**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра обчислювальної техніки**

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2023 р.

# Магістерська дисертація

**на здобуття ступеня магістра**

**за освітньо-професійною програмою «Комп'ютерні системи та мережі» зі спеціальності 123 «Комп'ютерна інженерія»**

**на тему: «**Система інтерактивних елементів Front End - Back End взаємодії у web застосунках**»**

Виконав (-ла):
студент VI курсу, групи ІО-22мп
Самутін Олексій Ігорович _____

Керівник:
проф. каф. ОТ, д.т.н., професор
Кулаков Юрій Олексійович _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор
Кулаков Юрій Олексійович _____

Рецензент:
доц. каф. ІСТ, к.т.н., доц.,
Коган Алла Вікторівна _____

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

Київ – 2023 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *123. Комп'ютерна інженерія*
(код і назва)

Спеціалізація *«Комп'ютерні системи та мережі»*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
*Сергій Стіренко*
(підпис) (ініціали, прізвище)
« » _____ 20***23*** р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Самутін Олексій Ігорович
(прізвище, ім'я, по батькові)

1. Тема дисертації «Система інтерактивних елементів Front End - Back End взаємодії у web застосунках»

Науковий керівник дисертації проф. каф. ОТ, д.т.н. Кулаков Ю.О.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «***07***» ***11*** ***2023*** р. № ***5168-с***

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження Процес створення спеціалізованої соціальної мережі у веб-середовищі

4. Предмет дослідження *Способи та методи створення інтерактивних веб-додатків, соціальних мереж*

5. Перелік завдань, які потрібно розробити: розробити спеціалізований крос-браузерний додаток для вирішення вузькоспеціалізованих задач

6. Консультанти розділів дисертації:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормконтроль | Кулаков Ю. О. | | |

7. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів дисертації | Примітка |
|---|---|---|---|
| 1 | *Вивчення літератури* | *01.02.2023* | |
| 2 | *Складання і узгодження технічного завдання* | *01.02.2023* | |
| 3 | *Написання вступної частини та огляд рішень* | *22.09.2023* | |
| 4 | *Моделювання розробленого способу* | *20.10.2023* | |
| 5 | *Оформлення документації ДП* | *10.11.2023* | |
| 6 | *Попередній захист та проходження нормативного контролю* | *26.11.2023* | |
| 7 | *Подання ДП рецензенту* | | |

Студент Олексій САМУТІН

Науковий керівник дисертації Юрій КУЛАКОВ

**РЕФЕРАТ**

**на магістерську дисертацію**

виконану на тему: *Система інтерактивних елементів Front End - Back End взаємодії у web застосунках*

**Актуальність.** Сучасні соціальні мережі мають великий вплив на комунікацію людей між собою, а тематиці безпілотних літальних пристроїв особливо стала популярною останні роки, а отже і створення спеціального засобу для комунікації, обміном досвіду та порад в цій сфері стає як ніколи актуальним, і завдяки спеціалізованій соціальній мережі це буде набагато легше зробити ніж готовими месенджерами або іншими соціальними мережами.

**Мета і завдання дослідження.** Метою магістерської роботи є розроблення спеціального інтерактивного веб-додатку в якому користувачі зможуть вільно спілкуватися та обмінюватися досвідом з використовування безпілотних літальних апаратів

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- Дослідження структури побудови веб-додатку
- Дослідження основних структур даних які могли бути використані в спеціалізованому середовищі безпілотних літальних пристроїв
- Створення веб-додатку на основі досліджених рішень, проблем та потреб в галузі безпілотників

**Об’єкт дослідження –** процес створення спеціалізованої соціальної мережі у веб-середовищі

**Предмет дослідження** – способи та методи створення інтерактивних веб-додатків, соціальних мереж

**Методи досліджень.** Аналіз літератури та наукових робіт, аналіз користувацького досвіду та відгуків.

Наукова новизна одержаних результатів роботи полягає у наступному:

- Основною новизною є спеціалізована спрямованість платформи. Дослідження та розробка соціальної мережі для пілотів квадрокоптерів враховує особливості цільової аудиторії та їхні потреби у взаємодії, обміні досвідом та безпеці польотів.
- Розроблено веб-додаток

Завдяки проведеному дослідженню була створена соціальна мережа яка має спеціалізацію саме на пілотів безпілотних літальних пристроїв.

**Особистий внесок здобувача.** Магістерське дослідження є самостійно виконаною роботою, в якій сформовано авторський підхід до розробки веб-додатку за допомогою різних інструментів розробки що відносяться до вирішення задачі створення власної інтерактивної системи інтерактивних елементів Front End - Back End взаємодії

**Практична цінність.** Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- Розробка та створення власних спеціалізованих соціальних мереж
- Розробка функціональності соціальної мережі, адаптованої спеціально для пілотів квадрокоптерів, може надати їм ефективний інструмент для обміну досвідом, спілкування та розвитку у своїй сфері.

**Ключові слова.** ВЕБ ДОДАТОК, СОЦІАЛЬНІ МЕРЕЖІ, БЕЗПІЛОТНІ ЛІТАЛЬНІ ПРИСТРОЇ, FRONT-END, BACK-END

## ABSTRACT

**Subject relevance**. Modern social networks have a great influence on the communication of people with each other, and the topic of unmanned aerial devices has become especially popular in recent years, and therefore the creation of a special tool for communication, exchange of experience and advice in this area is becoming more relevant than ever, and thanks to a specialized social network it will be much easier to do than ready-made messengers or other social networks.

**Purpose and research objectives**. The goal of the master's thesis is to develop a special interactive web application in which users will be able to freely communicate and exchange experience in using unmanned aerial vehicles

To achieve the goal of the research, the following tasks were set and solved:

- Studying the structure of building a web application.
- Research of basic data structures that could be used in the specialized environment of unmanned aerial devices.
- Creation of a web application based on researched solutions, problems and needs in the field of drones.

**Research object.** The process of creating a specialized social network in the web environment

**Research subject**. Ways and methods of creating interactive web applications, social networks.

**Research methods.** Analysis of literature and scientific works, analysis of user experience and reviews.

**Scientific novelty**. The scientific novelty of the obtained work results is as follows:

- The main novelty is the specialized orientation of the platform. Research and development of a social network for quadcopter pilots takes into account the specifics of the target audience and their needs for interaction, experience sharing and flight safety.
- Developed a web application

Thanks to the conducted research, a social network was created that specializes in pilots of unmanned aerial devices.

**Personal contribution of the acquirer.** The master's research is an independently completed work in which an author's approach to the development of a web application was formed using various development tools related to solving the problem of creating an own interactive system of interactive elements of Front End - Back End interaction.

**Practical significance of the obtained results**. The obtained results can be used in future research in the following directions:

- Development and creation of own specialized social networks.
- Developing social network functionality tailored specifically for quadcopter pilots can provide them with an effective tool to share experiences, communicate and develop in their field.

**Keywords**. WEB APPLICATION, SOCIAL NETWORKS, DRONES, FRONT-END, BACK-END

# Пояснювальна записка
# до магістерської дисертації

на тему: «Система інтерактивних елементів Front End - Back End взаємодії у web застосунках»

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БПЛА – Безпілотний Літальний Апарат

ESC – Electronic Speed Control

FPV – First Person View

API – Application Programming Interface

JWT – JSON Web Token

HTML – Hyper Text Markup Language

VTX – Video Transmitter

RX – Receiver

JSX – JavaScript XML

# ВСТУП

У сучасному інформаційному суспільстві інтеграція технологій у різноманітні сфери діяльності людей стає домінуючою тенденцією. З розвитком технологій зйомки з повітря за допомогою квадрокоптерів відкривається новий простір можливостей для використання цих пристроїв у великому спектрі галузей, включаючи рятувальні операції, аграрну сферу, моніторинг навколишнього середовища та багато інших.

Актуальність даної теми полягає в тому, що існуючі соціальні мережі недостатньо адаптовані для потреб пілотів дронів та фахівців, які працюють у цій сфері. Інформаційний обмін, навчання, обговорення та співпраця у цій галузі потребують спеціалізованого простору, який враховує особливості роботи з квадрокоптерами. Створення соціальної мережі, спеціалізованої саме на цій тематиці, може сприяти активізації інноваційного розвитку дрон-індустрії та покращенню якості надання послуг, пов'язаних з використанням квадрокоптерів у різних галузях. Такий проект може стати важливим кроком у сприянні взаєморозумінню, обміну досвідом та розвитку спільноти фахівців, що працюють з квадрокоптерами.

Додатково, інтеграція соціальної мережі для пілотів квадрокоптерів відкриває можливості для створення сприятливого середовища для обміну передовими технологічними рішеннями, новаторськими ідеями та найкращими практиками. Це сприяє швидшому впровадженню та поширенню новітніх розробок у сфері дрон-технологій.

Крім того, в умовах швидкого росту кількості застосувань квадрокоптерів у комерційній сфері та галузях загального використання, виникає необхідність у покращенні нормативно-правового поля та вирішенні питань безпеки та приватності. Спільнота пілотів квадрокоптерів через соціальну мережу може активно сприяти цьому процесу, сприяючи обговоренню найактуальніших питань та наданню рекомендацій компетентним органам.

Крім того, створення соціальної мережі для пілотів квадрокоптерів може сприяти розвитку професійної спільноти та підвищенню рівня професійної компетентності у цій сфері. Вона надає можливість обміну досвідом, участь у навчальних програмах та тренінгах, що сприяє покращенню якості роботи та безпеки використання квадрокоптерів.

Розроблений веб-сервіс матиме функціонал для написання користувачами постів, створення власного профілю в яких можна буде додавати власні квадрокоптери, додавати список змагань в яких приймали участь, та спілкування з іншими користувачами.

Отже, створення спеціалізованої соціальної мережі для пілотів квадрокоптерів має великий потенціал для активізації розвитку цієї перспективної галузі, сприяючи покращенню комунікаційних зв'язків, обміну досвідом та найкращими практиками, а також підвищенню рівня професійної компетентності та безпеки використання квадрокоптерів у різних сферах діяльності.

# РОЗДІЛ 1

# ОГЛЯД РІЗНИХ ТИПІВ СОЦІАЛЬНИХ МЕРЕЖ ТА ОСНОВНІ ЗАДАЧІ ЯКІ ВОНИ ВИКОНУЮТЬ

Для того щоб зрозуміти актуальність даної теми потрібно розглянути існуючі соцмережі, які основні задачі вони виконують, які в них є недоліки, та зрозуміти які задачі має виконувати новоствторена система, також в рамках цього розділу буде наведений коротке пояснення всіх термінів які будуть використовуватися надалі, огляд існуючих рішень.

## 1.1 Загальний огляд соціальних мереж

Соціальна мережа - це онлайн-платформа, яка дозволяє користувачам спілкуватися, обмінюватися інформацією, ділитися враженнями, фотографіями, відео та іншим контентом. Це віртуальне середовище, де люди створюють особисті профілі, підключаються до спільнот, коментують, лайкають і діляться вмістом.

Основна ідея соціальних мереж - об'єднати людей, незалежно від їхнього розташування, щоб вони могли спілкуватися, спільно долучатися до подій, дізнаватися новини, знайомитися з новими людьми тощо.

Найвідоміші приклади соціальних мереж включають Facebook, Instagram, Twitter, LinkedIn, і багато інших. Кожна з цих мереж має свою унікальну спрямованість і функціонал, але загальна мета - сприяти взаємодії та обміну інформацією між користувачами.

Додатково, соціальні мережі можуть мати специфічні особливості, спрямовані на різні групи користувачів або цілі:

Facebook: Одна з найбільших та найпопулярніших соціальних мереж у світі. Вона орієнтована на широкий загал користувачів та дозволяє спілкуватися, ділитися фотографіями та відео, створювати групи та події.

Instagram: Спеціалізується на візуальному контенті, такому як фотографії та відео. Користувачі можуть додавати свої зображення, створювати сторіс, взаємодіяти за допомогою "лайків" та коментарів.

Twitter: Фокусується на коротких повідомленнях, що називаються "твітами". Вони обмежені кількістю символів та можуть містити тексти, посилання, зображення та відео.

LinkedIn: Орієнтований на професійний зв'язок та мережування. Ця платформа дозволяє користувачам створювати професійні профілі, додавати зв'язки та спільно працювати над проектами.

Snapchat: Спеціалізується на надсиланні коротких відео та зображень, які автоматично зникають після перегляду.

YouTube: Хоча це не класична соціальна мережа, ця платформа дозволяє користувачам завантажувати та ділитися відео, а також взаємодіяти за допомогою коментарів та оцінок.

Кожна з цих соціальних мереж має свою унікальну аудиторію та спрямованість, що дозволяє користувачам обирати ту, яка найкраще відповідає їхнім потребам та інтересам. Далі спробуємо більш детально розглянути кожну з цих соцмереж щоб зрозуміти їх переваги та недоліки, та причину створення власної платформи.

### 1.2.1 Facebook

Facebook - це одна з найбільших та найпопулярніших соціальних мереж у світі. Заснована Марком Цукербергом у 2004 році, вона стала суцільною платформою для спілкування, обміну враженнями та відстеження подій та новин у житті користувачів.

Основні компоненти та функції Facebook: Профіль та Хроніка – це коли кожен користувач має власний профіль, де можна додавати інформацію про себе, фотографії, відео та інше. Хроніка - це особиста сторінка користувача, де відображається активність та дописи.

Новинна стрічка - головний розділ, де користувачі можуть бачити публікації своїх друзів, сторінок, груп та подій, на які вони підписані.

Друзі та Підписки - Користувачі можуть додавати інших користувачів у свій список друзів або підписатися на їхні сторінки. Це дозволяє отримувати оновлення та бачити активність обраних осіб.

Групи - користувачі можуть створювати та приєднуватися до груп зі спільними інтересами. Вони можуть бути загальнодоступними або закритими.

Події - користувачі можуть створювати події, запрошувати друзів та взяти участь у заходах.

Месенджер - Facebook має власний чат-сервіс для обміну повідомленнями з друзями та співробітниками.

Статуси та Фотографії - Користувачі можуть додавати статуси, фотографії та відео для поділу зі своїм колективом.

Реакції та Коментарі - користувачі можуть виражати свої емоції за допомогою реакцій (лайки, смайли тощо) та коментувати публікації.

Реклама та Бізнес-сторінки - Facebook надає можливість підприємствам та брендам створювати власні сторінки для рекламних та маркетингових цілей.

Facebook є платформою, що надає можливість спілкуватися та ділитися враженнями для мільярдів користувачів по всьому світу. За допомогою широкого спектру функцій, він дозволяє побудувати власну віртуальну спільноту та залишатися на зв'язку з друзями та рідними. (рис.1)

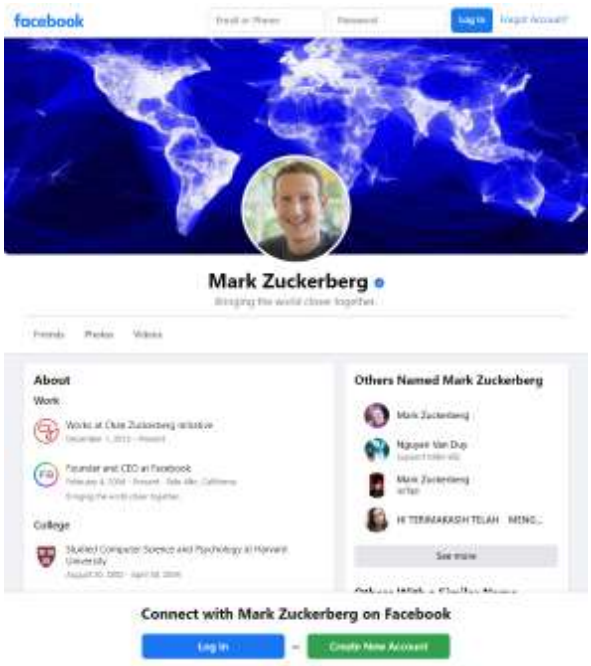


*Рис 1.1 Приклад сторінки на платформі Facebook* [1]

### 1.2.2 Instagram

Instagram - це соціальна мережа, спрямована на обмін фотографіями та відеозаписами. Вона створена Кевіном Сістромом та Майкою Крігером і була запущена у 2010 році. Отримавши велику популярність, Instagram швидко став однією з найпопулярніших платформ для обміну візуальним контентом.
Основні компоненти та функції Instagram:

Профіль - кожен користувач має свій власний профіль, де можна додати аватар, опис про себе та посилання на інші соціальні мережі.

Пост - основний тип контенту у Instagram - це пости. Користувачі можуть додавати фотографії та відео, а також додавати описи та хештеги.

Хронологія та Новини - В хронології відображаються всі пости користувача, а також ті, на які він підписаний. Сторінка "Новини" відображає оновлення від друзів та підписок.

Історії (Stories) - користувачі можуть додавати короткі відеоролики або фотографії, які будуть відображатися протягом 24 годин.

Live-трансляції - можливість транслювати відео в реальному часі та взаємодіяти з глядачами у режимі реального часу.

Direct (Особисті повідомлення) - можливість обмінюватися приватними повідомленнями з іншими користувачами.

Ігрові функції - деякі користувачі можуть додавати ігри в свої сторінки або відправляти запрошення до ігор.

Хештеги та Місця - використання хештегів дозволяє збільшити досяг та відкрити контент для широкого кола користувачів.

Реклама - Instagram надає можливість підприємствам рекламувати свої продукти та послуги за допомогою рекламних постів

Instagram (рис. 2) визнаний світовим лідером у сфері візуального контенту та є популярним майданчиком для взаємодії, натхнення та відкриття нових ідей.

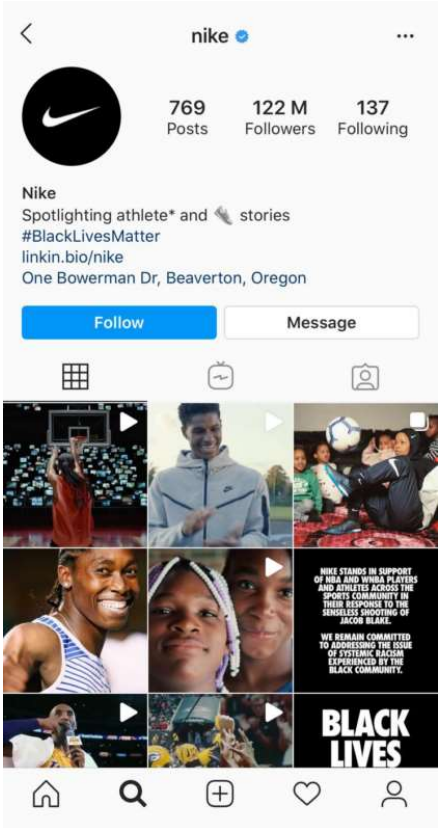


*Рис 1.2. Приклад сторінки на платформі Instagram* [2]

### 1.2.3 Twitter

Twitter - це популярна соціальна мережа, яка дозволяє користувачам надсилати та читати короткі повідомлення, відомі як "твіти". Платформа була заснована у 2006 році Джеком Дорсі та його командою, і вона швидко набула величезної популярності.

Основні компоненти та функції Twitter:

Твіти (Tweets) - це короткі повідомлення, що складаються з 280 символів або менше. Твіти можуть містити тексти, посилання, зображення, відео та інші медіа-елементи.

Профіль та Біографія - кожен користувач має свій власний профіль, де можна вказати ім'я, нікнейм, аватар, біографію та посилання на інші ресурси.

Фоловери та Фоловінг - користувачі можуть підписуватися на інших користувачів, щоб отримувати їхні оновлення. Взаємна підписка означає, що обидва користувачі підписані один на одного.

Хештеги - використання хештегів дозволяє класифікувати та згруповувати твіти за темою. Користувачі можуть шукати твіти за конкретними хештегами.

Ретвіти (Retweets) та Лайки (Likes) - користувачі можуть поширювати чужі твіти, натискуючи кнопку "Ретвіт", або виражати вподобання, натискуючи кнопку "Лайк".

Моменти (Moments) - це вибірка найцікавіших та найактуальніших подій, які відбуваються у світі.

Повідомлення (Direct Messages) - можливість обмінюватися приватними повідомленнями з іншими користувачами.

Twitter Spaces - функція для створення аудіо-конференцій та обговорення тем в режимі реального часу.

Тренди (Trending Topics) - список популярних тем та хештегів, які обговорюються в даний момент.

Twitter є платформою для обміну інформацією, новинами та думками в реальному часі. Він використовується як засіб спілкування, а також як засіб для навчання та інформування про події в усьому світі.

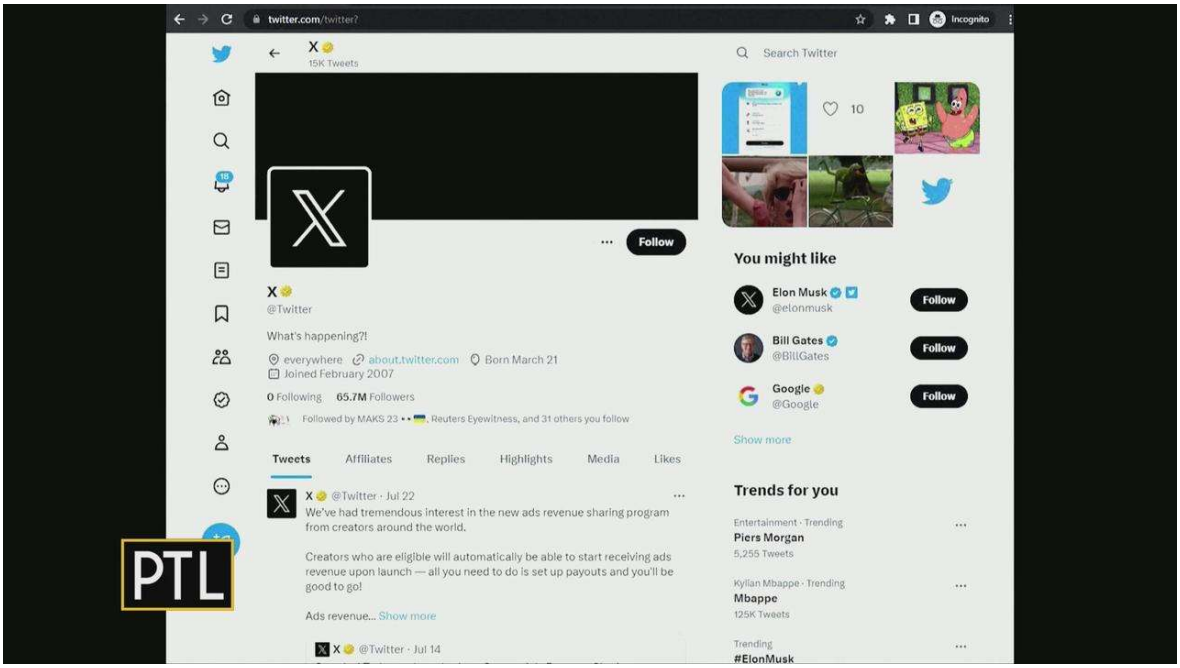


*Рис 1.3. Приклад сторінки на платформі Twitter* [3]

### 1.2.4 LinkedIn

LinkedIn - це професійна соціальна мережа, спрямована на зв'язки між фахівцями різних галузей та сфер бізнесу. Заснована у 2002 році Рідом Гоффманом, Алленом Блю та іншими співзасновниками, LinkedIn став однією з найвпливовіших платформ для кар'єрного розвитку та професійних контактів.

Основні компоненти та функції LinkedIn:

Профіль - кожен користувач має свій власний профіль, де можна вказати освіту, робочий досвід, навички, досягнення та іншу професійну інформацію.

Зв'язки - користувачі можуть додавати інших користувачів у свій список зв'язків, що дозволяє збільшити професійну мережу та мати доступ до їхньої інформації.

Рекомендації - користувачі можуть надавати та отримувати рекомендації від колег та партнерів.

Публікації та Контент - користувачі можуть публікувати статті, дописи, відео та інший контент, який стосується їхньої професійної діяльності.

Робочі Оголошення - користувачі можуть шукати та подавати заявки на вакансії, а також розміщувати оголошення про пошуки роботи.

Підприємства та Спільноти - компанії можуть створювати сторінки для просування свого бренду та привертання нових працівників.

Навчання - LinkedIn надає можливість проходити онлайн-курси та отримувати сертифікати у різних сферах.

Повідомлення - користувачі можуть обмінюватися повідомленнями зі своїми зв'язками та партнерами.

Аналітика та Звіти - LinkedIn надає статистику щодо відвідування та активності на профілі, щоб допомогти у вдосконаленні особистого бренду.

LinkedIn є важливим інструментом для професійного зростання, пошуку нових можливостей та розвитку бізнесу. Він дозволяє користувачам будувати і підтримувати професійні зв'язки, обмінюватися ідеями та навичками, а також знаходити нові кар'єрні можливості.

### 1.3 Порівняльна характеристика різних соціальних мереж

Провівши короткий огляд кожної з соціальних мереж, можна порівняти кожну з них та зрозуміти потреби в створенні власної спеціалізованої мережі. В якості порівняння були взяті самі популярні соц мережі так складений перелік функцій які вони можуть виконувати, також було додано власну розробку як останню колонку для порівняння можливостей з існуючими рішеннями. (Таблиця 1.1

*Таблиця 1.1*

*Порівняльна характеристика різних соціальних мереж*

| Функція | Twitter | LinkedIn | Facebook | Instagram | Власна розробка |
|---|---|---|---|---|---|
| Профіль | Так | Так | Так | Так | Так |
| Зв'язки/Друзі | Так | Так | Так | Так | Ні |
| Пост | Так | Ні | Так | Так | Так |
| Твіти | Так | Ні | Ні | Ні | Ні |
| Публікації | Так | Так | Так | Так | Ні |
| Рекомендації | Ні | Так | Ні | Ні | Ні |
| Ретвіти/Репости | Так | Ні | Так | Так | Ні |
| Лайки/Вподобання | Так | Так | Так | Так | Так |
| Прямі повідомлення | Так | Так | Так | Так | Ні |
| Групи/Спільноти | Ні | Так | Так | Ні | Ні |
| Події/Заходи | Так | Так | Так | Так | Так |
| Прямі трансляції | Так | Ні | Так | Так | Ні |
| Тренди (Trending Topics) | Так | Ні | Так | Ні | Ні |
| Моменти (Moments) | Так | Ні | Так | Ні | Ні |
| Рекламні можливості | Так | Так | Так | Так | Так |
| Резюме та Пошук роботи | Ні | Так | Ні | Ні | Ні |
| Бізнес-сторінки | Ні | Так | Так | Так | Ні |
| Ігри/Геймінг | Ні | Ні | Так | Ні | Ні |
| Аналітика | Частково | Так | Так | Частково | Ні |

## Висновки до розділу 1

Оглянувши більшість популярних соціальних мереж було зрозуміло що більшість з них націлені на широку аудиторію і не мають того потрібного функціоналу який потрібен для людей які займаються пілотуванням квадрокоптерів. А саме створювати профілі з власним переліком безпілотних літальних пристроїв, без можливості записуватися на змагання.

Саме тому було прийняте рішення створювати таку соціальну мережу яка не матиме лишнього функціоналу, який тільки відволікатиме увагу і не буде потрібен напряму для користувачів мережі, а залишити тільки необхідний функціонал для створення профілів, з можливістю додання власного переліку безпілотних літальних апаратів, можливістю записуватися на змагання, та вести дискусію в постах. Такий підхід дозволить користувачам максимально концентруватися на основній меті мережі - обміні досвідом та взаємодії у сфері заходів спрямованій на використання безпілотників. Крім того, завдяки відкритості платформи, ми можемо враховувати потреби та побажання користувачів у майбутньому, постійно вдосконалюючи сервіс та додаючи нові можливості.

# РОЗДІЛ 2

# МЕТОДИ РОЗРОБКИ СИСТЕМ ІНТЕРАКТИВНИХ ЕЛЕМЕНТІВ В FRONT-END BACK-END ЗАСТОСУНКАХ

## 2.1 Вибір методів для розробки Front-End системи

Front-end, або клієнтська частина, веб-додатка - це та частина програмного забезпечення, яка відповідає за відображення та взаємодію з користувачем у браузері чи на мобільному пристрої. Зазвичай Front-end складається з 3 частин, за структуру та семантику веб сторінки відповідає HTML, за стилізацію сторінок відповідає мова для програмування стилів CSS, а також всю поведінку та взаємодію елементів між собою виконує мова програмування JavaScript. Але напряму ці елементи використовуються досить рідко, оскільки є безліч бібліотек та інструментів які покращують базові мови для розмітки, стилізування, та програмування взаємодії елементів між собою. Для кращого розуміння обраних інструментів потрібно провести дослідження та зрозуміти які з існуючих інструментів будуть найкращими для вирішення поставлених задач, а саме створення інтерактивної системи елементів в Front-End частині.

### 2.1.1 Методи побудови системи UI елементів за допомогою React

React - це відкрита JavaScript бібліотека, яка призначена для розробки інтерфейсів користувача, зокрема для веб-додатків зі складним та динамічним вмістом. Реакт був розроблений компанією Facebook і зарекомендував себе як потужний та ефективний інструмент для створення сучасних веб-додатків.

По-перше, React - це всього лише бібліотека, а не фреймворк. В ній немає великої кількості вбудованих функцій, за допомогою яких можна легко створювати складні веб-додатки. Друга частина речення також важлива: для створення користувацьких інтерфейсів.

Строго кажучи, React - це всього лише бібліотека, яка спрощує розробку користувацьких інтерфейсів. Тут немає служб чи методів для викликів API, вбудованих моделей чи ORM. React цікавиться тільки користувацьким інтерфейсом. Можна сказати, що це частина "вигляду" нашого додатка. І це все! У цьому контексті час від часу можна прочитати, що React можна розглядати як "V" в архітектурі MVC (Model-View-Controller) або MVVM (Model-View-ViewModel). Це досить добре пояснює суть.

React пропонує декларативний спосіб моделювання стану, де стан відноситься до самого користувацького інтерфейсу. Просто кажучи, це означає, що ми вказуємо у нашому коді, як повинен виглядати наш інтерфейс для кожного стану компонента. Подивимося на простий приклад: Якщо користувач увійшов у систему, покажіть йому панель керування. Якщо ні, покажіть форму входу.

Логіка знаходиться в JavaScript-частині програми (де вона, власне кажучи, і повинна бути) і не в шаблонах, на відміну від багатьох інших веб-фреймворків, де це змішано. Це може здатися складним на початку, але незабаром стане зрозумілим, що це означає.

React заснований на компонентах. Ми пишемо інкапсульовані, функціональні компоненти, які можна комбінувати довільно та використовувати за власним бажанням. Розширення або успадкування компонентів теоретично можливе, але воно досить рідкісне в світі React. Замість роботи з успадкуванням, React сприяє композиції - об'єднанню кількох компонентів в "ціле".

Чи це означає, що ми не можемо будувати складні веб-додатки за допомогою React? Ні. Зовсім ні. У React є велика, високоякісна та дуже активна екосистема бібліотек, які, в свою чергу, базуються на, розширюють або доповнюють React таким чином, що йому не потрібно ховатися поза масштабними фреймворками, такими як Ember або Angular. Навпаки, після вступу в екосистему React та здобуття першого враження ми швидко виявимо кілька дійсно хороших інструментів та бібліотек, за допомогою яких ми

можемо будувати професійні, дуже індивідуальні та високо складні додатки. [4]

Основні концепції React

Компоненти - React дозволяє розбити інтерфейс на невеликі, незалежні компоненти. Кожен компонент може мати свій власний стан (state) та властивості (props).

Віртуальний DOM - React використовує віртуальний DOM, що дозволяє ефективно оновлювати та маніпулювати елементами інтерфейсу, мінімізуючи зайві операції зміни DOM.

JSX - це розширення синтаксису JavaScript, яке дозволяє описувати структуру компонентів за допомогою HTML-подібного коду прямо в JavaScript файлі.

Односторінкові додатки - React добре підходить для створення односторінкових додатків (SPA), які працюють в браузері та не вимагають перезавантаження сторінки.

Основні переваги React:

Ефективність - Завдяки віртуальному DOM та іншим оптимізаціям, React надає високу швидкодію та ефективну роботу навіть для великих та складних додатків.

Спільнота та екосистема - У React дуже активна спільнота розробників, що сприяє накопиченню багатьох корисних бібліотек та рішень для різних задач.

Розширюваність - React може бути комбінований з іншими бібліотеками та фреймворками для створення потужних та гнучких додатків.

Налаштованість - React надає можливість докладно налаштувати різні аспекти додатка, що дозволяє розробникам мати велику свободу та контроль.

Отже переглянувши всі переваги над використанням Реакт її було обрано як основну бібліотеку для створення інтерактивної системи на боці користувача, оскільки створювати систему елементів базуючись на концепції компонентів з JSX є не тільки зручно, але також полегшує написання нових компонентів, не задіючи при цьому HTML напряму, також React є одною з

найпопулярніших бібліотек серед розробників сьогодення, через свою швидкість, простоту.

### 2.1.2 Методи для управління станом додатку

Redux - це потужна бібліотека управління станом додатка в JavaScript. Вона дозволяє ефективно управляти станом додатка та забезпечує однозначну та передбачувану модель даних.

Основні Концепції Redux

Store (Сховище) - Redux використовує єдиний об'єкт - "store" - для зберігання всього стану додатка. Цей об'єкт є незмінним і може бути змінений тільки за допомогою дій (actions).

Actions (Дії) - Дії - це об'єкти, які вказують, що відбувається в додатку. Вони відправляються до "store" та містять тип дії та додаткові дані.

Reducers (Редуктори) - Редуктор - це чиста функція, яка приймає попередній стан та дію, і повертає новий стан. Вони використовуються для обробки дій та оновлення стану додатка.

Чисті функції - Redux надає можливість писати чисті функції, що полегшує тестування та розуміння коду.

Незмінність - Redux підтримує незмінність даних, що означає, що стан не може бути змінений напряму, а тільки через дії та редуктори.

### 2.1.3 Методи для збірки веб-додатків

Webpack - це потужний інструмент для збірки та оптимізації веб-додатків. Він дозволяє ефективно управляти залежностями, бандлити ресурси та оптимізувати продуктивність додатка.

Основні Концепції Webpack

Entry Points (Точки входу) - Webpack починає свою роботу з одного чи кількох вхідних файлів, які вказуються в конфігурації. Вони служать стартовими точками для збірки.

Loaders (Завантажувачі) - завантажувачі використовуються для обробки різних типів файлів (наприклад, CSS, зображення) та перетворення їх у модулі для включення у бандл.

Plugins (Плагіни) - плагіни розширюють функціонал Webpack та дозволяють виконувати різноманітні завдання, такі як оптимізація, генерація HTML файлів тощо.

Output (Вихідні файли) - Output конфігурує, куди і яким чином мають бути збережені результати роботи Webpack.

Модулі та Залежності - Webpack дозволяє керувати модульністю та залежностями, дозволяючи підключати зовнішні бібліотеки та інші ресурси.

Основні переваги Webpack

Модульність - Webpack дозволяє розділити додаток на невеликі модулі, що полегшує розробку та управління кодом.

Автоматизація - Завдяки конфігураційному файлу, Webpack автоматизує процес збірки та оптимізації додатка.

Розширюваність - З можливістю використання плагінів, Webpack може бути розширений для вирішення різних задач в рамках розробки.

Оптимізація продуктивності - Webpack дозволяє виконувати оптимізації, такі як мінімізація коду, оптимізація зображень, що полегшує швидкодію додатка.

Webpack - це важливий інструмент для розробників веб-додатків, який дозволяє ефективно управляти залежностями, оптимізувати та збирати проект. Використання Webpack допомагає створити швидкий та ефективний веб-додаток з покращеною організацією коду.

## 2.2 Особливості розробки системи Back-End

Back-end (або серверна частина) - це частина програмного забезпечення, яка відповідає за обробку даних та взаємодію з базами даних, а також за виконання логіки додатка на сервері. Основна роль back-end - це забезпечити

коректну роботу та обробку запитів, які надходять від користувачів чи клієнтської частини додатка (Front-end).

Завдання back-end включають:

Робота з базами даних: Зберігання, вибірка, оновлення та видалення даних в базі даних.

Обробка запитів: Опрацювання HTTP-запитів від клієнтської частини та надання відповіді.

Бізнес-логіка: Виконання різноманітних обчислень, операцій та операцій над даними відповідно до логіки додатка.

Безпека: Забезпечення безпеки додатка, включаючи аутентифікацію, авторизацію та захист від атак.

Управління сесіями та станом: Ведення сесій користувачів, зберігання та управління станом додатка.

Відправка та отримання повідомлень: Керування взаємодією зовнішніх сервісів та API.

Оптимізація продуктивності: Моніторинг та оптимізація швидкодії серверної частини для забезпечення ефективності додатка.

Комунікація між Front-end та Back-end зазвичай відбувається через мережу, з використанням протоколу HTTP або HTTPS. Back-end може бути написаний на різних мовах програмування, таких як JavaScript (за допомогою Node.js), Python, Ruby, Java, PHP, та інші. В данній дипломній роботі буде використовуватися середовище Node.js для розробки серверної (Back-end) частини

### 2.2.1 Методи для взаємодії з базами данних

MongoDB - це документ-орієнтована NoSQL база даних, призначена для зберігання великого обсягу даних.. MongoDB - це база даних, яка набула популярності приблизно наприкінці 2000-х років. Вона відноситься до категорії баз даних типу NoSQL.[5]

Основні Концепції та Особливості

Документи та Колекції:

В MongoDB дані зберігаються у вигляді документів, які є BSON (Binary JSON) об'єктами. Документи групуються в колекції, що дозволяє організувати дані за певною логікою.

Гнучкість Схеми:

MongoDB не вимагає строго визначеної схеми для документів. Це означає, що документи в одній колекції можуть мати різні набори полів.

Реплікація та Шардування:

MongoDB підтримує реплікацію для забезпечення високої доступності та шардування для розподілу даних на різних серверах.

Мови Запитів:

MongoDB використовує JavaScript-подібну мову запитів для взаємодії з базою даних.

Індекси:

Для оптимізації запитів, MongoDB підтримує різноманітні типи індексів, включаючи одно-, складні та текстові.

Порівняння MongoDB та MySQL

Тип Даних - MongoDB: Документо-орієнтована. Використовує BSON.

MySQL: Реляційна. Використовує SQL для операцій з даними.

Схема Даних - MongoDB: Гнучка схема. Дозволяє документам у колекції мати різну структуру.

MySQL: Статична схема. Вимагає строгого визначення таблиць та полів.

Мова Запитів MongoDB: JavaScript-подібна мова запитів.

MySQL: SQL (Structured Query Language).

Гнучкість та Швидкість:

MongoDB: Швидша для деяких видів операцій, особливо при великих обсягах даних та при роботі з великою кількістю документів.

MySQL: Ефективна для традиційних операцій реляційних баз даних, особливо при гарній структуризації даних.

Горизонтальна Масштабованість:

MongoDB: Добре підтримує горизонтальну масштабованість за допомогою шардування.

MySQL: Важко реалізувати горизонтальну масштабованість.

Також до переваг MongoDB можна віднести:

Масштабованість - MongoDB має дуже хорошу горизонтальну масштабованість завдяки можливості розподілу навантаження між різними серверами. Це особливо важливо для великих проектів з великою кількістю користувачів та великими обсягами даних.

Гнучкість в Роботі з Даними - завдяки документо-орієнтованій природі, MongoDB дозволяє легко працювати з різними типами даних. Ви можете зберігати структуровані та неструктуровані дані в одній колекції, що робить її дуже гнучкою.

Автономні Операції - MongoDB може працювати в автономному режимі без необхідності створення складних зв'язків між таблицями, що спрощує розробку та прискорює виконання запитів.

Гарантована Доставка Даних - MongoDB надає можливість гарантувати, що дані будуть доставлені до сервера в разі втрати з'єднання. Це важливо для забезпечення надійності та цілісності даних.

Повна Транзакційність - починаючи з версії 4.0, MongoDB підтримує повну транзакційність, що робить її конкурентоспроможною з реляційними базами даних у сфері транзакцій.

Обмеження MongoDB

Не підходить для таких сценаріїв:

В тих випадках, коли потрібно проводити складні зв'язки між даними та виконувати складні операції SQL, реляційні бази даних можуть бути кращим варіантом.

Вимоги до Ресурсів:

MongoDB може вимагати більшого обсягу пам'яті та обчислювальних ресурсів в порівнянні з деякими іншими базами даних.

Менша Підтримка Інструментів та Документації:

Хоча MongoDB має активну спільноту та хорошу документацію, іноді може бути складніше знайти відповіді на деякі запитання порівняно з більш популярними базами даних.

MongoDB та MySQL - це дві різних системи управління базами даних, кожна з яких має свої переваги та використовується у відповідних випадках. MongoDB найбільш схильна до роботи з невизначеною або змінною схемою та дозволяє працювати з великими обсягами невертикально збережених даних. MySQL, з іншого боку, надійно працює з традиційними структурованими даними та виявляється ефективною для реляційних операцій та складних запитів. Обираючи між ними, важливо враховувати конкретні потреби проекту та його характеристики. (Таблиця 2.1)

*Таблиця 2.1*

*Порівняльна характеристика MongoDB та MySQL*

| **Особливість** | **MongoDB** | **MySQL** |
|---|---|---|
| Тип даних | Документо-орієнтована | Реляційна |
| Схема даних | Гнучка | Статична |
| Мова запитів | JavaScript-подібна | SQL |
| Гнучкість у роботі з даними | Дозволяє зберігати дані різного типу в одній колекції | Вимагає строгого визначення структури таблиць |
| Горизонтальна масштабованість | Підтримується за допомогою шардування | Важко реалізувати |
| Швидкість роботи | Швидша для деяких операцій та великих обсягів даних | Ефективна для традиційних операцій реляційних баз даних |

*Продовження таблиці 2.1*

| **Колонка 1** | **Колонка 2** | **Колонка 3** |
|---|---|---|
| Транзакційність | Підтримка повноцінних транзакцій з версії 4.0 | Повноцінна транзакційність |
| Масштабування | Підтримує горизонтальне та вертикальне масштабування | Горизонтальне масштабування складніше |
| Операції з даними | Прості та швидкі для документо-орієнтованого підходу | Ефективні для складних операцій SQL |
| Ком'юніті та підтримка | Активна спільнота, хороша документація | Широка спільнота та добре документована |
| Обмеження | Вимагає більшого обсягу пам'яті та ресурсів | Менш вимоглива до ресурсів, але може бути менш гнучкою |

### 2.2.2 Інструменти для маршрутизації в веб-додатку

Express - це відносно невеликий фреймворк, який побудований на основі функціоналу веб-сервера Node.js для спрощення його API та додавання корисних нових можливостей. Він полегшує організацію функціональності вашого додатка за допомогою проміжного програмного забезпечення та маршрутизації; він надає корисні утиліти для об'єктів HTTP Node.js; сприяє відображенню динамічних HTML-виглядів; визначає стандарт легко впроваджуваної розширюваності.

Express - це фреймворк, що означає, що потрібно будувати свою програму в стилі Express. Проте підхід Express не є надто догматичним; він не накладає сувору структуру. Це означає, можна створити багато різних видів додатків, від додатків для відеочату до блогів та API.

Сам по собі Express, ймовірно, не робить все, що вам потрібно, і ймовірно, можна знайти велику кількість інших бібліотек, які інтегруються в свої додатки Express. У цьому відношенні він добре вписується в філософію Unix "робити одну річ добре".

Проте цей мінімалізм має двояке значення. Він гнучкий і додатки не навантажені невикористаними деталями, але в порівнянні з іншими фреймворками він робить дуже мало. Це означає, що якщо допустити помилки, потрібно приймати набагато більше рішень щодо архітектури додатку, і доведеться витрачати більше часу на пошук правильних сторонніх модулів. За замовчуванням ви отримуєте менше.

Деяким може сподобатися гнучкий фреймворк, інші можуть вимагати більш жорстких правил. Наприклад, PayPal використовує Express, але побудував над ним фреймворк, який більш жорстко виконує угоди для своїх багатьох розробників. Express не цікавиться тим, як ви структуруєте свої додатки, тому два розробники можуть приймати різні рішення. [6]

Тому, що вам дозволяють самим керувати напрямком розвитку свого додатку, ви можете прийняти нерозумне рішення, яке вам спізниться пізніше. Іноді, оглядаючи свої додатки, які я все ще вивчаю, побудовані на Express, я думаю: "Чому я зробив це так?"

Основні Особливості:

Маршрутизація – однією з ключових особливостей Express є його потужна система маршрутизації. Вона дозволяє легко визначати різні шляхи (routes) та обробляти їх.

Middleware - express використовує концепцію middleware, яка дозволяє виконувати функції під час обробки запиту. Це може бути використано для обробки даних, аутентифікації, журналювання та багатьох інших завдань.

Шаблонізатори - хоча Express сам по собі не має шаблонізатора, він дозволяє використовувати будь-який шаблонізатор за допомогою відповідного двигуна.

Статичні Файли - express надає простий спосіб обслуговувати статичні файли, такі як HTML, CSS та JavaScript, що дозволяє легко розгортати веб-сторінки.

Управління Сесіями та Кукісами - вбудовані можливості для роботи з сесіями та кукісами для відстеження стану користувача.

Аутентифікація та Авторизація - хоча Express сам по собі не надає вбудованої системи аутентифікації, він надає базові засоби для створення власних систем авторизації та аутентифікації.

Мультиплексування та Обробка Запитів - Express дозволяє обробляти різні типи запитів (GET, POST, PUT, DELETE) для того, щоб створювати різноманітні додатки та API.

Переваги:

Мінімалізм - Express є мінімалістичним та легким у використанні. Він надає базовий набір функцій, дозволяючи розробникам вибирати та використовувати додаткові компоненти за потреби.

Багатий Екосистема:

Велика кількість сторонніх middleware та модулів дозволяє розробникам швидко розширювати функціональність додатків.

Повна Свобода - Express не накладає обмежень щодо архітектури вашого додатка, дозволяючи вам будувати його так, як вам зручно.

Недоліки:

Відсутність Вбудованих Механізмів:

Деякі функціональності, такі як аутентифікація та авторизація, повинні бути реалізовані вручну або за допомогою сторонніх бібліотек.

Не підходить для складних проектів:

В дуже великих проектах може виникнути необхідність у додаткових компонентах та більш виразній структурі, яку Express не надає за замовчуванням.

Express.js є потужним та гнучким фреймворком для розробки веб-додатків на Node.js. Він надає базовий набір інструментів для розробки

широкого спектру додатків, від простих веб-сайтів до складних API. Однак важливо мати на увазі, що Express надає базовий каркас, і для певних завдань можливо знадобиться використовувати додаткові бібліотеки та компоненти.

### 2.2.3 Інструменти авторизації за допомогою jsonwebtoken (JWT)

JSON Web Token, або JWT ("джот") в скороченні, є стандартом безпечної передачі претензій в обмеженому просторі. Він знайшов свій шлях до всіх основних веб-фреймворків. Простота, компактність і зручність є ключовими особливостями його архітектури. Незважаючи на те, що використовуються набагато більш складні системи, у JWT є широкий спектр застосувань.

Хоча основна мета JWT - передача претензій між двома сторонами, можна сказати, що найважливішим аспектом є стандартизаційні зусилля у вигляді простого, за потреби перевіреного і/або зашифрованого, контейнерного формату. В минулому до цієї самої проблеми були запропоновані тимчасові рішення як приватно, так і публічно. Існують також старі стандарти для встановлення претензій до певних сторін. Те, що JWT привносить у цей процес - це простий, корисний, стандартний контейнерний формат. Навіть якщо деяке визначення поки що абстрактне, нескладно уявити, як їх можна використовувати: системи входу (хоча інші варіанти також можливі)[7]. Деякі з цих застосувань включають:

- Аутентифікація
- Авторизація
- Федеративна ідентичність
- Клієнтські сесії («безстатусні» сесії)
- Клієнтські секрети

Бібліотека jsonwebtoken - це інструмент для генерації та перевірки JSON Web Tokens (JWT) в Node.js. JWT - це стандарт, який визначає спосіб безпечно передавати інформацію між двома сторонами у вигляді JSON об'єкта. Він використовується для автентифікації та забезпечення безпеки даних.

Основні Поняття та Функціонал:

Створення Токену (Sign) - jsonwebtoken дозволяє підписувати дані (наприклад, користувача або певні додаткові властивості) та генерувати JWT. Цей токен містить дані та підпис, що дозволяє перевірити його подібність.

Валідація Токену (Verify):

Токен можна перевірити для того, щоб забезпечити, що він не був підроблений. Це важливо для гарантії, що дані, представлені у токені, є достовірними.

Використання Пайплайну Middleware:

JWT може бути використаний для аутентифікації користувачів у веб-додатках. Він може бути включений в пайплайн middleware, щоб перевіряти та обробляти токени під час кожного запиту.

Автоматичне Закінчення Токену (Expiration):

JWT може мати обмежений термін дії, після якого він стає недійсним. Це дозволяє забезпечити, що токен не може бути використаний назавжди.

Концепція Авторизації За Допомогою JWT Токенів

Авторизація за допомогою JWT токенів базується на передачі підписаних токенів між клієнтом та сервером. Основні кроки авторизації виглядають так:

Успішна Аутентифікація:

Користувач вводить свої облікові дані, і сервер перевіряє їх для визначення, чи є цей користувач дійсно тим, за кого він себе видає.

Генерація JWT Токену:

Після успішної аутентифікації, сервер генерує JWT токен, який містить інформацію про користувача та підпис.

Відправлення Токену Клієнту:

Токен відправляється до клієнта (зазвичай включається в заголовок запиту або як частина тіла відповіді).

При Кожному Запиті Перевіряється Токен:

При кожному запиті, клієнт включає токен у заголовок запиту. Сервер перевіряє цей токен, декодує його та перевіряє підпис. Якщо токен дійсний, то запит обробляється.

JWT токени використовуються для забезпечення безпеки та автентифікації даних у веб-додатках. Основні переваги включають:

Безпека:

JWT токени підписуються сервером, що робить їх важко підробити. Це дозволяє перевіряти їх подібність та гарантує, що дані в токені не були змінені.

Легкість Використання:

JWT токени можна легко передавати між клієнтом та сервером, дозволяючи ефективно аутентифікувати запити.

Масштабованість:

Використання токенів дозволяє легко автентифікувати користувачів у мікросервісах та розподілених системах.

Зменшення Запитів до Бази Даних:

Оскільки JWT токени містять основну інформацію про користувача, це може допомогти уникнути частих запитів до бази

### 2.3 Дослідження типів БПЛА для розробленого додатку

Різновиди дронів можна відрізнити за типом (планер, багатороторний, тощо), ступенем автономності, розміром та вагою, а також джерелом живлення. Ці характеристики є важливими, наприклад, для максимального радіусу польоту дрону, тривалості польоту та навантажувальної спроможності. Окрім самого дрону (тобто, 'платформи'), можна виділити різні типи навантажень, включаючи вантаж (наприклад, поштові відправлення, ліки, засоби для гасіння пожежі, рекламні матеріали тощо) та різноманітні типи сенсорів (наприклад, камери, детектори, метеорологічні сенсори тощо). Будуть описані застосування різних типів навантажень. Для проведення польоту дронам потрібна (певна кількість) бездротова комунікація з пілотом

на землі. Крім того, в більшості випадків потрібна комунікація з навантаженням, таким як камера чи сенсор. Для забезпечення цієї комунікації потрібний частотний спектр. Вимоги до частотного спектру залежать від типу дрону, характеристик польоту та навантаження. Оскільки частотний спектр не обмежується національними кордонами, потрібна міжнародна координація щодо його використання. Обговорюються правові питання використання частотного спектру та електронного обладнання (національні та міжнародні правові питання щодо частотного спектру та вимог до обладнання), а також частотний спектр і його вразливість (погляд на доступний частотний спектр і пов'язані ризики в його використанні) та нагляд та виконання (забезпечення використання частотного спектру, вимог до обладнання та необхідність міжнародного та європейського співробітництва). Нарешті, обговорюються майбутні тенденції у технології дронів. Тенденція полягає в тому, що дрони стають меншими, легшими, ефективнішими та дешевшими. В результаті дрони стануть все більш доступними для широкого загалу та будуть використовуватися для все більшого спектру завдань. Дрони будуть ставати все більш автономними і здатними працювати в угрупованнях. [8]

Для розробки системи яка буде генерувати необхідні комплектуючі для FPV квадрокоптеру потрібно розуміти з яких частин він складається, основні складові можна побачити на рис. 2.2.

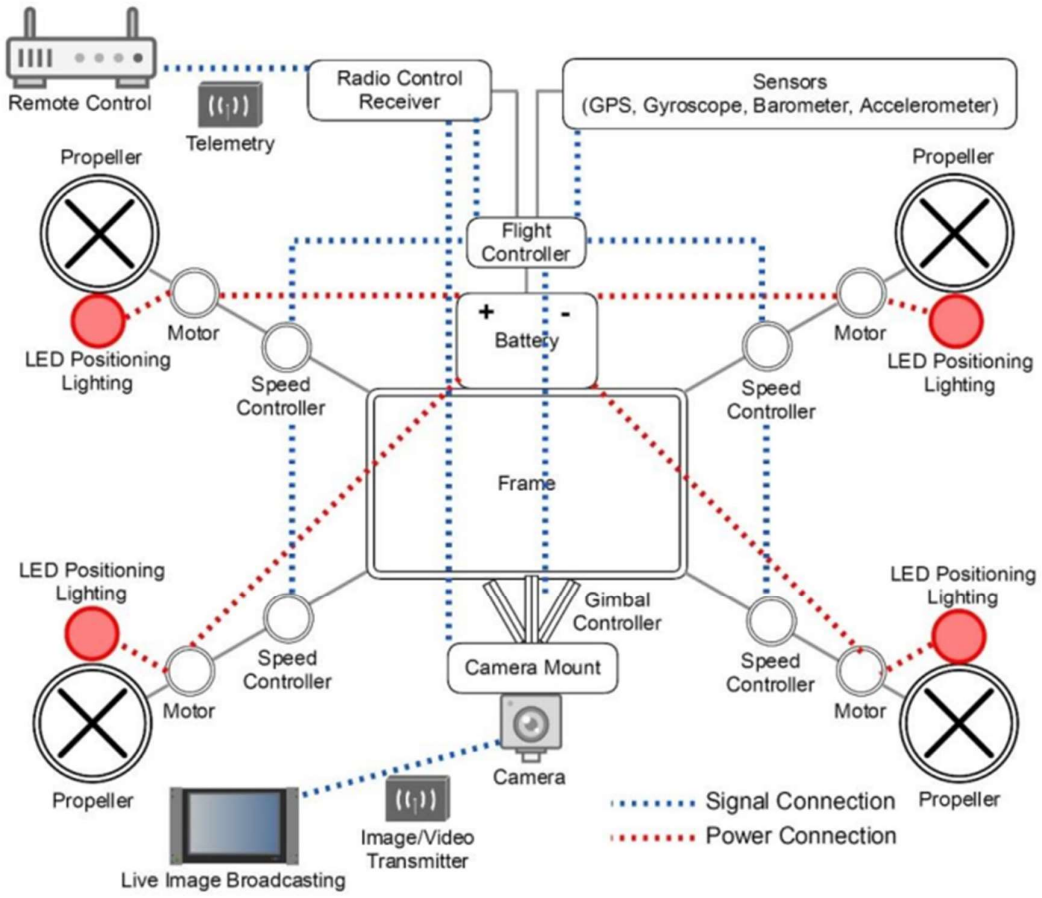


*Рис. 2.2 Основні складові квадрокоптеру [9]*

На малюнку можна побачити, що будь-який FPV квадрокоптер складається з 4 двигунів, камери, польотного контроллеру, додаткових сенсорів (таких як гіроскоп, барометр, акселерометр) та окремих регуляторів обертів (ESC). Також повинна бути система відеосигналу (VTX) та система радіосигналу (RX). На основі цих знань, будуються таблиці баз данних для конструювання квадрокоптерів, а також змінюючи деякі з них можна змінювати основні характеристики квадрокоптерів (такі як дальність та можлива підйомна вага квадрокоптеру) [7].

## Висновки до розділу 2

У другому розділі були розглянуті методи та технології які застосовуються при розробці систем інтерактивних елементів у Front-end та Back-end взаємодії, розглянуті детально підходи у розробці кожні з них та обгрунтовано чому варто використовувати відповідні інструменти для розробки відповідних рішень.

Детально приділена увага розгляду важливих архітектурних рішень у підборі таких важливий бібліотек для оформлення та створення систем для Front-end частини веб додатку, та оглянуто кожен з них, порівнявши всі переваги та недоліки, детально приділена увага таким інструментах як середовища збірки, менеджеру залежностей, а також бібліотек для створення компонентів користувацького інтерфейсу.

Також були досліджені основні складові для побудови квадрокоптерів, на основі отриманих даних будуватиметься система для створення необхідних квадрокоптерів в залежності від параметрів.

Також було детально розглянуто підхід при побудові серверної частини веб додатку, порівняно різні типи баз даних, огляд рішень для авторизації, та маршрутизації запитів.

# РОЗДІЛ 3

# РОЗРОБКА ІНТЕРАКТИВНОЇ СИСТЕМИ

## 3.1 Структура проекту

Структура проекту (рис. 3.1) складається з директорії для клієнтської частини додатку (Front-end частини) та інших компонентів які стосуються серверної частини веб-додатку (Back-end частини). В client директорії міститься папка з усіма встановленими залежностями для проекту (node-modules), директорія public з додатковими файлами, такими як іконками для відображення в вікні браузера, та основним файлом в якому буде відображатися кінцевий index.html файл. Та папка для вихідного коду Front-end частини (src). Також є файл package.json який містить інформацію про проект, включаючи його залежності, налаштування та скрипти для автоматизації різних завдань.

Серверна частина складається з  папки config, яка містить 2 файли, db.js – це файл для підключення до бази данних, файл default.json містить в собі JSON об’єкт з двох полів, перше mongoURI – шлях для підключення до бази данних, та jwtSecret – це секретний ключ для налаштування jwt токену.

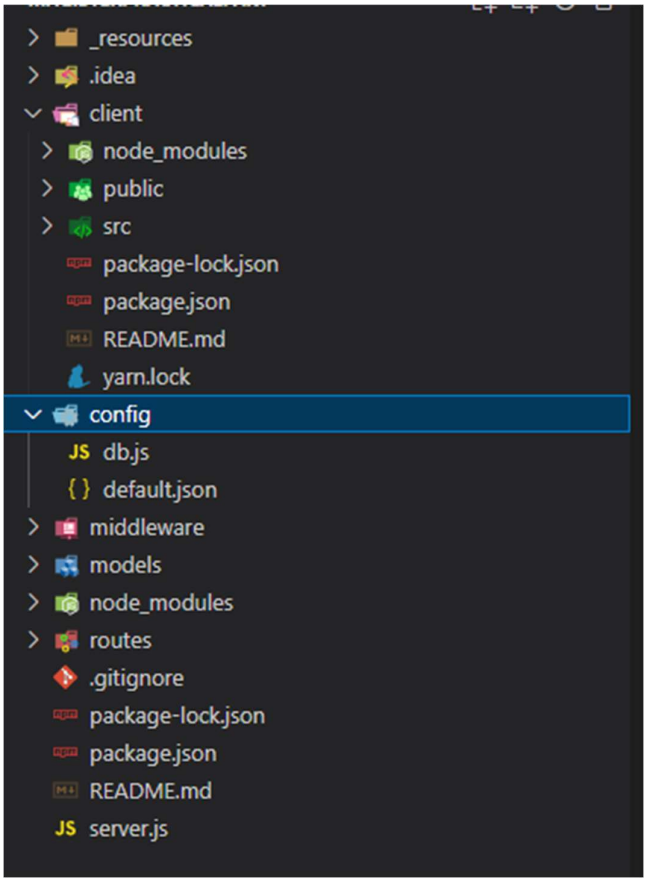


*Рис.3.1 Структура проекту*

### 3.2 Розробка серверної частини проекту

Для серверної частини використовується середа NodeJS, маршрутизація запитів відбувається за допомогою бібліотеки ExpressJs, а данні зберігаються в СУБД MongoDB. Отже розглянемо кожну з цих частин по черзі.

#### 3.2.1 Маршрутизація запитів

Шляхи для маршрутизації для додатку наведено в файлі server.js (рис. 3.2), всередині якого відбувається імпортування всіх інших Express.js роутерів. Всього в проекті використовується 5 роутерів: auth, posts, profile, quadConfiguration, users. Отже розглянемо їх більше детально, та перелік дій які можна виконувати в кожному з них.

README.md | server.js | auth.js

```
const express = require('express');
const connectDB = require('./config/db');
const path = require('path');

const app = express();

// Connect to Database
connectDB();

// Initialize Middleware
1 reference
app.use(express.json({ strict: false }));

// Define Routes
app.use('/api/users', require('./routes/api/users'));
app.use('/api/auth', require('./routes/api/auth'));
app.use('/api/profile', require('./routes/api/profile'));
app.use('/api/posts', require('./routes/api/posts'));
app.use('/api/quadConfiguration', require('./routes/api/quadConfiguration'));

// Serve Static assets in production
if (process.env.NODE_ENV === 'production') {
  // Set Static Folder
  app.use(express.static('client/build'));

  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
  });
}

// SERVER
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server started on PORT ${PORT}`));
```

*Рис. 3.2 Основний файл для серверної маршрутизації*

1. api/auth – використовується для отримання отримання інформації про користувача за домогою запиту GET, в запиті передається id користувача, а також метод POST в якому відбувається спочатку валідація чи введені обовязкові поля: електронна пошта та пароль, і якщо такий користувач існує (перевіряється на наявність користувача по полю email в таблиці users) і за допомогою функції bcrypt.compare відбувається порівняння гешованого паролю зі збереженим хешем паролю в базі даних. Якщо порівняння пройшло успішно, користувачу надається оновлений jwtToken, і користувач успішно завершує процес авторизації

2. api/users – використовується для реєстрації нового користувача, всередині роутера є тільки один метод POST в якому спочатку відбувається валідація обов’язкових полів, потім відбувається пошук на унікальність email з запиту в базі данних Users, якщо користувача з таким самою електронною поштою раніше не існувало, за допомогою функції bcrypt.genSalt() відбувається створення "солі" (salt) для хешування паролю за допомогою bcrypt. Сіль - це випадковий рядок, який додається до паролю перед його хешуванням. Вона ускладнює атаки, які використовують таблиці райдужних хешів (rainbow tables) і забезпечує більшу безпеку для збереження паролів у хешованому вигляді. Потім новий користувач зберігається в базу данних і йому створюється новий jwt токен який надсилається на клієнт щоб новостворений користувач зміг успішно авторизуватися.
3. api/profile – використовується для більшості дій з користувачем, а саме для отримання інформації про користувачів, додавання/видалення квадрокоптерів та додавання / перегляд інформації про участь в змагання.

   GET /me - Отримує профіль користувача, який авторизувався Перевіряє, чи існує профіль для користувача. Повертає профіль користувача з деякою додатковою інформацією.

   POST / - Створення або оновлення профілю користувача. Перевіряє валідаційні помилки. Збирає дані з запиту для створення або оновлення профілю. Якщо профіль існує, він оновлюється. В іншому випадку, створюється новий.

   GET /: - Отримання всіх профілів. Повертає всі профілі, які знаходяться в базі даних.

   GET /user/:user_id Отримання профілю за ID користувача. Повертає профіль для конкретного користувача.

DELETE /: Видалення профілю, користувача та пов'язаних з ними дописів. Видаляє всі пов'язані дописи. Видаляє профіль. Видаляє користувача.

PUT /quad: Додавання даних про квадрокоптер до профілю. Перевіряє валідаційні помилки. Додає дані про квадрокоптер до профілю.
DELETE /quad/:exp_id: Видалення даних про квадрокоптер з профілю. Знаходить і видаляє вказаний квадрокоптер.
PUT /competition: Додавання даних про участь в змаганнях до профілю. Перевіряє валідаційні помилки. Додає дані про конкурс до профілю.
DELETE /competition/:comp_id: Видалення даних користувача про участь в змаганнях з профілю. Знаходить і видаляє вказане змагання з таблиці змагань.

4. api/posts – використовується для взаємодії з постами користувачів, додавання, видалення, та для вподобання улюбених постів.
   POST /api/posts - Перевіряє валідаційні помилки (текст не може бути пустим). Створює новий допис з інформацією про автора. Зберігає допис у базі даних і повертає його.
   GET /api/posts - Знаходить всі дописи та сортує їх за датою.
   GET /api/posts/:id - Знаходить допис за його унікальним ідентифікатором.
   DELETE /api/posts/:id Перевіряє, чи існує допис з вказаним ідентифікатором. Перевіряє, чи поточний користувач є автором цього допису після цього видаляє допис.
   PUT /api/posts/like/:id: Перевіряє, чи користувач не ставив вже лайк цьому допису. Додає лайк до допису.
   PUT /api/posts/unlike/:id: - Перевіряє, чи користувач вже поставив лайк цьому допису. Видаляє лайк з допису.
   POST /api/posts/comment/:id: - Перевіряє валідаційні помилки (текст не може бути пустим). Створює новий коментар та додає його до допису.

DELETE /api/posts/comment/:id/:comment_id - Знаходить допис за його унікальним ідентифікатором. Перевіряє, чи коментар існує в дописі. Перевіряє, чи поточний користувач є автором цього коментаря. Видаляє коментар з допису.

5. api/quadConfiguration – містить в собі один POST запит /getConfig на основі якого відбувається знаходження найближчого до заданих в запиті конфігурації квадрокоптеру в базі данних quadsconfigs і повертає перше, найбільш підходящий до запиту результат.

### 3.2.2 Розробка структури таблиць в базі данних

Загалом в проекті використовується 4 різні таблиці (collections) баз данних.

Posts – використовується для збереження постів користувача (Таблиця 3.1).

*Таблиця 3.1*

*Таблиця Posts в базі даних*

| **Тип поля** | **Назва поля** | **Опис поля** | **Значення за Замовчуванням** |
|---|---|---|---|
| ObjectId (Reference) | user | Посилання на користувача в базі даних | - |

*Продовження таблиці 3.1*

| **Колонка 1** | **Колонка 2** | **Колонка 3** | **Колонка 4** |
|---|---|---|---|
| String (Обов'язкове) | text | Текст допису | - |
| String | name | Ім'я користувача (необов'язкове) | null |
| Array of Objects | likes | Масив з посиланнями на користувачів | [] |
| Array of Objects | comments | Масив коментарів | [] |

| Колонка 1 | Колонка 2 | Колонка 3 | Колонка 4 |
|---|---|---|---|
| Тип поля | Назва поля | Опис поля | Значення за Замовчуванням |
| Date | date | Дата створення допису | Поточна дата та час створення |

Profiles – таблиця використовується для додаткових данних про користувача, містить в собі інформацію в яких змаганнях брав участь користувач та якими квадрокоптерами пілотує. (Таблиця 3.2)

*Таблиця 3.2*

*Таблиця Profiles в базі даних*

| **Тип поля** | **Назва поля** | **Опис поля** | **Значення за Замовчуванням** |
|---|---|---|---|
| ObjectId (Reference) | user | Посилання на користувача в базі даних (таблицю user) | - |

*Продовження таблиці 3.2*

| **Колонка 1** | **Колонка 2** | **Колонка 3** | **Колонка 4** |
|---|---|---|---|
| String | rotorbuilds | Посилання на профіль в rotorbuilds | null |
| String | website | Веб-сайт користувача | null |
| String | location | Місцезнаходження користувача | null |
| String | status | Статус користувача | - |

| | | | |
|---|---|---|---|
| Array of Strings | videoType | Відеосистеми на яких літає пілот | [] |
| String | bio | Коротка біографія користувача | null |
| Array of Objects | quads | Інформація про квадрокоптери користувача | [] |
| Array of Objects | competitions | Інформація про змагання | [] |
| Тип поля | Назва поля | Опис поля | Значення за Замовчуванням |
| Object (вкладений) | social | Посилання на соціальні мережі користувача | Значення для кожного ключа – null |
| Date | date | Дата створення профілю | Поточна дата та час створення |

quadsConfigs (таблиця 3.3) – це таблиця яка використовується для конфігурації квадрокоптерів (вказуючи дальність польоту і відстань відбирається найближче значення з цієї таблиці)

*Таблиця 3.3*

*Таблиця quadsConfigs в базі данних*

| **Тип поля** | **Назва поля** | **Опис поля** | **Значення за замовчуванням** |
|---|---|---|---|
| Number | distance | Максимальна дальність польоту | - |

| | | | |
|---|---|---|---|
| Number | Weight | Підйомна вага квадрокоптеру | - |
| String | propSize | Розмір пропеллерів | - |
| String | motor | Назва мотору | - |
| String | frame | Назва рами | - |
| String | vtx | Назва відеопередавача | - |
| String | vtxAntenna | Назва відеоантенни | - |
| String | flightController | Назва польотного контроллеру | - |
| Тип поля | Назва поля | Опис поля | Значення за замовчуванням |
| String | esc | Назва регулятору обертів | - |
| String | receiver | Назва приймача | - |
| String | camera | Назва FPV камери | - |

Таблиця user – предназначена для зберігання освновних данних про користувача, такі як логін, захешований пароль, та дата створення аккаунту. (Таблиця 3.4)

*Таблиця 3.4*

*Таблиця user в базі данних*

| **Тип поля** | **Назва поля** | **Опис поля** | **Значення за замовчуванням** |
|---|---|---|---|
| String (required) | name | Ім’я користувача | - |
| String (required) | email | email (використовується як логін для авторизації) | - |
| String | avatar | Посилання на фотографію користувача | - |
| String (required) | password | Захешований пароль користувача | - |

### 3.3 Розробка клієнтської частини проекту

Клієнтська частина проекту зроблена за допомогою бібліотеки користувацького інтерфейсу React, в поєднанні з маршрутизацією за допомогою React-router, компоненти користувацького інтерфейсу реалізовані за допомогою синтаксису JSX. Користувачу доступні наступні сторінки:
/ - головна сторінка з якої користувач може зайти в профіль, зареєструватися, або переглянути інших пілотів

/quad-constructor – сторінка для підбору необхідних комплектуючих для квадрокоптерів

/profiles – сторінка на якій можна переглянути список всіх пілотів

/posts – сторінка на якій можна писати дописи та обмінюватися власними думками

/dashboard – сторінка особистого профілю користувача, в якій він може переглянути та доповнити особисті данні

/login – сторінка для авторизації користувача

/register – сторінка для реєстрації користувача

Для нового користувача доступні тільки посилання на сторінки /register та /login, до інших в них відкриється доступ виключно після успішної реєстрації

Всі клієнтські шляхи для маршрутизації знаходять в основному файлі для клієнтської частини App.js (рис 3.3), в ньому відбувається перевірка чи знаходиться jwtToken в локальному сховищі браузера (на випадок якщо користувач вже заходив раніше), а також відбуваєтсья локальна маршрутизація на сторінки.

Імпортуються всі стилі які використовуються для стилізації додатку (всі основні стилі прописані в файлі App.css) та відбувається ініціалізація основного локального сховища в межах  store яким оперує Redux бібліотека. Завдяки Redux можна викликати перелік різних дій, такі як отримати всі пости, додати пости, переглянути профілі. Перелік всіх дій які взаємодії з глобальних сховищем данних зберігається в окремій папці actions. Всі компоненти користувача зберігаються в папці components де в кожній папці назва якої є відповідна сторінка на користувацькому інтерфейсі містить в собі всі необхідні компоненти для побудови робочої системи інтерактивних елементів.

В кожному компоненті містяться файли для рендерингу відповідного інтерфейсу, а також можливі додаткові файли для обробки подій та взаємодії з Redux стором.

Для прикладу, компонент "PostList" містить файли для відображення списку постів, а також файли для взаємодії з Redux, такі як дії для отримання списку постів, видалення постів тощо.

До кожного компоненту можуть також додаватися відомості про його стилізацію та інші особливості відображення.

Все це дозволяє ефективно організовувати код та спрощує його розширення та підтримку в майбутньому.

```
const App = () => {
  useEffect(() => {
    // check for token in LS when app first runs
    if (localStorage.token) {
      // if there is a token set axios headers for all requests
      setAuthToken(localStorage.token);
    }
    // try to fetch a user, if no token or invalid token we
    // will get a 401 response from our API
    store.dispatch(loadUser());

    // log user out from all tabs if they log out in one tab
    window.addEventListener('storage', () => {
      no references found for type
      if (!localStorage.token) store.dispatch({ type: LOGOUT });
    });
  }, []);

  return (
    <Provider store={store}>
      <BrowserRouter>
        <Navbar />

        <Alert />

        <Routes>
          <Route path='/' element={<Landing />} />
          <Route path='register' element={<Register />} />
          <Route path='login' element={<Login />} />
          <Route path='profiles' element={<Profiles />} />
          <Route path='quad-constructor' element={<QuadConstructor />}/>
          <Route path='profile/:id' element={<Profile />} />
          <Route
            path='dashboard'
            element={<PrivateRoute component={Dashboard} />}
          />
          <Route
            path='create-profile'
            element={<PrivateRoute component={ProfileForm} />}
          />
          <Route
            path='edit-profile'
            element={<PrivateRoute component={ProfileForm} />}
          />
          <Route
            path='add-quad'
            element={<PrivateRoute component={AddQuad} />}
          />
          <Route
            path='add-competition'
            element={<PrivateRoute component={AddCompetition} />}
          />
          <Route path='posts' element={<PrivateRoute component={Posts} />} />
          <Route path='posts/:id' element={<PrivateRoute component={Post} />} />
          <Route path='/*' element={<NotFound />} />
        </Routes>
      </BrowserRouter>
    </Provider>
  );
};

export default App;

```

*Рис. 3.3 – Основний клієнтський файл App.js*

Всі шляхи використовують компоненти для маршрутизації <Routes> з вкладеними в себе компонентами <Route> які введуть окремо на кожні з сторінок користувацького інтерфейсу.

На клієнтській частині багато данних зберігаються за допомогою бібліотеки Redux, вона використовує концепцію однозначного дерева стану та прогнозованого невідмінного змінення цього стану через "actions" (екшини) та "reducers" (редюсери). Редюсер - це чиста функція, яка приймає поточний стан та екшин, і повертає новий стан.

Редюсери визначають, як саме має бути змінений стан додатку відповідно до отриманих екшинів.

Кожен редюсер відповідає за певну частину стану додатку (наприклад, один редюсер може відповідати за список задач, інший - за фільтр).

Приклад редюсера в застосунку можна побачити на рис 3.4 де наведено редюсер для авторизації. Екшини - це об'єкти, які описують зміну стану застосунку. Вони містять тип (type) та, можливо, додаткові дані (payload), які вказують, як саме змінився стан.

Екшини є важливою частиною у взаємодії користувача з додатком, подіями в додатку чи іншими асинхронними операціями.

Основна частина редюсера містить великий оператор switch, який обробляє різні типи екшинів. В залежності від типу екшину, відбуваються різні зміни стану.

USER_LOADED: Якщо отримано екшин USER_LOADED, стан оновлюється з інформацією про користувача, отриману з екшину.

REGISTER_SUCCESS та LOGIN_SUCCESS: Якщо отримано екшини REGISTER_SUCCESS або LOGIN_SUCCESS, стан оновлюється із спільними властивостями, які містяться в payload, а також встановлюються isAuthenticated та loading в true.

AUTH_ERROR, LOGOUT та ACCOUNT_DELETED: Якщо отримано будь-який із цих екшинів, стан оновлюється з властивостями, що вказують на втрату аутентифікації, завантаження та встановлення token та user в null.

default: Якщо тип екшину не відомий, повертається поточний стан без змін.

README.md | server.js | auth.js | Profile.js | Navb

```
import {
  REGISTER_SUCCESS,
  USER_LOADED,
  AUTH_ERROR,
  LOGIN_SUCCESS,
  LOGOUT,
  ACCOUNT_DELETED,
} from '../actions/types';

const initialState = {
  no references found for token
  token: localStorage.getItem('token'),
  no references found for isAuthenticated
  isAuthenticated: null,
  no references found for loading
  loading: true,
  no references found for user
  user: null,
};

const authReducer = (state = initialState, action) => {
  const { type, payload } = action;

  switch (type) {
    case USER_LOADED:
      return {
        4 references
        ...state,
        no references found for isAuthenticated
        isAuthenticated: true,
        no references found for loading
        loading: false,
        no references found for user
        user: payload,
      };
    case REGISTER_SUCCESS:
    case LOGIN_SUCCESS:
      return {
        4 references
        ...state,
        2 references
        ...payload,
        no references found for isAuthenticated
        isAuthenticated: true,
        no references found for loading
        loading: false,
      };
    case AUTH_ERROR:
    case LOGOUT:
    case ACCOUNT_DELETED:
      return {
        4 references
        ...state,
        no references found for token
        token: null,
        no references found for isAuthenticated
        isAuthenticated: false,
        no references found for loading
        loading: false,
        no references found for user
        user: null,
      };
    default:
      return state;
  }
};

export default authReducer;
```

*Рис 3.4 Редюсер authReducer*

### 3.3 Огляд функціоналу по отриманню конфігурації квадрокоптерів

Основним функціоналом данного проекту який відділяє його від конкурентів є наявність інтерактивного конструктору по підбору необіхдних

комплектуючих для квадрокоптеру для виконання певних задач. Зараз основною задачою для якою більшість дронів використовується є військова компонента, і основним характеристиками є максимальна дальність польоту і максимальна підйомна вага.

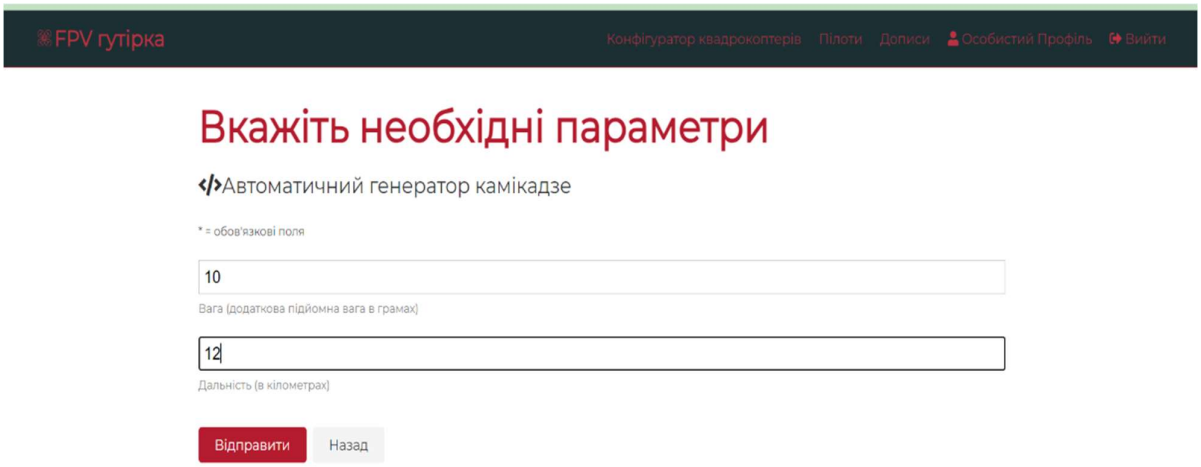


*Рис 3.5 Демонстрація функціоналу заповнення необхідних полів*

Користувачу достатньо заповнити необхідні поля (Рис. 3.5) і натиснути кнопку «Відправити» після чого система надасть перелік комплектуючих необхідних найближчих до запрошуваних користувачем, список завжди можна доповнити новими конфігураціями які зберігаються в окремій базі данних (quadsConfigs).

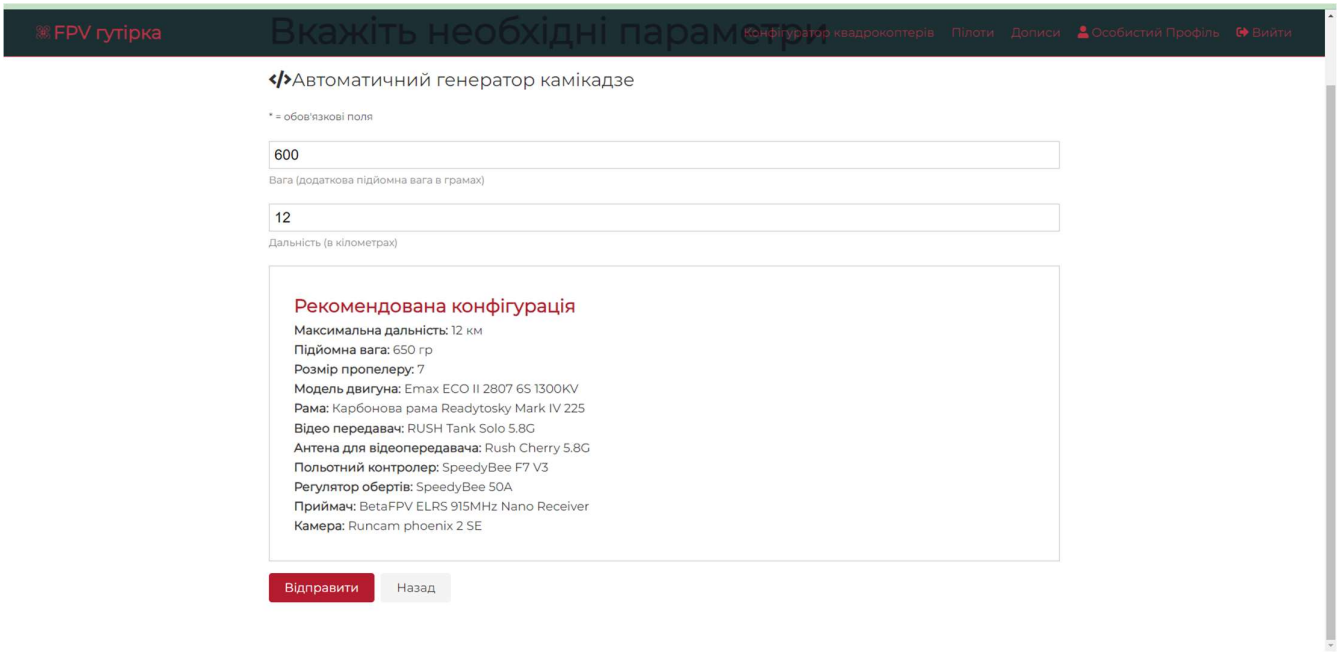


*Рис 3.6 Демонстрація отриманної відповіді від системи*

На рис. 3.6. показана відповідь від системи, наданий перелік є повний необхідний перелік необхідних запчастин з яких можна зробити квадрокоптер для виконання задач на необхідні вказані параметри. Данний метод є новітнім і немає в жодних соціальних мережах, оскільки ця система є спеціалізованою в ній є можливість реалізувати спеціалізований функціонал який предназначений саме для пілотів.

### 3.4 Огляд профілю пілота

В функціонал розробленого додатку також закладена можливість створювати особистий профіль пілоту, в якому він може заповнювати додаткові данні про себе, і також потім інші пілоти можуть переглядати данні про нього через сторінку «Пілоти». На основній сторінці, власник профілю може вносити зміни у профіль, додати або видалити зі свого списку квадрокоптери, додати інформацію про участь в нових змаганнях, а також є можливість взагалі видалити профіль. (рис. 3.7)

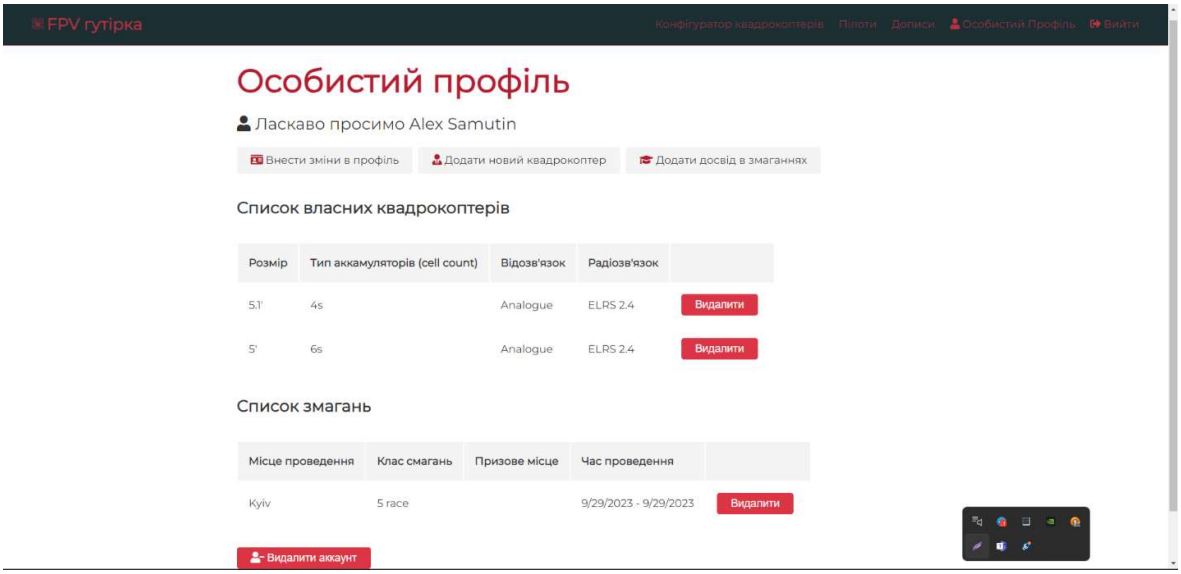


*Рис 3.7 Особистий профіль користувача*

Особистий профіль є основною візиткою користувача, адже інші користувачі будуть бачити основну інформацію яка буде вказана користувачем.

## Висновки до розділу 3

В цьому розділі було розглянуто основі рішення за допомогою яких була створена система інтерактивних елементів. Було детально розглянуто структуру проекту.

Описано як додаток був реалізований на стороні серверної частини, детально приділена увага структурі проекту та таблиці данних з якими сервер взаємодіє.

Окремо було детально як система взаємодіє на рівні користувача (Front-end). Основні можливості маршрутизаці на клієнті та перелік всіх можливих сторінок на які може переходити користувач.

Було розглянуто новітні можливості проекту по конфігурації квадрокоптерів під задані умови користувачем, та особистий профіль користувача

# Розділ 4

# РОЗРОБКА СТАРТАП ПРОЕКТУ

## 4.1 Опис ідеї проекту

FPV гутірка - це інноваційна онлайн-платформа, призначена для об'єднання пілотів квадрокоптерів у спільноту та надання їм можливостей для спілкування, обміну досвідом та участі у захоплюючих змаганнях. Проєкт виникає з усвідомлення росту популярності безпілотників та потреби в мережі для спілкування серед ентузіастів.

Головні можливості FPV гутірки включають:

- Створення особистих профілів: Пілоти можуть створити власні профілі, додати інформацію про себе, свої досягнення та обрані моделі квадрокоптерів.
- Обмін досвідом та знаннями: У спільноті пілотів можна обговорювати технічні особливості, виклики та рішення, що стосуються польотів на квадрокоптерах.
- Змагання та виклики: FPV гутірка надає можливість участі у власних та спільних змаганнях, що сприяє підвищенню майстерності та додатковому заохоченню.
- Онлайн-форуми: Для зручного спілкування та обговорення різноманітних аспектів пілотування квадрокоптерів.
- Мобільність: Платформа доступна на різних мобільних пристроях, що дозволяє пілотам бути на зв'язку навіть під час польотів.
- FPV гутірка прагне зробити пілотування квадрокоптерів більш соціальним та захоплюючим, створюючи простір для обміну досвідом та створення нових можливостей для спільного вдосконалення навичок.

### 4.2 Технічна інфраструктура

Frontend: Використання React для розробки користувацького інтерфейсу та Redux для управління станом додатку.

Backend: Використання Node.js з фреймворком Express.js для реалізації серверної частини.

База Даних: Використання MongoDB для зберігання інформації про користувачів, події та інші дані.

Аутентифікація: Використання бібліотеки jsonwebtoken для реалізації безпечної авторизації.

Хмарні Сховища: Використання AWS або Google Cloud для зберігання великих медіа-файлів.

### 4.3 Технологічний аудит ідеї проекту

1. Оцінка Технічної Спроможності: Визначення, чи маємо достатній рівень експертизи в області веб-розробки та розробки мобільних додатків для успішної реалізації цього проекту.

2. Вибір Технологій та Фреймворків: Аналіз сучасних технологій та фреймворків, які найкраще підходять для реалізації "FPV гутірки". Врахування ефективності, швидкодії та масштабованості.

3. Безпека та Конфіденційність: Оцінка можливих загроз безпеці та розробка стратегії для забезпечення конфіденційності даних користувачів.

4. Масштабування та Витривалість: Визначення, як система буде масштабуватися зі зростанням кількості користувачів та забезпечення надійності роботи.

5. Хмарні Технології: Розгляд можливостей використання хмарних сервісів для зберігання даних та виконання обчислень.

6. Забезпечення Сумісності: Впевнення, що платформа буде сумісною з різними пристроями та операційними системами.

7. Тестування та Оптимізація: Розробка плану тестування для виявлення та виправлення можливих помилок та оптимізація швидкодії системи.

8. Реалізація Монетизації: Вибір та налаштування механізмів монетизації проекту.

9. Стратегія Розширення: Розробка плану для подальшого розширення функціоналу та можливостей "FPV гутірки".

10. Оцінка Загальних Витрат: Проведення аналізу та оцінка витрат на розробку та запуск проекту.

11. Прогнозування Термінів: Визначення очікуваних термінів розробки та виходу на ринок.

12. Оцінка Ризиків: Визначення можливих ризиків та розробка стратегій для їх управління.

13. Після проведення технологічного аудиту, ми прийшли до висновку, що реалізація проекту "FPV гутірки" є технічно можливою та обіцяючою. Важливо приділити увагу безпеці, масштабованості та оптимізації системи для забезпечення якісної роботи та задоволення потреб користувачів.

### 4.4 Аналіз ринку

Огляд Галузі: Вивчення ринку додатків для пілотів квадрокоптерів та соціальних платформ для спілкування в цій галузі. Виявлення конкурентів та їхніх ключових характеристик.

Цільова Аудиторія: Визначення та аналіз цільової аудиторії "FPV гутірки". Розуміння їхніх потреб, проблем та очікувань від соціальної мережі для пілотів дронів.

Тенденції та Інновації: Вивчення найновіших тенденцій у сфері розробки додатків та соціальних мереж, врахування можливостей для інновацій та вдосконалення концепції.

Ринкова Лакуна: Визначення можливих прогалин у сервісах, які надають конкуренти. Виділення ключових функцій, які можуть зробити " FPV гутірки" унікальним.

Монетизація та Бізнес-Модель: Аналіз можливих шляхів отримання прибутку, включаючи преміум-плани, рекламні програми та партнерські угоди.

Партнерські Відносини: Розгляд можливостей для співпраці з виробниками обладнання, організаторами змагань та іншими партнерами для спільного розвитку та просування проекту.

Ризики та Вирішення: Аналіз можливих ризиків, які можуть вплинути на успішність проекту, та розробка стратегій для їх управління.

Оцінка Обсягу Ринку: Визначення потенційного кількісного та грошового обсягу ринку для "FPV гутірки".

Після проведення аналізу ринку, ми виявили, що існує значний попит на платформу для спілкування пілотів квадрокоптерів. Конкуренція на ринку присутня, але "FPV гутірки" може вирізнятися завдяки своїй спеціалізації та унікальним можливостям. Важливо вдосконалити стратегію маркетингу та реклами для привертання цільової аудиторії. Загалом, ринковий аналіз підтверджує потенціал успіху "FPV гутірки" і можливість його вдосконалення для задоволення потреб користувачів.

### 4.5 Бізнес модель

Бізнес-модель "FPV гутірки" передбачає створення цінності для пілотів квадрокоптерів, надаючи їм можливість взаємодіяти, обмінюватися досвідом та покращувати свої навички. Інноваційний підхід та спеціалізація робить цей продукт привабливим для цільової аудиторії. Орієнтація на розвиток спільноти та співпрацю з організаторами змагань сприяє подальшому росту та розвитку проєкту.

### 4.5.1 Цінність Продукту

"FPV гутірка" надає унікальну платформу для спілкування пілотів квадрокоптерів, що дозволяє обмінюватися досвідом, враженнями та знаннями у цій специфічній галузі.

Забезпечує можливість створення профілів, додавання власних дронів, запис на змагання та участь в обговореннях.

Допомагає пілотам розвивати свої навички та покращувати власну практику.

### 4.5.2 Інноваційність

"FPV гутірка" є інноваційним продуктом, оскільки на сьогоднішній день не існує аналогічної спеціалізованої платформи для пілотів квадрокоптерів.

Впроваджує нові можливості для спілкування та обміну досвідом у цій галузі.

### 4.5.3 Сегмент Споживачів

Початкові та досвідчені пілоти квадрокоптерів, які мають інтерес до спілкування з однодумцями, обговорення нових моделей дронів, та участі в змаганнях.

Команди та організатори змагань, які шукають платформу для просування своїх подій та взаємодії з учасниками.

## 4.6 Розроблення маркетингової програми

Маркетингова програма спрямована на акцентування уваги на ключових перевагах "FPV гутірки" в порівнянні з конкурентами, що дозволить привернути увагу цільової аудиторії та створити попит на продукт.

Маркетингова програма стартап-проєкту наведена в таблиці 4.1

*Таблиця 4.1*

*Маркетингова програма*

| Потреба | Вигода від товару | Ключові переваги над конкурентами |
|---|---|---|
| Об'єднання пілотів квадрокоптерів | Створення спільноти та обмін досвідом | Єдине спеціалізоване спільнота для пілотів квадрокоптерів |
| Можливість створення власного профілю | Особистий кабінет та участь в обговореннях | Можливість додавати та рекламувати власні моделі |
| Можливість записатися на змагання | Швидкий доступ до актуальних подій | Календар змагань та можливість взяти участь в них |
| Участь в обговореннях та форумах | Обмін досвідом та отримання порад | Активна спільнота користувачів та експертів для комунікації |
| Зручний пошук та фільтрація за інтересами | Швидке знаходження цікавих дописів | Розширені фільтри для точного підбору контенту |
| Можливість відкрито ділитися досвідом | Сприяння розвитку та покращення навичок | Стимулювання активності та взаємодопомога в спільноті |
| Зручний інтерфейс для мобільних пристроїв | Можливість використання в дорозі | Адаптивний дизайн для комфортного використання на будь-якому пристрої |

## Висновки до розділу 4

Розроблений стартап проєкт "FPV гутірка" спрямований на об'єднання пілотів квадрокоптерів у єдину спільноту, надаючи їм можливість обміну досвідом, участі у змаганнях та спілкування в онлайн-форумах. Проєкт має значний потенціал для розвитку та привертає увагу аудиторії, зацікавленої у польотах на безпілотних літальних апаратах. Ключові переваги "FPV гутірка" включають зручний інтерфейс, можливість додавання власних апаратів, участь у змаганнях та активну спільноту. Крім того, платформа надає можливість взаємодії на мобільних пристроях та сприяє взаємному навчанню та підтримці.

Стартап також має чітку бізнес-модель, що включає партнерські угоди з організаторами змагань та можливість рекламування товарів для безпілотників. Завдяки інноваційному підходу та активній підтримці користувачів, "FPV гутірка" має всі передумови для успіху на ринку.

Маркетингова програма спрямована на підкреслення унікальних переваг платформи та створення попиту серед цільової аудиторії. Це включає в себе активний контент для спільноти, спеціальні угоди для партнерів та ефективну рекламну кампанію.

У загальному, стартап "FPV гутірка" є перспективним проєктом, що має потенціал стати важливим гравцем у сфері пілотування квадрокоптерів та надати значний внесок у розвиток цієї галузі.

# ВИСНОВКИ

В цій магістерській роботі було розроблено систему інтерактивних елементів Front end – back end взаємодії у web застосунках на прикладі соціальної мережі для пілотів квадрокоптерів з функціоналом розрахунку потрібних комплектуючих для квадрокоптерів. Розроблена система задовольняє потреби пілотів, полегшує їх професійну взаємодію та поширенням знань в сфері квадрокоптерів. Основна частина дисертації містить 4 розділи.

У першому розділі було розглянуто інші соціальні мережі, розглянуто їхній функціонал та проведенно порівняння між ними, визначення основних потреб якими має володіти новостворена соціальна мережа. Враховуючи відсутність іншої спеціалізованої соціальної мережі під пілотів квадрокоптерів було розглянуто звичайні соціальні мережі під широке коло користувачів.

В другому розділі дисертації був присвячений огляду необхідних рішень для розробки власної соціальні соціальної мережі, огляд необхідних комплектуючих для розробки власного веб-конструктору для квадрокоптерів. В цьому розділі детально було розглянуто наступні питання:

1. Інструменти для розробки Front-End частини веб-додатку.
2. Інструменти для розробки Back-End частини веб-додатку.
3. Огляд необхідних деталей для розробки функціоналу підбору деталей квадрокоптеру в залежності від потреб.

В третьому розділі дисертації було розроблено веб-додаток. В рамках розділу було детально розглянуто розробку клієнтської та серверну частину, архітектуру веб-додатку, авторизацію, та розглянуто як технології розглянуті в розділі 2 були застовані при розброці системи інтерактивних елементів. Також було коротко оглянуто основний функціонал для підбору комплектуючих квадрокоптерів в залежності від заданих параметрів.

В четвертому розділі було розроблено стартап проект для обраної теми. В цьому розділі було детально розглянутий процес створення стартапу, а саме

розглянуто опис ідеї проекту, технічну інфраструктуру, розроблено технічний аудит проекту, проведено аналіз ринку, розроблено бізнес модель стартапу. В рамках бізнес моделі було розглянуто цінність продукту, інноваційність та сегмент споживачів. В кінці розділу була розроблена маркетингова програма для створеного стартапу.

Отже, розроблена соціальна мережа для пілотів квадрокоптерів, задовольняє основні потреби пілотів, має можливість підбирати комплектуючі для створення власних квадрокоптерів, та полегшує комунікацію між пілотами квадрокоптерів.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Wikipedia Facebook [Електронний ресурс] – Режим доступу до ресурсу: https://upload.wikimedia.org/wikipedia/en/6/64/Facebook_user_page.png
2. Yoast. Instagram for your business: a guide to getting started [Електронний ресурс] – Режим доступу до ресурсу: https://yoast.com/app/uploads/2020/10/Instagram-for-business-example-Nike-419x800.png
3. Quartz. Elon Musk changed Twitter's iconic logo to the letter X. [Електронний ресурс] – Режим доступу до ресурсу: https://i.kinja-img.com/image/upload/c_fit,q_60,w_1315/1fcd37d87fb36c56de981ef193ad9dab.jpg
4. Manuel Bieh, Sibylle Sehl. React Deep Dive, – С. 14 - 15
5. Alex Nordeen. Learn MongoDB in 24 Hours, – С. 22 - 23
6. Evan Hahn. Express in Action: Writing, building, and testing Node.js applications, – С. 32 - 41
7. Sebastion E. Peyrott. The JWT Handbook, – 2016. – С. 5-6
8. Bas Vergouw, Huub Nagel, Geert Bondt, Bart Clusters. The Future of Drone Use Opportunities and Threats From Ethical and Legal Perspectives – Springer, – 2016. – С. 21-22
9. Seungho Kim and Sangyong Kim, Opportunities for construction site monitoring by adopting first personal view (FPV) of a drone [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/profile/Seungho-Kim-5/publication/341566774_Opportunities_for_construction_site_monitoring_by_adopting_first_personal_view_FPV_of_a_drone/links/5ec77506458515626cbf48a8/Opportunities-for-construction-site-monitoring-by-adopting-first-personal-view-FPV-of-a-drone.pdf

10. Christian M. Mollica, FPV Flight Dynamics: Mastering Acro Mode on High-Performance Drones – C. 19-22

# ДОДАТОК

## Лістинг фрагментів коду програми

- Основний файл серверної частини server.js

```
const express = require('express');
const connectDB = require('./config/db');
const path = require('path');
const app = express();
// Connect to Database
connectDB();
// Initialize Middleware
app.use(express.json({ strict: false }));
// Define Routes
app.use('/api/users', require('./routes/api/users'));
app.use('/api/auth', require('./routes/api/auth'));
app.use('/api/profile', require('./routes/api/profile'));
app.use('/api/posts', require('./routes/api/posts'));
app.use('/api/quadConfiguration', require('./routes/api/quadConfiguration'));
// Serve Static assets in production
if (process.env.NODE_ENV === 'production') {
  // Set Static Folder
  app.use(express.static('client/build'));
  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
  });
}
// SERVER
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server started on PORT ${PORT}`));
```

- Основний файл клієнтської частини App.js

```
import { useEffect } from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Navbar from './components/layout/Navbar';
import Landing from './components/layout/Landing';
import Login from './components/auth/Login';
import Register from './components/auth/Register';
import Alert from './components/layout/Alert';
import Dashboard from './components/dashboard/Dashboard';
import ProfileForm from './components/profile-form/ProfileForm';
import AddQuad from './components/profile-form/AddQuad';
import AddCompetition from './components/profile-form/AddCompetition';
import Profiles from './components/profiles/Profiles';
import Profile from './components/profile/Profile';
import Posts from './components/posts/Posts';
import Post from './components/post/Post';
import NotFound from './components/layout/NotFound';
import PrivateRoute from './components/routing/PrivateRoute';
import QuadConstructor from './components/QuadConstructor/QuadConstructor';
// Redux
import { Provider } from 'react-redux';
import { store } from './store';
import { loadUser } from './actions/auth';
import { LOGOUT } from './actions/types';
import setAuthToken from './utils/setAuthToken';
import './App.css';
const App = () => {
  useEffect(() => {
    // check for token in LS when app first runs
```

```
  if (localStorage.token) {
    // if there is a token set axios headers for all requests
    setAuthToken(localStorage.token);
  }
  // try to fetch a user, if no token or invalid token we
  // will get a 401 response from our API
  store.dispatch(loadUser());

  // log user out from all tabs if they log out in one tab
  window.addEventListener('storage', () => {
    if (!localStorage.token) store.dispatch({ type: LOGOUT });
  });
}, []);

return (
  <Provider store={store}>
    <BrowserRouter>
      <Navbar />

      <Alert />

      <Routes>
        <Route path='/' element={<Landing />} />
        <Route path='register' element={<Register />} />
        <Route path='login' element={<Login />} />
        <Route path='profiles' element={<Profiles />} />
        <Route path='quad-constructor' element={<QuadConstructor />}/>
        <Route path='profile/:id' element={<Profile />} />
        <Route
          path='dashboard'
```

```
          element={<PrivateRoute component={Dashboard} />}
        />
        <Route
          path='create-profile'
          element={<PrivateRoute component={ProfileForm} />}
        />
        <Route
          path='edit-profile'
          element={<PrivateRoute component={ProfileForm} />}
        />
        <Route
          path='add-quad'
          element={<PrivateRoute component={AddQuad} />}
        />
        <Route
          path='add-competition'
          element={<PrivateRoute component={AddCompetition} />}
        />
        <Route path='posts' element={<PrivateRoute component={Posts} />} />
        <Route path='posts/:id' element={<PrivateRoute component={Post} />} />
        <Route path='/*' element={<NotFound />} />
      </Routes>
    </BrowserRouter>
  </Provider>
 );
};
export default App;
```

- Файл для підключення до бази данних

```
const mongoose = require('mongoose');
const config = require('config');
```

```
const db = config.get('mongoURI');

// Connection to MongoDB
const connectDB = async () => {
  try {
    await mongoose.connect(db, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      // useCreateIndex: true,
      // useFindAndModify: false,
    });
    console.log('MongoDB Connected...');
  } catch (err) {
    console.error(err.message);
    // Exit process with failure
    process.exit(1);
  }
};

module.exports = connectDB;
```

- Основний файл стилів App.css

```
@import
url('https://fonts.googleapis.com/css?family=Montserrat|Raleway&display=swap');
.m-1 {
  margin: 1rem;
}

.my-1 {
  margin: 1rem 0;
```

```
}

.p-1 {
  padding: 1rem;
}

.py-1 {
  padding: 1rem 0;
}

.m-2 {
  margin: 2rem;
}

.my-2 {
  margin: 2rem 0;
}

.p-2 {
  padding: 2rem;
}

.py-2 {
  padding: 2rem 0;
}

.m-3 {
  margin: 3rem;
}
```

```
.my-3 {
  margin: 3rem 0;
}

.p-3 {
  padding: 3rem;
}

.py-3 {
  padding: 3rem 0;
}

.m-4 {
  margin: 4rem;
}

.my-4 {
  margin: 4rem 0;
}

.p-4 {
  padding: 4rem;
}

.py-4 {
  padding: 4rem 0;
}

.m-5 {
  margin: 5rem;
```

```
}

.my-5 {
  margin: 5rem 0;
}

.p-5 {
  padding: 5rem;
}

.py-5 {
  padding: 5rem 0;
}

.container {
  max-width: 1100px;
  margin: auto;
  overflow: hidden;
  padding: 0 2rem;
  margin-top: 6rem;
  margin-bottom: 3rem;
}

.x-large {
  font-size: 4rem;
  line-height: 1.2;
  margin-bottom: 1rem;
}

.large {
```

```
  font-size: 3rem;
  line-height: 1.2;
  margin-bottom: 1rem;
}

.lead {
  font-size: 1.5rem;
  margin-bottom: 1rem;
}

.text-primary {
  color: #B51C2F;
}

.round-img {
  border-radius: 50%;
}

.line {
  height: 1px;
  background: #ccc;
  margin: 1.5rem 0;
}

.bg-primary {
  background-color: #B51C2F;
  color: #fff;
}

.bg-light {
```

```
  background-color: #f4f4f4;
  color: #333;
  border: #ccc 1px solid;
}

.bg-dark {
  background-color: #03191E;
  color: #fff;
}

.bg-success {
  background-color: #28a745;
  color: #fff;
}

.bg-danger {
  background-color: #dc3545;
  color: #fff;
}

.bg-white {
  background-color: #fff;
  color: #333;
  border: #ccc 1px solid;
}

.btn {
  display: inline-block;
  background: #f4f4f4;
  color: #333;
```

```
  padding: 0.4rem 1.3rem;
  border: none;
  cursor: pointer;
  font-size: 1rem;
  margin-right: 0.5rem;
  outline: none;
  -webkit-transition: all 0.2s ease-in;
  transition: all 0.2s ease-in;
  border-radius: 4px;
}

.btn.btn-primary {
  background-color: #B51C2F;
  color: #fff;
}

.btn.btn-primary:hover {
  background: #630f19;
}

.btn.btn-dark {
  background-color: #03191E;
  color: #fff;
}

.btn.btn-dark:hover {
  background: #4e433f;
}

.btn.btn-success {
```

```
  background-color: #28a745;
  color: #fff;
}

.btn.btn-success:hover {
  background: #2dbc4e;
}

.btn.btn-danger {
  background-color: #dc3545;
  color: #fff;
}

.btn.btn-danger:hover {
  background: #e04b59;
}

.btn:hover {
  background: #786762;
  color: #fff;
}

.alert {
  padding: 0.8rem;
  margin: 1rem;
  opacity: 0.9;
  background: #f4f4f4;
  color: #333;
}
```

```
.alert.alert-primary {
  background-color: #B51C2F;
  color: #fff;
}

.alert.alert-dark {
  background-color: #03191E;
  color: #fff;
}

.alert.alert-success {
  background-color: #28a745;
  color: #fff;
}

.alert.alert-danger {
  background-color: #dc3545;
  color: #fff;
}

.badge {
  font-size: 0.8rem;
  padding: 0.1rem;
  text-align: center;
  margin: 0.3rem;
  background: #f4f4f4;
  color: #333;
}

.badge.badge-primary {
```

```
  background-color: #B51C2F;
  color: #fff;
}

.badge.badge-dark {
  background-color: #03191E;
  color: #fff;
}

.badge.badge-success {
  background-color: #28a745;
  color: #fff;
}

.badge.badge-danger {
  background-color: #dc3545;
  color: #fff;
}

.dark-overlay {
  height: 100%;
  width: 100%;
  position: absolute;
  top: 0;
  left: 0;
  background-color: rgba(0, 0, 0, 0.5);
}

.table th,
.table td {
```

```
  padding: 1rem;
  text-align: left;
}

.table th {
  background: #f4f4f4;
}

.form-group {
  margin: 1.2rem 0;
}

.form-text {
  display: block;
  margin-top: 0.3rem;
  color: #888;
}

.form input[type='text'],
.form input[type='email'],
.form input[type='password'],
.form input[type='date'],
.form select,
.form textarea {
  display: block;
  width: 100%;
  padding: 0.4rem;
  font-size: 1.2rem;
  border: 1px solid #ccc;
}
```

```
.form input[type='submit'] {
  font: inherit;
}

.form .social-input {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
}

.form .social-input i {
  padding: 0.5rem;
  width: 4rem;
}

.form .social-input i.fa-twitter {
  color: #55acee;
}

.form .social-input i.fa-facebook {
  color: #3b5999;
}

.form .social-input i.fa-youtube {
  color: #cd201f;
}

.form .social-input i.fa-reddit {
  color: #ff5700;
```

```
}

.form .social-input i.fa-linkedin {
  color: #0077b5;
}

.form .social-input i.fa-instagram {
  color: #e4405f;
}

* {
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Montserrat', 'Raleway', sans-serif;
  font-size: 1rem;
  line-height: 1.6;
  background-color: #fff;
  color: #333;
}

a {
  text-decoration: none;
  color: #B51C2F;
}
```

```
ul {
  list-style-type: none;
}

img {
  width: 100%;
}

.navbar {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  -webkit-box-pack: justify;
  -ms-flex-pack: justify;
  justify-content: space-between;
  padding: 0.7rem 2rem;
  position: fixed;
  z-index: 1;
  width: 100%;
  top: 0;
  border-bottom: solid 1px #B51C2F;
  opacity: 0.9;
}

.navbar ul {
  display: -webkit-box;
  display: -ms-flexbox;
```

```
  display: flex;
}

.navbar a {
  color: #B51C2F;
  padding: 0.45rem;
  margin: 0 0.25rem;
}

.navbar a:hover {
  color: #fff;
}

.landing {
  position: relative;
  background: url('./img/background.jpg') no-repeat center center/cover;
  height: 100vh;
}

.landing-inner {
  color: #fff;
  height: 100%;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  -ms-flex-direction: column;
  flex-direction: column;
  width: 80%;
```

```
  margin: auto;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  -webkit-box-pack: center;
  -ms-flex-pack: center;
  justify-content: center;
  text-align: center;
}

.profile {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: 2fr 4fr 2fr;
  grid-template-columns: 2fr 4fr 2fr;
  grid-gap: 2rem;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  padding: 1rem;
  line-height: 1.8;
  margin-bottom: 1rem;
}

.profile-grid {
  display: -ms-grid;
  display: grid;
  grid-template-areas: 'top top' 'about about' 'exp edu' 'github github';
  grid-gap: 1rem;
}
```

```
.profile-grid .profile-top {
  -ms-grid-row: 1;
  -ms-grid-column: 1;
  -ms-grid-column-span: 2;
  grid-area: top;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  -ms-flex-direction: column;
  flex-direction: column;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  -webkit-box-pack: center;
  -ms-flex-pack: center;
  justify-content: center;
  text-align: center;
}

.profile-grid .profile-top img {
  width: 250px;
}

.profile-grid .profile-top .icons a {
  color: #fff;
  margin: 0 0.3rem;
}
```

```
.profile-grid .profile-top .icons a:hover {
  color: #03191E;
}

.profile-grid .profile-about {
  -ms-grid-row: 2;
  -ms-grid-column: 1;
  -ms-grid-column-span: 2;
  grid-area: about;
  text-align: center;
}

.profile-grid .profile-about .skills {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-pack: center;
  -ms-flex-pack: center;
  justify-content: center;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
  text-align: center;
}

.profile-grid .profile-exp {
  -ms-grid-row: 3;
  -ms-grid-column: 1;
  grid-area: exp;
```

```
}

.profile-grid .profile-edu {
  -ms-grid-row: 3;
  -ms-grid-column: 2;
  grid-area: edu;
}

.profile-grid .profile-exp > div,
.profile-grid .profile-edu > div {
  margin-bottom: 1rem;
  padding-bottom: 1rem;
  border-bottom: #ccc 1px dotted;
}

.profile-grid .profile-exp > div:last-child,
.profile-grid .profile-edu > div:last-child {
  border: none;
}

.profile-grid .profile-exp p,
.profile-grid .profile-edu p {
  margin: 0.5rem 0;
}

.profile-grid .profile-github {
  -ms-grid-row: 4;
  -ms-grid-column: 1;
  -ms-grid-column-span: 2;
  grid-area: github;
```

```
}

.profile-grid .profile-github .repo {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
}

.profile-grid .profile-github .repo > div:first-child {
  -webkit-box-flex: 7;
  -ms-flex: 7;
  flex: 7;
  -ms-flex-preferred-size: 70%;
  flex-basis: 70%;
}

.profile-grid .profile-github .repo > div:last-child {
  -webkit-box-flex: 3;
  -ms-flex: 3;
  flex: 3;
  -ms-flex-preferred-size: 20%;
  flex-basis: 20%;
}

.post-form-header {
  padding: 0.5rem;
}

.post {
  display: -ms-grid;
```

```
  display: grid;
  -ms-grid-columns: 1fr 4fr;
  grid-template-columns: 1fr 4fr;
  grid-gap: 2rem;
  -webkit-box-align: center;
  -ms-flex-align: center;
  align-items: center;
}

.post > div:first-child {
  text-align: center;
}

.post img {
  width: 150px;
}

.post .comment-count {
  background: var(--light-color);
  color: var(--primary-color);
  padding: 0.1rem 0.2rem;
  border-radius: 5px;
  font-size: 0.8rem;
}

.post .post-date {
  color: #aaa;
  font-size: 0.8rem;
  margin-bottom: 0.5rem;
}
```

```
.headerIcon {
  margin-right: 4px;
}

@media (max-width: 700px) {
  .hide-sm {
    display: none;
  }
  .container {
    margin-top: 8rem;
  }
  .x-large {
    font-size: 3rem;
  }
  .large {
    font-size: 2rem;
  }
  .lead {
    font-size: 1rem;
  }
  .navbar {
    display: block;
    text-align: center;
  }
  .navbar ul {
    text-align: center;
    -webkit-box-pack: center;
    -ms-flex-pack: center;
    justify-content: center;
```

```
}
.navbar h1 {
  margin-bottom: 1rem;
}
.dash-buttons a {
  display: block;
  width: 100%;
  margin-bottom: 0.2rem;
}
.profile {
  -ms-grid-columns: 1fr;
  grid-template-columns: 1fr;
  text-align: center;
}
.profile ul {
  display: none;
}
.profile-grid {
  grid-template-areas: 'top' 'about' 'exp' 'edu' 'github';
}
.profile-about .skills {
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
  -ms-flex-direction: column;
  flex-direction: column;
}
.post {
  -ms-grid-columns: 1fr;
  grid-template-columns: 1fr;
}
```

```
  .post a,
  .post button {
    padding: 0.3rem 0.4rem;
  }
}

.alert-wrapper {
  position: fixed;
  top: 4rem;
  right: 2rem;
  display: inline-block;
}
```