Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Інтернет-платформа
для продажу автомобілів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.170119.01.16.ВД

Виконав студент IV курсу група ІПЗ-17-1 _____ А.Г. Собко
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ О.Г. Онишко
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

15 червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
5 . 02 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Собку Артему Геннадійовичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Інтернет-платформа для продажу автомобілів

Керівник проєкту (роботи) Онишко Оксана Григорівна
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

кандидат педагогічних наук, доцент

Затверджена наказом ректора університету від 05.02.2021 р. № 11
2. Строк подання студентом проєкту (роботи) на кафедру 01.06.2021 р.
3. Вихідні дані до проєкту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування Інтернет-платформи, програмна реалізація, тестування програми

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 16 шт)

6. Консультанти розділів дипломного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийн
Нормоконтроль	Радельчук Г. І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання «05» лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Приміт
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних там ДП	01.12 – 30.12.2020	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2021	
3 Проектування програмного забезпечення	01.02 – 28.02.2021	
4 Програмна реалізація	01.03 – 10.04.2021	
5 Тестування програмного забезпечення	11.04 – 30.04.2021	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2021	
7 Попередній захист ДП	Травень 2021 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки)	26.05 – 30.05.2021	
9 Підготовка до захисту та захист ДП	з 01.06.2021	

Студент _____ А. Г. Собко
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О. Г. Онишко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: «Інтернет-платформа для продажу автомобілів».

Автор проекту: Собко Артем Геннадійович.

Керівник проекту: Онишко Оксана Григорівна.

Пояснювальна записка: 88 с, 24 рис., 5 табл., 3 дод., 12 джерел.

Графічна частина: 16 презентаційних слайдів.

ІНТЕРНЕТ-ПЛАТФОРМА, ПРОДАЖ АВТО, REST, MERN, MONGODB, JAVASCRIPT, NODE.JS, REACT.JS, ROBO 3T, MVC, SASS.

Метою проекту є розробити Інтернет-платформу, яка здатна перенести продажі автомобілів у Інтернет завдяки використанню Інтернет оголошень, що в свою буде корисним як і для продавців. В свою чергу додаток має мати зручний та інтуїтивний дизайн.

У дипломному проекті визначено поняття Інтернет-платформ; проаналізовано існуюча проекти з схожою тематикою; описано та порівняно різні типи архітектури; описано різні типи баз даних, таких як реляційна та нереляційна; описано принципи побудови фронт-енд та бек-енд частини; описано способи та методи взаємодії цих частин; розроблено модель бази даних; вибрано інструменти для реалізації Інтернет-платформи.

Для реалізації Інтернет-платформи використовується нереляційна база даних під керуванням MongoDB, мову програмування JavaScript. Для фронт-енда використовується бібліотека React.js, а для бек-енда платформа Node.js, а також фреймворк Express.js.

У результаті проектування була програмно реалізовано систему для публікації оголошень, а саме Інтернет-платформу для продажу автомобіля.

1.06.2021
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПШПЗ.170119.01.16.ПЗ	Пояснювальна записка	88		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	16		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПШПЗ. 170119.01.16.ПЗ		
Виконав		Собко А.Г.		11.06	Інтернет-платформа для продажу автомобілів Відомість документів	Літ.	Арк. 1 / Акрушів 1
Керівник		Онишко О.Г.		11.06			
Н. Контр.		Радельчук Г.І.		11.06		ХНУ, ІПЗ-17-1	
Зав. Каф.		Бедратюк Л.П.		15.06			

ЗМІСТ

ВСТУП

З кожним днем технології все більше і більше розвиваються, а разом з ними поширюються використання і вдосконалюються можливості Інтернету, не тільки для вивчення нової інформації, а і для отримання певних послуг. Прикладом таких послуг є Інтернет-платформи з продажу авто, які дозволяють користувачам як придбати, так і продати авто не прикладаючи зусиль, та, навіть, не виходячи із власного дому.

І справді, ймовірність того, що людина надасть перевагу онлайн-ресурсам, а не салонам чи авторинкам надзвичайно велика. Адже саме так покупець може одразу задати пошук по необхідним характеристикам, починаючи з пробігу автомобіля і закінчуючи роком його випуску та бажаною ціною. Що стосується продавця, то йому не потрібно буде орендувати місце на авторинку і чекати на потенційного покупця, тому що легше буде розмістити оголошення з продажу авто на популярних сайтах, вказати основну інформацію про автомобіль, характеристики та чекати покупця уже в режимі онлайн, не виходячи з дому і займаючись іншими справами. Також покупці не витрачають безліч часу на поїздки для огляду авто, адже вони можуть ознайомитись із фотографіями на сайті, а вже потім надати перевагу чомусь одному. Отже, обидві сторони залишаються задоволеними.

Саме тому, тема є актуальною, адже врахувавши недоліки існуючих сайтів з продажу авто можна створити оптимальний, зручний та конкурентоспроможний програмний продукт, який зможе задовольнити усі потреби користувачів. Завдяки безлічі фільтрів можна буде обрати бажане авто ще швидше, адже існуючі зараз Інтернет-платформи з продажу авто не можуть похизуватись великою кількістю таких фільтрів, що ускладнює покупцю пошук оптимального автомобіля з найбільш підходящими характеристиками.

Метою мого проекту є розробка Інтернет-платформи із продажу автомобілів, яка буде зручною та зрозумілою у використанні, яка на відміну від

сучасних сайтів буде містити більш розширений пошук, що дозволить швидше та ефектніше знайти бажане авто.

Для реалізації цього проекту потрібно:

– проаналізувати вибрану предметну область на предмет її актуальності;

– проаналізувати ринок уже готового програмного забезпечення з визначеною темою;

– визначитися з оптимальною архітектуро для реалізації запланованого програмного забезпечення;

– проаналізувати доступні реляційні та нереляційні бази даних, визначити їх особливості та властивості;

– спроектувати програмний продукт для подальшої його реалізації, за вибраною предметною областю;

– реалізувати спроектовану Інтернет-платформу для продажу автомобілів

– провести тестування готового програмного забезпечення.

Очікуваним результатом дипломного проекту є повноцінна та найоптимальніша Інтернет-платформа з продажу автомобілів, яка зможе скласти значну конкуренцію уже існуючим сайтам та повністю задовольнити усі потреби користувачів завдяки великому функціоналу, привабливому зовнішньому дизайну та комфортному використанню.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

З кожним днем все більше сервісів переходять на онлайн продажі, так як найти певні товари в офлайн магазинах не так і просто. На щастя з розвитком нових технологій онлайн продажі стали доступним майже для будь кого на планеті. Тисячі товарів продаються саме через Інтернет і ця цифра тільки зростає, тому що це швидко та зручно, не потрібно навіть вставати з ліжка.

Запланований мною сайт позбавить людей зайвих проблем, тому що не вигідно роз'їжджати по різним авторинкам та автосалонам з надією підібрати саме те авто яке задовільнить покупця. Продавцям також не вигідно цілими днями стояти і очікувати на клієнта, при тому ще й плативши оренду за місце, так як покупець може знаходитись у іншому місті.

Але, якщо усі ці оголошення перенести на спеціалізовану Інтернет-платформу, то можна буде дома обрати авто у будь якому місті країни, яке задовільнить по всім параметрам, а для зручності у виборі можна встановити різні фільтри, також це допоможе швидше та простіше продати власне авто, тому що його побачать у всьому світі. Це вигідно як і для покупців так і для продавців.

Такою є основна мета моєї Інтернет-платформи, який буде розроблений у процесі виконання дипломного проекту. Так як зараз у більшості є авто, а то й декілька, я вважаю що дана тема є доволі актуальною користувачам буде зручно обирати та продавати авто, а різні фільтри цьому посприяють.

На ринку існують різні засоби для продажу товарів через Інтернет, але не зважаючи на це в них є свої недоліки, основним з яких є доволі дорогі тарифні плани, а також різні баги, не великий вибір фільтрів для пошуку та мало функцій у кабінеті продавця.

Враховуючи усе вище написане. Розроблений програмний продукт, зможе скласти гарну конкуренцію уже існуючим та задовільнити більшість потреб для користувачів та спростити їхні життя.

Використаємо методологію IDEF0 моделювання для опису та аналізу бізнес-процесів. Контекстна діаграма роботи Інтернет-платформи з продажу авто зображена на рисунку 1.1. Вхідними даними роботи програмного продукті є запити користувачів, данні користувачів та редагування адміністрацією, механізмом роботи є системи управління та модератори, у результаті цього ми отримуємо оголошення опубліковані користувачами, результати пошуку автомобілів, успішно зареєстрованих користувачів.

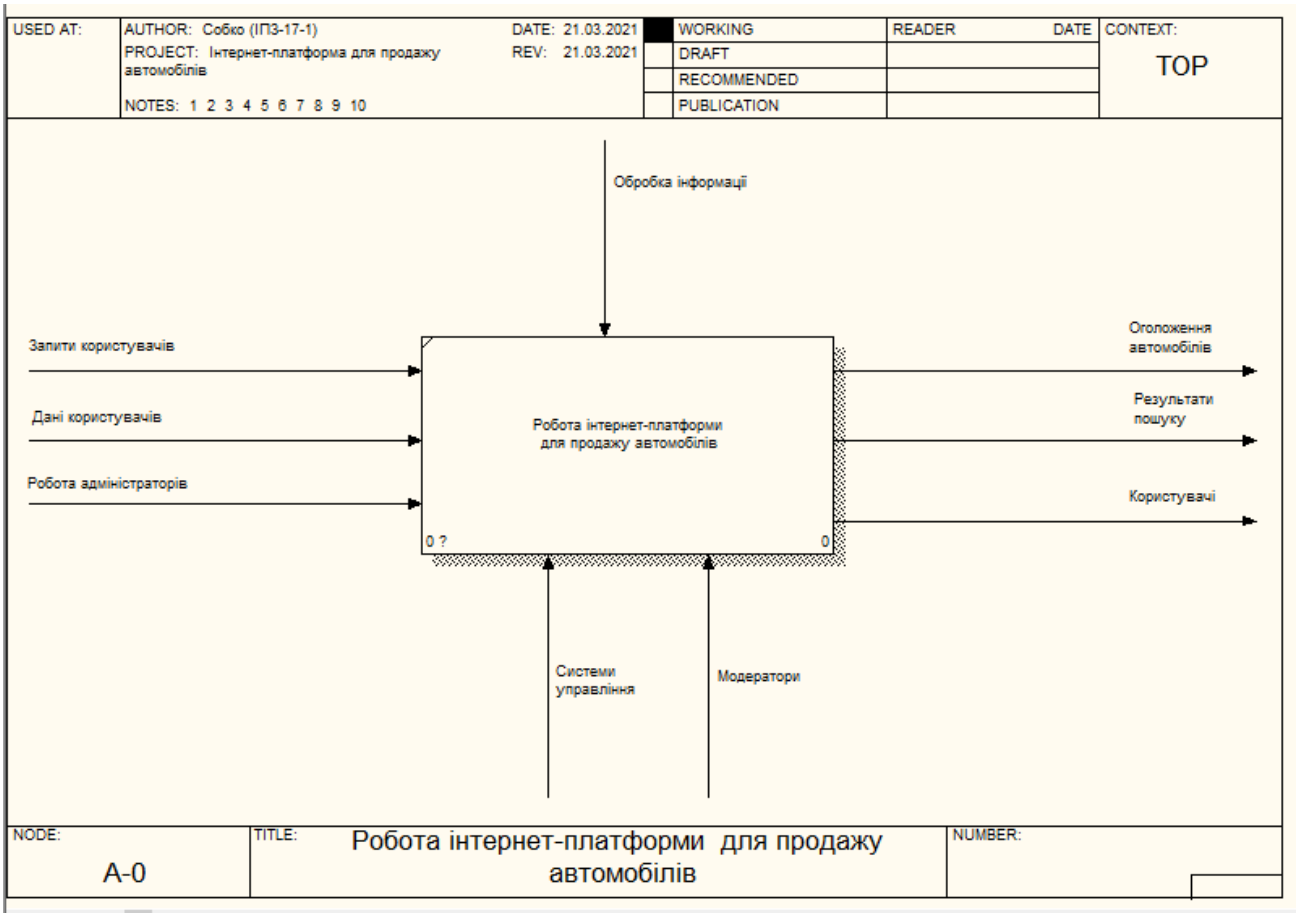


Рисунок 1.1 – Діаграма IDEF0 роботи Інтернет-платформи для продажу авто

Щоб детальніше розглянути основні особливості потрібно зробити декомпозицію діаграми IDEF0, результат декомпозиції представлений на

рисунку 1.2. Ми можемо виділити чотири основних функцій програмного продукту, такі як: створення оголошення, редагування існуючих оголошень, авторизація користувачів у системі, перегляд існуючих оголошень.

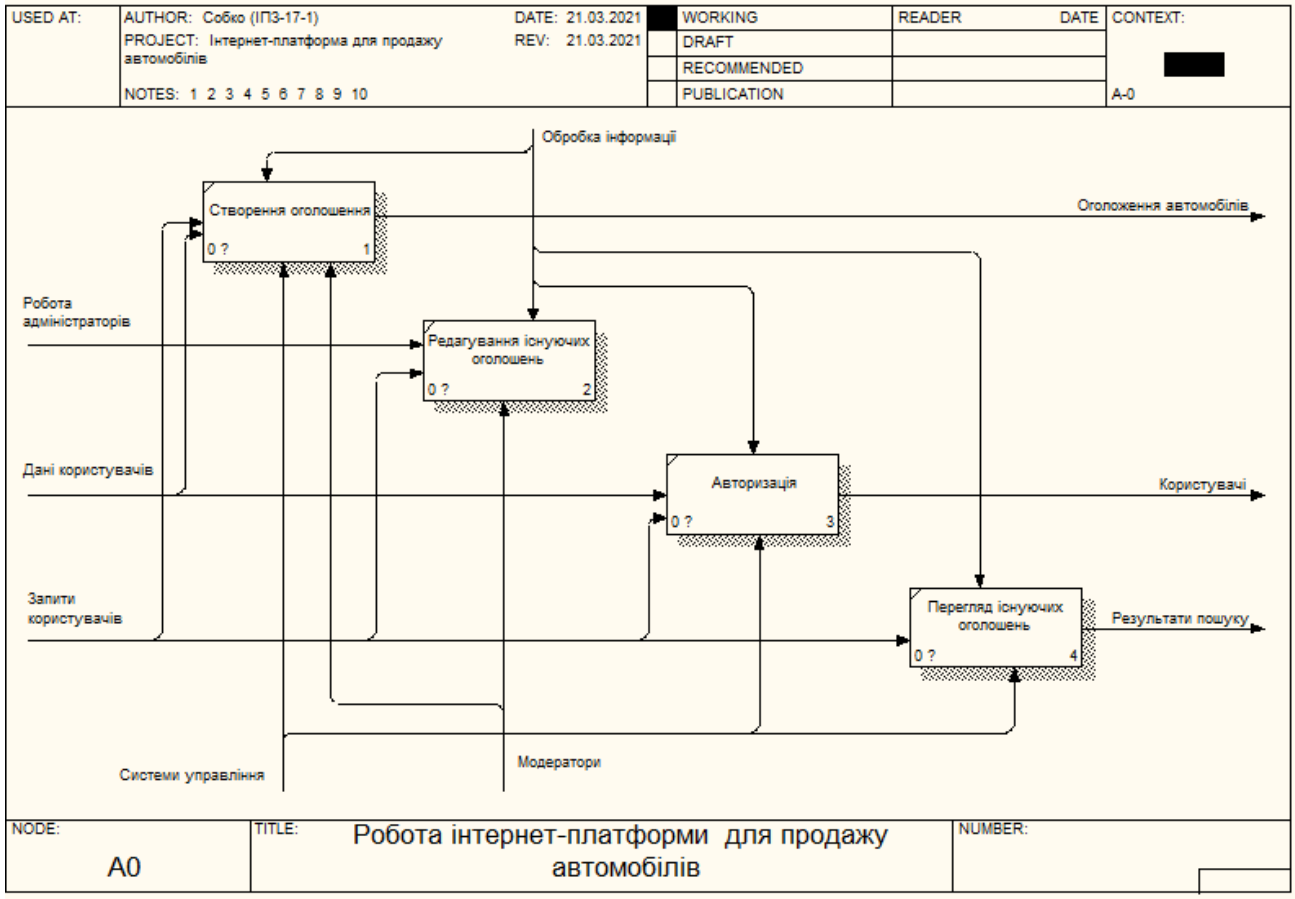


Рисунок 1.2 – Декомпозиція діаграми IDEF0 першого рівня

Для уникнення нецензурних, не сумісних з тематикою оголошень, а також іншого спаму було прийняте рішення публікувати контент після модерації.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Зараз на просторах Інтернету є Інтернет-платформи, які дозволяють користувачеві купити, а також продати авто. Усі вони користуються популярністю, але так чи інакше, мають свої недоліки. А це означає щоб мною

розроблювана Інтернет-платформа став популярним, потрібно прикласти не мало зусиль, виправити недоліки головних конкурентів.

Для прикладу порівняємо найбільш популярні торгові платформи, які призначені для продажу авто. Порівнювати платформи будемо за наступними пунктами: призначення, дизайн, функціональність, інтуїтивність, зручність.

Одним із найпопулярніших сервісів з продажу авто є https://auto.ria.com/, головна сторінка якого зображена на рисунку 1.3.

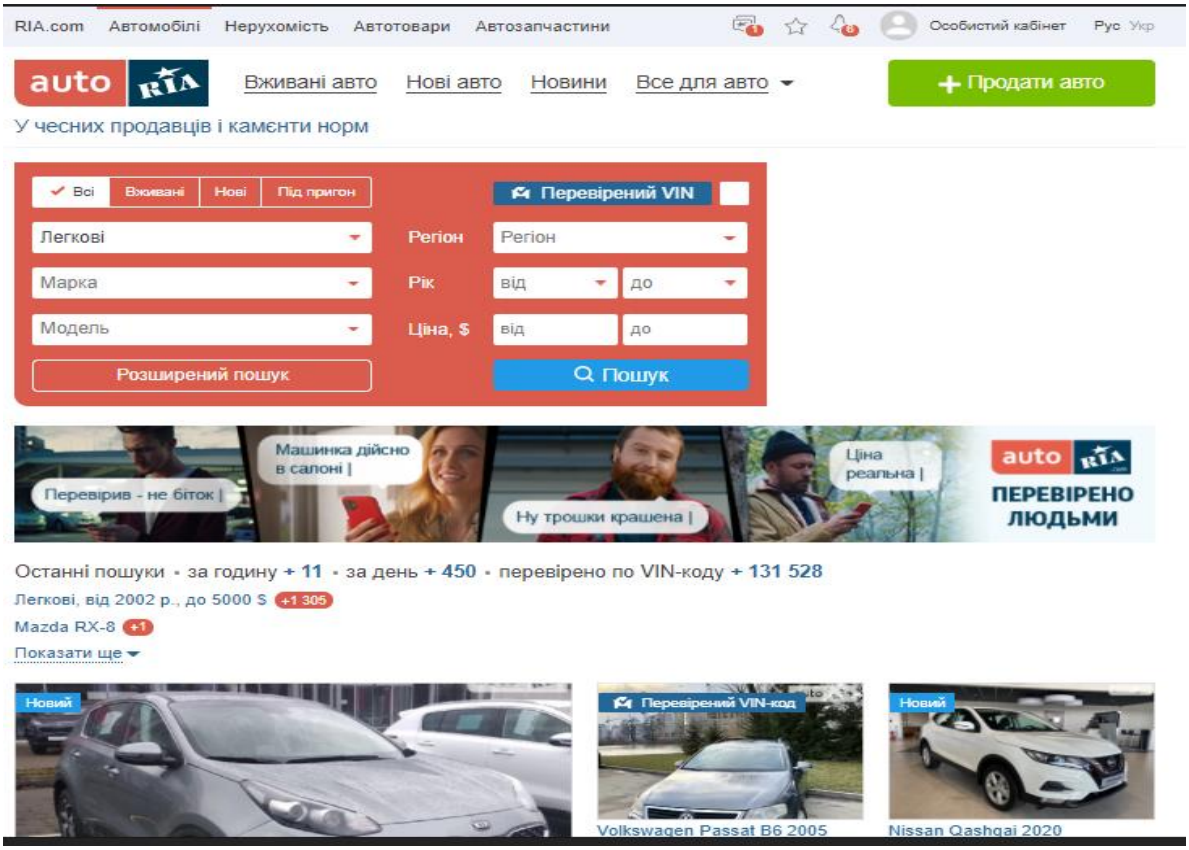


Рисунок 1.3 – Головна сторінка https://auto.ria.com/

На сайті є різні оголошення від продавців. Вони сортуються у відповідні категорії (легковики, вантажівки, причепи, спец техніка, автобуси та інші). Тут можна як і продати свій транспорт так і придбати. Кожне з оголошень має свій опис, фото та технічні характеристики.

Інтерфейс є доволі зрозумілим та має велику кількість функцій, дизайн мінімалістичний, але не досить привабливий. На даному порталі більшість

функцій є платні, а кількість безкоштовних оголошень обмежена одним оголошенням у півроку, а також багато реклами, що відштовхує потенційних клієнтів. Але з основними функціями сайт справляється.

Ще один портал для продажу транспорту представлений на рисунку 1.4 (https://rst.ua/).

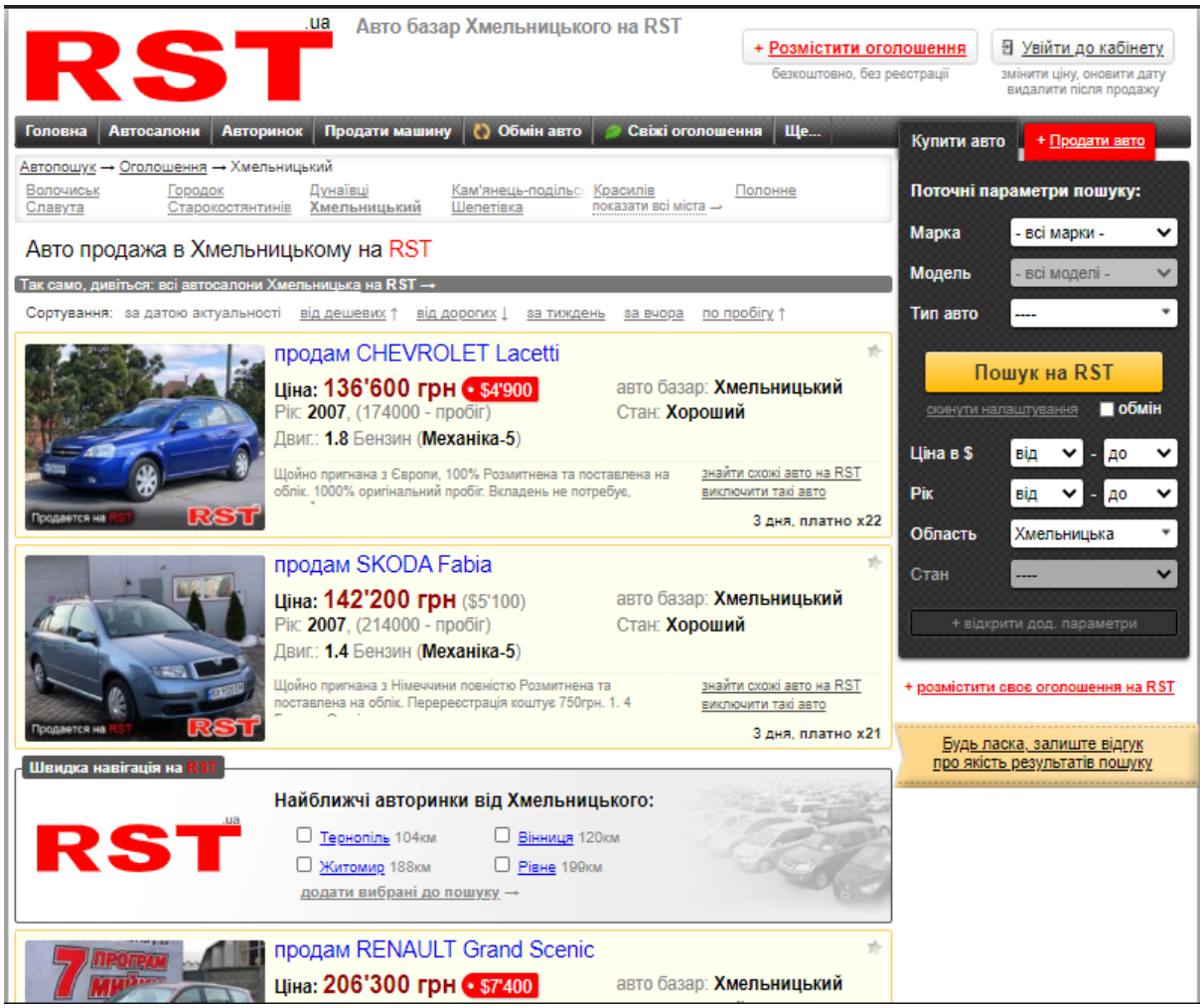


Рисунок 1.4 – Головна сторінка https://rst.ua/

Це ще одна популярна сторінка в Україні, яка відповідає такій тематиці. На ньому можна як і переглядати оголошення так і публікувати власні. Але одразу можна замітити що авторизація на даному порталі можлива тільки завдяки логіну та паролю, немає можливості авторизуватися за допомогою різним популярним обліковим записав, таким як Google, Facebook та інші. Функціонал менший ніж на попередньому сайті, але доволі інтуїтивний. На сайті

немає такої великої кількості реклами як на попередньому, проте дизайн не найкращий, на мою думку він є доволі застарілим та не сучасним.

Також відомою Інтернет-платформою є https://www.olx.ua/, на якій продаються багато різноманітних товарів, в тому числі і авто. Головна сторінка даного сайту представлена на рисунку 1.5.

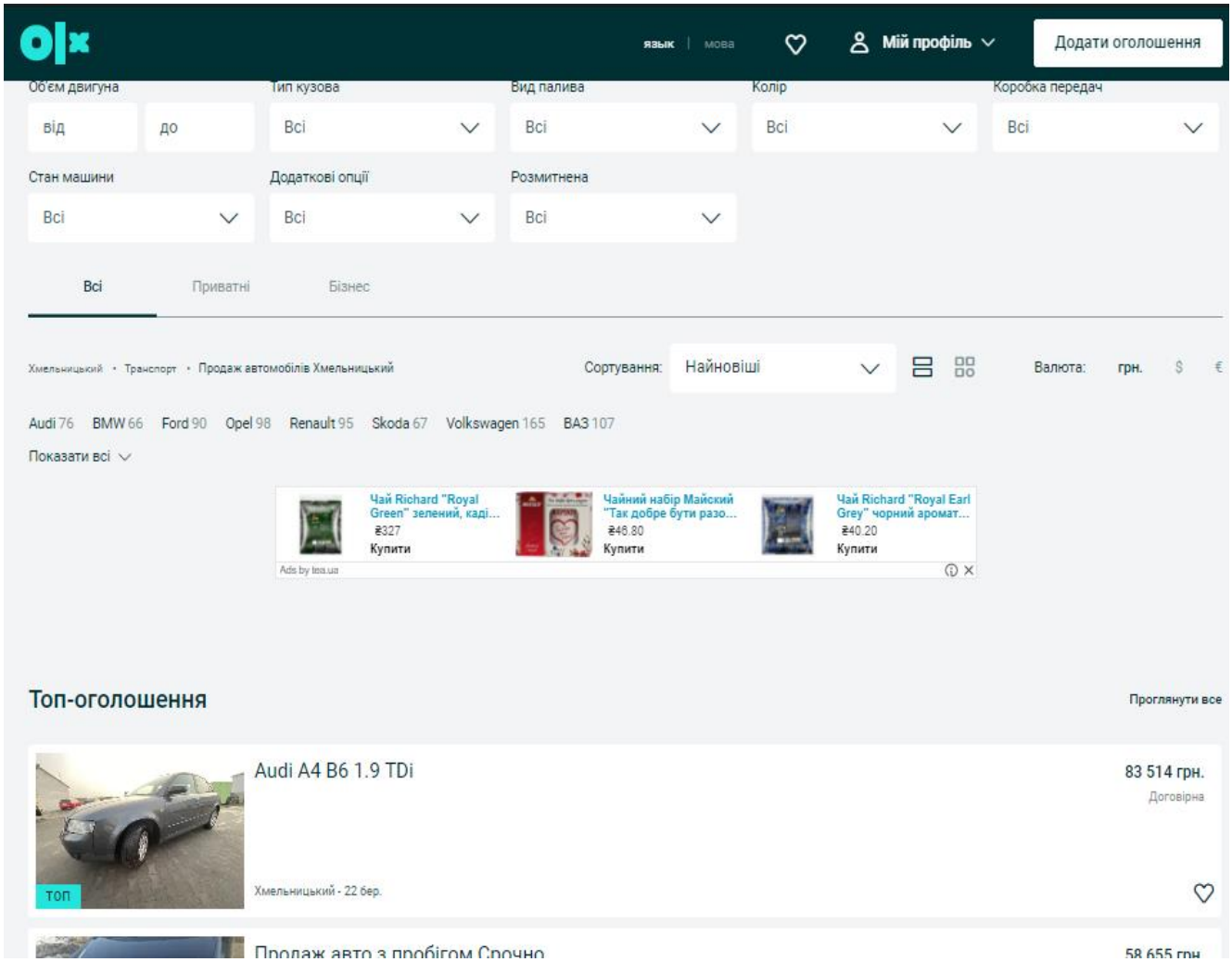


Рисунок 1.5 – Головна сторінка https://www.olx.ua/

Так як, представлена площадка призначена, для великого різноманіття товару та не є вузько спеціалізована, то вона немає достатньої кількості фільтрів для пошукі, але є доволі інтуїтивна. Також тут відсутня можливість покупки нового авто, на відміну від попередніх сайтів. З переваг варто відмітити, що є

доволі велика кількість безкоштовних пакетів послуг в певних категоріях, а реклама вже не є такою нав’язливою.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Визначимо вимоги до програмного забезпечення використавши популярну діаграму UML.

UML – уніфікована мова призначена для об'єктно-орієнтованого моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки додатків. Різні види діаграм які підтримуються UML і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було з легкістю перевести в програмний код.

Зобразимо акторів програмної системи та їх опис у таблиці 1.1, а у таблиці 1.2 представимо самі варіанти використання

Таблиця 1.1 – Опис акторів програмної системи

Актор	Короткий опис
1	2
Неавторизований користувач	Має можливість шукати та переглядати оголошення інших користувачів.

Кінець таблиці 1.1

1	2
Авторизований користувач	Додатково до функціоналу неавторизованого користувача має можливість створювати та видаляти власні оголошення переглядати статистику за ними, зберігати вподобані оголошення, редагувати данні власного облікового запису.
Авторизований користувач у ролі адміністратора або модератора	Адміністратори та модератори мають можливість видаляти оголошення користувачів, які на їхню думку мають зміст не сумісних з тематикою сайту, а також блокувати користувачів які порушують умови користувачів сайту.

Таблиця 1.2 – Опис варіантів використання

Актор	Найменування ВВ	Опис ВВ
	Реєстрація в системі	Користувач може створити новий обліковий запис за допомогою електронної пошти або облікового запису Google.
	Авторизація за допомогою електронної пошти або облікового запису Google	У випадку якщо користувач був зареєстрований раніше, він має можливість авторизуватися за допомогою електронної пошти або за допомогою облікового запису Google.
	Перегляд оголошень інших користувачів	Користувач може шукати оголошення інших користувачів,

Продовження таблиці 1.2

		використовуючи різноманітні фільтри, та переглядати повну інформацію по них.
	Редагування даних облікового запису	Користувач може змінювати власні дані у особистому кабінеті.
	Публікація оголошення та керування ними	Користувач може публікувати власні оголошення, видаляти застарілі, а також змінювати вже існуючі.
	Додавання до вподобаних оголошень	Користувач може вподобати оголошення, цим самим зберегти його в окремий приватний список, який доступний виключно йому.
	Перегляд оголошень інших користувачів	Користувач може шукати оголошення інших користувачів, використовуючи різноманітні фільтри, та переглядати повну інформацію по них.
	Видалення оголошення	Модератор може видаляти оголошення які не задовольняють користувацькі умови.
	Модерація оголошень	Модератор може переглядати оголошення і та вирішувати чи вони задовольняють умови

Кінець таблиці 1.2

		для публікації цим самим допускає оголошення або ж ні.
	Видалення або блокування облікового запису користувача	Якщо адміністратор замітив порушення правил користування сайту користувачем, може заблокувати обліковий запис на певний термін, а у разі злісного порушення правил користування може його взагалі видалити

На рисунку 1.6 зображено діаграму варіантів використання згідно з таблицями 1.1 та 1.2.

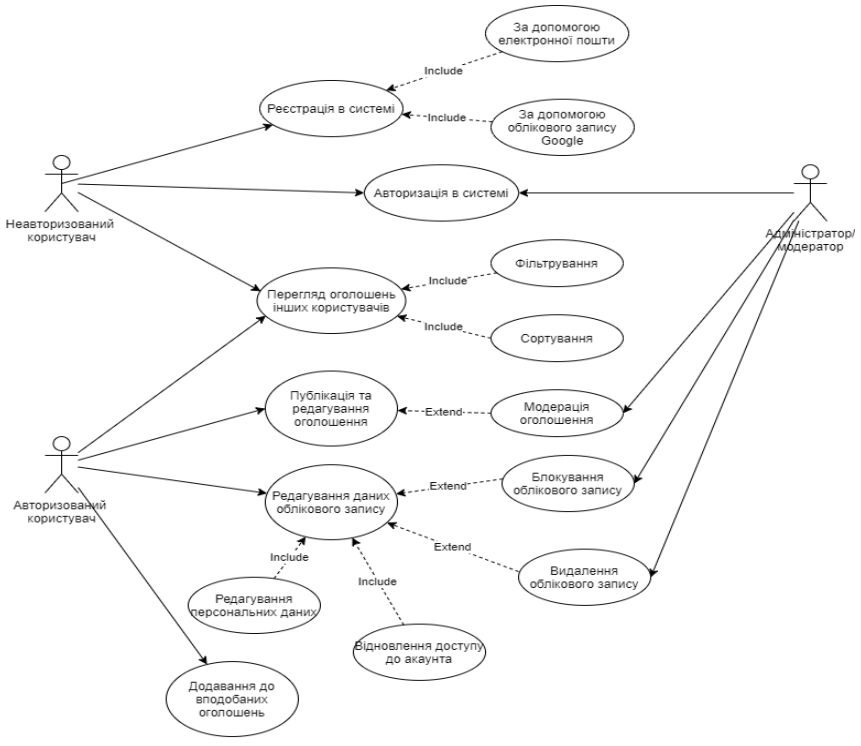


Рисунок 1.6 – Діаграма використання Інтернет-платформи для продажу авто

Технічне завдання на розробку Інтернет-платформи розміщено в додатку А.

Отже, було проведено дослідження предметної області та з’ясовано шо Інтернет-платформи для продажу товарів користуються доволі великим попитом та відіграють у житі сучасної людини велику роль, так як багато продажів та покупок здійснюються саме онлайн. А ще вони повинні бути зручними у використані, інтуїтивно зрозумілими та надійно зберігати персональні данні кожного користувача.

Проаналізувавши вже існуючі Інтернет-платформи з даною тематикою видно що існують декілька подібних додатків і вони користуються популярністю, вони відрізняються своїм дизайном, функціоналом та доступністю, проте вони не є ідеальними та мають свої як переваги, так і недоліки. Також, завдяки діаграмі використання, було визначено основні функціональні вимоги та ролі користувачів системи, які повинні бути реалізовані у програмному продукті.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури веб-додатка

Усі існуючі Інтернет-платформи та будь-які веб сервіси побудовані на клієнт-серверній архітектурі. Така архітектура набула своєї популярності у ході динамічного розвитку Інтернет мережі. Вона зосереджує велику частину своєї інформації на серверах, які обслуговують своїх клієнтів, що і є великою її перевагою. Архітектура такої мережі передбачає наступні компоненти: клієнт, сервер та мережа. У якості клієнта можна розглядати як комп'ютер з якого здійснюється запити на інші комп'ютери, також у якості клієнта можна розглядати клієнтське програмне забезпечення або з самих людей які намагаються за допомогою програмного забезпечення отримати бажану інформацію. Але найбільш загальноприйнятим варто вважати що програмні модулі і є клієнтами та серверами. Варіанти коли серверна і клієнтська програми розміщуються на одній одному комп'ютері називається локальним сервером, але найчастіше вони розташовані на різних комп'ютерах та у різних куточках землі та з'єднані за допомогою мережі Інтернет. Навіть електричний чайник може слугувати клієнтом, який відправить запит на сервер що закипів, а той у свою чергу відповідне повідомлення у мобільний додаток користувачеві. У якості сервера виступає комп'ютер який під'єднаний до локальної чи глобальної мережі. Зазвичай він значно потужніший за звичайні персональні комп'ютери та працює цілодобово, тому і здатний оброблювати великий потік інформації

На основі цього можна зробити висновок, що сервер це машина яка слугує виключно для зберігання та обробки даних, а також здатна отримувати, обробляти та повертати HTTP-запити. А у якості клієнта може слугувати будь-що, що здатне формувати та відправляти HTTP-запити. Дана архітектура за допомогою мережевих технологій надає доступ до різноманітних ресурсів, що зберігаються на серверах, а розробка клієнт-серверних додатків підвищує безпеку та корисність самих додатків в цілому.

Передача даних між сервером та клієнтом здійснюється по мережі за допомогою протоколу HTTP. Повна назва Hyper Text Transfer Protocol – це протокол що використовується для передачі даних в мережі. Дані представляють собою гіпертекст, а саме HTML, повна назва якого Hyper Text Markup Language. Отримавши HTML код від сервера, клієнтська частина розшифровує його завдяки цьому користувачі можуть відразу побачити готове зображення.

Також існує розширений протокол HTTP, а саме HTTPS який використовує SSL (Secure Sockets Layer) шифрування даних, якими обмінюється комп'ютер з веб-сервером. Використання такого захищеного з'єднання дозволяє забезпечити захист персональної інформації. Раніше його використовували для банківських систем та інших сервісів з конфіденційною інформацією, наразі протоколом HTTPS користується більшість сервісів у мережі Інтернет.

Отже, прийнято рішення даний дипломний проект реалізувати за допомогою клієнт-серверної архітектури для взаємодії.

2.2 Опис структури даних та моделі бази даних

Будь-які ресурси мають інформацію які зберігається та відображається. Найзручніше та безпечніше для цього використовувати базу даних. Вона допоможе надійно зберегти данні та надати зручний доступ до них. На сьогодні існує два головних типи баз даних які використовуються для розробки програмного забезпечення. Це реляційні та нереляційні бази даних.

У реляційній базі даних зручність відносин полягає у тому, що вони представлені у вигляді таблиць. Кожен рядок у такій таблиці є кортежом, а атрибутом називають стовпець. Виходить, що данні представлені двовимірно. Один рядок, або ж кортеж, має містити два атрибути, перший представляє тип об'єкта (до прикладу: ім'я користувача, електронна пошта), який має бути унікальним та екземпляр цього ж об'єкту в якому повинні розміщуватися елементи одного типу в цій таблиці, для кожного екземпляру даного об'єкту.

Головною вимогою є що один кортеж будь-якого відношення має відрізнятися від іншого будь-якого кортежу з цим відношенням, якщо сказати більш простими словами, то будь яке відношення має мати первинний ключ. Перший атрибут має бути ключовим, він однозначно ідентифікує таблицю. Один з ключових атрибутів має бути первинним, значення такого атрибуту взагалі не можуть обновлятися. У реляційної бази даних немає групових відносин, на відміну від мережевої та ієрархічної моделей даних.

Реляційні бази даних мають ряд переваг та недоліків. Перевагами є:

– таке зображання бази даних дає більш звичний для користувачів та розробників спосіб представити данні, так як результатом такої бази даних є набір таблиць з певними зв'язками;

– завдяки математичному апарату базуються строгі правила проектування;

– так як дані повністю незалежні то при змінні реляційної бази даних, зміни у програмі будуть мінімальні;

– не потрібно знати організацію конкретної бази даних у пам'яті для написання програмного забезпечення та побудові запитів.

У якості недоліків реляційної бази даних можна визначити наступні пункти:

– не завжди у вигляді таблиць можна представити предметну область;

– при створені великих проектів та відповідно логічного проектуванні бази даних створюється багато таблиць. Це викликає труднощі при розумінні структури даних;

– такий тип бази даних займає доволі таки багато зовнішньої пам'яті, а зі зростанням кількості користувачів та оголошень від них, кількість зайнятого простору буде тільки зростати, з цього з'являється наступний пункт;

– відносно невелика швидкість доступу до даних.

Для зручного доступу до бази даних використовуються різноманітні системи керування базами даних. Такі як: MySQL, MariaDB, PostgreSQL, SQLite.

Вони представляють собою взаємопов’язаний набір даних та програм, які забезпечують доступом до цих даних. Вони дають можливість створити, оновити, знайти або ж і взагалі видалити певний набір даних в базі, забезпечуючи безпечний доступ до них.

Також існують це нереляційні бази даних, які ще мають назву NoSQL. Ключовим фактором для створення нової концепції бази даних на противагу звичайні реляційній базі даних стало зростання даних у всесвітній павутині. З цим з’явилось нове поняття BigData. Це спосіб роботи з величезними обсягами пам’яті, які постійно зростають. Модель такої бази даних була націлена на швидкість роботи з великими масивами даних та масштабованість.

Загалом можна виділити основні способи збереження даних в NoSQL базах даних, їх є чотири види, це бази даних за принципом: «ключ-значення», документо-орієнтований, графовий та BigTable.

Бази даних по прикладу «ключ-значення». Такий тип структурування в базі даних є найпростішим, так як його можна вважати асоціативним, тут для кожного значення є свій унікальний ключ. Ніякого зв’язку між значеннями немає, так само як і обмежень на кількість елементів, так я це залежить від обчислювальних можливостей. Це є причиною чому такий тип побудови бази даних є характерний для хмарних сервісів. Популярними прикладами системи управління базами даних (СУБД), які зазвичай використовуються даного типу є: MemcacheDB, BerkeleyDB, Amazon DynamoDB, Riak.

Документо-орієнтований тип бази даних представляє собою систему яка ієрархічно зберігає дані, такий тип представлення має структуру дерева. Фактично така система є більш складною версією «ключ-значення». Вона як і дерево починається з кореневого вузла та маже містити декілька як внутрішніх так і зовнішніх вузлів. У ній можна здійснити доволі швидкий пошуку завдяки кінцевим даним, які зберігають зовнішні вузли в індекс бази при додаванні. Такий тип бази даних буде корисним при зберіганні впорядкованих даних та при умові, відсутності великої кількісті зв’язків між даними, а також немає

необхідності збирати статистику. Прикладами СУБД даного типу є: eXist, Couchbase, CouchDB, MarkLogic, MongoDB.

База даних графові. Вона найкраще покаже себе в реалізації проектів в яких передбачена графова структура даних. Для завдань такого характеру вони будуть значно кращими за реляційні бази даних. Відомими графовими СУБД є: FlockDB, Giraph, OrientDB, ArangoDB, HyperGraphDB, Neo4j.

BigTable, такі сховища мають багато спільного з документо-орієнтованими системами. В даній системі дані впорядковані у вигляді розрідженої матриці, а в якості ключів використовуються рядки та стовпці. Зазвичай даний тип бази даних використовується у веб-індустрії для або ж в інших завдання де будуть використовуватися великі обсяги даних. Прикладами таких СУБД є: Hypertable, SimpleDB, Cassandra, HBase.

MongoDB має можливість зберігання даних форматі JSON, а також володіє досить гнучкою мовою яка дозволяє формулювати запити, дає можливість для різних атрибутів створювати індекси, забезпечує зберігання великих бінарних об’єктів, яке є досить ефективним, може працювати відповідно до парадигми Map/Reduce, має можливість створити відмово стійку конфігурацію та підтримує реплікацію. Ще дана база даних має інструменти які забезпечують шардінг, які є вбудованими. Шардінг – це спосіб зберігання даних на різних серверах який передбачає розподіл різних наборів даних на основі певного ключа по різноманітним серверам. Модель, яка використовується для зберігання даних, а саме JSON дозволяє простіше кодувати та керувати за рахунок «без схемного стилю», який тут використовується, а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії.

Зіставивши усі переваги та недоліки різних типів баз даних було прийняте рішення реалізувати проект використовуючи нереляційну базу даних mongoDB. Це допоможе уникнути детального проектування таблиць та зв’язків між ними.

Основуючись на діаграмі варіантів використання яка була створена раніше та представлена на рисунку 1.6, можна виокремити наступні дані, які

потребують збереження, а саме: користувача, створені оголошення, знаючи потреби для бази даних можемо створити дві відповідні колекції під назвами «User» та «Post».

User – це колекція користувачів які зареєструвалися у програмній системі, та має наступні поля:

– IdUser – унікальний номер користувача;

– Role – це поле визначає роль користувача у системі, адміністратор або ж звичайний користувач;

– Avatar – шлях до фотографії профіля користувача;

– FirstName – ім’я користувача;

– LastName – прізвище користувача;

– Email – електронний адрес користувача, на який зареєстрований обліковий запис;

– Password_hash – хеш паролю користувача, який використовується для входу в обліковий запис;

– Date – дата реєстрації користувача.

Post – дана колекція позначає оголошення яке опубліковане користувачем про продаж авто та містить такі поля:

– IdPost – унікальний номер;

– IdUser – ідентифікатор користувача який створив оголошення;

– Images – масив з фотографіями авто;

– Posted_date – дата публікації оголошення;

– Title – заголовок оголошення;

– Price – ціна яку вказав користувач у своєму оголошені;

– Description – опис товару;

– ContactEmail – електрона адреса продавця

– ContactPhone – мобільний телефон продавця, як спосіб зв’язатися покупцеві з продавцем;

– Category – категорія представленого транспортного засобу;

– Make – марка авто;

– Year – рік виготовлення транспортного засобу;

– Gas – вид палива яке споживає авто;

– Model – модель транспортного засобу;

– Type – тип кузова;

– Transmission – варіант коробки передач.

Ми отримали велику кількість полів, усі вони будуть задіяні та є необхідні для того щоб додаток працював правильно. Можна помітити, що у обох колекціях повторюється ідентифікатор IdUser. Він допоможе зв'язати оголошення з користувачем, який його створив. Облікові дані користувачів будуть зберігатися у колекції User, а у Post розміщується повна інформація про оголошення, яке планує розмістити користувач. Отже, основуючись на логічній моделі, можна приступити до проектування такої ж фізичної моделі.

2.3 Проектування серверної частини Інтернет-платформи

Серверна частина Інтернет-платформи є однією з складових частин, так як без неї буде працювати виключно візуальна частина, яка не матиме ніякого функціоналу. Тому потрібно розробити серверну частину. Вона має пов'язати між собою клієнтську частину Інтернет-платформи та базу даних.

Серверна частина отримує запити від клієнта, обробляє їх, за потреби вносить зміни в базу даних або ж просто отримує інформацію звіти, та відправляє відповідь для відображення у користувача. Для керування запитами та передавання інформації буде використовуватися прикладний програмний інтерфейс REST API.

REST API розшифровується як REpresentational State Transfer, що перекладається як передача стану представлення, а означає він стиль архітектури який дозволяє взаємодіяти компонентів в розподіленому додатку

мережі. REST представляє собою набір обмежень, які є узгодженими та враховуються у процесі проектування розподіленої гіпермедіа-системи, а це приводить до наслідків, які представляються у підвищеній продуктивності і спрощені архітектури. Компоненти REST маю взаємодію на подобі взаємодії клієнтів і серверів. Цей термін був введений одним із засновників протоколу HTTP. Тому особливістю даного прикладного інтерфейсу є найбільш ефективне використання протоколу HTTP. REST тримає фокус на ресурсах, які використовуються і на тому наскільки ефективно можна виконати операції з ними завдяки використанню HTTP, так як він є базовим для REST.

Для отримання інформації з сервера використовуються POST та GET запити. Відрізняються вони тим що POST запит надсилається на сервер з метою збереження даних. Дані включаються в тіло запита, а сервер їх зберігає у себе в базі даних. Такий запит використовується при заповнені різноманітної інформації зі сторони клієнта, наприклад, при реєстрації нового клієнта або ж при створенні ним нового оголошення. На відміну від даного метода, метод HTTP GET призначений для отримання різноманітної інформації від сервера, яку вимагає користувач. До прикладу при відкриті головної сторінки надсилається запит для отримання оголошень, які вже є створенні іншими користувачами, а сервер в свою чергу формує відповідь та відправляє її до клієнта. Важливим є формат у якому подаються ці запити та відповіді. Ці формати є визначеними завдяки стандартному протоколу HTTP

2.4 Проектування клієнтської частини Інтернет-платформи

Усі Інтернет-платформі мають загально прийняті шаблони структури які складаються з декількох частин та є актуальними для усіх сторінок Інтернет-платформи. Перед появою HTML5 уся розмітка здійснювалася за допомогою елементів , до яких привласнювалися класи та ідентифікатори. Це

допомогло структуровано та наочно розмістити усі елементи, а саме завдяки ним HTML розміщувалися нижні та верхні колонтитули, бокові панелі, різні навігаційні панелі та багато чого іншого.

З приходом нового стандарту HTML5, прийшли й нові елементи для структурування, угрупування різного контенту та розмітки текстового вмісту. Нові елементи допомогли покращити структуру Інтернет-платформи. Елементи можна стилізувати на свій розсуд так як не задано жодних правил для відображення зовнішнього виду.

Для сучасних браузерів не завжди є зрозумілими елементи HTML5. Задавши попросту ширину та висоту таких елементів вони будуть неправильно відображатися у браузері. Для вирішення даної проблеми потрібно вказати для усіх HTML5-елементів які використовуються {display: block;}.

Загалом веб-додаток має три основні частини, такі як:

– хедер (шапка) – це верхня частина сторінки, який оформляється в тегах header. Це перша та одна з головних частин сайту, так я при відвідувані сайту користувач в першу чергу звертає на це увагу, а також грамотне оформлення цієї частини є головним тому, що у цій часині розміщуються назва сайту, коротка інформація, кнопки авторизації та реєстрації, також можуть розмішуватися кнопки навігації або ж вони будуть винесені в окрему частину;

– навігація по сторінці, яка зазвичай вона розміщується під хедером, або ж у самому хедері, а оформлення головної навігації на сторінці має розміщуватися у тегу nav. Також варто розміщувати навігацію елементами списку, це буде вважатися гарною практикою;

– основна частина, в якій оформляється головна основна частина сайту, яка займає найбільше місця на сторінці. Вся інформація яка цікава та необхідна для користувача. Текс повинен знаходиться в тегу article. Цей тег зберігає в собі блок з контентом та може використовуватися окремо від контексту сторінки в іншому місці сторінки;

– сайдбари (sidebar) або ж колонки з віджетами. Сайдбар це бічна панель яка відображає графічні елементи для керування програмним забезпеченням або для відображення різної додаткової інформації. Такий елемент знаходиться по праву або ліву частину сайту. Він не є обов'язковим але доволі інформативний та часто зустрічається на різноманітних сайтах. Для кожного окремого елемента сайдбара використовується блок aside;

– footer або ж підвал сайту. Він представляє собою нижній колонтитул. Зазвичай розміщується в низу сторінки та володіє інформацією про автора або ж інформацію про авторське право, дата публікації, контактна інформація та різна правова інформація.

HTML5 володіє багатьма іншими тегами, які допомагають розмістити потрібну інформацію на сторінці, перелічені вище теги – це тільки представлення основної структури сайту. Різноманітні теги допомагають освіжити структуру сайту та й це робить більш дружелюбнішим його до пошукових двигунів, в результаті цього сайт буде мати більш високий рейтинг при пошуковій видачі результатів сайтів.

Щоб представити загальний вигляд структури сайту з усіма частинами було створено рисунок 2.1.

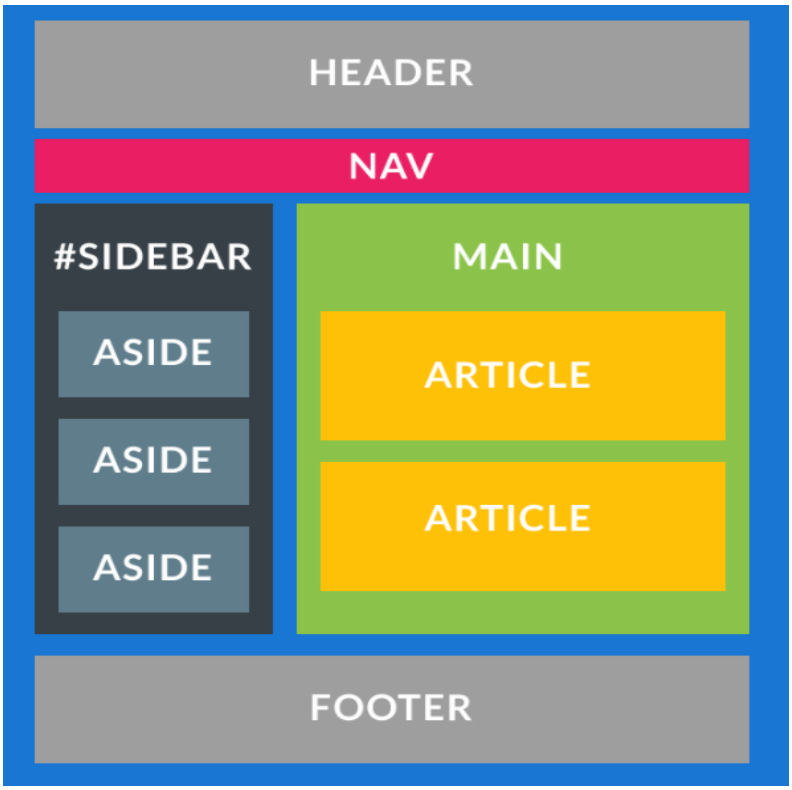


Рисунок 2.1 – Схематична структура сторінки HTML 5

2.5 Створення макетів Інтернет-платформи та дизайн

Щоб створити макет дизайну скористаємося методом прототипізації. Прототипізація – це один із етапів розробки програмного забезпечення, який передбачає створення макету або ж прототипу програми. Це дозволяє перевірити на придатність концепції, які були запропоновані для застосування, а також різноманітні інші рішення. Також дає можливість представити програму замовнику на початковій стадії розробки

Завдяки зворотному зв'язку від потенційних користувачів та замовників ми маємо можливість виправити помилки та нюанси на початкових етапах, коли це не є критичними змінами для проекту та не займають великої кількості часу та фінансових затрат. Прототипування має на меті перевірку концепції, а також зменшення ризиків на етапі розробки програмного забезпечення.

Для початку опишемо хедер, який є початковим на сторінці. На ньому повинно розміщуватися логотип та кнопка авторизації (рисунок 2.2). Логотип

одночасно буде відігравати роль кнопки повернення на головну сторінку. У авторизованого користувача трішки змінюються функції керування у хедері. Замість кнопки авторизації з'являється ім'я та прізвище користувача котрий авторизувався, а при натисненні на цю кнопку з'являється випадаюче меню з такими кнопками як:

– профіль, при натисненні на який у користувача відкриється вікно з можливістю перегляду інформації та її зміни;

– добавити оголошення, тут користувач зможе створити нове оголошення для публікації його на платформі;

– налаштування, тут користувач зможе змінити пароль та видалити обліковий запис;

– вийти, а ця кнопка допоможе користувач змінити обліковий запис або взагалі вийти.

Макет розробленого хедера для авторизованого користувача представлений нижче на рисунку 2.3.

Рисуок 2.2 – Хедер для не авторизованого користувача

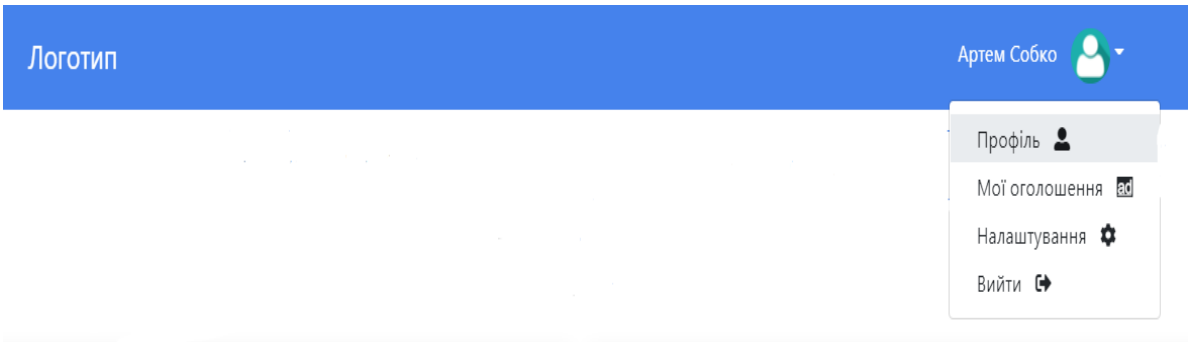


Рисунок 2.3 – Хедер для авторизованого користувача

Також змодельований макет футера з переліком посилань зображених на ньому, а саме:

– «зв’яжаться з нами» – дозволяє написати відгук або пропозицію адміністрації Інтернет-платформи.

– «про нас» – тут розповідається інформація про розробник платформи та короткі юридичні відомості.

– посилання на соціальні сторінки компанії, а саме Facebook, Instagram та Twitter, а на них з часом буде з’являтися інформація про нововведення з оновленням Інтернет-платформи.

Макет футера представлений на рисунку 2.4.

Рисунок 2.4 – Футер Інтернет-платформи

Далі спроектуємо контейнер який буде розміщувати основну інформацію. Це буде контейнер з оголошеннями. Тут повинні розміщуватися наступні елементи керування:

– Карточки на яких буде коротка інформація про транспортний засіб а саме його ціна та назва, а також його фотографія.

– Повинна розмішуватися форма для фільтрування оголошення, які відображаються на сторінці.

– Перелік сторінок з оголошеннями та можливість переходити на наступні сторінки та повертатися на попередні.

Макет контейнера з оголошеннями знаходиться на рисунку 2.5.

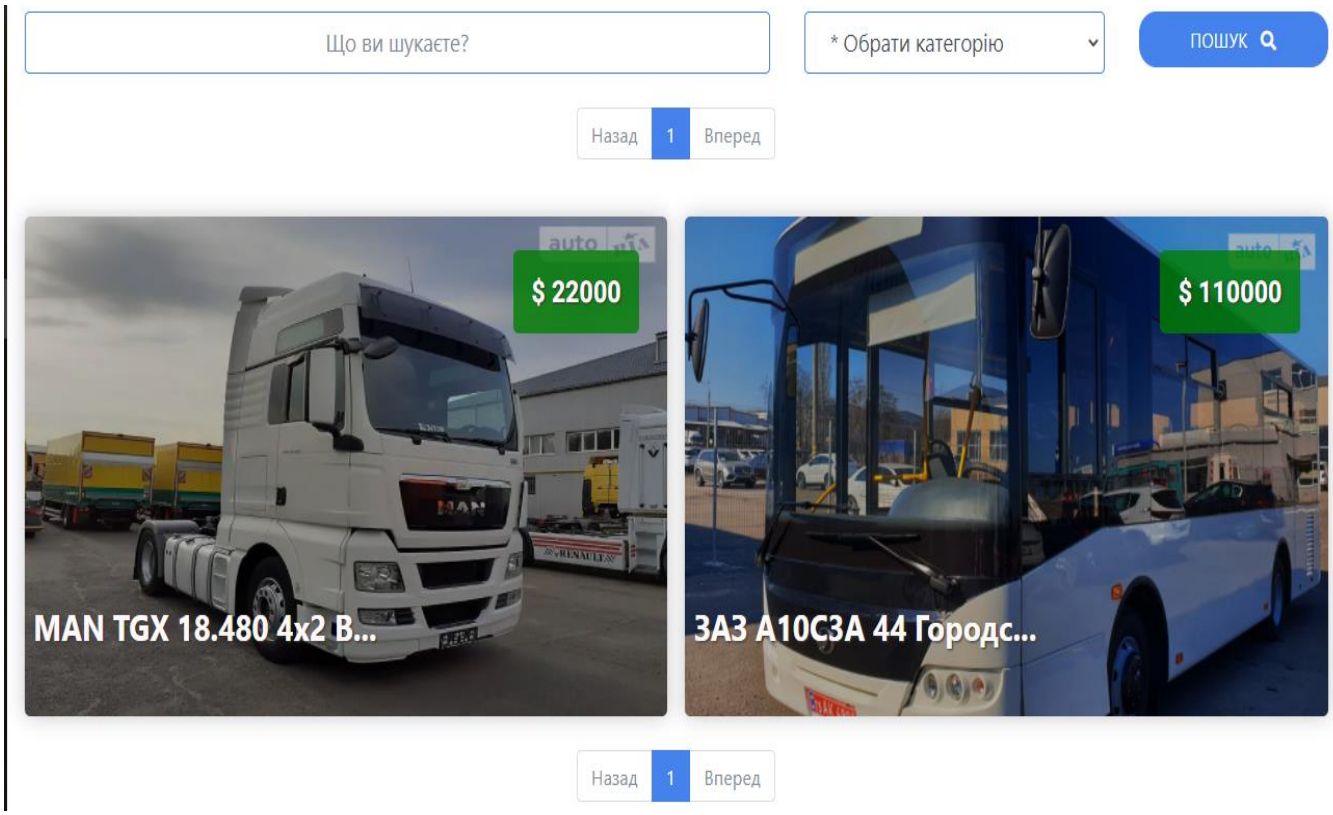


Рисунок 2.5 – Макет основного контейнеру з переліком оголошень

За основу макету для сторінки з переліком власних оголошень у клієнтському кабінеті було використано макет який використовується для відображення списку оголошень на головній сторінці (рисунок 2.4). Але даний макет потребує певних змін, замість вибору фільтру списку категорій у користувача буде можливість відфільтрувати оголошення які допущені до публікації на порталі, відхилені або ще розглядаються адміністрацією. Макет розміщення власних оголошень представлений на рисунку 2.6.

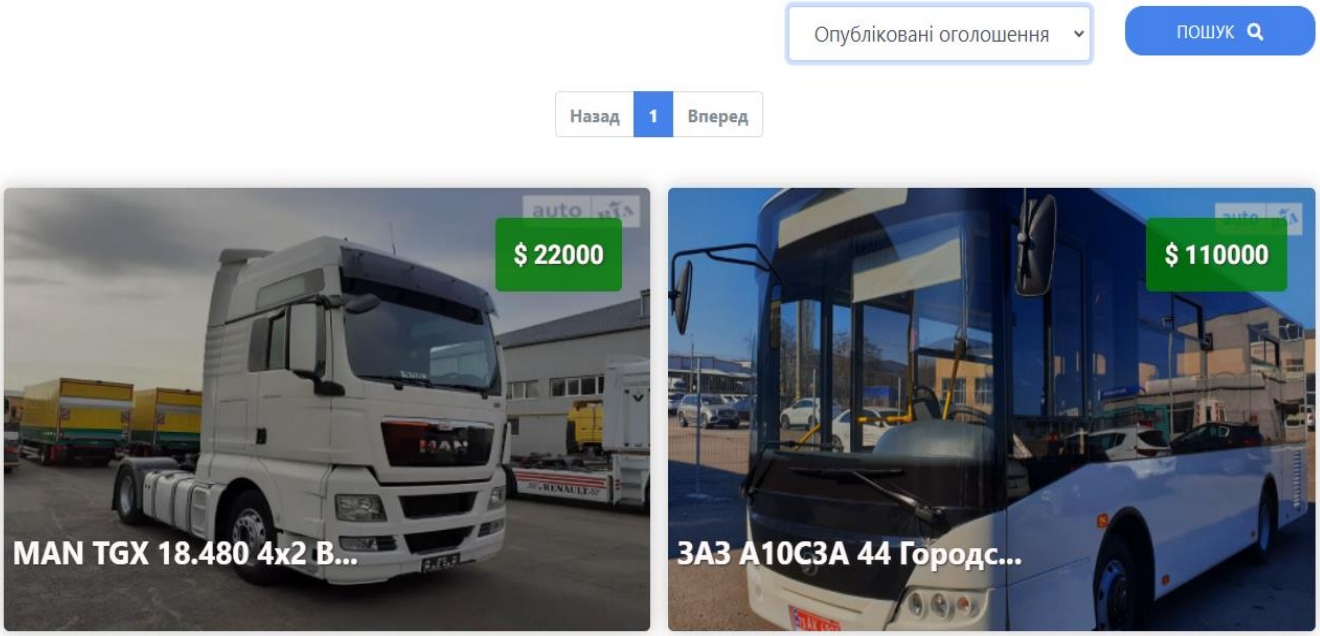


Рисунок 2.6 – Шаблон списку оголошень в персональному кабінеті

Також потрібно змоделювати шаблон для самого оголошення де буде розміщуватися конкретна інформація по конкретному транспортному засобі. Також тут має бути представлений наступний перелік інформації:

– фотографії;

– модель транспортного засобу;

– тип кузова;

– рік випуску;

– тип палива яке використовує транспортний засіб;

– марка транспортного засобу;

– контактна інформація для зв’язку з продавцем, така як номер телефону, електрона пошта та як звати продавця.

Уся ця інформація допоможе клієнту підібрати транспортний засіб, який відповідає заданим критеріям, а також з легкістю зв’язатись з продавцем. Макет даної сторінки представлений на рисунку 2.7.

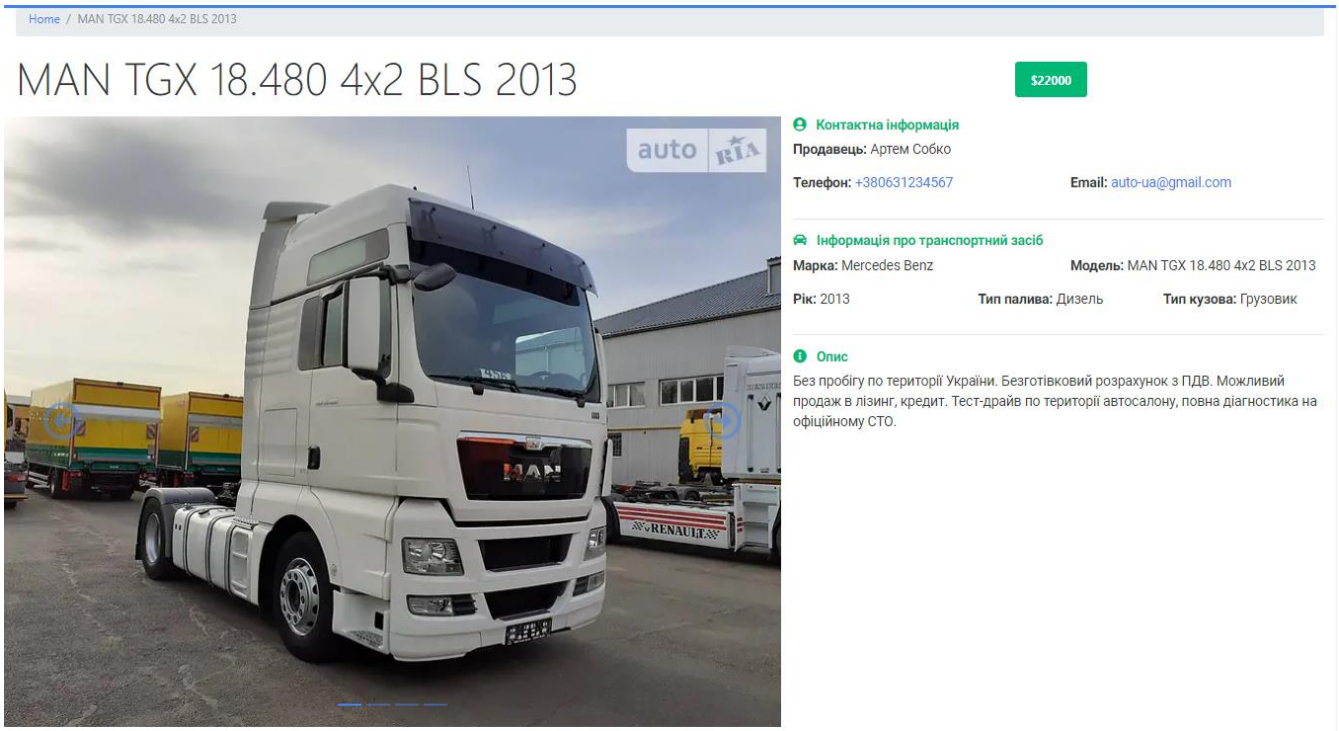


Рисунок 2.7 – Макет персональної карточки оголошення

2.6 Аналіз та вибір технологій і методів реалізації Інтернет-платформи

Для реалізації даного проекту було вибрано стек MERN. Існує багато різноманітних стеків і всі вони по суті є різними способами для досягнення однакової мети. При створені full-stack додатків потрібно передбачити в них створення клієнтської частини з якою буде взаємодіяти користувач, а ця клієнтська частина повинна взаємодіяти з серверною частиною та базою даних. Все це зробить даний процес більш простим та контрольованим. Можна сказати що стек - це певний набір технологій для вирішення поставленої задачі. Сьогодні існує доволі багато різноманітних стеків, одні з них більше популярні за інших, а одним з популярних і являється стек MERN. Він включає в себе:

– MongoDB;
– Express.js;
– React.js;
– Node.js.

MERN стек доволі похожий на стек MEAN, за виключенням того що використовується React.js замість Angular.js.

MongoDB – представляє собою систему керування нереляційною базою даних. Перевагами даної бази даних є те що вона використовується швидкими та масштабованими системами, а також оперує даними за типом ключ-значення та реляційними системами керування базами даних, має зручні та функціональні форми запитів. Також має можливість зберігати дані в JSON-подібному форматі.

Express.js – це гнучкий та мінімалістичний фреймворк який застосовується в парі з Node.js. він надає великий набір функцій які можна застосувати як у веб-додатках так і у мобільних додатках.

Node.js – це платформа яка надає можливість використовувати JavaScript не тільки для обробки даних на стороні клієнта, так як це було раніше, а й на стороні сервера, відправивши користувачу іще готовий результат виконання запиту. Завдяки Node.js JavaScript мова стала мовою загального використання та тепер налічує велику спільноту розробників.

JavaScript в свою чергу являється об’єктно-орієнтованою мовою програмування. Також вона є прототипною мовою програмування з динамічною типізацією. Найбільш популярна для використання при створені сценаріїв на веб сторінках зі сторони користувача, але завдяки Node.js стала популярною також і при програмуванні серверної частини додатку. Прототипна архітектуна не є єдиною для цієї мови програмування, частково вона може підтримувати й інші парадигми, такі як імперативна та функціональна. Але і має деякі архітектурні властивості такі як автоматичне кешування пам’яттю функції та об’єкти першого класу, слабка та динамічна типізація.

React.js – представляє собою відкриту бібліотеку JavaScript, яка спеціалізується на створені інтерфейсів користувача. Є корисною при використані її для часткового оновлення вмісту веб-додатка. Особливо актуальною така проблема є коли розробляються односторінкові додатки. Дана бібліотека дає можливість розробникам при створені великих веб-додатків

оновлювати з часом дані на сторінці без перезавантаження усієї сторінки. Метою цієї бібліотеки є створювати великі, прості, швидкі та масштабовані додатки. React дозволяє обробити лише користувацький інтерфейс у додатках. Таким чином він працює відповідно до шаблону модель-вид-контролер (MVC).

Модель-вид-контролер (MVC, Model-view-controller) – це шаблон архітектурної побудови програмного забезпечення, який застосовується при його розробці та проектуванні. У цьому шаблоні передбачається поділ на 3 частини які розміщуються взаємопов’язані між собою. Такими частинами є:

- модель (Model);
- вид (View);
- контролер (Controller).

Модель даних, це головний елемент шаблону MVC, відповідає за поведінку в додатку та модель збереження даних, поведінка не залежить від інтерфейсу користувача. Застосування моделі підлягає керування даними, логічними процесами та правилами застосування.

Вид відповідає за представлення інформації, яка відображається на виході, це може бути різноманітні діаграми та графіка. Для одночасного відображення може використовуватися не тільки один компонент Виду, їх може бути декілька і можуть вони відображати одні і ту саму інформацію. Має декілька областей, які є взаємопов’язаними.

Контролер є відповідальним за отримання вхідних даних в результаті дії користувача та повинен переробити їх в команди, які будуть надіслані для моделям та виглядам.

Також варто відзначити особливості бібліотеки React. В рендерер компоненту передаються властивості у вигляді HTML тегу. Це означає що змінювати властивості на пряму – компонент не може, проте може їх змінити завдяки функції callback.

У React є підтримка віртуального DOM, завдяки цьому можна не розраховувати виключно на DOM браузера. Це створено для того щоб найбільш

ефективно обновити DOM браузера, так як бібліотека шляхом порівняння актуального DOM з попередньо збереженою версією цього віртуального DOM визначає які частини змінилися. Для програміста створюється вигляд що сторінка оновлюється уся, але це не так, тому що оновлюються виключно деякі компоненти на сторінці, такі які вирішить обновити сама бібліотека.

DOM (Document Object Model) – це специфікація прикладного програмного інтерфейсу, яка використовується при роботі із структурованими документами. Ухвалюють та визначають цю специфікацію консорціум W3C. Є раціональним використовувати модель DOM лише для додатків в яких потрібно декілька разів звертатись до елемента в документі у будь-якому порядку. Це зумовлено тим що зміст усього документу зберігається і аналізується в комп'ютерній пам'яті, а його структура є представленою у вигляді дерева. У випадку ж коли доступ до пам'яті повинен бути одноразовим або ж послідовним рекомендується застосовувати послідовну модель роботи для структурованих даних, це зменшить час переробки а також обсяги комп'ютерної пам'яті які необхідні для цього.

Компоненти, які зазвичай використовуються в React, написані на JSX. Компілюючи код, який написаний на JSX, він перетворюється на виклики методів з бібліотеки React. А також при розробці в програмістів є можливість писати на чистому JavaScript.

Рендеринг HTML коду в браузері це лише одна із можливостей використання React. До прикладу, Facebook рендерить свої динамічні графіки в тегах <canvas>.

Для управління базою даних буде використовуватися Robo 3T. Robo 3T – це професійна багатоплатформний інструмент, який використовується для управління базою даних MongoDB з відкритим вихідним кодом. Також в нього вбудований JavaScript движок який підтримує MongoDB. На відміну від більшості інших інструментів призначеного для користувацького інтерфейсу адміністратора MongoDB, Robo 3T вбудовує фактичну оболонку Mongo в

інтерфейс з вкладками з доступом до командного рядка оболонки і взаємодією з графічним інтерфейсом.

Також при побудові frontend частини даного програмного проекту як і раніше зазначалося буде застосовуватись така мова розмітки як HTML, а також буде використовуватися скриптова метамова SASS, для створення привабливого стилю сторінки.

HTML (HyperText Markup Language) – це мова для розмітки яка оперує гіпертекстом при створені веб-додатку. З веб-сервера або ж з локальної пам'яті передаються HTML-документи у браузер, який обробляє їх та відображує для користувача. Це самий базовий блок, який застосовується на веб-сторінці. Завдяки ньому можна визначити структуру веб-контенту та вміст. Зазвичай разом з ним і застосовується другі різні технологій, JavaScript для функціональності додатку, про який раніше вже згадувалось та й для зовнішнього вигляду CSS або в нашому випадку SASS.

CSS (Cascading Style Sheets) – перекладається як «каскадні таблиці стилів» та означає своєрідний список інструкцій, який використовує браузер. В них позначається що та як має відобразити елементи браузер. Елементами вважають теги HTML та XHTML, а саме їх вміст. Майже не існують сайтів створених без застосування CSS. Ідеєю CSS закладається в тому щоб відокремити основний текст та вміст сторінки від дизайну оформлення документу.

При написані великих таблиць стилів стає важко її обслужити та пригадати де знаходиться потрібний елемент. Для цього потрібно використовувати препроцесор, який спростить задачу. А SASS володіє функціями які є недоступні для звичайного CSS, до прикладу: різні змінні, наслідування, вкладання та і так багато різноманітних приємних речей з'явилися в і так вже зручному CSS. Препроцесор обробляє SASS файл та зберігає його як звичайний CSS, для того шо можна було використовувати його на любому сайті. Після встановлення SASS, використовуючи термінал можна компілювати файл Sass в CSS завдяки команді Sass. Для цього необхідно прописати для Sass де

знаходиться файл Sass та в який файл CSS його скомпілювати. До прикладу після виконання команди sass input.scss output.css виконаної завдяки терміналу, буде повідомлено Sass де взяти вхідний файл input.scss, та те що вихідний результат повинен розміщуватись аналогічно у файлі output.css. Завдяки змінам можна зберігати інформацію на протязі створення усього проекту з цифрами, стеками шрифтів або ж інші значення CSS. А передача змінних здійснюється з використанням символу долара - $. А завдяки вложеній ієрархії можна зробити привабливий та зрозумілий код. Він буде відповідати візуальній ієрархії HTML. Але головне не перебільшувати з кількістю вложених деталей, так як це навпаки, погіршить читабельність коду.

Отже, в цьому розділі було проаналізовано різні архітектури для створення веб-додатків, а також вибрана архітектура для розроблюваної Інтернет-платформи, також описана структура моделі бази даних, спроектовано серверну та клієнтську частину. Для зручної розробки Інтернет-платформи були створені макети дизайну. Проаналізувавши різні технології реалізації було вибрано найбільш оптимальну для реалізації даного проекту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Розробивши різноманітні моделі у другому розділі, включаючи модель бази даних, можна приступити до її реалізації. Для реалізації даного дипломного проекту буде застосовуватися популярний редактор коду Visual Studio Code. Він має усі необхідні функції для зручної розробки даного програмного проекту, так як спеціалізується в більшості для веб-додатків. Visual Studio Code надається на безкоштовній основі та підтримуються усіма операційними системами Windows, Linux і OS X.

Так як зазначалося раніше ми використовуємо бібліотеку React. При підключені її автоматично створюється шаблон привітання. Перш за все нам необхідно створити базу даних та первинно її налаштувати. MongoDB – це вже і є система керування базою даних, для зручного її використання скористуємось графічною оболонкою, яка зробить процес роботи з базою даних більш привабливим та зручнішим. Існують різні графічні інтерфейси для mongoDB, такі як UMongo, NoSQLBooster, Robo 3T та інші. Останній з яких й буде використовуватись. Я вважаю він є найбільш привабливішим та зручнішим. Створення бази даних відбувається вручну у графічній системі керування базами даних Robo 3T. Після чого ми зможе у проекті до неї підключитись. Нижче представлений фрагмент коду використовуючи який буде реалізовано підключення до бази даних:

```
Mongoose

      .connect(mongoURI, { useNewUrlParser: true })
      .then((bd) => {
            console.log('Підключення до бази даних');
            app.listen(PORT, () => console.log(`Сервер запущений на порту
      ${PORT}`)); })

      .catch((err) => console.error(err));
```

3.2 Розробка програмних модулів

Проект складається з модулів, які є взаємозв'язані. Частина цих модулів була створена при встановлені бібліотеки React, а частина добавлена власноруч. Одним із моделей є модуль перевірки користувача, іншими словами – модуль валідації. Функціями якого є перевірка користувача чи він є зареєстрованим в системі, а у разі успішної авторизації надати доступ до усіх функцій сайту які доступні авторизованому користувачу. Фрагмент коду перевірки користувача представлений нижче:

```
module.exports = (req, rest, nexts) => {
  let {emails, passwords} = req.body
  let errors = {}
  Email = !isEmpty(email) ? email: ''
  Password = !isEmpty(password) ? password: ''
  //перевірка електронної пошти
  if (Validators.isEmpty(email)) {
    error.emails = "Введіть електрону пошту"
  } else if (!Validators.isEmail(email)) {
    error.emails = "Невірний формат пошти"
      }
  //перевірка пароля
  If (Validators.isEmpty(password)) {
    error.passwords = "Введіть пароль"
  } else if (!Validators.isLength(passwords, {min:7, max:32})){
    error.passwords = "Пароль не вірний"
 }
  if (!isEmpty(errors)) {
    return res.status(400).json(errors)
 }
  req.errors = error;
  next()}
```

Також в розроблюваній системі існує дві моделі які описуються данні отриманні від користувачів до нереляційної бази даних. Опис складається з переліку полів які характеризують об'єкт по певним критеріям, надалі цей опис буде використовувати для збереження в базу даних та для відображення на сайті. Також завдяки даним характеристикам користувачам буде зручніше найти потрібний товар використовуючи фільтр пошуку. Нижче наведено фрагмент

коду моделі, яка описує оголошення користувача, а у останній стрічці вказується, що об’єкт є моделлю бази даних:

```
const PostsModel = new Model ({
 holder: { type: Schema.Types.IdObject, ref:'user', required: true },
      imageses: [],
      title: { type: String, required: true },
cost: { type: Number, required: true },
      description: { type: String, required: true },
      categor: { type: String, required: true },
      Phone: { type: String, required: true },
      Email : { type: String, required: true },
      model: { type: String },
      makes: { type: String },
      gas: { type: String },
      years: { type: String },
date: { type: Date, default: new Date() }
});

module.exports = Post = mongoose.models('post', PostSchema);
```

Також нижче представлений фрагмент програмного коду для моделі користувача, який дозволяє описати інформацію, яка характеризує конкретно кожного користувача:

```
const UserModel = new Model({
      NameFirst: { type: String, required: true },
      NameLast: { type: String, required: true },
Verified: {type: Boolean},
      Phone: { type: String },
      Email: { type: String, required: true, unique: true },
      Password: { type: String, required: true },
      TokenReset: { type: String },
      Avatar: {
            type: String,
            default: 'avatar/avatar.jpg'
      },
      creation_date: { type: Date, default: new Date() },
}
      );
```

Створивши модель об’єкта яка заповнювалась користувачем у відповідній формі, необхідно записати отримані данні в базу даних. За це відповідають

контролери, яких також два. Для окремої моделі використовується свій контролер. Нижче наведено фрагмент коду метода, який відповідає за збереження даних про нового користувача:

```
exports. ControllersRegister = async (req, res) => {
      try {
            const { errors } = req;
            const users = await User.findOne({ email: req.body.email });
            if (users) {
                  errors.emails = 'Email вже зареєстрований';
                  return res.status(409).json(errors);
            } else {
            Const verifcationToken = await crypto.randomByte(32)
.toString('hex');

                  const expVerificationToken = Date.now() + 3500000;
                  console.log(verificationToken);

                  const { firstName, lastName, email, password } = req.body;

                  const newUser = new User({
                        firstName,
                        lastName,
                        email,
                        password,
                        verifcationToken,
                        expVerifcationToken
                  });

                  bcrypt.genSalt(10, (err, salt) => {
                        bcrypt.hash(newUsers.passwords, salt, async (err, hash)
=> {
                              try {
                                    if (err) throw err;
                                    newUsers.passwords = hash;
                                    //register the user in the Database
                                    await newUser.save();
                                    sendVerifcationToken(messageData);
                                    res.status(201).json({ msg: 'OK' });
                              } catch (error) {
                                    res.status(500).json({ error: `User
registration failed see ${errors}` });
                              }
                        });
});
}
      } catch (err) {
            res.status(503).json({ error: err.toString() });
      }};
```

Також є ще один з методів який відповідає за видалення користувача з системи. Після чого користувач більше не зможе потрапити в свій обліковий запис а лише створити новий. Також він автоматично видаляє усі публікації створені користувачем. Фрагмент код даного методу зображено нижче:

```
exports.DeleteAccountController = async (req, res) => {
	const { errors } = req;

	try {
		const { _id, password } = req.user;
		const { password: currentPassword } = req.body;
		const user = await User.findOne({ _id });

		if (!user) {
			errors.password = 'Unauthorized request';
			return res.status(401).json(errors);
		}
		const userPosts = await Post.find({ owner: _id });
		if (userPosts.length > 0) {
			//Merges the images path into a single Array
			const imagesURLs = [].concat.apply([], userPosts.map((post) =>
post.images));
			try {
				await imagesURLs.forEach((image) =>
fs.unlinkSync(image));

				await Post.deleteMany({ owner: _id });
			} catch (err) {
				errors.error = err;
				res.status(500).json(errors);
			}}

		const isMatch = await bcrypt.compare(currentPassword, password);
		if (!isMatch) {
			errors.password = 'Incorrect Password2';
			return res.status(401).json(errors);
		}
		if (user.avatar !== 'avatars/default.jpg') {
			fs.unlink(user.avatar, () => {});
		}

		await user.remove();
		res.status(200).json({ msg: 'User Deleted!' });
	} catch (err) {
		errors.error = err;
		res.status(500).json(errors);
	}
};
```

А також нижче представлено фрагмент коду контролера, який відповідає за видалення публікації користувача.

```
exports. ControllerDeletePost = async (req, res) => {
      const { errors } = req;
      try {
            const { _id: Iduser } = req.user;
            const { Idpost } = req.body;

            const posts = await Post.findOne({
                  id: Idpost,
                  owner: Iduser
            });

            const { image } = posts;

            if (images.length > 0) {
                  try {
                        image.forEach((images) => fs.unlinkSync(images));
                  }
catch (err) {
                        errors.errors = err;
                        res.status(600).json(error);
                  }}

            await posts.remove();

            res.status(200).json({ msg: 'Публікаця видалена' });
      } catch (errs) {
            error.errors = errs;
            res.status(500).json(error);
      }};
```

Також були створені контролери які відповідають за інші функції пов’язані з даними моделями такі, як:

– ControllerEditPassword () - відповідає за зміну пароля користувачем;

– ControllerUpdateUserProfile() – оновлення даних користувача у профілі;

– ControllerAllPosts() – відображає список усіх оголошень які існують у базі даних та є актуальними;

– ControllerCreatePost() – відповідає за створення нових оголошень;

– ControllerEditPost() – відповідає за редагування оголошень які уже були раніше створені;

– ControllerSinglePost() – даний метод повертає одне єдине оголошення.

Код реалізації усіх даних методів представлений у додатку Б, а також були створені ще наступні папки з файлами такі, як:

– createAd – тут розміщені файли які відповідають за візуальну форму для заповнення інформації про оголошення;

– editAd – у даній папці знаходяться файли з формами для редагування оголошень створених користувачем;

– listsItems – тут знаходиться список доступних категорій які з’являються при створені оголошення, а також при фільтруванні оголошення для його зручного пошуку;

– myAds – представлення де знаходяться форми для відображення оголошень створених користувачем;

– layout – папка яка розміщує у собі різноманітні представлення різних частин сайту, таких як футер, хедер та аватарку.

– profile – ця папка розміщує у собі файли, які створюють форму за допомогою якої користувач може налаштувати власний обліковий запис.

3.3 Керівництво користувача

Інструкція для користувача з використання Інтернет-платформи для продажу авто.

Перш за все потрібно відкрити браузер та перейти на відповідний сайт. На головній сторінці нас зустрінуть раніше створені оголошення користувачів. Вони відображаються у карточок на яких є деяка інформація, а саме це головне фото транспорту, його ціна та назва (рисунок 3.1).

Якщо звернути увагу на верхній правий кут можна замітити кнопку авторизації. Неавторизований користувач має не повний функціонал, для того щоб викласти власне оголошення, користувачу потрібно авторизуватися.

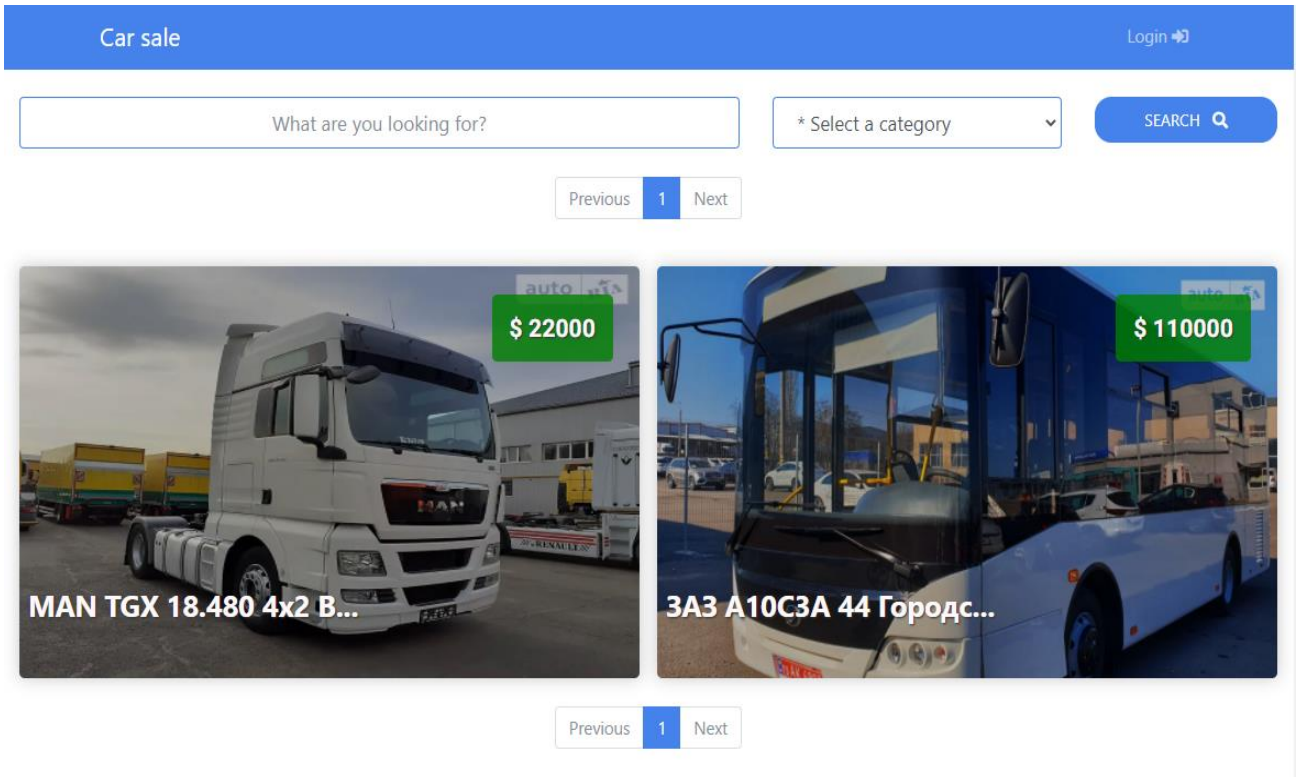


Рисунок 3.1 – Головна сторінка Інтернет-платформи

Форма для авторизації користувача представлена на рисунку 3.2, а у випадку якщо користувач ще й не зареєстрований, то йому потрібно зареєструватись, після чого він зможе авторизуватись. Також задля безпеки від спаму та не доброчесних користувачів – передбачено підтвердження облікового запису завдяки електронній пошті. На неї буде висланий лист з унікальним посиланням, перейшовши за яким обліковий запис користувача буде повністю підтверджений та він зможе користуватися усім функціоналом.

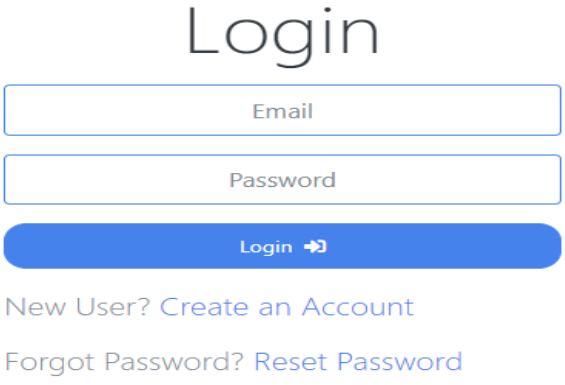


Рисунок 3.2 – Форма авторизації

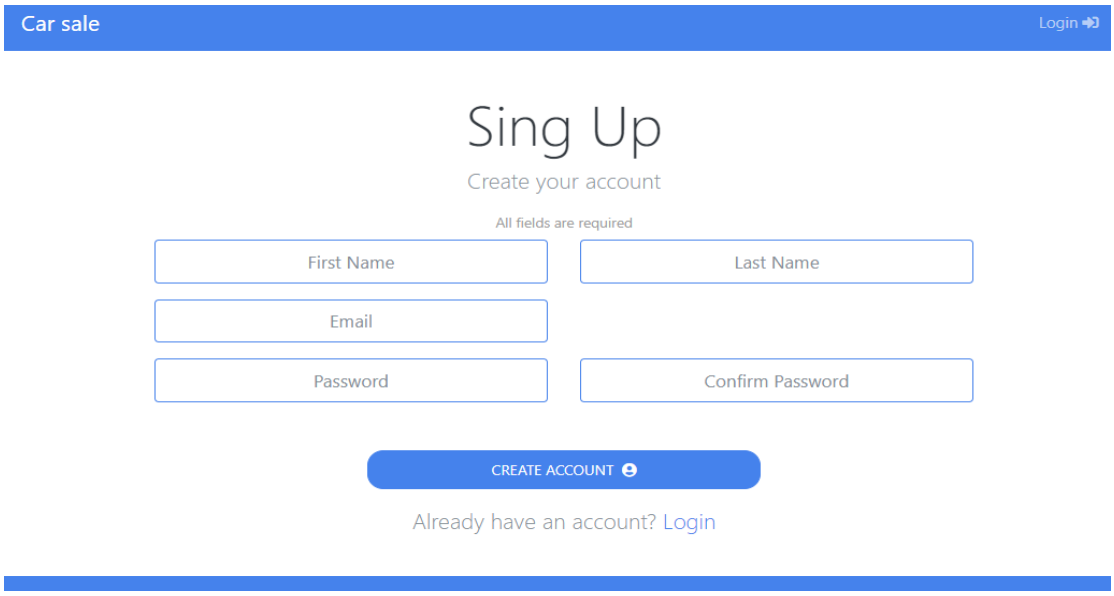


Рисунок 3.3 – Форма реєстрації

Також у випадку якщо користувач забув пароль, у нього є можливість його відновити. Для цього потрібно перейти на форму авторизації та натиснути відновити пароль. Далі потрібно електрону пошту, яка вказувалась при реєстрації. На неї буде надіслано посилання для відновлення паролю.

Після успішної реєстрації та авторизації, у користувача відкривається доступ до особистого кабінету (рисунок 3.4).

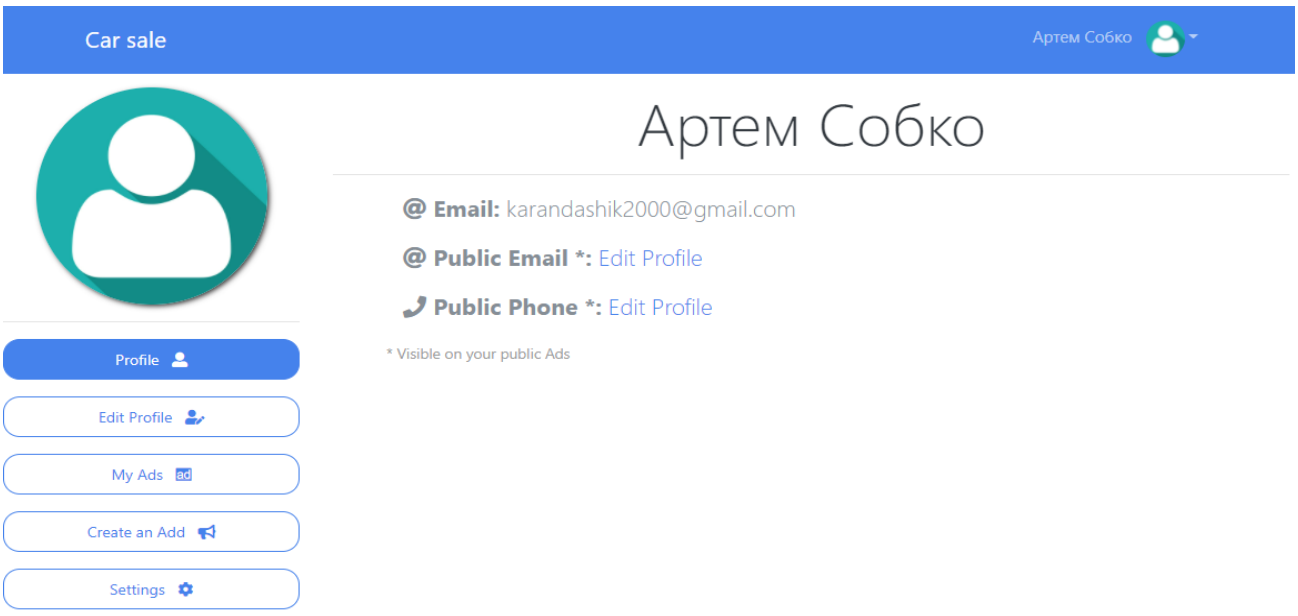


Рисунок 3.4 – Особистий кабінет користувача

Тут він може змінювати інформацію про себе, переглядати власні оголошення, створювати нові, змінити пароль або ж взагалі видалити свій обліковий запис.

Для створення нового оголошення достатньо перейти на вкладку створити оголошення. Там буде відображатися форма, яку потрібно буде заповнити щоб створити нове оголошення (рисунок 3.5). Ця форма включає перелік полів для заповнення, які характеризують транспортний засіб, спрямовані на покращення підбору потрібного товару для покупця. Також ця інформація буде використовуватися при фільтрації оголошень.

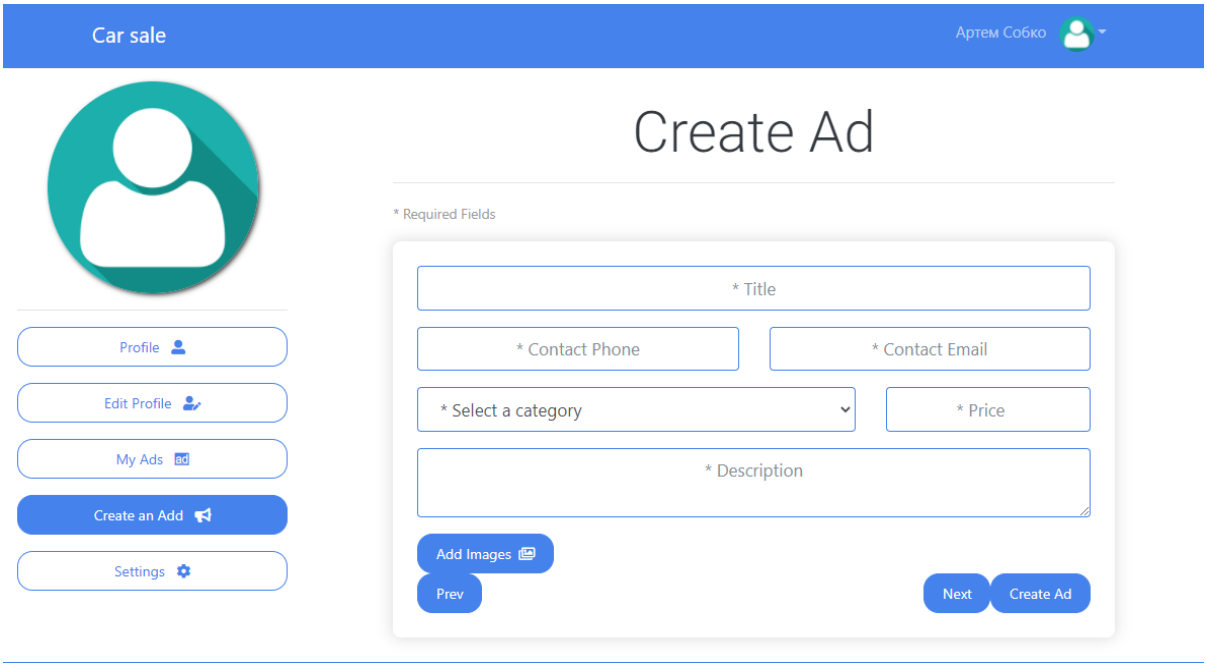


Рисунок 3.5 – Сторінка створення оголошення

Також, є наступна сторінка з більш детальною інформацією (рисунок 3.6), де користувач повинен буде вказати додаткові характеристики свого транспортного засобу. Його будуть розміщуватись на персональній сторінці кожного оголошення, та будуть доступними для усіх користувачів. По завершенню заповнення та натиснувши на кнопку створити оголошення, воно буде розмішене на головній сторінці сайту, та стане доступним для усіх користувачів Інтернет-платформи.

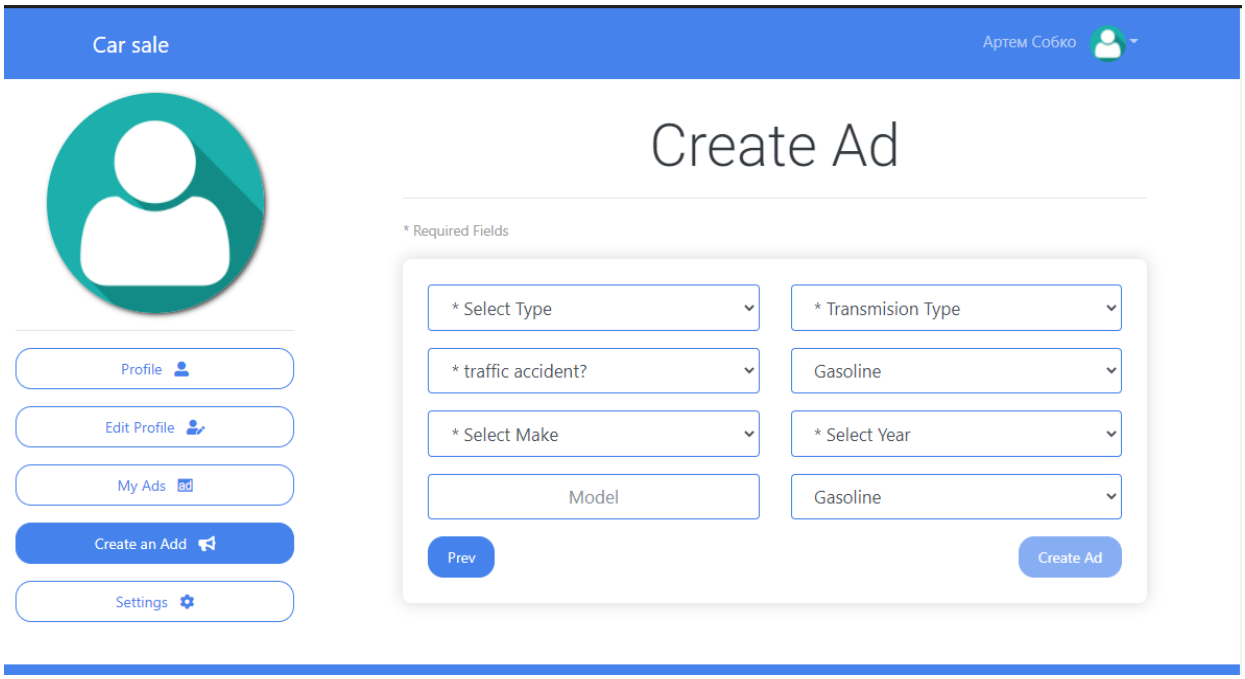


Рисунок 3.6 – Додаткова інформація про оголошення

Користувач, якого зацікавило оголошення, на головній сторінці може натиснути на карточку з оголошенням, та відкриється наступна сторінка представлена на рисунку 3.7.

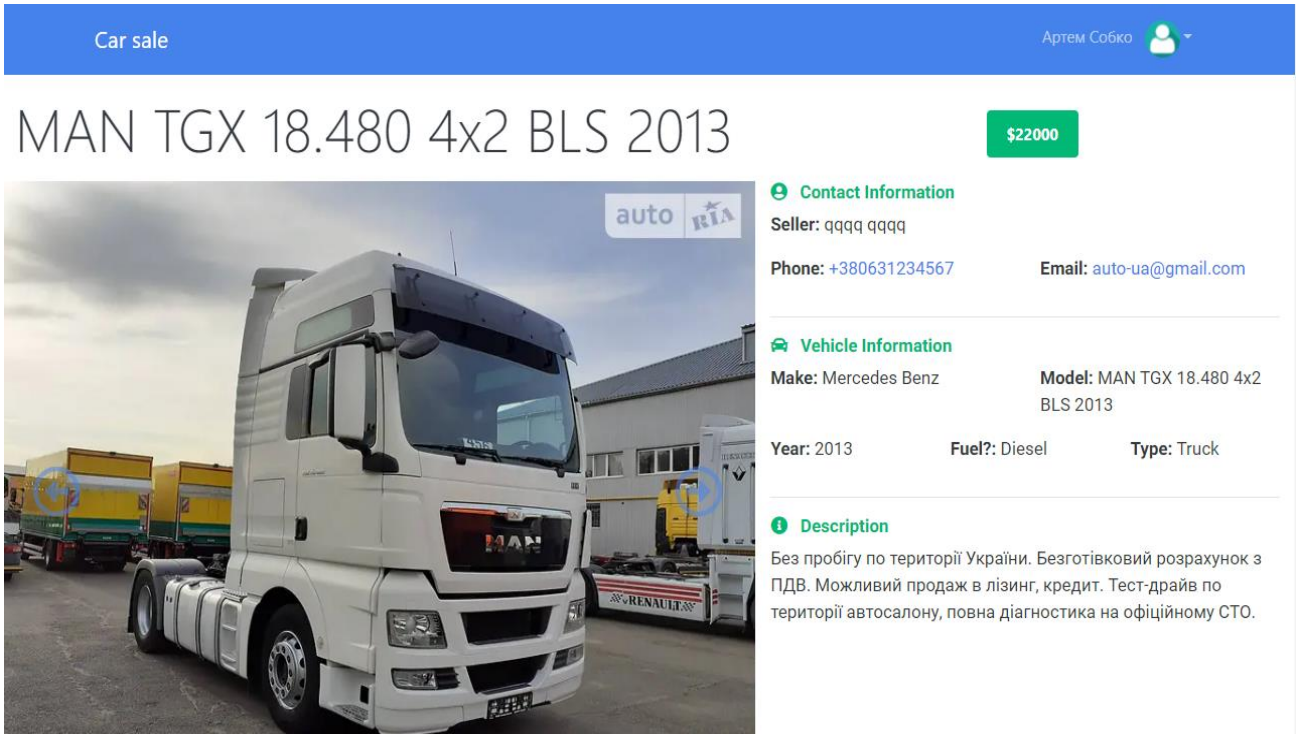


Рисунок 3.7 – Детальна інформація про оголошення

На відкритій сторінці користувач зможе ознайомитись з детальною інформацією про транспортний засіб, його технічним станом, технічними характеристиками, коротким описом від продавця, про особливості експлуатації або ж моментами з приводу переоформлення транспортного засобу та його контактною інформацією.

Внизу сторінки розміщена кнопка для зв'язку з адміністрацією (рисунок 3.8). Натиснувши на неї з'явиться відповідна сторінка з формою для заповнення, яка зображена на рисунку 3.9.

Рисунок 3.8 – Футер сторінки

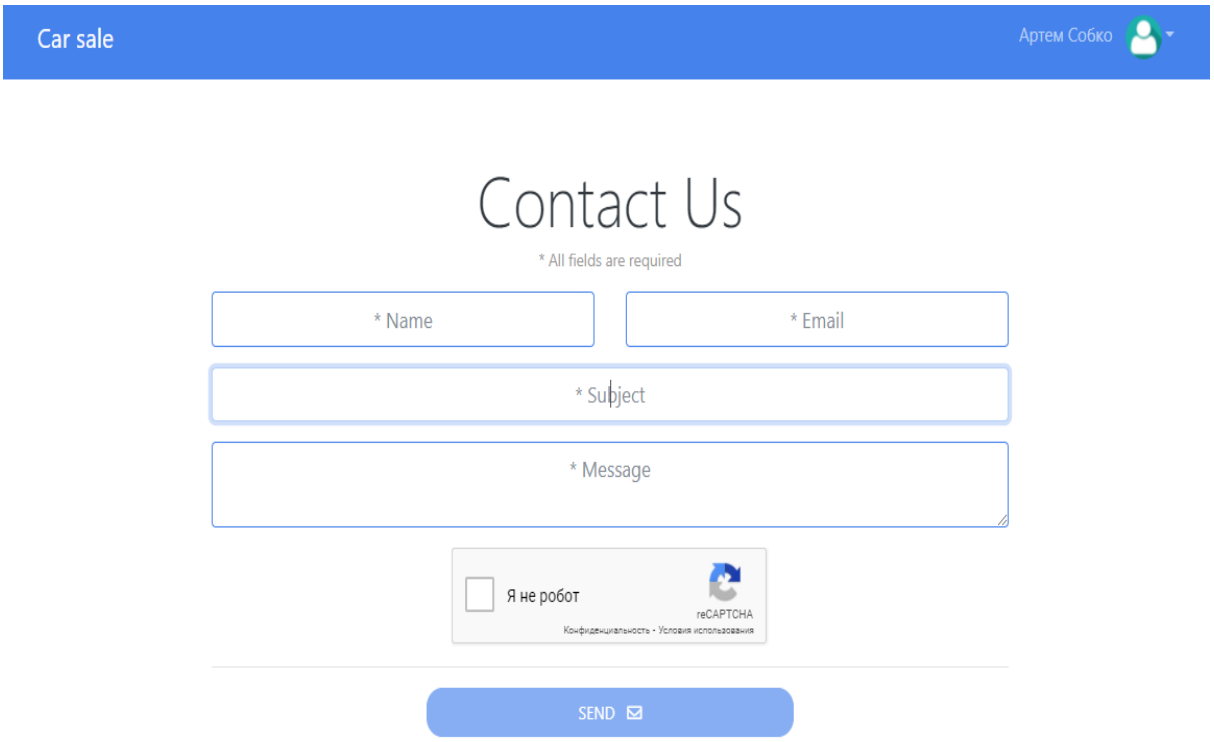


Рисунок 3.9 – Форма зв'язку з адміністрацією Інтернет-платформи

На цій сторінці у користувача буде можливість написати повідомлення з пропозицією або скаргою для адміністрації Інтернет-платформи, для цього користувачу потрібно заповнити усі поля, які є обов'язковими, з контактну інформацію (рисунок 3.9).

3.4 Технічні характеристики Інтернет-платформи

Для нормального функціонування додатку, зі сторони клієнтської частини, технічний пристрій повинен задовольняти наступні мінімальні технічні вимоги:

- 2- ядерний процесор intel/AMD (2.6ГГц);
- 2 Гб оперативної пам'яті;
- 4 Гб внутрішньої пам'яті, SSD або HDD;
- Доступ до мережі зі швидкістю більше 10Мбіт/с.

Мінімальні ж вимоги для серверної частини повинні забезпечити для клієнтів безперебійний робочий режим, який зробить Інтернет-платформу доступною в любу годину для клієнтів. Такими вимогами є:

- процесор 4 ядра, intel Xeon;
- 8гб оперативної пам'яті;
- 15гб внутрішньої пам'яті;
- мінімальна швидкість доступу до мережі Інтернет 15Мбіт/с;

3.5 Завантаження Інтернет-платформи на хостинг

Щоб надати усім користувачам Інтернету доступ до розроблюваного сайту, потрібно його розмістити на хостингу. Хостинг представляє собою потужний сервер який буде підтримувати роботу Інтернет платформи та працювати цілодобово. Фактично ми передаємо розроблюваний проект на чужий комп'ютер, який забезпечить безперервну роботу Інтернет-платформи

для продажу автомобілів. На даний момент існує доволі багато компаній надають доступ до своїх серверів для розміщення різних сайтів, як на безкоштовній основі, так і на платній, з різними тарифними планами. Ці тарифи впливають на рівень обмежень які надаються користувачу, а це в свою чергу, безпосередньо, впливає на вартість тарифного плану. Але усі вони класифікуються по різним категоріям. Перш за все існує безкоштовні та платні хостинги. Недоліками ж безкоштовних хостингів є: значно повільна швидкість роботи, ненадійність, реклама яка буде з'являтися на сайті та рекламувати даний хостинг, а також будуть відсутні функції які можуть бути дуже важливими.

Також хостинги можна класифікувати і за іншою категорією, а саме за типом ресурсу. Таким чином їх можна поділити ще на чотири види, такі як: виділений сервер, колокація, віртуальний хостинг, а також віртуальний виділений сервер. Усі вони розрізняються, наприклад:

- віртуальний виділений сервер ;
- віртуальний хостинг;
- колокація;
- віддалений хостинг.

Використовуючи віртуальний виділений сервер, користувачу виділяється певна кількість пам'яті на диску, частина глобальної пам'яті, а також процесорний час на сервері.

Віртуальний хостинг є досить економічним та підходить для невеликих сайтів, так як при такому застосуванні на одному сервері розміщується значна кількість сайтів.

Використовуючи колокацію клієнту виділяється місце від провайдера у самому дата центрі для того щоб установити та підключити можливість доступу до мережі Інтернету.

При використані виділеного хостингу клієнту надається персональний сервер у повне використання, а хостинг компанія, яка його надає, й буде відповідати цілком за обслуговування хостингу, тому даний тип розміщення

буде найбільш ефективним в більшості для великих Інтернет-платформ, яким необхідно зберігати та обробляти значну кількість даних;

Була обрана українська хостинг фірма ukrline.com.ua завдяки неї можна отримати доступ як і до хостингу так і до VPS. Тут є перелік тарифів які клієнт може обрати відповідно до саме свої потреб, мною ж був обраний тариф VPS-2, який має наступний перелік доступних характеристик:

- процесор на 4 ядра;
- 9GB оперативної пам'яті;
- 120 GB постійної пам’яті SSD;
- 120 GB постійної пам’яті формати HDD;
- трафік на 20TB;
- необмежена кількість сайтів;
- два виділених домених імені.

Також даний хостинг, як і більшість інших серйозних хостингів, надає безкоштовний пробний період для використання, у процесі якого клієнт може ознайомитися з повним його функціоналом та вирішити чи він підходить саме йому для потрібного використання.

Ресурси які були виділені хостингом є цілком достатніми для вимог розроблюваної Інтернет-платформи.

Для початку потрібно налаштувати VPS сервер, для цього потрібно підключитись до нього завдяки наданим параметрам входу. Для цього використовується SHH протокол підключення. Підключившись до сервера потрібно завантажити з GIT проект, який розроблювався для дипломної роботи. Налаштувавши доступ до сайту завдяки створеному файлу .env, завдяки якому можна задати логін та пароль.

Після отримання доступу до системи було завантажено докер, який у свою чергу завантажив необхідні програмні модулі для повного функціонування системи. Усі необхідні модулі як потребує Інтернет-платформа є прописаними в файлі докера. Зображення докера представлено на рисунку 3.9.

```
LABEL author="Felix Avelar"

WORKDIR /usr/src/app

COPY package*.json ./

COPY . .

# Install React dependencies
WORKDIR /usr/src/app/client
RUN npm install
# Build static files from React project
RUN npm run build

#Installing main dependencies
WORKDIR /usr/src/app
RUN npm install

# Set Enviroment variables for production
ENV NODE_ENV production
ENV SITE_KEY secret
ENV MONGO_URI mongoURI
ENV SECRET_OR_KEY secret
ENV GOOGLE_CLIENT_ID secret
ENV GOOGLE_CLIENT_SECRET secret
ENV SENDGRID_KEY secret
ENV SMTP_USER usergmail.com
ENV SMTP_PWD userpassword
ENV SMTP_SERVER smtp.gmail.com
ENV MY_HOST host
ENV PORT 80

EXPOSE 80
```

Рисунок 3.9 – структура файла докера

Отже у цьому розділі було розроблено інтернет-платформу для продажу автомобілів та завантажено її на хостинг, після усіх цих маніпуляцій наш сайт готовий до використання, а завдяки з’єднаю SHH, можна з легкістю у любий момент підключитися до сервера та ввести змини або ж оновити додаток.

4 ТЕСТУВАННЯ ІНТЕРНЕТ-ПЛАТФОРМИ

4.1 Вибір та обґрунтування методів тестування Інтернет-платформи

Процес тестування програмного забезпечення – це такий процес, що перевіряє поведінку, працездатність та відповідність до бізнес-вимог в цілому. Дане тестування – це необхідна частина розробки проекту, адже воно дозволяє забрати дефекти, які можуть проявлятися при системній роботі, а також дає гарантію того, що при розширенні проекту у функціональному плані усі процеси не зупиняться.

Для перевірки коректності роботи програмного продукту існують модульні тести. Програмісти застосовують і розробляють модульне тестування. Найпоширенішими результатами даного тестування є те, що код відповідає заданим вимогам, а також має передбачувану поведінку.

Серверна частина є найголовнішою частиною додатку. Тестування проводиться починаючи від низькорівневого до високорівневого компонентів.

Задля проведення тестування використовується інструменти Jest та Enzyme, з допомогою яких і будуть написані модульні тести, адже даний інструмент дозволяє виконувати різноманітні функціональні тести.

4.2 Модульне тестування Інтернет-платформи

Для початку потрібно розробити тестові сценарії, за допомогою яких зможемо протестувати функціональні можливості програми. В таблиці 4.1 наведені тестові сценарії модульних тестів.

Таблиця 4.1 – Сценарії модульних тестів

Ідетифікатор	Модуль	Вихідні дані	Очікуваний результат
1	2	3	4
М-А-1	Реєстрація користувача	Відсутні	Створений обліковий запис користувача

Кінець таблиці 4.1

1	2	3	4
М-А-2	Пошук оголошення	Назва оголошення	Список оголошень які відповідають критерію пошуку
М-А-3	Створення нового оголошення	Дані користувача	Оголошення створено
М-А-4	Завантаження фото транспорту	Фото оголошення	Фото успішно завантажено
М-А-5	Видалення оголошення	Запит на видалення	Оголошення успішно видалено
М-А-6	Відображення оголошення	Ідентифікатор оголошення	Відображення оголошення
М-А-7	Редагування оголошення	Створення запиту на оновлення оголошення	Оголошення відредаговано

Фрагмент коду який демонструє реалізацію тесту з ідентифікатором J-A-2 представлений нижче.

```
import React from 'react';
import { shallow } from 'enzyme';
import { SearchComponent } from '.';

describe('SearchComponent', () => {
  const posts = {
    content: [
      {
            id: '609ea0facc5d6b151472f338',
            posted_date" : "2021-05-13T17:24:56.849Z",
            title : "MAN TGX 18.480 4x2 BLS 2013 ",
            price : 22000,
            category : "vehicles",
            make : "Mercedes Benz",
            year : "2013",
            gas : "Diesel",
            transmision : "Manual",
      },
      {
            id: ' 609ea24fcc5d6b151472f339',
            posted_date" : " 2021-05-13T17:24:56.849Z",
            title : " ЗАЗ A10C3A 44 Городской АТ 160 к.с 2019",
            price : 110000,
            category : "vehicles",
```

```
            make : "ЗАЗ",
            year : "2019",
            gas : "Diesel",
            transmision : "bus",
        }
      ]
    };

    it('should search and filter job list correctly', () => {
      const wrapper = shallow(<SearchComponent posts={ posts }></SearchComponent>);
      expect(wrapper.find('#searchPosts')).toHaveLength(1);
      const mockedChangeEvent = { target: { value: 'Lead' } };
      expect(wrapper.state('search')).toBe('');
      wrapper.find('#searchPosts').simulate('change', mockedChangeEvent);
      expect(wrapper.state('search')).toBe('Lead');
      expect(wrapper.find('.post-list').children()).toHaveLength(1);
      expect(wrapper.html()).toMatchSnapshot();
    });
});
```

Тестування системи буде відбуватися відповідно до інструкцій користувача які наведені в керівництві з експлуатації та відповідно до функціональних вимог програмного забезпечення. Перелік деяких сценаріїв з тестування є наведений в таблиці 4.2.

Таблиця 4.2 – Тестові сценарії системних тестів

Ідетифікатор	Модуль	Вихідні дані	Очікуваний результат
1	2	3	4
TS-1	Головне сторінка	1. Натиснути на кнопку “Авторизуватися”; 2. Натиснути на кнопку “пошук оголошення”; 3. Натиснути на кнопку карточку оголошення.	1.Відображається сторінка авторизації; 2.Відображається список найдених оголошень; 3.Відображається сторінка з детальною інформацією про оголошення.
ST-2	Створення нового оголошення	1.Натиснути на певну категорію; 2.Натиснути на додавання фото;	1.Відображається список доступний для заповнення; 2.Відкривається менеджер для завантаження фото;

Кінець таблиці 4.2

1	2	3	4
		3. Натиснути на кнопку зберегти; 4. Ввести некоректні дані.	3. Повідомлення про то шо оголошення збережено. 4. Повідомлення для користувача про некоректно заповнені дані
ST-3	Відображення публікації	1. Натиснути на логотип сайту; 2. Ввести назву оголошення та натиснути “пошук”; 3. Вибрати категорію оголошення яка цікавить та натиснути “пошук”.	1. Відображається головна сторінка з усіма оголошеннями; 2.Відображення оголошень які відповідають заданій назві; 3. Відображення оголошень які відповідають обраній категорії.
ST-4	Зворотній зв’язок	1. Натиснути кнопку “зворотній зв’язок”; 2. Некоректно заповнені дані; 3. Вірно заповнені дані та натиснути кнопку “надіслати”.	1. Відображення сторінки з формою зворотного зв’язку; 2. Повідомлення про некоректно заповнені дані; 3. Повідомлення для користувача про те що його скарга/пропозиція була успішно надіслана.
ST-5	Авторизація	1. Ввести некоректні дані для авторизації; 2. Натиснути на кнопку “відновити пароль”; 3. Вірно введені дані авторизації	1. Повідомлення про некоректно введені дані; 2. Відображення форми для відновлення паролю; 3. Перезавантаження сайту з авторизованим користувачем та доступом до певного функціоналу.
ST-6	Реєстрація	1. Ввести некоректні дані; 2. Не заповнити обов’язкові поля форми; 3. Вірно заповнені дані.	1. Відображення повідомлення про некоректні дані; 2. Повідомлення про не вказані обов’язкові дані; 3. Відображення повідомленні про успішну реєстрацію.

4.3 Аналіз результатів тестування Інтернет-платформи

У результаті тестування були отримані результати, які показані на рисунку 4.1. Аналізуючи дані результати можна зробити висновок що система працює вірно, а усі заявлені функцію реалізовані та відповідають заявленим у завданні на виконання дипломного проекту.

```
 PASS  src/index.spec.tsx
  SearchComponent
    ✓ should search and filter list correctly (19ms)

-----------|----------|----------|----------|----------|-------------------|
File       |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
-----------|----------|----------|----------|----------|-------------------|
All files  |      100 |      100 |      100 |      100 |                   |
 index.tsx |      100 |      100 |      100 |      100 |                   |
-----------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 passed, 1 total
Time:        3.058s, estimated 5s
```

Рисунок 4.1 – результат тестування модуля пошуку оголошення

Опираючись на результати тестів можна створити наступну таблицю з результатами їх виконання (таблиця 4.3).

Таблиця 4.3 – Результати тестів

Ідентифікатор	Опис	Вхідні значення	Вихідні значення	Результат
1	2	3	4	5
JC-1	Пошук оголошення	Назва оголошення	Оголошення яке відповідає заданим параметрам	Правильно

Кінець таблиці 4.3

1	2	3	4	5
JC-2	Реєстрація користувача	Дані користувача	Створений обліковий запис користувача	Правильно
JC-3	Створення нового оголошення	Дані користувача з приводу оголошення	Створене нове оголошення	Правильно
JC-4	Видалення оголошення	Ідентифікатор оголошення та запит на видалення	Оголошення видалено з бази даних	Правильно
JC-5	Редагування оголошення	Нові дані для оголошення	Збережене відредаговане оголошення	Правильно
JC-6	Відображення переліку оголошень	Опубліковані оголошення	Сторінка з оголошеннями	Правильно
JC-7	Завантаження фото	Файл фотографії	Успішно завантажене фото	Правильно

Отже, протестувавши створений додаток, не було виявлено будь-яких помилок, а це означає що він повністю готовий до використання.

ВИСНОВКИ

Темою мого дипломного проектування є Інтернет-платформа для продажу авто. У сучасному світі технології з кожним днем стають все більш розвинуті, а отримання певних послуг за допомогою всесвітньої мережі Інтернет стає набагато простіше і оптимальніше.

Одними з таких послуг є Інтернет-платформи із продажу авто. Проаналізувавши ринок представлених додатків було вирішено що розроблюваний мною веб-додаток складе гарну конкуренцію вже існуючим застосункам. Тому що представлені веб-додатки мають ряд недоліків. Завдяки ним користувачі можуть обрати необхідний їм автомобіль, що буде задовольняти по всім параметрам у режимі онлайн з будь якого міста країни. Такі платформи є вигідними як для покупців, так і для продавців. Першим не потрібно буде витрачати час на відвідування великої кількості автосалонів чи авторинкам, у марних пошуках того самого авто, другі ж не будуть весь свій робочий день даремно очікувати на клієнта, а також платити оренду за місце на авторинках, не знаючи чи прийде той самий клієнт.

Основною ж метою мого сайту, розробити веб-додаток, який спростить життя для тисяч людей, економлячи їхній час та гроші. Даний сайт був розроблений у процесі виконання дипломного проекту. Дана тема і розроблений мною сайт є достатньо актуальними, у зв’язку із переходом великої кількості сервісів та магазинів у онлайн режим, а різні фільтри сприяють більшій зручності для обрання користувачам бажаного авто із заданими характеристиками.

При реалізації дипломного проекту було створено різні моделі та розглянуто різноманітні архітектурні рішення, які і були використані у даному проекті. Реалізувавши проект було проведено ряд тестувань які показали що розроблюваний проект є повністю функціональним, а реалізовані функції працюють вірно.

Отже, результатом виконання дипломного проекту є Інтернет-платформа для продажу авто, яка готова до використання та допоможе тисячам людям найти авто, а також продати власне. Вона має деякі переваги над вже існуючими рішеннями, що в свою чергу є важливим моментом для її популярності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Продаж на Інтернет-ринку: Найкращі платформи для продажу вашої продукції [Електронний ресурс] / EL News. – Режим доступу до ресурсу: https://elnews.com.ua/uk/prodazh-na-internet-rynkah-najkrashhi-platformy-dlya-prodazhu-vashoyi-produkcziyi/

2. Интернет-продажи: способы, технологии и инструменты маркетинга в Сети [Электронный ресурс] / Аргументы и факты. – Режим доступу к ресурсу: https://aif.ru/boostbook/internet-prodazhi.html

3. Нотация IDEF0 [Электронный ресурс] / Business Studio. – Режим доступу к ресурсу: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0

4. Уніфікована мова програмування UML [Електронний ресурс] / Портал знань. – Режим доступу до ресурсу: http://www.znannya.org/?view=uml

5. HTTP: Протокол, який повинен розуміти кожний веб-розробник [Електронний ресурс] / Envato tuts plus. – Режим доступу до ресурсу: https://code.tutsplus.com/uk/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177

6. HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. - М.: Эксмо, 2013

7. Реляційна модель баз даних [Електронний ресурс] / Бази даних. СУБД Access. – режим доступу до ресурсу: https://sites.google.com/site/bazidanihsubdaccess/bazi-danih-subd/ierarhicna-model-danih/relacijna-model-baz-danih

8. Что такое стек MERN, и как с ним работать? [Электронный ресурс] / Хабр // Tim Smith. – Режим доступу к ресурсу: https://habr.com/ru/company/piter-/blog/458096/

9. NOSQL – переваги ти недоліки нереляційних баз даних [Електронний ресурс] / QAinfo. – Режим доступу до ресурсу: https://www.quality-assurance-

group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/

10. Что такое MongoDB? [Электронный ресурс] / Coder Lessons. – Режим доступу к ресурсу: https://coderlessons.com/tutorials/bazy-dannykh/uchebnik-mongodb/1-chto-takoe-mongodb

11. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE [Electronic resource] / guru 99. - Avalilable from: https://www.guru99.com/unit-testing-guide.html

12. Модульне тестування [Електронний ресурс] / QaLight. – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/modulne-testuvannya/

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Виконується робота з розробки програмного забезпечення для дипломного проекту на тему «Інтернет-платформа для продажу автомобілів».

1 Підстава для розробки

Підставою для розробки програмного забезпечення є виконання дипломного проекту. До 01.05.2021 року повинна бути розробленні Інтернет-платформа для продажу автомобілів.

2 Призначення розробки

Інтернет-платформа для продажу автомобілів призначається для того щоб:

- допомога з продажом автомобілів для продавців;
- допомога для покупців обрати авто.

Дане програмне забезпечення має на меті спростити важливий етам життя з купівлею та продажом автомобіля, як і для покупців так і для продавців. Завдяки цьому не прийдеться відвідувати авторинки так як усе можна буде знайти з ліжка.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Інтернет-платформа повинна реалізовувати наступні функції:

- реєстрація нового користувача;

- авторизація для уже зареєстрованих;
- перегляд оголошень створених раніше користувачами;
- перегляд власних оголошень;
- створення власного оголошення;
- редагування персональних даних;
- зміна паролю;
- можливість видалення оголошення;
- можливість редагування оголошення;
- пошук оголошення за фільтром та назвою;
- перегляд детальної інформації про оголошення.

3.2 Вимоги до надійності

Для надійного використання системи повинен бути обраний надійний сервер, який забезпечить безперервну роботу системи. У свою чергу надійність систем повинна забезпечуватися завдяки надійні роботі усіх модулів системи. Також одним з важливих аспектів є збереження даних. Дані повинні надійно зберігатися. Передбачений захист від несанкціонованого доступу до них.

3.3 Умови експлуатації

Задля нормального використання програмного забезпечення умови використання повинні задовольняти санітарно-технічним нормам для експлуатації комп’ютерних пристроїв, згідно зі стандартом ГОСТ 15150-69 у якому зазначаються нормальні умови для правильного використання обчислювальної техніки.

Для забезпечення безпечної та стійкої роботи системи, для її обслуговування мають право долучатися лише спеціально навчені розробники та адміністратори. Проте скористатися системою для отримання потрібної інформації, може звичайний Інтернет-користувач.

3.4 вимоги до складу та параметрів технічних засобів

Для функціонування системи досить задовільнити мінімальні вимоги які поставлені будь яким браузером. Приклад технічних вимог наведено нижче для браузера Opera:

- Платформа: Windows 10, 8, 8.1, 7;
- Розрядність: x86 або ж x64;
- Внутрішня пам'ять: 400Mb;
- Оперативна пам’ять: 512
- Аудіокарта: Будь-яка;
- Контролер: Клавіатура, Миша;
- Інтернет: Стабільне з'єднання;
- роздільна здатність екрану: SVGA 800x600.

3.5 Вимоги до інформаційної та програмної сумісності

Створити серверну частину необхідно використовуючи наступні технології:

- JavaScript – мова програмування яка широко використовується при розробці веб-додатків;
- Node.js – бібліотека використовується для створення серверної частини;
- Express.js – каркас для створення серверної частини;

– MongoDB – база даних.

Для реалізації ж фронту буде використовуватися:

– React.js – бібліотека яка використовується для створення логіки на фронтовій частині сайту;
– CSS – таблиця каскадних стилів, використовується для створення привабливого дизайну сторінки;
– HTML – розмітка сторінки.

3.6 Спеціальні вимоги

Програмний продукт має коректно відображатися на усіх популярних браузерах та підтримувати зміну розширення екрану. Також повинен мати привабливий та мінімалістичний дизайн.

4 Вимоги до програмної документації

При відправлені проекту замовнику надається наступний перелік документації:

– інструкція користувача;
– текст програмного продукту;
– опис програми та її функціональних особливостей;
– інструкція для програмістів;
– технічне завдання.

5 Стадії та етапи розробки

Стадії та етапи розробки для Інтернет-платформи з продажу автомобілів наведено в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
Технічне завдання 02.01.21 – 31.01.21	Обгрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.21 – 14.02.21	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури системи, що буде розроблюватися
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури
15.02.21 – 28.02.21	Розробка технічного проекту	програми; остаточне визначення конфігурації технічних засобів
Робочий проект 01.03.21 – 10.04.21	Розробка програмного забезпечення	Реалізація програмного забезпечення; відладка; проведення попереднього тестування
Розробка програмної документації 11.04.21 – 20.04.21	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 21.04.21 – 30.04.21	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і передача програмного забезпечення; навчання персоналу використовуванню програмного забезпечення;

6 Порядок контролю та приймання

Контроль за процесом розробки Інтернет-платформи здійснюється розробником, а також користувачами, які прийняли участь у процесі тестування додатку. Прийняття Інтернет-платформи здійснюється екзаменаційною комісією.

ДОДАТОК Б

(обов’язковий)

ФРАГМЕНТИ КОДУ ПРОГРАМНОЇ СИСТЕМИ

Функція ControllerAllPosts, яка повертає список усіх оголошень

```
exports.AllPostsController = async (req, res) => {
      const { errors } = req;

      try {
            const escapeRegex = (text) => text.replace(/[-
[\]{}()*+?.,\\^$|#\s]/g, '\\$&');
            const PAGE = req.query.page || 1;
            const SEARCH = req.query.search;
            const CATEGORY = req.query.category;
            const PER_PAGE = 6;
            let TOTAL_ITEMS;
            const regex = new RegExp(escapeRegex(SEARCH), 'gi');

            let pagePosts;

            if (SEARCH === 'none' && CATEGORY === 'none') {
                  pagePosts = await
                        Post.find({})
                        .sort({"posted_date": -1})
                        .skip(PER_PAGE * PAGE - PER_PAGE)
                        .limit(PER_PAGE);
                  TOTAL_ITEMS = await Post.countDocuments();

            } else if (SEARCH === 'none' && CATEGORY !== 'none') {
                  pagePosts = await
                        Post.find({ category: CATEGORY })
                        .sort({"posted_date": -1})
                        .skip(PER_PAGE * PAGE - PER_PAGE)
                        .limit(PER_PAGE);
                  TOTAL_ITEMS = await Post.find({ category: CATEGORY
}).countDocuments();

            } else if (SEARCH !== 'none' && CATEGORY === 'none') {
                  pagePosts = await
                        Post.find({ $or: [ { title: regex }, { description: regex
} ] })
                        .sort({"posted_date": -1})
                        .skip(PER_PAGE * PAGE - PER_PAGE)
                        .limit(PER_PAGE);
                  TOTAL_ITEMS = await Post.find({ $or: [ { title: regex }, {
description: regex } ] }).countDocuments();

            } else {
                  pagePosts = await
                        Post.find({ category: CATEGORY }, { $or: [ { title: regex
}, { description: regex } ] })
                        .sort({"posted_date": -1})
```

```
                        .skip(PER_PAGE * PAGE - PER_PAGE)
                        .limit(PER_PAGE);

                  TOTAL_ITEMS = await
                        Post.find({ category: CATEGORY },{ $or: [ { title: regex
}, { description: regex } ] })
                        .countDocuments();
            }

            if (!pagePosts) {
                  errors.posts = 'No Posts found';
                  return res.status(403).json(errors);
            }

            res.status(200).json({
                  posts: pagePosts,
                  total_Items: TOTAL_ITEMS,
            });

      } catch (err) {
            res.status(500).json({ Error: `${err}` });
      }
};
```

Функція ControllerCreatePost яка додає нове оголошення

```
exports.CreatePostController = async (req, res) => {
      const { errors } = req;
      try {
            const { files } = req;
            const images = files.map((file) => file.path);
            const owner = req.user._id;
            const { contactEmail, contactPhone, title, description, category } =
req.body;
            const price = parseFloat(req.body.price);
            const { make, year, gas, model, type, transmision, trafficAccident }
= req.body;
            const { propertyType, transaction, rooms, bathrooms } = req.body;
            let newPostData = {
                  owner,
                  title,
                  price,
                  description,
                  contactEmail,
                  contactPhone,
                  description,
                  category,
                  images
            };
```

```
            const newPost = new Post(newPostData);
            await newPost.save();
            res.status(200).json({ msg: 'Post created' });
      } catch (err) {
            errors.error = err;
            res.status(500).json({ERROR: errors.toString()});
      }
};
```

Функція ControllerDeletePost видаляє допис завдяки ідентифікатору

```
exports.DeletePostController = async (req, res) => {
      const { errors } = req;

      try {
            const { _id: userId } = req.user;
            const { postId } = req.body;

            const post = await Post.findOne({
                  _id: postId,
                  owner: userId
            });

            const { images } = post;

            if (images.length > 0) {
                  try {
                        images.forEach((image) => fs.unlinkSync(image));
                  } catch (err) {
                        errors.error = err;
                        res.status(500).json(errors);
                  }
            }

            await post.remove();

            res.status(200).json({ msg: 'Post Deleted' });
      } catch (err) {
            errors.error = err;
            res.status(500).json(errors);
      }
};
```

Функція RegisterController створює нового користувача системи

```
exports. ControllerRegister = async (req, res) => {
      try {
```

```
            const { errors } = req;

            //checks if the email address already exist
            const user = await User.findOne({ email: req.body.email });
            if (user) {
                  //If exist return an error
                  errors.email = 'Email is already registered';
                  return res.status(409).json(errors);
            } else {
                  //Create a random Token
                  const verificationToken = await
crypto.randomBytes(32).toString('hex');

                  const expVerificationToken = Date.now() + 3600000; // Token
will expire in an Hour

                  console.log(verificationToken);

                  //If address does not exist, register the user
                  const { firstName, lastName, email, password } = req.body;

                  const newUser = new User({
                        firstName,
                        lastName,
                        email,
                        password,
                        verificationToken,
                        expVerificationToken
                  });

                  const messageData = { firstName, email, token:
verificationToken };

                  // Encrypt the password before save it in the DB
                  bcrypt.genSalt(10, (err, salt) => {
                        bcrypt.hash(newUser.password, salt, async (err, hash) =>
{
                              try {
                                    if (err) throw err;
                                    newUser.password = hash;
                                    //register the user in the Database
                                    await newUser.save();

                                    sendVerificationToken(messageData);

                                    res.status(201).json({ msg: 'OK' });
                              } catch (error) {
                                    res.status(500).json({ error: `User
registration failed see ${error}` });
                              }
                        });
```

```
                });
            }
        } catch (err) {
            res.status(503).json({ error: err.toString() });
        }
};
```

Функція оновлення даних користувача UpdateUserProfile

```
exports.UpdateUserProfile = async (req, res) => {
    try {
        const { _id } = req.user;
        const { firstName, lastName, public_email, phone, errors } =
req.body;

        if (req.file) {
            const avatar = /* "http://localhost:5000/" + */ req.file.path;
            updatedInfo = { firstName, lastName, public_email, phone,
avatar };
        } else {
            updatedInfo = { firstName, lastName, public_email, phone };
        }

        const updatedUser = await User.findByIdAndUpdate(_id, updatedInfo, {
new: true });
        if (!updatedUser) {
            errors.user = 'Invalid Request';
            res.status(400).json(errors);
        } else {
            const { id, firstName, lastName, email, role, avatar, phone,
public_email, isVerified } = updatedUser;

            const payload = { id, firstName, lastName, email, role, avatar,
phone, public_email, isVerified };

            jwt.sign(payload, secretOrKey, { expiresIn: '1d' }, (err,
token) => {
                if (err) throw err;
                res.status(200).json({
                    succsess: true,
                    token: `Bearer ${token}`
                });
            });
        }
    } catch (error) {
        console.error(error);
        res.status(500).json(error);
    }
};
```

Модуль з реєстрації користувача

```
const Validator = require('validator')
const isEmpty = require('./is-empty')

module.exports =  (req, res, next) => {

  let { firstName, lastName, email, password, password2 } = req.body

  let errors = {}

  firstName = !isEmpty(firstName) ? firstName : ''
  lastName = !isEmpty(lastName) ? lastName : ''
  email = !isEmpty(email) ? email : ''
  password = !isEmpty(password) ? password : ''
  password2 = !isEmpty(password2) ? password2 : ''

  //First Name Validation
  if(Validator.isEmpty(firstName)) {
    errors.firstName = "First Name is required"
  } else if (!Validator.isLength(firstName, {min: 2, max: 30})) {
    errors.firstName = "First Name must have between 2 and 30 characters"
  }

  //Last Name Validation
  if(Validator.isEmpty(lastName)) {
    errors.lastName = "Last Name is required"
  } else if(!Validator.isLength(lastName, {min: 2, max: 30})) {
    errors.lastName = "Last Name must have between 2 and 30 characters"
  }

  //Email Validation
  if(Validator.isEmpty(email)) {
    errors.email = "Email is required"
  } else if(!Validator.isEmail(email)) {
    errors.email = "Invaild Email format"
  }

  //Password Validation
  if(Validator.isEmpty(password)) {
    errors.password = "Password is required "
  } else if(!Validator.isLength(password, {min:6, max:30})){
    errors.password = "Password must be 6 characters in length"
  }

  if(Validator.isEmpty(password2)) {
    errors.password2 = "Confirm your password"
  } else if(!Validator.equals(password, password2)) {
    errors.password2 = "Passwords don't match"
  }
```

```
  if(!isEmpty(errors)) {
    return res.status(400).json(errors)
  }

  req.errors = errors;

  next()
}
```

Модуль створення відгуку завдяки контактній формі

```
const Validator = require('validator')
const isEmpty = require('./is-empty')

module.exports =  (req, res, next) => {

  let { name, email, subject, message } = req.body

  let errors = {}

  name = !isEmpty(name) ? name : ''
  email = !isEmpty(email) ? email : ''
  subject = !isEmpty(subject) ? subject : ''
  message = !isEmpty(message) ? message : ''

  //Name Validation
  if(Validator.isEmpty(name)) {
    errors.name = "Name field is Required"
  } else if (!Validator.isLength(name, {min: 1, max: 30})) {
    errors.name = "Name must have between 1 and 30 chars"
  }

  //Email Validation
  if(Validator.isEmpty(email)) {
    errors.email = "Email is Required"
  } else if(!Validator.isEmail(email)) {
    errors.email = "Invalid email format"
  }

   //Name Validation
   if(Validator.isEmpty(subject)) {
    errors.subject = "Subject field is Required"
  } else if (!Validator.isLength(subject, {min: 1, max: 100})) {
    errors.subject = "Subject must have between 1 and 60 chars"
  }

   //message Validation
   if(Validator.isEmpty(message)) {
```

```
    errors.message = "Message field is Required"
  } else if (!Validator.isLength(message, {min: 1, max: 500})) {
    errors.message = "Message must have between 1 and 500 chars"
  }

  if(!isEmpty(errors)) {
    return res.status(400).json(errors)
  }

  req.errors = errors;

  next()
}
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький Національний Університет
Факультет програмування та комп'ютерних
і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

Дипломний проект на тему: «Інтернет-платформа для продажу автомобілів»

Студент: Собко А. Г.
Керівник: кандидат пед. наук, доцент Онишко О. Г.

ВСТУП

З кожним днем все більше сервісів переходять на онлайн продажі, так як найти певні товари в офлайн магазинах не так і просто. На щастя з розвитком нових технологій онлайн продажі стали доступним майже для будь кого на планеті. Тисячі товарів продаються саме через інтернет і ця цифра тільки зростає, тому що це швидко та зручно, не потрібно навіть вставати з ліжка.

Запланований мною сайт позбавить людей зайвих проблем, тому що не вигідно роз'їжджати по різним авторинкам та автосалонам з надією підібрати саме те авто яке задовільнить покупця. Продавцям також не вигідно цілими днями стояти і очікувати на клієнта, при тому ще й плативши оренду за місце, так як покупець може знаходитись у іншому місті.

Але, якщо усі ці оголошення перенести на спеціалізовану інтернет-платформу, то можна буде дома обрати авто у будь якому місті країни, яке задовільнить по всім параметрам, а для зручності у виборі можна встановити різні фільтри, також це допоможе швидше та простіше продати власне авто, тому що його побачать у всьому світі. Це вигідно як і для покупців так і для продавців.

Актуальність теми

Актуальність теми дипломного проекту полягає у тому що існує доволі багато інтернет-платформ для продажу товарів, проте мало з них призначені для купівлі та продажу авто, а ті які існують мають свої недоліки, тому що більшість зосереджені на продажу побутових товарів. Не зважаючи на те що користуватись інтернет-платформами купівлі набагато зручніше, але для побутових речей можна і обійтись супермаркетами в яких представлено доволі багато товарів, а от для купівлі якісного авто яке задовільнить усі вимоги по комплектації та ціні без цього не обійтись, так як багато вигідних авто продаються в інших містах. Завдяки просторам інтернету це буде набагато швидше знайти, не відвідуючи різні авто-ринки та не відвідуючи інші міста не знаючи чи є там підходящий варіант.

Мета та завдання дипломного проекту

Метою проекту є створення Інтернет-платформи, призначеної для допомоги людям з продажом та купівлею авто, яка буде мати привабливий інтерфейс та відповідний функціонал.

Основними завданнями при виконані дипломного проекту є:

- Проведення аналізу предметної області та визначення основних особливостей

- Проведення аналізу ринку на наявність готового програмного забезпечення

- Визначення вимог до програмного забезпечення

- Проектування платформи за заданою предметно областю

Наявне програмне забезпечення

Зараз на просторах інтернету є інтернет-платформи які дозволяють користувачеві купити, а також продати авто. Усі вони користуються. популярністю, але так чи інакше, мають свої недоліки.

На сайті є різні оголошення від продавців. Вони сортуються у відповідні категорії (легковики, вантажівки, причепи, спец техніка, автобуси та інші). Тут можна як і продати свій транспорт так і придбати. Кожне з оголошень має свій опис, фото та технічні характеристики.

Інтерфейс є доволі зрозумілим та має велику кількість функцій, дизайн мінімалістичний. На даному порталі більшість функцій є платні, а кількість безкоштовних оголошень обмежена одним оголошенням у пів року, а також багато реклами, що відштовхує потенційних клієнтів.

Наявне програмне забезпечення

Також популярна сторінка в Україні, яка виконує свої функції. На ньому можна як і переглядати оголошення так і публікувати власні. Функціонал менший ніж на попередньому сайті, але доволі інтуїтивний. На сайті немає такої великої кількості реклами як на попередньому, проте дизайн не найкращий, на мою думку він є доволі застарілим та не сучасним.

Висновок з аналізу наявного програмного забезпечення

Провівши аналіз програмних засобів які вже представлені на ринку було вирішено що у нашому програмному продукті повинні бути реалізовані наступні вимоги:

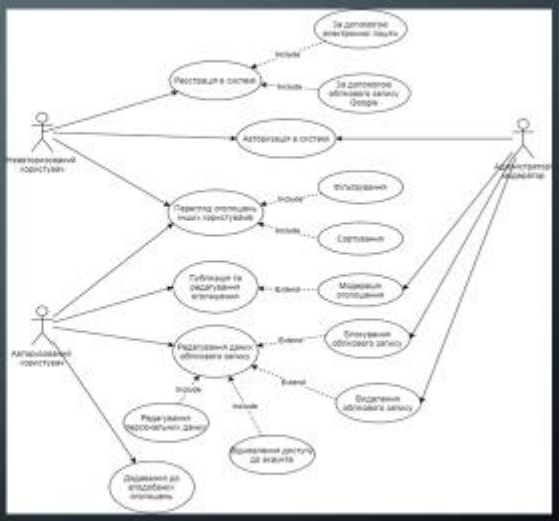
- Сучасний та інтуїтивний дизайн

- Різноманітність фільтрів пошуку

- Можливість додавання та перегляду оголошень

Проектування

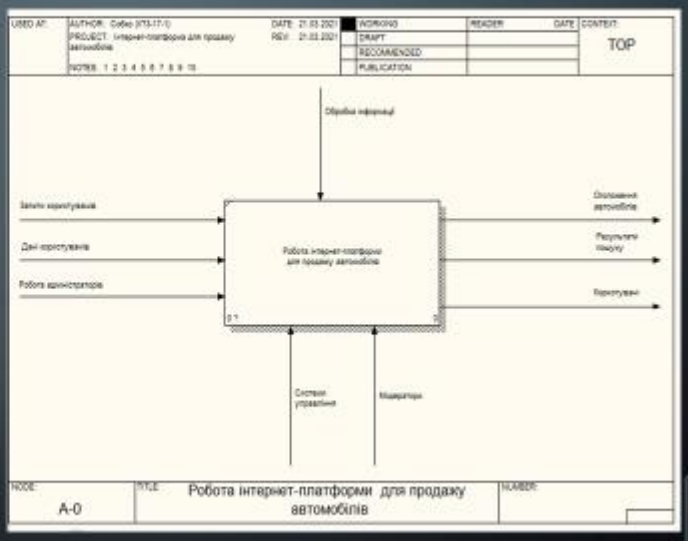
Для представлення функціонального призначення було побудовано діаграму використання. На ній зображено користувачів системи (акторів) з варіантами використання, які вони можуть зробити.



Діаграма варіантів використання

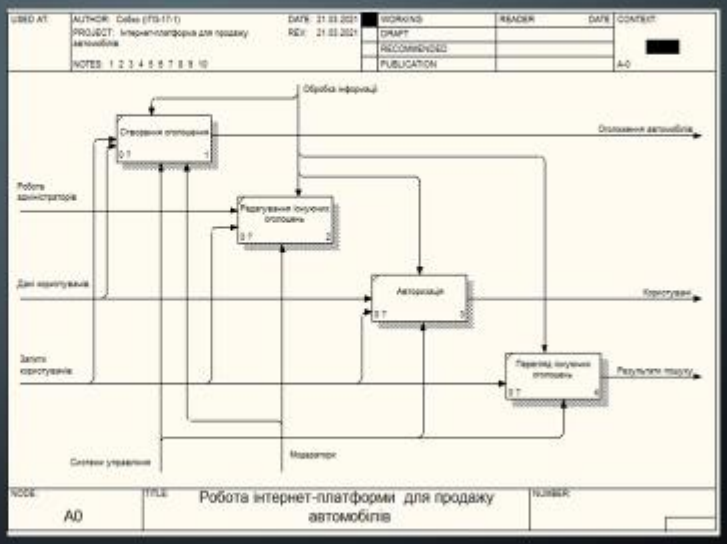
Проектування

Також побудована діаграма потоків даних першого рівня, для обробки даних інтернет-платформи

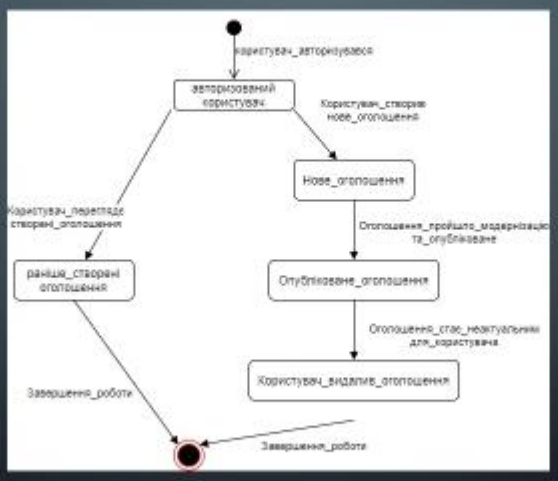


Проектування

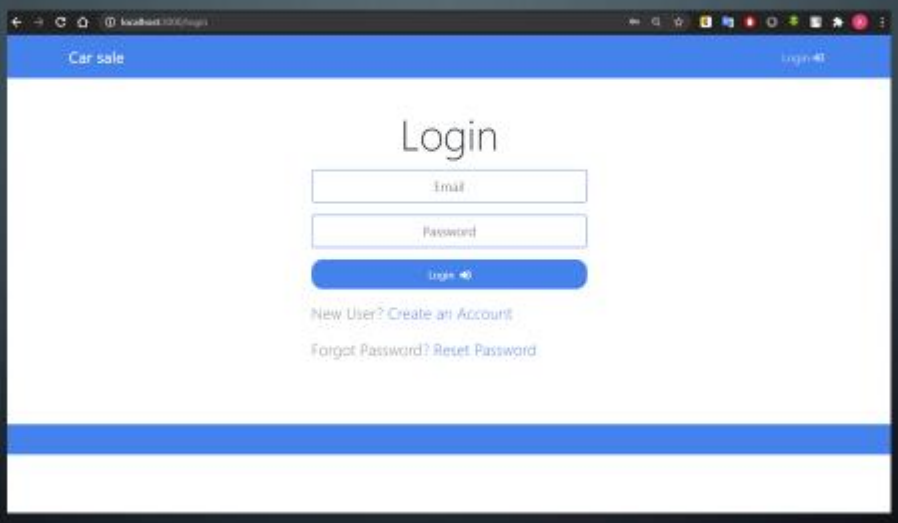
Декомпозиція діаграми потоків даних другого рівня.



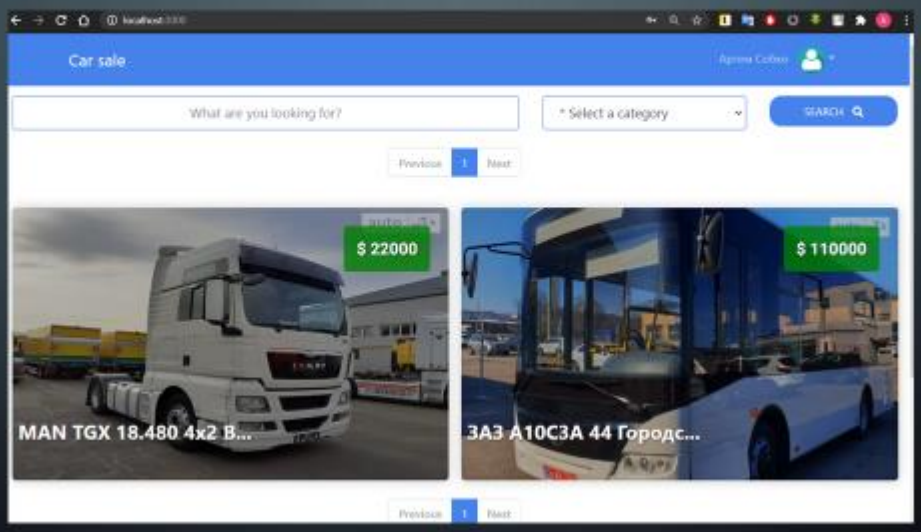
Принцип роботи реалізації



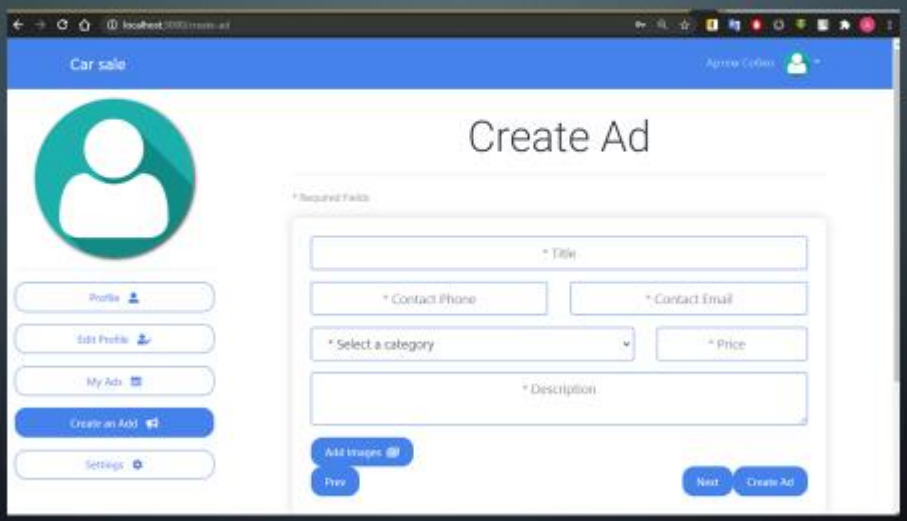
Форма авторизації



Головна сторінка інтернет-платформи



Сторінка створення оголошення



Висновок

У процесі виконання дипломного проекту було проаналізовано предметну область за темою «інтернет-платформа з продажу автомобілів», та встановлено що дана тема є доволі актуальною, так на ринку представлено мало конкурентів.

Проведено аналіз існуючого програмного забезпечення предметної області, за результатами якого було складено усі функціональні та не функціональні вимоги.

Після проведення проектування було обрано клієнт-серверний тип архітектури. Також була обрана база даних для швидкого сортування та зберігання даних.

У результаті виконання дипломного проекту було створено програмне забезпечення, яке відповідає поставленим функціональним вимогам та вирішує основні проблеми предметної області.

Дякую за увагу!

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Собко А. Г.
Прізвище, ініціали

факультет ПКТС, 4 курс, група ІПЗ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01.06.2021р.
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 93242 Назва: Інтернет-платформа для продажу автомобілів Додано в БД: 2021-06-11 Автора: А.Г. Собко Керівники: О.Г. Онишко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	70618	664	4687 (7%)	69 (10%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1008266532

Дата перевірки:
11.06.2021 09:28:46 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2021 09:29:29 EEST

ID користувача:
100005589

Назва документа: **Дипломний проект Собко Артем без додатків**

Кількість сторінок: **66** Кількість слів: **10428** Кількість символів: **77875** Розмір файлу: **4.20 MB** ID файлу: **1008337341**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.29%
Схожість

Найбільша схожість: **1.91%** з джерелом з Бібліотеки (ID файлу: **1008272523**)

3% Джерела з Інтернету	143	Сторінка 68
3.18% Джерела з Бібліотеки	49	Сторінка 69

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1
Підозріле форматування	13 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Собко Артем Геннадійович

Тема Інтернет-платформа для продажу автомобілів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень 0; кількість сторінок записки 88

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті проведений аналіз предметної області та проблеми, які потребують вирішення. Було проаналізовано програмні засоби які вже представленні на ринку, для визначення переваг та недоліків. Також були проаналізовані технології для реалізації даного дипломного проекту, на основі отриманих даних були розроблені технічні та функціональні вимоги до Інтернет-платформи. Реалізувавши додаток за шаблонами було проведено тестування Інтернет-платформи, результати якого показали що вона працює корентно.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект освітнього ступеня «бакалавр» в основному відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовується актуальність обраної теми дипломного проектування, формуються мета і завдання проектування, описується практична значимість дипломного проекту. У першому розділі проаналізовано предметну область, розглянуті існуючі рішення, які використовуються для вирішення поставленої задачі, викладено опис функціональних вимог до розроблюваної Інтернет-платформи. У наступних розділах розкриваються деталі проектування Інтернет-платформи, моделювання бази даних та системи в цілому, а також описана програмна реалізація системи та тестування.

4. Позитивні сторони проекту Тема дипломного проекту є актуальною, оскільки люди регулярно зіштовхуються з питанням купівлі та продажу автомобілів. До позитивних сторін реалізації Інтернет-платформи можна віднести те, що при розробці системи використовувались доволі нові технології які є найбільш оптимальними для даного проекту.

5. Негативні сторони проекту До негативних сторін проекту можна віднести те що дана Інтернет-платформа не є першою на задану тематику та має доволі серйозних конкурентів. Також інтерфейс хоч і є мінімалістичним, але доволі простий, не вистачає більшої кількості фільтрів.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконане відповідно до теми дипломного проекту. Графічне представлення виконано на задовільному рівні. Пояснювальна записка відповідає стандартам оформлення.

7. Відгук про дипломний проект в цілому В цілому дипломний проект заслуговує задовільної оцінки. Матеріал пояснювальної записки структурований та розписаний на рівні, що дозволяє зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал підкріплений інформацією та дозволяє наочно побачити результати виконання дипломного проекту.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект заслуговує оцінки «задовільно»

РЕЦЕНЗЕНТ Кисіль Тетяна Миколаївна, кандидат фіз.-мат. наук, доцент кафедри комп'ютерної-інженерії та системного програмування (КІСП) ХНУ

"14" серпня 2021 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Інтернет-платформа для продажу автомобілів»
Автор: Собко Артем Геннадійович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Онишко Оксана Григорівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 5,29% і адресується до 143 джерел з Інтернет та 49 джерел з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ О. Г. Онишко

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк