МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
«___» травня 2024 року

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 – Комп'ютерні науки,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна технологія супроводження діяльності евакуаторної компанії»
здобувача групи ІН.м-21н Бубенщикова Даниїла Євгеновича

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

_____ Даниїл БУБЕНЩИКОВ
(підпис)

Доцентка кафедри
комп'ютерних наук,
к.ф.-м.н., доцентка
Олексієнко Г.А.

_____ Галина ОЛЕКСІЄНКО
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо- наукової програми «Інформатика»
здобувача групи ІН.м-21н Бубенщикова Даниїла Євгеновича

1. Тема роботи: *«Інформаційна технологія супроводження діяльності евакуаторної компанії»* затверджую наказом по СумДУ від *«8» березня 2024 року № 0234-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 21 травня 2024 року*
3. Вхідні дані до кваліфікаційної роботи

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області та постановка завдань дослідження. 2) Огляд технологій розробки веб-продуктів. 3) Розробка механізмів роботи із веб-застосунками, доступність та адаптивність. 4) Вибір технологій реалізації та методології розробки. 5) Розробка окремого сайту, бази даних та взаємодія з картою. 6) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання « » р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області та постановка завдань дослідження*		
2	*Огляд технологій розробки Telegram-ботів*		
3	*Розробка механізмів інтенсифікації навчального процесу через Telegram-бот*		
4	*Вибір технологій реалізації та методології розробки*		
5	*Розробка бота та веб-серверу*		
6	*Аналіз результатів.*		
7	*Оформлення пояснювальної записки до кваліфікаційної роботи.*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підп

АНОТАЦІЯ

Записка: 79 стор., 23 рис., 1 додаток, 18 джерел.

Актуальність полягає у використанні зручного та зрозумілого веб-застосунку, який би міг врегулювати повноцінне функціонування евакуаторної компанії, є досить важливим, адже, під час війни на території України, значна кількість населення потребує допомоги у транспортуванні пошкодженого авто, його ремонті, забезпеченні армією волонтерською допомогою тощо. Веб-застосунок, що буде створено у результаті виконання даної кваліфікаційної роботи магістра, зможе заповнити цю потребу та, також, створить певну екосистему, адже буде працювати у парі із вже створеним мобільним додатком та буде забезпечувати доступність, адже з ним можна буде працювати на будь-якому пристрої та ОС відповідно.

Мета роботи – створення інформаційної технології засобами веб-розробки, яка б забезпечувала повноцінну роботу евакуаторної служби та забезпечувала доступність у використанні для всіх її користувачів.

Методи дослідження – відповідні методи інформаційного аналізу, такі як аналітичний огляд існуючих рішень, порівняльний аналіз, експериментальна розробка та статистичний аналіз.

Результатом першого розділу буде сформована постановка задачі та основні вимоги до застосунку, складені на основі аналізу конкурентів на ринку; результатом другого розділу є повноцінна розробка дизайну сайту та його шаблону; результатом третього розділу, практичної реалізації, є розробка застосунку, який працює у середовищі будь-якого існуючого веб-браузера та забезпечує повноцінну роботу евакуаторної компанії, більшу доступність серед потенційних користувачів та відтворює механізм, який дозволяє працювати як водіям евакуаторів, так і допомагати водіям у вирішенні проблем, що виникатимуть відповідно.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ВЕБ-РОЗРОБКА, ЕВАКУАТОРНА СЛУЖБА, JAVASCRIPT, DATABASES

ЗМІСТ

ВСТУП

Актуальність. У наші дні наявність зручного та зрозумілого у використанні веб-застосунку, який би міг врегулювати повноцінне функціонування евакуаторної компанії, є досить важливим, адже, під час війни на території України, значна кількість населення потребує допомоги у транспортуванні пошкодженого авто, його ремонті, забезпеченні армією волонтерською допомогою тощо.

Веб-застосунок, що буде створено у результаті виконання даної кваліфікаційної роботи магістра, зможе заповнити цю потребу та, також, створить певну екосистему, адже буде працювати у парі із вже створеним мобільним додатком та буде забезпечувати доступність, адже з ним можна буде працювати на будь-якому пристрої та ОС відповідно.

Об’єкт дослідження – деяка модель організації повноцінної роботи евакуаторної служби, головна мета якої забезпечити зв'язок між водіями евакуаторів (виконавців) та водіїв авто, які виступають клієнтами відповідно.

Предмет дослідження – механізми, методи та принципи сучасної веб-розробки.

Суперечність, що вирішується у роботі. Огляд рішень, що стосуються даної тематики сказав про наявність гострої необхідності даного програмного рішення, адже програмного продукту, який би міг охопити усі регіони України та забезпечити такий функціонал просто немає. Також, немає програмного рішення, яке було б адаптовано до роботи на будь-яких пристроях, не залежно від виробника, ОС та доступності на ринку відповідно.

Гіпотеза. Якщо розробити застосунок, який буде працювати у середовищі будь-якого існуючого веб-браузера та забезпечуватиме повноцінну роботу евакуаторної компанії, то можна забезпечити, по-перше, більшу доступність серед потенційних користувачів, по-друге, відтворити механізм, який дозволить працювати як водіям евакуаторів, так і допомагати водіям у вирішенні проблем, що виникатимуть.

Новизна даної кваліфікаційної роботи полягає в тому, що на відміну від наявних продуктів на ринку веб-застосунків, інформаційна технологія, що пропонується, здатна створити екосистему, де між собою будуть взаємодіяти як водії евакуаторів, так і водії авто, яким потрібна допомога, забезпечує адаптивність відповідно до пристрою, на якому буде працювати і, як наслідок, доступність та забезпечення потреб населення країни.

Структура. Дана кваліфікаційна робота магістра складається зі вступу, аналітичного огляду досліджуваного предмету щодо актуальності виявленої проблематики та визначенню потенціалу використання веб-рішення, як інформаційної технології для забезпечення повноцінного функціонування евакуаторної компанії; практичної реалізації інформаційної технології; висновків; списку використаних джерел та додатків відповідно.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Дослідження актуальності проблеми

Говорячи про дослідження актуальності проблеми, відразу можна помітити актуальність та необхідність розробки даного інформаційного рішення.

Маємо, що наявність сайту, який буде працювати і підтримуватися на будь-якому пристрої набуває нового сенсу, адже під час війни на території України, велика кількість громадян потребують негайної допомоги у вивезенні власного авто, наприклад, пошкодженого у результаті ворожої атаки, чи ремонті авто волонтерів тощо.

Програмне рішення, що розглядається, також, є актуальним через велику кількість транспортних пригод, що виникають, та стані доріг через наявну скрутну та нестабільну ситуацію.

Дане рішення готове вирішити всі ці питання, а адаптивність, яка забезпечена саме веб-технологіями, поширює коло осіб, які можуть скористатися цією допомогою відповідно.

1.1. Огляд подібних рішень

Після виконання пошукового запиту було отримано незначні результати, адже наявні сайти забезпечують можливість викликати евакуатор лише за телефоном, що там наведено. Це говорить про те, що наразі не існує цілісного рішення, який би засобами веб-розробки міг би забезпечити повноцінну взаємодію між клієнтом та водієм евакуатора, відслідковування результатів виконання тощо.

Рисунок 1.1 – Сайт типу «Landing-page» для виклику евакуатора у Сумській області

Також, можна помітити, що сайти, функціонал яких надає можливість лише викликати евакуатор по телефону, мають також територіальну залежність, тобто можуть працювати лише у певному регіоні країни відповідно.

1.2. Постановка задачі

Згідно до проведеного аналізу, який було реалізовано під час проходження практики, та інформації, яка було виявлена, було складено

Відповідно до матеріалу, наведеного вище, та отриманих знань було складено технічне завдання, як вихідний документ для формування вимог та задач, які потрібно виконати для успішної розробки застосунку.

Говорячи про задачі, поставлені для успішного виконання завдання, то у другому розділі необхідно проаналізувати технології, які будуть використані при розробці програмного продукту, особливості їх роботи та взаємодії з ними. Також, необхідно реалізувати макет сайту, відтворивши його UI-дизайн, відповідно, відтворивши основні секції та блоки, а також, розмітку відповідно.

У третьому розділі основна задача – це розробка сайту, як окремого програмного продукту, налаштування його взаємодії із створеною базою даних, забезпечення взаємодії із картою, реалізованою засобами Google API, та підвищенню доступності та адаптивності сайту відповідно.

1.3. Основні вимоги

Основні вимоги при виконанні розробки веб-продукту для обслуговування евакуаторної служби виглядають наступним чином:

- веб-застосунок (сайт) повинен взаємодіяти із базами даних, що знаходяться на віддаленому сервері;
- користувач повинен мати можливість зареєструватися та в подальшому авторизуватися, використовуючи особисті дані та сформований власний пароль;
- повинно бути реалізовано в зберігання даних, які були введені при реєстрації, а саме особиста інформація та дані, що характеризують авто клієнта;
- забезпечення взаємодії замовника та виконавця шляхом розділення їх на ролі, від вибору якого буде залежати функціонал;
- використання географічної карти для відображення координат користувачів додатку, адреси тощо;
- інтерактивність сайту, перехід на інші вікна, вихід із системи, інформаційна довідка тощо.

Для успішної імплементації сайту, як інформаційної технології, у роботу евакуаторної служби необхідно визначити, також, ключові технічні та функціональні вимоги.

Це дозволить забезпечити високу ефективність взаємодії користувачів, як водіїв евакуаторів, так і клієнтів, та системою відповідно.

Спочатку виділимо, так би мовити, технічні вимоги, реалізація яких необхідна для коректного функціонування продукту, а саме:

- система має бути здатна масштабуватися відповідно до кількості користувачів, яка буде зростати із часом і відповідно із збільшенням обсягу даним;
- важливо забезпечити захист даних користувачів, які вони вводять у процесі реєстрації та подальшої авторизації відповідно;
- сайт повинен забезпечувати доступність та власну адаптивність, а, також, зв'язок із мобільним додатком, роблячи певну екосистему;
- затратна частина на розробку та подальшу роботу повинна бути невелика та бути нижче середнього значення на ринку [3].

Тепер перейдемо до функціональних вимог відповідно. Маємо, що:

- сайт має забезпечувати автоматизацію процесів, а саме автоматичний прорахунок особливостей замовлення, відстань, вартість тощо;
- розроблювана інформаційна технологія повинна підтримувати інтерактивність, дозволяючи як водіям евакуаторів, так і клієнтам, водіям звичайних авто, відслідковувати статус замовлення у режимі реального часу, відмінити замовлення або ж змінити статус замовлення на успішно виконано відповідно [2];
- необхідна, також, наявність взаємодії між наявними ролями, тобто між клієнтом та виконавцем замовлення шляхом дзвінка по телефону або ж повідомленні на пошті.

Перелік цих вимог формує відповідну основу для, перш за все, коректної роботи сайту, його безпеки та адаптивності, і відповідає сучасним стандартам розробки веб-застосунків.

Можна побачити, що постановка задачі успішно сформована, а її зміст побудований на власних ідеях та на аналізі конкурентів та їх функціоналу та особливостей роботи.

2. АНАЛІТИЧНА ЧАСТИНА

2.1. Розробка UI-дизайну

Створення UI-дизайну – процес, основна мета якого відтворити близький до фінального вид сайту, беручи до уваги досвід користувачів, альтеративні програмні рішення, що вже існують, адаптивність та доступність відповідно [24].

Отже, наведемо певні особливості формування правильного дизайну користувача відповідно, такі як:

- чітка структура, використання лише основних елементів, відсутність подвійних стандартів;
- загальний вигляд сайту повинен бути не перевантажений анімацією, адже це негативно впливатиме на його подальшу доступність;
- повинні бути елементи, за якими будуть характеризувати цей продукт, такі як логотип, кольори тощо;
- однакова робота елементів розробки незалежно від вікна чи ролі, що є у додатку [15].

Основні вимоги до розробки UI-дизайну сформовані. Загальний вигляд інтерфейсу мобільного додатку було реалізовано за допомогою сервісу «Figma».

Демонстрацію результатів розробки сайту розпочинає початкова сторінка, яка призначена для входу у систему та переходу на головний екран сайту відповідно.

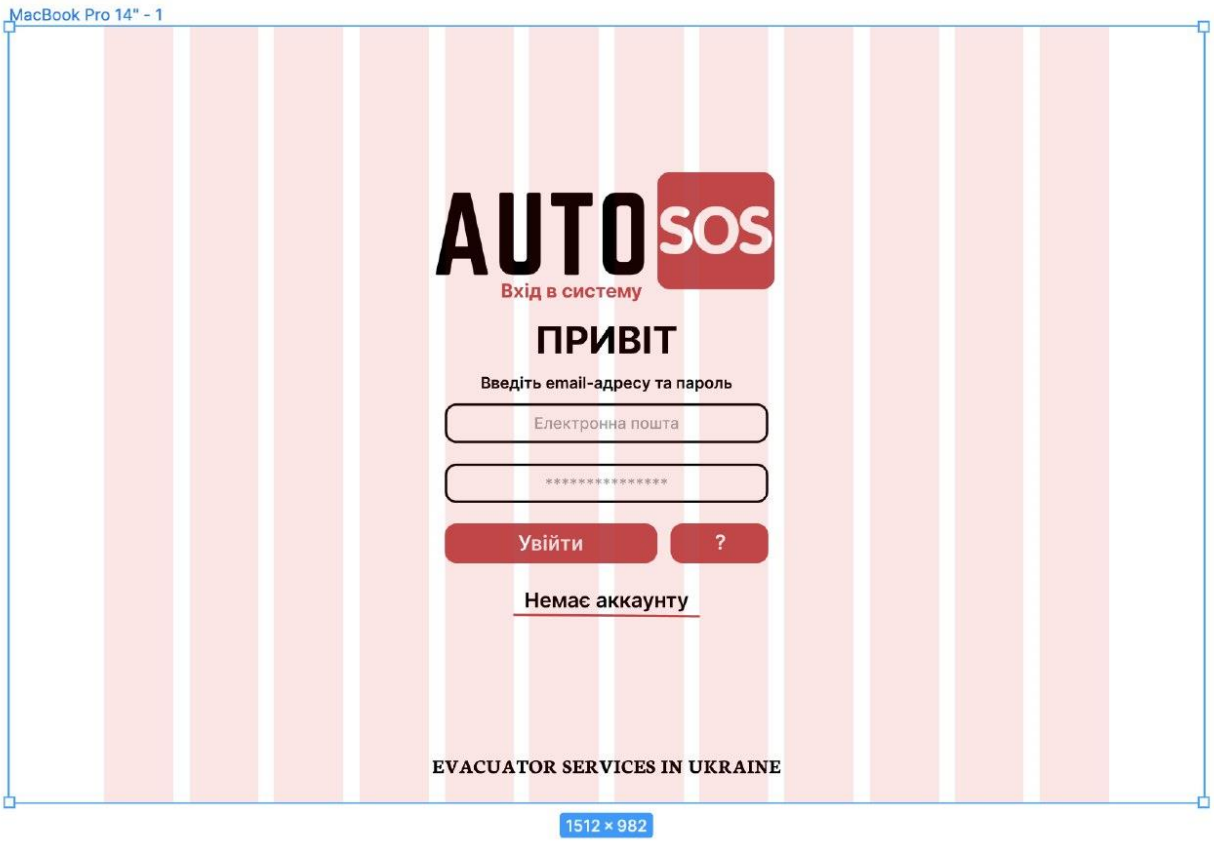


Рисунок 2.1 – Макет початкової сторінки сайту – вхід у систему

Для входу у систему потрібно ввести електронну адресу та пароль. Також, початкова сторінка забезпечує можливість переходу на сторінку реєстрації, якщо користувач новий відповідно.

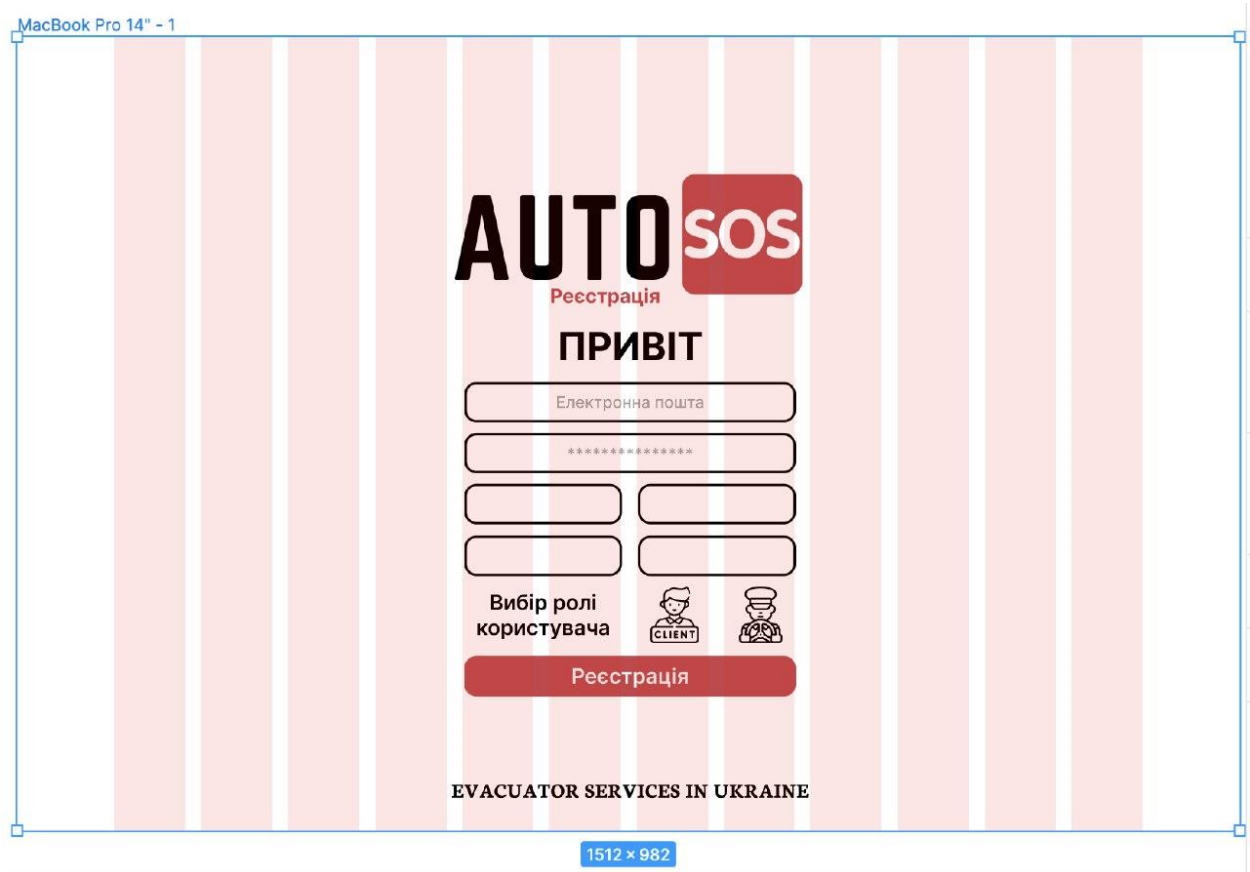


Рисунок 2.2 – Макет сторінки реєстрації нового користувача

У процесі реєстрації потрібно ввести вже не тільки пошту та пароль, а ще виділенні поля для запису особистої інформації та даних про машину. Також, обов’язково потрібно обрати роль, яка говорить нам про те, що новий користувач реєструється як водій евакуатора, чи як клієнт, якому потрібна допомога. Тепер перейдемо до розробки дизайну головного вікна програми.

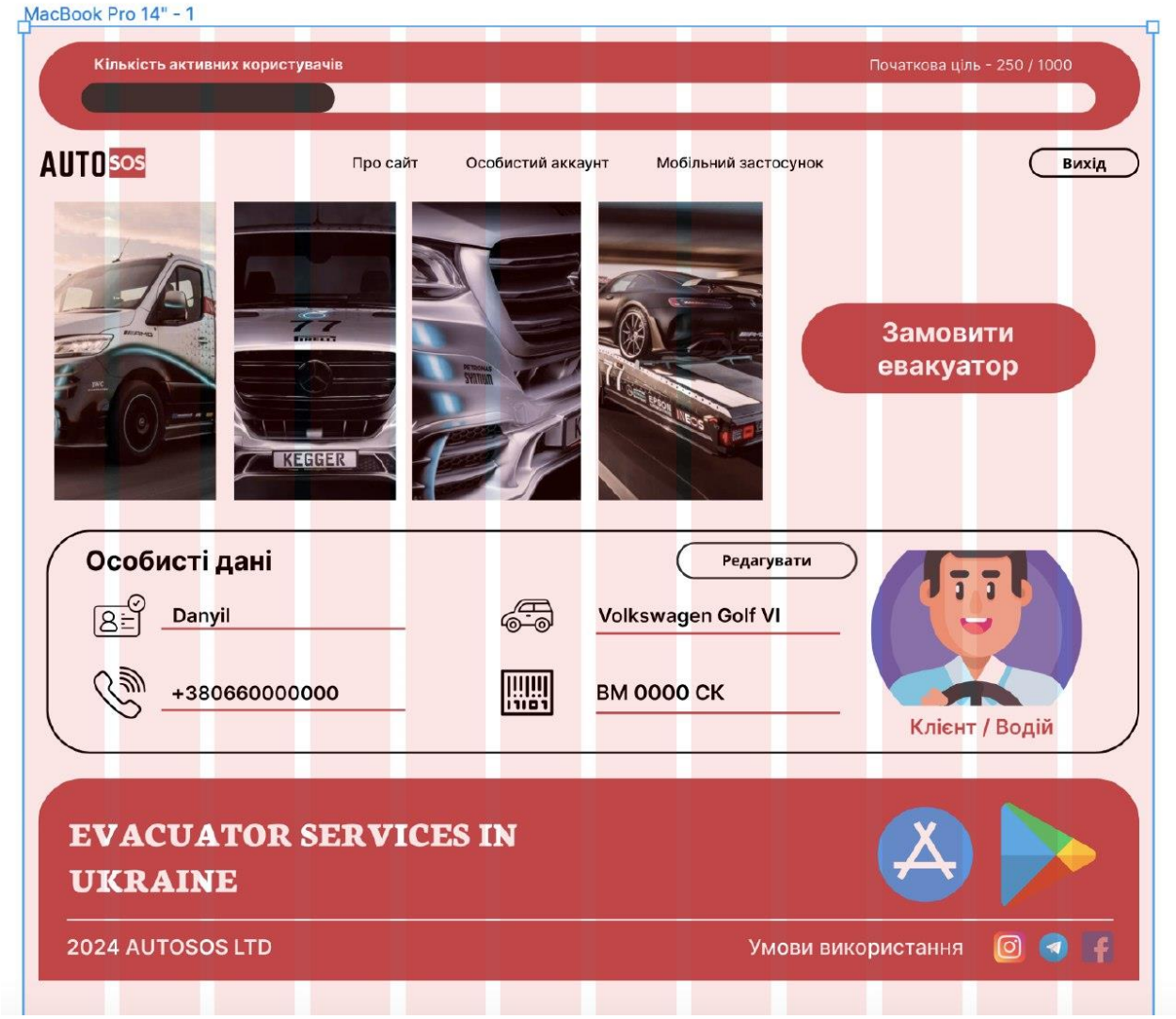


Рисунок 2.3 – Макет головної сторінки сайту

Головна сторінка сайту містить у собі меню додатку, коротку довідку про особисті дані активного користувача, та можливість, якщо ви клієнт, виклику евакуатора відповідно.

Також, наявна можливість редагування особистих даних. Це можна буде зробити на окремій сторінці, яка матиме наступний вигляд.

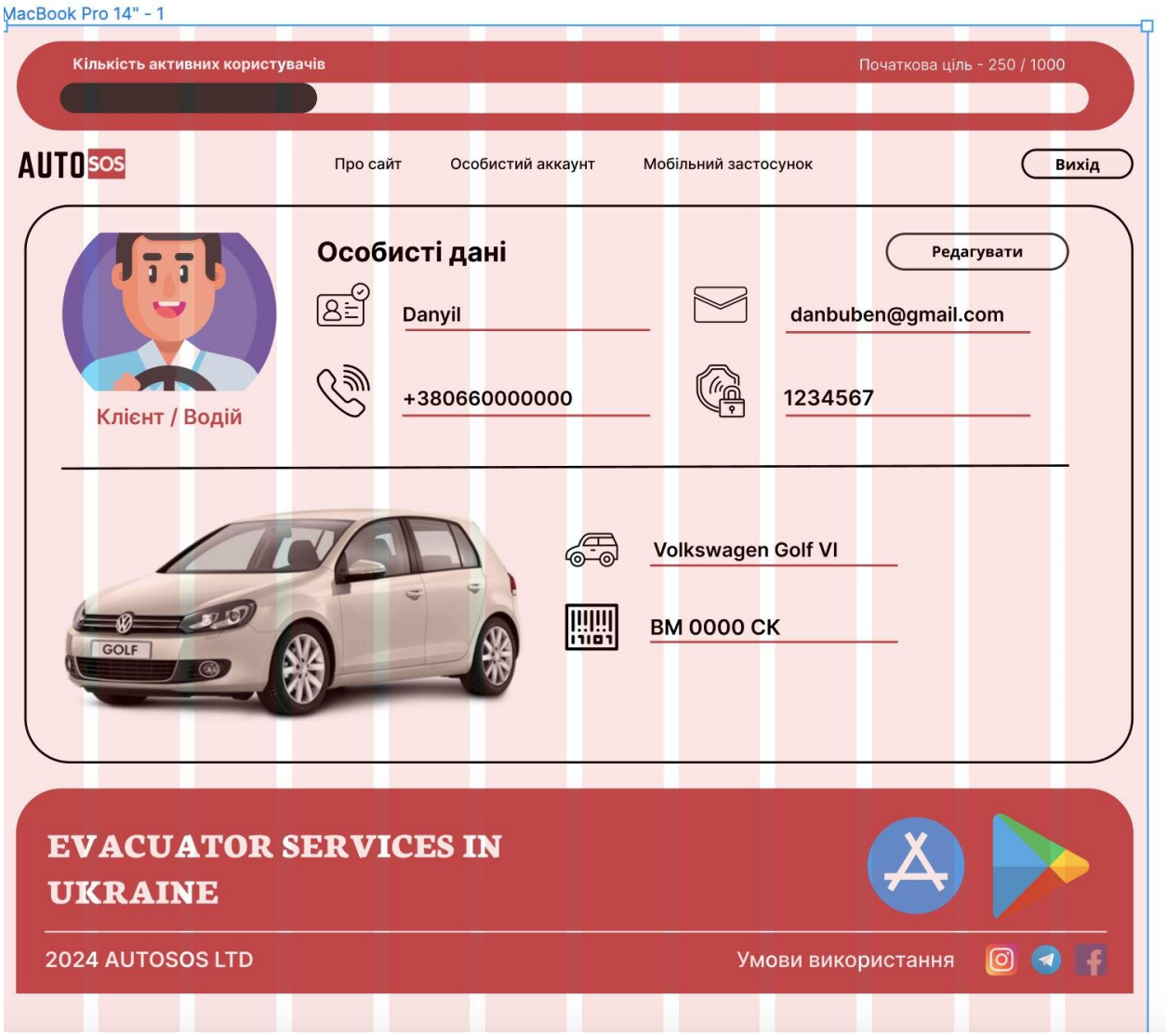


Рисунок 2.4 – Макет особистого кабінету на сайті

Можна побачити, що на окремій сторінці виведені дані, які були введені при реєстрації. Також, є можливість їх редагування засобами спеціальної кнопки.

Тепер продемонструємо дизайн сторінки виклику евакуатора та опишемо секції, які там будуть наявні відповідно.

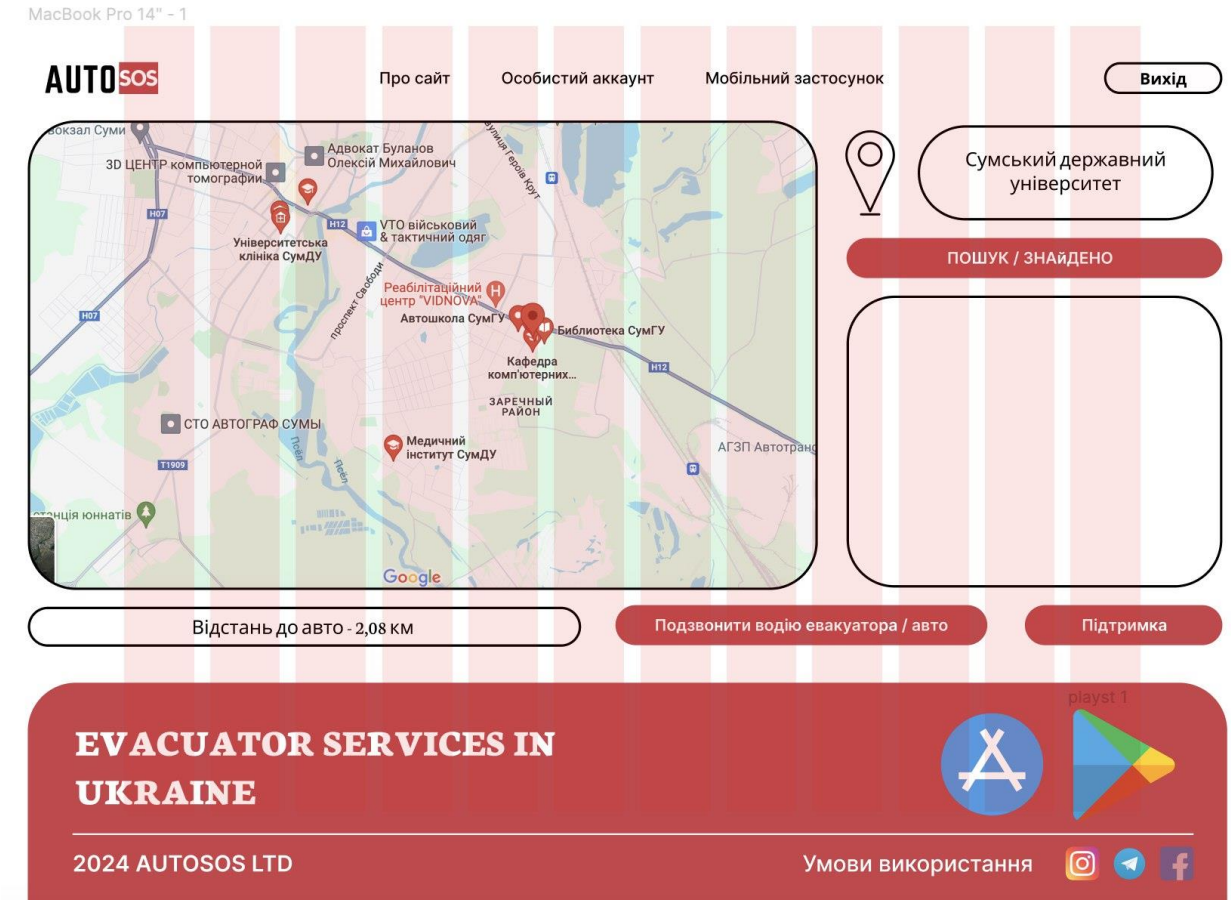


Рисунок 2.5 – Макет сторінки виклику евакуатора та процесу виконання замовлення

Головний функціональний елемент – це карта, яка і забезпечує клієнта та водія інформацією про актуальний стан виконання замовлення.

Також, є відповідна можливість зв’язку з користувачем, поточне місцерозташування, відстань між користувачами застосунку тощо.

Можна побачити, що дизайн основних сторінок сайту із забезпечення коректної роботи евакуаторної кампанії успішно наведено. Перейдемо до формування наступного розділу відповідно.

2.2. Структурно-функціональне моделювання

IDEF0 є стандартизованим підходом до візуального документування процесів за допомогою блоків, що відображають процеси, і стрілок, які показують входи (інформація або ресурси, необхідні для виконання функції),

виходи (результати діяльності), механізми (інструменти виконання функції) та керування (засоби, що впливають на процес) [11].

Основний процес, розглянутий у моделі – «Робота інформаційної технології із надання евакуаторних послуг» (Рисунок 2.6). У якості входів використовуються можливість авторизації та запити водіїв авто.

До механізмів належать учасники системи, сервер обробки та база даних. Процеси керуються правилами використання та інструкціями як для водіїв авто, так і для водіїв евакуаторів. Виходами є успішне виконання замовлення та інформованість обидвох ролей про статус виконання замовлення.

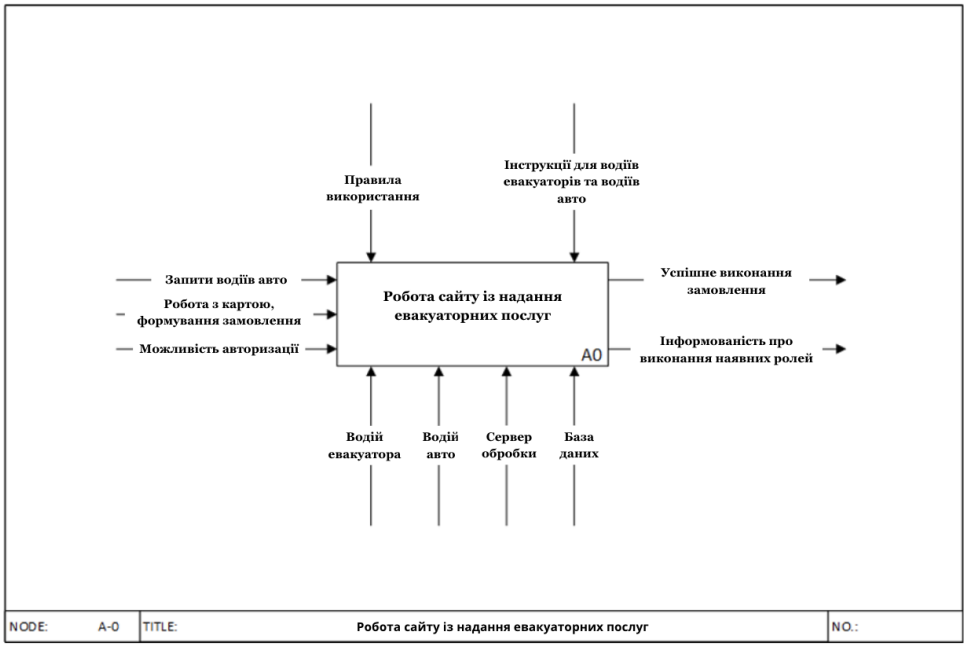


Рисунок 2.6 – Контекстна діаграма процесу «Робота окремого програмного продукту із надання евакуаторних послуг» в нотації IDEF0

У результаті декомпозиції основного процесу були виділені 3 підпроцеси відповідно (Рисунок 2.7).

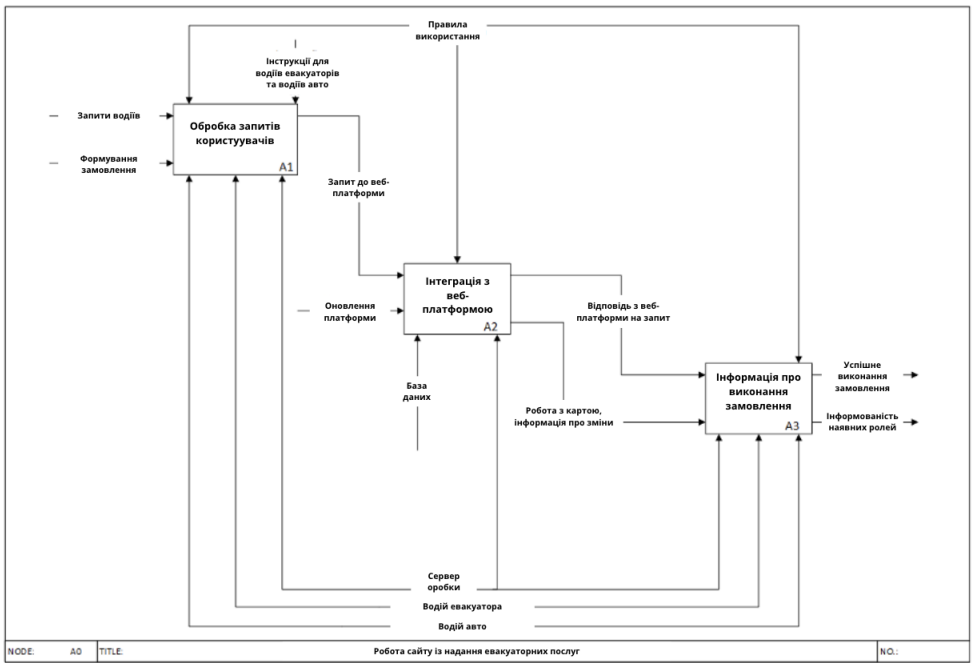


Рисунок 2.7 – Діаграма декомпозиції процесу «Робота сайту із надання евакуаторних послуг» в нотації IDEF0

Підпроцес обробки запитів користувачів включає прийом та аналіз вхідних запитів від учасників системи відповідно. В якості входів для цього процесу використовуються запити водіїв про необхідність допомоги, тобто, про необхідність формування нового замовлення.

Механізм обробки базується на сервері, який використовує алгоритми аналізу тексту для класифікації запиту і визначення подальших дій. Контроль за процесом здійснюється через набір правил і інструкцій, які допомагають визначити необхідність подальшої взаємодії з користувачем відповідно.

Інтеграція з веб-платформою забезпечує синхронізацію даних між обидвома ролями, а саме водієм евакуатора, який готовий виконати замовлення та водієм, якому потрібна допомога.

Вхідні дані включають повідомлення для учасників системи та оновлення на сайті, які обробляються сервером обробки даних. Керування процесом здійснюється через ряд інструкцій і протоколів, що гарантують коректність та безпеку передачі даних. Виходи цього підпроцесу включають

інформування про наявні зміни у роботі з картою та статус виконання замовлення відповідно.

Підпроцес отримання інформації про виконання замовлення має головну мету – постійне відслідковування статуту замовлення у реальному часі. Механізми включають сервер обробки та учасників системи. Керування здійснюється за допомогою правил використання. Виходом є отримання повідомлень кінцевими користувачами, що підтримує інформованість учасників і забезпечує зворотній зв’язок відповідно.

2.3. Огляд технологій

Цей розділ призначений для наведення переліку технологій, які необхідні нам для успішної розробки програмного продукту, який потрібно відтворити.

Мови розмітки HTML + CSS

HTML (мова розмітки гіпертексту) – найбільш відома та найбільш базована мові розмітки відповідно. У сполученні із мовою розмітки CSS, яка використовується для формування зовнішнього вигляду сайту формують каркас для наявних технологій веб-розробки відповідно [3].

Поєднуючи сторінки між собою, використовуючи посилання, що і є особливостями гіпертексту, можна створити власний веб-продукт та розгорнути його вже у мережі Інтернет, надаючи таким чином можливість для користування застосунком [4].

Мова програмування Javascript

JavaScript сьогодні є однією з найпопулярніших мов програмування, і це не дивно. За допомогою JavaScript можна покращувати взаємодію з веб-сайтами, робити їх дизайн більш динамічним та навіть створювати повноцінні додатки [7].

JavaScript – це сценарна мова програмування з динамічною типізацією і прототипною системою.

В JavaScript типи даних можуть змінюватися на льоту. Це означає, що значення можуть змінювати свій тип під час виконання програми. Наприклад, число може стати рядком і навпаки. Це надає розробникам велику гнучкість при написанні коду [4].

JavaScript використовує прототипи для створення об'єктів. Об'єкти-прототипи можуть служити основою для створення інших об'єктів зі схожими характеристиками. Наприклад, прототип може бути загальним описом кола, а екземпляри – конкретними колами з різними радіусами. Прототипне програмування є одним з видів об'єктно-орієнтованого програмування [5].

JavaScript часто використовується для написання сценаріїв (скриптів), які виконують певні дії в заданій послідовності. Це схоже на сценарії у фільмах чи серіалах, де описуються дії акторів. Завдяки цій властивості, JavaScript є потужним інструментом для створення динамічних веб-сторінок і додатків. [6].

Платформа для взаємодії із базами даних Firebase

Firebase – це утиліта, головна мета якої полегшити роботу з базами даних при розробці мобільних застосунків та сайтів різних типів.

Саме її можливості, такі як створення баз даних, можливість хостинку, реалізація авторизації та шифрування даних користувачів зробили її досить популярної серед користувачів відповідно.

Таким чином, можна зробити висновки, що саме засобами функціоналу Firebase Database виникла можливість створити безкоштовно БД, яка зможе гарантувати коректну роботу застосунку відповідно.

Інтерфейси програмного забезпечення Google APIs

Google APIs – програмні рішення, розроблені компанією «Google» для взаємодії із службами програми, які можуть забезпечувати можливість використання ГІС-систем, даних про погоду тощо [8].

Основою для взаємодії клієнта (замовника) із водієм евакуатора (виконавцем) стала саме карта, робота з якою можлива саме задля Google Maps API. Її функціонал було додано до коду продукту за допомогою спеціалізованого ключа, який є унікальним і надається окремо кожному розробнику [17].

Отже, маємо, що для повноцінного функціоналу веб-продукту для відтворення діяльності евакуаторної компанії, необхідне використання мов розмітки веб-продуктів HTML + CSS, універсальна мова програмування Javascript та створена віддалено база даних, яка буде забезпечувати зберігання даних активних користувачів відповідно. Маємо, що огляд технологій та їх можливостей успішно наведено.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1. Розробка моделі інформаційної технології

Модель інформаційної системи є набором даних, що описують певні характеристики та стани програмного продукту або процесу, призначених для відтворення переходів і взаємодій між різними компонентами додатку [11].

Спершу покажемо схему, що ілюструє взаємодію клієнта з сайтом евакуаторної служби, коли він потребує допомоги.

На основі аналізу рисунка 3.1, де представлена схема взаємодії клієнта з сайтом для замовлення послуги евакуатора, можна зрозуміти логіку виконання послідовних етапів цього процесу.

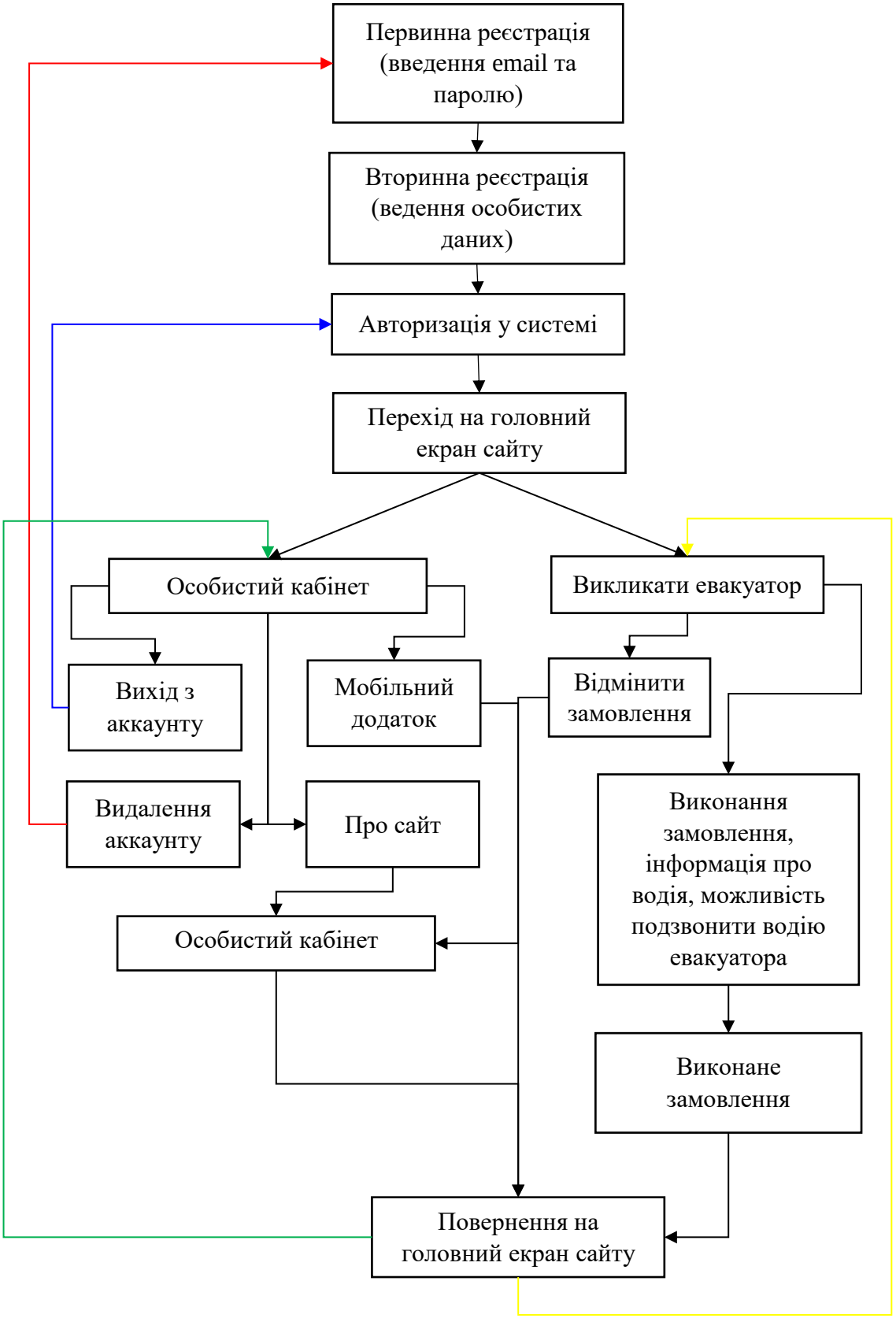


Рисунок 3.1 – Алгоритм роботи клієнта із програмним продуктом [24]

Продублюємо дану схему, але реалізуємо її вже у ролі виконавця, тобто зі сторони водія евакуатора, який розпочинає авторизацію у застосунку.

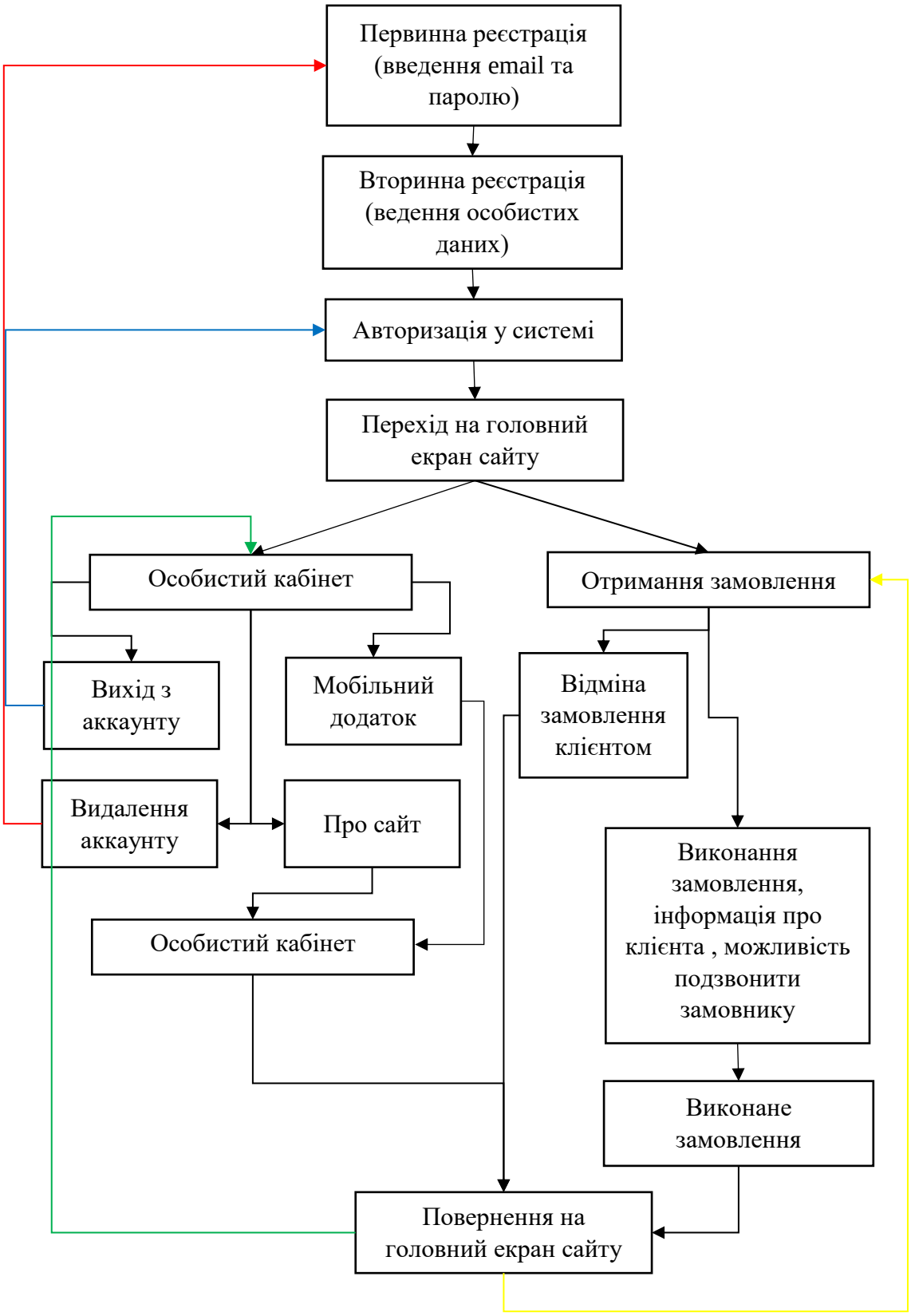


Рисунок 3.2 – Алгоритм роботи виконавця із програмним продуктом [24]

Таким чином, можна зробити висновок, що модель інформаційної системи у вигляді відповідних схем, які відображають логіку роботи сайту як повноцінного веб-продукту, розроблена успішно. Тепер переходимо до формування наступного пункту.

3.2. Розробка бази даних на основі Firebase

Для роботи програмного продукту відповідно до поставлених вимог та задач, необхідна наявність бази даних, яка буде працювати віддалено та забезпечувати можливість авторизації, реєстрації та шифруватиме дані.

Для вирішення поставлених задач було обрано, як програмне рішення, Firebase Realtime Database відповідно, яке було застосовано і при розробці мобільного додатку. Таким чином, ми розбудовуємо екосистему, адже збільшуємо кількість користувачів та використовуємо для цього єдинну СУБД.

Рисунок 3.3 –База даних для роботи із веб-застосунком для обслуговування евакуаторної компанії

Для проведення тестування коректності роботи сайту, база даних була попередньо очищена. Так буде легше відслідковувати запис нових даних та їх подальшу зміну відповідно.

3.3. Код програми та його реалізація

Код блоків програмного продукту, який забезпечує функціонал сайту, його зовнішній вигляд та файли, де було підключено сторонній функціонал, представлений у додатках до звіту відповідно.

Опис основних частин сайту базується на симуляції одночасної роботи клієнта сайту, так і потенційного виконавця замовлення. Цей процес буде супроводжуватися пояснювальними коментарями та відповідними скріншотами.

Спочатку потрібно запустити сайт та перейти на початковий екран сайту, де буде доступна форма авторизації відповідно.

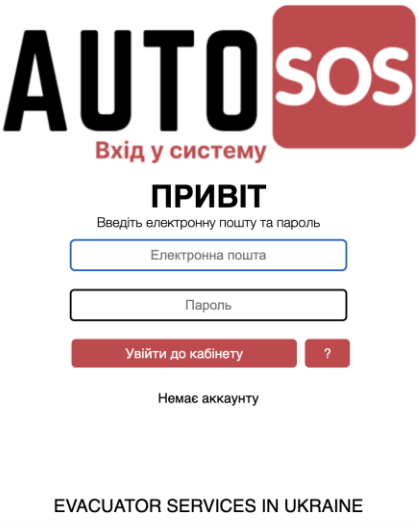


Рисунок 3.4 – Початковий екран сайту для обслуговування евакуаторної служби «AutoSOS»

Так як аккаунтів ще не створено, перейдемо до сторінки реєстрації, натиснувши на посилання «Немає аккаунту» відповідно. Так, почнемо моделювання ситуації із реєстрації аккаунту саме для водія евакуатора.

Рисунок 3.5 – Процес реєстрації нового користувача – водія евакуатора

На рисунку 3.5 розташована форма реєстрації, яка дає можливість новому користувачу ввести особисті дані, обрати роль, та, відповідно до ролі, вказати особливості власного транспортного засобу.

При коректному вводі даних та вибору ролі, після натискання кнопки «Реєстрація» запис про нового користувача повинен зявтитися у нашій базі даних відповідно, а користувача буде переміщено до головної сторінки сайту.

Рисунок 3.6 – Запис у БД після реєстрації нового користувача [24]

Рисунок 3.6 наочно демонструє наявну структуру бази даних та її коректну роботу відповідно.

Отже, було створено нові вузли та організовано ієрархію "Users → Drivers → Driver_ID". Зазначимо, що унікальний ідентифікатор водія генерується автоматично. Також видно, що поля з даними користувача були щойно внесені, як показано на попередньому рисунку [24].

На рисунку 3.6 представлений вузол "Available Drivers", який містить підвузол з ідентифікатором користувача та координатами його поточного місця розташування (довгота і широта). Якщо значення поля "Online Status" є істинним, і водій евакуатора не зайнятий виконанням замовлення, він доступний для нових завдань [24].

Повторний вхід цього ж користувача вже буде успішним, йому не потрібно буде заново реєструватися, а користувача відразу буде переміщено на основне вікно сайту відповідно. Його зовнішній вид продемонстровано на рисунку 3.7

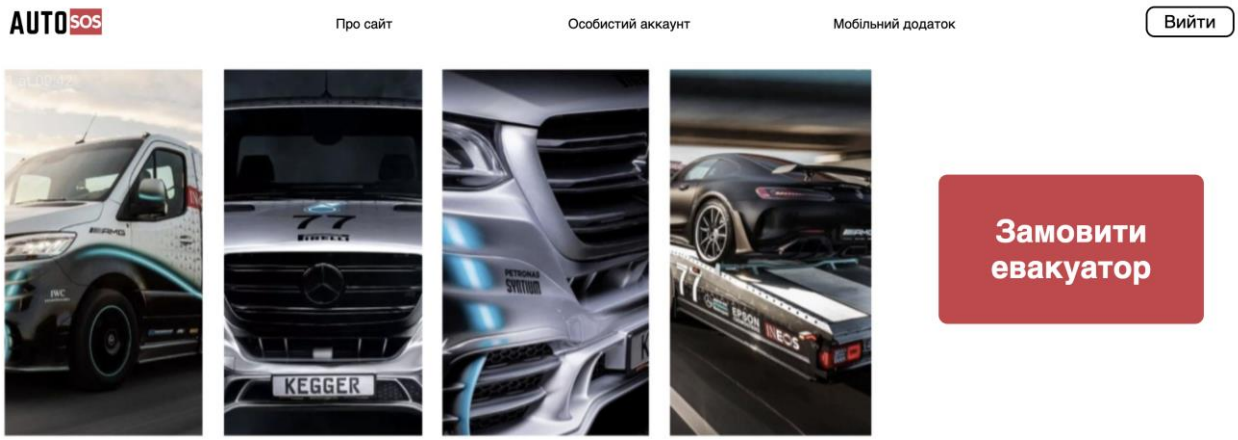


Рисунок 3.7 – Головний екран сайту

Тепер продемонструємо особистий кабінет водія евакуатора і вкажемо наявні там блоки і кнопки відповідно.

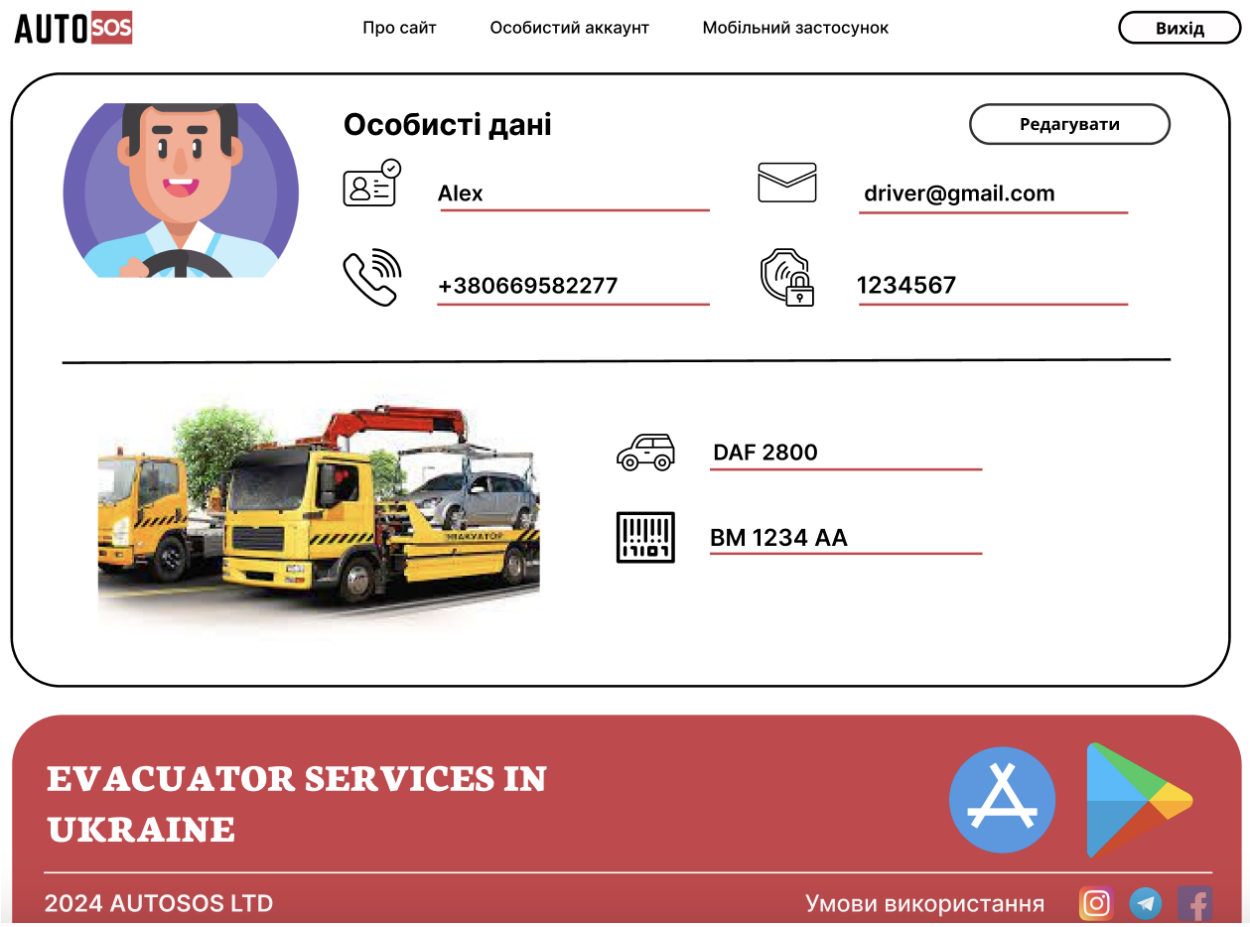
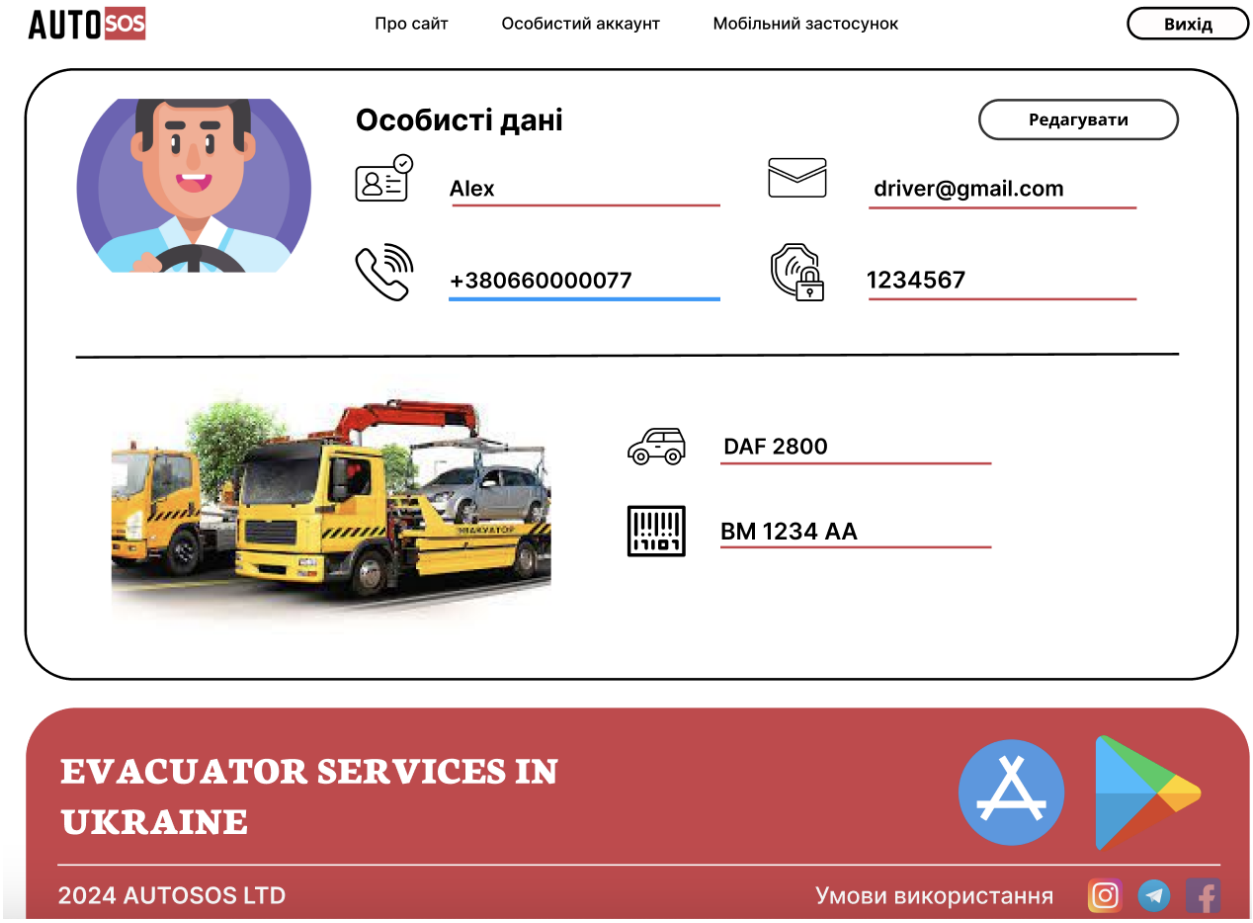


Рисунок 3.8 – Особистий кабінет водія евакуатора, аккаунт якого було зареєстровано

Отже, маємо, що у вікні «Особистий кабінет» сайту, що розроблювався, можна побачити усі поля з особистою інформацією, яка була введена при реєстрації відповідно.

Також, можна вийти з аккаунту та редагувати дані відповідно. Для цього відтворимо ситуацію, коли водію потрібно замінити особистий номер телефону на робочий, наприклад. Маємо наступні результати (Рисунок 3.9)



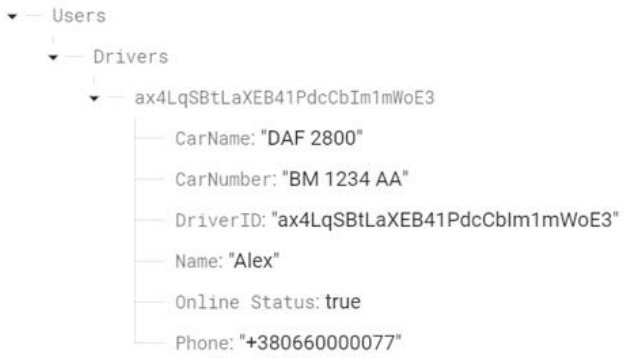


Рисунок 3.9 – Зміна номера телефона на сайті та результати цих дій у БД

При взаємодії із спеціалізованою кнопкою, номер змінився, при цьому інформація про це у БД змінилася у режимі реального часу, а не після завершення сеансу роботи із сайтом.

Для продовження тестування роботи сайту створимо ще одного користувача. Він буде представляти роль замовника, або ж клієнта відповідно.

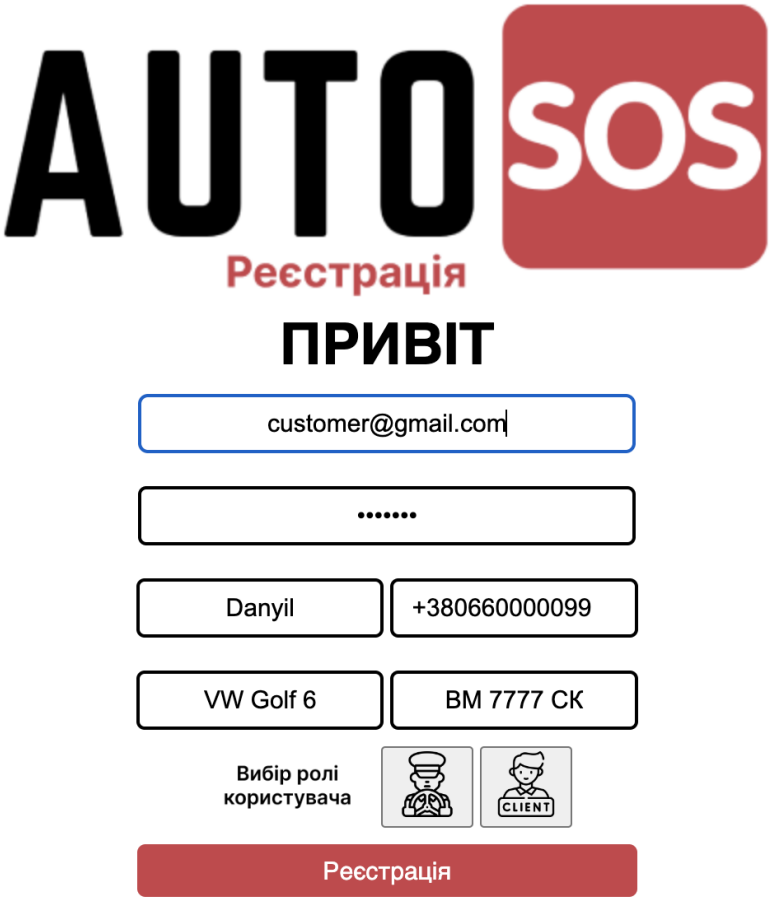


Рисунок 3.10 – Реєстрація нового користувача у ролі клієнта (замовника) сайту

Дані, як і в першому випадку, успішно додані до бази даних, але сформували вже нову асоціативну таблицю «Customers» відповідно.

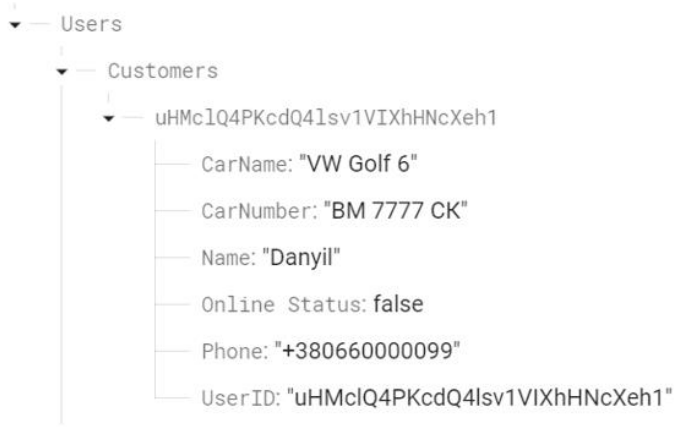


Рисунок 3.11 – Особисті дані щойно зареєстрованого користувача у ролі звичайного водія авто [24]

Можна побачити, що було утворено нові асоціативні зв’язки та відтворена ієрархія типу «Users → Customers → User_ID» відповідно [24].

База даних працює таким чином, що кожному користувачу надається власний унікальний ідентифікатор, за допомогою якого і йде звернення до даних окремого користувача відповідно.

Для подальшої роботи потрібно увімкнути на одному пристрої сайт та зайти у систему у якості водія евакуатора. На іншому пристрої зайти у профіль водія, скориставшись, до прикладу застосунком для штучного ведення власного місцерозташування відповідно.

Рисунок 3.12 – Стан вузлів БД до початку тестування процедури замовлення послуг [24]

Для замовлення профільної послуги, потрібно натиснути відповідне поле. Це призведе до переходу на нове вікно сайту та зміні його елементів відповідно. В базі даних сформуються нові асоціативні зв’язки «Customer Requests» із розташуванням його на момент замовлення.

Рисунок 3.13 – Процес пошук евакуатора та формування нових асоціативних вузлів у БД [24]

Пошук відбувається за географічним принципом. Спочатку береться найменший радіус, який програмно наразі дорівнює 1 кілометру, та з кожним кроком масиву збільшується на одиницю, поки не знайде найближчого виконавця замовлення, тобто водія евакуатора відповідно.

Процедура пошуку сформована таким чином, що від поточного місцезнаходження клієнта формується спеціальне коло (зона) із початковим радіусом в 1 км. Якщо пошук невдалий, радіус з кожним кроком збільшується на 1 км, поки не знайде найближчого водія, який може прийняти замовлення.

Під час активної фази виконання замовлення функціонально неможливо змінювати дані у БД, адже це може вплинути на правильність роботи сайту відповідно.

Рисунок 3.14 – Процес виконання замовлення та взаємодія обидвох можливих рольових моделей

Наведений вище малюнок показує, що при прийнятті замовлення, водій евакуатора, який прийняв замовлення видаляється. У нашому випадку видаляється, також, уся таблиця «Available Drivers», так як водій лише один зареєстрований. Але, паралельно з цим, формується нова таблиця «Driver Working», головна мета якої утворити пару унікальних ключів відповідно.

Після виконання замовлення, всі, так би мовити, додаткові таблиці та асоціативні зв’язки буде видалено, і знову будуть існувати у БД лише дві таблиці, які формуватимуть окремі списки із користувачів відповідно до їх ролі на сайті.

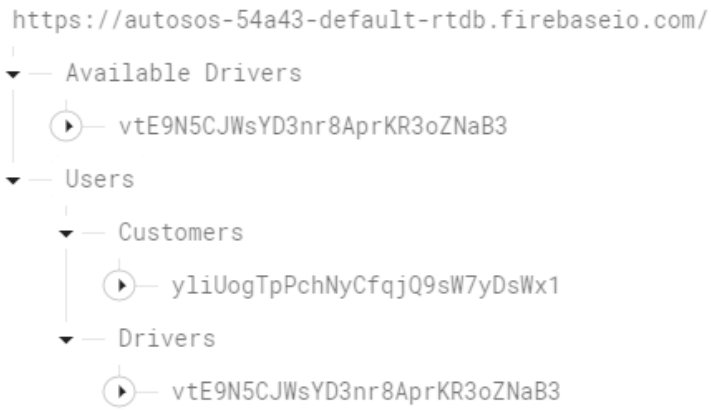


Рисунок 3.15 – Відображення таблиць даних після виконання замовлення

Отже, можна побачити, що опис основних сегментів сайту, їх взаємодія та наявний функціонал успішно наведено. Дана розробка дозволяє виконувати основні задачі для підтримання функціонування евакуаторної компанії, а у зв’язці із мобільним додатком значно розширює кількість потенційних користувачів, надаючи їм доступність та адаптивність.

Це, в свою чергу, говорить про потенційні вдосконалення як функціоналу, так і самої інформаційної технології задля підвищення її стресостійкості та інтерактивності відповідно.

ВИСНОВОК

У ході виконання магістерської кваліфікаційної роботи була розроблена інформаційна технологія, що забезпечує правильну роботу евакуаторної служби, надаючи користувачам максимальну доступність та адаптивність під будь-який пристрій відповідно.

Проведений детальний аналіз існуючих веб-систем та застосунків виявив їх основні недоліки. На основі цього аналізу було визначено ключові напрямки для покращення продукту, що розроблявся.

Розроблена модель формує певну екосистему, яка забезпечує взаємодію між водіями евакуаторів, які надають відповідні послуги, та водіями авто, яким ця допомога потрібна. Інтерфейс сайту розроблено з акцентом на інтуїтивність та доступність, а рішення відтворити продукт саме засобами веб-розробки надає доступність та широку адаптивність відповідно.

Результати реалізації проекту підтверджують гіпотезу про те, що розробка застосунку, який буде працювати у середовищі будь-якого існуючого веб-браузера та забезпечуватиме повноцінну роботу евакуаторної компанії гарантує більшу доступність серед потенційних користувачів та відтворить механізм, який дозволить працювати як водіям евакуаторів, так і допомагати звичайним водіям у вирішенні проблем, що виникатимуть.

У висновку, створена інформаційна технологія представляє повноцінний продукт на ринку України, що зміг би у повній мірі забезпечити роботу евакуаторної компанії у наданні професійних послуг водіям авто відповідно. Цей проект може бути використаний як основа для подальших досліджень і розвитку інноваційних освітніх технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goodman, Danny. Dynamic HTML: The definitive reference: A comprehensive resource for HTML, CSS, DOM & JavaScript. " O'Reilly Media, Inc.", 2002.

2. Alawar, Mariam W., and Samy S. Abu Naser. "CSS-Tutor: An intelligent tutoring system for CSS and HTML." International Journal of Academic Research and Development 2.1 (2017): 94-98.

3. Park, Thomas H., Brian Dorn, and Andrea Forte. "An analysis of HTML and CSS syntax errors in a web development course." ACM Transactions on Computing Education (TOCE) 15.1 (2015): 1-21.

4. Jensen, Simon Holm, Anders Møller, and Peter Thiemann. "Type analysis for JavaScript." Static Analysis: 16th International Symposium, SAS 2009, Los Angeles, CA, USA, August 9-11, 2009. Proceedings 16. Springer Berlin Heidelberg, 2009.

5. Crockford, Douglas. JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.", 2008.

6. Yue, Chuan, and Haining Wang. "Characterizing insecure JavaScript practices on the web." Proceedings of the 18th international conference on World wide web. 2009.

7. Selakovic, Marija, and Michael Pradel. "Performance issues and optimizations in javascript: an empirical study." Proceedings of the 38th International Conference on Software Engineering. 2016.

8. Abiteboul, Serge, Richard Hull, and Victor Vianu. Foundations of databases. Vol. 8. Reading: Addison-Wesley, 1995.

9. Celko, Joe. Joe Celko's SQL for smarties: advanced SQL programming. Elsevier, 2010.

10. Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

11. Chatterjee, Nilanjan, et al. "Real-time communication application based on android using Google firebase." Int. J. Adv. Res. Comput. Sci. Manag. Stud 6.4 (2018).

12. Insfran, Emilio, and Adrian Fernandez. "A systematic review of usability evaluation in web development." Web Information Systems Engineering–WISE 2008 Workshops: WISE 2008 International Workshops, Auckland, New Zealand, September 1-4, 2008. Proceedings 9. Springer Berlin Heidelberg, 2008.

13. Indriana, Marcelli, and Muhammad Leyri Adzani. "UI/UX analysis & design for mobile e-commerce application prototype on Gramedia. com." 2017 4th International Conference on New Media Studies (CONMEDIA). IEEE, 2017.

14. Nasution, Winda Suci Lestari, and Patriot Nusa. "UI/UX design web-based learning application using design thinking method." ARRUS Journal of Engineering and Technology 1.1 (2021): 18-27.

15. Rawls, Carmen G., and Mark A. Turnquist. "Pre-positioning planning for emergency response with service quality constraints." OR spectrum 33 (2011): 481-498.

16. TypeORM - Amazing ORM for TypeScript and JavaScript (ES7, ES6, ES5). Supports MySQL, PostgreSQL, MariaDB, SQLite, MS SQL Server, Oracle, WebSQL databases. Works in NodeJS, Browser, Ionic, Cordova and Electron platforms. [Electronic resource]. URL: https://typeorm.io/ .

17. Express web framework (Node.js/JavaScript) - Learn web development | MDN [Electronic resource]. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs . Express web framework (Node.js/JavaScript) - Learn web development | MDN [Electronic resource]. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs .

18. Pros and Cons of Java Development in 2023 [Electronic resource]. URL: https://www.netguru.com/blog/java-pros-and-cons.

19. Інформаційна стаття на тему «Що таке UX/UI-дизайн і як потрапити до цієї професії».

20. Посилання: https://skillbox.ru/media/design/ux_ui_dizayn_chto_eto_takoe/.

21. Сайт для замовлення евакуатора у місті Суми. Посилання: http://evakuatorsumy.com.ua/

22. Інформаційна стаття на тему «UX/UI-дизайнер: від нуля до профі». Посилання: https://thecentralacademy.com/blog/ux-ui-designer-from-the-scratch-to-the-professional/.

23. Інформаційна стаття на тему «Що таке Firebase?». Посилання: https://avada-media.ua/ua/services/firebase/.

24. Бубенщиков Д.Є. Розробка мобільного додатку на базі ОС Android для успішного функціонування евакуаторної служби. [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 122 – комп`ютерні науки (інформатика) / Д.Є. Бубенщиков; наук. керівник О.Б. Проценко – Суми: СумДУ, 2022 – 127 с.

ДОДАТКИ

У даному розділі наведено вміст усіх додатків, сформованих для викладення відповідних блоків програмної реалізації, необхідної для коректного функціонування сайту, як повноцінного веб-продукту відповідно.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css"/>
  <title>Document</title>
</head>
<body>
  <div class="input">
    <img class="image" src="inputLogo.png"/>
    <div class="text-input1">ПРИВІТ</div>
    <div class="text-input2">Введіть електронну пошту та пароль</div>
    <div class="text-input3">
      <input class="input-email" type="text" placeholder="Електронна пошта"></input>
    </div>
    <div class="text-input3">
      <input class="input-password" type="password" placeholder="Пароль"></input>
    </div>
    <div class="input-button">
      <button class="input-button1">Увійти до кабінету</button>
      <button class="about-button">?</button>
    </div>
    <div class="inputlink">
      <a class="registrationlink" href="registration.html">Немає аккаунту</a>
    </div>
    <div class="inputline"></div>
    <div class="phrase">EVACUATOR SERVICES IN UKRAINE</div>
  </div>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css"/>
  <title>Document</title>
</head>
<body>
  <div class="registration">
    <img class="image" src="registrationLogo.png"/>
    <div class="text-reg">ПРИВІТ</div>
    <div class="text-reg2">
      <input class="reg-email" type="text" placeholder="Електронна пошта"></input>
    </div>
    <div class="text-reg3">
      <input class="reg-password" type="password" placeholder="Пароль"></input>
    </div>
    <div class="text-reg4">
      <input class="reg-name" type="text" placeholder="I`мя"></input>
```

```
      <input class="reg-phone" type="number" placeholder="Номер телефону"></input>
    </div>
    <div class="text-reg5">
      <input class="reg-car" type="text" placeholder="Марка машини"></input>
      <input class="reg-car-number" type="text" placeholder="Номер"></input>
    </div>
    <div class="text-reg6">
      <button class="role"><img class="image-test" src="test.png"/></button>
      <button class="role-driver"><img class="image-dr"
src="roleDriver.png"/></button>
      <button class="role-client"><img class="image-cl"
src="roleClient.png"/></button>
    </div>
    <div class="input-button">
      <button class="reg-button1">Реєстрація</button>
    </div>
  </div>
</body>
</html>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="globals.css" />
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="account">
      <div class="div">
        <div class="overlap">
          <div class="overlap-group"><div class="rectangle"></div></div>
          <div class="text-wrapper">Кількість активних користувачів</div>
          <p class="p">Початкова ціль - 250 / 1000</p>
        </div>
        <div class="overlap-2">
          <div class="rectangle-2"></div>
          <div class="text-wrapper-2">Вихід</div>
        </div>
        <img class="image" src="img/image-1.png" />
        <div class="text-wrapper-3">Про сайт</div>
        <div class="text-wrapper-4">Особистий аккаунт</div>
        <div class="text-wrapper-5">Мобільний застосунок</div>
        <div class="overlap-group-2">
          <div class="text-wrapper-6">Замовити евакуатор</div>
          <div class="rectangle-3"></div>
          <img class="line" src="img/line-9.svg" />
          <div class="text-wrapper-7">1234567</div>
          <div class="text-wrapper-8">driver@gmail.com</div>
          <img class="img" src="img/line-7.svg" />
          <img class="pass" src="img/pass-1.png" />
          <img class="em" src="img/em-1.png" />
          <img class="line-2" src="img/line-4.svg" />
          <div class="text-wrapper-9">BM 1234 AA</div>
          <div class="text-wrapper-10">DAF 2800</div>
          <img class="auto" src="img/auto-1.png" />
          <p  class="element"><span  class="span">+</span>  <span  class="text-wrapper-
11">380660000077</span></p>
          <img class="free-icon-driver" src="img/free-icon-driver-8583437-1.png" />
          <div class="text-wrapper-12">Особисті дані</div>
          <div class="rectangle-4"></div>
          <div class="text-wrapper-13">Редагувати</div>
```

```
          <img class="name" src="img/name-1.png" />
          <img class="phone" src="img/phone-1.png" />
          <div class="text-wrapper-14">Alex</div>
          <img class="line-3" src="img/line-1.svg" />
          <img class="line-4" src="img/line-6.svg" />
          <img class="line-5" src="img/line-3.svg" />
          <img class="carnumber" src="img/carnumber-1.png" />
          <img class="line-6" src="img/line-8.svg" />
          <img class="image-2" src="img/image-2.png" />
        </div>
        <div class="overlap-3">
          <img class="appst" src="img/appst-1.png" />
          <img class="playst" src="img/playst-1.svg" />
          <img class="telegram" src="img/telegram-1.png" />
          <img class="insta" src="img/insta-1.png" />
          <img class="facebook" src="img/facebook-1.png" />
          <div class="text-wrapper-15">Умови використання</div>
          <img class="line-7" src="img/line-5.svg" />
          <div class="text-wrapper-16">2024 AUTOSOS LTD</div>
          <div class="text-wrapper-17">EVACUATOR SERVICES IN UKRAINE</div>
        </div>
      </div>
    </div>
  </body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css"/>
  <title>Document</title>
</head>
<body>
  <div class="count">

  </div>
  <div class="menu">
    <div class="logo">
      <img class="logopic" src="logo.png"/>
    </div>
    <div class="about">
      <a href="">Про сайт</a>
    </div>
    <div class="account">
      <a href="">Особистий аккаунт</a>
    </div>
    <div class="mobileapp">
      <a href="">Мобільний додаток</a>
    </div>
    <div class="exit">
      <button class="exitbutton">Вийти</button>
    </div>
  </div>
  <div class="order">
    <div class="picture1">
      <img class="banner" src="banner.png"/>
    </div>
    <div class="book">
      <button class="bookbutton">Замовити евакуатор</button>
    </div>
```

```
  </div>
  <div class="footer">
    <div class="">EVACUATOR SERVICES IN UKRAINE</div>
    <div class="line"></div>
    <div class="Social">
      <div>
      </div>
      <div>
      </div>
    </div>
  </div>
</body>
</html>
```

.....

```
.input {
  margin: 3% 35% 2% 32%;
  float: left;
  text-align: center;
}

.text-input1 {
  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: #000000;
  font-size: 35px;
  text-align: center;
  letter-spacing: 0;
  line-height: normal;
}

.text-input2 {
  font-family: "Inter-Bold", Helvetica;
  font-weight: 200;
  color: #000000;
  font-size: 15px;
  text-align: center;
  letter-spacing: 0;
  line-height: normal;
}

.input-email {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 60%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.input-password {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 60%;
  height: 30px;
```

```
  text-align: center;
  font-size: 15px;
}

.input-button1 {
  border: none;
  border-radius: 5px;
  margin: 10px 5px;
  background: rgba(204, 64, 73, 1);
  color: white;
  height: 32px;
  width: 50%;
  font-family: "Inter-Bold", Helvetica;
  font-size: 15px;
}

.about-button {
  border: none;
  border-radius: 5px;
  margin: 10px auto;
  background: rgba(204, 64, 73, 1);
  color: white;
  height: 32px;
  width: 10%;
  font-family: "Inter-Bold", Helvetica;
  font-size: 15px;
}

.inputlink {
  margin: 15px auto;
}

.registrationlink {
  color: #000000;
  text-decoration: none;
  font-family: "Inter-Bold", Helvetica;
  font-size: 15px;
}

.phrase {
  margin-top: 100px;
  font-family: "Inter-Bold", Helvetica;
  font-size: 20px;
  font-weight: 400;
}



.registration {
  margin: 3% 35% 2% 32%;
  float: left;
  text-align: center;
}

.text-reg {
  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: #000000;
  font-size: 35px;
  text-align: center;
  letter-spacing: 0;
  line-height: normal;
```

```
}

.reg-email {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 60%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.reg-password {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 60%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.reg-name {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 29%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.reg-phone {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 29%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.reg-car {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 29%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}

.reg-car-number {
  border: 2px solid black;
  border-radius: 5px;
  margin: 10px auto;
  width: 29%;
  height: 30px;
  text-align: center;
  font-size: 15px;
}
```

```
.image-test {
  height: 40px;
}

.role {
  border: none;
  outline: none;
  background-color: white;
  text-align: center;
}

.image-dr {
  height: 40px;
  width: 40px;
}

.image-cl {
  height: 40px;
  width: 40px;
}

.reg-button1 {
  border: none;
  border-radius: 5px;
  margin: 10px 5px;
  background: rgba(204, 64, 73, 1);
  color: white;
  height: 32px;
  width: 62%;
  font-family: "Inter-Bold", Helvetica;
  font-size: 15px;
}

.menu {
  margin-top: 1%;
  display: flex;
}

.logo {
  width: 8%;
  height: 2%;
  position: relative;
  margin: 10px;
  text-align: center;
}

.logopic {
  width: 100%;
}

.about {
  width: 20%;
  margin: 10px;
  margin-left: 10%;
  text-align: center;
  position: relative;
}

.menu a {
  color: #000000;
  text-decoration: none;
  font-family: "Inter-Bold", Helvetica;
```

```
  font-size: 15px;
}

.about a {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-right: -50%;
  transform: translate(-50%, -50%);
}

.account {
  width: 20%;
  margin: 10px;
  text-align: center;
  position: relative;
}

.account a {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-right: -50%;
  transform: translate(-50%, -50%);
}

.mobileapp {
  width: 20%;
  margin: 10px;
  text-align: center;
  position: relative;
}

.mobileapp a {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-right: -50%;
  transform: translate(-50%, -50%);
}

.exit {
  width: 10%;
  height: 2%;
  margin: 10px;
  margin-left: 8%;
  text-align: center;
}

.exitbutton {
  background-color: white;
  border: 2px solid black;
  border-radius: 10px;
  font-size: 20px;
  font-family: "Inter-Bold", Helvetica;
  width: 100px;
  height: 35px;
}

.order {
  margin-top: 1%;
  display: flex;
```

```
}

.picture1 {
  width: 70%;
  text-align: center;
  position: relative;
}

.banner {
  width: 100%;
}

.book {
  width: 30%;
  height: 420px;
  text-align: center;
  position: relative;
}

.bookbutton {
  position: absolute;
  top: 50%;
  left: 50%;
  margin-right: -50%;
  transform: translate(-50%, -50%);

  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: white;
  font-size: 35px;
  text-align: center;
  letter-spacing: 0;
  line-height: normal;
  width: 70%;
  height: 40%;

  background: rgba(204, 64, 73, 1);
  border-radius: 10px;
  border: rgba(204, 64, 73, 1);
}

.footer {
  background-color: rgba(204, 64, 73, 1);
  height: 100px;
}

.account {
  background-color: #ffffff;
  display: flex;
  flex-direction: row;
  justify-content: center;
  width: 100%;
}

.account .div {
  background-color: #ffffff;
  width: 1512px;
  height: 1500px;
  position: relative;
}

.account .overlap {
```

```
  position: absolute;
  width: 1475px;
  height: 118px;
  top: 20px;
  left: 20px;
  background-color: #cc4049;
  border-radius: 60px;
}

.account .overlap-group {
  position: absolute;
  width: 1358px;
  height: 40px;
  top: 55px;
  left: 57px;
  background-color: #ffffff;
  border-radius: 60px;
}

.account .rectangle {
  width: 340px;
  height: 40px;
  background-color: #333333;
  border-radius: 60px;
}

.account .text-wrapper {
  width: 405px;
  height: 30px;
  top: 14px;
  left: 74px;
  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: #ffffff;
  font-size: 20px;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .p {
  width: 307px;
  height: 23px;
  top: 18px;
  left: 1077px;
  font-family: "Inter-Regular", Helvetica;
  font-weight: 400;
  color: #ffffff;
  font-size: 20px;
  text-align: right;
  white-space: nowrap;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .overlap-2 {
  position: absolute;
  width: 148px;
  height: 41px;
  top: 161px;
  left: 1346px;
```

```
}

.account .rectangle-2 {
  position: absolute;
  width: 148px;
  height: 40px;
  top: 1px;
  left: 0;
  background-color: #fffdfd;
  border-radius: 60px;
  border: 3px solid;
  border-color: #000000;
}

.account .text-wrapper-2 {
  width: 148px;
  height: 40px;
  top: 0;
  left: 0;
  font-family: "Neuton-Bold", Helvetica;
  font-weight: 700;
  color: #000000;
  font-size: 20px;
  text-align: center;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .image {
  width: 148px;
  height: 40px;
  top: 162px;
  left: 19px;
  position: absolute;
  object-fit: cover;
}

.account .text-wrapper-3 {
  height: 24px;
  top: 169px;
  left: 439px;
  font-family: "Inter-Medium", Helvetica;
  font-weight: 500;
  color: #000000;
  font-size: 20px;
  text-align: center;
  white-space: nowrap;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .text-wrapper-4 {
  height: 24px;
  top: 169px;
  left: 592px;
  font-family: "Inter-Medium", Helvetica;
  font-weight: 500;
  color: #000000;
  font-size: 20px;
  text-align: center;
```

```
  white-space: nowrap;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .text-wrapper-5 {
  height: 24px;
  top: 169px;
  left: 847px;
  font-family: "Inter-Medium", Helvetica;
  font-weight: 500;
  color: #000000;
  font-size: 20px;
  text-align: center;
  white-space: nowrap;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .overlap-group-2 {
  position: absolute;
  width: 1464px;
  height: 738px;
  top: 236px;
  left: 17px;
  border-radius: 60px;
}

.account .text-wrapper-6 {
  width: 333px;
  height: 99px;
  top: 144px;
  left: 1054px;
  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: #ffffff;
  font-size: 36px;
  text-align: center;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .rectangle-3 {
  position: absolute;
  width: 1464px;
  height: 738px;
  top: 0;
  left: 0;
  background-color: #ffffff;
  border-radius: 60px;
  border: 3px solid;
  border-color: #000000;
}

.account .line {
  position: absolute;
  width: 327px;
  height: 3px;
  top: 576px;
```

```
  left: 839px;
}

.account .text-wrapper-7 {
  position: absolute;
  width: 314px;
  height: 50px;
  top: 229px;
  left: 1015px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 28px;
  letter-spacing: 0;
  line-height: normal;
}

.account .text-wrapper-8 {
  position: absolute;
  width: 312px;
  height: 50px;
  top: 119px;
  left: 1024px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 26px;
  letter-spacing: 0;
  line-height: normal;
}

.account .img {
  position: absolute;
  width: 323px;
  height: 3px;
  top: 277px;
  left: 1018px;
}

.account .pass {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 211px;
  left: 897px;
  object-fit: cover;
}

.account .em {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 97px;
  left: 897px;
  object-fit: cover;
}

.account .line-2 {
  position: absolute;
  width: 327px;
  height: 3px;
  top: 277px;
```

```
  left: 512px;
}

.account .text-wrapper-9 {
  width: 365px;
  height: 50px;
  top: 532px;
  left: 839px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 28px;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .text-wrapper-10 {
  position: absolute;
  width: 360px;
  height: 50px;
  top: 430px;
  left: 843px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 26px;
  letter-spacing: 0;
  line-height: normal;
}

.account .auto {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 421px;
  left: 727px;
  object-fit: cover;
}

.account .element {
  width: 314px;
  height: 50px;
  top: 229px;
  left: 512px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 28px;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .span {
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 28px;
  letter-spacing: 0;
}
```

```
.account .text-wrapper-11 {
  font-size: 26px;
}

.account .free-icon-driver {
  position: absolute;
  width: 289px;
  height: 209px;
  top: 37px;
  left: 60px;
  object-fit: cover;
}

.account .text-wrapper-12 {
  width: 402px;
  height: 50px;
  top: 36px;
  left: 399px;
  font-family: "Inter-Bold", Helvetica;
  font-weight: 700;
  color: #000000;
  font-size: 36px;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .rectangle-4 {
  position: absolute;
  width: 242px;
  height: 50px;
  top: 37px;
  left: 1150px;
  background-color: #ffffff;
  border-radius: 60px;
  border: 3px solid;
  border-color: #333333;
}

.account .text-wrapper-13 {
  width: 242px;
  height: 52px;
  top: 34px;
  left: 1150px;
  font-family: "Neuton-Bold", Helvetica;
  font-weight: 700;
  color: #000000;
  font-size: 20px;
  text-align: center;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .name {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 97px;
  left: 399px;
  object-fit: cover;
}
```

```
.account .phone {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 211px;
  left: 399px;
  object-fit: cover;
}

.account .text-wrapper-14 {
  position: absolute;
  width: 312px;
  height: 50px;
  top: 119px;
  left: 512px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #000000;
  font-size: 26px;
  letter-spacing: 0;
  line-height: normal;
}

.account .line-3 {
  position: absolute;
  width: 323px;
  height: 3px;
  top: 164px;
  left: 516px;
}

.account .line-4 {
  position: absolute;
  width: 323px;
  height: 3px;
  top: 167px;
  left: 1018px;
}

.account .line-5 {
  position: absolute;
  width: 327px;
  height: 3px;
  top: 475px;
  left: 839px;
}

.account .carnumber {
  position: absolute;
  width: 70px;
  height: 70px;
  top: 523px;
  left: 727px;
  object-fit: cover;
}

.account .line-6 {
  position: absolute;
  width: 1330px;
  height: 7px;
  top: 343px;
```

```
  left: 62px;
}

.account .image-2 {
  width: 530px;
  height: 324px;
  top: 369px;
  left: 105px;
  position: absolute;
  object-fit: cover;
}

.account .overlap-3 {
  position: absolute;
  width: 1475px;
  height: 271px;
  top: 1007px;
  left: 19px;
  background-color: #cc4049;
  border-radius: 60px 60px 0px 0px;
}

.account .appst {
  position: absolute;
  width: 140px;
  height: 140px;
  top: 32px;
  left: 1118px;
  object-fit: cover;
}

.account .playst {
  position: absolute;
  width: 140px;
  height: 140px;
  top: 33px;
  left: 1283px;
}

.account .telegram {
  position: absolute;
  width: 42px;
  height: 42px;
  top: 205px;
  left: 1339px;
  object-fit: cover;
}

.account .insta {
  position: absolute;
  width: 42px;
  height: 42px;
  top: 205px;
  left: 1280px;
  object-fit: cover;
}

.account .facebook {
  position: absolute;
  width: 42px;
  height: 42px;
  top: 205px;
```

```
  left: 1398px;
  object-fit: cover;
}

.account .text-wrapper-15 {
  width: 312px;
  height: 42px;
  top: 204px;
  left: 951px;
  font-family: "Inter-Medium", Helvetica;
  font-weight: 500;
  color: #ffffff;
  font-size: 28px;
  position: absolute;
  letter-spacing: 0;
  line-height: normal;
}

.account .line-7 {
  position: absolute;
  width: 1402px;
  height: 2px;
  top: 188px;
  left: 38px;
}

.account .text-wrapper-16 {
  position: absolute;
  width: 291px;
  height: 42px;
  top: 204px;
  left: 38px;
  font-family: "Inter-SemiBold", Helvetica;
  font-weight: 600;
  color: #ffffff;
  font-size: 28px;
  letter-spacing: 0;
  line-height: normal;
}

.account .text-wrapper-17 {
  position: absolute;
  width: 689px;
  height: 140px;
  top: 32px;
  left: 41px;
  font-family: "Neuton-ExtraBold", Helvetica;
  font-weight: 800;
  color: #ffffff;
  font-size: 48px;
  letter-spacing: 0;
  line-height: normal;
}



.......



public class CustomerInfoRegistrationActivity extends AppCompatActivity {
```

```
    EditText      customerInfoName,      customerInfoPhone,      customerInfoCarName,
customerInfoCarNumber;
    Button customerCreateButton;

    private FirebaseAuth mAuth;
    private DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_info_registration);

        customerInfoName = (EditText) findViewById(R.id.customerInfoName);
        customerInfoPhone = (EditText) findViewById(R.id.customerInfoPhone);
        customerInfoCarName = (EditText) findViewById(R.id.customerInfoCarName);
        customerInfoCarNumber = (EditText) findViewById(R.id.customerInfoCarNumber);
        customerCreateButton = (Button) findViewById(R.id.customerCreateButton);

        mAuth = FirebaseAuth.getInstance();
        databaseReference                                                    =
FirebaseDatabase.getInstance().getReference().child("Users").child("Customers");

        customerCreateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                validateAndSaveInfo();
            }
        });
    }

    private void validateAndSaveInfo() {
        if (TextUtils.isEmpty(customerInfoName.getText().toString())) {
            Toast.makeText(this,     "Заповніть     поле     з     Вашим     іменем",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(customerInfoPhone.getText().toString())) {
            Toast.makeText(this,    "Заповніть    поле    з    номером    телефону",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(customerInfoCarName.getText().toString())) {
            Toast.makeText(this,   "Заповніть   поле   інформації   про   Ваше   авто",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(customerInfoCarNumber.getText().toString())) {
            Toast.makeText(this,  "Заповніть  поле  інформації  про  Ваш  номер  авто",
Toast.LENGTH_SHORT).show();
        }
        else {
            HashMap<String, Object> userMap = new HashMap<>();
            userMap.put("UserID", mAuth.getCurrentUser().getUid());
            userMap.put("Name", customerInfoName.getText().toString());
            userMap.put("Phone", customerInfoPhone.getText().toString());
            userMap.put("CarName", customerInfoCarName.getText().toString());
            userMap.put("CarNumber", customerInfoCarNumber.getText().toString());
            userMap.put("Online Status", true);


databaseReference.child(mAuth.getCurrentUser().getUid()).updateChildren(userMap);
            startActivity(new           Intent(CustomerInfoRegistrationActivity.this,
CustomerMapsActivity.class));
        }
    }
```

```
}


public    class    CustomerMapsActivity    extends    FragmentActivity    implements
OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        com.google.android.gms.location.LocationListener {

    private GoogleMap mMap;
    private ActivityCustomerMapsBinding binding;

    TextView locationEditText;
    TextView      driverMapsName,      driverMapsPhoneNumber,      driverMapsCarInfo,
driverMapsCarNumber;
    RelativeLayout driverInfoRelativeLayout, customerUpdateDriverPosition;
    Button     customerMenuButton,     callDriverButton,     calCurrentDriverButton,
customerCancelOrderButton;

    GoogleApiClient googleApiClient;
    Location lastLocation;
    LocationRequest locationRequest;
    Marker driverMarker, pickUpMarker;

    private String customerID;
    private LatLng customerPosition;
    private int radius = 1;
    private Boolean driverFound = false, requestType = false;

    private String driverFoundID;
    private DatabaseReference customerDatabaseRef;
    private DatabaseReference driversAvailableRef;

    private DatabaseReference driversLocationRef;
    private DatabaseReference driversRef;

    private FirebaseAuth mAuth;
    private FirebaseUser currentUser;
    private Boolean currentLogOutDriverStatus;

    private ValueEventListener driverLocationRefListener;
    GeoQuery geoQuery;

    private Boolean currentUserOrderingStatus = false;

    public static String menuCustomerLatitude, menuCustomerLongitude;
    public static Location menuCurrentCustomerCoordinates;
    public static String currentDriverPhoneNumber;

    private String cancelOrderDriverID;
    private LatLng DriverLatLng;

    @SuppressLint("DefaultLocale")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityCustomerMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        locationEditText = (TextView) findViewById(R.id.locationEditText);
        driverMapsName = (TextView) findViewById(R.id.driverMapsName);
```

```
        driverMapsPhoneNumber = (TextView) findViewById(R.id.driverMapsPhoneNumber);
        driverMapsCarInfo = (TextView) findViewById(R.id.driverMapsCarInfo);
        driverMapsCarNumber = (TextView) findViewById(R.id.driverMapsCarNumber);
        driverInfoRelativeLayout                =                (RelativeLayout)
findViewById(R.id.driverInfoRelativeLayout);
        driverInfoRelativeLayout.setVisibility(View.INVISIBLE);
        calCurrentDriverButton = (Button) findViewById(R.id.calCurrentDriverButton);
        calCurrentDriverButton.setVisibility(View.INVISIBLE);
        callDriverButton = (Button) findViewById(R.id.callDriverButton);
        customerMenuButton = (Button) findViewById(R.id.customerMenuButton);
        customerCancelOrderButton                      =                    (Button)
findViewById(R.id.customerCancelOrderButton);
        customerCancelOrderButton.setVisibility(View.INVISIBLE);

        mAuth = FirebaseAuth.getInstance();
        currentUser = mAuth.getCurrentUser();

        customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        customerDatabaseRef                                                    =
FirebaseDatabase.getInstance().getReference().child("Customers Requests");
        driversAvailableRef                                                    =
FirebaseDatabase.getInstance().getReference().child("Available Drivers");
        driversLocationRef                                                     =
FirebaseDatabase.getInstance().getReference().child("Driver Working");

        customerUpdateDriverPosition               =               (RelativeLayout)
findViewById(R.id.customerUpdateDriverPosition);
        customerUpdateDriverPosition.setVisibility(View.INVISIBLE);

        SupportMapFragment         mapFragment         =         (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        callDriverButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (requestType) {
                    requestType = false;
                    GeoFire geofire = new GeoFire(customerDatabaseRef);
                    geofire.removeLocation(customerID);
                    if (pickUpMarker != null) {
                        pickUpMarker.remove();
                    }
                    if (driverMarker != null) {
                        driverMarker.remove();
                    }

                    callDriverButton.setText("Викликати евакуатор");
                    if (driverFound != null) {
                        driversRef = FirebaseDatabase.getInstance().getReference()

.child("Users").child("Drivers").child(driverFoundID).child("CustomerRideID");
                        driversRef.removeValue();
                        driverFoundID = null;
                    }

                    driverFound = false;
                    radius = 1;
                } else {
                    requestType = true;
```

```
                    GeoFire geofire = new GeoFire(customerDatabaseRef);
                    geofire.setLocation(customerID,                                    new
GeoLocation(lastLocation.getLatitude(), lastLocation.getLongitude()));

                    customerPosition    =    new    LatLng(lastLocation.getLatitude(),
lastLocation.getLongitude());
                    pickUpMarker                =                mMap.addMarker(new
MarkerOptions().position(customerPosition).title("Я тут")

.icon(BitmapDescriptorFactory.fromResource(R.drawable.location_image)));

                    callDriverButton.setText("Пошук...");
                    getNearbyDrivers();
                }
            }
        });

        CheckCurrentUserOrdering();
        customerMenuButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentUserOrderingStatus) {
                    Toast toast = Toast.makeText(CustomerMapsActivity.this, "Завершіть
замовлення або відмініть замовлення",
                            Toast.LENGTH_SHORT);
                    toast.setGravity(Gravity.CENTER, 0, 0);
                    toast.show();
                }
                else {
                    Intent    intent    =    new    Intent(CustomerMapsActivity.this,
CustomerMenuActivity.class);
                    startActivity(intent);
                }
            }
        });

        customerCancelOrderButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                driverInfoRelativeLayout.setVisibility(View.INVISIBLE);
                calCurrentDriverButton.setVisibility(View.INVISIBLE);
                customerCancelOrderButton.setVisibility(View.INVISIBLE);
                customerUpdateDriverPosition.setVisibility(View.INVISIBLE);
                callDriverButton.setText("Викликати евакуатор");
                locationEditText.setText("Приємного користування");
                CancelCurrentOrder();
                driverMarker.remove();
                pickUpMarker.remove();
            }
        });

        customerUpdateDriverPosition.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getNearbyDrivers();
            }
        });

        calCurrentDriverButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                int                             permissionCheck                             =
ContextCompat.checkSelfPermission(CustomerMapsActivity.this,
                        Manifest.permission.CALL_PHONE);

                if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(
                            CustomerMapsActivity.this,                                  new
String[]{Manifest.permission.CALL_PHONE},
                            123);
                }
                else {
                    Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel: "
+ currentDriverPhoneNumber));
                    startActivity(intent);
                }
            }
        });
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        buildGoogleApiClient();
        if                                  (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)  !=  PackageManager.PERMISSION_GRANTED  &&
ActivityCompat.checkSelfPermission(this,  Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            return;
        }
        mMap.setMyLocationEnabled(true);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        locationRequest = new LocationRequest();
        locationRequest.setInterval(1000);
        locationRequest.setFastestInterval(1000);
        locationRequest.setPriority(locationRequest.PRIORITY_HIGH_ACCURACY);

        if                                  (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
                !=                PackageManager.PERMISSION_GRANTED               &&
ActivityCompat.checkSelfPermission(this,  Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            return;
        }
        LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient,
locationRequest, this);
    }

    @Override
    public void onConnectionSuspended(int i) {}

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {}

    @SuppressLint("DefaultLocale")
    @Override
    public void onLocationChanged(@NonNull Location location) {
        lastLocation = location;

        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
```

```
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(12));

        menuCustomerLatitude = String.format("%.3f", lastLocation.getLatitude());
        menuCustomerLongitude = String.format("%.3f", lastLocation.getLongitude());
    }

    protected synchronized void buildGoogleApiClient() {
        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        googleApiClient.connect();
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    private void getNearbyDrivers() {
        GeoFire geoFire = new GeoFire(driversAvailableRef);
        GeoQuery           geoQuery           =           geoFire.queryAtLocation(new
GeoLocation(customerPosition.latitude, customerPosition.longitude), radius);
        geoQuery.removeAllListeners();

        geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
            @Override
            public void onKeyEntered(String key, GeoLocation location) {
                if (!driverFound && requestType) {
                    driverFound = true;
                    final String driverFoundID = key;

                    cancelOrderDriverID = key;

                    driversRef                                                    =
FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers").child(d
riverFoundID);
                    HashMap driverMap = new HashMap();
                    driverMap.put("CustomerRideID", customerID);
                    driversRef.updateChildren(driverMap);

                    driverLocationRefListener                                     =
driversLocationRef.child(driverFoundID).child("l").
                            addValueEventListener(new ValueEventListener() {
                                @SuppressLint({"SetTextI18n", "DefaultLocale"})
                                @Override
                                public    void    onDataChange(@NonNull    DataSnapshot
dataSnapshot) {
                                    if (dataSnapshot.exists() && requestType) {
                                        List<Object>       driverLocationMap       =
(List<Object>) dataSnapshot.getValue();
                                        double locationLat = 0;
                                        double locationLng = 0;

                                        callDriverButton.setText("Знайдено");


driverInfoRelativeLayout.setVisibility(View.VISIBLE);

calCurrentDriverButton.setVisibility(View.VISIBLE);
```

```
customerCancelOrderButton.setVisibility(View.VISIBLE);

customerUpdateDriverPosition.setVisibility(View.VISIBLE);
                                    DatabaseReference          reference          =
FirebaseDatabase.getInstance().getReference()

.child("Users").child("Drivers").child(driverFoundID);

                                    reference.addValueEventListener(new
ValueEventListener() {
                                        @Override
                                        public     void     onDataChange(@NonNull
DataSnapshot dataSnapshot) {
                                            if     (dataSnapshot.exists()     &&
dataSnapshot.getChildrenCount() > 0) {
                                                String           name           =
dataSnapshot.child("Name").getValue().toString();
                                                String           phone          =
dataSnapshot.child("Phone").getValue().toString();
                                                String          carName         =
dataSnapshot.child("CarName").getValue().toString();
                                                String         carNumber        =
dataSnapshot.child("CarNumber").getValue().toString();

                                                driverMapsName.setText(name);

driverMapsPhoneNumber.setText(phone);

driverMapsCarInfo.setText(carName);

driverMapsCarNumber.setText(carNumber);

                                                currentDriverPhoneNumber = phone;
                                            }
                                        }

                                        @Override
                                        public     void     onCancelled(@NonNull
DatabaseError databaseError) {}
                                    });

                                    if (driverLocationMap.get(0) != null) {
                                        locationLat                                =
Double.parseDouble(driverLocationMap.get(0).toString());
                                    }
                                    if (driverLocationMap.get(1) != null) {
                                        locationLng                                =
Double.parseDouble(driverLocationMap.get(1).toString());
                                    }
                                    DriverLatLng   =   new   LatLng(locationLat,
locationLng);

                                    if (driverMarker != null) {
                                        driverMarker.remove();
                                    }

                                    Location location1 = new Location("");

location1.setLatitude(customerPosition.latitude);

location1.setLongitude(customerPosition.longitude);
```

```
                                    Location location2 = new Location("");
                                    location2.setLatitude(DriverLatLng.latitude);
location2.setLongitude(DriverLatLng.longitude);
                                    float                Distance                =
location1.distanceTo(location2);
                                    if (Distance < 100 && Distance != 0) {
                                        callDriverButton.setText("Майже        на
місці");
                                    } else {
                                        float roadDistance = Distance / 1000;
                                        locationEditText.setText("Відстань     до
авто " + String.format("%.2f", roadDistance) + " км");
                                    }

                                    driverMarker       =       mMap.addMarker(new
MarkerOptions().position(DriverLatLng)
                                            .title("Ваше                    такси
тут").icon(BitmapDescriptorFactory.fromResource(R.drawable.tow_truck_image)));

                                    if (Distance == 0) {
                                        Intent      resultIntent      =      new
Intent(CustomerMapsActivity.this, CustomerOrderResultActivity.class);
                                        startActivity(resultIntent);
                                        CancelCurrentOrder();
                                    }
                                }
                            }

                            @Override
                            public    void    onCancelled(@NonNull    DatabaseError
databaseError) {}
                        });
                }
            }

            @Override
            public void onKeyExited(String key) {}

            @Override
            public void onKeyMoved(String key, GeoLocation location) {}

            @Override
            public void onGeoQueryReady() {
                if (!driverFound)
                {
                    radius = radius + 1;
                    getNearbyDrivers();
                }
            }

            @Override
            public void onGeoQueryError(DatabaseError error) {}
        });
    }

    private void CheckCurrentUserOrdering() {
        String customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        DatabaseReference                        checkRef                        =
FirebaseDatabase.getInstance().getReference().child("Customers Requests")
```

```
                .child(customerID);

        checkRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    currentUserOrderingStatus = true;
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {}
        });
    }

public class CustomerMenuActivity extends AppCompatActivity {

    EditText      menuCustomerName,      menuCustomerPhone,      menuCustomerCarInfo,
menuCustomerCarNumber;
    ImageView closeCustomerMenuButton, saveCustomerMenuButton;
    TextView latitudeCustomerInfo, longitudeCustomerInfo;
    RelativeLayout   exitCustomerButton,   deleteCustomerButton,   siteCustomerButton,
aboutCustomerButton;

    private FirebaseAuth mAuth;
    private DatabaseReference databaseReference, driverAvailableRef;

    private Boolean currentLogOutCustomerStatus = false;
    private Boolean currentDriverWorkingStatus = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_menu);

        menuCustomerName = (EditText) findViewById(R.id.menuCustomerName);
        menuCustomerPhone = (EditText) findViewById(R.id.menuCustomerPhone);
        menuCustomerCarInfo = (EditText) findViewById(R.id.menuCustomerCarInfo);
        menuCustomerCarNumber = (EditText) findViewById(R.id.menuCustomerCarNumber);

        closeCustomerMenuButton                    =                    (ImageView)
findViewById(R.id.closeCustomerMenuButton);
        saveCustomerMenuButton                    =                    (ImageView)
findViewById(R.id.saveCustomerMenuButton);

        latitudeCustomerInfo = (TextView) findViewById(R.id.latitudeCustomerInfo);
        latitudeCustomerInfo.setText(CustomerMapsActivity.menuCustomerLatitude);
        longitudeCustomerInfo = (TextView) findViewById(R.id.longitudeCustomerInfo);
        longitudeCustomerInfo.setText(CustomerMapsActivity.menuCustomerLongitude);

        exitCustomerButton = (RelativeLayout) findViewById(R.id.exitCustomerButton);
        deleteCustomerButton                    =                    (RelativeLayout)
findViewById(R.id.deleteCustomerButton);
        siteCustomerButton = (RelativeLayout) findViewById(R.id.siteCustomerButton);
        aboutCustomerButton                    =                    (RelativeLayout)
findViewById(R.id.aboutCustomerButton);

        mAuth = FirebaseAuth.getInstance();

        databaseReference                                                    =
FirebaseDatabase.getInstance().getReference().child("Users").child("Customers");
        getUserInformation();
```

```
        CheckCurrentUserOrdering();

        closeCustomerMenuButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent    backIntent    =    new    Intent(CustomerMenuActivity.this,
CustomerMapsActivity.class);
                startActivity(backIntent);
            }
        });

        saveCustomerMenuButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                validateAndSaveInfo();
            }
        });

        exitCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentDriverWorkingStatus) {
                    Toast.makeText(CustomerMenuActivity.this,     "Завершіть     Ваше
замовлення", Toast.LENGTH_SHORT).show();
                }
                else {
                    currentLogOutCustomerStatus = true;
                    LogoutCustomer();
                    DisconnectCustomer();
                    mAuth.signOut();
                }
            }
        });

        deleteCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentDriverWorkingStatus) {
                    Toast.makeText(CustomerMenuActivity.this,     "Завершіть     Ваше
замовлення", Toast.LENGTH_SHORT).show();
                }
                else {
                    removeCurrentCustomer();
                    LogoutCustomer();
                }
            }
        });

        siteCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent    siteIntent    =    new    Intent(CustomerMenuActivity.this,
SiteActivity.class);
                siteIntent.putExtra("type", "Customer");
                startActivity(siteIntent);
            }
        });

        aboutCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                Intent    aboutIntent    =    new    Intent(CustomerMenuActivity.this,
AboutActivity.class);
                aboutIntent.putExtra("type", "Customer");
                startActivity(aboutIntent);
            }
        });

    }

    private void validateAndSaveInfo() {
        if (TextUtils.isEmpty(menuCustomerName.getText().toString())) {
            Toast.makeText(this, "Заповніть поле Name", Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(menuCustomerPhone.getText().toString())) {
            Toast.makeText(this, "Заповніть поле Phone", Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(menuCustomerCarInfo.getText().toString())) {
            Toast.makeText(this, "Заповніть поле Auto", Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(menuCustomerCarNumber.getText().toString())) {
            Toast.makeText(this, "Заповніть поле Auto", Toast.LENGTH_SHORT).show();
        }
        else {
            HashMap<String, Object> userMap = new HashMap<>();
            userMap.put("UserID", mAuth.getCurrentUser().getUid());
            userMap.put("Name", menuCustomerName.getText().toString());
            userMap.put("Phone", menuCustomerPhone.getText().toString());
            userMap.put("CarName", menuCustomerCarInfo.getText().toString());
            userMap.put("CarNumber", menuCustomerCarNumber.getText().toString());


databaseReference.child(mAuth.getCurrentUser().getUid()).updateChildren(userMap);
            startActivity(new                        Intent(CustomerMenuActivity.this,
CustomerMapsActivity.class));
        }
    }

    private void getUserInformation() {


databaseReference.child(mAuth.getCurrentUser().getUid()).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists() && snapshot.getChildrenCount() > 0) {
                    String name = snapshot.child("Name").getValue().toString();
                    String phone = snapshot.child("Phone").getValue().toString();
                    String carInfo = snapshot.child("CarName").getValue().toString();
                    String                          carNumber                          =
snapshot.child("CarNumber").getValue().toString();

                    menuCustomerName.setText(name);
                    menuCustomerPhone.setText(phone);
                    menuCustomerCarInfo.setText(carInfo);
                    menuCustomerCarNumber.setText(carNumber);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {}
        });
    }
```

```
    private void DisconnectCustomer() {
        String customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();

        DatabaseReference                    customerDatabaseRef                    =
FirebaseDatabase.getInstance().getReference()
                .child("Users").child("Customers").child(customerID).child("Online
Status");
        customerDatabaseRef.setValue(false);

        DatabaseReference                    customerRequestRef                    =
FirebaseDatabase.getInstance().getReference().child("Customers Requests");
        GeoFire geoFire = new GeoFire(customerRequestRef);
        geoFire.removeLocation(customerID);
    }

    private void LogoutCustomer() {
        Intent    roleSelectionIntent    =    new    Intent(CustomerMenuActivity.this,
RoleSelectionActivity.class);
        startActivity(roleSelectionIntent);
        finish();
    }

    private void removeCurrentCustomer() {

        String customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        Task<Void>                          removeReference                          =
FirebaseDatabase.getInstance().getReference().child("Users")
                .child("Customers").child(customerID).removeValue();

        FirebaseUser currentUser = mAuth.getCurrentUser();
        currentUser.delete();
    }

    private void CheckCurrentUserOrdering() {
        String customerID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        DatabaseReference                          checkRef                          =
FirebaseDatabase.getInstance().getReference().child("Customers Requests")
                .child(customerID);

        checkRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    currentDriverWorkingStatus = true;
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {}
        });
    }
}


public class CustomerRegistrationActivity extends AppCompatActivity {

    EditText registrationEmailCustomerEditText, registrationPasswordCustomerEditText;
    Button registrationCustomerButton;

    FirebaseAuth firebaseAuth;
    DatabaseReference customerDatabaseRef;
```

```
    String onlineCustomerID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_registration);

        registrationEmailCustomerEditText                =                (EditText)
findViewById(R.id.registrationEmailCustomerEditText);
        registrationPasswordCustomerEditText               =               (EditText)
findViewById(R.id.registrationPasswordCustomerEditText);
        registrationCustomerButton                      =                  (Button)
findViewById(R.id.registrationCustomerButton);

        firebaseAuth = FirebaseAuth.getInstance();

        registrationCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if
(TextUtils.isEmpty(registrationEmailCustomerEditText.getText().toString())) {
                    Toast.makeText(CustomerRegistrationActivity.this, "Заповніть поле
email-адреси",
                            Toast.LENGTH_SHORT).show();
                }
                else                                                                if
(TextUtils.isEmpty(registrationPasswordCustomerEditText.getText().toString())) {
                    Toast.makeText(CustomerRegistrationActivity.this, "Заповніть поле
з паролем",
                            Toast.LENGTH_SHORT).show();
                }
                else {
                    String                          email                          =
registrationEmailCustomerEditText.getText().toString();
                    String                         password                        =
registrationPasswordCustomerEditText.getText().toString();
                    RegistrationCustomerMethod(email, password);
                }
            }
        });
    }

    private void RegistrationCustomerMethod(String email, String password) {
        firebaseAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {

                    onlineCustomerID = firebaseAuth.getCurrentUser().getUid();
                    customerDatabaseRef                                             =
FirebaseDatabase.getInstance().getReference()

.child("Users").child("Customers").child(onlineCustomerID);

                    Intent             mainScreenIntent             =             new
Intent(CustomerRegistrationActivity.this, CustomerInfoRegistrationActivity.class);
                    startActivity(mainScreenIntent);

                    Toast.makeText(CustomerRegistrationActivity.this,     "Реєстрація
успішна",
                            Toast.LENGTH_SHORT).show();
```

```
                }
                else {
                    Toast.makeText(CustomerRegistrationActivity.this,        "Помилка
реєстрації нового користувача",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}


public class CustomerWelcomeActivity extends AppCompatActivity {

    EditText editTextCustomerEmail, editTextCustomerPassword;
    Button signInCustomerButton, roleBackCustomerButton, signUpCustomerButton;

    FirebaseAuth firebaseAuth;
    DatabaseReference customerDatabaseRef;
    String onlineCustomerID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_welcome);

        editTextCustomerEmail = (EditText) findViewById(R.id.editTextCustomerEmail);
        editTextCustomerPassword                    =                    (EditText)
findViewById(R.id.editTextCustomerPassword);
        signInCustomerButton = (Button) findViewById(R.id.signInCustomerButton);
        signUpCustomerButton = (Button) findViewById(R.id.signUpCustomerButton);
        roleBackCustomerButton = (Button) findViewById(R.id.roleBackCustomerButton);

        firebaseAuth = FirebaseAuth.getInstance();

        signInCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = editTextCustomerEmail.getText().toString();
                String password = editTextCustomerPassword.getText().toString();
                SignInCustomerMethod(email, password);
            }
        });

        signUpCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent registrationIntent = new Intent(CustomerWelcomeActivity.this,
CustomerRegistrationActivity.class);
                startActivity(registrationIntent);
            }
        });

        roleBackCustomerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent roleSelectionIntent = new Intent(CustomerWelcomeActivity.this,
RoleSelectionActivity.class);
                startActivity(roleSelectionIntent);
            }
        });
    }
```

```
    private void SignInCustomerMethod(String email, String password) {
        firebaseAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(CustomerWelcomeActivity.this,   "Вхід   успішний",
Toast.LENGTH_SHORT).show();

                    onlineCustomerID = firebaseAuth.getCurrentUser().getUid();
                    customerDatabaseRef                                                =
FirebaseDatabase.getInstance().getReference()

.child("Users").child("Customers").child(onlineCustomerID).child("Online Status");
                    customerDatabaseRef.setValue(true);

                    Intent mainScreenIntent = new Intent(CustomerWelcomeActivity.this,
CustomerMapsActivity.class);
                    startActivity(mainScreenIntent);
                }
                else {
                    Toast.makeText(CustomerWelcomeActivity.this, "Помилка входу",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}


public class DriverInfoRegistrationActivity extends AppCompatActivity {

    EditText driverInfoName, driverInfoPhone, driverInfoCarName, driverInfoCarNumber;
    Button driverCreateButton;

    private FirebaseAuth mAuth;
    private DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_driver_info_registration);

        driverInfoName = (EditText) findViewById(R.id.driverInfoName);
        driverInfoPhone = (EditText) findViewById(R.id.driverInfoPhone);
        driverInfoCarName = (EditText) findViewById(R.id.driverInfoCarName);
        driverInfoCarNumber = (EditText) findViewById(R.id.driverInfoCarNumber);
        driverCreateButton = (Button) findViewById(R.id.driverCreateButton);

        mAuth = FirebaseAuth.getInstance();
        databaseReference                                                              =
FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers");

        driverCreateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                validateAndSaveInfo();
            }
        });
    }
```

```
    private void validateAndSaveInfo() {
        if (TextUtils.isEmpty(driverInfoName.getText().toString())) {
            Toast.makeText(this,     "Заповніть     поле     з     Вашим     іменем",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(driverInfoPhone.getText().toString())) {
            Toast.makeText(this,    "Заповніть    поле    з    номером    телефону",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(driverInfoCarName.getText().toString())) {
            Toast.makeText(this,  "Заповніть  поле  інформації  про  Ваш  евакуатор",
Toast.LENGTH_SHORT).show();
        }
        else if (TextUtils.isEmpty(driverInfoCarNumber.getText().toString())) {
            Toast.makeText(this,  "Заповніть  поле  інформації  про  Ваш  номер  авто",
Toast.LENGTH_SHORT).show();
        }
        else {
            HashMap<String, Object> userMap = new HashMap<>();
            userMap.put("DriverID", mAuth.getCurrentUser().getUid());
            userMap.put("Name", driverInfoName.getText().toString());
            userMap.put("Phone", driverInfoPhone.getText().toString());
            userMap.put("CarName", driverInfoCarName.getText().toString());
            userMap.put("CarNumber", driverInfoCarNumber.getText().toString());
            userMap.put("Online Status", true);


databaseReference.child(mAuth.getCurrentUser().getUid()).updateChildren(userMap);
            startActivity(new             Intent(DriverInfoRegistrationActivity.this,
DriverMapsActivity.class));
        }
    }
}


public     class     DriverMapsActivity     extends     FragmentActivity     implements
OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        com.google.android.gms.location.LocationListener {
    private GoogleMap mMap;
    private ActivityDriverMapsBinding binding;

    GoogleApiClient googleApiClient;
    Location lastLocation;
    LocationRequest locationRequest;
    Marker driverPickUpMarker;

    private FirebaseAuth mAuth;
    private FirebaseUser currentUser;
    private Boolean currentLogOutDriverStatus = false;
    private DatabaseReference assignedCustomerRef, assignedCustomerPositionRef;
    private String driverID, customerID = "";

    private ValueEventListener assignedCustomerPositionListener;

    public static String menuDriverLatitude, menuDriverLongitude;
    public static Location menuCurrentDriverCoordinates;

    TextView   locationDriverTextView,   customerMapsName,   customerMapsPhoneNumber,
customerMapsCarInfo, customerMapsCarNumber;
    Button driverMenuButton, calCurrentCustomerButton;
```

```
    RelativeLayout customerInfoRelativeLayout;

    private Boolean currentUserOrderingStatus = false;

    public static String currentCustomerPhoneNumber;
    public static float roadDriverCustomerDistance;

    private LatLng driverCurrentPosition;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityDriverMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        mAuth = FirebaseAuth.getInstance();
        currentUser = mAuth.getCurrentUser();
        driverID = mAuth.getCurrentUser().getUid();

        locationDriverTextView                        =                        (TextView)
findViewById(R.id.locationDriverTextView);
        customerMapsName = (TextView) findViewById(R.id.customerMapsName);
        customerMapsPhoneNumber                        =                        (TextView)
findViewById(R.id.customerMapsPhoneNumber);
        customerMapsCarInfo = (TextView) findViewById(R.id.customerMapsCarInfo);
        customerMapsCarNumber = (TextView) findViewById(R.id.customerMapsCarNumber);

        calCurrentCustomerButton                          =                         (Button)
findViewById(R.id.calCurrentCustomerButton);
        calCurrentCustomerButton.setVisibility(View.INVISIBLE);
        driverMenuButton = (Button) findViewById(R.id.driverMenuButton);

        customerInfoRelativeLayout                  =                  (RelativeLayout)
findViewById(R.id.customerInfoRelativeLayout);
        customerInfoRelativeLayout.setVisibility(View.INVISIBLE);

        SupportMapFragment         mapFragment         =         (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        CheckCurrentUserOrdering();
        driverMenuButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentUserOrderingStatus) {
                    Toast toast = Toast.makeText(DriverMapsActivity.this, "Завершіть
Ваше замовлення",
                            Toast.LENGTH_SHORT);
                    toast.setGravity(Gravity.CENTER, 0, 0);
                    toast.show();
                }
                else {
                    Intent    intent    =    new    Intent(DriverMapsActivity.this,
DriverMenuActivity.class);
                    startActivity(intent);
                }
            }
        });

        calCurrentCustomerButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                int                         permissionCheck                         =
ContextCompat.checkSelfPermission(DriverMapsActivity.this,
                        Manifest.permission.CALL_PHONE);

                if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(
                            DriverMapsActivity.this,                                new
String[]{Manifest.permission.CALL_PHONE},
                            123);
                }
                else {
                    Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel: "
+ currentCustomerPhoneNumber));
                    startActivity(intent);
                }
            }
        });

        getAssignedCustomerRequest();
    }

    private void getAssignedCustomerRequest() {
        assignedCustomerRef                                                         =
FirebaseDatabase.getInstance().getReference().child("Users")
                .child("Drivers").child(driverID).child("CustomerRideID");

        assignedCustomerRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    customerID = snapshot.getValue().toString();

                    getAssignedCustomerPosition();
                }
                else {
                    customerID = "";
                    customerInfoRelativeLayout.setVisibility(View.INVISIBLE);
                    calCurrentCustomerButton.setVisibility(View.INVISIBLE);
                    locationDriverTextView.setText("Приємного користування");
                    if (driverPickUpMarker != null) {
                        driverPickUpMarker.remove();
                    }
                    if (assignedCustomerPositionListener != null) {

assignedCustomerPositionRef.removeEventListener(assignedCustomerPositionListener);
                    }
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }

    private void getAssignedCustomerPosition() {
        assignedCustomerPositionRef                                                 =
FirebaseDatabase.getInstance().getReference().child("Customers Requests")
                .child(customerID).child("l");
```

```
        assignedCustomerPositionListener                                    =
assignedCustomerPositionRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    List<Object>      customerPositionMap      =      (List<Object>)
snapshot.getValue();
                    double                        locationLat                       =
Double.parseDouble(customerPositionMap.get(0).toString());
                    double                        locationLng                       =
Double.parseDouble(customerPositionMap.get(1).toString());

                    LatLng customerLatLng = new LatLng(locationLat, locationLng);
                    driverPickUpMarker              =              mMap.addMarker(new
MarkerOptions().position(customerLatLng).title("Забрати клієнта тут")

.icon(BitmapDescriptorFactory.fromResource(R.drawable.location_image)));
                    mMap.moveCamera(CameraUpdateFactory.newLatLng(customerLatLng));
                    mMap.animateCamera(CameraUpdateFactory.zoomTo(11));

                    customerInfoRelativeLayout.setVisibility(View.VISIBLE);
                    calCurrentCustomerButton.setVisibility(View.VISIBLE);
                    DatabaseReference                   reference                   =
FirebaseDatabase.getInstance().getReference()
                            .child("Users").child("Customers").child(customerID);

                    reference.addValueEventListener(new ValueEventListener() {
                        @SuppressLint({"DefaultLocale", "SetTextI18n"})
                        @Override
                        public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                            if               (dataSnapshot.exists()               &&
dataSnapshot.getChildrenCount() > 0) {
                                String                      name                    =
dataSnapshot.child("Name").getValue().toString();
                                String                     phone                    =
dataSnapshot.child("Phone").getValue().toString();
                                String                    carName                   =
dataSnapshot.child("CarName").getValue().toString();
                                String                   carNumber                  =
dataSnapshot.child("CarNumber").getValue().toString();

                                customerMapsName.setText(name);
                                customerMapsPhoneNumber.setText(phone);
                                customerMapsCarInfo.setText(carName);
                                customerMapsCarNumber.setText(carNumber);

                                currentCustomerPhoneNumber = phone;
                            }
                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError databaseError)
{}
                    });

                    Location location1 = new Location("");
                    location1.setLatitude(driverCurrentPosition.latitude);
                    location1.setLongitude(driverCurrentPosition.longitude);

                    Location location2 = new Location("");
```

```
                    location2.setLatitude(customerLatLng.latitude);
                    location2.setLongitude(customerLatLng.longitude);

                    float Distance = location1.distanceTo(location2);
                    float roadDistance = Distance / 1000;
                    locationDriverTextView.setText("Відстань до авто " +
String.format("%.2f", roadDistance) + " км");


                    if (Distance == 0) {

                        Intent resultIntent = new Intent(DriverMapsActivity.this,
DriverOrderResultActivity.class);
                        startActivity(resultIntent);
                    }
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }


    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        buildGoogleApiClient();
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            return;
        }
        mMap.setMyLocationEnabled(true);
    }


    @Override
    public void onConnected(@Nullable Bundle bundle) {
        locationRequest = new LocationRequest();
        locationRequest.setInterval(100000);
        locationRequest.setFastestInterval(100000);
        locationRequest.setPriority(locationRequest.PRIORITY_HIGH_ACCURACY);

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION)
!= PackageManager.PERMISSION_GRANTED) {
            return;
        }
        LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient,
locationRequest, this);
    }

    @Override
    public void onConnectionSuspended(int i) {

    }
```

```
    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @SuppressLint("DefaultLocale")
    @Override
    public void onLocationChanged(Location location) {
        if (getApplicationContext() != null) {
            lastLocation = location;
            LatLng       latLng       =       new       LatLng(location.getLatitude(),
location.getLongitude());
            mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
            mMap.animateCamera(CameraUpdateFactory.zoomTo(12));

            driverCurrentPosition      =      new      LatLng(location.getLatitude(),
location.getLongitude());

            menuCurrentDriverCoordinates = location;
            menuDriverLatitude = String.format("%.3f", location.getLatitude());
            menuDriverLongitude = String.format("%.3f", location.getLongitude());

            String userID = FirebaseAuth.getInstance().getCurrentUser().getUid();

            DatabaseReference                  availableDriversRef                  =
FirebaseDatabase.getInstance().getReference().child("Available Drivers");
            GeoFire geoFireAvailability = new GeoFire(availableDriversRef);

            DatabaseReference                    driverWorkingRef                    =
FirebaseDatabase.getInstance().getReference().child("Driver Working");
            GeoFire geoFireWorking = new GeoFire(driverWorkingRef);

            switch (customerID) {
                case "":
                    geoFireWorking.removeLocation(userID);
                    geoFireAvailability.setLocation(userID,                        new
GeoLocation(location.getLatitude(), location.getLongitude()));
                    break;
                default:
                    geoFireAvailability.removeLocation(userID);
                    geoFireWorking.setLocation(userID,                             new
GeoLocation(location.getLatitude(), location.getLongitude()));
                    break;
            }
        }
    }

    protected synchronized void buildGoogleApiClient() {
        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        googleApiClient.connect();
    }

    @Override
    protected void onStop() {
        super.onStop();
        if(!currentLogOutDriverStatus) {
            DisconnectDriver();
```

```
        }
    }

    private void DisconnectDriver() {
        String userID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        DatabaseReference                    availableDriversRef                    =
FirebaseDatabase.getInstance().getReference().child("Available Drivers");
        GeoFire geoFire = new GeoFire(availableDriversRef);
        geoFire.removeLocation(userID);
    }

    private void LogoutDriver() {
        Intent      welcomeIntent      =      new      Intent(DriverMapsActivity.this,
RoleSelectionActivity.class);
        startActivity(welcomeIntent);
        finish();
    }

    private void CheckCurrentUserOrdering() {
        String driverID = FirebaseAuth.getInstance().getCurrentUser().getUid();
        DatabaseReference                          checkRef                          =
FirebaseDatabase.getInstance().getReference().child("Driver Working")
                .child(driverID);

        checkRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    currentUserOrderingStatus = true;
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {}
        });
    }
}
```