ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для управління нерухомістю агентства «BrilliantHome» мовою Python з застосуванням фреймворків Django та Angular»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Абдуль Фаттах МАХМУД
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Абдуль-Фаттах МАХМУД

Керівник: Тимур ДОВЖЕНКО
к.т.н.

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____ Махмуду Абдуль Фаттаху Мазеновичу_____

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для управління нерухомістю агентства «BrilliantHome» мовою Python з застосуванням фреймворків Django та Angular»
керівник кваліфікаційної роботи к.т.н., доцент кафедри ІПЗ Тимур ДОВЖЕНКО, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: технічна література пов’язана з питань пов’язних з розробки мовою програмування Python та фреймворками Angular та Django. Теоретичні відомості про методи продажу та оренди нерухомості. Опис методів побудови архітектури web-застосунку.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1.Огляд та аналіз існуючих аналогів web-застосунків для продажу та оренди нерухомості.
2. Аналіз та вибір технологій для розробки web-застосунку.
3. Програмна реалізація та опис функціонування web-застосунку для прод
та оренди нерухомості “BrilliantHome”.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до застосунку.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Діаграма бази даних.
7. Діаграма активності перегляду оголошень.
8. Діаграма карти сайту.
9. Екранні форми.
10. Демонстрація роботи web-застосунку.
11. Апробація результатів.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих web-застосунків для оренди та продажу нерухомості	14.03-17.03.2024	
4	Проектування web-застосунку для оренди та продажу нерухомості	18.03-24.03.2024	
5	Розробка web-застосунку	25.03-21.04.2024	
6	Тестування web-застосунку	22.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Абдуль-Фаттах МАХМУД
(підпис)

Керівник
кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 61 стор., 10 табл., 71 рис., 13 джерел.

Мета роботи – підтримка процесу продажу та оренди нерухомості, створений за допомогою фреймворків (Angular та Django).

Об'єкт дослідження – процес продажів та оренди нерухомості.

Предмет дослідження – web-застосунок для оренди та продажу нерухомості.

Короткий зміст роботи: У дипломній роботи було проведено аналіз та порівняння інших web-застосунків. Було проаналізовано та виявлено переваги та недоліки інших додатків.

Візуальну частину web-застосунку було створено за допомогою фреймворків Angular, а для стилю сторінок було використано CSS. Серверну частину було створено за допомогою фреймворку Django. Web-застосунку було створено у середовищах для розробки "PyCharm IDE" та "WebStorm IDE"; завдяки цим програмам розробка стає більш швидкою та продуктивною. Для створення макету та елементів графічного дизайну була використана програма Figma. Цей web-застосунок буде створений для того, щоб користувачі мали можливість для безкоштовного опублікування нерухомості.

Для зручності для користувачів було додано англійську та українську мову, також буде реалізовано перемикач світлої та темної теми для зручності користувачів

Сферою використання застосунку "BrilliantHome" є продаж та оренда нерухомості, забезпечуючи користувачам зручний інструмент для безкоштовного опублікування та перегляду оголошень.

КЛЮЧОВІ СЛОВА: WEBSTORM IDE, ANGULAR, PYCHARM IDE, PYTHON, DJANGO

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARB (.arb) – Application Resource Bundle

JSON (.json) – JavaScript Object Notation

XLIFF (.xlf) – XML Localization Interchange File Format

XMB (.xmb) – XML Message Bundle

SEO – Оптимізація для пошукових систем

ORM – Object-Relational Mapping

SPA – Single Page Application

MVC – Model-View-Controller

ORM – Object Relational Mapping

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

FAQ – Frequently Asked Question

ВСТУП

Об’єктом дослідження - є процес продажу та оренди будинків та квартир “BrilliantHome”

Предметом дослідження - є технології якими буде розроблено web-застосунку, для верстки сайту використовуватиметься HTML та CSS, для розробки frontend буде використано фреймворк Angular, для серверної частини використовуватиметься Python з підключенням фреймворку Django.

У наш час зростає потреба у використанні онлайн-платформ для купівлі, продажу та оренди житла. Метою моєї дипломної роботи є розробка web-застосунку під назвою “BrilliantHome” для полегшення процесу оренди, купівлі та продажу нерухомості. Ця онлайн-платформа була розроблена для надання клієнтам зручний та ефективний інструмент для різних операцій з нерухомістю.

Для створення web-застосунку використовувались сучасні мови програмування та надійні технології:

Задачі дипломної роботи:

1. Провести аналіз предметної галузі та визначити основні переваги та недоліки існуючих web-застосунків.
2. Обрати засоби розробки для створення web-застосунку, проаналізувати фреймворки та інструменти
3. Розробити web-застосунок для оренди та продажу нерухомості
4. Провести тестування розробленого web-застосунку за допомогою фреймворку Jasmine

HTML (HyperText Markup Language) - використовувався для створення структури веб-сторінок і розміщення таких елементів, як текст, зображення і форми, а також для того, щоб зробити інтерфейси додатків зручними для користувача.

CSS (Cascading Style Sheets) - використовувався для формування веб-сторінок і надання їм більш привабливого вигляду, полегшення розпізнавання інформації користувачами та покращення взаємодії з додатками.

Angular - використовувався для створення структурованого та модульного коду веб-додатку “BrilliantHome”, а його оптимізація полягає в застосуванні *Lazy Loading* для ефективного завантаження ресурсів та *AOT Compilation* для зменшення часу завантаження сторінок.

Python(Django) - використовувався для створення потужного та надійного серверного backend для BrilliantHome, забезпечуючи обробку запитів користувачів, доступ до бази даних та взаємодію з frontend для оптимальної роботи всього додатку.

Усі ці технології дають перевагу для створення web-застосунку, а також надає користувачам надійний захист та безпеку.

Переваги додатку - швидкий доступ до актуальної інформації про нерухомість, можливість взаємодіяти з іншими користувачами, просте управління операціями з продажу та оренди житла, а також оптимізована швидкість завантаження сторінок з високою продуктивністю та ефективністю завдяки використанню таких технологій, як Angular та Django.

1 АНАЛІЗ РОЗРОБКИ WEB-ЗАСТОСУНКУ

1.1 Аналіз та характеристика web-застосунку

У наш час web-застосунки стали невід’ємні майже кожної людини, тому критерії оцінювання стають все більш вимогливими. Користувачі очікують від web-застосунків не тільки великої функціональності, а також і естетичного дизайну, також безпеки та зручності використання. З цим зростанням очікувань від користувачів стає все важливіше для розробників використовувати останні технології так стежити за останніми технологічними практиками розробки. Також стає обов’язковим використання передових інструментів, таких як (TypeScript, Angular, React, або Vue.js) ці фреймворки дозволяють створювати високоякісні web-застосунки, які будуть задовольняти потреби користувачів, а також відповідати сучасним стандартам. Загалом інновації у web-розробці дуже важливі, щоб забезпечити створення якісних web-застосунків, які будуть відповідати всім різноманітним потребам та очікуванням користувача.

Web-застосунок BrilliantHome це електронні сторінки які зберігаються на сервері там можуть бути доступними через інтернет.

При створенні web-застосунку було застосовано фреймворк Angular, це дало можливість застосовувати різноманітні бібліотеки для створення візуальних частин. Для створення multilanguage була застосована бібліотека Angular: i18n, ця бібліотека була створена для забезпечення локалізації web-застосунку. Вона дозволяє перекласти текстові рядки інтерфейсу на різні мови за допомогою (typescript), та усі слова можна зберігати у файлі з форматом (.json, .xlf, .arb, .xmb, .xtb). Також налаштовувати відображення відповідно до налаштувань користувача. Отже використання бібліотеки Angular i18n дозволяє створювати багатомовні web-застосунки, що є дуже важливим для забезпечення комфортного досвіду для користувачі з різних країн та культур.

Також для створення зміни теми інтерфейсу було застосовано Angular Material UI, це дає можливість легко та ефективно реалізувати зміну теми інтерфейсу в web-застосунку. Додавши дві темі світлу і темну це надає користувачу можливість вибору теми, яка найбільше відповідає їх вподобанням та потребам. Це підвищує зручність додатку при його використанні, оскільки користувач може налаштувати його згідно своїм уподобанням у конкретний період часу. Такий підхід до дизайну має забезпечити приємний користувацький від використання web-застосунку «BrilliantHome».

У розробці також дуже важлива частина backend, хоча вона і не являється візуальною частиною. Python, здобув свою популярність завдяки своїй простоті у читабельності коду, а також завдяки дуже великому вибору різноманітних бібліотек. Django є однією з тих бібліотек яка дозволяє швидко та ефективно розробляти backend частину web-застосунків. Його функціональність, готові компоненти, та велика документація робить його більш зручним для розробника. Також Django забезпечує готові рішення для багатьох аспектів web-застосунків, таких як обробка даних користувачів, керування базами даних, та встановлення системи авторизації. Це дозволяє навіть не спеціалістам без глибоких знань у програмуванні підтримувати роботу web-застосунку у Admin панелі. Таким чином, Python та Django роблять процес розробки доступним для різних користувачів та розробників для підтримки та реалізації своїх ідей.

1.2 Аналіз конкурентів

Аналіз конкурентів є невід’ємним при створенні web-застосунку «BrilliantHome». Тільки після аналізу не тільки слабких, а і сильних сторін конкурентів можна пристосувати web-застосунок до потреб ринку та залучити більше користувачів. Крім того аналіз конкурентів допоможе уникнути повторення помилок та виявити можливості для web-застосунку.

1.1.1 OLX

Ця онлайн платформа для купівлі пропонує користувачам великий обсяг різноманітних категорій, включаючи нерухомість та багато іншого. [7]Інтерфейс платформи є простим та інтуїтивно зрозумілим, що робить процес розміщення пошуку легким. Але не зважаючи на ці переваги є мінусі які треба враховувати:

- ❖ Відсутність контролю призводить до появи неякісних та обманливих оголошень.
- ❖ Оскільки OLX це платформа для приватних продажів, то користувачі не можуть маті гарантій стосовно якості товару, що може призвести до незадоволеності покупців.
- ❖ У разі виникнення проблемі з угодою між продавцем та покупцем буде важко отримати підтримку з боку платформи, оскільки OLX фокусуються на самостійних угодах між користувачами.

Також іншим негативним фактором OLX є ведення платних послуг для другого або наступних оголошень.

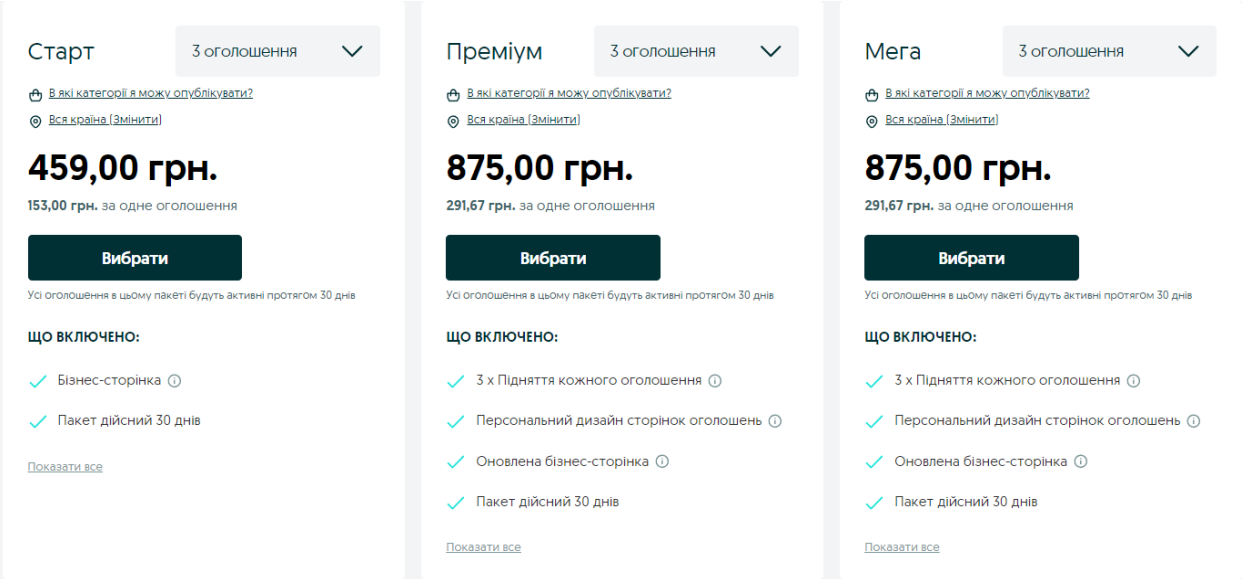


Рис 1.1 OLX (вибір пакету оголошень)

Це може призвести до збільшення витрат для користувачів, які розміщують свої оголошення, і обмежити їх можливість ефективно продавати чи орендувати нерухомість. Також треба зазначити, що мінімальна кількість для придбання оголошень три. Більшість користувачів відмовляються за високих цін.

Але не дивлячись на всі мінуси, ця онлайн web-платформа для купівлі пропонує велике різноманіття категорій, включаючи нерухомості, автомобілі, та багато іншого. Також web-застосунок має інтуїтивно зрозумілий інтерфейс, це робить процес придбання та продажу зручним та більш приємним для користувача web-застосунком, незалежно від досвіду користувача.

Таблиця 1.1

Усі домени OLX

olx.ua	**olx.kz**	**olx.com**	**olx.uz**	**olx.pl**	**olx.ro**
Україна	Казахстан	Загальний	Узбекистан	Польща	Румунія

Також ще однією за переваг web-застосунку це можливість взаємодії між продавцем та покупцями, це дає сприятливі умови для уточнення деталей угоди та уточнення інформації. Завдяки своєму охопленню по світу, OLX відкриває нові можливості для міжнародної торгівлі та спілкування з користувачами web-застосунку з різних країн.

1.1.2 UkrXata

UkrXata пропонує користувачам великий вибір оголошень про продаж та оренду нерухомості в Україні [8]. Його інтуїтивний інтерфейс покращує користувацький досвід при використанні web-застосунку. Однією з переваг є те, що користувач має можливість безкоштовно розміщувати оголошення. Однак сайт може бути перевантаженим інформацією, а також клієнтам не гарантується

достовірність всіх оголошень. Також можуть виникати обмеження в можливості фільтрації та сортування оголошень.

Онлайн платформа УкрХата не має англомовної версії сайту. Це може бути недоліком для туристів. Відсутність англомовної версії web-застосунку доступність ресурсу для людей з різних країн, які шукають нерухомість або мають інтерес до ринку нерухомості в Україні. Це може становити перешкоду для міжнародних інвесторів або туристів які шукають житло.

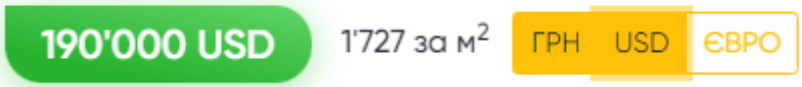


Рис. 1.2 UkrXata (перемикач валюти)

Оголошення на онлайн платформі мають перевагу в тому, що користувач має можливість переглядати ціну у різних валютах (рис. 1.2) таких як: Гривні (₴); Доларах ($); Євро (€);

Це дозволяє зручно перевіряти ціни в різних валютах і легше орієнтуватись для користувачів з різних країн або для тих, хто веде операції у різних валютних одиницях. Такий підхід полегшує процес вибору нерухомості та сприяє доступності інформації для різних користувачів.

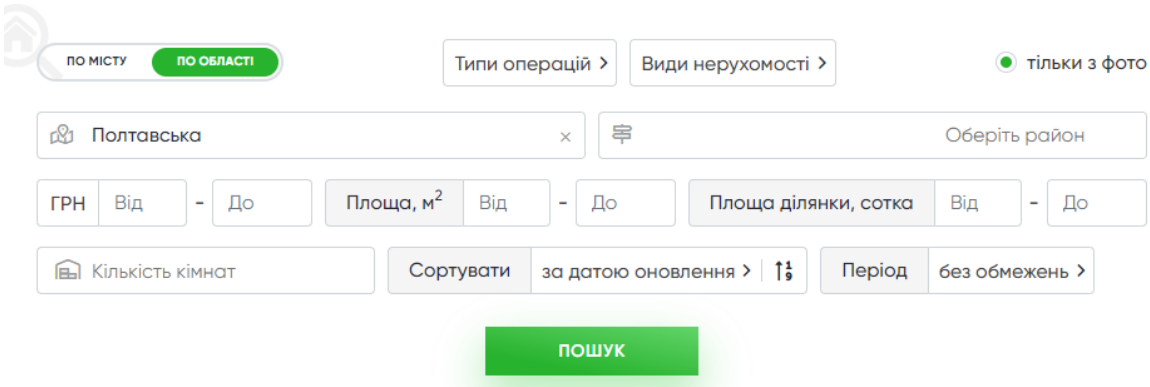


Рис. 1.3 UkrXata (сортування видів нерухомості)

Ще однією перевагою web-застосунку є можливість доступу до різних типів нерухомості, включаючи квартири, будинки, земельні ділянки та комерційну нерухомість. Це дає користувачам можливість знаходити різноманітні варіанти для

свої потреб, незалежно від того, чи шукають вони житло для проживання, чи шукають нерухомість для інвестицій. Широкий спектр доступних опцій робить сайт привабливим для різних категорій користувачів з різними потребами та бажаннями.

1.1.3 LUN.UA

Це популярна онлайн платформа для пошуку нерухомості в Україні. LUN.ua пропонує широкий вибір об’єктів нерухомості [6]. Сайт має доволі простий на інтуїтивно зрозумілий інтерфейс який полегшує пошук нерухомості. Однак головнім недоліком цієї онлайн платформи є відсутність можливості самостійно додавати оголошення про продаж або оренду житла. Це обмеження дає плюси, що кожне оголошення було перевірено за допомогою модерації LUN.ua, однак це стає не сприятливим фактором для користувачів які бажають шукати варіанти для продажу або оренді власної нерухомості.

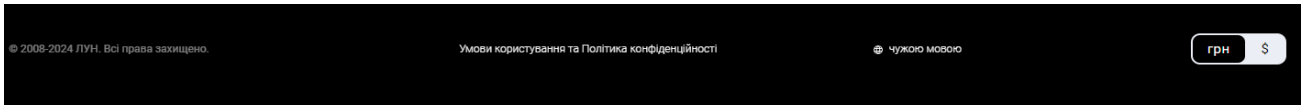


Рис. 1.4 LUN.ua (відсутність перемикача мови)

Ще одним недоліком є відсутність англомовної версії сайту. Це може ускладнити користування онлайн платформою для іноземних користувачів, які не володіють українською мовою. Відсутність англомовної версії інтерфейсу обмежує міжнародну аудиторію та може призвести до втрати потенційних клієнтів та туристів із-за не недостотної доступності інформації.

В онлайн платформі LUN.ua відсутня можливість зміни інтерфейсу. Відсутність такої функції обмежує користувача у можливості адаптувати вигляд інтерфейсу до власних потреб. Наприклад деякі користувачі віддають перевагу темній темі інтерфейсу для зручності використання вночі або в умовах обмеженого освітлення. Відсутність функції зміни теми інтерфейсу може вплинути на зручність користування платформою для деяких користувачів.

LUN.ua має декілька значних переваг. Перш за все web-застосунок LUN.ua пропонує великий вибір нерухомості на продаж та оренду, що охоплює різні міста України, та також охоплює різні типи власності. Також він має інтуїтивно зрозумілий інтерфейс, що призводить для полегшення користувацького досвіду у використанні web-застосунку. Крім того модерація оголошень на платформі допомагає забезпечити у дотриманні правил у поданих оголошень.

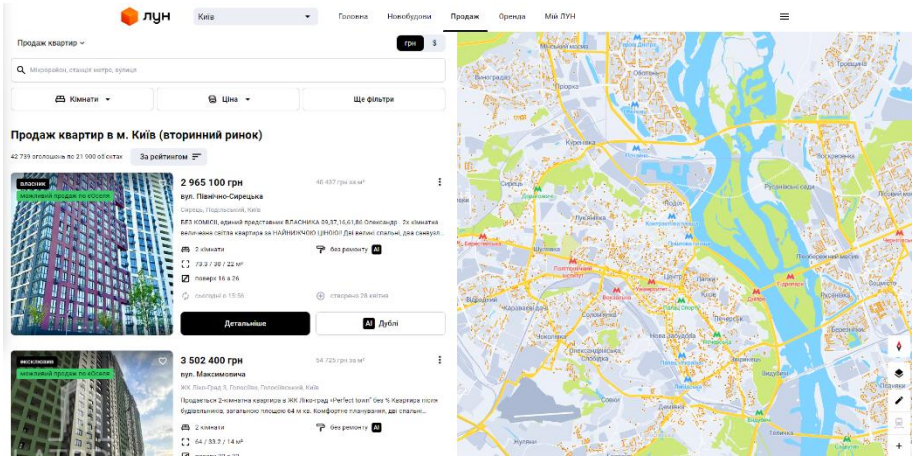


Рис 1.5 LUN.ua (Наявність карти)

Додатковою перевагою є можливість змінити відображення цін у різних валютах. Також на (рис 1.5) зображено зручний перегляд розташування нерухомості на карті, це дозволяє користувачу одразу вибрати бажане розташування, або подивитися де знаходиться будівля яку користувач бажає купити чи взяти в оренду.

1.1.4 Порівняння

1) У наш час наявність можливість перекладу web-застосунку на різні мови є дуже важливою для привертання та збереження уваги туристів з різних країн. Multilanguage дозволяє користувачам вибирати мову інтерфейсу, яка є для них зручнішою або більш зрозумілою. Але особливо важливою є наявність англомовної версії сайту, оскільки англійська одна із найпоширеніших мов світу та є мовою комунікації у багатьох галузях, таких як бізнес, наука так технології. Тому

наявність англомовного інтерфейсу дозволяє web-застосунку привертати більше користувачів з різних країн.

Рис. 1.6 LUN.ua (перемикач)

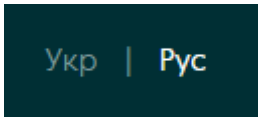


Рис. 1.7 OLX (перемикач)

Рис. 1.8 UkrXata (перемикач відсутній)

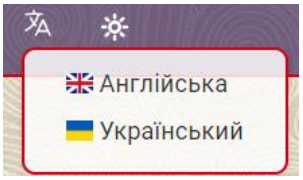


Рис. 1.9 BrilliantHome (перемикач)

На зображеннях вище продемонстровано наявність англійської мови на трьох web-застосунках. Отже ми бачимо наявність англійської мови тільки у «BrilliantHome». Це робить його більш доступним для англомовних користувачів. В інших двох web-застосунків, які зображені на рисунках (1.6), (1.7) відсутня англійська мова, у web-застосунку «UkrXata» (Рис. 1.8) взагалі відсутній перемикач мови, це обмежує доступ до цих платформ для зарубіжних користувачів.

2) Сучасні web-застосунки, такі як LUN.ua та OLX, UkrXata, відіграють значну роль у сучасному світі, надаючи широкий спектр послуг для користувачів. Кожен web-застосунок має свою унікальну спеціалізацію та цільову аудиторію.

Наприклад «LUN.ua» зосереджений на наданні інформації про нерухомість, допомагаючи користувачам знайти відповідне житло до їх бажань та потреб.

З іншого боку, «OLX» став онлайн платформою для різноманітними товарами та послугами від приватних осіб та компаній.

«UkrXata» спеціалізується на наданні інформації про нерухомість. Допомагаючи користувачам знайти відповідну нерухомість, що відповідає потребам та бюджету.

«BrilliantHome» спеціалізуєтеся на ринку нерухомості, забезпечуючи можливість знаходити та розміщувати оголошення про продаж оренду нерухомості. Цей web-застосунок дає користувачам можливість спілкування між потенційним покупцем та користувачем що надає оголошення.

Отже web-застосунки LUN.ua, OLX, УкрХата та BrilliantHome кожен має свою унікальну спеціалізацію.

«LUN.ua» спеціалізується на нерухомості де всі користувачі можуть шукати оголошення про оренду або продаж нерухомості, але не мають можливості додати своє оголошення. [6]

«OLX» – це універсальна платформа де можна знайти не тільки нерухомість, а також і багато чого іншого наприклад автомобілі або електротовари. [7]

«UkrXata» – це онлайн-платформа для купівлі, продажу та оренди нерухомості в Україні. На цьому сайті ви знайдете широкий вибір оголошень про купівлю, продаж та оренду квартир, будинків, земельних ділянок, комерційних будівель та іншої нерухомості по всій Україні. [8]

Що до «BrilliantHome», його спеціальність виключно зосереджена на нерухомості. Надаючи користувачам можливість шукати та розміщувати оголошення про продаж або оренду нерухомості. Кожен з web-застосунків має власну нішу та мету, це дуже важливо враховувати при порівнянні та аналізі.

3) На сайтах LUN.ua, UkrXata та OLX відсутня можливість переключення теми інтерфейсу між темної і світлої теми, що може стати суттєвим недоліком для деяких користувачів web-застосунку. На сайті BrilliantHome присутня функція перемикача між світлої та темної теми інтерфейсу, що може бути корисним для тих, хто віддає перевагу певній темі інтерфейсу, або шукає оптимальний тему web-застосунку для роботи в різних умовах освітлення.

Можливість між світлою і темною темами дозволяє адаптувати інтерфейс web-застосунку до особистих уподобань користувачів, це забезпечує комфортне користування web-застосунком у будь який час доби, або при будь-яких умовах освітлення. Такий функціонал сприяє зручності для користувача, та може позитивно вплинути на загальне враження web-застосунку.

4) На онлайн платформі «LUN.ua» відсутня можливість користувачам самостійно виставляти оголошення про нерухомість. Це може виглядати як обмеження для користувачів, але має свої переваги. Оскільки всі оголошення проходять обов’язкову перевірку та перегляд, це гарантує, що на платформі буде

менше шахрайства. Користувачі можуть бути впевненими у достовірності оголошень та актуальності інформації про нерухомість.

У разі «OLX» можна розмістити тільки одне безкоштовне оголошення про нерухомість, що може бути обмеженням для тих хто бажає активно рекламувати свою нерухомість на цій платформі. Хоча це може здатися обмеженням, але це робить процес розміщення більш контрольованим, це допомагає уникнути спаму та повторення оголошення.

На онлайн платформі «UkrXata» користувачі мають змогу виставляти стільки оголошень, скільки їм потрібно. Не має обмежень на кількість оголошень, які може розмістити користувач web-застосунком. Це дозволяє приватним особам нерухомості ефективно рекламувати свої об’єкти для продажу чи оренди, та перевитрати більше уваги потенційних покупців.

У порівнянні з цим, «BrilliantHome» розміщення оголошень про нерухомість є безкоштовним. Це дозволяє користувачам вільно публікувати оголошення про продаж або оренду нерухомості без додаткових не обов'язкових витрат. Такий підхід зробив BrilliantHome для тих користувачів хто шукає простий та доступний спосіб розмістити своє оголошення та пригорнути увагу потенційних клієнтів.

5) Користувачі web-застосунку «LUN.ua» мають можливість зареєструватися або залогінитися за допомогою різних соціальних мереж, таких як:

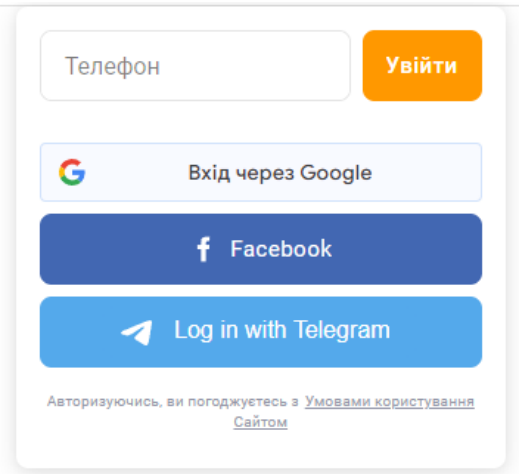


Рис 1.10 LUN.ua (Логін)

Це дозволяє користувачам обрати найзручніший для користувачів метод реєстрації або логіну, і використання існуючих облікових записів у відомих соціальних мережах для швидкого доступу до платформи. Це робить процес більш швидким та легше.

У web-застосунку «OLX» також доступна можливість реєстрації через різні соціальні мережі, такі як:

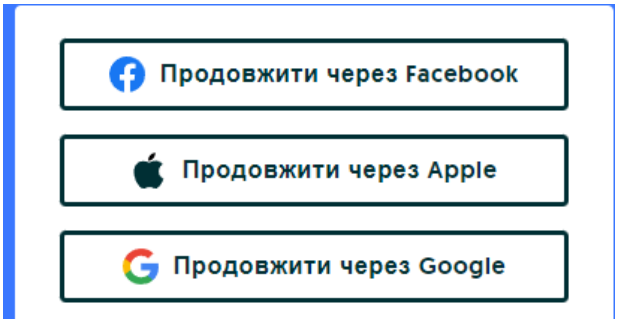


Рис 1.11 OLX (Логін)

Це також дає користувачам вибір у виборі зручного способу реєстрації або логіну, а також забезпечує швидкий та зручний доступ до платформи для тих хто вже був зареєстрований у відповідних соціальних мережах.

У web-застосунку «UkrXata» відсутня можливість входу та реєстрації за допомогою соціальних мереж. Користувачам необхідно створити обліковий запис, водячи Email, номер телефону та пароль. Це може бути недоліком для тих, хто звик до швидкого входу за допомогою своїх облікових записів у соціальних мережах, але з іншого боку це забезпечує більшу безпеку контролю над особистими даними користувача.

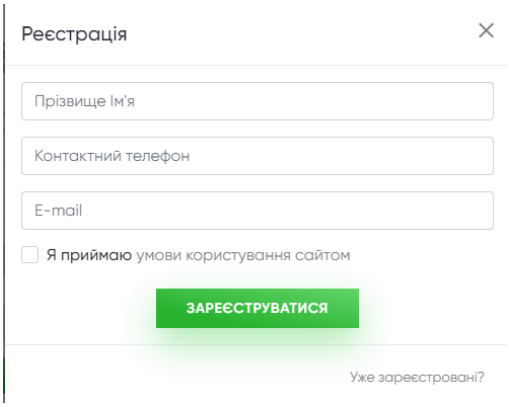


Рис 1.12 UkrXata (Логін)

Користувачі web-застосунку «BrilliantHome» можуть зареєструватися або залогінитися за допомогою соціальних мережах таких як:

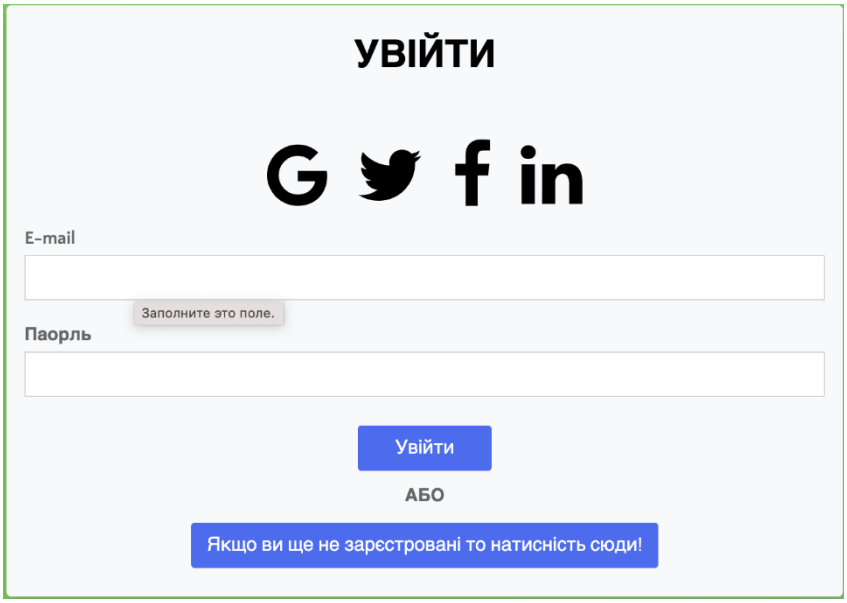


Рис 1.13 BrilliantHome (Логін)

Наявність більшої кількості варіантів реєстрації або логіну дозволяє користувачу вибрати найбільш зручний для них метод, та забезпечує більш широкий доступ до платформи.

6) Швидкість завантаження сайту – це важливий фактор, який впливає на враження та досвід користувача при використанні web-застосунку. Повільне завантаження може призвести до роздратування користувача та негативного враження, таким чином зменшуючи ймовірність повернення на сайт у майбутньому. За вимірами швидкості на трьох платформах.

Таблиця 1.2

Порівняння швидкості

LUN.ua	OLX	UkrXata	BrilliantHome
2040 ms	2650 ms	1028 ms	827 ms

Зазначений час завантаження вказує що BrilliantHome має найшвидше завантаження серед цих чотирьох платформ. Це може бути важливою перевагою, оскільки користувачі web-застосунку шукають швидкість та зручність при роботі з web-застосунком. Швидка загрузка сторінок web-застосунку сприяє позитивному досвіду користувачів при використанні web-застосунку. З іншого боку web-застосунок OLX має час завантаження набагато довше, це може стати недоліком при роботі користувачів з web-застосунком, особливо для тих хто шукає миттєвої відповіді від додатку та негайного доступу до інформації. Такі затримки можуть вплинути на задоволеність при використанні web-застосунком.

У сучасному світі треба враховувати швидкість завантаження, особливо коли користувачі цінують швидкість та ефективність, важливо забезпечити оптимізацію web-застосунку для швидкого завантаження. Це допомагає зберегти користувачів web-застосунку, забезпечивши їм позитивний доступ використання. Також забезпечення швидкості завантаження web-застосунку позитивно впливає на SEO, оскільки пошукові системи такі як «Google» та «Bing», враховують швидкість завантаження сторінок при визначенні їх у рейтингу пошукових результатів.[5]

Отже, детальніше розглянемо отриманні результати. Аналізуючи web-застосунки LUN.ua, OLX, UkrXata та BrilliantHome, можна визначити кілька важливих відмінностей.

Таблиця 1.3

Порівняння web-застосунків

Функції	LUN.UA	OLX	UkrXata	BrilliantHome
Наявність англійської мови у мультимовності	Ні	Ні	Ні	Так
Можливість перемикання між темною або світлою темою	Немає	Немає	Немає	Присутня
Безкоштовне виставлення оголошення	Немає	1 оголошення	Немає	Безліч
Можливість реєстрації за допомогою соціальних мереж	Присутня	Присутня	Відсутня	Присутня
Швидкість загрузки сайту	2040 ms	2650 ms	1028 ms	827 ms

На таблиці (1.3 Порівняння web-застосунків) переведено приклади порівнянь між трьома web-застосунками. Ця таблиця дозволяє зрозуміти, що кожен з web-застосунків відрізняється у різних аспектах, і як це впливає на користувачів. Аналізуючи ці порівняння, можна зробити висновок про те що web-застосунок найбільш відповідає потребам користувачів. Такий аналіз допомагає зрозуміти, як кожен з web-застосунків відповідає на потреби користувача, і які переваги можуть надати.

Web-застосунок BrilliantHome виділяється серед інших своєю унікальністю та комбінацією переваг. Пропонує широкий спектр можливостей для різних користувачів, орієнтованих на нерухомість, що може включати продаж та оренду. Ці переваги роблять web-застосунок більш привабливим вибором для користувачів, які шукають надійну та зручну і ефективну платформу для оренди, купівлі або оренди нерухомості.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Опис технологій для розробки web-застосунку

2.1.1 HTML

HTML (Hyper Text Markup Language) – використовуються як основна мова розмітки та використовується для створення структури web-застосунку. [3] HTML складається з набору атрибутів та тегів, що визначають різні елементи на web-застосунку та їх взаємодії.

- **Атрибути:** вони вказують додаткову інформацію про елементи HTML. Зазвичай додаються до відкриваючого тегу та мають значення наприклад атрибут “href” зазвичай використовується для визначення посилання у “<a href=”URL”>”.
- **Теги:** основні елементи HTML, що визначають зміст та структуру web-застосунку. Теги починаються з відкриваючого тегу <tag> та завершуються закриваючим тегом </tag>. Наприклад теги <h1> - використовується для заголовків; <p> - використовується для абзаців:
- **Загальна структура:** HTML додаток зазвичай складається з таких елементів як: <head> | <title> | <body>. Він також може містити секції для ресурсів, таких як CSS та JavaScript.
- **Елементи:** HTML надає елементи для створення форм, таких як: <input>; <textarea>;ь<select>; Ці елементи дозволяють користувачам вводити дані та вибрати опції в web-застосунку. Після того як користувач введе або вибере потрібні дані, їх можна відправити на обробку даних на сервер.
- **Списки:** у HTML є можливість створювати різноманітні списки та таблиці, так яка і нумеровані так і не нумеровані за допомогою тегів <ol> та <ul>, для нумерованих використовується <ol>, а да маркованих <ul>.

- **Графічні елементи:** За допомогою HTML також можна вставляти графічні зображення за допомогою тега <img> та для відео за допомогою тега <video>.

HTML є основним для будь якого web-застосунку, оскільки він визначає структуру та зміст сторінок [3]. Знати HTML обов'язково важливо для будь якого web-розробника, оскільки ця мова є фундаментом для подальшого вивчення CSS та JavaScript.

2.1.2 CSS

CSS (Cascading Style Sheets) – це мова стилю, яка визначає зовнішній вигляд web-сторінки.[3] Основна мета цієї мови полягає в відокремленні вмісту HTML від його представлення, щоб забезпечити більш гнучке та ефективне управління виглядом web-сторінки. CSS дозволяє користувачу стилізувати різні елементи HTML, включаючи фон, текст, розміри, та багато чого іншого. Основні концепції CSS:

- Вони визначають, які елементи HTML будуть застосовані до правил стилю. Поширені селектори включають елементові селектори наприклад "p" для всіх абзаців, також є класовий селектор наприклад ".my-class" для всіх елементів з класом "my-class" та ідентифікатори селектор наприклад "#mi-id" для елемента з ідентифікатором "mi-id".
- В CSS визначає конкретні аспекти стилю, такі як колір тексту або розмір шрифту, тощо. Значення встановлюють конкретні параметри для цих властивостей. Наприклад якщо користувач бажає задати колір тексту синій та треба прописати 'color: blue'.
- Каскадність в CSS визначає пріоритетність стилів при застосуванні їх до елементів, якщо властивість задана в кількох правилах, вона буде використана відповідно до селектора та порядку вказаних стилів.
- CSS також дозволяє міняти та контролювати розміри та позиції елементів на сторінці. Користувач має можливість застосувати бажану висоту, ширину та

відступи та інші параметри для керування щодо розміщення елементів на web-сторінці.

- Також мова стилів CSS дозволяє створювати анімацію та переходи для створення ефектів руху. Користувач може зробити анімацію зміни властивостей, змінити положення, розмір, кольори, а також застосовувати переходи для плавних змін на web-застосунку.

  CSS є кращим вибором в деяких випадках через його простоту та стандартизованість. Використання CSS є більш простим для початківців, та оскільки вона є основною мовою стилів для web-розробки та не потребує додаткових інструментів [3]. Також CSS має широку підтримку серед браузерів та інших інструментів, це забезпечує стабільну роботу web-застосунку.

2.1.3 Angular

При розробці web-застосунку для оренди та продажу нерухомості під назвою BrilliantHome було обрано Angular як основний фреймворк для розробки. Angular – це інструмент який надає багато можливостей для створення масштабованих web-застосунків.[2, 4]

Angular відзначаються своєю комплексністю та масштабованістю, що робіть його більш зручнішим для великих та складних проектів.

- **Розширені функціональні можливості:**

  Angular надає багато вбудованих функцій та інструментів, наприклад таких як маршрутизація, обробка форм, HTTP-запити, та багато чого іншого. Це дозволяє користувачам легко створювати тяжку функціональність у web-застосунку.

- **Типізація:**

  Angular використовує TypeScript, який дозволяє створювати код зі строгою типізацією. Це дає користувачам полегшення у виявленні помилок та робить код більш стабільним у великих проектах.

- **Офіційна документація:**

  Також Angular має широку та дуже гарно документовану спільноту, а також офіційну документацію. Що робить процес навчання та вирішення проблем більш зручним та швидким для користувачів.

Таблиця 2.1

Порівняння web фреймворків

Функція	**Angular**	**React**	**Vue.js**
Мова програмування	Використовує TypeScript	JavaScript та TypeScript	JavaScript та TypeScript
Віртуальній DOM	Ні	Так	Так
Величина проекту	Для великих проектів	Для будь-яких проектів	Для середніх проектів
Навчальна крива	Помірна	Помірна	Низька
Спільнота	Велика	Велика	Велика
Популярність	Висока	Висока	Висока

За таблицею (2.1) можна зрозуміти, що кожен з представлених фреймворків має свої переваги та недоліки, але Angular виділяється своєю стабільністю та різноманітних можливостей.

Рис 2.1Angular (логотип)

Рис 2.2 React (логотип)

Рис 2.3Vue.js (логотип)

Тому за варіантів (Рис. ‘2.1’, ‘2.2’, ‘2.3’) було обрано Angular оскільки він більше відповідає потребам web-застосунку BrilliantHome. Завдяки використання TypeScript, Angular забезпечує строгу типізацію що сприяє покращення стабільності для проектів.

2.1.4 Django

Django як фреймворк для розробки серверної частини web-застосунку. Django – це фреймворк для Python, який надає широкий спектр функціональності та зручний для розробки серверних web-застосунків.[1, 4]

Рис 2.4 Django (логотип)

Рис 2.5 Flask (логотип)

Рис 2.6 Node.js (логотип)

Рис 2.7 Spring (логотип)

- **Мова програмування:**

  Django побудована на мові програмування Python, він відомий своєю простотою та читабельністю. Він є одним з найпопулярніших мов для програмування у світі, що робить Django доступним для багатьох розробників.

- **Функціональність:**

  Django надає багато вбудованих функцій, таких як аутентифікація користувачів, маршрутизація URL, взаємодія з базою даних через ORM, також наявність адміністративної панелі, та багато іншого. Це дозволяє розробнику швидко та якісно створювати web-застосунки.

- **Швидкодія**

  Django відомий своєю високою швидкодією, також відомий те що він ефективний у роботі з великими обсягами даних. Також він має вбудовану систему кешування, яка дозволяє прискорити роботу web-застосунка та зменшити навантаження на сервер.

- **Документація та спільнота:**

  Django має велику офіційну документацію що робить процес вивчення та пошуку помилок у проекті більш легкою та зручною, особливо для починаючого розробника. Документація дозволяє розробнику швидко знайти відповіді на питання під час розробки.

Таблиця 2.2

Порівняння мов програмування для backend

Функція	**Django**	**Flask**	**Node.js**	**Spring**
Мова	Python	Python	JavaScript	Java
Складність	Помірна	Низька	Висока	Помірна
Швидкодія	Добра	Добра	Висока	Добра
Спільнота	Велика	Велика	Велика	Велика
Розширюваність	Висока	Середня	Висока	Висока
Функціональність	Велика	Середня	Велика	Велика

Таблиця (2.2) надає огляд та порівняння основних характеристик кожного з представлених фреймворків та допомагає визначити що краще обрати для розробки web-застосунку BrilliantHome.

2.2 Вибір та опис інструментів для розробки

2.2.1 VSCode

Visual Studio Code (VSCode) був обраний для верстки web-застосунку BrilliantHome, тому що він має:

Рис. 2.8 Visual Studio Code (логотип)

- **Багатофункціональність:**
  VSCode хоч і вважається блокнотом але має дуже багато корисних функцій таких як підсвічування синтаксису, автодоповнення коду а також інтегрована керування версіями (таких як Git), також VSCode має вбудований термінал, ці функції полегшують роботу розробникам.
- **Розширення:**
  Також VSCode дозволяє налаштувати його під конкретні задачі розробника, В програмі є розширення для підтримки різних мов програмування, також наявна функція роботи зі стилями CSS, також для автоматизації завдань наприклад засоби автоматичного перекладу LESS або Sass, SCSS в CSS, плагіни для роботи з різними фреймворками та бібліотеками.
- **Кросплатформеність:**
  VSCode підтримується на багатьох операційних системах, включаючи такі операційні системи як Windows, MacOS, та дистрибутиви Linux, що робить його найчастішим вибором для розробників які використовують різні операційні системи.

- **Регулярні оновлення:**

  Розробники VSCode постійно вдосконалюють продукт, виправляють помилки, додаючи нові функції та оптимізують програму. Це означає що користувачі можуть бути впевнені в роботі програми.

- **Інтеграція:**

  VSCode дуже просто інтегрується з іншими популярними інструментами розробки, такими як Docker, Kubernetes, та також інструментами тестування та розгортання, що дозволяє розробниками працювати в єдиному середовищі без необхідності переходити між різними програмами.

2.2.2 WebStorm

WebStorm має вбудовану підтримку Angular, він надає розширенні можливості розробки Angular-додатків [11], наприклад як автоматичне розпізнавання структури Angular, та інтегрована підтримка шаблонів Angular та інше.

Рис. 2.9 WebStorm (логотип)

WebStorm має багато вбудованих інструментів для рефакторингу коду та аналізу, також автоматичне виявлення неправильного коду, виправлення проблем стилю та підказки як краще оптимізувати код.

Angular використовує TypeScript як основну мову програмування, WebStorm може забезпечити широкі можливості для роботі з TypeScript наприклад як автодоповнення, перевірка типів та інші інструменти, що дає змогу полегшення роботи розробникам.

WebStorm має багато вбудованих засобів для написання та запуску тестів Angular-додатку, це полегшує процес тестування та написання Unit testing, це підтримує виску якість написаного коду.

Окрім Angular, WebStorm підтримує широкий спектр інших технологій, таких як HTML, CSS, JavaScript, Node.js, що дозволяє комбінувати різні технології у web-проектах.

Таблиця 2.3

Порівняння додатків

Функція	**WebStorm**	**Visual Studio Code (VSCode)**
Підтримка Angular	Так	Через розширення
Рефакторинг коду	Багато інструментів	Основні можливості
Інтегрованість з TypeScript	Повна підтримка TypeScript	Через розширення
Інтеграція з іншими інструментами	Підтримка інтеграцій	Добра інтеграція, але може не стабільна робота
Підтримка тестування	Вбудовані засоби тестування	Через розширення
Вартість	Платна	Безкоштовна

На таблиці (2.3) порівняно два редактора WebStorm та Visual Studio Code (VSCode), обидва редактори мають свою перевагу, але WebStorm більш професіональна середа для розробки ніж VSCode.

2.2.3 PyCharm

PyCharm дає повну інтеграцію з фреймворком Django, це означає що IDE повністю розуміє структуру проектів Django [12], також автоматично генерує код, та надає підказки та підтримку для об’єктів та методів Django, також полегшує виконання різноманітних завдань таких як створення нових web-застосунках, моделей, тощо.

Рис. 2.10 PyCharm (логотип)

PyCharm має вбудовані засоби для виявлення помилок у коді також він надає можливість відлагодження Django-додатків. Для розробників це полегшує виявлення та виправлення помилок у коді, це значно полегшує та скорочує час розробки коду.

PyCharm має можливість легко створювати та управляти віртуальними середовищами Python, це дозволяє ізолювати залежності між проектами. Також PyCharm надає інтегровану підтримку для роботи з системами керування версіями, та автоматизації завдань та іншими інструментами завдань.

IDE також має велику кількість плагінів, що полегшують роботу з проектами Django та іншими фреймворками. Це дає змогу налаштувати середовище розробки під власні побажання та вимоги проекту.

Також ця IDE має вбудовані інструменти для рефакторингу коду та аналізу його якості, це дозволяє розробником зберігати код організованим та легко зрозумілим.

Таблиця 2.4

Таблиця порівняння IDE

Функція	**PyCharm**	**Visual Studio Code (VSCode)**
Підтримка Django	Автоматичне визначення проектів Django.	Можливо через розширення
Виявлення помилок	Вбудовані інструменти для виявлення помилок .	Можливо через розширення, але погано працює
Віртуальні середовища	Інтеграція з віртуальними середовищами.	Можливо через розширення
Рефакторинг	Інструменти для рефакторингу та аналізу.	Можливо через розширення, але погано працює
Розширення та плагіни	Велика кількість плагінів та інтеграцій.	Набагато менше

На таблиці (2.4) показано, що хоча обидва редактора можна використовувати для розробки проектів Django. Але все ж PyCharm має переваги в області інтеграції Django та вбудованих інструментів для роботи з цим або іншими фреймворками.

2.3 Вибір системи управління бази даних

Для розробки web-застосунку «BrilliantHome» була обрана база даних MongoDB, ця база даних NoSQL. Вона відрізняється від звичайних баз даних таких як PostgreSQL, та хмарних баз даних таких як.

Рис. 2.11 MongoDB (логотип)

MongoDB не потребує строгого визначення схеми. Це дає розробникам додавати та змінювати поля в документах безпосередньо в самій базі даних, це полегшує розвиток web-застосунка.

Ця база даних розроблена для горизонтального масштабування, це означає що її можна легко розширити на багато серверів для обробки великих обсягів даних та великої кількості запитів.

Також він може бути швидшим ніж інші бази даних, MongoDB є найшвидшим якщо мова йдеться про операції читання та запис великих обсягів даних, це відбувається завдяки своїй структурі даних та горизонтального масштабування.

Ще одним плюсом MongoDB є що дані зберігаються у вигляді документів у форматі JSON, ц робить базу даних легко зрозумілою та більш зручною для розробників web-застосунків.

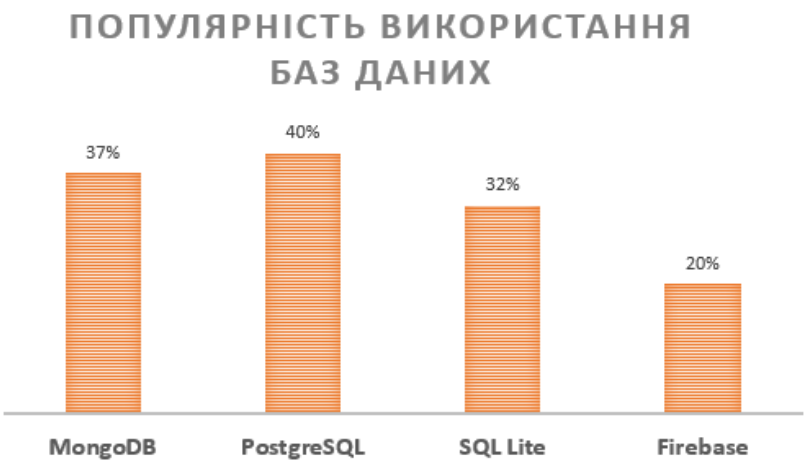


Рис. 2.12 (Популярність використання баз даних)

Таблиця. 2.5

Порівняння баз даних

Функція	**MongoDB**	**PostgreSQL**	**SQL Lite**	**Firebase**
Схема даних	Гнучка, не потребує строгої схеми	Строга, потребує точної структури	Гнучка, але має обмеження	Гнучка, але має обмеження
Масштабованість	Горизонтальне	Горизонтальне та вертикальне	Вертикальне	Горизонтальне
Швидкодія	Швидкий навіть у випадку великих обсягах	Швидкий але не при великому обсязі даних	Швидкий але не при великому обсязі даних	Залежить від обсягу даних
Формат даних	JSON	Традиційний структурований	Традиційний, зберігає у файлах	JSON

Отже при виборі бази даних MongoDB є надійною та гнучкою базою даних, вона підходить для багатьох різних web-застосунків, особливо для тих де потрібна гнучка схема даних та швидка реакція на зміну даних.

2.4 Розробка макету web-застосунку

Для розробки макету була застосована программа Figma вона є найпопулярнішим інструментом для створення UI/UX дизайну завдяки своїй функціональності

Рис. 2.12 Figma (логотип)

У дипломній роботі було розроблено макет web-застосунку BrilliantHome вибір Figma для створення макету обумовлений тим що вона має кросплатформеність це означає, що програма працює в браузері, і це робить її доступною для будь якого пристрою без необхідності встановлення додаткового програмного забезпечення. Тако Figma надає можливість для створення інтерактивних прототипів, які можна використовувати для тестування функціональності та юзабіліті до початку програмування. Вона також інтегрується за іншими інструментами для розробки наприклад такими як: Zeplin, Jira, Slack ці програми полегшують передачу макетів розробникам.

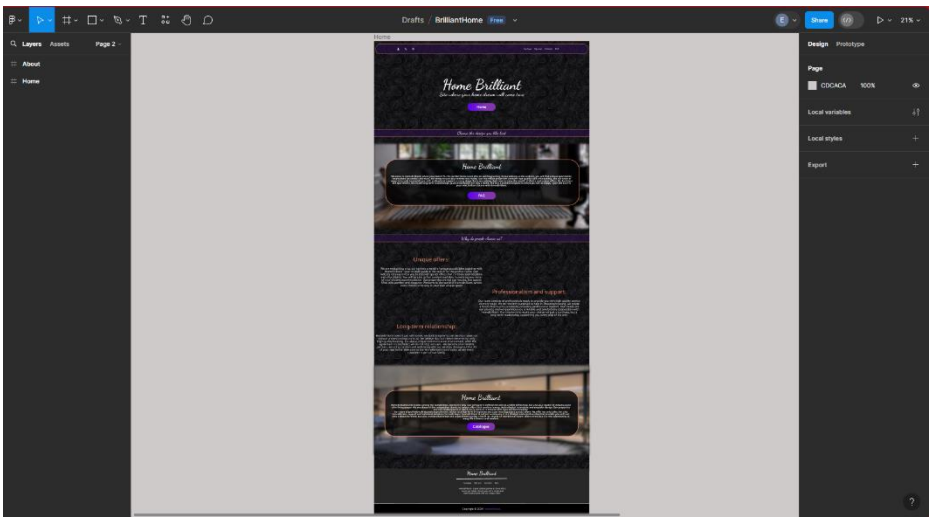


Рис. 2.12 Layers Figma (макет головної сторінки)

На рисунку 2.12 представлений макет головної сторінки web-застосунку BrilliantHome який був створений за допомогою Figma. На цьому макеті можна основні функціональні елементи та навігацію по web-застосунку, це дозволяє побачити майбутній функціонал сторінки.

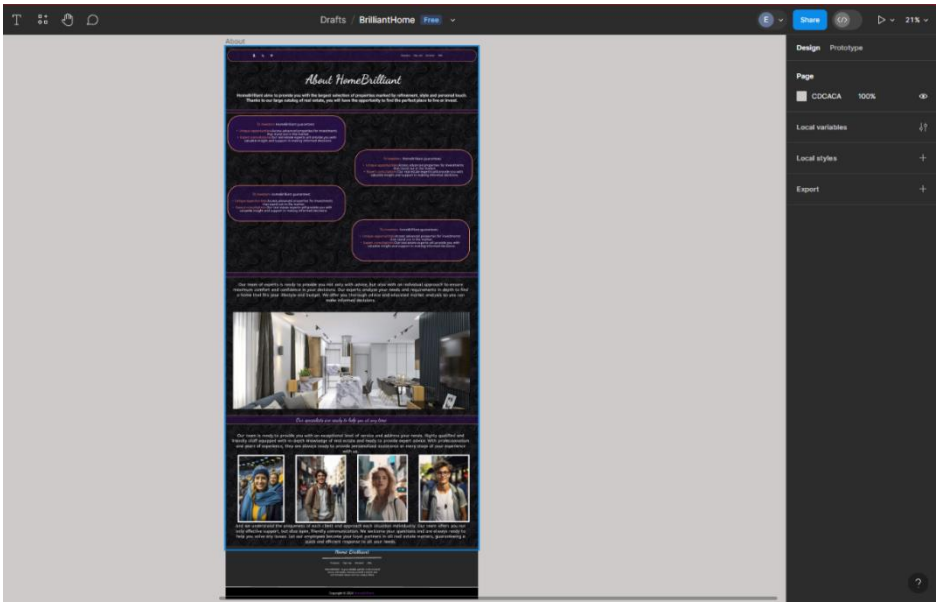


Рис. 2.13 Layers Figma (макет сторінки About)

На рисунку 2.13 зображенний макет сторінки About він включає інформацію про web-застосунок це дозволяє більше дізнатися про сторінку до її розробки. Використання Figma у процесі розробки web-застосунку BrilliantHome дозволило забезпечити якісний дизайн, та розрахувати зручність для кінцевих користувачів та значно допомагає скоротити час розробки.

3 РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ

3.1 Оцінка та планування web-застосунку

У процесі розробки web-застосунку BrilliantHome необхідно скласти список задач для того щоб реалізувати його для мінімальної життєздатності (MVP).[4]

Для початку розробки web-застосунку BrilliantHome було вирішено створити функціональні та нефункціональні вимоги. Визначення цих вимог є важливим етапом у процесі створення web-застосунку тому, шо вони створюють критерії, які має виконувати застосунок та забезпечують всі необхідні аспекти для коректної роботи.

Таблиця. 3.1

Функціональні вимоги:

1	Створення облікового запису із збереженням особистої інформації та здатністю увійти до системи.
2	Користувачі можуть додавати оголошення про продаж та оренду нерухомості.
3	Користувачі можуть сортувати нерухомість за критеріями такими як ціна.
4	Можливість вибору мови інтерфейсу (українська та англійська) та перемикання між світлою і темною темою інтерфейсу.

На таблиці 3.1зображенні функціональні вимоги які визначають конкретні функції, які система має виконувати. Вони описують, що система має робити та встановлюють правила взаємодії між користувачем та системою. Зазвичай функціональні вимоги визначають опис основних функцій

Таблиця. 3.2

Нефункціональні вимоги:

1	Забезпечення конфіденційності персональних даних користувачів з застосуванням відповідних методів шифрування.
2	Швидке завантаження сторінок та швидкі відгуки на запити користувачів. (827 ms)
3	Забезпечення коректної роботи в різних браузерах. (Google Chrome, Opera, Safari, Edge)
4	Адаптивний дизайн з (375px на 667px) до (2560px на 10180px)

На таблиці 3.2 зображенні нефункціональні вимоги визначають системні атрибути та його умови. Воно описує як саме система має виконувати різні функції, До нефункціональних вимог відносяться вимоги до продуктивності та безпеки, маштабованості та багато інших аспектів якості системи.

Також була розроблена діаграма варіантів використання (Use Case Diagram) вона є не менш важливим внстркментом у розробці програмного забезпечення, він використовується для візуалізації функціональних вимог web-застосунку. Також Use Case Diagram показує взаємодію між акторами та системою.

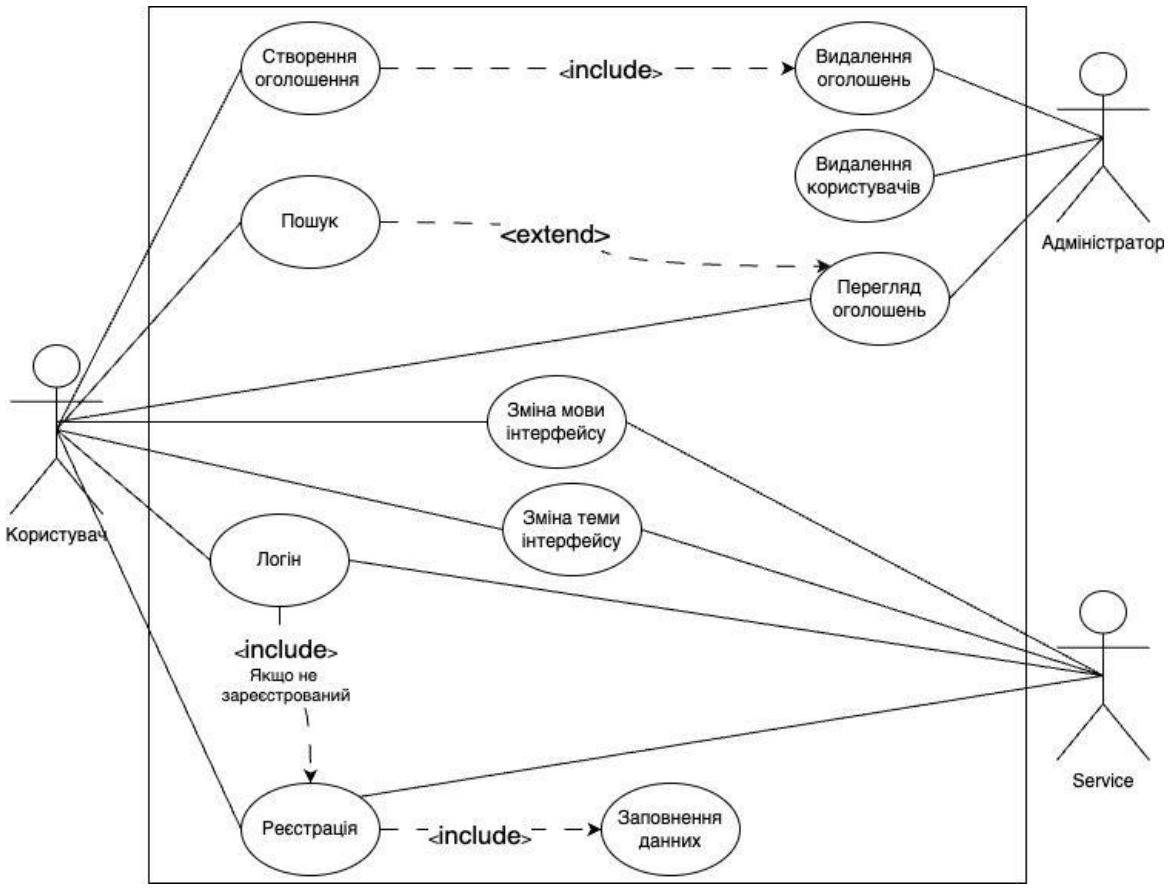


Рис 3.1 (Use Case Diagram)

На рис 3.1 представлена Use Case Diagram яка відображає взаємодію між користувачем системи та її функціями. У Use Case Diagram є три основні актора це “Користувач”,”Адміністратор” та “Service”.

Користувач це основний в системі, який може виконувати різні дії, такі як реєстрація. логін, зміна теми налаштувань та інші. Адміністратор це користувач з розширеними правами, який має можливість видаляти оголошення та користувачів, та переглядати оголошення. Service цей актор взаємодіє з системою для виконання певних сервісних дій.

<include> - цей тип взаємозв'язку означає, що один варіант використання включає в себе інший. Наприклад те що “Реєстрація” включає в себе заповнення даних.

<extend> - цей тип взаємозв'язку означає, що один варіант використання може розширювати інший. Наприклад “Пошук” може розширюватися до “Перегляд оголошень”.

Отже ця Use Case Diagram показує, як різні типи користувачі взаємодіють ші системою та які функції можуть виконувати.

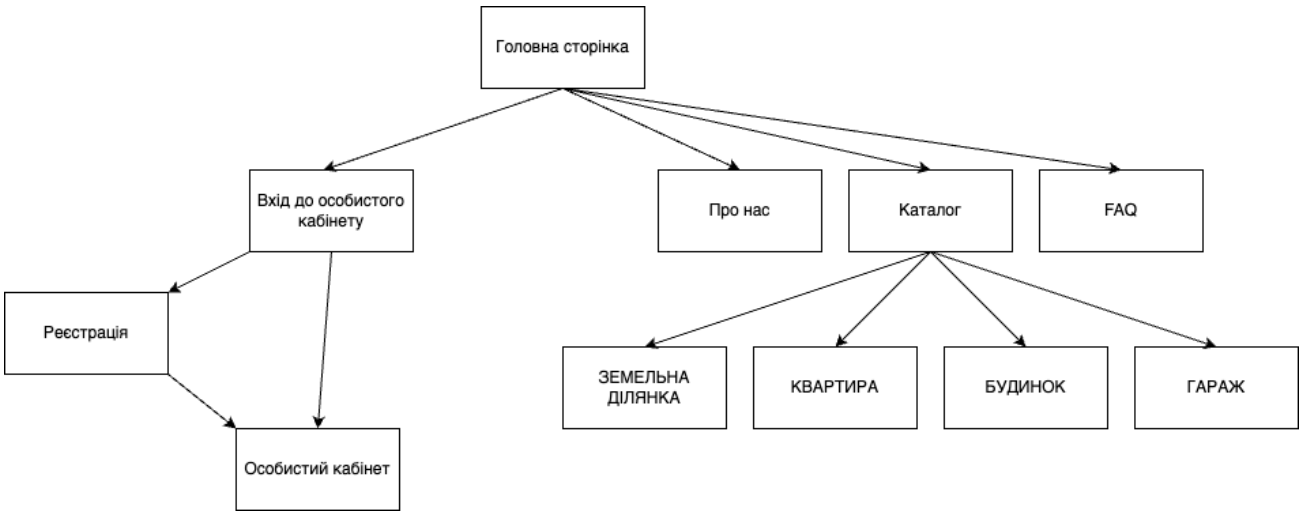


Рис 3.2 (Діаграма карти сайту)

На рис. 3.2 показа карта сайту, починаючи з головної сторінки. З головної сторінки можна перейти в особистий кабінет, але якщо користувач не зареєстрований то він може перейти на форму реєстрації, після чого він матиме доступ до особистого кабінету. Також користувач має можливість переходити на сторінки як (Про нас, Каталог, FAQ), з каталогу користувач може перейти на ще три сторінки з вибором каталогів(Земельна ділянка, Квартира, Будинок, Гараж). Ця структура відображає основні шляхи навігації користувача на сайті.

3.2 Опис інтерфейсу користувача

Опис інтерфейсу користувача це важливий розділ, що дає більш детальну інформацію про те, як користувачі будуть взаємодіяти із системою

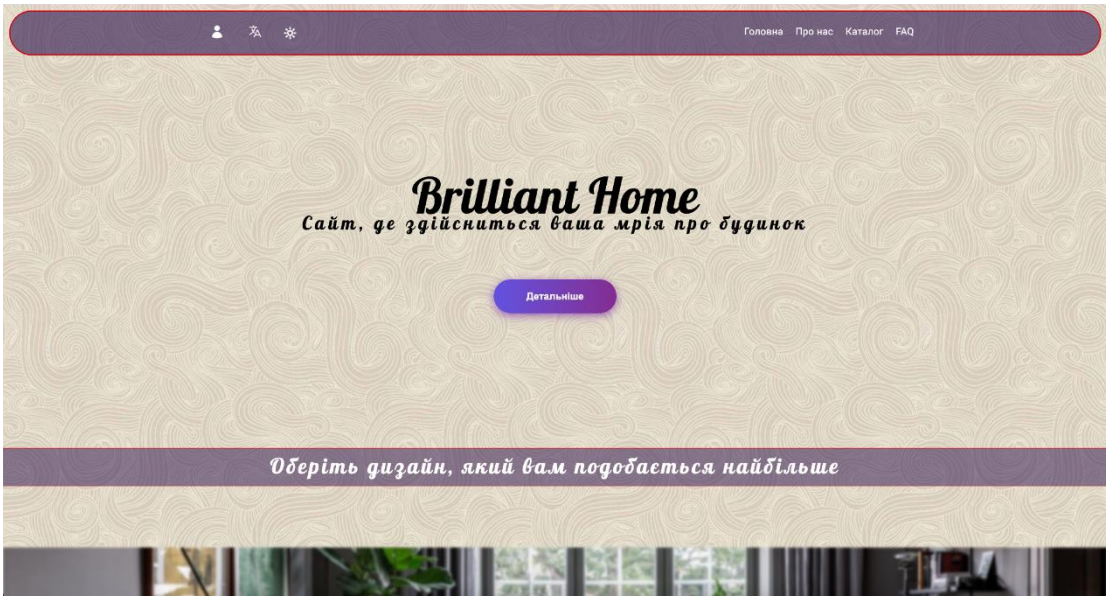


Рис. 3.3 Головна сторінка

На рисунку 3.3 зображення головна сторінка web-застосунку BrilliantHome, на цій сторінці користувачі можуть побачити основну навігаційну панель яка включає в себе основні посилання на розділи(Особистий кабінет, Про нас, Каталог, FAQ)

Меню навігації розташоване у верхній частині сторінки та воно не залишається доступним при прокручуванні сторінки. Та воно включає в себе наступні елементи:

- Головна - повернення на головну сторінку.
- Про нас - інформація про web-застосунок.
- Каталог - перехід на сторінку де представлений каталог.
- FAQ - на цій сторінці представленні питання що задаються частіше.

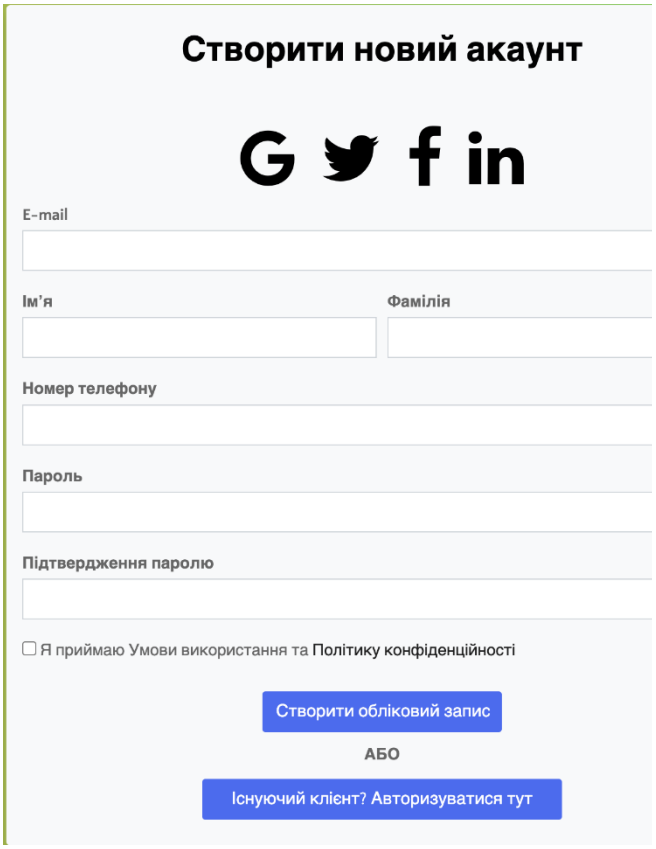


Рис. 3.4 Форма реєстрації

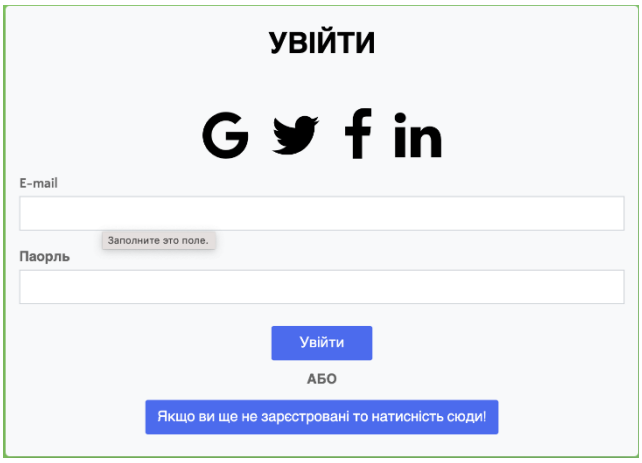


Рис. 3.5 Форма логіну

На рис. 3. зображено форму реєстрації вона в себе включає поля для ведення особистих даних користувача, таких як (Ім'я Прізвище, пароль та електронна адреса) після заповнення всіх даних користувач натискає на кнопку "Створити обліковий запис", після чого система переверія усі введені дані та реєструє користувача

На рис. 3.5 зображено форму логіну вона в себе включає поля для ведення особистих даних раніше зареєстрованого користувача, такі як (електронна пошта та пароль) після заповнення их даних користувач повинен натиснути кнопку "Увійти" після натискання система здійснює перевірку якщо всі дані введені правильно то користувач переходить на сторінку особистого кабінету, але якщо користувач не зареєстрований або невірно ввів дані то сторінка видасть помилку та відчистить усі поля.

Рис 3.6 Головна сторінка (мобільна адаптація)

Рис 3.7 Головна сторінка (мобільна адаптація з відкритим меню)

На мобільних пристроях інтерфейс автоматично адаптується як на рис. 3.6 змінюючи розташування елементів для того щоб користувачам було зручно використовувати web-застосунок на мобільних пристроях. Наприклад для зручності меню навігації перетворюється на “гамбургер-меню” як на рис 3.7, а всі кнопки та форми змінюють положення та розмір для оптимального відображення на мобільних пристроях.

Рис. 3.8 Головна сторінка (світла тема)

Рис. 3.9 Головна сторінка (темна тема)

Користувач може змінювати тему інтерфейсу на рис 3.8 та 3.9 зображені світла та темна тема яку користувач може обрати натиснувши на иконку сонця або місяця, залежно від обраного положення буде обрана світла або темна тема

Рис. 3.10 Головна сторінка (Українська мова)

Рис. 3.11 Головна сторінка (Англійська мова)

Також користувач може змінювати мову інтерфейсу на рис 3.10 зображено українська версія web-застосунку, а на малюнку 3.11 Англійська версія, також на зображеннях було зображено перемикач для мови.

3.3 Розробка клієнтської частини

В цьому розділі детально описується процес розробки клієнтської частини (frontend) для web-застосунку BrilliantHome з використанням фреймворку Angular. Цей фреймворк використовується для побудови структури та функціональності односторінкового додатку (SPA) [2, 9], також Angular надає вибір інструментів для розробки та можливість для тестування за допомогою встановлених бібліотек. Також використовувалися HTML5 для структурування контенту та CSS3 для стилізації та оформлення web-застосунку. Замість звичайного JavaScript, для написання коду використовується TypeScript це дає кращу типізацію та інструменти для розробки.

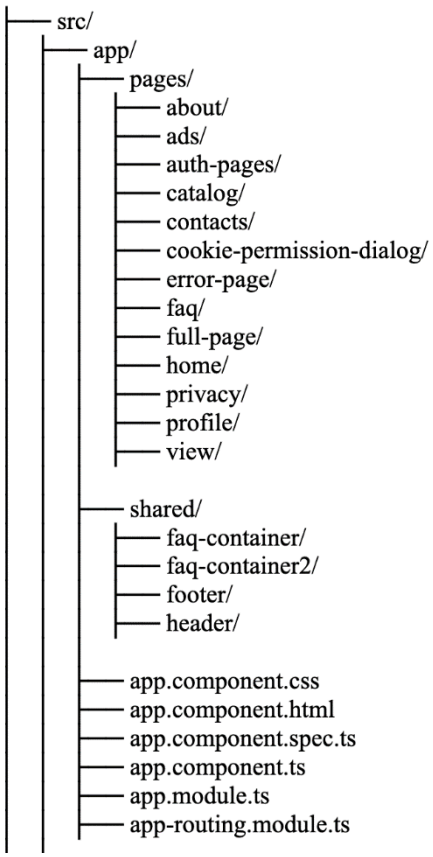
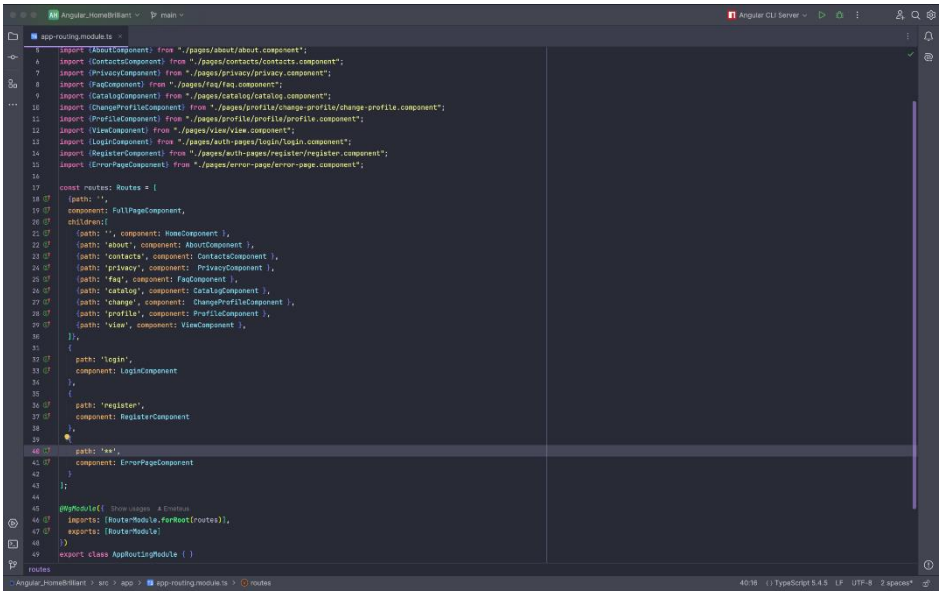


Рис. 3.12 Структура проекту Angular

Основною папкою є “src” вона містить весь код, наступна папка це “app” - головна папка для додатку Angular вона містить всі компоненти та модулі.



```
import { AboutComponent } from "./pages/about/about.component";
import { ContactsComponent } from "./pages/contacts/contacts.component";
import { PrivacyComponent } from "./pages/privacy/privacy.component";
import { FaqComponent } from "./pages/faq/faq.component";
import { CatalogComponent } from "./pages/catalog/catalog.component";
import { ChangeProfileComponent } from "./pages/profile/change-profile/change-profile.component";
import { ProfileComponent } from "./pages/profile/profile/profile.component";
import { ViewComponent } from "./pages/view/view.component";
import { LoginComponent } from "./pages/auth-pages/login/login.component";
import { RegisterComponent } from "./pages/auth-pages/register/register.component";
import { ErrorPageComponent } from "./pages/error-page/error-page.component";

const routes: Routes = [
  {path: '',
  component: FullPageComponent,
  children:[
    {path: '', component: HomeComponent },
    {path: 'about', component: AboutComponent },
    {path: 'contacts', component: ContactsComponent },
    {path: 'privacy', component:  PrivacyComponent },
    {path: 'faq', component: FaqComponent },
    {path: 'catalog', component: CatalogComponent },
    {path: 'change', component:  ChangeProfileComponent },
    {path: 'profile', component: ProfileComponent },
    {path: 'view', component: ViewComponent },
  ]},
  {
    path: 'login',
    component: LoginComponent
  },
  {
    path: 'register',
    component: RegisterComponent
  },
    path: '**',
    component: ErrorPageComponent
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Рис 3.13 app.routing.module.ts

Файл “app.routing.module.ts” рис 3.13 відповідає за маршрутизацію у додатку. Він визначає які компоненти повинні завантажуватися для різних URL. Для

початку треба імпортувати компонент сторінки на яку ми хочемо перейти це робиться за допомогою:

```
import {FullPageComponent} from
"./pages/full-page/full-page.component";
```

Рис 3.14 Импортування компоненту

Після того як компонент був імпортований рис 3.14треба йому написати URL шлях зазвичай усі web-застосунки поки розробляються мають адресу http://localhost:4200/ , але для того щоб перейти нам на іншу сторінку треба оголосити новий маршрут.

```
{path: 'about', component: AboutComponent },
```

Рис 3.15 Визначення масиву “path”

Тут ми визначаємо масив об'єктів рис 3.15 маршрутів, і тут кожен об'єкт має “path” це сам шлях URL та “component” це компонент який повинен завантажуватись при навігації цього шляху. Якщо вказати порожній шлях (“ ‘ ’ ”) то це буде завантаження головної сторінки.

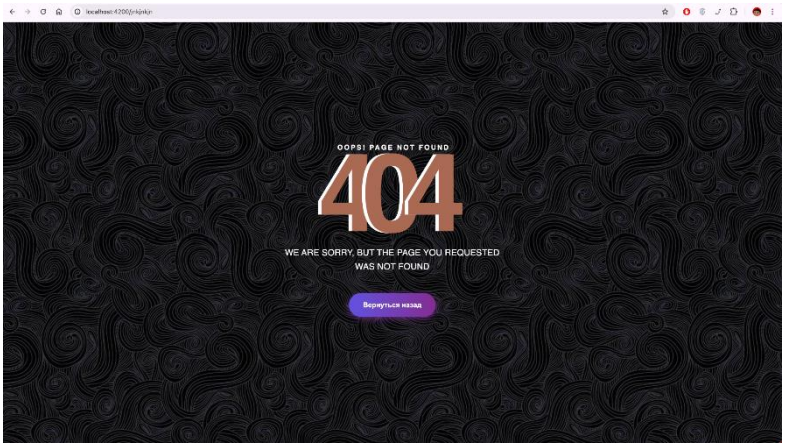


Рис 3.16 (404 page)

Якщо вказати маршрут (“ ‘**’ ”) цей шлях відповідає за будь-якому, який не був визначений раніше, цей шлях використовуються для сторінки 404 або Error Page рис. 3.16.

Для реалізації перемикача світлої та темної теми у web-застосунку була використана мова програмування TypeScript. Логіка самого перемикача була написана в основному компоненти додатку (“app.component.ts”) рис 3.17. Для реалізації був написаний метод “SetTheme”, він визначає яку тему треба застосовувати та зберігає її у “cookies” нижче приведений приклад коду для реалізації.

```
    private setTheme(theme: string, body: HTMLElement,
main: HTMLElement | null, moonIcon: HTMLElement | null,
sunIcon: HTMLElement | null, footer: HTMLElement | null):
void {
     if (theme === 'dark') {
      this.applyDarkTheme(body, main, moonIcon, sunIcon,
footer);
    } else {
      this.applyLightTheme(body, main, moonIcon, sunIcon,
footer);
    }
    this.cookieService.set('theme', theme, 365); }
```

Рис 3.17 Визначення масиву “path”

Також для керування відображенням іконок перемикача теми були написані методи “ShowMoonIcon” та “ShowSunIcon” нижче приведено приклад написаного коду для реалізації перемикача рис. 3.18.

```
    private showMoonIcon(moonIcon: HTMLElement | null,
sunIcon: HTMLElement | null): void {
      if (moonIcon) {
        moonIcon.style.display = 'block';
      }
      if (sunIcon) {
        sunIcon.style.display = 'none';
      }
    }

    private showSunIcon(moonIcon: HTMLElement | null,
sunIcon: HTMLElement | null): void {
      if (moonIcon) {
        moonIcon.style.display = 'none';
      }
      if (sunIcon) {
        sunIcon.style.display = 'block';
      }
    }
```

Рис 3.18 Реалізація перемикача світлої та темної теми

У результаті використання цього коду, користувач web-застосунку при бажанні може змінювати тему з світлої на темну та навпаки за допомогою простого перемикача. Та поточна тема зберігається у cookies, це дозволяє зберігати вибір користувача навіть після перезавантаження сторінки, дані зберігаються у cookies 365 днів, або пока користувач не видалить їх самостійно.

Для реалізації мультимовності в web-застосунку який написаний за допомогою Angular була використана бібліотека “ngx-translate” та “i18n”. Ці бібліотеки дають змогу для додавання багатомовності та дозволяють легко змінювати мову інтерфейсу в реальному часі та зберігати вибір користувач у файлах cookies.

Для початку була встановлена бібліотека за допомогою команди “npm install @ngx-translate/core та @ngx-translate/http-loader” Та після встановлення пакетів обов'язково треба налаштувати “app.module.ts”.

```
    imports: [
     HttpClientModule,
     TranslateModule.forRoot({
       loader: {
         provide: TranslateLoader,
         useFactory: HttpLoaderFactory,
         deps: [HttpClient]
```

Рис 3.19 Налаштування модулів

У коді наведеному вище рис. 3.19 показано налаштування для ‘TranslateModule’ та також зазначено “TranslateHttpLoader” він використовується для завантаження файли перекладу через HTTP.

```
    export function HttpLoaderFactory(http: HttpClient) {
          return       new       TranslateHttpLoader(http,
'./assets/i18n/', '.json');
    }
```

Рис 3.20 Логіка файлу ‘TranslateModule’

Файли перекладу збереженні у папці “assets/i18n” та файли повинні бути збережені у форматі JSON, всі налаштування показні вище у файлі “app.module.ts” рис . 3.20.

```
"home": "Home",
"about": "About",
"catalog": "Catalogue",
"faq": "FAQ",
"login": "Login",
"register": "Register",
"eng": "English",
"ukr": "Ukrainian",
```

Рис 3.21 en.json

```
"home": "Головна",
"about": "Про нас",
"catalog": "Каталог",
"faq": "FAQ",
"login": "Увійдіть",
"register": "Зареєструйся",
"eng": "Англійська",
"ukr": "Український",
```

Рис 3.22 ua.json

На рис. 3.21 та 3.22 зображенні JSON файли, кожен з них представляє, окрему мову наприклад en.json представляє сайт на англійській мові, а файл ua.json представляє сайт на українській мові. Для відображення перекладів у шаблонах компонентів HTML використовується директива “translate” рис. 3.23.

```
<li><a routerLink="" data-lang-id="home">{{"home" | translate}}</a></li>
<li><a routerLink="about" data-lang-id="about-hed">{{"about" | translate}}</a></li>
```

Рис 3.23 Директива відображення.

Метод “ngOnInit” використовується в Angular проектах для виконання дій для ініціалізації компонента. ngOnInit - це метод циклу,який викликається після створення компоненту і цей метод не може існувати без другого методу “:void” він означає що метод повертає значення рис. 3.24.

```
  ngOnInit(): void {  no usages
  }

}
```

Рис 3.24 Метод ngOnInit

Також для зміни заголовка сторінки, для того щоб допомагати користувачам швидко розуміти, яку сторінку вони відкрили, також Встановлення відповідних заголовків сторінок покращує видимість сайту у пошукових системах. Для реалізації в TypeScript треба прописати “document.title = ‘..’ ” як зображено на рисунку 3.25

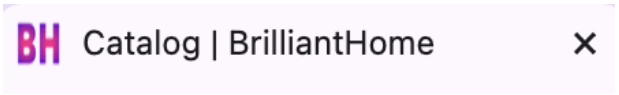


Рис 3.25document.title

Рис .3.26 Приклад зміни назви вкладки

На рис.3.26 зображено що після переходу на сторінку каталогу та спрацював метод document.title і назва вкладки була змінена на яка була зазначена на рис. 3.25 та змінила свою назву на “Catalog | BrilliantHome”.

3.4 Розробка серверної частини

В розділі було детально описано процес розробки серверної частини (backend) для web-застосунку BrilliantHome. Була застосована мова програмування Python та з використанням фреймворку Django, це дозволяє ефективно реалізувати логіку для серверної частини. Та реалізувати взаємодію з базою даних і обробку HTTP-запитів.[1, 10]

Серверна частина BrilliantHome є ключовою роллю у забезпеченні функціональності web-застосунку, воно повинно обробляти запити від клієнтів, також повинен виконувати взаємодію з базою даних, та забезпечити безпеку даних. Використання мови програмування Python з використанням фреймворком Django дозволяє швидко та ефективно реалізувати складні функціональні можливості

Django має свою архітектуру серверної частини базується на принципах MVC(Model-View-Controller) та REST API. Model відповідає за структуру даних і взаємодію з базою даних. View відповідає за відображення даних, та формує HTTP запити.

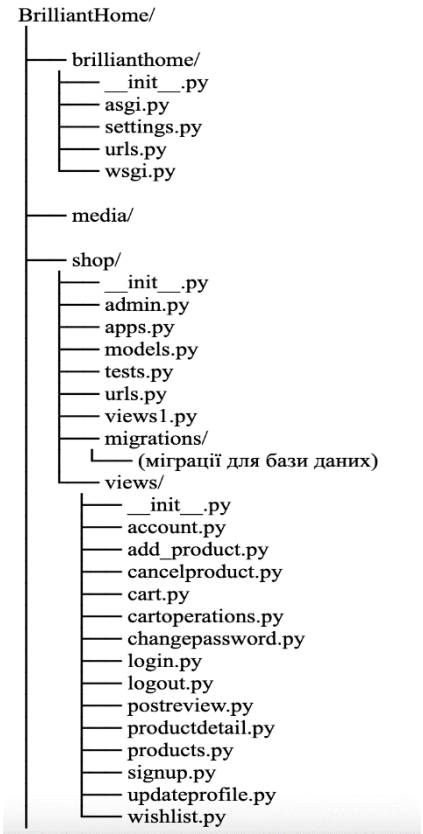


Рис 3.27 Архітектура проекту Django

На рис. 3.27 представлена архітектура проекту Django, на ній ми можемо побачити головну папку з проектом під назвою “BrilliantHome”, у цій папці головними файлами є “brillianthome ” та “shop”, У папці “brillianthome ” є файл під назвою settings.py він є важливо частиною проекту оскільки він містить налаштування та конфігурації які необхідні для коректної роботи web-застосунку.

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'brilliant',
        'CLIENT': {
            'host': 'mongodb+srv://afmakhmud:OCiJHXcMZCKcaNZL@brilliant.5qphaag.mongodb.net/?retryWrites=true&w=majority&appName=brilliant'
            'tlsCAFile': certifi.where(),
        }
    }
}
```

Рис 3.28 Підключення до бази даних

На рис. 3.28 зображено фрагмент коду за файла settings.py, цей код допомагає підключитися до бази даних MongoDB. Цей код містить ім’я бази та “host”, хост це посилання на базу даних, це посилання містить ім’я користувача який відправляє

POST запити також є унікальний пароль для підключення бази даних, також це посилання містить назву. Після підключення потрібно застосувати міграції. Команди “makemigrations | migrate”, дозволяють автоматизувати зміни у структурі бази даних, щоб вона відповідала змінам у моделях додатку.

```
(.venv) emeteus@MacBook-Pro-Emeteus BrilliantHome % python3 manage.py makemigrations
Migrations for 'shop':
  shop/migrations/0002_product_stock_alter_cart_id_alter_category_id_and_more.py
    - Add field stock to product
    - Alter field id on cart
    - Alter field id on category
    - Alter field id on customer
    - Alter field profile_pic on customer
    - Alter field code on order
    - Alter field id on order
    - Alter field id on orderproduct
    - Alter field id on product
    - Alter field tag on product
    - Alter field id on review
    - Alter field id on wishlist
(.venv) emeteus@MacBook-Pro-Emeteus BrilliantHome %
```

Рис 3.29 Виконання комани makemigrations

На рис. 3.29 показано виконання роботи команди “python manage.py makemigrations”, ця команда ствоює нові міграції на основі змін у проекті, також вона аналізує моделі та генерує файли міграцій які описують зміни, які потрібно обробити в базі даних.

```
(.venv) emeteus@MacBook-Pro-Emeteus BrilliantHome % python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, shop
Running migrations:
  Applying shop.0002_product_stock_alter_cart_id_alter_category_id_and_more... OK
(.venv) emeteus@MacBook-Pro-Emeteus BrilliantHome %
```

Рис 3.30 Виконання комани migrate

На рис. 3.30 показано виконання роботи команди “python manage.py migrate”, ця команда застосовє міграції до бази даних, вона виконує ті зміни. які були описані

в файлах міграцій, створених командою makemigrations. Ця команда змінює структуру бази даних, створюючи таблиі, також може змінювати колонки, видаляти таблиці та інші структури Отже команди “python manage.py makemigrations” та “python manage.py migrate” синхронізують зміни у моделях Django проетку з базою даних MongoDB, та забезпечує простий спосіб керування змістами в структурі бази даних.

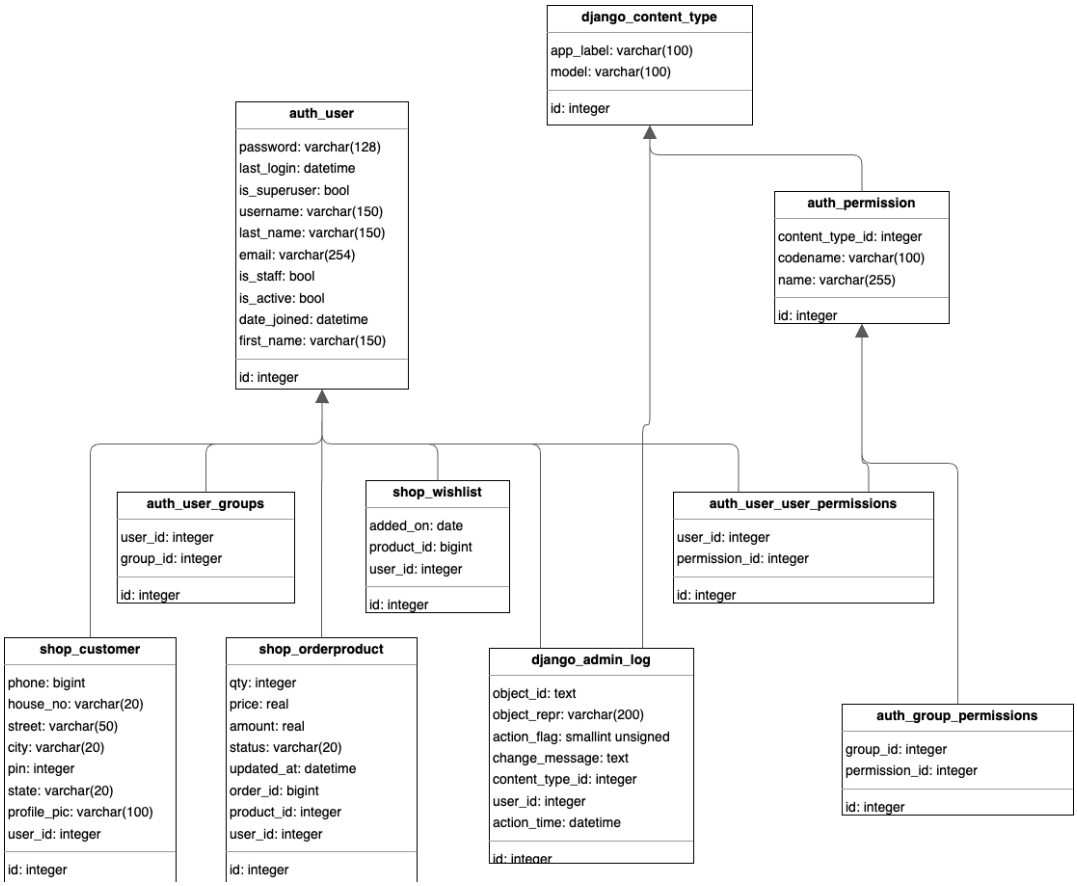


Рис 3.31 Діаграма бази даних

На рис. 3.31 зображена діаграма класів вона представляє структуру бази для системи управління користувачами там web-застосунком на основі Django, вона складається з взаємопов'язаних таблиць. Основною таблицею є “auth_user” вона зберігає інформацію про користувача, та зв'язується з іншими таблицями таких як “auth_user_groups”, “auth_user user_permissions”.

Таблиця “auth_permission” містить дозволи та зв'язується з таблицею “django_content_type”, яка визначає типи контенту. Адміністративні дії

проводяться в таблиці “django_admin_log”, яка також зв'язується з “auth_user” та “django_content_type”. Така структура може забезпечити комплексне управління користувачами, групами та дозволами та діями у системі.

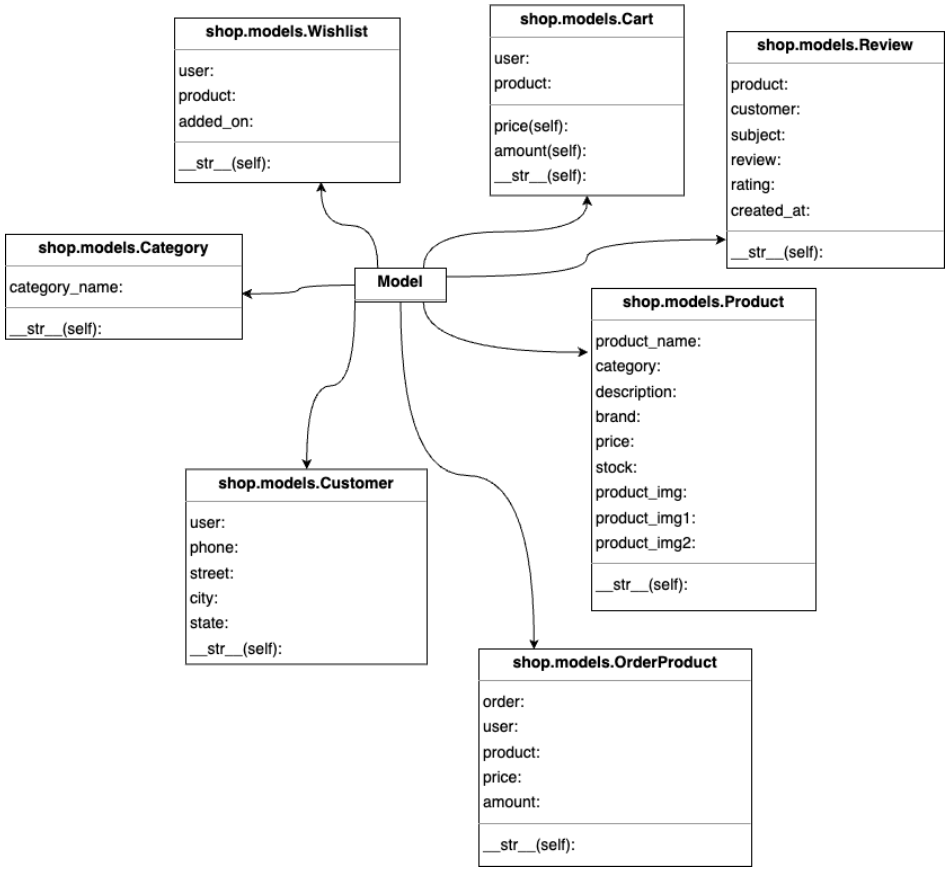


Рис 3.32 Діаграма класів

На рис. 3.32 зображена діаграма класів яка представляє моделі Django. Вона представляє, що всі класи успадковуються від базового класу “model.py” Це загальна практика Django [1, 10], де всі моделі створюються як підкласи від “django.db.models.Model”. Це забезпечує основний функціонал (ORM), це дозволяє автоматично перетворювати записи з бази даних у Python та забезпечує основні методи для створення, оновлення та читання, видалення записів у базі даних. Також це дозволяє легко створювати зв’язки типу “один до одного”, “один до багатьох” та “багато до багатьох” між різними моделями.

Отже спадкування від базового класу “model.py” це фундоментальна частина структури Django проекту, що забезпечує гнучкість та зручність роботи за базами даних.

Адмін панель Django це вбудований інструмент для управління контентом та користувачами web-застосунку. Він дозволяє адміністраторам легко керувати даними, які зберігаються в базі даних, через інтерфейс.

```python
from django.contrib import admin
from .models import *

@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    list_display = ['user', 'phone', 'profile_pic']
    search_fields = ['user__username', 'phone']
    list_filter = ['user']

@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    list_display = ['id', 'product_name', 'category', 'brand', 'price', 'stock', 'tag', 'product_img']
    search_fields = ['product_name', 'category__name', 'brand__name']
    list_filter = ['category', 'brand']

@admin.register(Cart)
class CartAdmin(admin.ModelAdmin):
    list_display = ['user', 'product', 'qty', 'price', 'amount']
    search_fields = ['user__username', 'product__product_name']
    list_filter = ['user']

@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ['code', 'user', 'first_name', 'last_name','house_no', 'street', 'city', 'pin', 'state', 'total', 'placed_at']
    search_fields = ['code', 'user__username', 'first_name', 'last_name', 'city']
    list_filter = ['user', 'city', 'state']

@admin.register(OrderProduct)
class OrderProductAdmin(admin.ModelAdmin):
    list_display = ['order','user', 'product', 'price', 'qty', 'amount', 'status']
    search_fields = ['order__code', 'user__username', 'product__product_name']
    list_filter = ['status']

admin.site.register(Category)
admin.site.register(Review)
admin.site.register(Wishlist)

```

Рис 3.33 admin.py

На рис. 3.32 зображений файл admin.py, в цьому файлі реалізовано реєстрація моделей, налаштування адмін панелі. Також адміністратори можуть створювати, читати, оновлювати та видаляти записи в базі даних, можливий пошук записів за визначеними полями, також є можливість фільтрації.

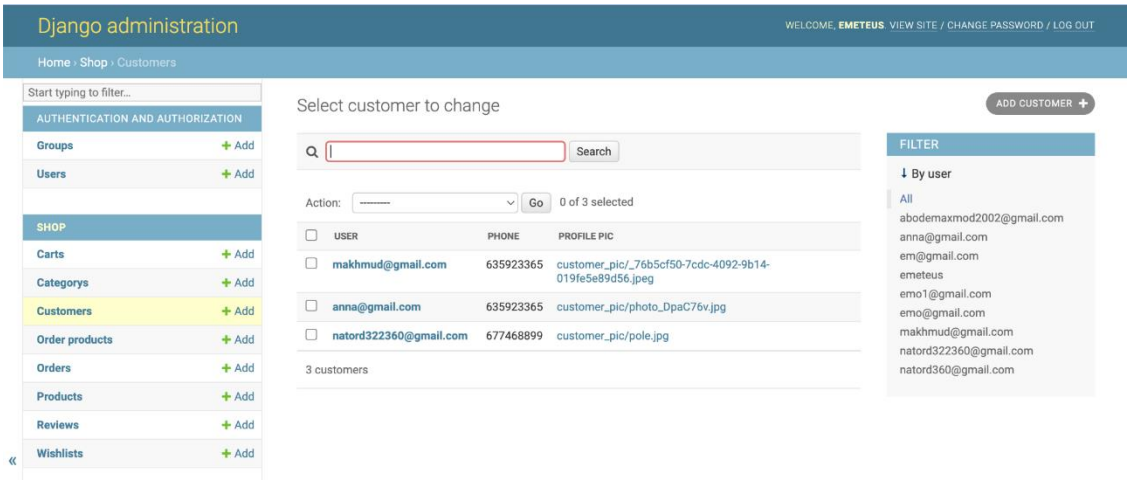


Рис 3.34 Панель адміністратора

На рис 3.34 показано зображення інтерфейсу панелі адміністратора Django. Завдяки такому інтерфейси всі операції виконуються за допомогою UI інтерфейсу, також можна побачити зареєстрованих користувачів та при потребі видалити чи налаштувати їх права. Отже панель адміністратора Django дозволяє виконувати операції з базою даних через інтерфейс.

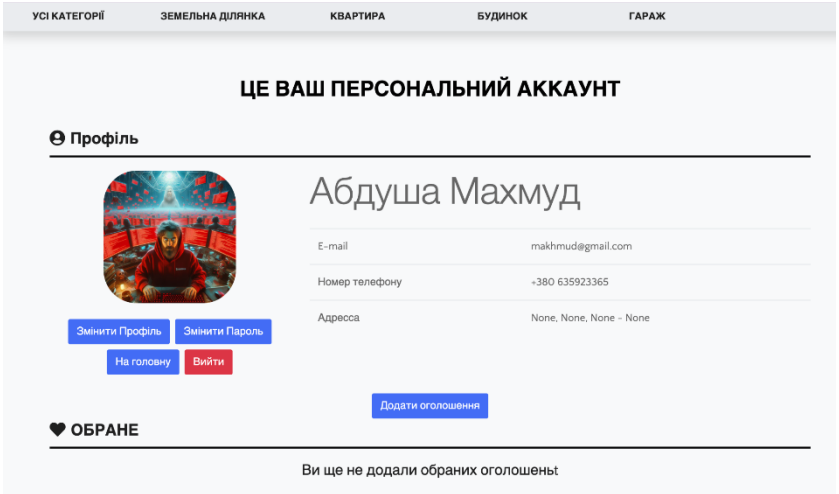


Рис 3.35 Особистий кабінет

На рис. 3.35 зображений кабінет web-застосунку BrilliantHome, особистий кабінет це ключова функція бо вона дозволяє переглядати та управляти своїми даними.

```python
from shop.models import Category, Customer, Order, OrderProduct, Cart, Wishlist
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.contrib.auth.models import User

@login_required(login_url='/login')
def account(request):
    orderprs = []
    current_user = request.user

    try:
        customer = Customer.objects.get(user=current_user)
    except Customer.DoesNotExist:
        customer = None

    carts = Cart.objects.filter(user=current_user)
    qty = sum(cart.qty for cart in carts)
    total = sum(cart.amount for cart in carts)

    categories = Category.objects.all()

    try:
        wishlists = Wishlist.objects.filter(user=current_user)
    except Wishlist.DoesNotExist:
        wishlists = None

    orders = Order.objects.filter(user=current_user).order_by("-placed_at")
    for order in orders:
        pr = OrderProduct.objects.filter(order=order)
        orderprs.append(pr)

    details = {
        'customer': customer,
        'orders': orders,
        'orderprs': orderprs,
        'qty': qty,
        'total': total,
        'carts': carts,
        'categories': categories,
        'wishlists': wishlists
    }
```

Рис 3.36 account.py

На рис. 3.35 зображений спосіб реалізації для особистого кабінету. Для початку імпортуються усі головні модулі та імпортуються декоратори.

```python
@login_required(login_url='/login')
```

Рис 3.37 Декоратор

Декоратори рис. 3.37 забезпечують, що доступ до особистого кабінету можливий лише для авторизованих користувачів. Якщо користувач не авторизований то його буде перенаправлений на сторінку входу в особистий кабінет (‘/login’)

Функція “account” забезпечує роботу інтерфейсу для користувача, де можна переглянути особисті також оновити свої дані.

```python
@login_required(login_url='/login')
def updateprofile(request):
    if request.method == "GET":
        current_user = request.user
        carts = Cart.objects.filter(user_id=current_user.id)
        qty = 0
        total = 0
        for cart in carts:
            total = total + cart.amount
            qty = qty + cart.qty
        customer = Customer.objects.get(user_id=current_user.id)
        details = {
            'customer': customer,
            'qty': qty,
            'total': total,
            'carts': carts,
        }
        return render(request, template_name: 'updateprofile.html', details)

    if request.method == "POST":
        current_user = request.user
        fname = request.POST['fname']
        lname = request.POST['lname']
        email = request.POST['email']
        phone = request.POST['phone']
        profile_pic = ''
        try:
            profile_pic = request.FILES['pic']
        except:
            pass

        if len(fname) > 10 and len(lname) > 10:
            messages.success(request, message: "First or Last Name too long!!!")
            return HttpResponseRedirect('/account/updateprofile')
        if not fname.isalpha() or not lname.isalpha():
            messages.warning(request, message: "Name must contain only letters.")
            return HttpResponseRedirect('/account/updateprofile')
        if len(str(phone)) != 10:
            messages.warning(request, message: "Phone number must contain 10 digits.")
            return HttpResponseRedirect('/account/updateprofile')

        update_user = User.objects.get(id=current_user.id)
        update_user.username = email
        update_user.first_name = fname
        update_user.last_name = lname
        update_user.email = email
```

Рис 3.38 updateprofile.py

На рис. 3.38 зобрежна функция “updateprofile” вона обробляє запити користувача на оновлення свого профілю. Якщо система отримує GET запит, то функція збирає дані про авторизованого користувача та відображає його дані про профіль. Якщо метод POST то функція перевіряє та обробляє введені дані користувачем для оновлення профілю. Якщо валідація проходить успішно, то функція оновлює дані у базі даних та відображає повідомлення про успішне оновлення даних та виводить вже оновленні дані облікового запису.

```python
def products(request):
    current_user = request.user
    customer = []
    try:
        customer = Customer.objects.get(user_id=current_user.id)
    except:
        pass
    categories = Category.objects.all()

    categoryid = 0
    filtered_category = []

    try:
        categoryid = request.GET['category']
    except:
        pass

    if categoryid:
        products = Product.objects.filter(category_id=categoryid)
        filtered_category = Category.objects.get(id=categoryid)
    else:
        products = Product.objects.all()

    n = len(products)

    current_user = request.user
    carts = Cart.objects.filter(user_id=current_user.id)
    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    params = {
        'customer':customer,
        'products': products,
        'categories': categories,
        'filtered_category': filtered_category,
        'n': n,
        'qty': qty,
        'total':total,
        'carts':carts,
        }
```

Рис 3.39 product.py

На рис. 3.39 зображена функція “products”, вона обробляє запити для відображення сторінки з переліком оголошень про нерухомість. Ця функція завантажує всі категорії оголошень та перевіряє чи є у них ідентифікатор, ця функція має змогу відфільтровувати оголошення по категоріям через параметри запиту.

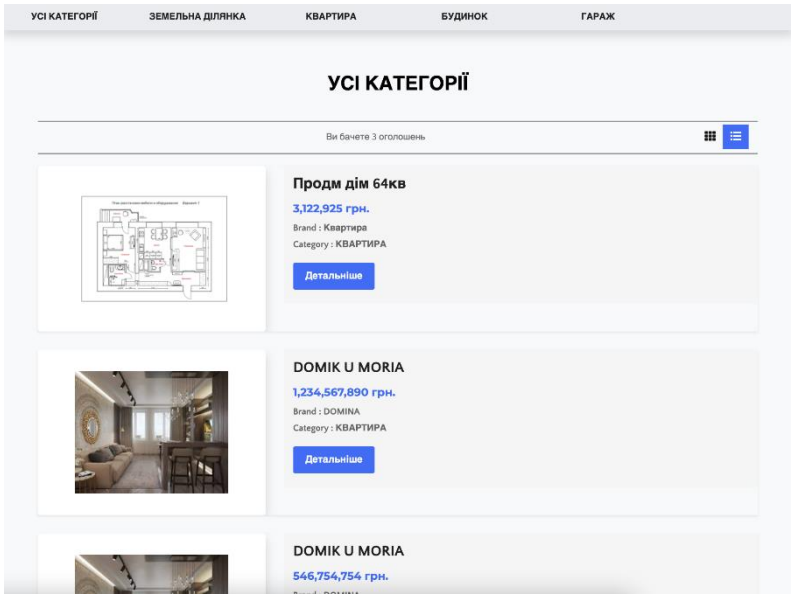


Рис 3.40 Сторінка оголошень

На малюнку 3.40 зображено інтерфейс де виводяться усі оголошення та є вибір по категоріям .

```python
@login_required(login_url='/login')
def addtowishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    wishlist = Wishlist(user_id=current_user.id, product_id=prid)
    wishlist.save()
    messages.success(request, message: f"{wishlist.product.product_name} added to your Wishlist.")
    return HttpResponseRedirect(url)

1 usage
@login_required(login_url='/login')
def removefromwishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    product = get_object_or_404(Product, id=prid)
    try:
        wishlist_item = Wishlist.objects.get(user_id=current_user.id, product_id=prid)
        wishlist_item.delete()
        messages.success(request, message: f"{product.product_name} removed from your Wishlist.")
    except Wishlist.DoesNotExist:
        messages.error(request, message: "Wishlist item does not exist.")
    return HttpResponseRedirect(url)
```

Рис 3.41 wishlist.py

На рис. 3.41 зображено фрагмент коду який представляє дві функції web-застосунку, які взаємодіють зі списком бажань. Перша функція “addwishlist” він додає обране оголошення до списку бажань. Після цього створюється новий запис у моделі “Wishlist”який пов'язує поточного користувача з вказаним продуктом та відправляється повідомлення про успішне додавання продукту до списку бажань.

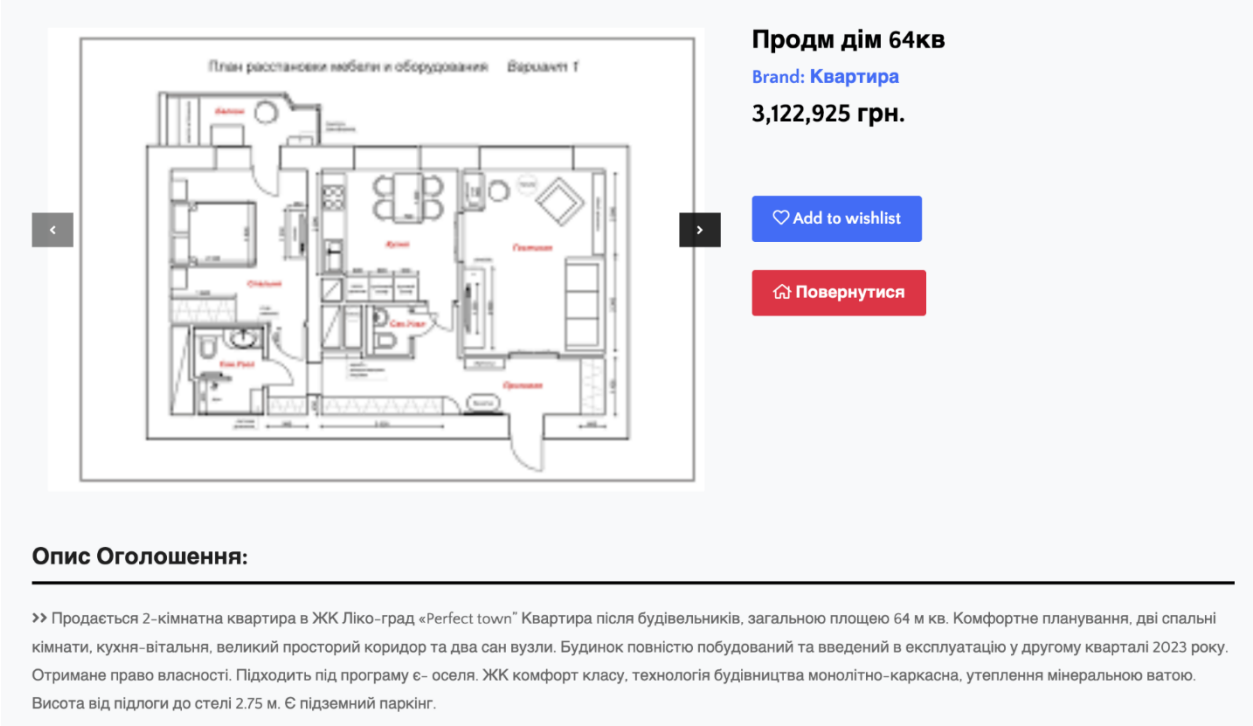
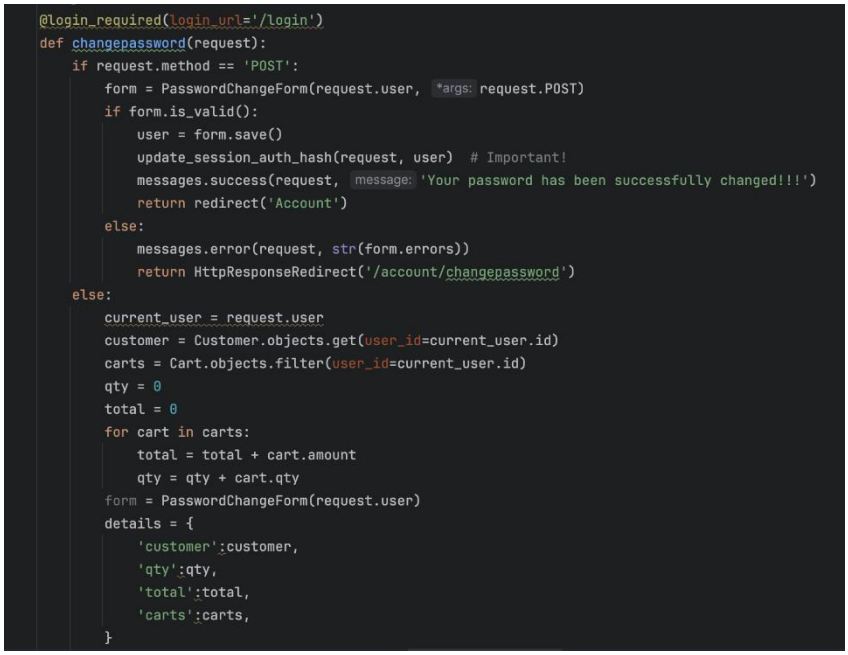


Рис 3.42 Перегляд оголошення

Друга функція “removefromwishlist” ця функція видаляє продукт зі списку бажань користувача. Вона теж перевіряє, чи користувач увійшов у систему, та знаходить продукт за його ідентифікатором, далі йде віддалення відповідної моделі. У випадку, якщо запис у списку бажань не знайдено, то користувачу виводить помилку. Обидві функції вимагають авторизації



```
@login_required(login_url='/login')
def changepassword(request):
    if request.method == 'POST':
        form = PasswordChangeForm(request.user, *args: request.POST)
        if form.is_valid():
            user = form.save()
            update_session_auth_hash(request, user)  # Important!
            messages.success(request, message: 'Your password has been successfully changed!!!')
            return redirect('Account')
        else:
            messages.error(request, str(form.errors))
            return HttpResponseRedirect('/account/changepassword')
    else:
        current_user = request.user
        customer = Customer.objects.get(user_id=current_user.id)
        carts = Cart.objects.filter(user_id=current_user.id)
        qty = 0
        total = 0
        for cart in carts:
            total = total + cart.amount
            qty = qty + cart.qty
        form = PasswordChangeForm(request.user)
        details = {
            'customer':customer,
            'qty':qty,
            'total':total,
            'carts':carts,
        }
```

Рис 3.43 Реалізація зміни пароля

На рис. 3.43 зображено реалізація функції “changepassword”, тут код обробляє зміну паролю користувача в web-застосунку. В разі отримання POST запиту зі зміною пароля, воно створю є форму, ящо форма є дійсною, пароль оновлюється та сесія зберігається. Якщо форма містить помилки , то користувач отримує відповідне повідомлення з помилкою та перенаправляє його назад на сторінку зі зміною пароля. Ця функція обов'язково вимагає авторизованого користувача.

4 ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ

4.1 Тестування роботи web-застосунку

Щоб забезпечити якість та надійність web-застосунку BrilliantHome який було розроблено за допомогою фреймворку Angular, було проведено повне тестування з використанням фреймворку Jasmine для написання юніт тестів, та використання інструменту Karma для їх виконання.

Юніт тести - це автоматизовані тести для перевірки роботи окремих частин коду, наприклад таких як функцій або методів класів [4, 13]. Метою юніт тестів є ізоляція кожного компонента програми для того щоб підтвердити його правильну роботу незалежно від інших частин коду.

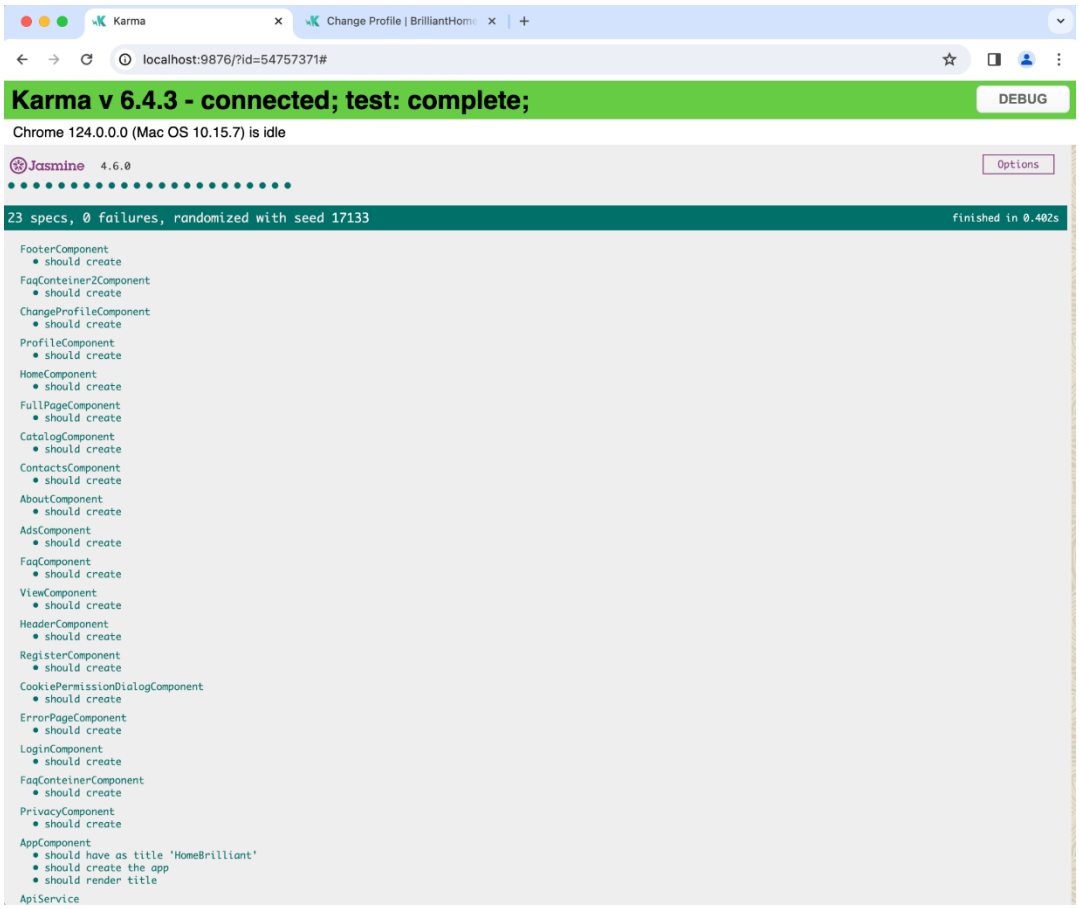


Рис 4.1 Karma (тестування)

На рис. 4.1 представлений результат тестування, він продемонстрував успішне проходження всіх юніт тестів без жодної помилки. Загалом було проведено 23 тестових специфікації для перевірки різних компонентів web-застосунку. Кожна специфікація перевірила створення кожного компонента, також коректність функціонування.

Результати тестування показали, що всі компоненти утворилися коректно і не мають помилок. Тесті були завершені за (0.402) секунди. Проведене тестування показало що web-застосунок працює коректно та всі компоненти проекту Angular теж працюють коректно. Ці тести забезпечують впевненість та надійність у роботі web-застосунку.

ВИСНОВКИ

У результаті виконання дипломної роботу було розроблено застосунок для оренди та продажу нерухомості за допомогою фреймворків Angular та Django. Актуальність розробки web-застосунку полягає в необхідності цифрових рішень для динамічного ринку нерухомості. Ці застосунки автоматизують процеси пошуку нерухомості, та дають доступність до ринку у будь-який час та з будь-якого місця.

Проведено аналіз предметної галузі, згідно теми роботи. Було проаналізовано три існуючих web-застосунки таких як: LUN.ua, OLX, UkrXata, при аналізі було виявлено їх основні переваги та недоліки.

На основі аналізу предметної галузі та порівняння існуючих аналогів було визначено технічні завдання та були сформовані функціональні та нефункціональні вимоги до web-застосунку.

Для розробки web-застосунку було проаналізовано засоби розробки та було обрано для клієнтської частини фреймворк Angular, для серверної частини був обраний фреймворк Django, була обрана база даних MongoDB, також були обранні середовища розробки WebStorm та PyCharm.

Розроблено web-застосунок для оренди та продажу нерухомості з використанням мовои Python з застосуванням фреймворків Django та Angular

Проведено тестування web-застосунку за допомогою фреймворку Jasmine та інструменту Karma.

Робота пройшла апробацію на Всеукраїнській науково-технічній конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» за темами «Розробка web-застосунку для управління нерухомістю агентства brillianthome мовою python з застосуванням фреймворків django та angular» та «Розробка web-платформи з використанням технологій html, css, angular, django та mongodb»

ПЕРЕЛІК ПОСИЛАНЬ

1. Vincent W. S. Django for Beginners: Build websites with Python and Django. Independently published, 2018. 323 с.(дата звернення 10.04.2024).
2. Murray N., Coury F., Lerner A., Taborda C. ng-book: The Complete Guide to Angular. Fullstack.io, 2018. 681 с.(дата звернення 12.04.2024).
3. Прищенко С. В. HTML і CSS: Дизайн та створення веб-сайтів. Видавництво Кондор, 2011. 512 с. (дата звернення 20.04.2024).
4. Хартман Н., Тарасевич П. Повномасштабна веб-розробка з Django і Angular: Розробка повнофункціональних, реальних веб-застосунків за допомогою Django і Angular. Видавництво Апрес, 2018. 280 с. (дата звернення 24.04.2024).
5. Беркович М., Коваленко О., Мельник І., Соколовський В. SEO: Повний путівник по оптимізації для пошукових систем. Fullstack.io, 2023. 754 с. (дата звернення 12.05.2024)
6. Сайт оголошень LUN.ua. URL: https://lun.ua/ (дата звернення 28.04.2024).
7. Сайт оголошень OLX. URL: https://www.olx.ua/uk/ (дата звернення 02.05.2024).
8. Сайт оголошень UkrXata. URL: https://ukrxata.info/ (дата звернення 05.05.2024).
9. Official Angular Documentation. URL: https://angular.io/docs (дата звернення 20.05.2024).
10. Official Django Documentation. URL: https://www.djangoproject.com/start/ (дата звернення 06.05.2024).
11. Official WebStorm Documentation. URL: https://www.jetbrains.com/help/webstorm/ (дата звернення 09.05.2024).
12. Official PyCharm Documentation. URL: https://www.jetbrains.com/help/pycharm/ (дата звернення 15.05.2024).
13. Official Jasmine(Karma) Documentation. URL: https://jasmine.github.io/ (дата звернення 17.05.2024).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ НЕРУХОМІСТЮ АГЕНТСТВА «BRILLIANTHOME» МОВОЮ PYTHON З ЗАСТОСУВАННЯМ ФРЕЙМВОРКІВ DJANGO ТА ANGULAR

Виконав студент 4 курсу
групи ПД-42
Махмуд Абдуль-Фаттах Мазенович
Керівник роботи
к.т.н., доцент кафедри ІПЗ Довженко Тимур Павлович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підтримка процесу продажу та оренди нерухомості, створений за допомогою фреймворків (Angular та Django).

Об'єкт дослідження - процес продажів та оренди нерухомості.

Предмет дослідження - web-застосунок для продажі та оренди нерухомості.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз предметної галузі та визначити основні переваги та недоліки існуючих web-застосунків.

2. Обрати засоби розробки для створення web-застосунку, проаналізувати фреймворки та інструменти

3. Розробити web-застосунок для оренди та продажу нерухомості

4. Провести тестування розробленого web-застосунку за допомогою фреймворку Jasmine

3

АНАЛІЗ АНАЛОГІВ

Функції	LUN.UA	OLX	UkrXata	BrilliantHome
Наявність англійської мови у мультимовності	Ні	Ні	Ні	Так
Можливість перемикання між темною або світлою темою	Немає	Немає	Немає	Присутня
Безкоштовне виставлення оголошення	Немає	1 оголошення	Безліч	Безліч
Можливість реєстрації за допомогою соціальних мереж	Присутня	Присутня	Відсутня	Присутня
Швидкість загрузки сайту	2040 ms	2650 ms	1028 ms	827 ms

4

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні вимоги:

1. Створення облікового запису із збереженням особистої інформації та здатністю увійти до системи
2. Користувачі можуть додавати оголошення про продаж та оренду нерухомості.
3. Користувачі можуть сортувати нерухомість за критеріями такими як ціна.
4. Можливість вибору мови інтерфейсу (українська та англійська) та перемикання між світлою і темною темою інтерфейсу.

Нефункціональні вимоги:

1. Забезпечення конфіденційності персональних даних користувачів з застосуванням відповідних методів шифрування.
2. Швидке завантаження сторінок та швидкі відгуки на запити користувачів. (827 ms)
3. Забезпечення коректної роботи в різних браузерах. (Google Chrome, Opera, Safari, Edge)
4. Адаптивний дизайн з (375px на 667px) до (2560px на 10180px)

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Angular

Python(Django)

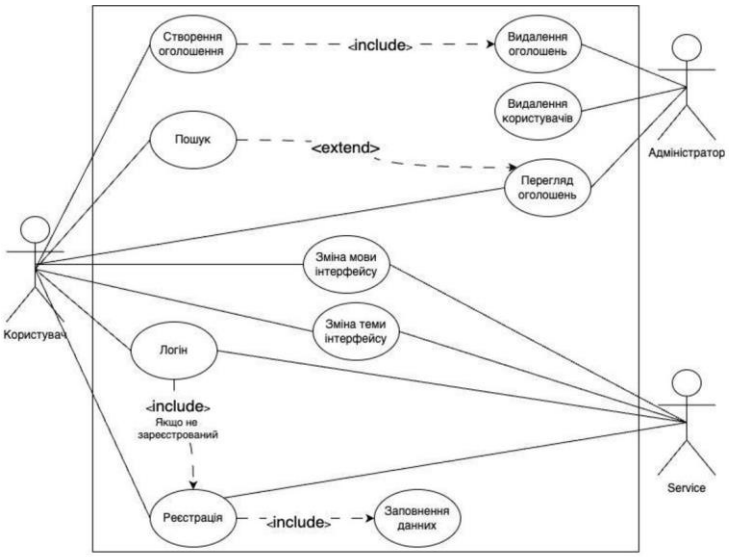
MongoDB

WebStorm

PyCharm

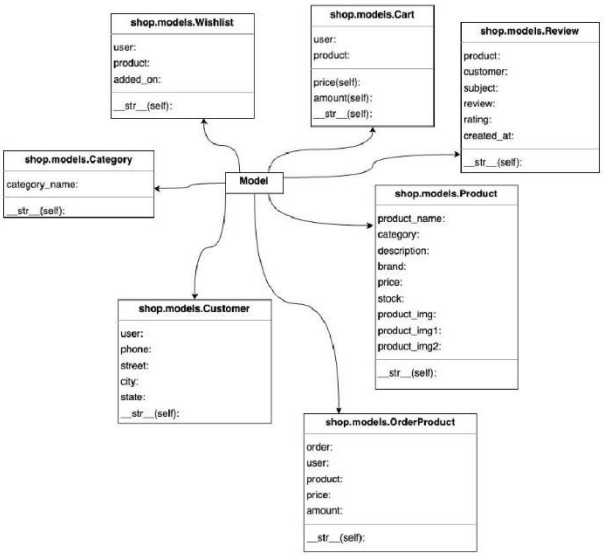
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



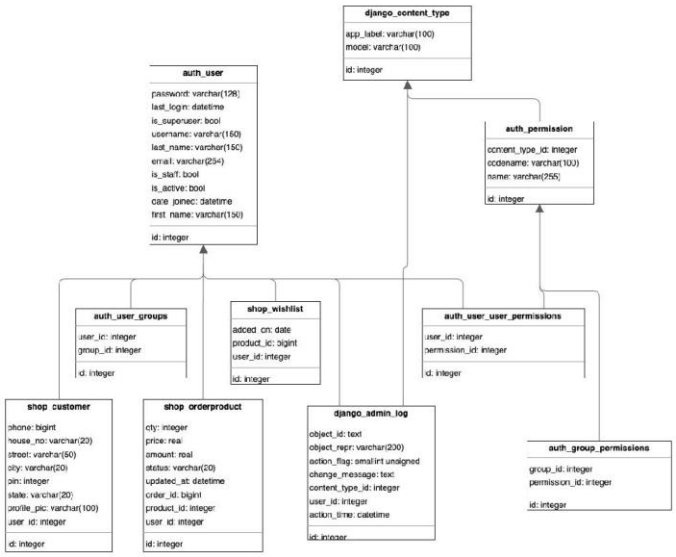
7

ДІАГРАМА КЛАСІВ



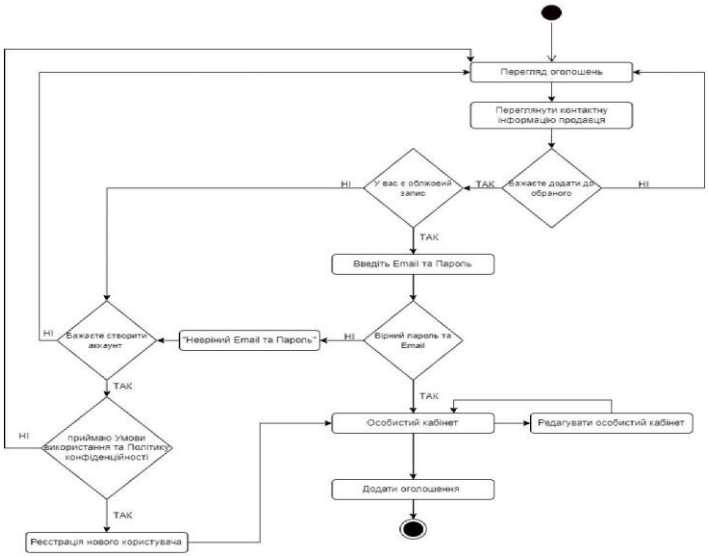
8

ДІАГРАМА БАЗИ ДАНИХ



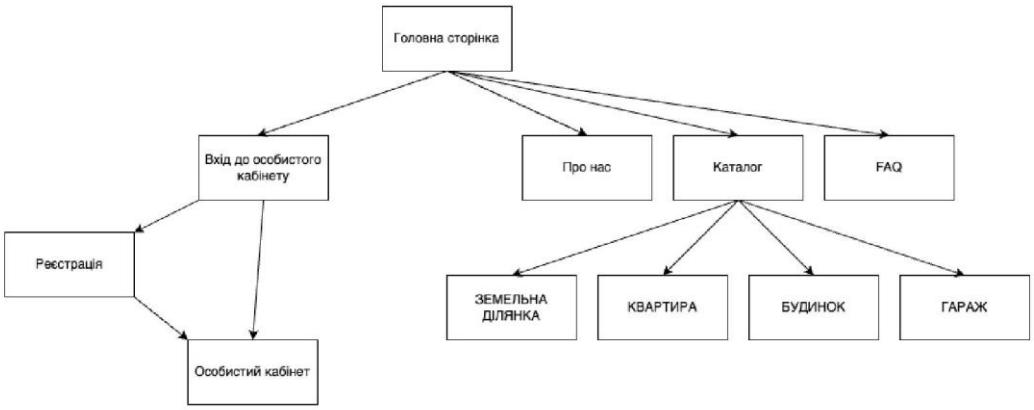
9

ДІАГРАМА АКТИВНОСТІ ПЕРЕГЛЯДУ ОГОЛОШЕНЬ



10

ДІАГРАМА КАРТИ САЙТУ



11

ЕКРАННІ ФОРМИ

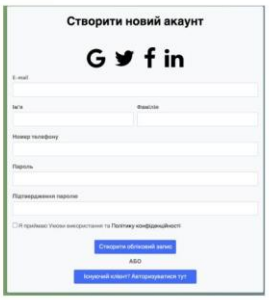
Головна сторінка (світла тема)

Головна сторінка (темна тема)

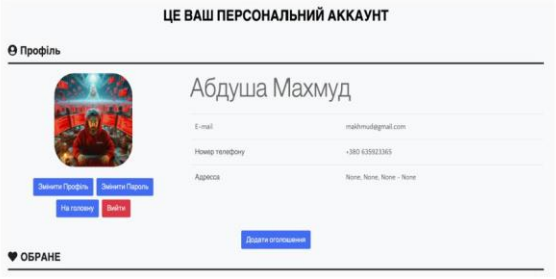
12

ЕКРАННІ ФОРМИ

Вхід в особистий кабінет



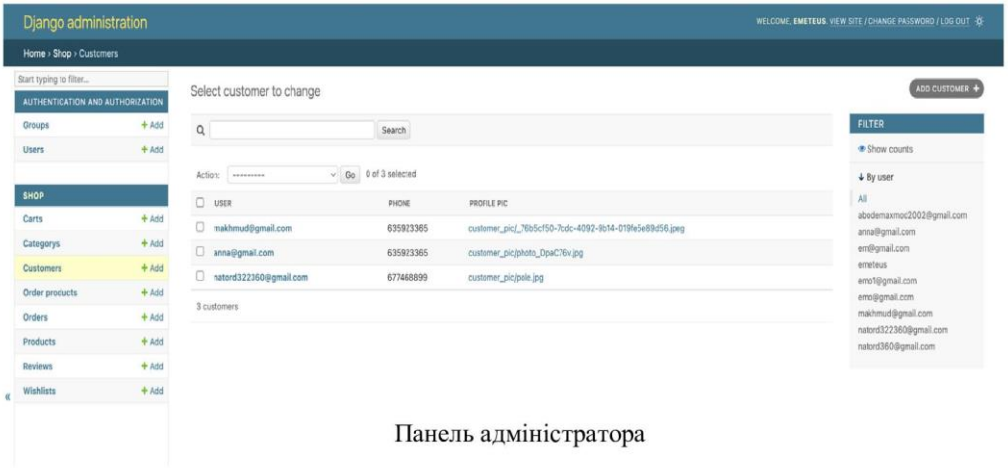
Реєстрація



Особистий кабінет

13

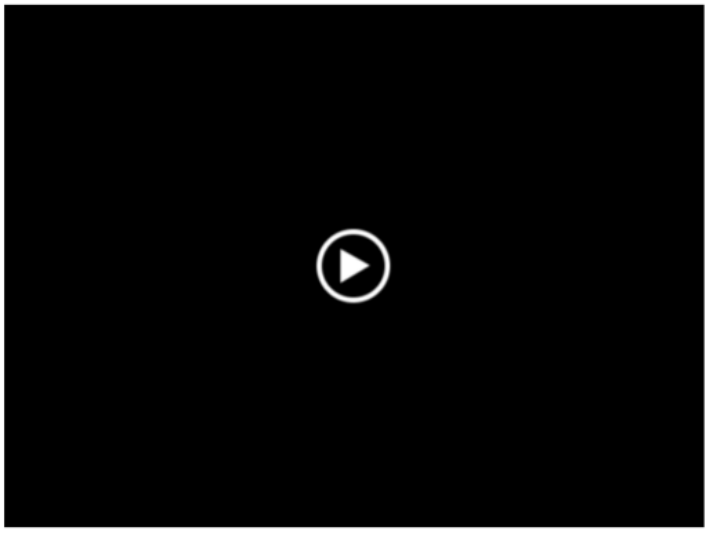
ЕКРАННІ ФОРМИ



Панель адміністратора

14

ДЕМОНСТРАЦІЯ РОБОТИ WEB-ЗАСТОСУНКУ



15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Махмуд А-Ф. М. Розробка web-застосунку для управління нерухомістю агентства brillianthome мовою python з використанням фреймворків django та angular / Довженко Т. П., Махмуд А-Ф. М. // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 20 квітня 2024 року, Київ, Державний університет інформаційно-комунікаційно технологій. Збірник тез. К.: ДУІКТ, 2024, С.241-242.

2. Махмуд А-Ф. М. Розробка web-додатку з використанням технологій html, css, angular, django та mongodb/ Довженко Т. П., Махмуд А-Ф. М. // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 20 квітня 2024 року, Київ, Державний університет інформаційно-комунікаційно технологій. Збірник тез. К.: ДУІКТ, 2024, С.160-161.

16

ВИСНОВКИ

1. Проведено аналіз предметної галузі, згідно теми роботи. Було проаналізовано три існуючих web-застосунки таких як: LUN.ua, OLX, UkrXata, при аналізі було виявлено їх основні переваги та недоліки.

2. Для розробки web-застосунку було проаналізовано засоби розробки та було обрано для клієнтської частини фреймворк Angular, для серверної частини був обраний фреймворк Django, була обрана база даних MongoDB, також були обранні середовища розробки WebStorm та PyCharm.

3. Розроблено web-застосунок для оренди та продажу нерухомості з використанням мовои Python з застосуванням фреймворків Django та Angular

4. Проведено тестування web-застосунку за допомогою фреймворку Jasmine та інструменту Karma.

ДОДАТОК Б ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

```
import {Component, OnInit}
from '@angular/core';

@Component({
 selector: 'app-about',
 templateUrl:
'./about.component.html',
 styleUrl:
'./about.component.css'
})
export class AboutComponent
implements OnInit {

 constructor() { }

 ngOnInit(): void {
   document.title = 'About |
BrilliantHome';
   if (typeof document !==
'undefined') {
     if (document.readyState
=== 'complete') {
     this.initSlider();
     } else {

document.addEventListener('DOM
ContentLoaded',
this.initSlider);
     }
   }
 }

 initSlider(): void {
   const slider: SliderElement
| null =
document.querySelector('.slide
r');
   if (!slider) {
     console.error("Слайдер не
найден.");
     return;
   }
   const slides:
NodeListOf<HTMLElement> =
document.querySelectorAll('.sl
ide');
   const prevBtn: HTMLElement
| null =
document.querySelector('.prev'
);
   const nextBtn: HTMLElement
| null =
document.querySelector('.next'
);

   if (!prevBtn || !nextBtn) {
     console.error(".");
     return;
   }

   let currentIndex: number =
0;

   function showSlide(index:
number): void {
     if (index < 0) {
     currentIndex =
slides.length - 1;
     } else if (index >=
slides.length) {
     currentIndex = 0;
     } else {
     currentIndex = index;
     }

     const translateValue:
string = `-${currentIndex *
100}%`;
     if (slider) {
     slider.style.transform =
`translateX(${translateValue})
`;
     }
   }

   function nextSlide(): void
{
     showSlide(currentIndex +
1);
   }

   function prevSlide(): void
{
     showSlide(currentIndex -
1);
   }

   if (nextBtn)
nextBtn.addEventListener('clic
k', nextSlide);
```

```
   if (prevBtn)
prevBtn.addEventListener('clic
k', prevSlide);

   setInterval(nextSlide,
5000);
 }
}

interface SliderElement
extends Element {
 style: {
   transform: string;
 };
}

import { Component } from
'@angular/core';
import { MatDialogRef } from
'@angular/material/dialog';

@Component({
 selector:
'app-cookie-permission-dialog'
,
 templateUrl:
'./cookie-permission-dialog.co
mponent.html',
 styleUrls:
['./cookie-permission-dialog.c
omponent.css']
})
export class
CookiePermissionDialogComponen
t {
 constructor(public dialogRef:
MatDialogRef<CookiePermissionD
ialogComponent>) {}

 allowCookies(): void {
   this.dialogRef.close(true);
 }

 denyCookies(): void {

this.dialogRef.close(false);
 }
}
import {Component, ElementRef,
OnInit} from '@angular/core';

@Component({
 selector: 'app-faq',
 templateUrl:
'./faq.component.html',
 styleUrl:
'./faq.component.css'
})
export class FaqComponent
implements OnInit {

 constructor(private
elementRef: ElementRef) { }

 ngOnInit(): void {

     document.title = 'FAQ |
BrilliantHome';



   const items =
this.elementRef.nativeElement.
querySelectorAll(".accordion
button");

   function
toggleAccordion(this:
HTMLElement) {
     const itemToggle =
this.getAttribute('aria-expand
ed');

     for (let i = 0; i <
items.length; i++) {
     (items[i] as
HTMLElement).setAttribute('ari
a-expanded', 'false');
     }

     if (itemToggle ===
'false') {

this.setAttribute('aria-expand
ed', 'true');
     }
   }

   items.forEach((item:
HTMLElement) =>
item.addEventListener('click',
toggleAccordion.bind(item)));
 }
}

import { Component, OnInit }
from '@angular/core';
import { MatDialog } from
'@angular/material/dialog';
import { TranslateService }
from '@ngx-translate/core';
```

```
import { CookieService } from
'ngx-cookie-service';
import {
CookiePermissionDialogComponen
t } from
'./pages/cookie-permission-dia
log/cookie-permission-dialog.c
omponent';

@Component({
 selector: 'app-root',
 templateUrl:
'./app.component.html',
 styleUrls:
['./app.component.css']
})
export class AppComponent
implements OnInit {
 title = 'HomeBrilliant';

 constructor(
      private translateService:
TranslateService,
      private cookieService:
CookieService,
      private dialog: MatDialog
 ) {}

 ngOnInit() {
   let savedLanguage =
this.cookieService.get('select
edLanguage');
   const DEFAULT_LANGUAGE =
'en';

   if (!savedLanguage) {
     savedLanguage =
navigator.language ||
DEFAULT_LANGUAGE;

this.cookieService.set('select
edLanguage', savedLanguage,
365);
   }


this.translateService.setDefau
ltLang(DEFAULT_LANGUAGE);


this.translateService.setDefau
ltLang(DEFAULT_LANGUAGE);

this.translateService.use(save
dLanguage).subscribe({
      next: () => {
      console.log(`Translation
file for ${savedLanguage}
found and loaded
successfully.`);
      },
      error: () => {
      console.warn(`Translation
file for ${savedLanguage} not
found. Falling back to
${DEFAULT_LANGUAGE}.`);

this.translateService.use(DEFA
ULT_LANGUAGE);
      }
   });



   if (typeof document !==
'undefined') {

document.addEventListener('DOM
ContentLoaded', () => {
      const savedTheme =
this.cookieService.get('theme'
);
      const body: HTMLElement =
document.body;
      const main: HTMLElement |
null =
document.querySelector('main')
;
      const moonIcon:
HTMLElement | null =
document.querySelector('.moon-
icon');
      const sunIcon:
HTMLElement | null =
document.querySelector('.sun-i
con');
      const footer: HTMLElement
| null =
document.querySelector('footer
');
      const toggleButton:
HTMLElement | null =
document.getElementById('toggl
eButton');

      if (savedTheme) {
      this.setTheme(savedTheme,
body, main, moonIcon, sunIcon,
footer);
      if (savedTheme ===
'dark') {
```

```
this.showMoonIcon(moonIcon,
sunIcon);
      } else {

this.showSunIcon(moonIcon,
sunIcon);
      }
      } else {
      this.setTheme('light',
body, main, moonIcon, sunIcon,
footer);

this.showSunIcon(moonIcon,
sunIcon);
      }

      if (toggleButton) {

toggleButton.addEventListener(
'click', () => {

body.classList.toggle('dark');
            if (main) {

main.classList.toggle('dark');

main.classList.toggle('theme-d
ark');
            }
            const isDarkTheme:
boolean =
body.classList.contains('dark'
);
            if (isDarkTheme) {

this.setTheme('dark', body,
main, moonIcon, sunIcon,
footer);

this.showMoonIcon(moonIcon,
sunIcon);
            } else {

this.setTheme('light', body,
main, moonIcon, sunIcon,
footer);

this.showSunIcon(moonIcon,
sunIcon);
            }
      });
      }
      });
   }
```

```
   if
(!this.cookieService.get('cook
iePermission')) {

this.showCookiePermissionDialo
g();
   }
 }

 switchLanguage(language:
string) {

this.translateService.use(lang
uage).subscribe(() => {

this.cookieService.set('select
edLanguage', language, 365);
   });
 }

 private setTheme(theme:
string, body: HTMLElement,
main: HTMLElement | null,
moonIcon: HTMLElement | null,
sunIcon: HTMLElement | null,
footer: HTMLElement | null):
void {
   if (theme === 'dark') {
     this.applyDarkTheme(body,
main, moonIcon, sunIcon,
footer);
   } else {

this.applyLightTheme(body,
main, moonIcon, sunIcon,
footer);
   }

this.cookieService.set('theme'
, theme, 365);
 }

 private applyDarkTheme(body:
HTMLElement, main: HTMLElement
| null, moonIcon: HTMLElement
| null, sunIcon: HTMLElement |
null, footer: HTMLElement |
null): void {
   if (moonIcon) {
     moonIcon.style.display =
'none';
   }
   if (sunIcon) {
     sunIcon.style.display =
'block';
   }
```

```
    body.classList.add('dark');

body.classList.add('theme-dark
');
    if (main) {

main.classList.add('dark');

main.classList.add('theme-dark
');
    }
    if (footer) {
      footer.style.background =
'#111';
      footer.style.color =
'#fff';
    }
  }

  private applyLightTheme(body:
HTMLElement, main: HTMLElement
| null, moonIcon: HTMLElement
| null, sunIcon: HTMLElement |
null, footer: HTMLElement |
null): void {
    if (moonIcon) {
      moonIcon.style.display =
'block';
    }
    if (sunIcon) {
      sunIcon.style.display =
'none';
    }

body.classList.remove('dark');

body.classList.remove('theme-d
ark');
    if (main) {

main.classList.remove('dark');

main.classList.remove('theme-d
ark');
    }
    if (footer) {
      footer.style.background =
'';
      footer.style.color = '';
    }
  }

  private
showMoonIcon(moonIcon:
HTMLElement | null, sunIcon:
HTMLElement | null): void {
    if (moonIcon) {
      moonIcon.style.display =
'block';
    }
    if (sunIcon) {
      sunIcon.style.display =
'none';
    }
  }

  private showSunIcon(moonIcon:
HTMLElement | null, sunIcon:
HTMLElement | null): void {
    if (moonIcon) {
      moonIcon.style.display =
'none';
    }
    if (sunIcon) {
      sunIcon.style.display =
'block';
    }
  }

  private
showCookiePermissionDialog():
void {
    const dialogRef =
this.dialog.open(CookiePermiss
ionDialogComponent);

dialogRef.afterClosed().subscr
ibe(allow => {
      if (allow) {

this.cookieService.set('cookie
Permission', 'true', 365);

      let savedLanguage =
this.cookieService.get('select
edLanguage');
      const DEFAULT_LANGUAGE =
'en';
      if (!savedLanguage) {
      savedLanguage =
navigator.language ||
DEFAULT_LANGUAGE;

this.cookieService.set('select
edLanguage', savedLanguage,
365);
      }

this.switchLanguage(savedLangu
age);
      }
    });
```

```
 }
}
import { NgModule } from
'@angular/core';
import { BrowserModule } from
'@angular/platform-browser';
import { HttpClientModule,
HttpClient } from
'@angular/common/http';
import { FormsModule } from
'@angular/forms';
import { MatButtonModule }
from
'@angular/material/button';
import { TranslateModule,
TranslateLoader,
TranslateService } from
'@ngx-translate/core';
import { TranslateHttpLoader }
from
'@ngx-translate/http-loader';
import { NgOptimizedImage }
from '@angular/common';
import {
BrowserAnimationsModule } from
'@angular/platform-browser/ani
mations';
import { AppRoutingModule }
from './app-routing.module';
import { AppComponent } from
'./app.component';

import { AboutComponent } from
'./pages/about/about.component
';
import { CatalogComponent }
from
'./pages/catalog/catalog.compo
nent';
import { ContactsComponent }
from
'./pages/contacts/contacts.com
ponent';
import { ErrorPageComponent }
from
'./pages/error-page/error-page
.component';
import { FaqComponent } from
'./pages/faq/faq.component';
import { FullPageComponent }
from
'./pages/full-page/full-page.c
omponent';
import { HomeComponent } from
'./pages/home/home.component';
import { PrivacyComponent }
from
'./pages/privacy/privacy.compo
nent';
import { ViewComponent } from
'./pages/view/view.component';
import {
ChangeProfileComponent } from
'./pages/profile/change-profil
e/change-profile.component';
import { LoginComponent } from
'./pages/auth-pages/login/logi
n.component';
import { RegisterComponent }
from
'./pages/auth-pages/register/r
egister.component';
import { FaqConteinerComponent
} from
'./shared/faq-conteiner/faq-co
nteiner.component';
import {
FaqConteiner2Component } from
'./shared/faq-conteiner2/faq-c
onteiner2.component';
import { FooterComponent }
from
'./shared/footer/footer.compon
ent';
import { HeaderComponent }
from
'./shared/header/header.compon
ent';
import { AdsComponent } from
'./pages/ads/ads.component';
import {
CookiePermissionDialogComponen
t } from
'./pages/cookie-permission-dia
log/cookie-permission-dialog.c
omponent';

export function
HttpLoaderFactory(http:
HttpClient) {
 return new
TranslateHttpLoader(http,
'./assets/i18n/', '.json');
}

@NgModule({
 declarations: [
   AppComponent,
   AboutComponent,
   CatalogComponent,
   ContactsComponent,
```

```
    ErrorPageComponent,
    FaqComponent,
    FullPageComponent,
    HomeComponent,
    PrivacyComponent,
    ViewComponent,
    ChangeProfileComponent,
    LoginComponent,
    RegisterComponent,
    FaqConteinerComponent,
    FaqConteiner2Component,
    FooterComponent,
    HeaderComponent,
    AdsComponent,

CookiePermissionDialogComponen
t,
    FaqConteinerComponent
 ],
 imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    MatButtonModule,
    NgOptimizedImage,
    BrowserAnimationsModule,
    TranslateModule.forRoot({
      loader: {
      provide: TranslateLoader,
      useFactory:
HttpLoaderFactory,
      deps: [HttpClient]
      }
    })
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {
 constructor(private
translateService:
TranslateService) {

this.translateService.addLangs
(['en', 'ua']);

this.translateService.setDefau
ltLang('ua');
 }
}
import { NgModule } from
'@angular/core';
import { RouterModule, Routes
} from '@angular/router';
import {HomeComponent} from
"./pages/home/home.component";
import {FullPageComponent}
from
"./pages/full-page/full-page.c
omponent";
import {AboutComponent} from
"./pages/about/about.component
";
import {ContactsComponent}
from
"./pages/contacts/contacts.com
ponent";
import {PrivacyComponent} from
"./pages/privacy/privacy.compo
nent";
import {FaqComponent} from
"./pages/faq/faq.component";
import {CatalogComponent} from
"./pages/catalog/catalog.compo
nent";
import
{ChangeProfileComponent} from
"./pages/profile/change-profil
e/change-profile.component";
import {ProfileComponent} from
"./pages/profile/profile/profi
le.component";
import {ViewComponent} from
"./pages/view/view.component";
import {LoginComponent} from
"./pages/auth-pages/login/logi
n.component";
import {RegisterComponent}
from
"./pages/auth-pages/register/r
egister.component";
import {ErrorPageComponent}
from
"./pages/error-page/error-page
.component";

const routes: Routes = [
 {path: '',
 component: FullPageComponent,
 children:[
    {path: '', component:
HomeComponent },
    {path: 'about', component:
AboutComponent },
    {path: 'contacts',
component: ContactsComponent
},
    {path: 'privacy',
component:  PrivacyComponent
},
```

```
   {path: 'faq', component:
FaqComponent },
   {path: 'catalog',
component: CatalogComponent },
   {path: 'change', component:
ChangeProfileComponent },
   {path: 'profile',
component: ProfileComponent },
   {path: 'view', component:
ViewComponent },
 ]},
 {
   path: 'login',
   component: LoginComponent
 },
 {
   path: 'register',
   component:
RegisterComponent
 },
 {
   path: '**',
   component:
ErrorPageComponent
 }
];

@NgModule({
 imports:
[RouterModule.forRoot(routes)]
,
 exports: [RouterModule]
})
export class AppRoutingModule
{ }


import {Component, OnInit}
from '@angular/core';

@Component({
 selector: 'app-full-page',
 templateUrl:
'./full-page.component.html',
 styleUrl:
'./full-page.component.css'
})
export class FullPageComponent
implements OnInit{
 constructor() { }

 ngOnInit(): void {
   if (typeof document !==
'undefined') {
document.addEventListener("DOM
ContentLoaded", () => {
     const languageButton =
document.getElementById("langu
ageButton") as HTMLElement;
     const languageList =
document.querySelector(".langu
age-list") as HTMLElement;


languageButton.addEventListene
r("click", (event) => {
     event.stopPropagation();

languageList.classList.toggle(
"active");
     });


document.addEventListener("cli
ck", () => {

languageList.classList.remove(
"active");
     });


languageList.addEventListener(
"click", (event) => {
     event.stopPropagation();
     });
     });
   }
 }
}
import { Component, OnInit }
from '@angular/core';
import { MatDialog } from
'@angular/material/dialog';
import { TranslateService }
from '@ngx-translate/core';
import { CookieService } from
'ngx-cookie-service';
import {
CookiePermissionDialogComponen
t } from
'./pages/cookie-permission-dia
log/cookie-permission-dialog.c
omponent';

@Component({
 selector: 'app-root',
 templateUrl:
'./app.component.html',
```

```
 styleUrls:
['./app.component.css']
})
export class AppComponent
implements OnInit {
 title = 'HomeBrilliant';

 constructor(
     private translateService:
TranslateService,
     private cookieService:
CookieService,
     private dialog: MatDialog
 ) {}

 ngOnInit() {
   let savedLanguage =
this.cookieService.get('select
edLanguage');
   const DEFAULT_LANGUAGE =
'en';

   if (!savedLanguage) {
     savedLanguage =
navigator.language ||
DEFAULT_LANGUAGE;

this.cookieService.set('select
edLanguage', savedLanguage,
365);
   }


this.translateService.setDefau
ltLang(DEFAULT_LANGUAGE);


this.translateService.setDefau
ltLang(DEFAULT_LANGUAGE);

this.translateService.use(save
dLanguage).subscribe({
     next: () => {
     console.log(`Translation
file for ${savedLanguage}
found and loaded
successfully.`);
     },
     error: () => {
     console.warn(`Translation
file for ${savedLanguage} not
found. Falling back to
${DEFAULT_LANGUAGE}.`);

this.translateService.use(DEFA
ULT_LANGUAGE);
     }
   });



   if (typeof document !==
'undefined') {

document.addEventListener('DOM
ContentLoaded', () => {
     const savedTheme =
this.cookieService.get('theme'
);
     const body: HTMLElement =
document.body;
     const main: HTMLElement |
null =
document.querySelector('main')
;
     const moonIcon:
HTMLElement | null =
document.querySelector('.moon-
icon');
     const sunIcon:
HTMLElement | null =
document.querySelector('.sun-i
con');
     const footer: HTMLElement
| null =
document.querySelector('footer
');
     const toggleButton:
HTMLElement | null =
document.getElementById('toggl
eButton');

     if (savedTheme) {
     this.setTheme(savedTheme,
body, main, moonIcon, sunIcon,
footer);
     if (savedTheme ===
'dark') {

this.showMoonIcon(moonIcon,
sunIcon);
     } else {

this.showSunIcon(moonIcon,
sunIcon);
     }
     } else {
     this.setTheme('light',
body, main, moonIcon, sunIcon,
footer);
```

```
this.showSunIcon(moonIcon,
sunIcon);
      }

      if (toggleButton) {

toggleButton.addEventListener(
'click', () => {

body.classList.toggle('dark');
            if (main) {

main.classList.toggle('dark');

main.classList.toggle('theme-d
ark');
            }
            const isDarkTheme:
boolean =
body.classList.contains('dark'
);
            if (isDarkTheme) {

this.setTheme('dark', body,
main, moonIcon, sunIcon,
footer);

this.showMoonIcon(moonIcon,
sunIcon);
            } else {

this.setTheme('light', body,
main, moonIcon, sunIcon,
footer);

this.showSunIcon(moonIcon,
sunIcon);
            }
      });
      }
      });
   }

   if
(!this.cookieService.get('cook
iePermission')) {

this.showCookiePermissionDialo
g();
   }
 }

 switchLanguage(language:
string) {

this.translateService.use(lang
uage).subscribe(() => {

this.cookieService.set('select
edLanguage', language, 365);
   });
 }

 private setTheme(theme:
string, body: HTMLElement,
main: HTMLElement | null,
moonIcon: HTMLElement | null,
sunIcon: HTMLElement | null,
footer: HTMLElement | null):
void {
   if (theme === 'dark') {
     this.applyDarkTheme(body,
main, moonIcon, sunIcon,
footer);
   } else {

this.applyLightTheme(body,
main, moonIcon, sunIcon,
footer);
   }

this.cookieService.set('theme'
, theme, 365);
 }

 private applyDarkTheme(body:
HTMLElement, main: HTMLElement
| null, moonIcon: HTMLElement
| null, sunIcon: HTMLElement |
null, footer: HTMLElement |
null): void {
   if (moonIcon) {
     moonIcon.style.display =
'none';
   }
   if (sunIcon) {
     sunIcon.style.display =
'block';
   }
   body.classList.add('dark');

body.classList.add('theme-dark
');
   if (main) {

main.classList.add('dark');

main.classList.add('theme-dark
');
   }
```

```
    if (footer) {
      footer.style.background =
'#111';
      footer.style.color =
'#fff';
    }
  }

  private applyLightTheme(body:
HTMLElement, main: HTMLElement
| null, moonIcon: HTMLElement
| null, sunIcon: HTMLElement |
null, footer: HTMLElement |
null): void {
    if (moonIcon) {
      moonIcon.style.display =
'block';
    }
    if (sunIcon) {
      sunIcon.style.display =
'none';
    }

body.classList.remove('dark');

body.classList.remove('theme-d
ark');
    if (main) {

main.classList.remove('dark');

main.classList.remove('theme-d
ark');
    }
    if (footer) {
      footer.style.background =
'';
      footer.style.color = '';
    }
  }

  private
showMoonIcon(moonIcon:
HTMLElement | null, sunIcon:
HTMLElement | null): void {
    if (moonIcon) {
      moonIcon.style.display =
'block';
    }
    if (sunIcon) {
      sunIcon.style.display =
'none';
    }
  }

  private showSunIcon(moonIcon:
HTMLElement | null, sunIcon:
HTMLElement | null): void {
    if (moonIcon) {
      moonIcon.style.display =
'none';
    }
    if (sunIcon) {
      sunIcon.style.display =
'block';
    }
  }

  private
showCookiePermissionDialog():
void {
    const dialogRef =
this.dialog.open(CookiePermiss
ionDialogComponent);

dialogRef.afterClosed().subscr
ibe(allow => {
      if (allow) {

this.cookieService.set('cookie
Permission', 'true', 365);

      let savedLanguage =
this.cookieService.get('select
edLanguage');
      const DEFAULT_LANGUAGE =
'en';
      if (!savedLanguage) {
      savedLanguage =
navigator.language ||
DEFAULT_LANGUAGE;

this.cookieService.set('select
edLanguage', savedLanguage,
365);
      }

this.switchLanguage(savedLangu
age);
      }
    });
  }
}


@login_required(login_url='/lo
gin')
def changepassword(request):
    if request.method ==
'POST':
```

```
      form =
PasswordChangeForm(request.use
r, request.POST)
      if form.is_valid():
            user = form.save()

update_session_auth_hash(reque
st, user)  # Important!

messages.success(request,
'Your password has been
successfully changed!!!')
            return
redirect('Account')
      else:

messages.error(request,
str(form.errors))
            return
HttpResponseRedirect('/account
/changepassword')
   else:
      current_user =
request.user
      customer =
Customer.objects.get(user_id=c
urrent_user.id)
      carts =
Cart.objects.filter(user_id=cu
rrent_user.id)
      qty = 0
      total = 0
      for cart in carts:
            total = total +
cart.amount
            qty = qty + cart.qty
      form =
PasswordChangeForm(request.use
r)
      details = {
            'customer':customer,
            'qty':qty,
            'total':total,
            'carts':carts,
      }

def products(request):
   current_user = request.user
   customer = []
   try:
      customer =
Customer.objects.get(user_id=c
urrent_user.id)
   except:
      pass

   categories =
Category.objects.all()

   categoryid = 0
   filtered_category = []

   try:
      categoryid =
request.GET['category']
   except:
      pass

   if categoryid:
      products =
Product.objects.filter(categor
y_id=categoryid)
      filtered_category =
Category.objects.get(id=catego
ryid)
   else:
      products =
Product.objects.all()

   sort_by =
request.GET.get('sort_by',
None)
   if sort_by == 'price_asc':
      products =
products.order_by('price')
   elif sort_by ==
'price_desc':
      products =
products.order_by('-price')

   n = len(products)

   current_user = request.user
   carts =
Cart.objects.filter(user_id=cu
rrent_user.id)
   qty = 0
   total = 0
   for cart in carts:
      total = total +
cart.amount
      qty = qty + cart.qty

   params = {
      'customer':customer,
      'products': products,
      'categories': categories,
      'filtered_category':
filtered_category,
      'n': n,
      'qty': qty,
      'total':total,
```

```
        'carts':carts,
        }

class Signup(View):
    def get(self, request):
        current_user =
request.user
        if current_user.id:

messages.success(request,
'Already logged in!!!')
            return
redirect('Account')
        return render(request,
'signup.html')

    def post(self, request):
        email =
request.POST['email']
        fname =
request.POST['fname']
        lname =
request.POST['lname']
        phone =
request.POST['phone']
        pass1 =
request.POST['pass1']
        pass2 =
request.POST['pass2']
        uname = email
        print(uname, fname,
lname, email, phone, pass1)

        values = {
            'email': email,
            'fname': fname,
            'lname': lname,
            'phone': phone,
        }
        if
User.objects.filter(email=emai
l):

messages.success(request,
"E-mail Already
Registered!!!")
            return
render(request, 'signup.html',
values)
        if len(fname) > 10 and
len(lname) > 10:

messages.success(request,
"First or Last Name too
long!!!")
            return
render(request, 'signup.html',
values)
        if not fname.isalpha() or
not lname.isalpha():

messages.warning(request,
"Name must contain only
letters.")
            return
render(request, 'signup.html',
values)
        if len(str(phone)) != 10:

messages.warning(request,
"Phone number must contain 10
digits.")
            return
render(request, 'signup.html',
values)
        if len(pass1) < 5:

messages.warning(request,
"Password too short!!! It must
have at least 5 characters.")
            return
render(request, 'signup.html',
values)
        if pass1 != pass2:

messages.warning(request,
"Passwords don't match!!!")
            return
render(request, 'signup.html',
values)

        new_user =
User.objects.create_user(
            uname,
            email=email,
            password=pass1,

first_name=fname.capitalize(),

last_name=lname.capitalize(),
        )
        new_user.save()

        customer = Customer(
            user=new_user,
            phone=phone,
        )
        customer.save()

        user =
authenticate(request,
```

```
username=email,
password=pass1)
      if user is not None:
            login(request, user)

      messages.success(request,
"Account Created
Successfully!!!")
      return
redirect('Account')
@login_required(login_url='/lo
gin')
def updateprofile(request):
   if request.method == "GET":
      current_user =
request.user
      carts =
Cart.objects.filter(user_id=cu
rrent_user.id)
      qty = 0
      total = 0
      for cart in carts:
            total = total +
cart.amount
            qty = qty + cart.qty
      customer =
Customer.objects.get(user_id=c
urrent_user.id)
      details = {
            'customer':
customer,
            'qty': qty,
            'total': total,
            'carts': carts,
      }
      return render(request,
'updateprofile.html', details)

   if request.method ==
"POST":
      current_user =
request.user
      fname =
request.POST['fname']
      lname =
request.POST['lname']
      email =
request.POST['email']
      phone =
request.POST['phone']
      profile_pic = ''
      try:
            profile_pic =
request.FILES['pic']
      except:
            pass
      if len(fname) > 10 and
len(lname) > 10:

messages.success(request,
"First or Last Name too
long!!!")
            return
HttpResponseRedirect('/account
/updateprofile')
      if not fname.isalpha() or
not lname.isalpha():

messages.warning(request,
"Name must contain only
letters.")
            return
HttpResponseRedirect('/account
/updateprofile')
      if len(str(phone)) != 10:

messages.warning(request,
"Phone number must contain 10
digits.")
            return
HttpResponseRedirect('/account
/updateprofile')

      update_user =
User.objects.get(id=current_us
er.id)
      update_user.username =
email
      update_user.first_name =
fname
      update_user.last_name =
lname
      update_user.email = email
      update_user.save()

      update_customer =
Customer.objects.get(user_id=u
pdate_user.id)
      update_customer.phone =
phone
      if profile_pic:

update_customer.profile_pic =
profile_pic
      update_customer.save()

      messages.success(request,
"Your Account has been
successfully updated!!!")
      return
redirect('Account')
```

```
@login_required(login_url='/login')
def addtowishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    wishlist = Wishlist(user_id=current_user.id, product_id=prid)
    wishlist.save()
    messages.success(request, f"{wishlist.product.product_name} added to your Wishlist.")
    return HttpResponseRedirect(url)

@login_required(login_url='/login')
def removefromwishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    product = get_object_or_404(Product, id=prid)
    try:
        wishlist_item = Wishlist.objects.get(user_id=current_user.id, product_id=prid)
        wishlist_item.delete()
        messages.success(request, f"{product.product_name} removed from your Wishlist.")
    except Wishlist.DoesNotExist:
        messages.error(request, "Wishlist item does not exist.")
    return HttpResponseRedirect(url)


class Login(View):
    def get(self, request):
        if request.user.is_authenticated:

            messages.success(request, 'Already logged in!!!')
            return redirect('Account')
        return render(request, 'login.html')

    def post(self, request):
        email = request.POST.get('email')
        password = request.POST.get('password')
        user = authenticate(request, username=email, password=password)
        if user is not None:
            login(request, user)
            return redirect('Account')
        else:

            messages.warning(request, "E-mail or Password Incorrect!!!")
            return render(request, 'login.html', {'email':email})

def cancelProduct(request, orid, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    product = OrderProduct.objects.get(order_id=orid, user_id=current_user.id, product_id=prid)
    product.status = 'Cancelled'
    product.save()
    messages.success(request, product.product.product_name + " has been cancelled.")
    return HttpResponseRedirect(url)
@login_required(login_url='/login')
def add_product(request):
    if request.method == 'POST':
        product_name = request.POST.get('product_name')
        category_id = request.POST.get('category')
```

```
      description =
request.POST.get('description'
)
      brand =
request.POST.get('brand')
      price =
request.POST.get('price')
      tag =
request.POST.get('tag')
      stock =
request.POST.get('stock')
      product_img =
request.FILES.get('product_img
')
      product_img1 =
request.FILES.get('product_img
1')
      product_img2 =
request.FILES.get('product_img
2')


      category =
Category.objects.get(id=catego
ry_id)

      product = Product(

product_name=product_name,
            category=category,

description=description,
            brand=brand,
            price=price,
            tag=tag,
            stock=stock,

product_img=product_img,

product_img1=product_img1,

product_img2=product_img2
      )
      product.save()

      return
redirect('Products')
@login_required(login_url='/lo
gin')
def account(request):
   orderprs = []
   current_user = request.user

   try:
      customer =
Customer.objects.get(user=curr
ent_user)
   except
Customer.DoesNotExist:
      customer = None

   carts =
Cart.objects.filter(user=curre
nt_user)
   qty = sum(cart.qty for cart
in carts)
   total = sum(cart.amount for
cart in carts)

   categories =
Category.objects.all()

   try:
      wishlists =
Wishlist.objects.filter(user=c
urrent_user)
   except
Wishlist.DoesNotExist:
      wishlists = None

   orders =
Order.objects.filter(user=curr
ent_user).order_by("-placed_at
")
   for order in orders:
      pr =
OrderProduct.objects.filter(or
der=order)
      orderprs.append(pr)

   details = {
      'customer': customer,
      'orders': orders,
      'orderprs': orderprs,
      'qty': qty,
      'total': total,
      'carts': carts,
      'categories': categories,
      'wishlists': wishlists
   }

class Customer(models.Model):
   user =
models.OneToOneField(User,
on_delete=models.CASCADE,
default="")
   phone =
models.BigIntegerField()
```

```
    house_no =
models.CharField(null=True,
max_length=20)
    street =
models.CharField(null=True,
max_length=50)
    city =
models.CharField(null=True,
max_length=20)
    pin =
models.IntegerField(null=True)
    state =
models.CharField(null=True,
max_length=20)
    profile_pic =
models.ImageField(null=True,
upload_to="customer_pic",
default="userimg.png")

    def __str__(self):
        return
self.user.first_name + " " +
self.user.last_name


class Category(models.Model):
    category_name =
models.CharField(max_length=50
)

    def __str__(self):
        return self.category_name


TAG = (
    ('New', 'New'),
    ('Bestseller',
'Bestseller'),
    ('Trending', 'Trending'),
    ('Featured', 'Featured'),
    ('Sale', 'Sale'),
)


class Product(models.Model):
    product_name =
models.CharField(max_length=10
0)
    category =
models.ForeignKey(Category,
on_delete=models.CASCADE)
    description =
models.TextField(max_length=10
00)
    brand =
models.CharField(max_length=20
, default="")
    price =
models.IntegerField(default=0)
    tag =
models.CharField(default="New"
, choices=TAG, max_length=20)
    stock =
models.IntegerField(default=10
)
    product_img =
models.ImageField(upload_to="i
mages", default="product.jpg")
    product_img1 =
models.ImageField(upload_to="i
mages", default="product.jpg")
    product_img2 =
models.ImageField(upload_to="i
mages", default="product.jpg")

    def __str__(self):
        return self.product_name


class Cart(models.Model):
    user =
models.ForeignKey(User,
on_delete=models.SET_NULL,
null=True)
    product =
models.ForeignKey(Product,
on_delete=models.SET_NULL,
null=True)
    qty = models.IntegerField()

    @property
    def price(self):
        return self.product.price

    @property
    def amount(self):
        return self.qty *
self.product.price

    def __str__(self):
        return
self.product.product_name + "
by " + self.user.username



class Order(models.Model):
```

```
    user =
models.ForeignKey(User,
on_delete=models.CASCADE)
    first_name =
models.CharField(max_length=10
)
    last_name =
models.CharField(max_length=10
)
    phone =
models.BigIntegerField(null=Tr
ue)
    house_no =
models.CharField(null=True,
max_length=20)
    street =
models.CharField(null=True,
max_length=50)
    city =
models.CharField(null=True,
max_length=20)
    pin =
models.IntegerField(null=True)
    state =
models.CharField(null=True,
max_length=20)
    total = models.FloatField()
    code =
models.CharField(max_length=15
, default="")
    placed_at =
models.DateTimeField(auto_now_
add=True)

    def __str__(self):
        return self.code + " to "
+ self.first_name + "\t" +
self.last_name

class
OrderProduct(models.Model):
    order =
models.ForeignKey(Order,
on_delete=models.CASCADE)
    user =
models.ForeignKey(User,
on_delete=models.CASCADE)
    product =
models.ForeignKey(Product,
on_delete=models.CASCADE)
    qty = models.IntegerField()
    price = models.FloatField()
    amount =
models.FloatField()
    status =
models.CharField(max_length=20
, choices=STATUS,
default='Placed')
    updated_at =
models.DateTimeField(auto_now=
True)

    def __str__(self):
        return
self.product.product_name + "
by " + self.user.first_name

class Review(models.Model):
    product =
models.ForeignKey(Product,
on_delete=models.CASCADE)
    customer =
models.ForeignKey(Customer,
on_delete=models.CASCADE,
default="")
    subject =
models.CharField(max_length=50
)
    review =
models.CharField(max_length=20
0)
    rating =
models.IntegerField()
    created_at =
models.DateTimeField(auto_now_
add=True)

    def __str__(self):
        return
self.product.product_name + "
by " + self.user.first_name


class Wishlist(models.Model):
    user =
models.ForeignKey(User,
on_delete=models.CASCADE)
    product =
models.ForeignKey(Product,
on_delete=models.CASCADE)
    added_on =
models.DateField(default=now,
editable=False)

    def __str__(self):
        return
self.user.first_name + " - " +
self.product.product_name
```

```
def productdetail(request,
prid):
    current_user = request.user
    product =
Product.objects.get(id=prid)
    reviews =
Review.objects.filter(product_
id=prid)
    carts =
Cart.objects.filter(user_id=cu
rrent_user.id)
    wishlist =
Wishlist.objects.filter(user_i
d=current_user.id,
product_id=prid)

    customer = []
    try:
      customer =
Customer.objects.get(user_id=c
urrent_user.id)
    except:
      pass

    try:
      pr_qty =
Cart.objects.get(user_id=curre
nt_user.id, product_id=prid)
      pr_qty = pr_qty.qty
    except:
      pr_qty = 0


    no_of_reviews =
len(reviews)

    rating = 0
    for review in reviews:
      rating = rating +
review.rating
    try:
      rating =
rating/no_of_reviews
    except:
      pass

    desc = product.description
    descs =
list(desc.split("#"))

    qty = 0
    total = 0
    for cart in carts:
      total = total +
cart.amount
      qty = qty + cart.qty

    details = {
      'customer':customer,
      'product':product,
      'descs':descs,
      'reviews':reviews,

'no_of_reviews':no_of_reviews,
      'rating':rating,
      'qty':qty,
      'total':total,
      'carts':carts,
      'wishlist':wishlist,
      'pr_qty':pr_qty
    }
urlpatterns = [
    path("", account.account,
name="Account"),
    path("account/",
account.account,
name="Account"),
    path("signup/",
signup.Signup.as_view(),
name="SignUp"),
    path("login/",
login.Login.as_view(),
name="LogIn"),
    path("products/",
products.products,
name="Products"),
    path('add_product/',
add_product.add_product,
name='add_product'),

path("productdetail/<int:prid>
",
productdetail.productdetail,
name="ProductDetail"),

path("deletefromcart/<int:prid
>",
cartoperations.deletefromcart,
name="DeletefromCart"),

path("deleteallfromcart/<int:p
rid>",
cartoperations.deleteallfromca
rt, name="DeleteAllfromCart"),

path("addtowishlist/<int:prid>
", wishlist.addtowishlist,
name="AddfromWishlist"),

path("removefromwishlist/<int:
prid>",
```

```
wishlist.removefromwishlist,
name="RemovefromWishlist"),
   path("cart/", cart.cart,
name="Cart"),
   path("clearcart/",
cartoperations.clearcart,
name="ClearCart"),

path("account/updateprofile/",
updateprofile.updateprofile,
name="UpdateProfile"),

path("account/changepassword/"
,
changepassword.changepassword,
name="ChangePassword"),
   path("logout/",
logout.logout_view,
name="Logout")
]
```