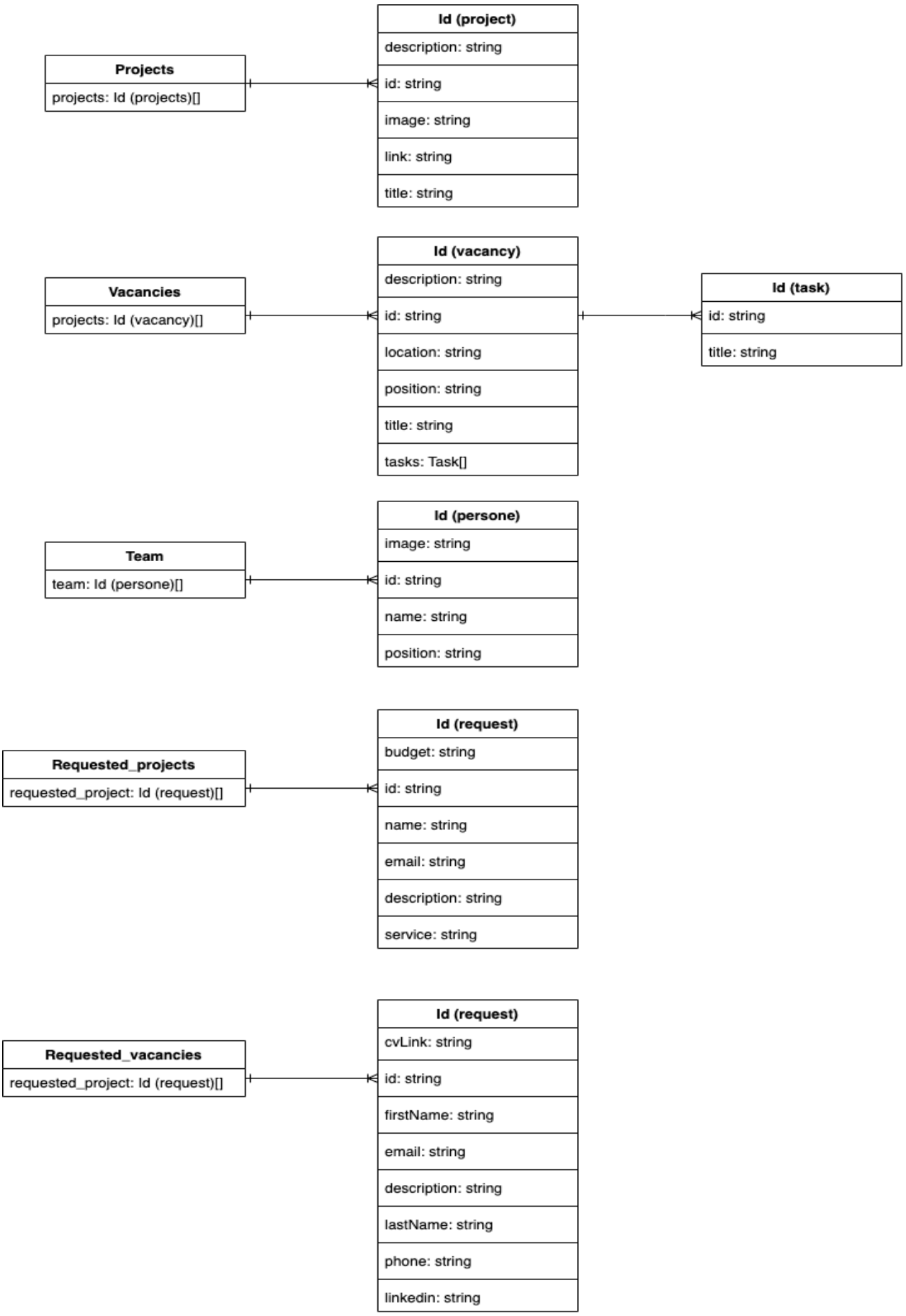

Рисунок 2.7 — Логічна схема бази даних.

Висновки: на стадії проектування системи були розроблені інтерфейс користувача, спроектовані основні алгоритми для веб-додатку та обрана база даних Firebase. На основі розроблених цьому розділі алгоритмів був написаний основний код програми, який має пройти тестування та, якщо буде потрібно, налагодження.

3. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

3.1. Тестування методом «білої скриньки»

Метод “saveItem”.

Метод змінює поточний елемент, такі як проєкт, вакансія або робітник, у базі даних.

Спочатку користувачу потрібно заповнити форму валідними даними, вибрати фотографію, якщо потрібно, та натиснути на кнопку “Save”. Якщо збереження елемента у базу даних пройшло успішно, буде виведене повідомлення "Item changed successfully!".

Вхідні дані:

Змінна type типу string приймає значення: ["vacancies", "team", "projects"] (неявний вигляд);

Змінна id типу string приймає значення: string (неявний вигляд);

Змінна data типу object приймає значення:

link: строка, яка підпадає під регулярний вираз [/((https?):\/\/)?(www.)?[a-z0-9]+(\.[a-z]{2,}){1,3}(#?\/?[a-zA-Z0-9#]+)*\/?(\?[a-zA-Z0-9-_]+=[a-zA-Z0-9-%]+&?)?$/], та її мінімальна к-сть символів - 2.

image: строка виду base64,

title: строка, яка підпадає під регулярний вираз [/^[a-яA-Яa-zA-Z\s]+$/], та її максимальна к-сть символів - 50, мінімальна к-сть символів - 2.

description: пусте значення або строка, яка підпадає під регулярний вираз[/^[a-яA-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.’“ ”\-%]+$/], та її максимальна к-сть символів - 512, мінімальна к-сть символів - 2.

id: строка (неявний вигляд),

Вихідні дані:

Повідомлення про успішне змінення елементу “Item changed successfully!”, або повідомлення про неправильні введені значення.

Метод “login”.

Метод авторизації користувача, після вводу валідного логіну та паролю користувач натискає на кнопку “Log In”, якщо логін або пароль не відповідають валідації, або невірний логін або пароль, виводиться відповідне повідомлення. У разі успішної авторизації, користувача перенаправляє у панель адміністратора, зберігає його дані (логин), та виводить повідомлення "Welcome!".

Вхідні дані:

Змінна email типу string яка підпадає під регулярний вираз [(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(2(5[

0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])] та її максимальна к-сть символів 50, мінімальна - 2.

Змінна password типу string яка підпадає під регулярний вираз [/^[a-zA-z]+$/] та її максимальна к-сть символів 512, мінімальна - 6.

Змінна callBack типу void (неявний вигляд);

Вихідні дані:

Повідомлення про неправильне введення логіна і пароля або повідомлення про успішну авторизацію та редирект на панель адміністратора.

Тестування метода “saveItem”

Приклад на введення валідних даних крім title (рис. 3.1).
Вхідні дані:
title = “bad title <:>”;
description = “test description”;
image = “base64”;
link = “https://www.mcdonalds.com/ua/uk-ua.html”;
Вихідні дані: повідомлення про невалідний title, “Is not in correct format”;

Приклад на введення валідних даних крім link (рис. 3.2.).
Вхідні дані:
title = “McDonalds”;
description = “test description”;
image = “base64”;
link = “bad link”;
Вихідні дані: повідомлення про невалідний link, “link must be a valid URL”;

Приклад на введення валідних даних крім description (рис. 3.3.).
Вхідні дані:
title = “McDonalds”;
description = “bad description <:>”;
image = “base64”;
link = “https://www.mcdonalds.com/ua/uk-ua.html”;
Вихідні дані: повідомлення про невалідний description, “<, >, : characters are prohibited”;

Приклад на введення валідних даних крім image (рис. 3.4.).
Вхідні дані:
title = “McDonalds”;
description = “test description”;
image = “null”;
link = “https://www.mcdonalds.com/ua/uk-ua.html”;

Вихідні дані: повідомлення про невалідний image, “Only jpeg, png, jpg extensions are allowed”;

Приклад на введення всіх валідних даних (рис. 3.5.).
Вхідні дані:
title = “McDonalds”;
description = “test description”;
image = “base64”;
link = “https://www.mcdonalds.com/ua/uk-ua.html”;
Вихідні дані: повідомлення про успішне збереження елемента “Item changed successfully!”;
Покриття умов для методу “saveItem” наведено у табл. 3.1.

Тестування метода “login”
Приклад на введення валідних даних крім email (рис. 3.6.).
Вхідні дані:
“email = testEmail”;
“password = adminadmin”;
Вихідні дані: повідомлення про невалідний email, “email must be a valid email”;

Приклад на введення валідних даних крім password (рис. 3.7.).
Вхідні дані:
“email = adminmail@mail.com”;
“password = 123456”;
Вихідні дані: повідомлення про невалідний password, “Password is invalid”;

Приклад на введення валідних даних крім email (рис. 3.8.).
Вхідні дані:
“email = пусте значення”;
“password = adminadmin”;
Вихідні дані: повідомлення про невалідний email, “Required”;

Приклад на введення валідних даних крім password (рис. 3.9.).
Вхідні дані:
“email = adminmail@mail.com”;
“password = пусте значення”;
Вихідні дані: повідомлення про невалідний password, “Required”;

Приклад на введення всіх валідних даних (рис. 3.10.).
Вхідні дані:
“email = adminmail@mail.com”;
“password = adminadmin”

Вихідні дані: повідомлення про успішну авторизацію та перенаправлення на панель адміністратора;
Покриття умов для методу “login” наведено у табл. 3.2.

Таблиця 3.1 — Покриття умов (метода saveItem).

	isValidDescription		isValidTitle		isValidImage		isValidLink		isValidForm	
	+	-	+	-	+	-	+	-	+	-
1	*			*	*		*			*
2	*		*		*			*		*
3		*	*		*		*			*
4	*		*			*	*			*
5	*		*		*		*		*	

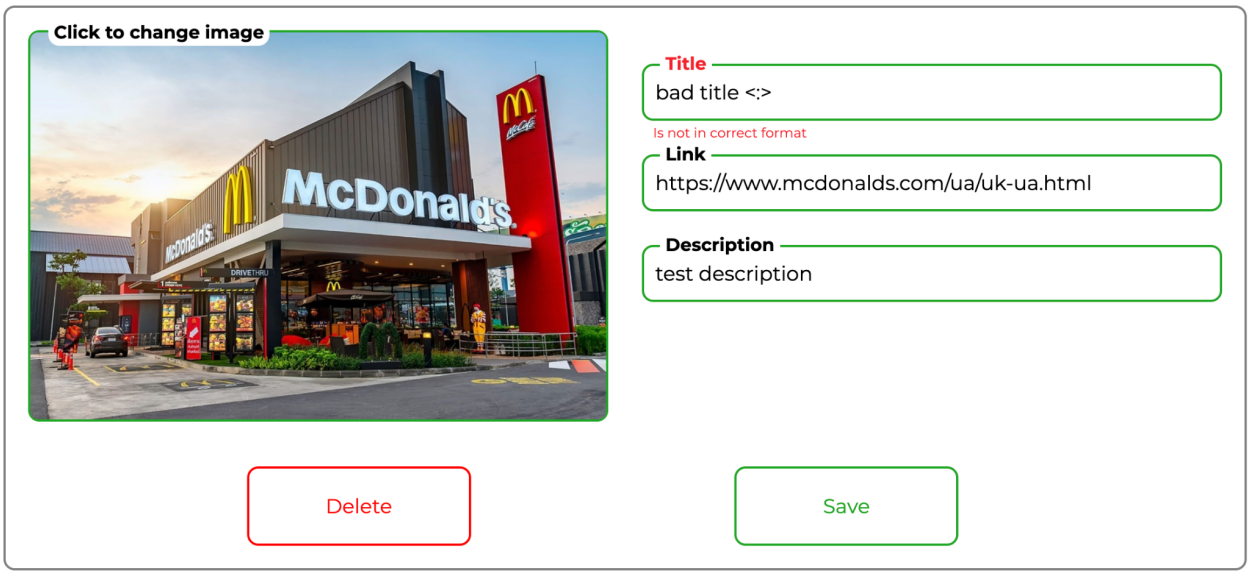

Рисунок 3.1 — Тестування методу “saveItem”.

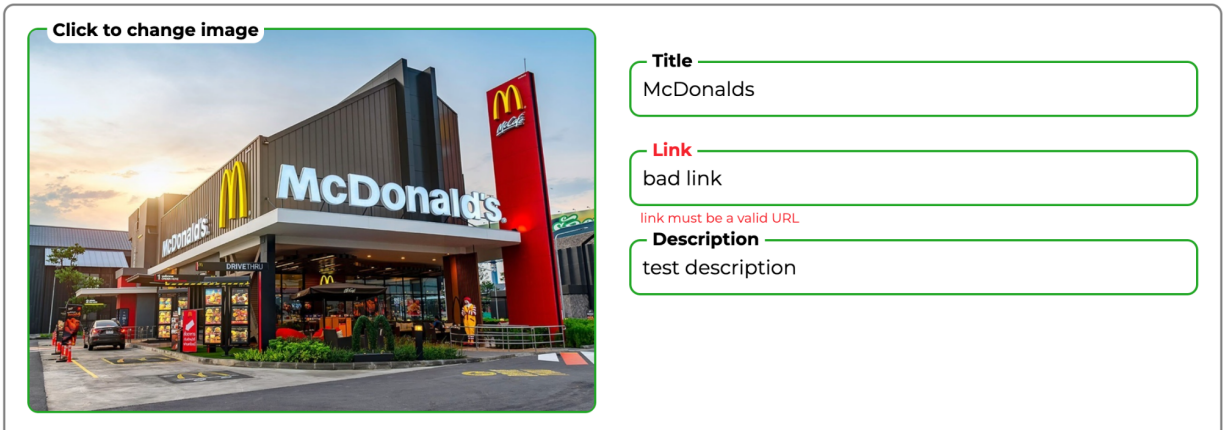

Рисунок 3.2 — Тестування методу “saveItem”.

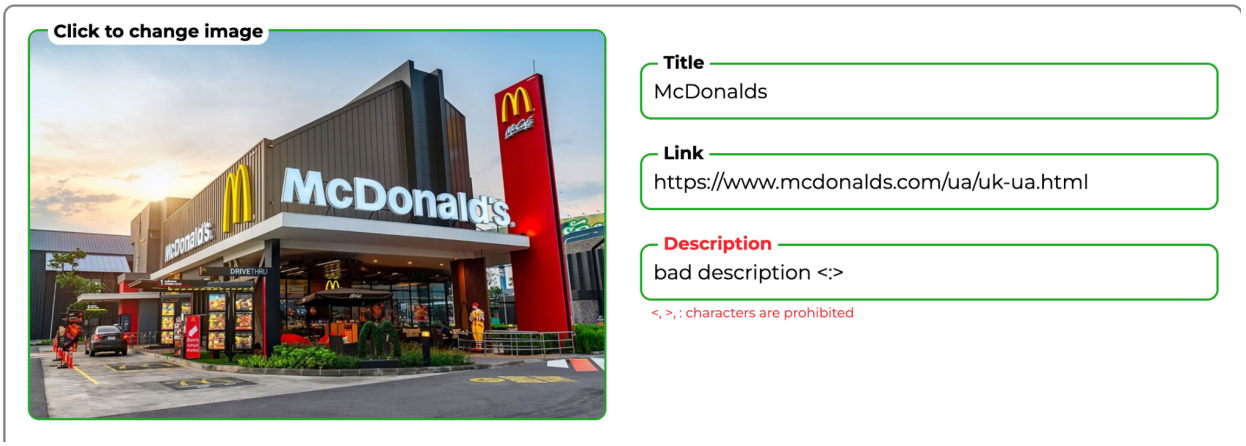

Рисунок 3.3 — Тестування методу “saveItem”.

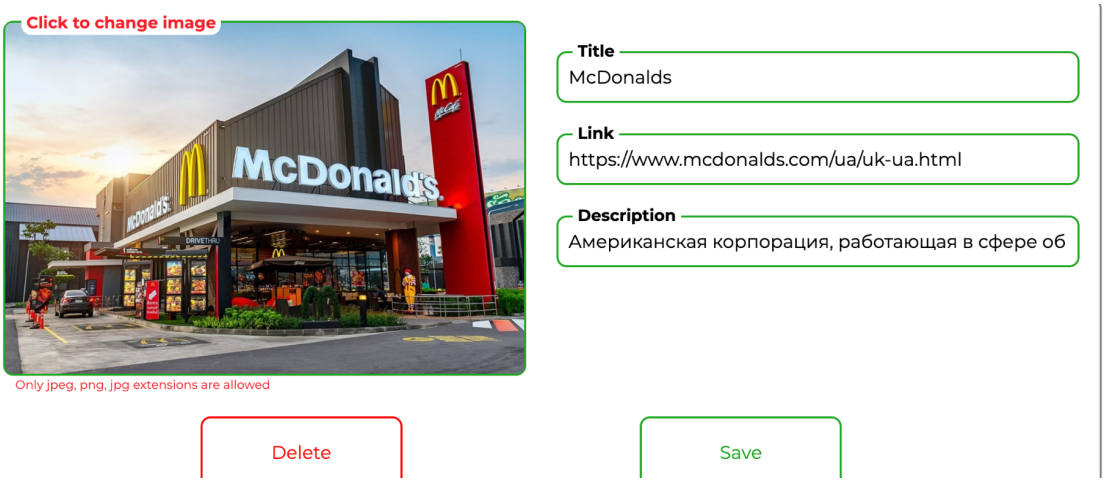

Рисунок 3.4 — Тестування методу “saveItem”.

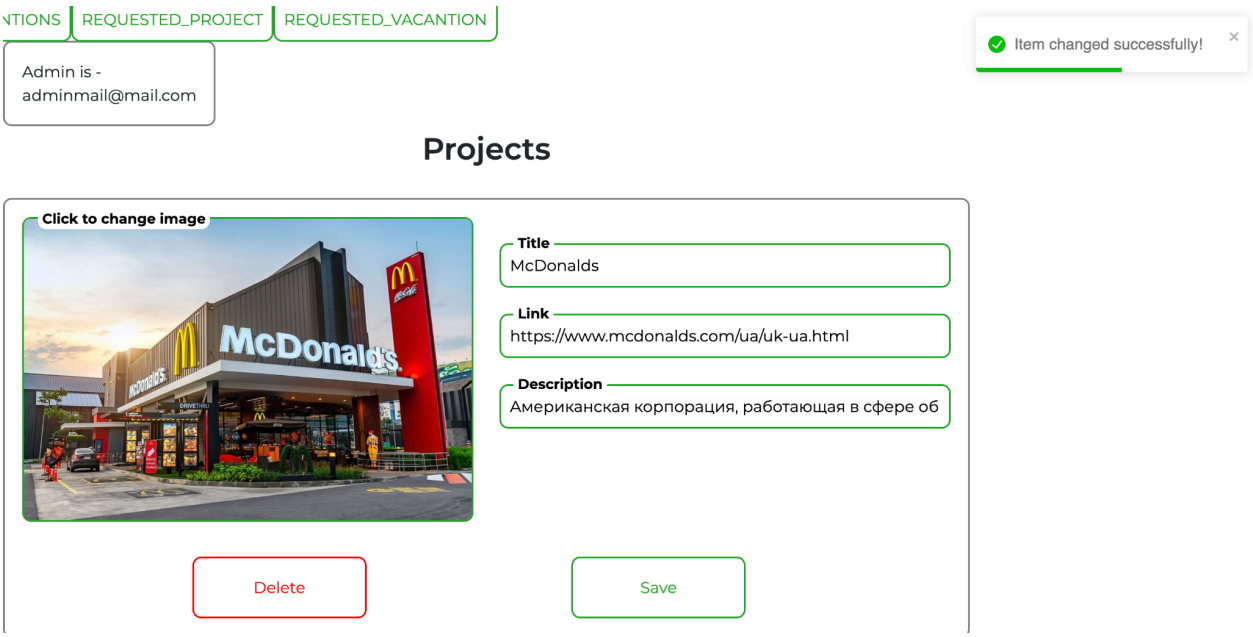

Рисунок 3.5 — Тестування методу “saveItem”.

Таблиця 3.2 — Покриття умов (метода login).

	isValidEmail		isValidPassword		isEmptyEmail		isEmptyPassword		isValidForm	
	+	-	+	-	+	-	+	-	+	-
1		*	*			*		*		*
2	*			*		*		*		*
3		*	*		*			*		*
4	*			*		*	*			*
5	*		*		*		*		*	

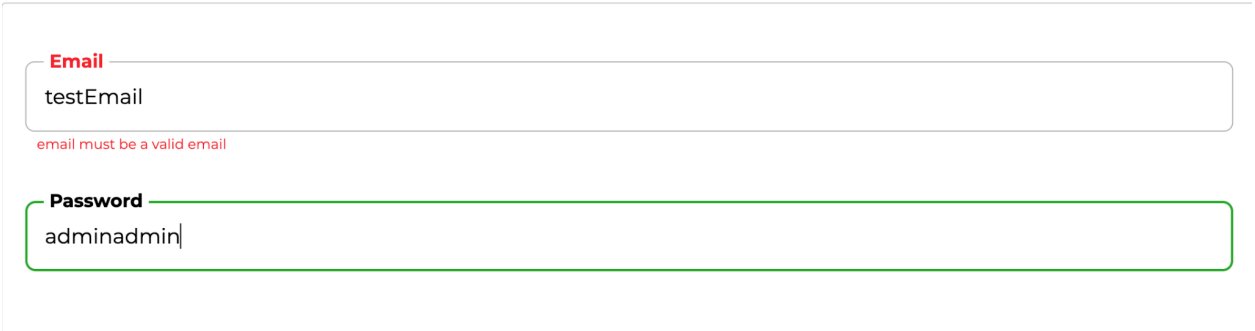

Рисунок 3.6 — Тестування методу “login”.

Рисунок 3.7 — Тестування методу “login”.

Рисунок 3.8 — Тестування методу “login”.

Рисунок 3.9 — Тестування методу “login”.

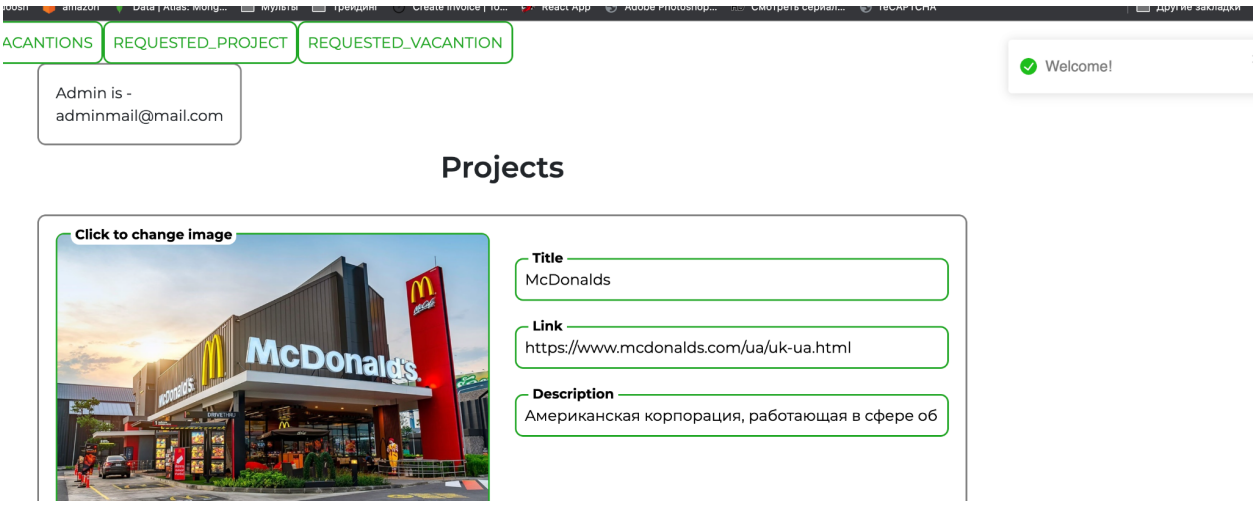

Рисунок 3.10 — Тестування методу “login”.

3.2. Тестування методом «чорної скриньки»

3.2.1. Специфікація функції

Для демонстрації тестування програми методом чорної скриньки було обрано функцію “SubmitProjectForm”, яка зберігає отримані від користувача дані про замовлений проєкт.

Функція приймає на вхід:

у явному вигляді: 5 строк типу string,

у неявному вигляді: тип форми type та id збереженої форми.

Функція повертає на вихід:

у явному виді: повідомлення про успішну відправку даних,

у неявному виді: збережений об’єкт форми у базі даних.

3.2.2. Розробка тестів та демонстрація їх відповідності методам тестування.

3.2.2.1. Класи еквівалентності

Виділені класи наведено у таблиці класів еквівалентності (табл. 3.3), у дужках зазначено номери класів. Правильні класи було сформовано так, щоб значення відповідних вхідних даних задовольняли специфікацію. Результати тестування наведені у вигляді таблиці (табл. 3.4). В ході виконання тестів встановлено, що отримані результати за класами збігається з очікуваним

Таблиця 3.3 — Класи еквівалентності.

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
1	2	3
Строка типу String “service”	Обрана строка з вибірки ["Web Design", "Web Shop", "Logo Design", "Ecxhange Service", "SaaS Platform", "Landingpage", "Other"] (1)	Необрана строка (2)
Строка типу String “description”	Строка, яка підпадає під регулярний вираз [/^[а-яА-Яа-zA-Z0-9_"'/ $!@#^*+=?&();,.’“ ”\-%]+$/], та її максимальна к-сть символів - 512, мінімальна к-сть символів - 2. (3)	Строка, яка не підпадає під регулярний вираз [/^[а-яА-Яа-zA-Z0-9_"'/ $!@#^*+=?&();,.’“ ”\-%]+$/]. (4) Строка довша за 512 символів. (5) Строка коротша за 2 символи. (6)

Продовження табл. 3.3.

1	2	3
Строка типу String “email”	Строка, яка підпадає під регулярний вираз [(?:[a-z0-9!#$%&'*+/=?^_`{\|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])] та її максимальна к-сть символів 50, мінімальна - 2 (7).	Строка, яка не підпадає під регулярний вираз [(?:[a-z0-9!#$%&'*+/=?^_`{\|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9]))\.){3}(?:(2(5[0-5]|[0-4][0-9])|1[0-9][0-9]|[1-9]?[0-9])|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])]. (8) Строка довша за 50 символів. (9) Строка коротша за 2 символи. (10)
Строка типу String “name”	Строка, яка підпадає під регулярний вираз [/^[а-яА-Яа-zA-Z\s]+$/], та її максимальна к-сть символів - 50, мінімальна к-сть символів - 2. (11)	Строка, яка не підпадає під регулярний вираз [/^[а-яА-Яа-zA-Z0-9_"'/$!@#^*+=?&();,.’“ ”\-%]+$/]. (12) Строка довша за 50 символів. (13) Строка коротша за 2 символи. (14)

Закінчення табл. 3.3.

1	2	3
Строка типу String “budget”	Обрана строка з вибірки ["5-10k", "10-50k", "more than 50k"]. (15)	Необрана строка. (16)

3.2.2.2. Тестування методом граничних умов.

Метод граничних умов для даного прикладу не є ефективним з огляду на те, що обмеження накладаються лише типами даних.

3.2.2.3. Тестування методом припущення про помилку.

1) Якщо поля “service” та/або “budget” не буде обрано, то чи завершиться виконання функції повідомленням про помилку?
2) Якщо поля “description”, “email”, “name” не буде заповнено, то чи завершиться виконання функції повідомленням про помилку?
3) Якщо поля “description”, “email”, “name” буде заповнено більшою кількістю символів, то чи завершиться виконання функції повідомленням про помилку?
4) Якщо поля “description”, “email”, “name” буде заповнено з недопустими символами, то чи завершиться виконання функції повідомленням про помилку?

Результати тестування наведені у вигляді таблиці (табл. 3.5).

3.2.2.4. Тести за методом еквівалентних розбиттів.

Таблиця 3.4 — Тести методом еквівалентних розбиттів.

Клас, що покривається	Тест	Вхід	Вихід
1	2	3	4
2, 4, 5, 6, 8, 9, 10, 12, 13, 14, 16.	1	service = пусте значення description = "" email = "" name = “""” budget = пусте значення	service = Повідомлення про помилку “Services must be selected” description = Повідомлення про помилку “<, >, : characters are prohibited” email = Повідомлення про помилку “Email must be a valid email” name = Повідомлення про помилку “Name is not valid” budget = Повідомлення про помилку “Budget must be selected”

Закінчення табл. 3.4.

1	2	3	4
1, 3, 7, 11, 15	2	service = перший елемент вибірки. description = “тестовий текст” email = “testEmail@mail com” name = “Ваня” budget = перший елемент вибірки	Повідомлення про успішну відправку форми.

3.2.2.5. Тести за методом припущення про помилку.

Таблиця 3.5 — Тести методом припущення про помилку.

Припущення	Тест	Вхід	Вихід
1	2	3	4
1	1	service = пусте значення. description = “тестовий текст” email = “testEmail@mail com” name = “Ваня” budget = пусте значення.	2 повідомлення про помилку заповнення форми: “Services must be selected”, “Budget must be selected”. Дані не відправлені до бази даних.

Продовження табл. 3.5.

1	2	3	4
2	2	service = перший елемент вибірки. description = пусте значення. email = пусте значення. name = перший елемент вибірки. budget = перший елемент вибірки.	3 повідомлення про помилку заповнення форми: “Too Short!” “Too Short!” “Too Short!”. Дані не відправлені до бази даних.
3	3	service = перший елемент вибірки. description = "Тестовий текст Тестовий текст …Тестовий текст Тестовий текст Тестовий текст" email = "testtexttesttexttesttexttesttexttesttext@mail.com" name = “Тестовий текст Тестовий текст Тестовий текст Тестовий текст Тестовий текст ” budget = перший елемент вибірки.	3 повідомлення про помилку заповнення форми: “Too Long!” “Too Long!” “Too Long!”. Дані не відправлені до бази даних.

Закінчення табл. 3.5.

1	2	3	4
4	4	service = перший елемент вибірки. description = "" email = "" name = “""” budget = перший елемент вибірки.	3 повідомлення про помилку заповнення форми: “<, >, : characters are prohibited” “Email must be a valid email” “Name is not valid”. Дані не відправлені до бази даних.

3.2.2.6. Unit-тестування

Код тестів.

```
describe('Перевірка форми замовлення проекту', () => {
  beforeEach(() => {
    cy.visit(baseUrl)
  })

  it('Перевірка валідації полів на спроможність захисту від введення стороннього коду', () => {
    cy.get('[data-cy="name-field"]')
      .type(scriptMessage)
      .blur();
    cy.get('[data-cy="name-field"]')
      .siblings('.input__error')
```

```
      .should('have.text', 'Name is not valid');

    cy.get('[data-cy="email-field"]')
      .type(scriptMessage)
      .blur();
    cy.get('[data-cy="email-field"]')
      .siblings('.input__error')
      .should('have.text', 'email must be a valid email');

    cy.get('[data-cy="description-field"]')
      .type(scriptMessage)
      .blur();
    cy.get('[data-cy="description-field"]')
      .siblings('.input__error')
      .should('have.text', '<, >, : characters are prohibited');
  });

  it('Перевірка форми на захист від спаму повідомлень', () => {
    cy.get('[data-cy="name-field"]')
      .type('Ваня');

    cy.get('[data-cy="email-field"]')
      .type('testEmail@mail.com');

    cy.get('[data-cy="description-field"]')
      .type('Тестовий текст');

    cy.get('[data-cy="service-button"]')
      .eq(0)
      .click();

    cy.get('[data-cy="budget-button"]')
      .eq(0)
      .click();

    cy.get('[data-cy="send-button"]')
      .click()
      .should('be.disabled');
  });
});
```

Результати проходження Unit-тестів показано на рис. 18.

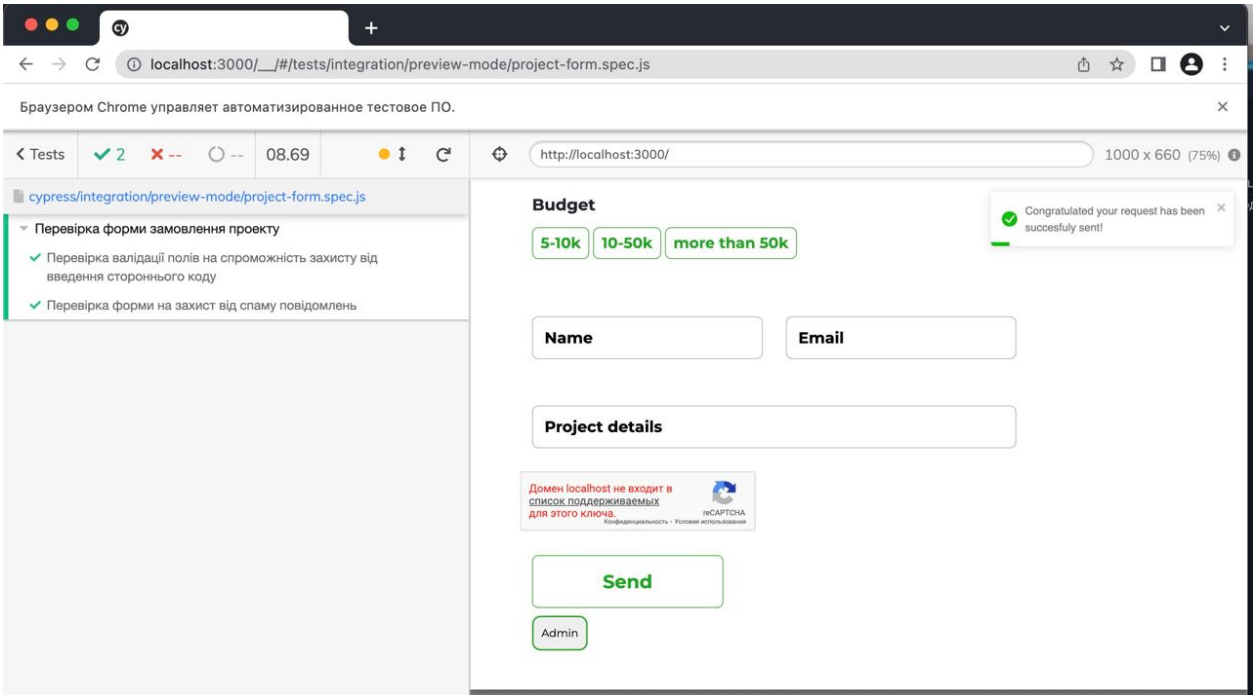

Рисунок 3.11 — Демонстрація проходження Unit-тестів.

3.3 Налагодження програми.

В ході тестування написаного веб-додатку помилок при роботі програми виявлено не було. Завдяки проведеному тестуванню вдалося перевірити валідацію полів на введення стороннього коду для XSS атак. Валідація та відправка даних працюють коректно.

Висновки: на етапі тестування веб-додатку було перевірено програму на можливі помилки за допомогою методів “білої скриньки” та “чорної скриньки”, на основі останьої були створені авто тести, які допомагають перевіряти додаток на вразливості до спам-атак на XSS атак.

ВИСНОВКИ

Результатом роботи є розроблений веб-додаток, який демонструє захист для одного із фреймворків, як для наявних базових методів, так і для власних методів розробника.

Під час виконання дипломної роботи були написані веб-додаток на основі фреймворку React, на мові Typescript, та авто тести, які демонструють базові методи захисту вбудовані у фреймворк. На основі автотестів було складено аналіз вразливостей та методи їх усунення.

Проведений аналіз загроз для веб-додатків і було обрано найбільш загрозливу та найпоширенішу з них - XSS атаку та її модифікацію XSSI. Базуючись на лазівках які ці атаки використовують, був написаний власний метод перевірки вразливості для React. Суть методу в тому, що ми перевіряємо можливі місця вразливості до стороннього коду для їх подальшого усунення. Даний метод допомагає скоротити час роботи програміста про розробці веб-додатку.

Базуючись на проведеному аналізі було виявлено вразливість веб-додатку до мульти натискання кнопок форм з метою надсилання спамових запитів.

Для усунення цієї проблеми було прийнято рішення вимикати кнопку після першого натискання та активувати її через деякий час, що дозволить запобігти надсиланню недостовірної інформації до бази даних.

Ще одним рішенням для захисту додатку від спам атак було використано відому перевірку від роботів Google Recaptcha, що дозволило запобігти відправки публічних форм без участі людини.

Результати свідчать про те, що методи захисту всіх фреймворків є недостатньо надійними і при написанні власного веб-застосунку розробник повинен не тільки використовувати базові методи захисту фреймворків, а й писати власні дивлячись на те, для чого проектувався застосунок і які на нього атаки можуть бути проведені.

Для перевірки захисту від XSS атак та спаму було зімітовано XSS атаку у вигляді автоматичного скрипта (боту), який вводить сторонній шкідливий фрагмент коду в допустимі поля. Під імітації цієї атаки написаний авто тест на бібліотеці Cypress, який перевіряє валідацію доступних полів та вимкнення кнопки для надсилання даних “Send”. Щоб у бота була можливість відправляти форми був відключений сервіс “ReCaptcha”. У звичайних умовах “ReCaptcha” ввімкнена, що не дозволяє боту відсилати форми, тим самим попереджаючи випадки спам атак. Результати роботи авто тестів наведені на рис. 4.1.

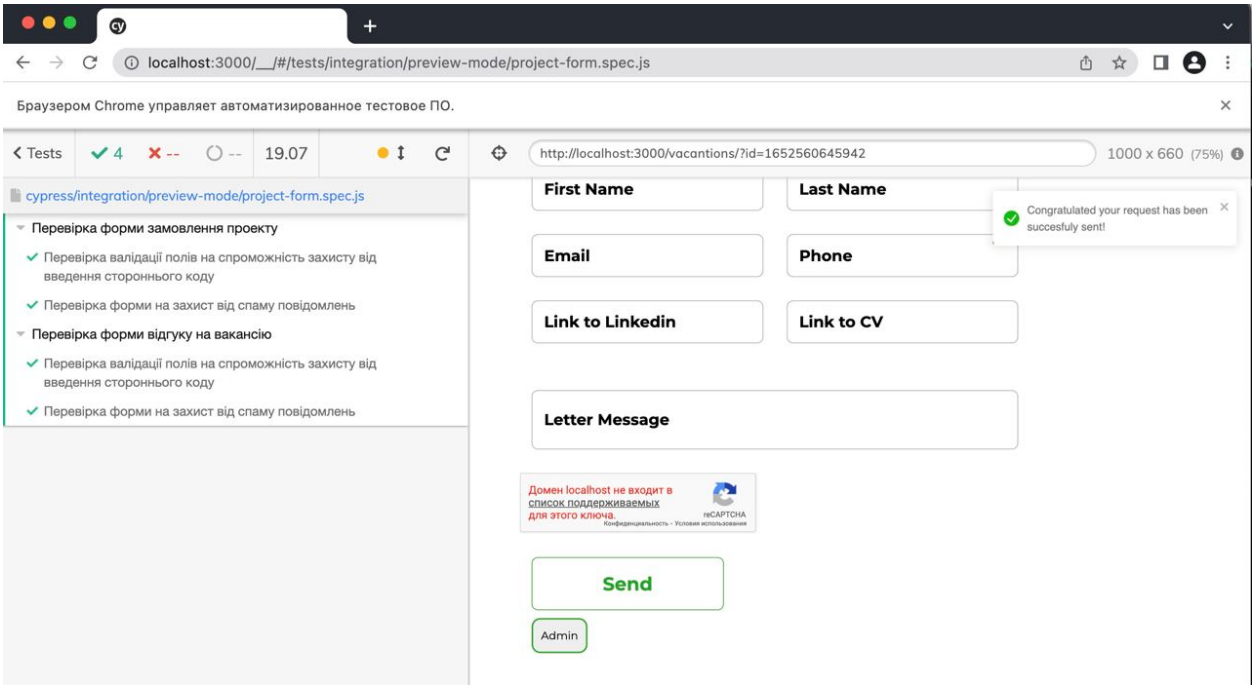

Рисунок 4.1 — Результат роботи авто тестів.

ЛІТЕРАТУРА

1. Навчальний посібник для підготовки кваліфікаційної роботи на здобуття ОС Бакалавр
2. Технічна документація з React [Електронний ресурс] https://ru.reactjs.org/docs/getting-started.html
3. Що таке XSS-вразливість https://habr.com/ru/post/511318/
4. Фреймворки в веб-разработке [Електронний ресурс] https://web-creator.ru/articles/about_frameworks.
5. Адаптивні CSS-фреймворки, сітки, класи видимості [Електронний ресурс] http://klondikestudio.ru/blog/responsive-css-framework/.
6. Технічна документація з Typescript [Електронний ресурс] https://www.typescriptlang.org/docs/
7. Результати тестування шести провідних фреймворків на продуктивність http://www.alrond.com/ru/2007/jan/25/rezultaty-testirovanija-6-frameworks
8. Що це означає, коли кажуть, що React захищений XSS? https://stackoverflow.com/questions/33644499/what-does-it-mean-when-they-say-react-is-xss-protected
9. Що таке Firebase? Розкриваємо всі таємниці [Електронний ресурс] https://blog.back4app.com/ru/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-firebase/

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету науки і
технологій
Анатолій РАДКЕВИЧ
18.02.22

ВЕБ САЙТ “COMPANY”

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01234-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
18.02.22

Керівник розробки
Вадим АНДРЮЩЕНКО
18.02.22

Виконавець
Ілля ОЛІЙНИК
18.02.22

Нормоконтролер
Олена КУРОП’ЯТНИК
18.02.22

2022

ЗАТВЕРДЖЕНО
44165850.01234-01-ЛЗ

ВЕБ САЙТ “COMPANY”

Технічне завдання

44165850.01234-01-ЛЗ

Листів 16

2022

ЗМІСТ

Вступ……4
1 Підстави для розробки……...5
2 Призначення розробки……...……6
3 Вимоги до програмного продукту……7
3.1 Вимоги до функціональних характеристик……..….7
3.2 Вимоги до надійності……..….7
3.3 Вимоги експлуатації…….8
3.4 Вимоги до складу та параметрів технічних засобів……..…8
3.5 Вимоги до інформаційної та програмної сумісності……..…..8
3.6 Вимоги до маркування і упаковки……..……9
3.7 Вимоги до транспортування та зберігання……..……..9
4 Вимоги до програмної документації……...10
5 Стадії та етапи розробки…….……11
6 Порядок і контроль приймання……12
7 Бібліографічний список……13

ВВЕДЕННЯ

Веб сайт «Company», котрий буде розроблятися, призначений для тестування методів захисту JavaScript фреймворків у браузері. На цьому сайті потрібно реалізувати подібний функціонал: захист від спам атак зі сторони клієнта, захист від “xss” та “xssi” атак, валідація вхідних даних, аутентифікація користувача.

Для веб сайту потрібно буде створити приклади, які поділені на операції:

- “Login”;
- “Add employee”;
- “Edit employee”;
- “Delete employee”;
- “Add project”;
- “Edit project”;
- “Delete project”;
- “Add vacancy”;
- “Edit vacancy”;
- “Delete vacancy”;
- “Submit form”.

Приклади можуть бути виконані в режимі реального часу.

Користувач, використовуючи цей продукт, може вносити, редагувати та видаляти своїх робітників, проекти, вакансії, та аналізувати безпеку сайту, запускаючи автотесті, і перевіривши ввод неприпустимих даних. Також даний сайт зберігає всі дані в базі даних і користувач може отримати доступ до них з будь-якого пристрою на котрому є браузер та доступ до інтернету.

Програмний продукт є безкоштовний, що робить його більш цікавим на фоні інших аналогів.

Програмний продукт призначений для людей віком від 18 до 65 років.

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 08.12.21 №77ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи - “Розробка програмних засобів аналізу методів захисту JavaScript фреймворків”. Керівник - доцент Андрющенко В. О.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт використовує браузер для доступу до інтернету та синхронізації даних на всіх пристроях. Наглядно демонструє захист JavaScript фреймворків від популярних видів атак.

Існує 11 видів операцій:

- “Add employee”;
- “Edit employee”;
- “Delete employee”;
- “Add project”;
- “Edit project”;
- “Delete project”;
- “Add vacancy”;
- “Edit vacancy”;
- “Delete vacancy”;
- “Submit form”;

Існує два режими, режим адміністратора, який має права на редагування сайту, та режим глядача, який має право на перегляд інформації.

Експлуатаційне призначення – за допомогою програмного продукту виконується тестування та аналіз методів захисту функцій сайту. Що дає змогу більш детально дослідити методи захисту JavaScript фреймворків.

3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Веб сайт повинен забезпечити можливість керування робітниками, проектами та вакансіями компанії.

До неї належать такі основні сутності:

- проект;
- вакансія;
- робітник;

У веб сайті необхідно реалізувати імітацію ведення сайту компанії, та дослідити методи захисту JavaScript фреймфорків.

Вхідні данні:

- робітник (вакансія, ФІП, фото);
- проект (назва, посилання, фото);
- вакансія (назва, вимоги);
- форма (коментар, вид проекту, бюджет проекту);

Дані вводяться з клавіатури, файли завантажуються із комп’ютера.

Вихідні дані:

- список робітників;
- список проектів;
- список вакансій;

Вихідні дані відображаються у вигляді інтерфейсу.

Самі дані зберігаються у базі даних Firebase.

Також необхідно реалізувати авто тести, які будуть тестувати методи захисту JavaScript фреймворків.

3.2 Вимоги до надійності

Вимоги до надійності наступні:

- для полів ведення необхідно забезпечити контроль ведених даних;
- при оновленні даних забезпечити показ відповідних повідомлень про стан роботи програми та результати виконання операцій;
- наявність архівної копії тексту програми на зовнішньому носії;
- наявність резервної копії бази даних на зовнішньому носії (сервері де розміщена база даних).

3.3 Умови експлуатації

Програмний продукт повинен використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.31-99 (див. табл. 1).

Таблиця 3.1 — Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.1-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		оптимальна	оптимальна	оптимальна
Холодна	легка-1 а	22 - 24	40 - 60	0,1
	легка-1 б	21 - 23	40 - 60	0,1
Тепла	легка-1 а	23 - 25	40 - 60	0,1
	легка-1 б	22 - 24	40 - 60	0,2

Працювати з програмою може людина, що має навички роботи з персональним комп’ютером та ознайомлена з керівництвом користувача.

3.4 Вимоги до складу та параметрів технічних засобів

Розроблюваний програмний продукт повинен використовуватись на ЕОМ, що мають:

- 4 ГБ оперативної пам’яті;
- 69 ГБ простору на жорсткому диску;
- клавіатуру;
- мишу;
- доступ до інтернету;
- монітор;
- USB-порт.

3.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем.Обов’язковим до встановлення є будь-який браузер з переліку: “Opera”, “Google Chrome”, “Firefox”, “Safari”.

3.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних). На упаковці повинно бути вказана назва продукту, номер версії (якщо вона змінювалась), мінімальні системні вимоги. На зворотній стороні упаковки вказується розробник та його юридична адреса.

3.7 Вимоги до транспортування і зберігання

Транспортування повинне забезпечувати збереження програмного продукту, його цілісність і запобігання несанкціонованого доступу до нього. Програмний виріб міститься на хмарному носії, переданий на флешці.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- специфікація;
- текст програми;
- опис програми;

Вся документація до програмного додатку повинна задовольняти вимоги до програмної документації.

5. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1 — Стадії та етапи розробки.

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.22 - 18.02.22
Робочий проект	Програмування та відладка програми.	19.02.22 - 01.05.22
	Тестування програми	02.05.22 - 24.05.22
	Розробка, узгодження і затвердження програмної документації.	25.05.22 - 12.06.22

6. ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Андрющенко В. О.

Прийом здійснюється уповноваженою комісією.

7. БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

Додаток Б

Текст програми

index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import {BrowserRouter} from "react-router-dom";
import App from "./App";

import { Provider } from "react-redux";
import { store } from "./store";

ReactDOM.render(
 <Provider store={store}>
  <React.StrictMode>
   <BrowserRouter>
    <App />
   </BrowserRouter>
  </React.StrictMode>
 </Provider>,
 document.getElementById("root")
);
```

store/index.ts

```
import { applyMiddleware, createStore, combineReducers }
from "redux";
import createSagaMiddleware from "redux-saga";
import reducers from "./reducers";
import { composeWithDevTools } from
"redux-devtools-extension";
import { rootWatcher } from "./sagas";
import Admin from "../api/Admin";

const sagaMiddleware = createSagaMiddleware();
const rootReducer = combineReducers(reducers);

export const Apis = new Admin();

export const store = createStore(
  rootReducer,
  composeWithDevTools(
    applyMiddleware(sagaMiddleware)
  )
);

sagaMiddleware.run(rootWatcher);

export type RootState = ReturnType<typeof store.getState>
export type AppDispatch = typeof store.dispatch;
```

api/Admin.ts

```
/* eslint-disable no-undef */
import firebase from "firebase";
import { IProject } from "../models/IProject";
import { ITeam } from "../models/ITeam";
import { ETypeOfContent } from "../models/ITypeOfContent";
import { IVacancy } from "../models/IVacancy";
import { toast } from "react-toastify";
import { User } from "store/reducers/admin/types";
import { IPeopleForm, IProjectForm } from "models/IForms";

export default class Admin {
  static app: firebase.app.App;
  static auth: firebase.auth.Auth;

  constructor() {
    const firebaseConfig = {
      apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
      authDomain:
process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
      projectId:
process.env.REACT_APP_FIREBASE_PROJECT_ID,
      storageBucket:
process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
      messagingSenderId:
process.env.REACT_APP_FIREBASE_SENDER_ID,
      appId: process.env.REACT_APP_FIREBASE_APP_ID,
      databaseURL: process.env.REACT_APP_DATABASE_URL
    };

    const app = firebase.initializeApp(firebaseConfig);
    const auth = firebase.auth(app);
```

```
    Admin.auth = auth;
    Admin.app = app;
  }
  async signIn(email: string, password: string): Promise<User | undefined> {
    try {
      const result = await Admin.auth.signInWithEmailAndPassword(email, password);
      return {
        photoURL: result.user?.photoURL || "",
        displayName: result.user?.displayName || "",
        email: result.user?.email || "",
      };
    } catch (error) {
      toast.error("You are unauthorized!");
    }
  }
  async getValueByDirectory(directory: string) {
    const db = firebase.database();
    const ref = db.ref(directory);
    const snapshot = await ref.once("value");
    const result = snapshot.val();
    return result;
  }
  async getProjects(): Promise<IProject[]> {
    const projects = await this.getValueByDirectory(ETypeOfContent.PROJECTS);
    if (!projects) return [];
    return Object.values(projects);
  }
  async getTeam(): Promise<ITeam[]> {
    const team = await this.getValueByDirectory(ETypeOfContent.TEAM);
    if (!team) return [];
    return Object.values(team);
  }
  async getVacancies(): Promise<IVacancy[]> {
    const vacantions = await this.getValueByDirectory(ETypeOfContent.VACANTION);
    if (!vacantions) return [];
    return Object.values(vacantions);
  }
  changeItem = (directory: string, data: IVacancy | IProject | ITeam | IPeopleForm | IProjectForm) => {
    const db = firebase.database();
    db.ref(`${directory}/${data.id}`).set(data);
  }
  deleteItem = (directory: string, id: string) => {
    const db = firebase.database();
    db.ref(`${directory}/${id}`).set(null);
  }
}
```

models/IProject.ts

```
export interface IProject {
  link: string,
  image?: string,
  title: string,
  description: string,
  id: string,
}
```

models/ITeam.ts

```
export interface ITeam {
  name: string,
  position: string,
  image?: string,
  id: string,
}
```

models/ITypeOfContent.ts

```
export enum ETypeOfContent {
  VACANTION = "vacantions",
  TEAM = "team",
  PROJECTS = "projects",
  REQ_VACANTION = "requested_vacantion",
  REQ_PROJECT = "requested_project",
}
export type ITypeOfContent = ETypeOfContent.VACANTION | ETypeOfContent.TEAM | ETypeOfContent.PROJECTS
export type ITypeOfForm = ETypeOfContent.REQ_VACANTION | ETypeOfContent.REQ_PROJECT
```

models/IVacancy.ts

```
export interface IVacancy {
  position: string,
  location: string,
  title: string,
  description: string,
  id: string,
  tasks: {
    title: string,
    id: string
  }[]
}
```

reducers/types.ts

```
import { IVacancy } from "../../../models/IVacancy";
import { ITeam } from "../../../models/ITeam";
import { IProject } from "../../../models/IProject";
import { ITypeOfContent } from
"../../../models/ITypeOfContent";
import { IProjectForm, IPeopleForm } from "models/IForms";
import { ContentActionEnum } from "../content/types";

export interface AdminState {
  user: User | null,
  projects: IProject[];
  team: ITeam[];
  vacantions: IVacancy[];
  isAuth: boolean;
  forms: Forms,
}

export type User = {
  displayName: string,
  email: string,
  photoURL: string,
}

export type Forms = {
  requestedProjects: IProjectForm[],
  requestedVacancies: IPeopleForm[],
}

export enum AdminActionEnum {
  AUTH = "AUTH",
  SET_PROJECTS = "SET_PROJECTS",
  SET_TEAM = "SET_TEAM",
  SET_VACANTIONS = "SET_VACANTIONS",
  SET_USER = "SET_USER",
  SAVE_ITEM = "SAVE_ITEM",
  SET_FORMS = "SET_FORMS",
  ADD_ITEM = "ADD_ITEM",
  DELETE_ITEM = "DELETE_ITEM",
}

export interface Login {
  type: AdminActionEnum.AUTH;
  payload: {
    callBack: () => void,
    email: string,
    password: string,
  }
}
export interface SetProjectsAction {
  type: AdminActionEnum.SET_PROJECTS;
  payload: IProject[]
}

export interface SetTeamAction {
  type: AdminActionEnum.SET_TEAM;
  payload: ITeam[]
}

export interface SetVacanciesAction {
  type: AdminActionEnum.SET_VACANTIONS;
  payload: IVacancy[]
}

export interface SetAuthAction {
  type: AdminActionEnum.SET_USER;
  payload: User;
}

export interface SetForms {
```

```
  type: AdminActionEnum.SET_FORMS;
  payload: {
    requestedProjects: IProjectForm[],
    requestedVacancies: IPeopleForm[],
  }
}

export interface SaveItem {
  type: AdminActionEnum.SAVE_ITEM;
  payload: {
    type: ITypeOfContent,
    data: IProject | ITeam | IVacancy,
    id: string,
  };
}

export interface AddItem {
  type: AdminActionEnum.ADD_ITEM;
  payload: {
    type: ITypeOfContent,
    data: IProject | ITeam | IVacancy,
  };
}

export interface DeleteItem {
  type: AdminActionEnum.DELETE_ITEM;
  payload: {
    type: ITypeOfContent,
    id: string,
  };
}

export type AdminAction =
  SetProjectsAction |
  SetTeamAction |
  SetVacanciesAction |
  SetAuthAction |
  AddItem |
  DeleteItem |
  SaveItem |
  SetForms
```

models/IForms.ts

```
export interface IProjectForm {
 name: string,
 email: string,
 description: string,
 budget: string,
 service: string,
 id: string,
}

export interface IPeopleForm {
 firstName: string,
 lastName: string,
 email: string,
 phone: string,
 linkedin: string,
 cvLink: string,
 description: string,
 id: string,
}
```

sagas/index.ts

```
import { spawn, all } from "redux-saga/effects";
import { watchFetchDataSaga } from "./fetchData";
import { watchChangeItem } from "./changeItem";
import { watchAuth } from "./auth";

export function* rootWatcher() {
  yield spawn(watchFetchDataSaga);
  yield spawn(watchChangeItem);
  yield spawn(watchAuth);
}
```

App.tsx

```
import React from "react";
import { Element as ScrollElement } from "react-scroll";

// Styles
import "./assets/styles/index.scss";
```

```
// Components
import Header from "./components/Header";
import Footer from "./components/Footer";
import Loader from "./components/Loader";
import AppRouter from "./components/AppRouter";
import { ToastContainer } from "react-toastify";

const App = () => {
 return (
  <div className="main-container">
    <Loader />
    <Header />
    <AppRouter />
    <ScrollElement name="Bottom">
      <Footer />
    </ScrollElement>
    <ToastContainer />
  </div>
 );
};

export default App;
```

styles/index.scss

```
@charset "utf-8";
* {
  margin: 0;
  padding: 0;
  border: 0;
  box-sizing: border-box;
  text-decoration: none;
}

a {
  text-decoration: none;
}

html {
  font-size: 16px;
}

body {
  min-width: 320px;
  font-family: 'Montserrat', sans-serif;
}

html,
body {
  height: 100%;
  display: flex;
  flex-direction: column;
}

.site-wrap {
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: auto;
}

@import "tools/vars";
@import "tools/animation";
@import "tools/main";
@import "tools/media";
@import "tools/utils";
@import "react-toastify/dist/ReactToastify.css";
```

components/Loader/index.tsx

```
// Styles
import "./Loader.scss";

import { useTypedSelector } from
"../../hooks/useTypedSelector";
import React from "react";
import Logo from "../Logo";

const Loader = () => {
 const showLoader = useTypedSelector(state =>
state.content.showLoader);
 const rootEl = document.querySelector("#root");

 if (!rootEl) return <></>;

 if (!showLoader) {
```

```
  rootEl.classList.remove("hidden");
 } else {
  rootEl.classList.add("hidden");
 }

 return (
  <div className={`loader ${showLoader ? "loader--show" :
""}`}>
    <Logo isBlue />
  </div>
 );
};

export default Loader;
```

hooks/useTypedSelector.ts

```
import { TypedUseSelectorHook, useSelector } from
"react-redux";
import { RootState } from "../store";

export const useTypedSelector:
TypedUseSelectorHook<RootState> = useSelector;
```

components/Header/index.tsx

```
// Core
import React, { FC, useState } from "react";
import { Link } from "react-router-dom";
import { Link as SlowLink } from "react-scroll";

// Styles
import "./Header.scss";

// Images
import { useLocation } from "react-router-dom";

// Helpers
import { getDurationOfAnimation } from "../../helper";
import Logo from "../Logo";

const Header: FC = () => {
 const [showMenu, setShowMenu] = useState(false);
 const location = useLocation();

 const headerLinks = [
  {
   title: "Projects",
   href: "/#projects",
  },
  {
   title: "Services",
   href: "/#services",
  },
  {
   title: "Company",
   href: "/#about"
  }
 ];

 const isNotMainPage = () => {
  const routes = ["/vacantions", "/projects"];
  for (const route of routes) {
   if (location.pathname.startsWith(route)) {
    return true;
   }
  }
  return false;
 };

 const isAdminPage = () => {
  return location.pathname.includes("/admin") ||
location.pathname.includes("/login");
 };

 const toggleMenu = () => {
  setShowMenu(!showMenu);
  const rootEl = document.querySelector("#root");

  if (!rootEl) return;

  if (showMenu) {
   rootEl.classList.remove("hidden");
  } else {
```

```
    rootEl.classList.add("hidden");
  }
};

if (isAdminPage())
  return <></>;

return (
  <header id="header" className={`container-fluid header ${isNotMainPage() ? "header--blue" : ""}`}>
    <div>
      <nav className="top-header navbar navbar-expand-md navbar-light">
        <Link to="/">
          {isNotMainPage() ? <Logo isBlue /> : <Logo />}
        </Link>
        <div className="collapse navbar-collapse" id="navbar-menu">
          <div className="navbar-nav">
            {headerLinks.map(({ title, href }) => isNotMainPage() ? (
              <Link to={href} key={title} className="link_animation">{title}</Link>
            ) : (
              <SlowLink key={title} className="link_animation" to={title} smooth={true} duration={getDurationOfAnimation()}>
                {title}
              </SlowLink>
            ))}
          </div>
        </div>
        <div onClick={toggleMenu} className={`header__menu d-md-none ${showMenu ? "open" : ""}`}>
          <span />
          <span />
          <span />
        </div>
      </nav>

      <div className={`header-menu js-menu ${showMenu ? "open" : ""}`}>
        <ul>
          {headerLinks.map(({ title, href }) => (
            <li key={title} className="link_animation">
              {
                isNotMainPage() ? (
                  <Link onClick={toggleMenu} to={href} key={title} className="scroll js-mob-scroll">{title}</Link>
                ) : (
                  <SlowLink onClick={toggleMenu} key={title} className="scroll js-mob-scroll" to={title} smooth={true} duration={getDurationOfAnimation()}>
                    {title}
                  </SlowLink>
                )
              }
            </li>
          ))}
        </ul>
      </div>
    </div>
  </header>
);
};

export default Header;
```

components/Footer/index.tsx

```
// Core
import { useHistory, useLocation } from "react-router-dom";

// Styles
import "./Footer.scss";

// Components
import PeopleForm from "../PeopleForm";
import ProjectForm from "../ProjectForm";
import * as Yup from "yup";
import { Formik, Form, Field } from "formik";
import CustomInput from "../CustomInput";
import Modal from "react-modal";

// Constants
import { socialsData } from "../../constants";
import React, { FC, useState } from "react";
import { useActions } from "hooks/useActions";
```

```
const Footer: FC = () => {
 const location = useLocation();
 const [isOpenModal, setIsOpenModal] = useState(false);
 const [disableButton, setDisableButton] = useState(false);
 const { login } = useActions();
 const history = useHistory();

 const renderProjectForm = () => {
  return !location.pathname.includes("/vacantions");
 };

 const isAdminPage = () => {
  return location.pathname.includes("/admin") ||
location.pathname.includes("/login");
 };

 if (isAdminPage())
  return <></>;

 const schema = Yup.object().shape({
  email: Yup
   .string()
   .trim()
   .email()
   .max(50, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),

  password: Yup
   .string()
   .trim()
   .matches(/^[a-zA-z]+$/, "Password is invalid")
   .max(512, "Too Long!")
   .min(6, "Too Short!")
   .required("Required"),
 });

 const submitForm = (values: { email: string, password: string })
=> {
  setDisableButton(true);
  login({
   ...values,
   callBack: () => history.push("/admin")
  });

  setTimeout(() => {
   setDisableButton(false);
  }, 5000);
 };

 const customStyles = {
  content: {
   top: "50%",
   left: "50%",
   right: "auto",
   bottom: "auto",
   marginRight: "-50%",
   transform: "translate(-50%, -50%)",
  },
 };

 return (
  <footer className="footer container-fluid
container__indent" id="bottom">
   <div className="row">
    {renderProjectForm() ? <ProjectForm /> : <PeopleForm />}
   </div>
   <button
    className="submit__rounded submit"
    onClick={() => setIsOpenModal(true)}
   >
    Admin
   </button>
   <Modal
    isOpen={isOpenModal}
    contentLabel="Sign in Modal"
    style={customStyles}
   >

    <Formik
     initialValues={{
```

```
        password: "",
        email: "",
       }}
       validationSchema={schema}
       onSubmit={submitForm}
      >
       <Form className="form">
        <div className="row peopleForm">
         <div className="col-md-12">
          <Field data-cy="email-field" name="email"
component={CustomInput} text="Email" />
         </div>
         <div className="col-md-12 mt-30">
          <Field data-cy="password-field" name="password"
component={CustomInput} text="Password" />
         </div>
         <div className="col-md-12 mt-30 ">
          <button
           disabled={disableButton}
           className="form-btn submit"
           type="submit"
           data-cy="send-button"
          >
           Log In
          </button>
          <button className="form-btn submit" onClick={() =>
setIsOpenModal(false)}>
           Close
          </button>
         </div>
        </div>
       </Form>
      </Formik>
    </Modal>
  </footer >
 );
};

export default Footer;
```

hooks/useActions.ts

```
import { useDispatch } from "react-redux";
import { bindActionCreators } from "redux";
import { allActionCreators } from
"../store/reducers/action-creators";

export const useActions = () => {
  const dispatch = useDispatch();
  return bindActionCreators(allActionCreators, dispatch);
};
```

components/CustomInput/index.tsx

```
import { FC } from "react";
import { FieldProps } from "formik";

interface CustomInputProps {
 isTextArea: boolean,
 text: string,
}

const CustomInput: FC<CustomInputProps & FieldProps> = ({
 isTextArea,
 field, // { name, value, onChange, onBlur }
 form,
 form: { touched, errors, submitCount }, // also values, setXXXX,
handleXXXX, dirty, isValid, status, etc.
 ...props
}) => {
 const showErrorIfRequired = () => {
  return (touched[field.name] && errors[field.name] &&
field.value) || (submitCount && !field.value &&
errors[field.name]);
 };

 return (
  <div className="input">
   {
    !isTextArea ? (
     <>
      <input
       {...props}
       {...field}
       className="input__field"
      />
      <span className={`input__label ${touched[field.name]
|| field.value ? "input__label--active" : ""}
${showErrorIfRequired() ? "input__label--error" : ""}`}>
```

```
          {props.text}
        </span>
        {showErrorIfRequired() ? (
          <span className="input__error">{(errors as
any)[field.name]}</span>
        ) : null}
       </>
     ) : (
       <>
        <textarea
          {...props}
          {...field}
          className="input__field"
        />
        <span className={`input__label ${touched[field.name]
|| field.value ? "input__label--active" : ""}
${showErrorIfRequired() ? "input__label--error" : ""}`}>
          {props.text}
        </span>
        {showErrorIfRequired() ? (
          <span className="input__error">{(errors as
any)[field.name]}</span>
        ) : null}
       </>
     )
   }

  </div>
 );
};

export default CustomInput;
```

components/PeopleForm/index.tsx

```
/* eslint-disable no-undef */
import { useRef, useState } from "react";
import { Formik, Form, Field } from "formik";
import { ETypeOfContent } from "../../models/ITypeOfContent";
import * as Yup from "yup";
import ReCAPTCHA from "react-google-recaptcha";

// Styles
import "./PeopleForm.scss";

// Componrnts
import CustomInput from "../CustomInput";
import { useActions } from "../../hooks/useActions";
import { toast } from "react-toastify";
import { generateIdByTime } from "helper";

const PeopleForm = () => {
 const [disableButton, setDisableButton] = useState(false);
 const { saveForm } = useActions();
 const [recaptcha, setRecaptcha] = useState("");
 const grecaptcha = useRef<any>(null);

 const submitForm = async (values: any, { resetForm }: any) => {
  if (Number(process.env.REACT_APP_IS_ENABLED_CAPTCHA)
&& !recaptcha) {
   return;
  }

  setDisableButton(true);
  resetForm({});

  saveForm({
   type: ETypeOfContent.REQ_VACANTION,
   data: { ...values, id: generateIdByTime(), },
  });

  setTimeout(() => {
   setDisableButton(false);
   grecaptcha.current.reset();
  }, 5000);
 };

 const schema = Yup.object().shape({
  firstName: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Name is not valid")
   .min(2, "Too Short!")
   .max(50, "Too Long!")
   .required("Required"),
```

```
  lastName: Yup
    .string()
    .trim()
    .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Name is not valid")
    .min(2, "Too Short!")
    .max(50, "Too Long!")
    .required("Required"),

  email: Yup
    .string()
    .trim()
    .email()
    .max(50, "Too Long!")
    .min(2, "Too Short!")
    .required("Required"),

  phone: Yup
    .string()
    .trim()
.matches(/^(\+?\d{0,4})?\s?\s?(\(?\d{3}\)?)\s?\s?(\(?\d{3}\)?)\
s?\s?(\(?\d{4}\)?)?$/, "Phone number is not valid")
    .max(20, "Too Long!")
    .min(2, "Too Short!")
    .required("Required"),

  linkedin:
Number(process.env.REACT_APP_IS_ENABLED_HACK_MODE)
    ? Yup
      .string()
      .trim()
      .max(512, "Too Long!")
      .min(2, "Too Short!")
      .required("Required")
    : Yup
      .string()
      .trim()
      .url()
      .max(512, "Too Long!")
      .min(2, "Too Short!")
      .required("Required"),

  cvLink:
Number(process.env.REACT_APP_IS_ENABLED_HACK_MODE)
    ? Yup
      .string()
      .trim()
      .max(512, "Too Long!")
      .min(2, "Too Short!")
      .required("Required")
    :
    Yup
      .string()
      .trim()
      .url()
      .max(512, "Too Long!")
      .min(2, "Too Short!")
      .required("Required"),

  description: Yup
    .string()
    .trim()
    .matches(/^[а-яА-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.'“
”\-%]+$/, "<, >, : characters are prohibited")
    .max(512, "Too Long!")
    .min(2, "Too Short!"),
 });

 return (
  <div className="col-lg-9 peopleForm">
    <Formik
      initialValues={{
        firstName: "",
        lastName: "",
        email: "",
        phone: "",
        linkedin: "",
        cvLink: "",
        description: "",
      }}
      validationSchema={schema}
      onSubmit={submitForm}
    >
```

```
    {({ errors, touched, values, handleChange, isValid }) => (
      <Form className="form">
        <div className="row">
          <div className="col-md-6">
            <Field data-cy="firstName-field" name="firstName"
component={CustomInput} text="First Name" />
          </div>
          <div className="col-md-6">
            <Field data-cy="lastName-field" name="lastName"
component={CustomInput} text="Last Name" />
          </div>
        </div>
        <div className="row">
          <div className="col-md-6">
            <Field data-cy="email-field" name="email"
component={CustomInput} text="Email" />
          </div>
          <div className="col-md-6">
            <Field data-cy="phone-field" name="phone"
component={CustomInput} text="Phone" />
          </div>
        </div>
        <div className="row">
          <div className="col-md-6">
            <Field data-cy="linkedin-field" name="linkedin"
component={CustomInput} text="Link to Linkedin" />
          </div>
          <div className="col-md-6">
            <Field data-cy="cvLink-field" name="cvLink"
component={CustomInput} text="Link to CV" />
          </div>
          <div className="col-md-12">
            <Field data-cy="description-field" isTextArea
name="description" component={CustomInput} text="Letter
Message" />
          </div>
          <ReCAPTCHA
            sitekey={process.env.REACT_APP_CAPTCHA_KEY || ""}
            onChange={(value) => {
              setRecaptcha(value as string);
            }}
            ref={grecaptcha}
          />
          <div className="col-md-12">
            <button
              disabled={disableButton}
              className="form-btn submit"
              type="submit"
              data-cy="send-button"
            >
              Send
            </button>
          </div>
        </div>
      </Form>
    )}
  </Formik>
 </div>
);
};

export default PeopleForm;
```

components/ProjectForm/index.tsx

```
/* eslint-disable no-undef */
import React, { useEffect, useRef, useState } from "react";
import { Formik, Form, Field } from "formik";
import ReCAPTCHA from "react-google-recaptcha";
import * as Yup from "yup";

// Components
import CustomInput from "../CustomInput";
import { useActions } from "../../hooks/useActions";
import { generateIdByTime } from "helper";
import { ETypeOfContent } from "models/ITypeOfContent";

const ProjectForm = () => {
 const [service, setService] = useState("");
 const [budget, setBudget] = useState("");
 const [recaptcha, setRecaptcha] = useState("");
 const [budgetError, setBudgetError] = useState(false);
 const [serviceError, setServiceError] = useState(false);
 const { saveForm } = useActions();
 const grecaptcha = useRef<any>(null);
```

```
 const [disableButton, setDisableButton] = useState(false);


 const servicesContent = ["Web Design", "Web Shop", "Logo
Design", "Ecxhange Service", "SaaS Platform", "Landingpage",
"Other"];


 const budgetContent = ["5-10k", "10-50k", "more than 50k"];


 const submitForm = async (values: any, { resetForm }: any) => {
  if (Number(process.env.REACT_APP_IS_ENABLED_CAPTCHA)
&& !recaptcha) {
    return;
  }
  if (!service || !budget) {
    return;
  }


  setDisableButton(true);
  clearData();
  resetForm({});


  saveForm({
    type: ETypeOfContent.REQ_PROJECT,
    data: {
      ...values,
      budget,
      service,
      id: generateIdByTime(),
    },
  });
  setTimeout(() => {
    setDisableButton(false);
  }, 3000);
  grecaptcha.current.reset();
};


 const checkServices = () => {
  if (!service) {
    setServiceError(true);
  }
  if (!budget) {
    setBudgetError(true);
  }
};


const clearData = () => {
 setService("");
 setBudget("");
};


useEffect(() => {
 if (service) {
   setServiceError(false);
 }
 if (budget) {
   setBudgetError(false);
 }
}, [service, budget]);


const schema = Yup.object().shape({
 name: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Name is not valid")
   .min(2, "Too Short!")
   .max(50, "Too Long!")
   .required("Required"),


 email: Yup
   .string()
   .trim()
   .email()
   .max(50, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),


 description: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.‘“
”\-%]+$/, "<, >, : characters are prohibited")
   .max(512, "Too Long!")
   .min(2, "Too Short!")
```

```
    .required("Required"),
 });


 return (<div className="col-lg-9">
  <div>
   <h2>Create your project</h2>
  </div>
  <div>
   <h6 className="mt-5">Services{serviceError && <span
className="errorMessage"> (Services must be
selected)</span>}</h6>
   {
    servicesContent.map(item => (
     <button
      data-cy="service-button"
      onClick={() => setService(item)}
      key={item}
      className={`button__services ${item === service ?
"button__services--active" : ""}`}
     >
      {item}
     </button>
    ))
   }
  </div>
  <div className="pb-3">
   <h6 className="mt-5">Budget{budgetError && <span
className="errorMessage"> (Budget must be
selected)</span>}</h6>
   {
    budgetContent.map(item => (
     <button
      data-cy="budget-button"
      onClick={() => setBudget(item)}
      key={item}
      className={`button__services ${item === budget ?
"button__services--active" : ""}`}
     >
      {item}
     </button>
    ))
   }
  </div>
  <div className="mt-5">
   <Formik
    initialValues={{
     name: "",
     email: "",
     description: "",
    }}
    validationSchema={schema}
    onSubmit={submitForm}
   >
    {({ errors, touched, values, handleChange, isValid,
resetForm, setFieldValue }) => (
     <Form className="form">
      <div className="row peopleForm">
       <div className="col-md-6">
        <Field data-cy="name-field" name="name"
component={CustomInput} text="Name" />
       </div>
       <div className="col-md-6">
        <Field data-cy="email-field" name="email"
component={CustomInput} text="Email" />
       </div>
       <div className="col-md-12 mt-30">
        <Field data-cy="description-field" name="description"
component={CustomInput} text="Project details" />
       </div>
       <ReCAPTCHA
        sitekey={process.env.REACT_APP_CAPTCHA_KEY || ""}
        onChange={(value) => {
         setRecaptcha(value as string);
        }}
        ref={grecaptcha}
       />
       <div className="col-md-12 mt-30 ">
        <button
         onClick={checkServices}
         disabled={disableButton}
         className="form-btn submit"
         type="submit"
         data-cy="send-button"
        >
         Send
        </button>
```

```
          </div>
        </div>
       </Form>
      )}
    </Formik>
  </div>
 </div>
 );
};

export default ProjectForm;
```

components/AppRouter.tsx

```
import React from "react";
import { Switch, Route, Redirect } from "react-router-dom";
import { privateRoutes, publicRoutes, RouteNames } from
"../router";
import { useTypedSelector } from "../hooks/useTypedSelector";
import { FC } from "react";

const AppRouter: FC = () => {
 const isAuth = useTypedSelector(state => state.admin.isAuth);

 return (
  isAuth ?
   <Switch>
    {privateRoutes.map(route =>
     <Route path={route.path}
      exact={route.exact}
      component={route.component}
      key={route.path}
     />
    )}
    <Redirect to={RouteNames.HOME} />
   </Switch>
   :
   <Switch>
    {publicRoutes.map(route =>
     <Route path={route.path}
      exact={route.exact}
      component={route.component}
      key={route.path}
     />
    )}
    <Redirect to={RouteNames.HOME} />
   </Switch>
 );
};

export default AppRouter;
```

router/index.tsx

```
import React from "react";
import AdminPanel from "../pages/AdminPanel";
import Home from "../pages/Home";
import ProjectsPage from "../pages/ProjectsPage";
import Vacancies from "../pages/Vacancies";

export interface IRoute {
  path: string;
  component: React.ComponentType;
  exact?: boolean;
}

export enum RouteNames {
  VACANTIONS = "/vacantions",
  PROJECTS = "/projects",
  ADMIN_PANEL = "/admin",
  HOME = "/",
}

export const publicRoutes: IRoute[] = [
  {path: RouteNames.VACANTIONS, exact: true, component:
Vacancies},
  {path: RouteNames.PROJECTS, exact: true, component:
ProjectsPage},
  {path: RouteNames.HOME, exact: true, component: Home},
];

export const privateRoutes: IRoute[] = [
  {path: RouteNames.VACANTIONS, exact: true, component:
Vacancies},
  {path: RouteNames.PROJECTS, exact: true, component:
ProjectsPage},
```

```
  {path: RouteNames.HOME, exact: true, component: Home},
  {path: RouteNames.ADMIN_PANEL, exact: true, component: AdminPanel}
];
```

pages/Home/index.tsx

```
import { FC, useEffect } from "react";
import { Element as ScrollElement } from "react-scroll";

// Components
import MainBanner from "../../components/MainBanner";
import Projects from "../../components/Projects";
import Team from "../../components/Team";
import Carrer from "../../components/Carrer";
import { useTypedSelector } from "../../hooks/useTypedSelector";

const Home: FC = () => {
 const showLoader = useTypedSelector(state => state.content.showLoader);

 useEffect(() => {
  const hash = window.location.hash;

  if (!hash) {
   return;
  }

 }, [window.location]);

 return (
  <>
   <MainBanner />
   <ScrollElement name="Projects">
    <Projects limit={6} showAllBtn title='Projects' subTitle='Projects we are proud of' />
   </ScrollElement>
   <ScrollElement name="Services">
    <Carrer />
   </ScrollElement>
   <ScrollElement name="Company">
    <Team />
   </ScrollElement>
  </>
 );
};

export default Home;
```

components/MainBanner/index.tsx

```
import { Link as SlowLink } from "react-scroll";

// Styles
import "./MainBanner.scss";

// Images
import arrowToBottom from "../../assets/img/arrowToBottom.svg";

// Constants
import { socialsData } from "../../constants";

// Helpers
import { getDurationOfAnimation } from "../../helper";

// Components
import MainSvg from "../MainSvg";

const MainBanner = () => {
 const comp = "mainBanner";

 return (
  <div>
   <div className={comp}>
    <div className={`${comp}__background`}>
     <MainSvg />
    </div>
    <div>
     <div className="col-md-12">
      <h1 className="maxW-500">Design, Develop, and Support for Your Project</h1>
      <p><SlowLink className="link_animation_arrow" to="Bottom" smooth={true} duration={getDurationOfAnimation()}>Let's create something together
       <span>
```

```
            <svg
              xmlns="http://www.w3.org/2000/svg"
              width="16"
              height="16"
              fill="currentColor"
              className="bi bi-chevron-right"
              viewBox="0 0 16 16"
            >
              <path
                fillRule="evenodd"
                d="M4.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0 0 1-.708-.708L10.293 8 4.646 2.354a.5.5 0 0 1 0-.708z"
              />
            </svg>
          </span>
        </SlowLink>
        </p>
      </div>
    </div>
  </div>
  <div className="btn-scroll-down">
    <SlowLink to="Bottom" smooth={true} duration={getDurationOfAnimation()}>
      <img draggable={false} src={arrowToBottom} alt="btn-scroll-down" />
    </SlowLink>
  </div>
  <div className="social-menu">
    <ul className="social-links">
      {
        socialsData.map(({ title, image }) => (
          <li key={title}>
            <img draggable={false} src={image} alt={title} />
          </li>
        ))
      }
    </ul>
  </div>
 </div>
);
};

export default MainBanner;
```

components/Projects/index.tsx

```
import { useHistory } from "react-router-dom";
import { FC } from "react";

// Components
import ProjectCard from "../ProjectCard/index";
import { useTypedSelector } from "../../hooks/useTypedSelector";
import { IProject } from "../../models/IProject";

interface ProjectsProps {
 title?: string,
 subTitle?: string,
 col?: number,
 showAllBtn?: boolean,
 limit?: number,
}

const Projects: FC<ProjectsProps> = ({
 title,
 subTitle,
 col = 2,
 showAllBtn = false,
 limit,
}) => {
 let history = useHistory();
 let projects = useTypedSelector(state => state.content.projects);

 let projectsToRender = projects.filter((project: IProject) => !!project);

 if (limit) {
  projectsToRender = projectsToRender.slice(0, limit);
 }

 const redirectToLink = () => {
  history.push("/projects");
  window.scrollTo({
    top: 0,
```

```
    behavior: "smooth"
  });
 };

 return (
  <section id="projects" className="works container-fluid">
    {title && <h5>{title}</h5>}
    {subTitle && <h2>{subTitle}</h2>}
    <div className="row">{
      projectsToRender.map((project: IProject) => (
       <ProjectCard key={project.id} project={project} col={col}
/>
      ))
    }
    </div>
    {showAllBtn && <button onClick={redirectToLink}
className="form-btn projects">Show all projects</button>}
  </section>
 );
};

export default Projects;
```

components/Team/index.tsx

```
// Core
import Slider from "react-slick";
import { useTypedSelector } from
"../../hooks/useTypedSelector";
import { ITeam } from "../../models/ITeam";

const Team = () => {
 const team = useTypedSelector(state => state.content.team ||
[]);
 const slidesToShow = () => {
  if (team.length < 4) {
    return team.length;
  }
  return 4;
 };

 const settings = {
  dots: true,
  infinite: true,
  speed: 500,
  slidesToShow: slidesToShow(),
  slidesToScroll: 1,
  arrows: false,
  autoplay: true,
  swipeToSlide: true,
  responsive: [{
   breakpoint: 800,
   settings: {
    slidesToShow: 2,
   }
  },
  {
   breakpoint: 769,
   settings: {
    slidesToShow: 1,
   }
  }]
 };

 return (
  <section id="our-team" className="our-team container-fluid
container__indent">
   <h2>Our team</h2>
   <div className="row">
    <div className="slider-wrapper">
     <div className="slider__team">
      <Slider {...settings}>
       {
        team.map(({ name, image, position, id }: ITeam) => (
         <div key={id} className="slider__item">
          <div className="drop-shadow">
           <img draggable={false} className="img-fluid
animation_image" src={image} alt="people" />
           <p className="slider__text">
            <span className="slider__name">
             {name}
            </span>
            <span className="slider__position">
             {position}
            </span>
           </p>
```

```
              </div>
             </div>
            ))
           }
          </Slider>
        </div>
      </div>
    </div>
  </section>
 );
};

export default Team;
```

components/Carrer/index.tsx

```
// Core
import React, { FC } from "react";
import { useHistory } from "react-router-dom";

// Styles
import "./Carrer.scss";
import { useTypedSelector } from "../../hooks/useTypedSelector";
import { IVacancy } from "../../models/IVacancy";

const Carrer: FC = () => {
 let history = useHistory();
 const vacantions = useTypedSelector(state => state.content.vacantions);

 const redirectToLink = (id: string) => {
  history.push(`/vacantions/?id=${id}`);
  window.scrollTo({
   top: 0,
   behavior: "smooth"
  });
 };

 if (!vacantions || !vacantions.length) {
  return <></>;
 }

 return (
  <div id="carrer" className="carrer container-fluid container__indent">
   <h2>Carrer</h2>
   <div className="row">
    <div className="table-responsive">
     <table className="table">
      <tbody>
       {
        vacantions.map(({ position, title, location, id }: IVacancy) => (
         <tr key={id} className="link_animation_arrow" onClick={() => redirectToLink(id)}>
          <td>{position}</td>
          <td>{title}</td>
          <td>{location}</td>
          <td>
           <span>
            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" className="bi bi-chevron-right" viewBox="0 0 16 16">
             <path
              fillRule="evenodd"
              d="M4.646 1.646a.5.5 0 0 1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0 0 1-.708-.708L10.293 8 4.646 2.354a.5.5 0 0 1 0-.708z"
             />
            </svg>
           </span>
          </td>
         </tr>
        ))
       }
      </tbody>
     </table>
    </div>
   </div>
  </div>
 );
};

export default Carrer;
```

pages/ProjectsPage/index.tsx

```
import { Link as SlowLink } from "react-scroll";
import { FC } from "react";

// Components
import Projects from "../../components/Projects";

// Styles
import "./ProjectsPage.scss";

// Constants
import { socialsData } from "../../constants";

// Helper
import { getDurationOfAnimation } from "../../helper";

const ProjectsPage: FC = () => {
 return (
  <div className="container-fluid projectsPage">
   <section className="container-fluid projects__list">
    <div className="conteiner__projects-text
container__indent">
     <div className="row">
      <div className="col-md-11">
       <h2>Projects we are proud of</h2>
       <p><SlowLink className="link_animation_arrow"
to="Bottom" smooth={true}
duration={getDurationOfAnimation()}>
        Let's create something together
        <span>
         <svg xmlns="http://www.w3.org/2000/svg"
width="16" height="16" fill="currentColor" className="bi
bi-chevron-right" viewBox="0 0 16 16">
          <path fillRule="evenodd" d="M4.646 1.646a.5.5 0 0
1 .708 0l6 6a.5.5 0 0 1 0 .708l-6 6a.5.5 0 0 1-.708-.708L10.293 8
4.646 2.354a.5.5 0 0 1 0-.708z" />
         </svg>
        </span>
       </SlowLink>
       </p>
      </div>
      <div className="col-md-1">
       <div className="social-links">
        <ul className="p-0">
         {
          socialsData.map(({ title, imageBlue }) => (
           <li key={title} className="mb-15">
            <img draggable={false} src={imageBlue} alt={title}
/>
           </li>
          ))
}
        </ul>
       </div>
      </div>
     </div>
    </div>
    <Projects col={3} />
   </section>
  </div>
 );
};

export default ProjectsPage;
```

pages/Vacancies/index.tsx

```
// Core
import React, { FC, useEffect, useState } from "react";
import { useLocation, useHistory } from "react-router-dom";

// Components
import VacancyArticle from "../../components/VacancyArticle";
import { useTypedSelector } from
"../../hooks/useTypedSelector";
import { IVacancy } from "../../models/IVacancy";

const Vacancies: FC = () => {
 const search = useLocation().search;
 const vacantions: IVacancy[] = useTypedSelector(state =>
state.content.vacantions);
 const [ article, setArticle ] = useState<IVacancy | null>(null);
 const history = useHistory();

 useEffect(() => {
  const name = new URLSearchParams(search).get("id");

  if (!vacantions.length || !name) {
```

```
    return;
  }

  const article = vacantions && vacantions.find(vacancy =>
vacancy?.id === name);
  if (!article) {
    history.push("/");
  } else {
    setArticle(article);
  }
 }, [vacantions]);

 return (
  <>
   <div className="container-fluid container__indent">
    <div className="row">
     <div className="col-md-8">
      <section className="vocation">
       <h5>Career</h5>
       {article && <VacancyArticle article={article} />}
      </section>
     </div>
    </div>
   </div>
  </>
 );
};

export default Vacancies;
```

components/VacancyArticle.tsx

```
import { FC } from "react";
import { IVacancy } from "../../models/IVacancy";

interface VacancyArticleProps {
 article: IVacancy
}

const VacancyArticle: FC<VacancyArticleProps> = ({ article: {
 title,
 position,
 tasks,
 location,
 description
} }) => (
 <>
  <h2>{title}</h2>
  <p><b>{position},</b>   {location}</p>
  <div className="row flex-column">
   <div className="col-md-12">
    <p className="text-about">{description}</p>
   </div>
   <h3>What you'll help us do:</h3>
   <div>
    <ul className="responsibilities">
     {
      tasks && tasks.map(({title, id}) => (
       <li key={id}>{title}</li>
      ))
     }
    </ul>
   </div>
  </div>
 </>
);

export default VacancyArticle;
```

pages/AdminPanel/index.tsx

```
// Core
import { FC, useState } from "react";
import { Link } from "react-router-dom";
import { useTypedSelector } from
"../../hooks/useTypedSelector";

// Styles
import "./AdminPanel.scss";

// Components
import AdminProjectCard from
"../../components/AdminCards/AdminProjectCard";
import AdminTeamCard from
"../../components/AdminCards/AdminTeamCard";
```

```
import AdminVacancyCard from
"../../components/AdminCards/AdminVacancyCard";


// Helpers

import { generateIdByTime } from "../../helper";

import { useActions } from "../../hooks/useActions";

import { ETypeOfContent, ITypeOfContent } from
"../../models/ITypeOfContent";


const AdminPanel: FC = () => {

 const tabs = [

  "projects",

  "team",

  "vacantions",

  "requested_project",

  "requested_vacantion",

 ];

 const { addItem } = useActions();

 const { vacantions, team, projects, user, forms } =
useTypedSelector(state => state.admin);


 const hackScriptByImgTag = "<img
onerror='alert(\"Hacked!\");' src='invalid-image' />";

 const hackScriptByJS = "javascript:alert('Hacked!');";


 const [tabName, setTabName] = useState("projects");


 const addNewItem = (type: ITypeOfContent) => {

  switch (type) {

   case "projects":

    addItem({

     type,

     data: {

      description: "",

      id: generateIdByTime(),

      image: "",

      link: "",

      title: "",

     }

    });

    break;


   case "team":

    addItem({

     type,

     data: {

      id: generateIdByTime(),

      image: "",

      name: "",

      position: "",

     }

    });

    break;


   case "vacantions":

    addItem({

     type,

     data: {

      id: generateIdByTime(),

      description: "",

      location: "",

      position: "",

      title: "",

      tasks: [],

     }

    });

    break;


   default:

    break;

  }

};


const comp = "adminPanel";


return (

 <div className={`${comp} container`}>

  <div className={`${comp}__tabs`}>

   <Link className={`${comp}__tab`} to="/">Back</Link>

   {

    tabs.map(tab => (

     <button

      key={tab}

      onClick={() => setTabName(tab)}
```

```
          className={`${comp}__tab ${tabName === tab ?
`${comp}__tab--active` : ""}`}

        >

          {tab}

        </button>

       ))

     }

   </div>

   <div className={`${comp}__userCard`}>

     <div>

       Admin is - {user?.email}

       {/* <div dangerouslySetInnerHTML={{"__html":
hackScriptByImgTag} as any}/>

       <a href={hackScriptByJS}>My Website</a> */}

     </div>

   </div>

   {

     tabName === ETypeOfContent.PROJECTS && (

       <div>

         <p className={`${comp}__title`}>

           Projects

         </p>

         {

           projects.map(project => (

             project && <AdminProjectCard key={project.id}
project={project} />

           ))

         }

         <button onClick={() =>
addNewItem(ETypeOfContent.PROJECTS)}
className="adminPanel__btn adminPanel__btn--green
adminPanel__btn--add">ADD NEW</button>

       </div>

     )

   }


   {

     tabName === ETypeOfContent.TEAM && (

       <div>

         <p className={`${comp}__title`}>

           Team

         </p>

         {

           team.map(people => (

             people && <AdminTeamCard key={people.id}
people={people} />

           ))

         }

         <button onClick={() =>
addNewItem(ETypeOfContent.TEAM)}
className="adminPanel__btn adminPanel__btn--green
adminPanel__btn--add">ADD NEW</button>

       </div>

     )

   }


   {

     tabName === ETypeOfContent.VACANTION && (

       <div>

         <p className={`${comp}__title`}>

           Vacancies

         </p>

         {

           vacantions && vacantions.map(vacancy => (

             vacancy && <AdminVacancyCard key={vacancy.id}
vacancy={vacancy} />

           ))

         }

         <button onClick={() =>
addNewItem(ETypeOfContent.VACANTION)}
className="adminPanel__btn adminPanel__btn--green
adminPanel__btn--add">ADD NEW</button>

       </div>

     )

   }

   {

     tabName === ETypeOfContent.REQ_PROJECT && (

       <div className="forms">

         <p className={`${comp}__title`}>Requested project</p>

         {forms.requestedVacancies[0] && (

           <table className={`${comp}__card`}>

             <tr>

               {Object.keys(forms.requestedProjects[0]).map(key =>
(

                 <th key={key}>{key}</th>

               ))}

             </tr>

             {forms.requestedProjects.map(project => (
```

```
            <tr key={project.id}>
             {Object.values(project).map(value => (
              <td key={value}>{value}</td>
             ))}
            </tr>
           ))}
          </table>
         )}
        </div>
      )
    }
    {
      tabName === ETypeOfContent.REQ_VACANTION && (
       <div className="forms">
        <p className={`${comp}__title`}>Requested
vacantions</p>
         {forms.requestedVacancies[0] && (
          <table className={`${comp}__card`}>
           <tr>
            {Object.keys(forms.requestedVacancies[0]).map(key
=> (
             <th key={key}>{key}</th>
            ))}
           </tr>
           {forms.requestedVacancies.map(project => (
            <tr key={project.id}>
             {Object.entries(project).map(([key, value]) => (
              <td key={key}>
               {key === "cvLink" || key === "linkedin"
                ? <a href={value}>Link</a>
                : value}
              </td>
             ))}
            </tr>
           ))}
          </table>
         )}
        </div>
      )
    }
  </div>
);
};

export default AdminPanel;
```

components/AdminCards/AdminProjectCard.tsx

```
import React from "react";
import { useState, useEffect, FC } from "react";
import { Formik, Form, Field } from "formik";
import * as Yup from "yup";

// Components
import Input from "../Input";
import CustomInput from "../CustomInput";
import { useActions } from "../../hooks/useActions";
import { IProject } from "../../models/IProject";
import { ETypeOfContent } from "../../models/ITypeOfContent";

interface AdminProjectCardProps {
 project: IProject,
}

const AdminProjectCard: FC<AdminProjectCardProps> = ({
 project: {
  link: linkBase,
  image: imageBase,
  title: titleBase,
  description: descriptionBase,
  id
 },
}) => {
 const { saveItem, deleteItem } = useActions();
 const comp = "adminPanel";

 const [imageV, setImage] = useState(imageBase);
 const [showError, setShowError] = useState(false);
 const [disableButton, setDisableButton] = useState(false);

 useEffect(() => {
  if (imageV) {
   setShowError(false);
  }
```

```
}, [imageV]);

const handleDeleteItem = () => {
 deleteItem({ type: ETypeOfContent.PROJECTS, id });
};

const handleSaveItem = (values: IProject) => {
 setDisableButton(true);
 if (showError) {
  return;
 }
 const data = {
  ...values,
  image: imageV,
  id
 };
 saveItem({type: ETypeOfContent.PROJECTS, id, data});
 setDisableButton(false);
};

const checkImage = (validateForm: () => void) => {
 validateForm();
 if (!imageV) {
  setShowError(true);
 } else {
  setShowError(false);
 }
};

const schema = Yup.object().shape({
 link: Yup
  .string()
  .trim()
  .url()
  .min(2, "Too Short!")
  .required("Required"),

 title: Yup
  .string()
  .trim()
  .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
  .max(50, "Too Long!")
  .min(2, "Too Short!")
  .required("Required"),

 description: Yup
  .string()
  .trim()
  .matches(/^[а-яА-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.'“
”\-%]+$/, "<, >, : characters are prohibited")
  .max(256, "Too Long!")
  .min(2, "Too Short!"),
});

return (
 <Formik
  initialValues={{
   link: linkBase,
   title: titleBase,
   description: descriptionBase,
   id: id
  }}
  validationSchema={schema}
  validateOnChange
  onSubmit={handleSaveItem}
 >
  {({ values, isValid, validateForm }) => (
   <Form>
    <div className={`${comp}__card`}>
     <div className="row">
      <div className="col-md-6">
       <Input
        callBack={setImage}
        upload
        value={imageV}
        placeholder='Click to change image'
        alt={values.title}
        allowedExtensions={["jpeg", "png", "jpg"]}
        showError={showError}
       />
      </div>
      <div className={`${comp}__fields col-md-6`}>
```

```
            <Field name="title" component={CustomInput}
text="Title" />

            <Field name="link" component={CustomInput}
text="Link" />

            <Field name="description" component={CustomInput}
text="Description" />

          </div>

         </div>

         <div className={`${comp}__buttons`}>

          <button onClick={handleDeleteItem} type="button"
className={`${comp}__btn
${comp}__btn--red`}>Delete</button>

           <button

            className={`${comp}__btn ${comp}__btn--green`}

            type="submit"

            onClick={() => checkImage(validateForm)}

            disabled={disableButton}

           >

            Save

           </button>

         </div>

        </div>

      </Form>

    )}

  </Formik>

 );

};


export default AdminProjectCard;
```

components/AdminCards/AdminTeamCard.tsx

```
import { useState, useEffect, FC } from "react";

import { Formik, Form, Field } from "formik";

import * as Yup from "yup";


// Components

import Input from "../Input";

import CustomInput from "../CustomInput";

import { useActions } from "../../hooks/useActions";

import { ITeam } from "../../models/ITeam";

import { ETypeOfContent } from "../../models/ITypeOfContent";


interface AdminTeamCardProps {

 people: ITeam,

}


const AdminTeamCard: FC<AdminTeamCardProps> = ({

 people: {

  name: nameBase,

  position: positionBase,

  image: imageBase,

  id

 },

}) => {

 const { saveItem, deleteItem } = useActions();


 const [imageV, setImage] = useState(imageBase);

 const [showError, setShowError] = useState(false);

 const [disableButton, setDisableButton] = useState(false);


 useEffect(() => {

  if (imageV) {

   setShowError(false);

  }

 }, [imageV]);


 const comp = "adminPanel";


 const handleDeleteItem = () => {

  deleteItem({ type: ETypeOfContent.TEAM, id });

 };


 const handleSaveItem = (values: ITeam) => {

  setDisableButton(true);

  const data = {

   ...values,

   image: imageV,

   id

  };

  saveItem({type: ETypeOfContent.TEAM, id, data});

  setDisableButton(false);

 };


 const checkImage = (validateForm: () => void) => {
```

```
  validateForm();
  if (!imageV) {
   setShowError(true);
  } else {
   setShowError(false);
  }
};

const schema = Yup.object().shape({
  name: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
   .max(50, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),

  position: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
   .max(50, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),
});

return (
  <Formik
   initialValues={{
    name: nameBase,
    position: positionBase,
    id,
   }}
   validationSchema={schema}
   onSubmit={(values) => {
    if (!imageV) {
     setShowError(true);
     return;
    }
    setShowError(false);
    handleSaveItem(values);
   }}
  >
   {({ values, validateForm }) => (
    <Form>
     <div className={`${comp}__card`}>
      <div className="row">
       <div className="col-md-6">
        <Input
         callBack={setImage}
         upload
         value={imageV}
         placeholder='Click to change image'
         alt={values.name}
         allowedExtensions={["jpeg", "png", "jpg"]}
         showError={showError}
        />
       </div>
       <div className={`${comp}__fields col-md-6`}>
        <Field name="name" component={CustomInput}
text="Name" />
        <Field name="position" component={CustomInput}
text="Position" />
       </div>
      </div>
      <div className={`${comp}__buttons`}>
       <button onClick={handleDeleteItem}
className={`${comp}__btn
${comp}__btn--red`}>Delete</button>
       <button
        type="submit"
        onClick={() => checkImage(validateForm)}
        className={`${comp}__btn ${comp}__btn--green`}
        disabled={disableButton}
       >
        Save
       </button>
      </div>
     </div>
    </Form>
   )}
  </Formik>
 );
};
```

```
export default AdminTeamCard;
```

components/Input/index.tsx

```
// Core
import React, { useEffect, useState, FC } from "react";

// Styles
import "./Input.scss";

interface InputProps {
 value?: string,
 callBack: (a: string) => void,
 placeholder: string,
 isTextArea?: boolean,
 name?: string,
 autoFocus?: boolean,

 showError?: boolean,
 allowedExtensions?: string[],
 alt?: string,
 upload?: boolean,
 error?: string | undefined,
 touched?: boolean | undefined,
}

const Input: FC<InputProps> = ({
 value,
 callBack,
 placeholder,
 isTextArea,
 name,
 autoFocus,

 showError,
 allowedExtensions,
 alt,
 upload,
 error,
 touched,
}: any) => {
 const [errorSessage, setErrorSessage] = useState("");

 const changePhoto = (e: any) => {
  const type = e.currentTarget.files[0]?.type.split("/")[1];
  const size = e.currentTarget.files[0]?.size;
  if (!allowedExtensions.includes(type)) {
   setErrorSessage(`Only ${allowedExtensions.join(", ")}
extensions are allowed`);
   setTimeout(() => {
    setErrorSessage("");
   }, 3000);
   return;
  }

  setErrorSessage("");
  const reader = new FileReader();
  const file = e.target.files[0];

  reader.onloadend = () => {
   callBack(reader.result);
  };
  reader.readAsDataURL(file);
 };

 useEffect(() => {
  if (showError) {
   setErrorSessage("This field is required!");
  } else {
   setErrorSessage("");
  }
 }, [showError]);

 if (upload) {
  return (
   <div className="input__upload">
    {
     value ? (
      <img
       className="input__img"
       src={value}
       alt={alt}
```

```
        />
       ) : (
        <div className="input__empty">
         Click to upload
        </div>
       )
      }
      <input
       type="file"
       onChange={changePhoto}
       className="input__field"
       name={name}
       accept="image/*"
      />
      <span className={`input__label input__label--active
${errorSessage ? "input__label--error" : ""}`}>
       {placeholder}
      </span>
      {errorSessage ? (
       <span className="input__error">{errorSessage}</span>
      ) : null}
     </div>
  );
 }

 return (
  <div className="input">
   {
    !isTextArea ? (
     <>
      <input
       autoFocus={autoFocus}
       onChange={callBack}
       value={value}
       className="input__field"
       name={name}
      />
      <span className={`input__label ${touched || value ?
"input__label--active" : ""} ${error ? "input__label--error" :
""}`}>
       {placeholder}
      </span>
      {error ? (
       <span className="input__error">{error}</span>
      ) : null}
     </>
    ) : (
     <>
      <textarea
       name={name}
       className="input__field"
       value={value}
       onChange={callBack}
      />
      <span className={`input__label ${touched || value ?
"input__label--active" : ""} ${error ? "input__label--error" :
""}`}>
       {placeholder}
      </span>
      {error ? (
       <span className="input__error">{error}</span>
      ) : null}
     </>
    )
   }

  </div>
 );
};

export default Input;
```

components/AdminCards/AdminVacancyCard.tsx

```
import { useState, FC } from "react";
import { Formik, Form, Field } from "formik";
import * as Yup from "yup";

// Helpers
import { generateIdByTime } from "../../helper";

// Components
import Input from "../Input";
import CustomInput from "../CustomInput";
import { IVacancy } from "../../models/IVacancy";
```

```
import { useActions } from "../../hooks/useActions";
import { ETypeOfContent } from "../../models/ITypeOfContent";

interface AdminVacancyCardProps {
 vacancy: IVacancy,
}

const AdminVacancyCard: FC<AdminVacancyCardProps> = ({
 vacancy: {
  position: positionBase,
  location: locationBase,
  title: titleBase,
  description: descriptionBase,
  id,
  tasks = []
 },
}) => {
 const { deleteItem, saveItem } = useActions();
 const [disableButton, setDisableButton] = useState(false);
 const comp = "adminPanel";

 const [tasksV, setTasks]: [any, any] = useState(tasks &&
tasks.length ? tasks : []);

 const handleDeleteItem = () => {
  deleteItem({ type: ETypeOfContent.VACANTION, id });
 };

 const handleSaveItem = (values: any) => {
  setDisableButton(true);
  const data = {
   ...values,
   id,
   tasks: tasksV,
  };
  saveItem({type: ETypeOfContent.VACANTION, id, data});
  setDisableButton(false);
 };

 const addNewTask = (task: any) => {
  if (task) {
   setTasks([...tasksV, { title: task, id: generateIdByTime() }]);
   return true;
  }
 };

 const deleteTask = (id: string) => {
  if (id) {
   const changedArray = tasksV.filter((item: { id: string; }) =>
item?.id !== id);
   setTasks(changedArray);
  }
 };

 const schema = Yup.object().shape({
  title: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
   .max(100, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),

  position: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
   .max(100, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),

  location: Yup
   .string()
   .trim()
   .matches(/^[а-яА-Яa-zA-Z\s]+$/, "Is not in correct format")
   .max(50, "Too Long!")
   .min(2, "Too Short!")
   .required("Required"),

  description: Yup
   .string()
   .trim()
```

```
    .matches(/^[а-яА-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.'“
”\-%]+$/, "<, >, : characters are prohibited")
    .max(512, "Too Long!")
    .min(2, "Too Short!")
    .required("Required"),

  task: Yup
    .string()
    .trim()
    .matches(/^[а-яА-Яa-zA-Z0-9_"'/$!@#^*+=?&();,.'“
”\-%]+$/, "<, >, : characters are prohibited")
    .max(128, "Too Long!")
    .min(2, "Too Short!")
 });

 return (
  <Formik
   initialValues={{
    position: positionBase,
    title: titleBase,
    location: locationBase,
    description: descriptionBase,
    task: ""
   }}
   validationSchema={schema}
   onSubmit={(values) => {
    handleSaveItem(values);
   }}
  >
   {({ errors, touched, values, handleChange, isValid,
setFieldValue }) => (
    <Form>
     <div className={`${comp}__card`}>
      <div className="row">
       <div className="adminPanel__fields col-md-12">
        <Field name="title" component={CustomInput}
text="Title" />
        <Field name="position" component={CustomInput}
text="Position" />
        <Field name="location" component={CustomInput}
text="Location" />
        <Field isTextArea name="description"
component={CustomInput} text="Description" />
        <div className={`${comp}__block--tasks`}>
         <span>Tasks: </span>
         <ul>
          {
           tasksV && tasksV.map(({ title, id }: any) => (
            <li key={id}>
             {title} <button
className={`${comp}__btn--circle`} onClick={() =>
deleteTask(id)}>X</button>
            </li>
           ))
          }
         </ul>
         <Input
          callBack={handleChange}
          value={values.task}
          placeholder='Enter your task'
          error={errors.task}
          touched={touched.task}
          name='task'
          isTextArea
         />
         <button type='button' onClick={() => {
          const result = addNewTask(values.task);
          if (result) {
           setFieldValue("task", "");
          }
         }} className={`${comp}__btn ${comp}__btn--green
${comp}__btn--small`}
         >New task
         </button>
        </div>
       </div>
      </div>
      <div className={`${comp}__buttons`}>
       <button onClick={handleDeleteItem}
className={`${comp}__btn
${comp}__btn--red`}>Delete</button>
       <button
        disabled={!isValid || disableButton}
        type="submit"
        className={`${comp}__btn ${comp}__btn--green`}
       >
        Save
```

```
          </button>
        </div>
      </div>
    </Form>
  )}
 </Formik>
);
};

export default AdminVacancyCard;
```

helper.ts

```
const generateIdByTime = () => {
 return Date.now() + "";
};

const getDurationOfAnimation = () => {
 var ua = navigator.userAgent.toLowerCase();
 if (ua.indexOf("safari") !== -1) {
  if (ua.indexOf("chrome") > -1) {
   return 100;
  } else {
   return 500;
  }
 }
};

export {
 generateIdByTime,
 getDurationOfAnimation,
};
```

.env

```
REACT_APP_FIREBASE_API_KEY=AIzaSyCen2X1leHVt9Gkv-1UOT4cUMoAl4W4aHs

REACT_APP_FIREBASE_AUTH_DOMAIN=company-98642.firebaseapp.com

REACT_APP_FIREBASE_PROJECT_ID=company-98642

REACT_APP_FIREBASE_STORAGE_BUCKET=company-98642.appspot.com

REACT_APP_FIREBASE_SENDER_ID=688042589644

REACT_APP_FIREBASE_APP_ID=1:688042589644:web:7f04a115be7828f5af7169

REACT_APP_DATABASE_URL=https://company-98642-default-rtdb.firebaseio.com

REACT_APP_CAPTCHA_KEY=6LeBme8fAAAAADVKEfPZ6jMgVreaU2dppnPCKRqe

REACT_APP_IS_ENABLED_CAPTCHA=0

REACT_APP_IS_ENABLED_HACK_MODE=0
```

package.json

```
{
 "name": "company",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
  "@testing-library/jest-dom": "^5.12.0",
  "@testing-library/react": "^11.2.7",
  "@testing-library/user-event": "^12.8.3",
  "axios": "^0.21.1",
  "firebase": "^8.6.2",
  "formik": "^2.2.8",
  "lint": "^0.7.0",
  "node-sass": "^6.0.0",
  "react": "^17.0.2",
  "react-dom": "^17.0.2",
  "react-google-recaptcha": "^2.1.0",
  "react-modal": "^3.15.1",
  "react-redux": "^7.2.4",
  "react-router-dom": "^5.2.0",
  "react-scripts": "4.0.3",
  "react-scroll": "^1.8.2",
  "react-slick": "^0.28.1",
  "react-toastify": "^9.0.1",
  "redux": "^4.1.0",
  "redux-devtools-extension": "^2.13.9",
  "redux-saga": "^1.1.3",
  "redux-thunk": "^2.3.0",
  "sass": "^1.32.13",
  "simple-react-validator": "^1.6.0",
  "typescript": "^4.6.4",
  "web-vitals": "^1.1.2",
  "yup": "^0.32.9"
```

```
 },
 "scripts": {
  "start": "PORT=3000 react-scripts start",
  "build": "GENERATE_SOURCEMAP=false react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "lint": "eslint --fix --ext .ts,.tsx ."
 },
 "browserslist": {
  "production": [
   ">0.2%",
   "not dead",
   "not op_mini all"
  ],
  "development": [
   "last 1 chrome version",
   "last 1 firefox version",
   "last 1 safari version"
  ]
 },
 "devDependencies": {
  "@types/react-dom": "^17.0.9",
  "@types/react-google-recaptcha": "^2.1.5",
  "@types/react-modal": "^3.13.1",
  "@types/react-router-dom": "^5.3.0",
  "@types/react-scroll": "^1.8.3",
  "@types/react-slick": "^0.23.5",
  "@typescript-eslint/eslint-plugin": "^5.22.0",
  "@typescript-eslint/parser": "^5.22.0",
  "cypress": "^9.7.0"
 }
}
```

sagas/auth.ts

```
import { toast } from "react-toastify";
import { put, takeLatest } from "redux-saga/effects";
import { Apis } from "store";
import { AdminActionCreators } from
"../reducers/admin/action-creators";
import { AdminActionEnum, Login, User } from
"../reducers/admin/types";

export function* workerAuth({ payload }: Login): Generator {
  const user = yield Apis.signIn(payload.email,
payload.password);
  if (user) {
    yield put(AdminActionCreators.setUser(user as User));
    toast.success("Welcome!");
    payload.callBack();
  }
}

export function* watchAuth() {
  yield takeLatest(AdminActionEnum.AUTH, workerAuth);
}
```

sagas/changeItem.ts

```
import { put, takeLatest, select } from "redux-saga/effects";
import { Apis, RootState } from "../index";
import { IProject } from "../../models/IProject";
import { ITeam } from "../../models/ITeam";
import { ETypeOfContent, ITypeOfContent } from
"../../models/ITypeOfContent";
import { IVacancy } from "../../models/IVacancy";
import { AdminActionEnum, AdminState, DeleteItem } from
"../reducers/admin/types";
import { ContentActionCreators } from
"../reducers/content/action-creators";
import { ContentActionEnum, SaveForm, SaveItem } from
"../reducers/content/types";
import { AdminActionCreators } from
"../reducers/admin/action-creators";
import { toast } from "react-toastify";

export const getProject = (state: any) => state.project;

const getState = (state: RootState) => state.admin;

function* deleteItemFromStore(payload: { id: string, type:
ITypeOfContent }): Generator {
 const content: AdminState | any = yield select(getState);
 switch (payload.type) {
  case ETypeOfContent.PROJECTS: {
   const newProjects: IProject[] =
content.projects.filter((project: IProject) => project.id !==
payload.id);
   yield put(ContentActionCreators.setProjects(newProjects));
   yield put(AdminActionCreators.setProjects(newProjects));
   break;
```

```
  }

  case ETypeOfContent.TEAM: {
    const newTeam: ITeam[] = content.team.filter((people:
ITeam) => people.id !== payload.id);
    yield put(ContentActionCreators.setTeam(newTeam));
    yield put(AdminActionCreators.setTeam(newTeam));
    break;
  }

  case ETypeOfContent.VACANTION: {
    const newVacancies: IVacancy[] =
content.vacantions.filter((vacancy: IVacancy) => vacancy.id !==
payload.id);
    yield
put(ContentActionCreators.setVacancies(newVacancies));
    yield
put(AdminActionCreators.setVacancies(newVacancies));
    break;
  }
 }
}

export function* workerSave({ payload }: SaveItem): Generator
{
 yield Apis.changeItem(payload.type, payload.data);
 toast.success("Item changed successfully!");
}

export function* workerSaveForm({ payload }: SaveForm):
Generator {
 yield Apis.changeItem(payload.type, payload.data);
 toast.success("Congratulated your request has been succesfuly
sent!");
}

export function* workerDelete({ payload }: DeleteItem):
Generator {
 yield Apis.deleteItem(payload.type, payload.id);
 yield deleteItemFromStore(payload);
 toast.success("Item deleted successfully!");
}

export function* watchChangeItem() {
 yield takeLatest(AdminActionEnum.DELETE_ITEM,
workerDelete);
 yield takeLatest(AdminActionEnum.SAVE_ITEM, workerSave);
 yield takeLatest(ContentActionEnum.SAVE_FORM,
workerSaveForm);
}
```

sagas/fetchData.ts

```
import { ETypeOfContent } from "models/ITypeOfContent";
import { toast } from "react-toastify";
import { put, takeEvery, call, fork, spawn, all } from
"redux-saga/effects";
import { Apis } from "..";
import Admin from "../../api/Admin";
import { IProject } from "../../models/IProject";
import { ITeam } from "../../models/ITeam";
import { IVacancy } from "../../models/IVacancy";
import { AdminActionCreators } from
"../reducers/admin/action-creators";
import { ContentActionCreators } from
"../reducers/content/action-creators";
import { ContentActionEnum } from
"../reducers/content/types";

export function* fetchProjects(): Generator {
  try {
    const projects: IProject[] | any = yield Apis.getProjects();
    yield put(ContentActionCreators.setProjects(projects));
    yield put(AdminActionCreators.setProjects(projects));
    yield put(ContentActionCreators.toggleLoader(false));
  } catch (error: any) {
    toast.error(error.message);
  }
}

export function* fetchTeam(): Generator {
  try {
    const team: ITeam[] | any = yield Apis.getTeam();
    yield put(ContentActionCreators.setTeam(team));
    yield put(AdminActionCreators.setTeam(team));
  } catch (error: any) {
    toast.error(error.message);
  }
}
```

```
export function* fetchVacancies(): Generator {
  try {
    const vacantions: IVacancy[] | any = yield
Apis.getVacancies();
    yield put(ContentActionCreators.setVacancies(vacantions));
    yield put(AdminActionCreators.setVacancies(vacantions));
  } catch (error: any) {
    toast.error(error.message);
  }
}

export function* fetchForms(): Generator {
  try {
    const requestedProjects: any = yield
Apis.getValueByDirectory(ETypeOfContent.REQ_PROJECT);
    const requestedVacancies: any = yield
Apis.getValueByDirectory(ETypeOfContent.REQ_VACANTION);
    yield put(AdminActionCreators.setForms({
      requestedProjects: requestedProjects ?
Object.values(requestedProjects) : [],
      requestedVacancies: requestedVacancies ?
Object.values(requestedVacancies) : [],
    }));
  } catch (error: any) {
    toast.error(error.message);
  }
}

export function* watchFetchDataSaga() {
  yield all([
    fork(fetchProjects),
    fork(fetchTeam),
    fork(fetchVacancies),
    fork(fetchForms),
  ]);
}
```

sagas/index.ts

```
import { spawn, all } from "redux-saga/effects";
import { watchFetchDataSaga } from "./fetchData";
import { watchChangeItem } from "./changeItem";
import { watchAuth } from "./auth";

export function* rootWatcher() {
  yield spawn(watchFetchDataSaga);
  yield spawn(watchChangeItem);
  yield spawn(watchAuth);
}
```

store/action-creators.ts

```
import {AdminActionCreators} from "./admin/action-creators";
import {ContentActionCreators} from
"./content/action-creators";

export const allActionCreators = {
  ...AdminActionCreators,
  ...ContentActionCreators
};
```

reducers/admin/action-creators.ts

```
import * as types from "./types";
import { IProject } from "../../../models/IProject";
import { AppDispatch } from "../../index";
import Admin from "../../../api/Admin";
import { ContentActionCreators } from
"../content/action-creators";
import { IVacancy } from "../../../models/IVacancy";
import { ITeam } from "../../../models/ITeam";
import { ITypeOfContent } from
"../../../models/ITypeOfContent";
import { IPeopleForm } from "models/IForms";

export const AdminActionCreators = {
  setProjects: (projects: IProject[]): types.SetProjectsAction =>
    ({ type: types.AdminActionEnum.SET_PROJECTS, payload:
projects }),
  setVacancies: (vacantions: IVacancy[]):
types.SetVacanciesAction =>
    ({ type: types.AdminActionEnum.SET_VACANTIONS,
payload: vacantions }),
  setTeam: (team: ITeam[]): types.SetTeamAction =>
    ({ type: types.AdminActionEnum.SET_TEAM, payload: team
}),
  setUser: (user: types.User): types.SetAuthAction =>
    ({ type: types.AdminActionEnum.SET_USER, payload: user
}),
```

```
  setForms: (forms: types.Forms): types.SetForms =>
    ({ type: types.AdminActionEnum.SET_FORMS, payload: forms }),
  addItem: (payload: { type: ITypeOfContent, data: IProject | ITeam | IVacancy }): types.AddItem =>
    ({ type: types.AdminActionEnum.ADD_ITEM, payload }),
  saveItem: (payload: { type: ITypeOfContent, id: string, data: IProject | ITeam | IVacancy }): types.SaveItem =>
    ({ type: types.AdminActionEnum.SAVE_ITEM, payload }),
  deleteItem: (payload: { type: ITypeOfContent, id: string }): types.DeleteItem =>
    ({ type: types.AdminActionEnum.DELETE_ITEM, payload }),
  login: (payload: { callBack: () => void }) =>
    ({ type: types.AdminActionEnum.AUTH, payload })
};
```

reducers/admin/index.ts

```
import * as types from "./types";

const initialState: types.AdminState = {
 user: null,
 projects: [],
 team: [],
 vacantions: [],
 isAuth: false,
 forms: {
  requestedProjects: [],
  requestedVacancies: [],
 }
};

export default function (state = initialState, { type, payload }: any): types.AdminState {
 switch (type) {
  case types.AdminActionEnum.SET_USER:
   return { ...state, isAuth: true, user: payload };

  case types.AdminActionEnum.SET_PROJECTS:
   return { ...state, projects: payload };

  case types.AdminActionEnum.SET_TEAM:
   return { ...state, team: payload };

  case types.AdminActionEnum.SET_VACANTIONS:
   return { ...state, vacantions: payload };

  case types.AdminActionEnum.SET_FORMS:
   return { ...state, forms: payload };

  case types.AdminActionEnum.SAVE_ITEM:
   switch (payload.type) {
    case "projects": {
     const newArray = state.projects.map(item => {
      return item.id === payload.id ? payload.data : item;
     });
     return {
      ...state,
      projects: newArray
     };
    }

    case "team": {
     const newArray = state.team.map(item => {
      return item.id === payload.id ? payload.data : item;
     });
     return {
      ...state,
      team: newArray
     };
    }

    case "vacantions": {
     const newArray = state.vacantions.map(item => {
      return item.id === payload.id ? payload.data : item;
     });
     return {
      ...state,
      vacantions: newArray
     };
    }

    default:
     return state;
   }
```

```
  case types.AdminActionEnum.ADD_ITEM:
    switch (payload.type) {
      case "projects":
        return {
          ...state,
          projects: [...state.projects, payload.data]
        };

      case "team":
        return {
          ...state,
          team: [...state.team, payload.data]
        };

      case "vacantions":
        return {
          ...state,
          vacantions: [...state.vacantions, payload.data]
        };

      default:
        return state;
    }

  default:
    return state;
 }
}
```

reducers/admin/types.ts

```
import { IVacancy } from "../../../models/IVacancy";
import { ITeam } from "../../../models/ITeam";
import { IProject } from "../../../models/IProject";
import { ITypeOfContent } from
"../../../models/ITypeOfContent";
import { IProjectForm, IPeopleForm } from "models/IForms";
import { ContentActionEnum } from "../content/types";

export interface AdminState {
  user: User | null,
  projects: IProject[];
  team: ITeam[];
  vacantions: IVacancy[];
  isAuth: boolean;
  forms: Forms,
}

export type User = {
  displayName: string,
  email: string,
  photoURL: string,
}

export type Forms = {
  requestedProjects: IProjectForm[],
  requestedVacancies: IPeopleForm[],
}

export enum AdminActionEnum {
  AUTH = "AUTH",
  SET_PROJECTS = "SET_PROJECTS",
  SET_TEAM = "SET_TEAM",
  SET_VACANTIONS = "SET_VACANTIONS",
  SET_USER = "SET_USER",
  SAVE_ITEM = "SAVE_ITEM",
  SET_FORMS = "SET_FORMS",
  ADD_ITEM = "ADD_ITEM",
  DELETE_ITEM = "DELETE_ITEM",
}

export interface Login {
  type: AdminActionEnum.AUTH;
  payload: {
    callBack: () => void,
    email: string,
    password: string,
  }
}
export interface SetProjectsAction {
  type: AdminActionEnum.SET_PROJECTS;
  payload: IProject[]
```

```
}

export interface SetTeamAction {
  type: AdminActionEnum.SET_TEAM;
  payload: ITeam[]
}

export interface SetVacanciesAction {
  type: AdminActionEnum.SET_VACANTIONS;
  payload: IVacancy[]
}

export interface SetAuthAction {
  type: AdminActionEnum.SET_USER;
  payload: User;
}

export interface SetForms {
  type: AdminActionEnum.SET_FORMS;
  payload: {
    requestedProjects: IProjectForm[],
    requestedVacancies: IPeopleForm[],
  }
}

export interface SaveItem {
  type: AdminActionEnum.SAVE_ITEM;
  payload: {
    type: ITypeOfContent,
    data: IProject | ITeam | IVacancy,
    id: string,
  };
}

export interface AddItem {
  type: AdminActionEnum.ADD_ITEM;
  payload: {
    type: ITypeOfContent,
    data: IProject | ITeam | IVacancy,
  };
}

export interface DeleteItem {
  type: AdminActionEnum.DELETE_ITEM;
  payload: {
    type: ITypeOfContent,
    id: string,
  };
}

export type AdminAction =
  SetProjectsAction |
  SetTeamAction |
  SetVacanciesAction |
  SetAuthAction |
  AddItem |
  DeleteItem |
  SaveItem |
  SetForms
```

reducers/content/action-creators.ts

```
import * as types from "./types";
import { IProject } from "../../../models/IProject";
import { IVacancy } from "../../../models/IVacancy";
import { ITeam } from "../../../models/ITeam";
import { ITypeOfForm } from "../../../models/ITypeOfContent";
import { IProjectForm, IPeopleForm } from "models/IForms";

export const ContentActionCreators = {
  fetchData: (): types.FetchData =>
    ({ type: types.ContentActionEnum.FETCH_DATA }),
  setProjects: (projects: IProject[]): types.SetProjectsAction =>
    ({ type: types.ContentActionEnum.SET_PROJECTS, payload:
projects }),
  setVacancies: (vacantions: IVacancy[]):
types.SetVacanciesAction =>
    ({ type: types.ContentActionEnum.SET_VACANTIONS,
payload: vacantions }),
  setTeam: (team: ITeam[]): types.SetTeamAction =>
    ({ type: types.ContentActionEnum.SET_TEAM, payload:
team }),
  saveForm: (payload: { type: ITypeOfForm, data: IProjectForm
| IPeopleForm }): types.SaveForm =>
    ({ type: types.ContentActionEnum.SAVE_FORM, payload }),
```

```
  toggleLoader: (status?: boolean): types.ToggleLoader =>
    ({ type: types.ContentActionEnum.TOGGLE_LOADER,
payload: status }),
};
```

reducers/content/index.ts

```
import { IProject } from "../../../models/IProject";
import { ITeam } from "../../../models/ITeam";
import { IVacancy } from "../../../models/IVacancy";
import * as types from "./types";

const initialState: types.ContentState = {
 projects: [],
 team: [],
 vacantions: [],
 showLoader: true,
};

export default function (state = initialState, { type, payload }:
types.ContentAction): types.ContentState {
 switch (type) {
  case types.ContentActionEnum.SET_PROJECTS:
   return { ...state, projects: payload };

  case types.ContentActionEnum.SET_TEAM:
   return { ...state, team: payload };

  case types.ContentActionEnum.SET_VACANTIONS:
   return { ...state, vacantions: payload };

  case types.ContentActionEnum.TOGGLE_LOADER:
   return { ...state, showLoader: payload || !state.showLoader
};

  case types.ContentActionEnum.SAVE_ITEM:
   switch (payload.type) {
    case "projects": {
     let newArray: IProject[] = [];
     const newItem = state.projects.find((item: IProject) =>
item.id === payload.id);
     if (newItem) {
      newArray = state.projects.map((item: IProject) => item.id
=== payload.id ? payload.data : item) as IProject[];
     } else {
      newArray = [...state.projects, payload.data] as IProject[];
     }
     return {
      ...state,
      projects: newArray
     };
    }

    case "team": {
     let newArray: ITeam[] = [];
     const newItem = state.team.find((item: ITeam) => item.id
=== payload.id);
     if (newItem) {
      newArray = state.team.map((item: ITeam) => item.id ===
payload.id ? payload.data : item) as ITeam[];
     } else {
      newArray = [...state.team, payload.data] as ITeam[];
     }
     return {
      ...state,
      team: newArray
     };
    }

    case "vacantions": {
     let newArray: IVacancy[] = [];
     const newItem = state.vacantions.find((item: IVacancy) =>
item.id === payload.id);
     if (newItem) {
      newArray = state.vacantions.map((item: IVacancy) =>
item.id === payload.id ? payload.data : item) as IVacancy[];
     } else {
      newArray = [...state.vacantions, payload.data] as
IVacancy[];
     }
     return {
      ...state,
      vacantions: newArray
     };
    }

    default:
```

```
      return state;
    }

  default:
    return state;
 }
}
```

reducers/content/types.ts

```
import { IVacancy } from "../../../models/IVacancy";
import { ITeam } from "../../../models/ITeam";
import { IProject } from "../../../models/IProject";
import { ITypeOfContent, ITypeOfForm } from
"../../../models/ITypeOfContent";
import { IPeopleForm, IProjectForm } from "models/IForms";

export interface ContentState {
  projects: IProject[];
  team: ITeam[];
  vacantions: IVacancy[];
  showLoader: boolean,
}

export enum ContentActionEnum {
  SET_PROJECTS = "SET_PROJECTS",
  SET_TEAM = "SET_TEAM",
  SET_VACANTIONS = "SET_VACANTIONS",
  TOGGLE_LOADER = "TOGGLE_LOADER",
  SAVE_ITEM = "SAVE_ITEM",
  SAVE_FORM = "SAVE_FORM",
  FETCH_DATA = "FETCH_DATA",
  DELETE_ITEM = "DELETE_ITEM",
}

export interface FetchData {
  type: ContentActionEnum.FETCH_DATA;
  payload?: undefined
}

export interface ToggleLoader {
  type: ContentActionEnum.TOGGLE_LOADER;
  payload?: boolean
}

export interface SaveItem {
  type: ContentActionEnum.SAVE_ITEM;
  payload: {
    type: ITypeOfContent,
    data: IProject | ITeam | IVacancy,
    id: string,
  };
}
export interface SaveForm {
  type: ContentActionEnum.SAVE_FORM;
  payload: {
    type: ITypeOfForm,
    data: IPeopleForm | IProjectForm,
  };
}

export interface SetProjectsAction {
  type: ContentActionEnum.SET_PROJECTS;
  payload: IProject[]
}

export interface SetTeamAction {
  type: ContentActionEnum.SET_TEAM;
  payload: ITeam[]
}

export interface SetVacanciesAction {
  type: ContentActionEnum.SET_VACANTIONS;
  payload: IVacancy[]
}

export interface DeleteItem {
  type: ContentActionEnum.DELETE_ITEM;
  payload: {
    type: ITypeOfContent,
    id: string,
  };
}
```

```
export type ContentAction =
  SetProjectsAction |
  SetTeamAction |
  SetVacanciesAction |
  ToggleLoader |
  SaveItem |
  FetchData |
  DeleteItem |
  SaveForm
```

cypress/support/constants.js

```
export const scriptMessage = "<img
onerror='alert(\"Взламано!\");' src='invalid-image' />"
```

cypress/integration/preview-mode/project-form.spec.js

```
import { scriptMessage } from "../../support/constants";
// const baseUrl = "https://company-98642.web.app";
const baseUrl = "localhost:3000";

describe('Перевірка форми замовлення проекту', () => {
 beforeEach(() => {
  cy.visit(baseUrl)
 })

 it('Перевірка валідації полів на спроможність захисту
від введення стороннього коду', () => {
  cy.get('[data-cy="name-field"]')
   .type(scriptMessage)
   .blur();
  cy.get('[data-cy="name-field"]')
   .siblings('.input__error')
   .should('have.text', 'Name is not valid');

  cy.get('[data-cy="email-field"]')
   .type(scriptMessage)
   .blur();
  cy.get('[data-cy="email-field"]')
   .siblings('.input__error')
   .should('have.text', 'email must be a valid email');

  cy.get('[data-cy="description-field"]')
   .type(scriptMessage)
   .blur();
  cy.get('[data-cy="description-field"]')
   .siblings('.input__error')
   .should('have.text', '<, >, : characters are prohibited');
 });

 it('Перевірка форми на захист від спаму повідомлень',
() => {
  cy.get('[data-cy="name-field"]')
   .type('Ваня');

  cy.get('[data-cy="email-field"]')
   .type('testEmail@mail.com');

  cy.get('[data-cy="description-field"]')
   .type('Тестовий текст');

  cy.get('[data-cy="service-button"]')
   .eq(0)
   .click();

  cy.get('[data-cy="budget-button"]')
   .eq(0)
   .click();

  cy.get('[data-cy="send-button"]')
   .click()
   .should('be.disabled');
 });
});

describe('Перевірка форми відгуку на вакансію', () => {
 beforeEach(() => {
  cy.visit(`${baseUrl}/vacantions/?id=1652560645942`)
 })

 it('Перевірка валідації полів на спроможність захисту
від введення стороннього коду', () => {
  cy.get('[data-cy="firstName-field"]')
   .type(scriptMessage)
   .blur();
```

```
cy.get('[data-cy="firstName-field"]')
  .siblings('.input__error')
  .should('have.text', 'Name is not valid');

cy.get('[data-cy="lastName-field"]')
  .type(scriptMessage)
  .blur();
cy.get('[data-cy="lastName-field"]')
  .siblings('.input__error')
  .should('have.text', 'Name is not valid');

cy.get('[data-cy="email-field"]')
  .type(scriptMessage)
  .blur();
cy.get('[data-cy="email-field"]')
  .siblings('.input__error')
  .should('have.text', 'email must be a valid email');

cy.get('[data-cy="description-field"]')
  .type(scriptMessage)
  .blur();
cy.get('[data-cy="description-field"]')
  .siblings('.input__error')
  .should('have.text', '<, >, : characters are prohibited');

cy.get('[data-cy="phone-field"]')
  .type(scriptMessage)
  .blur();
cy.get('[data-cy="phone-field"]')
  .siblings('.input__error')
  .should('have.text', 'Phone number is not valid');

cy.get('[data-cy="linkedin-field"]')
  .type(scriptMessage)
  .blur();
cy.get('[data-cy="linkedin-field"]')
  .siblings('.input__error')
  .should('have.text', 'linkedin must be a valid URL');

  cy.get('[data-cy="cvLink-field"]')
    .type(scriptMessage)
    .blur();
  cy.get('[data-cy="cvLink-field"]')
    .siblings('.input__error')
    .should('have.text', 'cvLink must be a valid URL');
 });

 it('Перевірка форми на захист від спаму повідомлень',
() => {
  cy.get('[data-cy="firstName-field"]')
    .type('Ваня');

  cy.get('[data-cy="lastName-field"]')
    .type('Ткач');

  cy.get('[data-cy="email-field"]')
    .type('testEmail@mail.com');

  cy.get('[data-cy="description-field"]')
    .type('Тестовий текст');

  cy.get('[data-cy="phone-field"]')
    .type('0666666666');

  cy.get('[data-cy="linkedin-field"]')
    .type('https://testsite.com');

  cy.get('[data-cy="cvLink-field"]')
    .type('https://testsite.com');

  cy.get('[data-cy="send-button"]')
    .click()
    .should('be.disabled');
 });
});
```