Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмна розробка прототипу мережі для соціальних комунікацій»

Студента 4 курсу, 10 групи, Бобко Влада Анатольовича

спеціальності
122 «Комп'ютерні науки» _____
підпис студента

Науковий керівник
Кандидат технічних наук, доцент _____ Козлов Валерій Володимирович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____ Пурський О. І.
« 12 » грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

Бобко Владу Анатольєвичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи

«Програмна розробка прототипу мережі для соціальних комунікацій»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *29 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка прототипу мережі для соціальної комунікації з урахуванням сучасних тенденцій побудови соціальних мереж.
Об'єкт дослідження: процес прототипування соціальних мереж.
Предмет дослідження: засоби створення мереж для соціальної комунікації.
4. Перелік графічного матеріалу_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Козлов В. В.	15.12.2022 р.	15.12.2022 р.
2	Козлов В. В.	15.12.2022 р.	15.12.2022 р.
3	Козлов В. В.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП
РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОІ ОБЛАСТІ
1.1. Дослідження предметної області
1.2. Аналіз існуючих рішень
Висновки до розділу
РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ
2.1. Вимоги до програмного продукту
2.2. Алгоритм роботи програмного продукту
Висновки до розділу
РОЗДІЛ 3. Розробка програмного продукту
3.1. Обґрунтування вибору інструментальних засобів розроблення
3.2. Опис компонентів програмного продукту
3.3. Опис користувацького інтерфейсу
3.4. Опис роботи програмного продукту
Висновки до розділу
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТКИ

7. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.10.2022*	*01.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*Розділ 1. Характеристика та аналіз предметної області*	*26.02.2023*	*26.02.2023*
5	*Розділ 2. Моделювання та проектування програмного продукту*	*06.04.2023*	*06.04.2023*
6	*Розділ 3. Розробка програмного продукту*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	*31.05.2023 -01.06.2023*
10	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	02.06.2023
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	05.06.2023
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

9. Керівник випускної кваліфікаційної роботи

Козлов В. В.

(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П. Г.

(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник

Бобко В. А.

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

Керівник випускної кваліфікаційної роботи
_________________30.05.2023 р.
(підпис, дата)

РЕФЕРАТ

Робота містить 44 сторінки, 29 рисунків, перелік джерел посилань з 7 найменувань, 1 додаток.

РОЗРОБКА СОЦІАЛЬНОЇ МЕРЕЖІ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ SPRING BOOT

Перелік ключових слів: соціальна мережа, база даних, вимоги.

Об’єктом є методології та технології проектування та розроблення веб-системи на всіх етапах життєвого циклу.

Мета роботи полягає в проектуванні та розробленні соціальної мережі з елементами маркетплейсу.

Інструменти розроблення роботи. IntelliJ IDEA , HTML5, CSS, Java, Bootstrap, MySQL, case-засіб lucidChart, Microsoft Word.

Апаратура використана при розробленні роботи – ноутбук – Apple MacBook Pro 13 inch 2019.

Результати досягнуті в процесі роботи – було розроблено соціальну мережу та проведено її тестування, яке показало, що дана система є працездатною та повністю задовольняє поставлені вимоги.

Одержані результати можуть бути використані соціальними мережами або звичайними групами людей, які прагнуть розширити свою цільову аудиторію, при цьому не витрачаючи великих коштів на розробку багатосторінкового сайту.

Рік виконання роботи – 2023.

Рік захисту роботи – 2023.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	База даних
ВИП	Вихідні повідомлення
ДМ	Даталогічна модель
ЕОМ	Електронно-обчислювальна машина
ІМ	Інфологічна модель
ІС	Інформаційна система
ОС	Операційна система
СКБД	Система керування базами даних
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
MVC	Model View Controller
SQL	Structured query language
SWOT	Strengths Weaknesses Opportunities Threats
UML	Unified Modeling Language

ЗМІСТ

ВСТУП

Соціальна мережа (англ. Social Network) — веб сайт або інша служба у Веб, яка дозволяє користувачам створювати публічну або напів публічну анкету, складати список користувачів, з якими вони мають зв'язок та переглядати власний список зв'язків і списки інших користувачів. Природа та номенклатура зв'язків може різнитись у залежності від системи.

Актуальність. Люди у всі часи об’єднувались в групи за якимось критерієм, Людина є соціальним організмом та складовою частиною соціуму, з цього витікає, що актуальність соціальних мереж висока і такою буде завжди. Соціальна мережа [1] - веб-сайт або інша служба у Веб, яка дозволяє користувачам створювати публічну або напів публічну анкету, складати список користувачів, з якими вони мають зв'язок та переглядати власний список зв’язків і списки інших користувачів

Метою роботи є реалізація комп’ютерної системи «Соціальна мережа» з використанням фреймворку Spring Boot для отримання більш технологічно свіжого та швидшого продукту на відміну від існуючих рішень.

Пошуки шляхів досягнення цієї мети обумовили необхідність визначення наступних **завдань:**

- визначити вимоги до комп’ютерної системи (функціональні та не функціональні);
- дослідити предметну область для якої створюється комп’ютерна система;
- скласти структурний алгоритм роботи комп’ютерної системи;
- обґрунтувати вибір інструментальних засобів розробки комп’ютерної системи;
- спроектувати архітектуру комп’ютерної системи;
- спроектувати базу даних для комп’ютерної системи;
- спроектувати користувацький інтерфейс та забезпечувальну

частину комп'ютерної системи;

- скласти тест-кейси для перевірки вимог до комп'ютерної системи;
- представити прототип комп'ютерної системи.

Практичне значення одержаних результатів полягає в тому, що розроблений веб-додаток може бути використаним для того щоб продати свою ідею, купити чиюсь ідею та знаходити і зв'язуватись з людьми, які є авторами ідей, що йому сподобались.

Використані інструментальні засоби:

- середовище розробки IntelliJ IDEA 2023.1.3;
- система проектування баз даних MySQL;
- CASE-засіб Lucid chart;
- операційна система MacOS.

Та текстовий редактор Microsoft Word для підготовки та оформлення пояснювальної записки до Дипломного проекту.

Структура роботи зумовлена метою і завданнями та складається зі вступу, трёх розділів, висновків, додатків та переліку джерел посилання.

РОЗДІЛ 1. ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

Соціальна мережа дуже важлива складова теперішнього часу. Людині притаманно усвідомлювати, що те що є цінним, важливим для неї, є також цінним для когось. Людина хоче знайти таких людей і об'єднатися із ними для різних цілей. Якщо це підприємець, він шукає підприємців, для того щоб, дізнатись як робити треба, а як не треба, тобто для перейняття досвіду, можливо вони об'єднаються в групу підприємців для того щоб відкрити якісь новий спільний бізнес. Весь час люди об'єднувались в групи. Соціальність це невід'ємна частина нашого життя.

1.2 Аналіз існуючих рішень

Серед існуючих програмних рішень ідентичних немає, але беручі в увагу поняття «соціальна мережа» можна виділити наступні.

Instagram – соціальна мережа, основний посил якої спочатку був направлений на обмін фотографіями та відеозаписами [3]. Instagram має мінімалістичний та досить привабливий дизайн(див. рис. 1.1). Створена у 2010 році Кевіном Сістром и Майком Кригером. У квітні 2012 року Instagram був придбаний компанією Facebook за 1 мільярд доларів. Компанія отримала шалений успіх, оскільки на той момент такого формату обміну інформацією ще не було. Основні функції Instagram :

- можливість створити свій профіль;
- можливість додати фотографію в профіль;
- можливість додати відеозапис довжиною до однієї хвилини у профіль;
- можливість підписатися та відстежувати профілі інших людей;
- можливість обміну повідомленнями між користувачами;

- можливість ділитися відеозаписами довжиною до п’ятнадцяти секунд, які автоматично зникають через сутки;
- можливість використовувати Instagram як через телефон, так і через комп’ютер та планшет;
- можливість кастомізувати інтерфейс під себе.

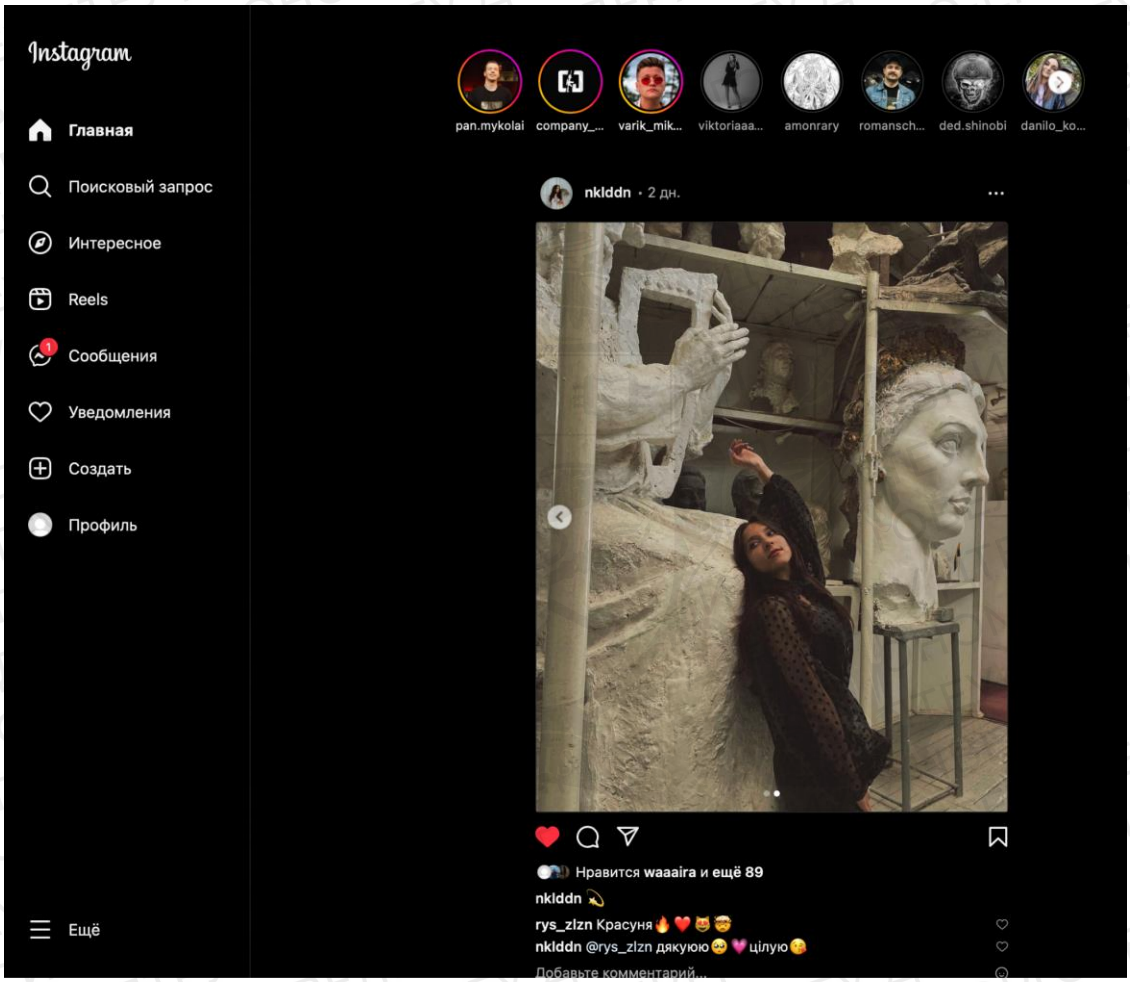

Рисунок 1.1 - Головна сторінка Instagram

Facebook – найбільша соціальна мережа в світі , що працює з 4 лютого 2004 року [2]. Була створена студентами з американського університету «Гарвард». Засновником та головою сервісу прийнято вважати Марка Цукерберга. Спочатку соціальна мережа була використовувана тільки для студентів Гарварду, але згодом компанією було вирішено розвивати продукт і вони почали розширення території використання сервісу. Цей сайт був першим, хто використав модель онлайн соціальної сторінки людини і завдяки

цьому отримав шалений успіх. Facebook також обладає досить мінімалістичним дизайном(див. рис. 1.2).Станом на 2023 рік Facebook має більш ніж 2.8 мільярда активних користувачів на місяць.

Основні функції Facebook :

- можливість створити особистий профіль;
- можливість створити рекламний профіль;
- можливість переглядати контент, що автоматично, на основі машинного вивчення Вам рекомендується;
- можливість додавати в друзі профілі інших людей;
- можливість обміну повідомленнями між користувачами;
- можливість знаходити людей не знаючи їх ім’я та прізвище, а знаючи лише деяку інформацію про них;
- можливість використовувати Facebook як через сайт, так і через програми на телефоні та планшеті.

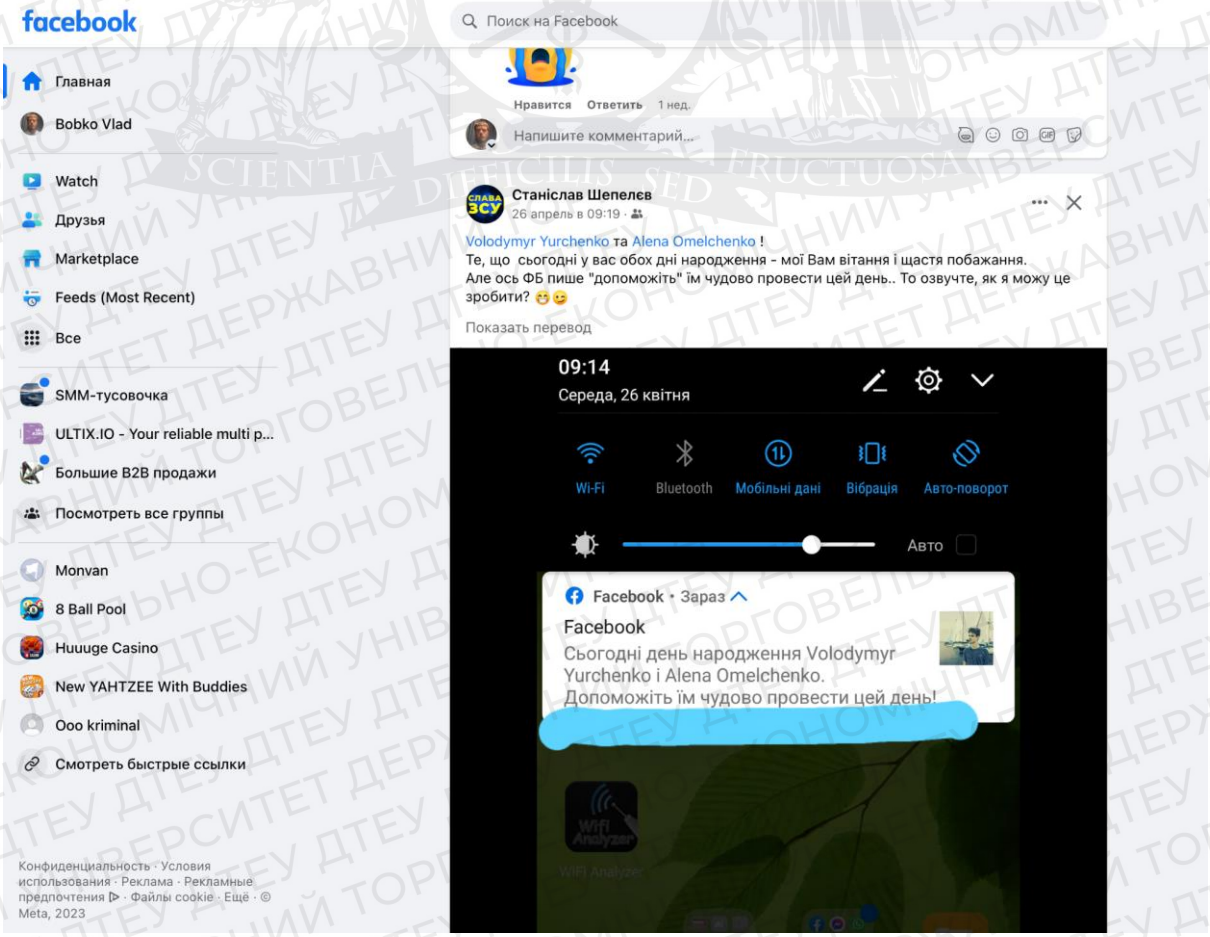

Рисунок 1.2 - Інтерфейс Facebook

Висновки до розділу 1

Детально проаналізувавши всі наявні конкуруючі проекти та існуючі рішення було проведено дослідження предметної області. Також було визначено його основні функції, переваги та недоліки.

1 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1Вимоги до програмного продукту

До програмного продукту КС «ІдеЯ» було винесено наступні вимоги. Бізнес вимоги (див. рис. 2.1)

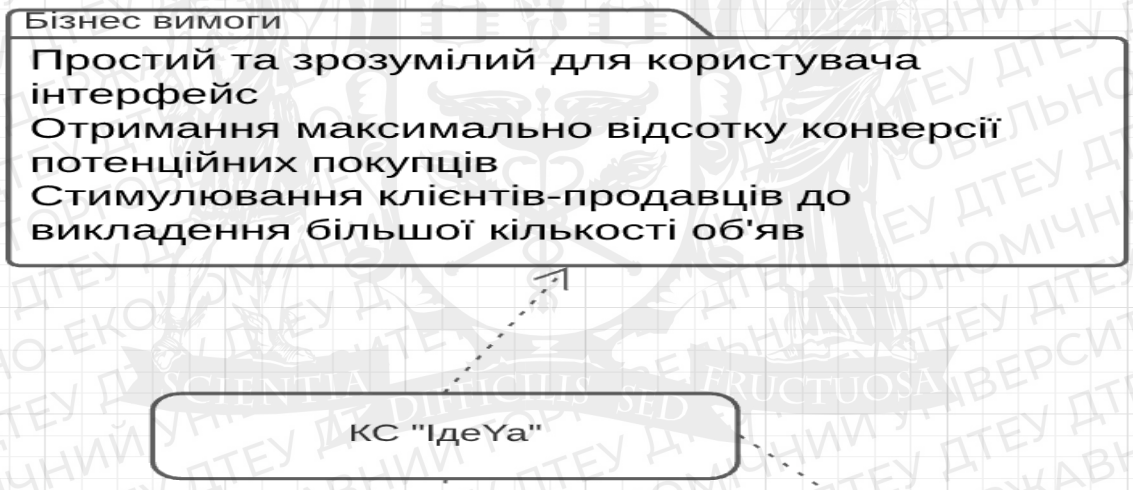

Рисунок 2.1 - Бізнес вимоги

Функціональні вимоги до системи представлені діаграмою нижче (див. рис. 2.2)

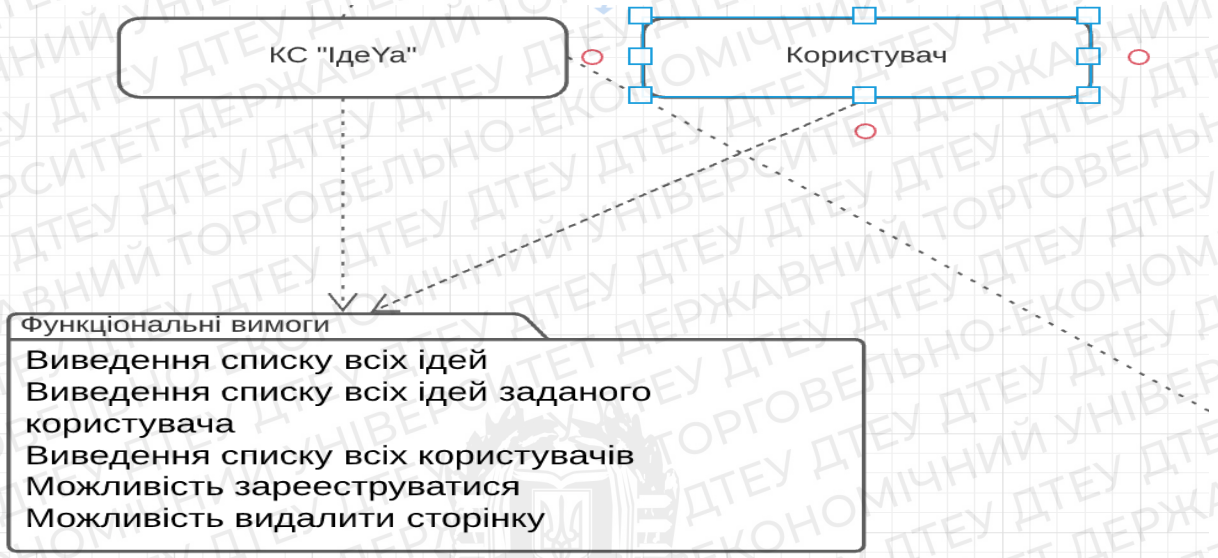

Рисунок 2.2 - Функціональні вимоги

Загальні нефункціональні вимоги були сформовані наступним чином(див. рис. 2.3)

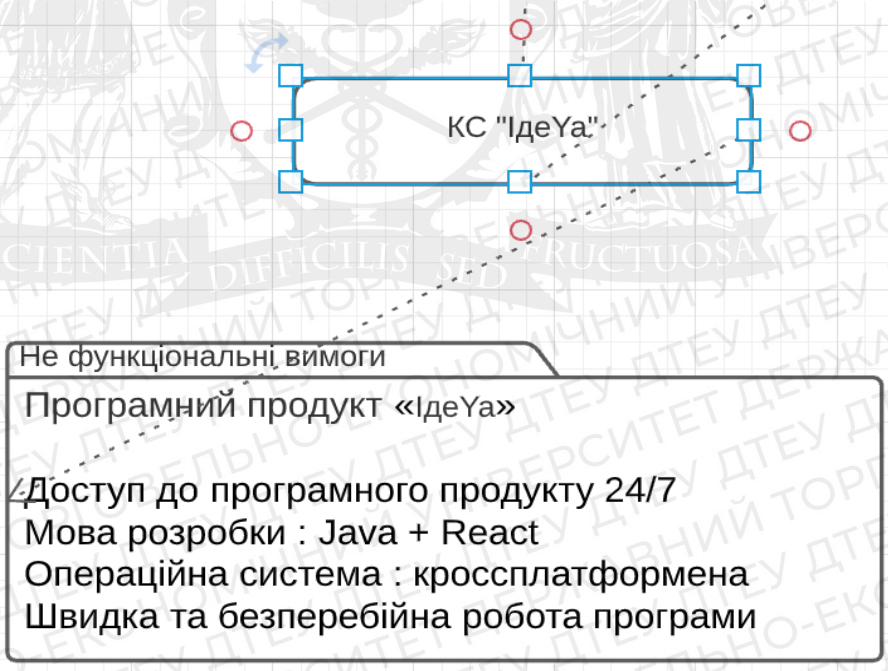

Рисунок 2.3 - Не функціональні вимоги

Вимоги користувача «Соціальна мережа «ІдеЯ» подано у на рисунку нижче(див. рис. 2.4)

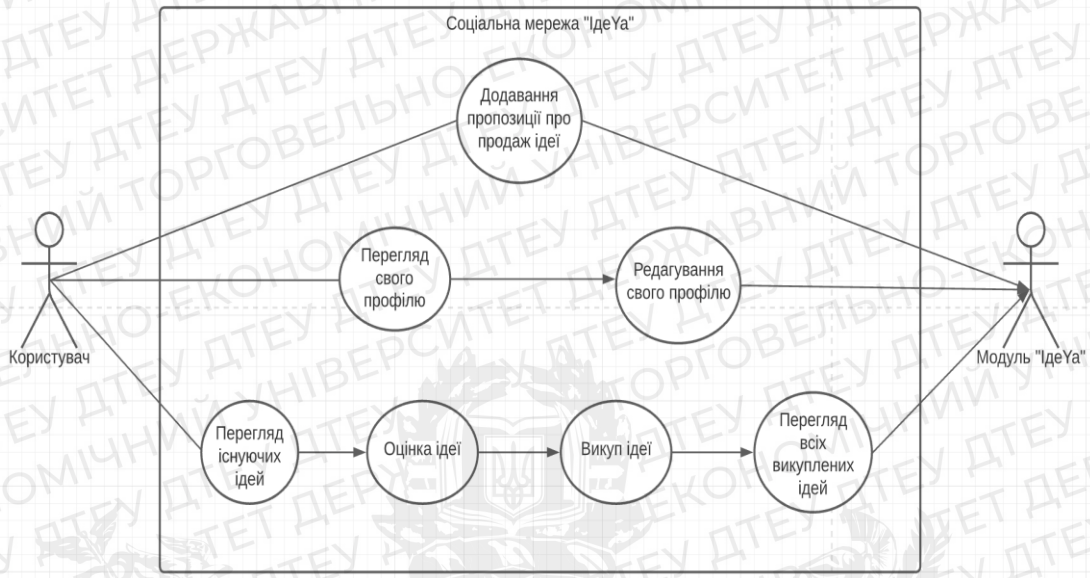

Рисунок 2.4 - Вимоги користувача

2.2 Алгоритм роботи програмного продукту

Алгоритм роботи комп'ютерної системи «Соціальна мережа «ІдеЯ» подано у вигляді UML-діаграми діяльності (див. рис. 2.5).

Рисунок 2.5 - Алгоритм комп'ютерної системи «Соціальна мережа «ІдеЯ»

Висновки до розділу 2

Проаналізувавши детально всі можливі вимоги та існуючі рішення було розроблено веб-додаток, продумано та задокументовано її функціональні можливості та процеси на логічному рівні як зі сторони

користувача так і з сторони програмного продукту. Також було визначено його основні компоненти та їх поведінку при взаємодії користувача з системою.

Отже, виконавши роботу розділу другого, можна зробити висновок про кінцевий продукт, а саме про його приблизний дизайн, функції, а також строк його готовності до роботи.

2 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Обґрунтування вибору інструментальних засобів розроблення

В якості мови програмування серверної частини було обрано Java 11.

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java[5]. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Для розробки комп’ютерної системи «Платіжна система» було обрано середовище розробки IntelliJ IDEA від компанії JetBrains.

IntelliJ IDEA— комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains[4]. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію.

Для розробки інтерфейсу було обрано бібліотеку React.

React - відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків[7]. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS

Для розробки бази даних було обрано систему проектування баз даних MySQL.

MySQL — вільна система керування реляційними базами даних[6].

MySQL був розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

3.2 Опис компонентів програмного продукту

Виходячи з основних обмежень та функціональних можливостей системи «ІдеЯ» можна зробити більш детальний опис алгоритму роботи системи, його компонентів з серверної та клієнтської сторони, та способу взаємодії з продуктом.

На стороні сервера можна виділити структуру репозиторіїв, за допомогою яких відбуваються запроси до бази даних та робота з сутностями(див. рис. 3.1).

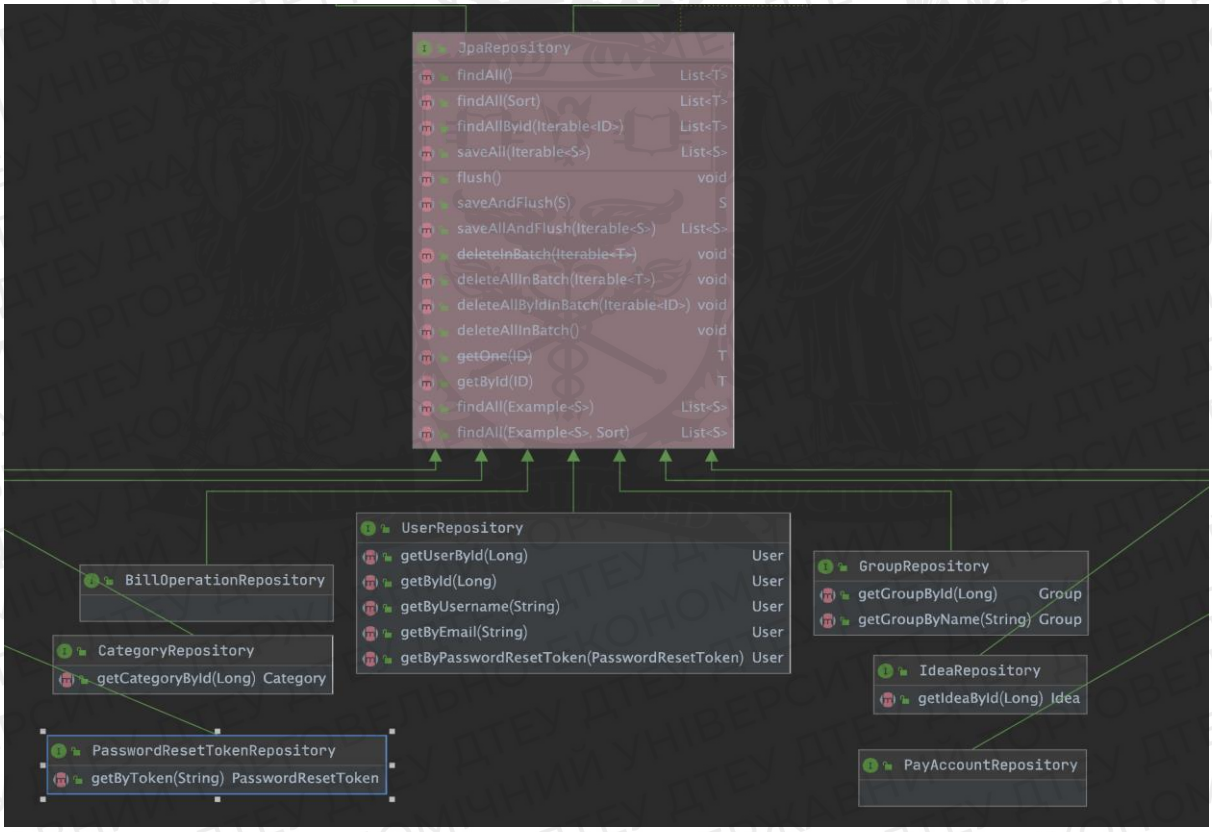

Рисунок 3.1 – Діаграма репозиторіїв системи «ІдеЯ»

Методи репозиторіїв вкладаються у сервіси, де проходить обробка даних. Можна виділити такі сервіси(див. рис.3.2):

- GroupService – для роботи з публікаціями;
- IdeaService – для роботи з ідеями;
- UserService – для роботи з користувачами;

- PayService– для роботи з фінансами;
- PasswordResetService– для роботи зі зміною пароля;
- CategoryService – для роботи з категоріями;
- MailService – для роботи з поштою.

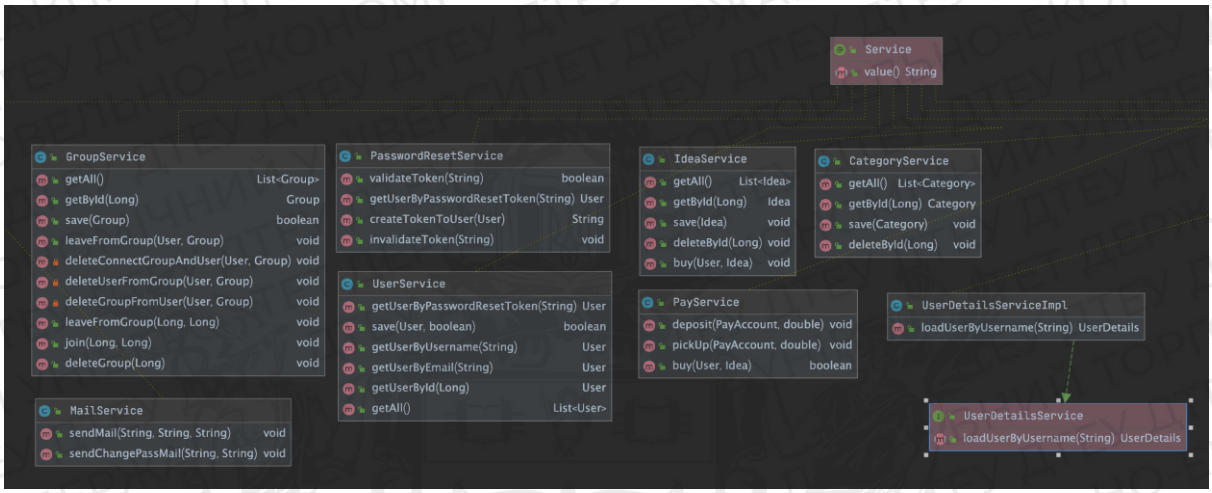

Рисунок 3.2 – Діаграма сервісів системи «Online-платформа для спілкування»

Методи сервісів використовуються контролерами для передачі отриманих даних безпосередньо в саму сторінку, що бачить користувач.

Нижче наведена структура контролерів(див. рис.3.3)

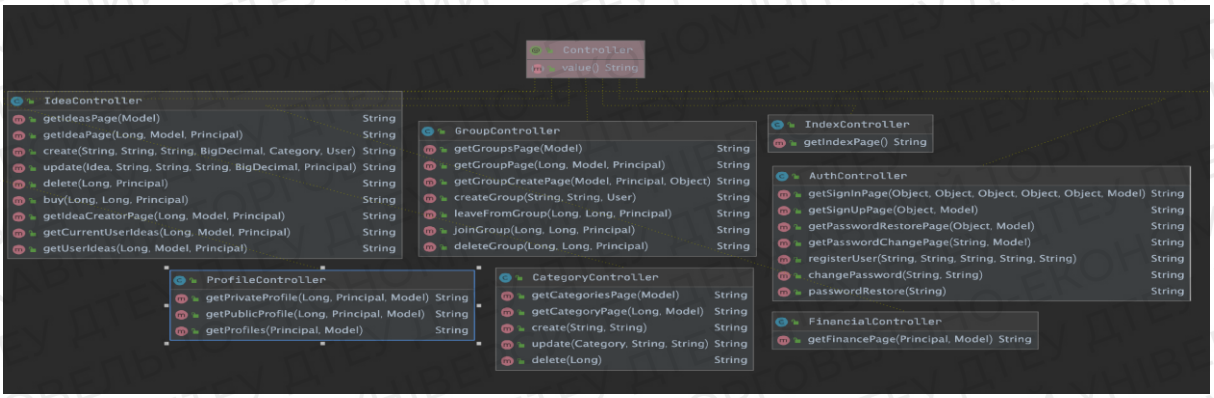

Рисунок 3.3 – Діаграма контролерів системи «Online-платформа для спілкування»

3.3 Опис користувацького інтерфейсу

Для створення графічного інтерфейсу було використано HTML, Bootstrap та Freemarker.

Веб-сторінка «Авторизація» відображає сторінку з полями які потрібно заповнити для входу (див. рис. 3.4).

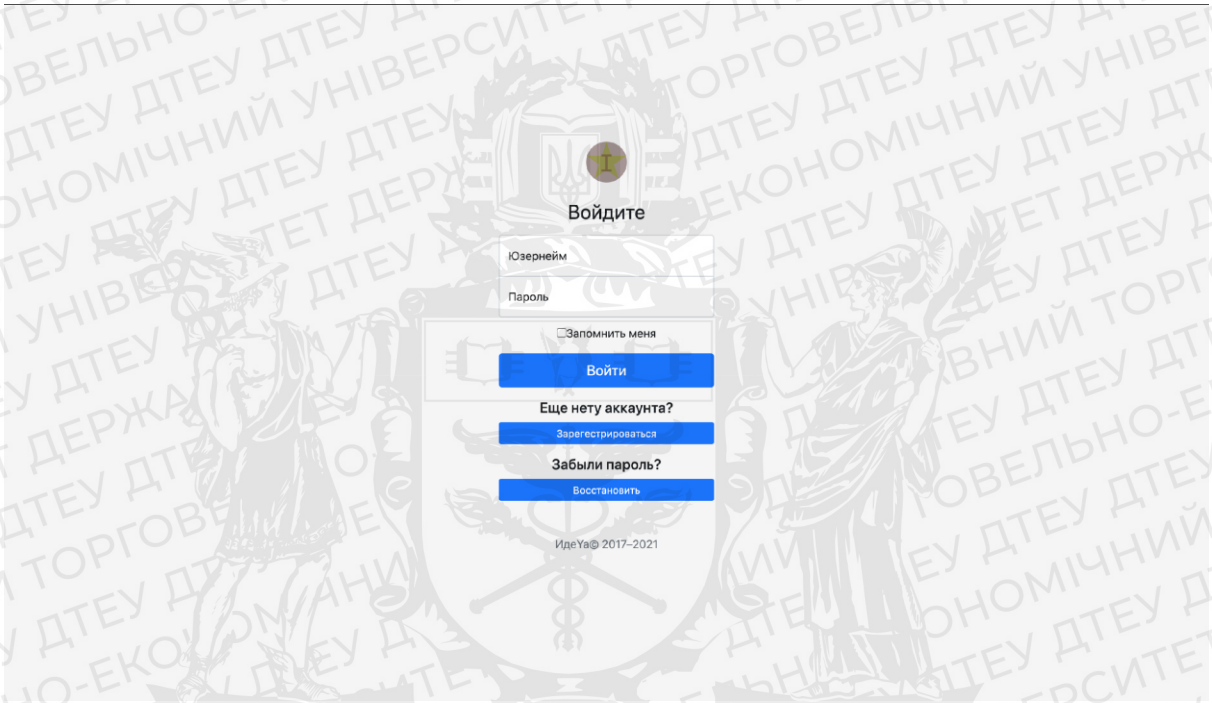

Рисунок 3.4 - Веб-сторінка «Авторизація»

Веб-сторінка «Реєстрація» відображає поля які повинен заповнити користувач для реєстрації.(див. рис. 3.5).

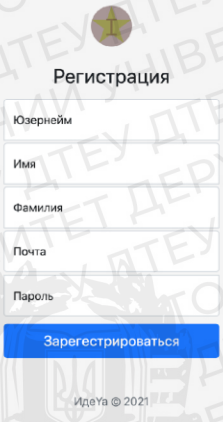

Рисунок 3.5 - Веб-сторінка «Реєстрація»

Веб-сторінка «Відновлення пароля» надає можливість користувачі відновити забутий пароль за допомогою прив’язаної до аканту пошти (див. рис. 3.6).

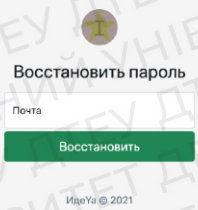

Рисунок 3.6 - Веб-сторінка «Відновлення пароля»

Веб-сторінка «Профілі» відображає всіх зареєстрованих користувачів (див. рис. 3.7).

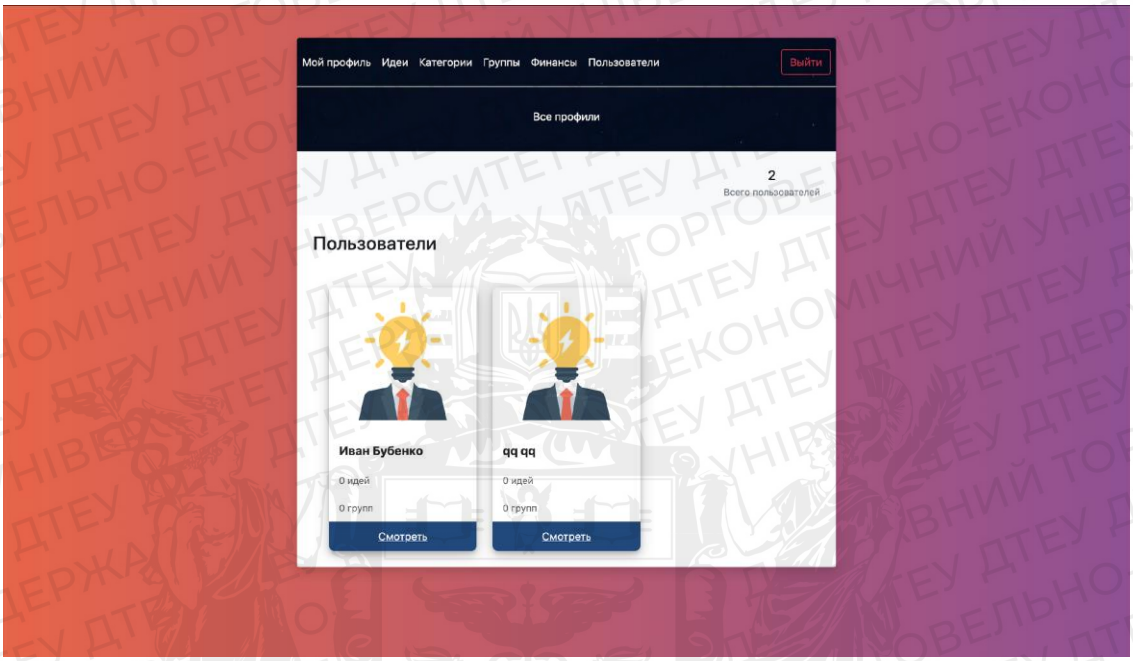

Рисунок 3.7 - Веб-сторінка «Профілі»

Веб-сторінка «Створення ідеї» відображає форму для створення нового оголошення про продаж ідеї (див. рис. 3.8)

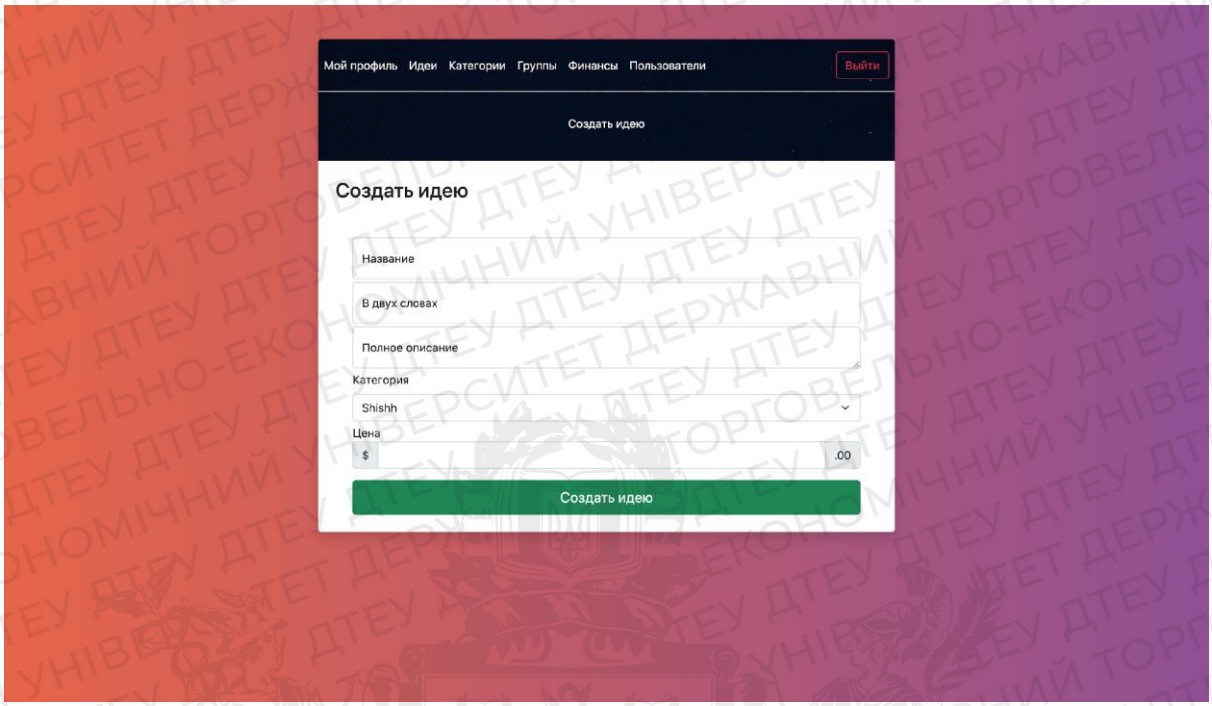

Рисунок 3.8 - Веб-сторінка «Створення ідеї»

Веб-сторінка «Ідеї» список всіх ідей з можливістю переходу на окрему сторінку окремої ідеї (див. рис. 3.9).

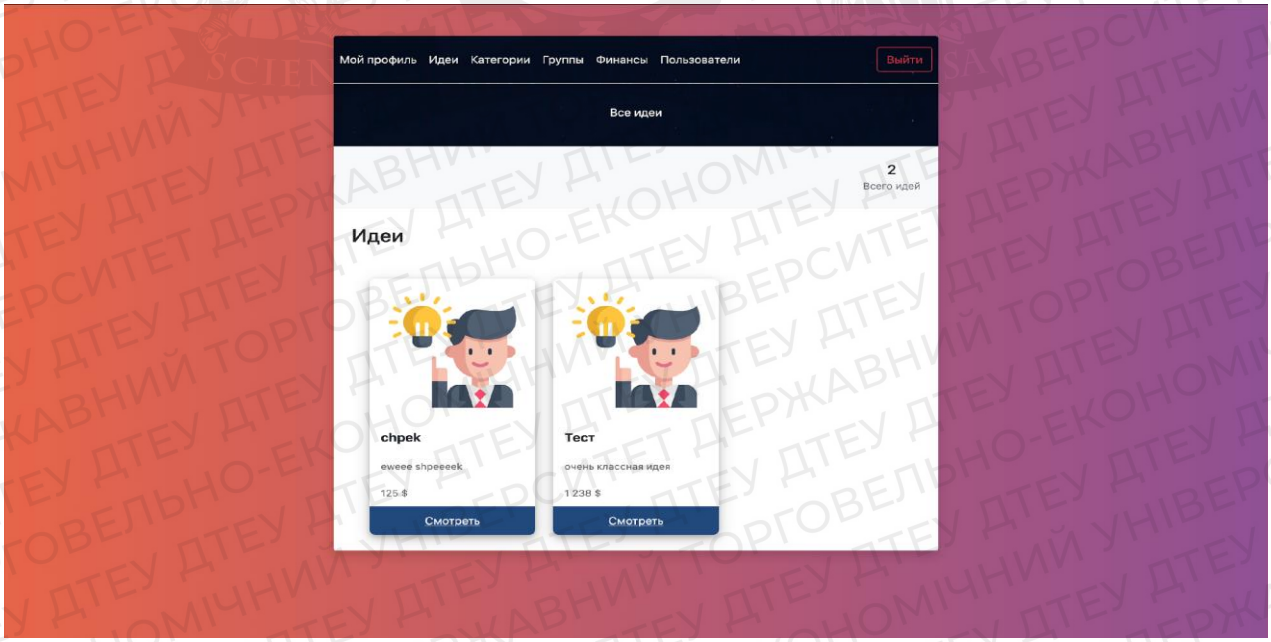

Рисунок 3.9 - Веб-сторінка «Ідеї»

Веб-сторінка «Ідея» відображає всю інформацію по певній ідеї та надає можливість переглянути профіль її власника та придбати її (див. рис. 3.10).

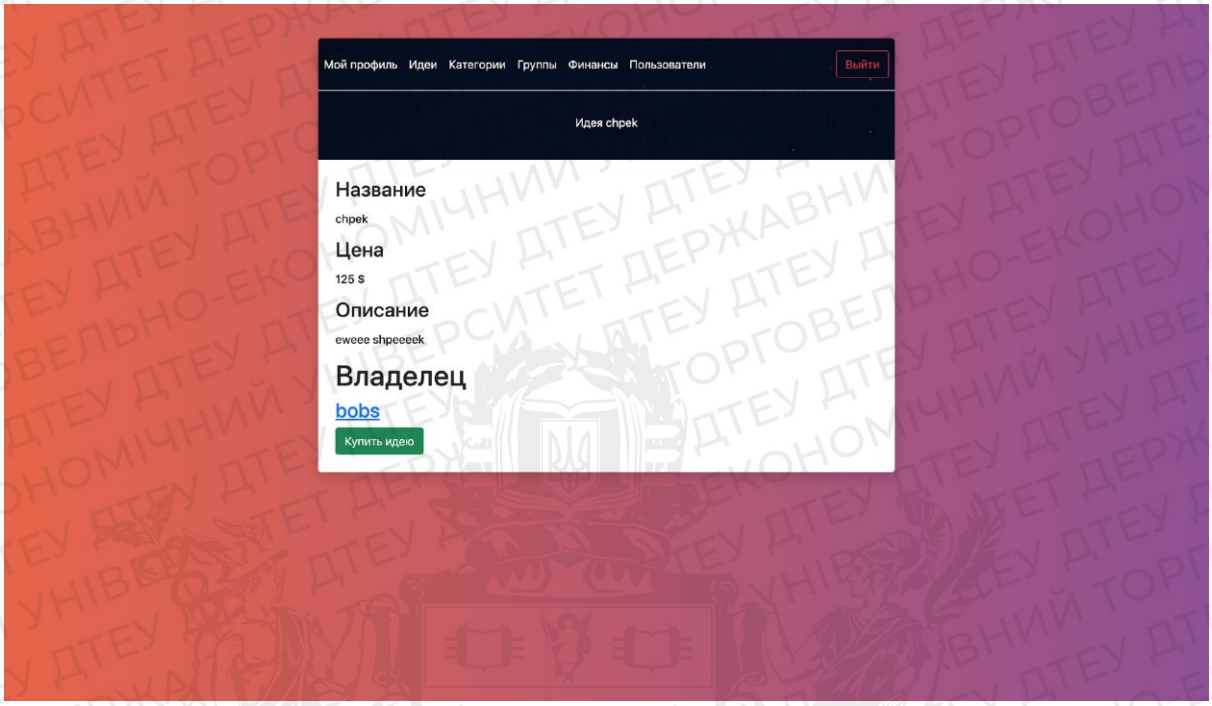

Рисунок 3.10 - Веб-сторінка «Ідея»

Веб-сторінка «Категорії» відображає список всіх категорій, до яких прив’язані всі ідеї (див. рис. 3.11).

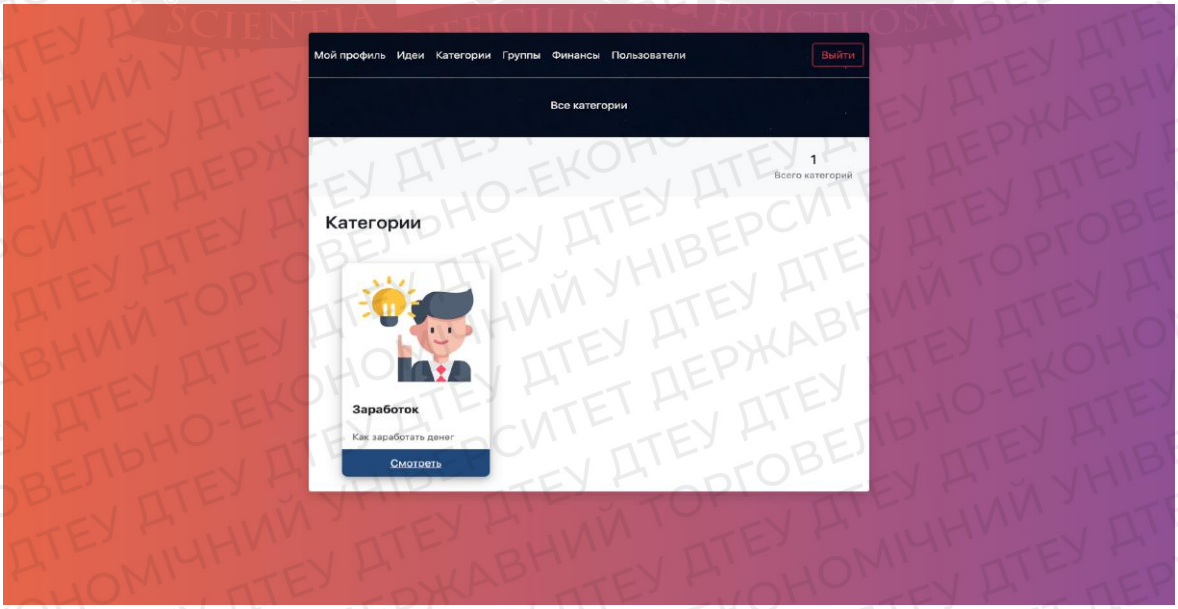

Рисунок 3.11 - Веб-сторінка «Категорії»

Веб-сторінка «Категорія» відображає список всіх ідей, що відносяться до певної категорії та їх кількість (див. рис. 3.12)

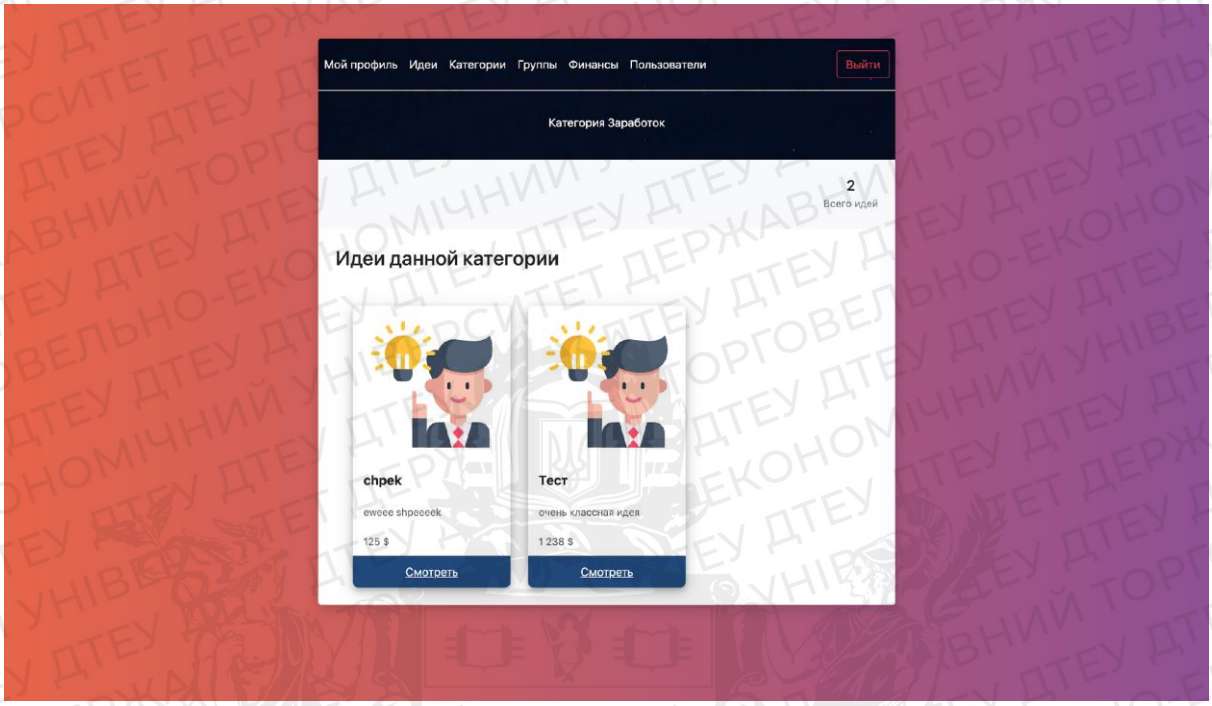

Рисунок 3.12 - Веб-сторінка «Категорія»

Веб-сторінка «Мої ідеї» відображає список всіх Ваших ідей, надає змогу кожну з них переглянути окремо та видалити (див. рис. 3.13).

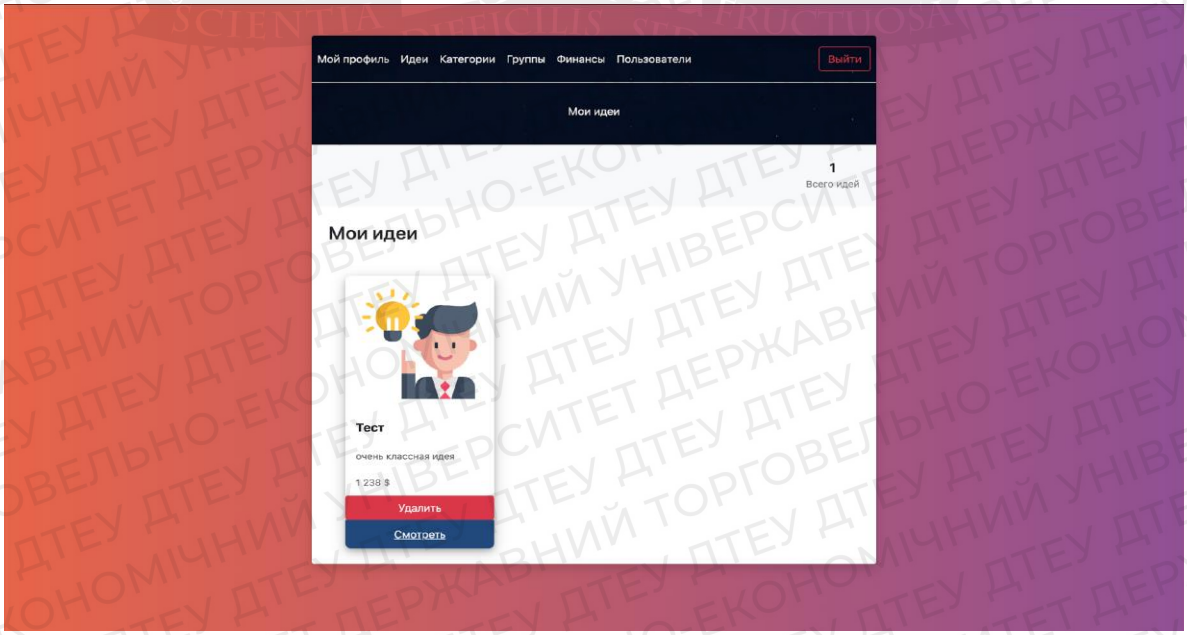

Рисунок 3.13 - Веб-сторінка «Мої ідеї»

Веб-сторінка «Створити групу» надає можливість створити нову групу (див. рис. 3.14).

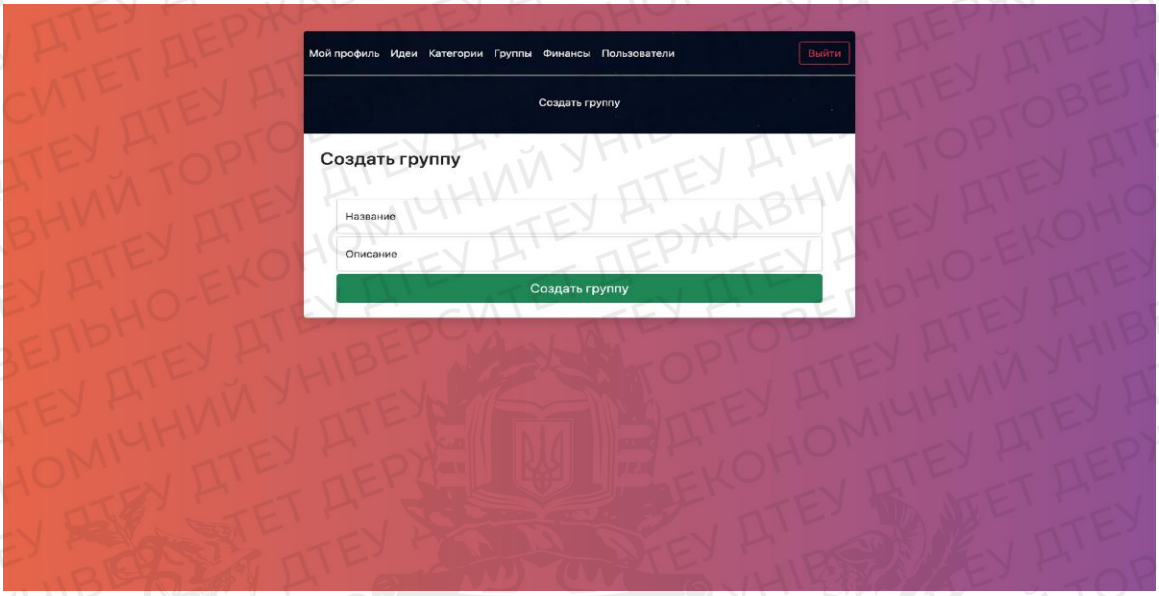

Рисунок 3.14 - Веб-сторінка «Створити групу»

Веб-сторінка «Група» надає можливість переглянути усю інформацію стосовно обраної групи, перейти на сторінку її власника та кожного з її учасників. Також у випадку якщо, Ви не є частиною цієї групи, Вам буде запропоновано приєднатися до неї, та навпаки, коли Ви є її учасником, Вам буде запропоновано покинути її. (див. рис. 3.15).

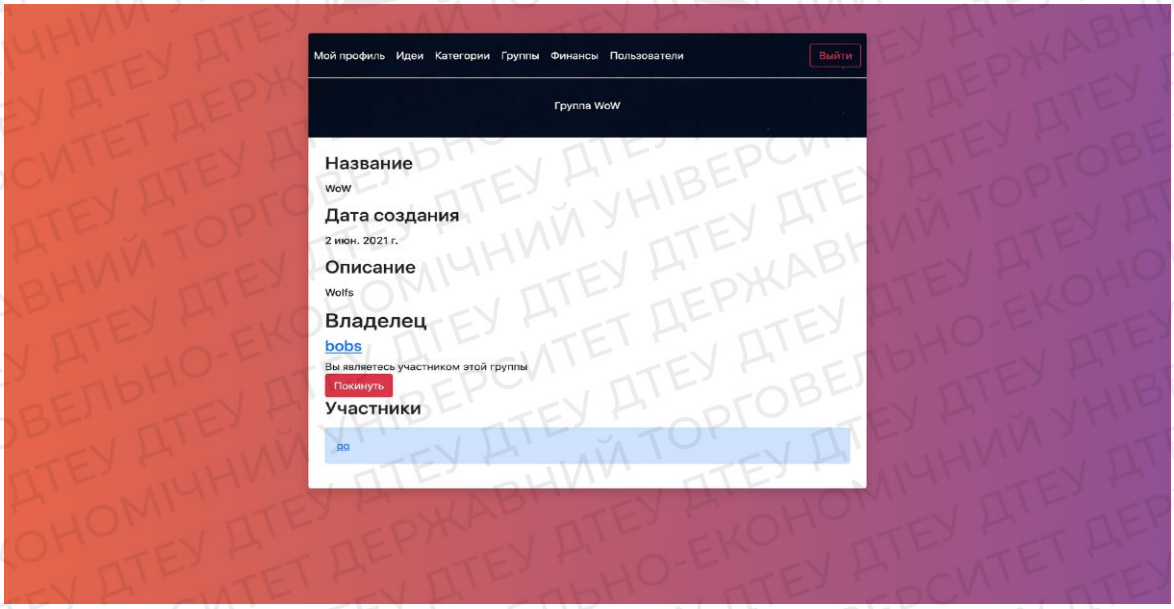

Рисунок 3.15 - Веб-сторінка «Група»

Веб-сторінка «Ідеї користувача» відображає всі ідеї конкретного користувача. (див. рис. 3.16).

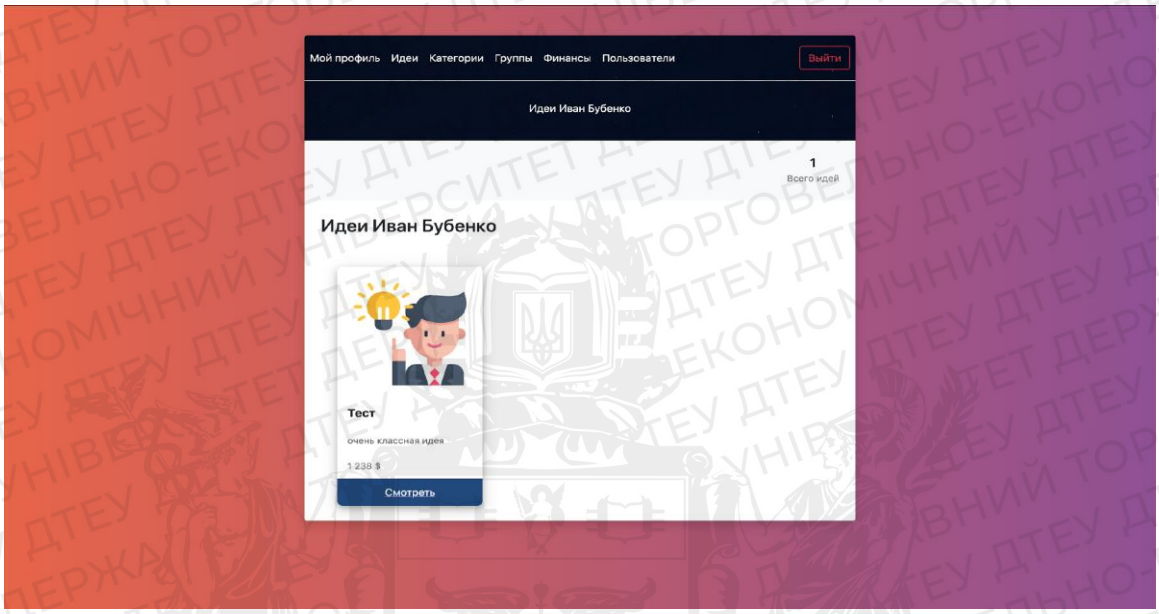

Рисунок 3.16 - Веб-сторінка «Ідеї користувача»

Веб-сторінка «Фінанси» відображає Ваш баланс та Вашу історію платежів . (див. рис. 3.17).

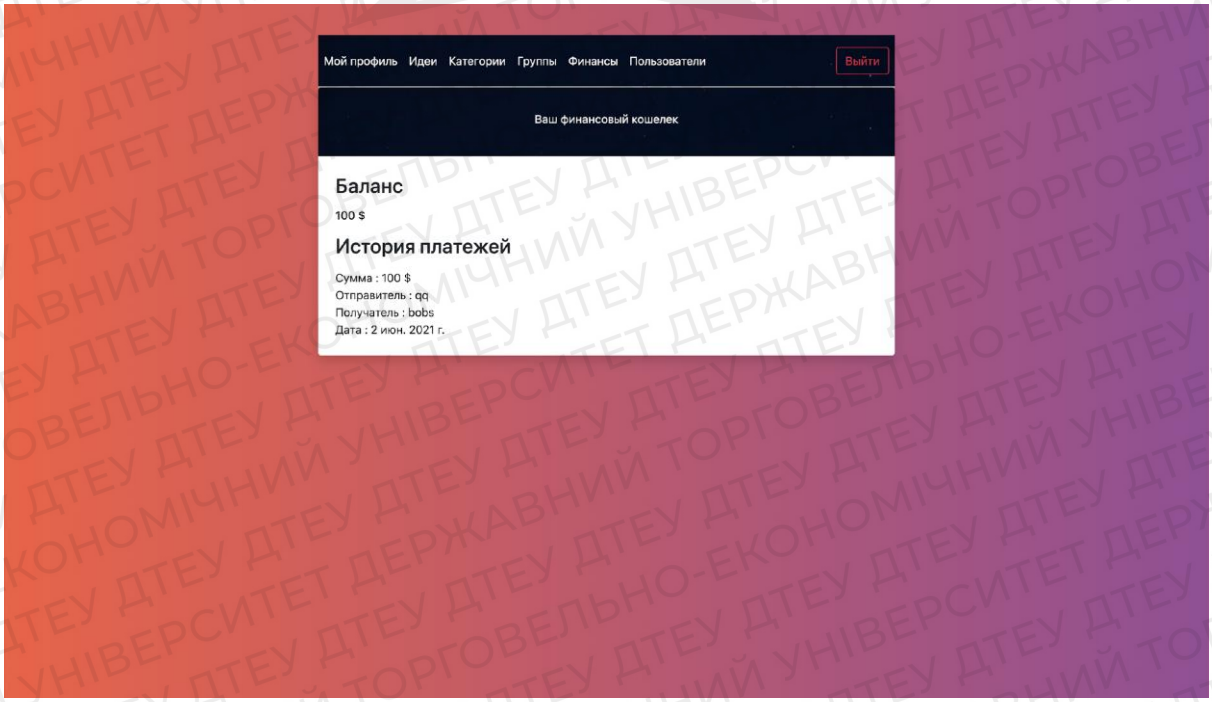

Рисунок 3.17 - Веб-сторінка «Фінанси»

Веб-сторінка «Мій профіль» відображає список ваших ідей, груп та куплених Вами ідей(див. рис. 3.16).

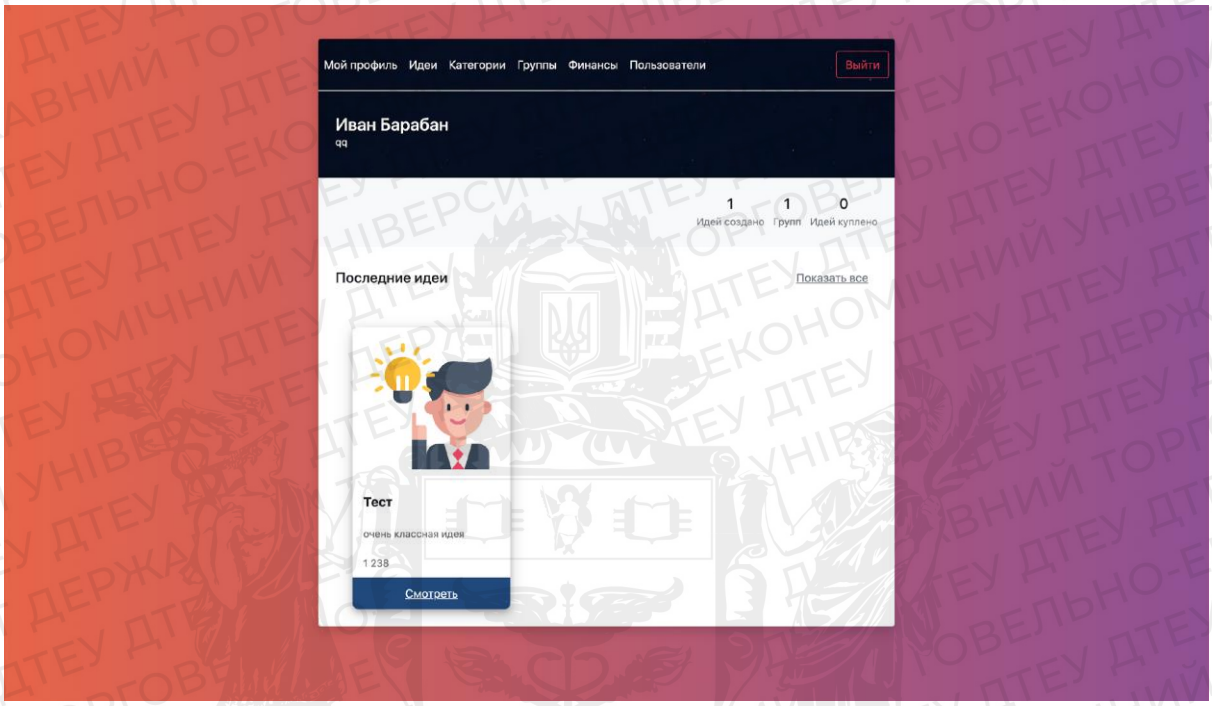

Рисунок 3.16 - Веб-сторінка «Мій профіль»

3.4 Опис роботи програмного продукту

Робота системи починається з вікна привітання користувача. Далі користувач натискає кнопку «Войти» для переходу до форми авторизації. Якщо користувач зареєстрован, він вводить логін та пароль для входу, якщо ні наттискає на клавішу «Зарегистрироваться».Якщо ж користувач забув пароль, він натискає на кнопку «Забыл пароль» та в наступному вікні вводить свою поштову скриньку, на яку йому одразу ж приходить посилання, переходячи по якому він змінює свій пароль. Після переходу для форми реєстрації треба ввести всі данні. Після авторизації ви можете вступати до всіх груп, створювати об’явлення про продаж ідеї, переглядати профілі всіх користувачів, переглядати їх ідеї, купляти їх, створювати групи та переглядати свій фінансовий кабінет.

Висновки до розділу 3

Вибір мови програмування Java є досить вдалий через її орієнтованості на клієнт-серверного програмного забезпечення та більш безпечною для розробки багатопотокових, великих, безпечних проектів.

Основними компонентами моєї системи є Користувач, Ідея, Категорія, Група, Фінансовий профіль. . Всі ці компоненти дають змогу розширюватись в проекті накопичуючи додаткові компоненти на основі даних заданих стандартних. Всі модулі розробленої системи абсолютно точно можуть працювати автономно на різних серверах для контролю навантаження на шар серверів.

ВИСНОВКИ

Розробка даного проекту провела по кожному кроку створення програмного продукту і дала можливість реалізувати знання здобуті за навчання. Проект «Соціальна мережа» має непогані перспективи на ринку за досить лояльної концепції спілкуванням з клієнтами та можливої створення фан-бази користувачів які будуть використовувати її. Соціальна сфера з кожним роком розвивається великими темпами тому проект в подальшому не втратить актуальності.

Інтернет як ресурс на який буде викладено соціальну мережу чудово підходить для нового продукту, адже є безліч можливостей . Розробка проектів в інтернеті є набагато швидшою через доступні публічні інструменти, які є абсолютно відкритими та безкоштовними.

Самі інструменти та технології розробки проекту показали себе з різних сторін, які потрібно буде враховувати далі, вибір стеку вважаємо вдалим через досить стабільність на ринку на сьогоднішній день. Весь інструментарій є безкоштовний . Стек технологій перевірений часом, тому при розробці швидко знаходяться відповіді на незрозумілі реалізації концепцій.

Розрахувавши необхідний початковий бюджет проекту і план монетизації та розвитку проекту можливо звернутись до інвесторів для вливання коштів. Зацікавленість проекту можлива завдяки економічної ефективності та капіталовкладень. Собівартість проекту для інвесторів досить приваблива.

Також для повної легалізації та законності існування проекту було розписано основні права по охороні праці, які запобігають травми на виробництві.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Соціальна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0_(%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)

2. Facebook [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Facebook.

3. Instagram [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Instagram

4. IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/IntelliJ_IDEA

5. Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java

6. MySQL [Електронний ресурс] – Режим доступу до ресурсу: http://www.mysql.ru/docs/man/

7. React [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/React

ДОДАТКИ

UserService.java

```
package fop.bobko.diploma.services;

import fop.bobko.diploma.entities.PayAccount;
import fop.bobko.diploma.entities.Role;
import fop.bobko.diploma.entities.User;
import fop.bobko.diploma.repositories.PayAccountRepository;
import fop.bobko.diploma.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

@Service
public class UserService {

    private final BCryptPasswordEncoder passwordEncoder;

    private final UserRepository userRepository;

    private final PayAccountRepository payAccountRepository;
    private final PasswordResetService passwordResetService;
    @Autowired
    public UserService(BCryptPasswordEncoder passwordEncoder, UserRepository userRepository,
PayAccountRepository payAccountRepository, PasswordResetService passwordResetService) {
        this.passwordEncoder = passwordEncoder;
        this.userRepository = userRepository;
        this.payAccountRepository = payAccountRepository;
        this.passwordResetService = passwordResetService;
    }
    public User getUserByPasswordResetToken(String token){
        if(passwordResetService.validateToken(token))
            return passwordResetService.getUserByPasswordResetToken(token);
        else
            return null;
    }
    public boolean save(User user, boolean firstSave) {

        user.setPassword(passwordEncoder.encode(user.getPassword()));

        if (firstSave) {
            if (userRepository.getByUsername(user.getUsername())!=null)
                return false;
            //(FIXME (no, just like this green color) | 4 what create token from start, ar u stupid?
//            user.setPasswordResetToken(new PasswordResetToken(UUID.randomUUID().toString()));
            user.setIdeasBought(new ArrayList<>());
            user.setIdeas(new ArrayList<>());
            user.setRole(Collections.singleton(new Role(1L, "ROLE_USER")));
            user.setEnable(true);
            user.setGroups(new ArrayList<>());
            PayAccount payAccount = new PayAccount(new BigDecimal(0), new ArrayList<>());
            payAccountRepository.save(payAccount);
```

```
            user.setPayAccount(payAccount);
        }
        userRepository.save(user);
        return true;
    }

    public User getUserByUsername(String username){

        return userRepository.getByUsername(username);
    }

    public User getUserByEmail(String email) {
        return userRepository.getByEmail(email);
    }
    public User getUserById(Long id) {
        return userRepository.getUserById(id);
    }
    public List<User> getAll(){
        return userRepository.findAll();
    }
}
```

PasswordResetService.java

```
package fop.bobko.diploma.services;

import fop.bobko.diploma.entities.PasswordResetToken;
import fop.bobko.diploma.entities.User;
import fop.bobko.diploma.repositories.PasswordResetTokenRepository;
import fop.bobko.diploma.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Calendar;
import java.util.UUID;

@Service
public class PasswordResetService {

    private final PasswordResetTokenRepository tokenRepository;
    private final UserRepository userRepository;


    @Autowired
    public PasswordResetService(PasswordResetTokenRepository tokenRepository, UserRepository
userRepository) {
        this.tokenRepository = tokenRepository;
        this.userRepository = userRepository;
    }

    public boolean validateToken(String token) {
        PasswordResetToken passwordResetToken = tokenRepository.getByToken(token);
        if (passwordResetToken==null)
            return false;
        if (Calendar.getInstance().after(passwordResetToken.getExpiredTime())) {
            tokenRepository.delete(passwordResetToken);
            return false;
```

```
        }
        return true;
    }

    public User getUserByPasswordResetToken(String token) {
        PasswordResetToken passwordResetToken = tokenRepository.getByToken(token);
        return userRepository.getByPasswordResetToken(passwordResetToken);
    }
    public String createTokenToUser(User user){
        String token = UUID.randomUUID().toString();
        PasswordResetToken passwordResetToken = new PasswordResetToken(token);
        user.setPasswordResetToken(passwordResetToken);
        tokenRepository.save(passwordResetToken);
        userRepository.save(user);
        return token;
    }

    public void invalidateToken(String token) {
        PasswordResetToken passwordResetToken = tokenRepository.getByToken(token);
        User user = userRepository.getByPasswordResetToken(passwordResetToken);
        user.setPasswordResetToken(null);
        userRepository.save(user);
        tokenRepository.delete(passwordResetToken);
    }
}
```

GroupService.java

```
package fop.bobko.diploma.services;

import fop.bobko.diploma.entities.Group;
import fop.bobko.diploma.entities.User;
import fop.bobko.diploma.repositories.GroupRepository;
import fop.bobko.diploma.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

@Service
public class GroupService {

    private final GroupRepository groupRepository;
    private final UserRepository userRepository;
    @Autowired
    public GroupService(GroupRepository groupRepository, UserRepository userRepository) {
        this.groupRepository = groupRepository;
        this.userRepository = userRepository;
    }
    public List<Group> getAll() {
        return groupRepository.findAll();
    }
    public Group getById(Long id) {
        return groupRepository.getGroupById(id);
    }
```

```
public boolean save(Group group) {
    if (groupRepository.getGroupByName(group.getName())!=null)
        return false;
    groupRepository.save(group);
    return true;
}
public void leaveFromGroup(User user, Group group) {
    List<User> groupUsers = group.getUsers();
    if (groupUsers.contains(user)){
        groupUsers.remove(user);
        group.setUsers(groupUsers);
        groupRepository.save(group);
    }
    List<Group> memberGroups = user.getGroups();
    if (memberGroups.contains(group)){
        memberGroups.remove(group);
        user.setGroups(memberGroups);
        userRepository.save(user);
    }
}

private void deleteConnectGroupAndUser(User user, Group group) {
    deleteGroupFromUser(user, group);
    deleteUserFromGroup(user,group);
}

private void deleteUserFromGroup(User user, Group group) {
    List<User> groupUsers = group.getUsers();
    if (groupUsers.contains(user)){
        groupUsers.remove(user);
        group.setUsers(groupUsers);
        groupRepository.save(group);
    }

}
private void deleteGroupFromUser(User user, Group group){
    List<Group> memberGroups = user.getGroups();
    if (memberGroups.contains(group)){
        memberGroups.remove(group);
        user.setGroups(memberGroups);
        userRepository.save(user);
    }
}
public void leaveFromGroup(Long userID, Long groupID) {
    User member = userRepository.getUserById(userID);
    Group group = groupRepository.getGroupById(groupID);
    deleteUserFromGroup(member,group);
    deleteGroupFromUser(member,group);
}


public void join(Long userID, Long groupID) {
    User member = userRepository.getUserById(userID);
    Group group = groupRepository.getGroupById(groupID);
    List<User> groupUsers = group.getUsers();
    if (!groupUsers.contains(member)){
```

```
            groupUsers.add(member);
            group.setUsers(groupUsers);
            groupRepository.save(group);
        }
        List<Group> memberGroups = member.getGroups();
        if (!memberGroups.contains(group)) {
            memberGroups.add(group);
            member.setGroups(memberGroups);
            userRepository.save(member);
        }
    }

    public void deleteGroup(Long groupID) {
        Group group = groupRepository.getGroupById(groupID);
        List<User> groupUsers = group.getUsers();
        groupUsers.forEach(member->deleteGroupFromUser(member,group));
        groupRepository.delete(group);

        //every user delete this group
    }
}
```

AuthController.java

```
package fop.bobko.diploma.controllers;

import fop.bobko.diploma.entities.User;
import fop.bobko.diploma.services.MailService;
import fop.bobko.diploma.services.PasswordResetService;
import fop.bobko.diploma.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class AuthController {
    private final UserService userService;
    private final MailService mailService;
    private final PasswordResetService passwordResetService;

    @Autowired
    public AuthController(UserService userService, MailService mailService, PasswordResetService
passwordResetService) {
        this.userService = userService;
        this.mailService = mailService;
        this.passwordResetService = passwordResetService;
    }

    @GetMapping("/auth/signin")
    public String getSignInPage(@RequestParam(name = "newUser", required = false) Object
nowRegistered,
                                @RequestParam(name = "checkMail", required = false) Object checkMail,
                                @RequestParam(name = "passwordChanged", required = false) Object
passwordChanged,
                                @RequestParam(name = "tokenExpired", required = false) Object tokenExpired,
```

```
                @RequestParam(name = "badCredentials", required = false) Object badCredentials,
                Model model) {
    if (nowRegistered != null)
        model.addAttribute("newRegistered", true);
    if (checkMail != null)
        model.addAttribute("checkMail", true);
    if (passwordChanged != null)
        model.addAttribute("passwordChanged", true);
    if (tokenExpired != null)
        model.addAttribute("tokenExpired", true);
    if (badCredentials != null)
        model.addAttribute("badCredentials", true);

    return "signin";
}

@GetMapping("/auth/signup")
public String getSignUpPage(@RequestParam(value = "error",required = false)Object error,
                Model model) {
    if (error!=null)
        model.addAttribute("error",true);
    return "signup";
}

@GetMapping("/auth/restorePassword")
public String getPasswordRestorePage(@RequestParam(value = "error", required = false) Object error,
                    Model model) {
    if (error != null)
        model.addAttribute("error", true);
    return "passwordrestore";
}

@GetMapping("/auth/passwordChange")
public String getPasswordChangePage(@RequestParam("token") String token,
                    Model model) {
    User user = userService.getUserByPasswordResetToken(token);
    if (user == null)
        return "redirect:/auth/signin?tokenExpired";
    model.addAttribute("token", token);
    return "passwordchange";
}

@PostMapping("/auth/registration")
public String registerUser(@RequestParam("username") String username,
                @RequestParam("email") String email,
                @RequestParam("password") String password,
                @RequestParam("firstName") String firstName,
                @RequestParam("lastName") String lastName) {
    User user = new User(username, email, password, firstName, lastName);
    if (userService.save(user, true))
        return "redirect:/auth/signin?newUser";
    else
        return "redirect:/auth/signup?error";
}

@PostMapping("/auth/changePassword")
```

```
    public String changePassword(@RequestParam("token") String token,
                                 @RequestParam("password") String newPassword) {
        User user = userService.getUserByPasswordResetToken(token);
        if (user == null)
            return "redirect:/auth/signin?tokenExpired";
        user.setPassword(newPassword);
        userService.save(user, false);
        passwordResetService.invalidateToken(token);
        return "redirect:/auth/signin?passwordChanged";
    }

    @PostMapping("/auth/restore")
    public String passwordRestore(@RequestParam("email") String email) {
        User userByEmail = userService.getUserByEmail(email);
        if (userByEmail == null)
            return "redirect:/auth/restorePassword?error";

        mailService.sendChangePassMail(email, passwordResetService.createTokenToUser(userByEmail));
        return "redirect:/auth/signin?checkMail";
    }
}
```

GroupController.java

```
package fop.bobko.diploma.controllers;

import fop.bobko.diploma.entities.Group;
import fop.bobko.diploma.entities.User;
import fop.bobko.diploma.services.GroupService;
import fop.bobko.diploma.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.security.Principal;

@Controller
public class GroupController {
    private final GroupService groupService;
    private final UserService userService;

    @Autowired
    public GroupController(GroupService groupService, UserService userService) {
        this.groupService = groupService;
        this.userService = userService;
    }

    @GetMapping("/groups")
    public String getGroupsPage(Model model) {
        model.addAttribute("groups", groupService.getAll());
        return "groups";
    }

    @GetMapping("/group/{id}")
```

```
public String getGroupPage(@PathVariable("id") Long id,
                           Model model,
                           Principal principal) {
    if (principal == null)
        return "redirect:/auth/signin";

    if (groupService.getById(id) == null)
        return "redirect:/groups";
    User currentUser = userService.getUserByUsername(principal.getName());
    model.addAttribute("currentUser", currentUser);
    Group group = groupService.getById(id);
    model.addAttribute("group", group);
    return "group";
}

@GetMapping("/groupsCratePage")
public String getGroupCreatePage(Model model,
                                 Principal principal,
                                 @RequestParam(value = "error",required = false) Object error) {
    if (principal == null)
        return "redirect:/auth/signin";
    if (error != null)
        model.addAttribute("error", true);
    User user = userService.getUserByUsername(principal.getName());
    model.addAttribute("user", user);
    return "groupcreate";
}

@PostMapping("/groups/create")
public String createGroup(@RequestParam("description") String description,
                          @RequestParam("name") String name,
                          @RequestParam("owner") User owner) {
    Group group = new Group(description, name, owner);
    if (!groupService.save(group))
        return "redirect:/groupsCratePage" + "?error";
    return "redirect:/groups";
}

@PostMapping("/groups/leave")
public String leaveFromGroup(@RequestParam("userid") Long userID,
                             @RequestParam("groupid") Long groupID,
                             Principal principal) {
    String redirectTo = "redirect:/group/" + groupID;
    if (principal == null)
        return redirectTo;
    User userByPrincipal = userService.getUserByUsername(principal.getName());
    if (userByPrincipal == null)
        return redirectTo;
    User userByID = userService.getUserById(userID);
    if (userID == null)
        return redirectTo;
    if (userByPrincipal.getId().equals(userByID.getId()))
        groupService.leaveFromGroup(userID, groupID);
    return redirectTo;
}
```

```
    @PostMapping("/groups/join")
    public String joinGroup(@RequestParam("userid") Long userID,
                            @RequestParam("groupid") Long groupID,
                            Principal principal) {
        String redirectTo = "redirect:/group/" + groupID;
        if (principal == null)
            return redirectTo;
        User userByID = userService.getUserById(userID);
        if (userID == null)
            return redirectTo;
        User userByPrincipal = userService.getUserByUsername(principal.getName());
        if (userByPrincipal == null)
            return redirectTo;
        if (userByPrincipal.getId().equals(userByID.getId()))
            groupService.join(userID, groupID);
        return redirectTo;
    }

    @PostMapping("/groups/delete")
    public String deleteGroup(@RequestParam("userid") Long userID,
                              @RequestParam("groupid") Long groupID,
                              Principal principal) {
        String redirectTo = "redirect:/groups";
        if (principal == null)
            return redirectTo;
        User userByID = userService.getUserById(userID);
        if (userID == null)
            return redirectTo;
        User userByPrincipal = userService.getUserByUsername(principal.getName());
        if (userByPrincipal == null)
            return redirectTo;
        if (userByPrincipal.getId().equals(userByID.getId()))
            groupService.deleteGroup(groupID);
        return redirectTo;
    }
}
```