Application Development for Mobile Petrol and Servicing

A PROJECT REPORT

Submitted by

H.RAGHUL[211420104334]
S.BARATH[211420104320]
S.YOGAPRAKASH[211420104312]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"APPLICATION DEVELOPMENT MOBILE PETROL SERVICING"**is the bonafide work of H. RAGHUL [211420104334], S.BARATH [211420104320] & S.YOGAPRAKASH [211420104312] who carried out the project work under my supervision.

Signature of the HOD with date

DR L.JABASHEELA M.E.,Ph.D.,

Professor and Head,

Department of Computer Science and Engineering,

Panimalar Engineering College,

Chennai – 123

Signature of the Supervisor with date

Dr. Sathya preiya M.E.,Ph.D.,

Associate Professor

Department of Computer Science and Engineering,

Panimalar Engineering College,

Chennai - 123

Submitted for the Project Viva – Voce examination held on 25/03/2024.

INTERNAL EXAMINER **EXTERNAL EXAMINE**

DECLARATION BY THE STUDENT

We H. RAGHUL [211420104334], S.BARATH [211420104320] & S.YOGAPRAKASH [211420104312] hereby declare that this project report titled " APPLICATION DEVELOPMENT MOBILE PETROL SERVICING ", under the guidance of Dr.Sathya preiya M.E.,Ph.D. Professor is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his benevolent words and fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express my heartfelt thanks to **Dr. L. JABASHEELA,M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Dr.N.PUGHAZENDI** and Mrs. Sathya preiya and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

H.RAGHUL

S.BARATH

S.YOGAPRAKASH

ABSTRACT

The use of mobile phones (smart phones) have gone beyond just making phone calls and sending text messages to specialized applications such as object recognition, patient health monitoring, security and navigation, amongst others. In this paper, BRAZ named after the authors is an android App for monitoring fuel situation in gas stations. This research was particularly motivated by the uncertainty faced by consumers of Premium Motor Spirit (PMS) in locating the closest gas station. PMS is an essential commodity in the daily life of almost every individual. Consumers have been complaining about the high petrol prices and everyone wanting to access the cheapest and closest station available in order to save expenses. Therefore, with a single click on BRAZ, the list of gas stations with the product is provided to the user in real-time. Also included are the price/litre, car density, and a GPS location of the station which can lead user to the location via Google map. This is particularly of advantage to visitors who are not familiar with the road network of the town. A validation test carried out on a random sample of 60 users gave 80% of users' acceptability. If the petrol and servicing operations are managed manually, it could involve physical records, paperwork, and human intervention at every step. This can be time-consuming, error-prone, and lacks efficiency. Legacy Software: If there's any existing software, it might be outdated, lack modern features, or not be user-friendly.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview

In the fast-paced world we live in today, convenience is key. We understand that your time is valuable, and so is the health and maintenance of your vehicle. That's why we're thrilled to introduce [App Name], a cutting-edge solution designed to bring petrol delivery and servicing right to your fingertips.

Convenience at your doorstep: We bring the fuel station and service center to you. On-Demand Fueling: Refuel your vehicle wherever you are, on your schedule. Comprehensive Servicing: Skilled technicians handle routine maintenance and repairs with ease.

Time and Cost Efficiency: Skip the queues and save time with our mobile services.

Quality Fuel: Ensure optimal performance and fuel efficiency for your vehicle.

In today's dynamic world, where convenience and efficiency are highly valued, the automotive industry has witnessed a significant shift towards innovative solutions that cater to the evolving needs of vehicle owners. The emergence of mobile petrol and servicing applications exemplifies this trend, offering a revolutionary approach to fueling up and maintaining vehicles. Mobile petrol and servicing applications are revolutionizing the way vehicle owners interact with their automobiles. These apps provide a seamless platform for users to access fuel delivery services, schedule maintenance tasks, and manage their vehicles' upkeep directly from their smartphones. By harnessing the power of technology, these applications empower users to take.

1.1Problem Definition

Enable users to request fuel delivery and schedule vehicle servicing directly from their smartphones, eliminating the need for visits to gas stations or service centers. Time Savings: Streamline the fueling and maintenance process, reducing wait times and allowing vehicle owners to allocate their time more efficiently. Safety Assurance: Offer safe and reliable fueling options at users' preferred locations, while also promoting regular vehicle maintenance to ensure roadworthiness and safety. Transparency and Accountability: Provide users with real-time updates on fuel delivery and service appointments, as well as access to detailed maintenance records and service history. Environmental Sustainability: Encourage eco-friendly practices such as efficient fuel delivery routes and promoting vehicle maintenance to reduce emissions and minimize environmental impact. By addressing these challenges and providing these benefits, the development of a mobile petrol and servicing application aims to revolutionize the way vehicle owners manage their fueling and maintenance needs, offering a more convenient, efficient, and environmentally sustainable solution. If the petrol and servicing operations are managed manually, it could involve physical records, paperwork, and human intervention at every step. This can be time-consuming, error-prone, and lacks efficiency. Legacy Software: If there's any existing software, it might be outdated, lack modern features, or not be user-friendly.

CHAPTER 2

LITERATURE REVIEW

Title and Journal: Lee, C. and Bagheri, B: "Digital Transformation in the Automotive Service Industry: Implications of Mobile Petrol and Servicing Apps" (2021).

Description:

This journal explores the broader impact of mobile petrol and servicing applications on the automotive service industry. The paper explores the impact of digital transformation on the automotive service industry, focusing specifically on the implications of mobile petrol and servicing applications. It investigates how these apps are reshaping the way vehicle owners interact with their automobiles and how automotive service providers deliver their services

Advantages

- This eliminates the need for time-consuming visits to gas stations or service centers, saving users valuable time and effort.
- Enhanced operational efficiency and reduced service costs**.**

Disadvantages

- Resistance to change within traditional service models.

1)Title and Journal: Smith, J. et al: "Enhancing User Experience in Mobile Petrol and Servicing Applications (2022).

Description

This journal explores the significance of user experience in the context of mobile applications for petrol and servicing. Developing a mobile application for petrol (gasoline) and servicing can be a valuable tool for both consumers and service providers. Here's a description outlining the key features and functionalities such an application could offer.

Advantages

- Increased user satisfaction and engagement.
- **Real-Time Information**: The application provides real-time updates on fuel prices, availability, and service offerings, ensuring users make informed decisions.

Disadvantages

Development complexity associated with maintaining a high level of user- friendliness.

CHAPTER 3

THEORETICAL BACKGROUND

3.1Proposed System

Before proposing a methodology for developing the mobile petrol and servicing application, it's essential to understand the existing system. The existing system likely involves manual processes or outdated software that might not fully meet the needs of users or the business. Here are some common aspects to consider in understanding the existing system: Manual Processes: If the petrol and servicing operations are managed manually, it could involve physical records, paperwork, and human intervention at every step. This can be time-consuming, error-prone, and lacks efficiency. Legacy Software: If there's any existing software, it might be outdated, lack modern features, or not be user-friendly. It could also have limitations in scalability and integration with other systems. Customer Experience: Analyze how customers currently interact with the system. Are there any pain points, such as long wait times, difficulty in accessing services, or lack of transparency in the process. Operational Efficiency: Assess the efficiency of operations from the business perspective. Are there any bottlenecks or inefficiencies in the current system that could be improved with technology.If the petrol and servicing operations are managed manually, it could involve physical records, paperwork, and human intervention at every step. This can be time-consuming, error-prone, and lacks efficiency. Legacy Software: If there's any existing software, it might be outdated, lack modern features, or not be user-friendly.

3.2 IMPLEMENTATION ENVIRONMENT

Implementing a mobile petrol and servicing application requires careful consideration of various technical aspects to ensure smooth functionality and user experience. Below are some key components and considerations for the implementation environment:

Platform Selection: Decide whether to develop a native app (iOS/Android) or a cross-platform app (using frameworks like React Native, Flutter, or Xamarin). This decision depends on factors like target audience, budget, and development resources.

Backend Infrastructure: Set up a robust backend infrastructure to handle user data, payments, and other functionalities. Consider using cloud-based solutions like AWS, Google Cloud Platform, or Microsoft Azure for scalability and reliability.

Database: Choose an appropriate database system to store user information, vehicle details, service history, etc. Options include SQL databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB, depending on the data structure and scalability requirements.

APIs: Develop APIs to facilitate communication between the mobile app and the backend server. Use RESTful or GraphQL APIs for efficient data exchange and integration with third-party services like payment gateways, mapping APIs, and vehicle diagnostics systems.Testing and deployment are critical phases in the development lifecycle of a mobile fuel app, ensuring its functionality, reliability, and usability before it reaches users' hands. Initially, rigorous unit testing is conducted to verify the functionality of individual components, followed by integration testing

3.3SYSTEM ARCHITECTURE

The system architecture for a mobile petrol and servicing application typically involves several layers and components to ensure scalability, reliability, and maintainability. Here's an overview of the key components and their interactions:

Presentation Layer

Mobile App Interface: This layer represents the user interface (UI) of the mobile application through which users interact with the system. It includes screens, forms, buttons, and other UI elements.

Web Interface (Optional): If the application has a web-based counterpart, there may be a web interface for users to access the same functionalities through a web browser.

Application Layer

Client-Side Logic: This includes the logic and functionality implemented within the mobile app or web interface, such as user authentication, input validation, and user interaction handling.

Server-Side Logic**:** The backend logic responsible for processing requests from the client-side, interacting with the database, and executing business rules. This layer may include services for handling user management, petrol ordering, service scheduling, and more.

Security Layer: Authentication and Authorization: Secure user authentication mechanisms (e.g., OAuth, JWT) ensure that only authorized users can access the application's features and data.

3.3.1 ARCHITECTURE

After finding the nearest petrol station, the user can select a specific station. The user can then view prices and select a service type. The user schedules an appointment, confirms it, and views the service history. Additionally, the user can view detailed service information before making a payment.

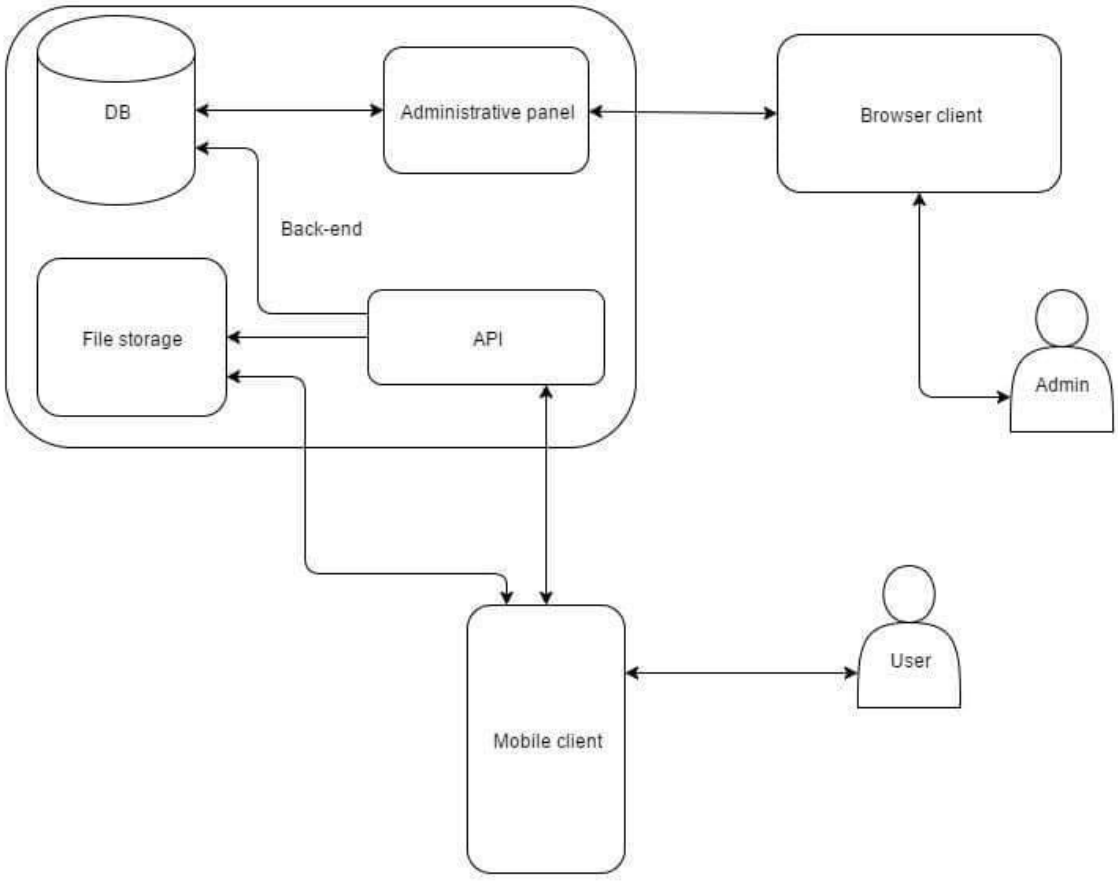


fig 3.3.1 (system architecture)

The payment process involves sending payment information, processing the payment, and confirming the payment status. After receiving service, the user can provide feedback, which is sent to the system. The system notifies the admin about the new feedback

3.3.2 LOGIN PAGE

Users can provide feedback, which the Admin can view and act upon. Admin can send notifications to users based on various triggers such as promotional offers, service reminders, etc. User payments for petrol or services are processed by the system, which is managed by the Admin. Admin manages user accounts and service records, which are accessed and utilized by users as needed.

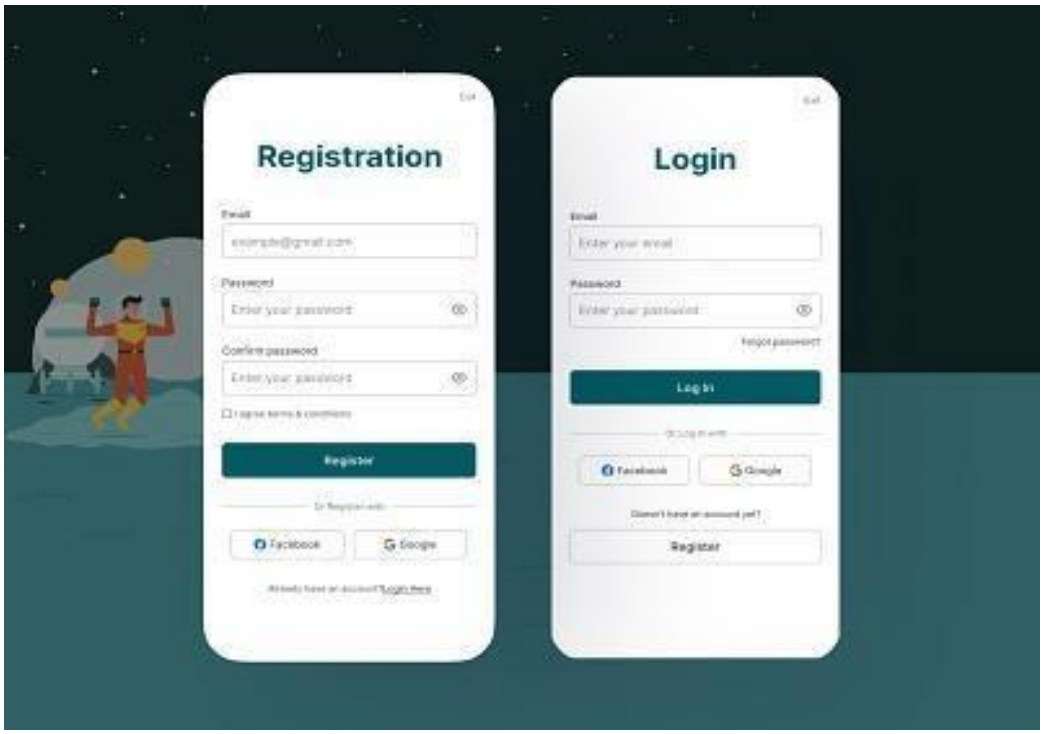


fig 3.3.2 login page

User Interface: Sends user requests for finding nearest petrol station, viewing prices, scheduling appointments, viewing service history, making payments, and providing feedback. Receives notifications from the system and displays them to users. Retrol and Servicing System:

3.3.3 Module Design

Use Case Diagram

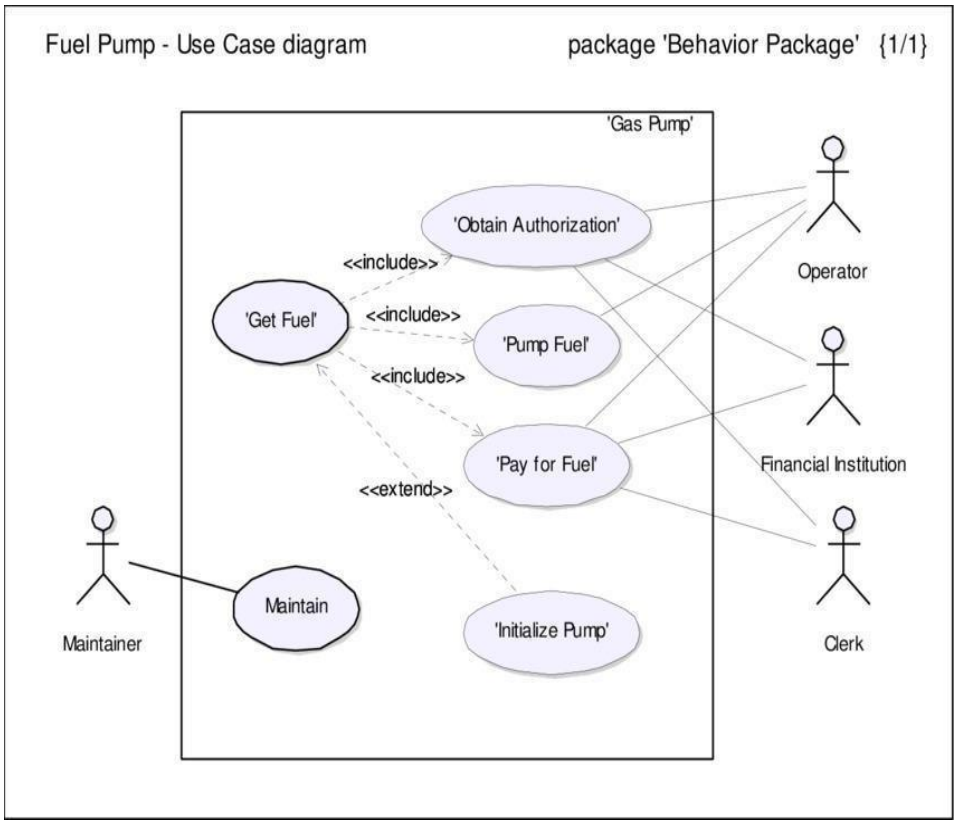


Fig 3.3.3 (Use Case Diagram)

Admin manages backend operations such as managing prices, appointments, and feedback. Users can provide feedback, which the Admin can view and act upon. Admin can send notifications to users based on various triggers such as promotional offers, service

3.3.4Data Flow Diagram

User Interface: sends user requests for finding nearest petrol station, viewing prices, scheduling appointments, viewing service history, making payments, and providing feedback. Receives notifications from the system and displays them to users. Petrol and Servicing System: Petrol Station Database: Stores information about petrol stations including locations, prices, and available services

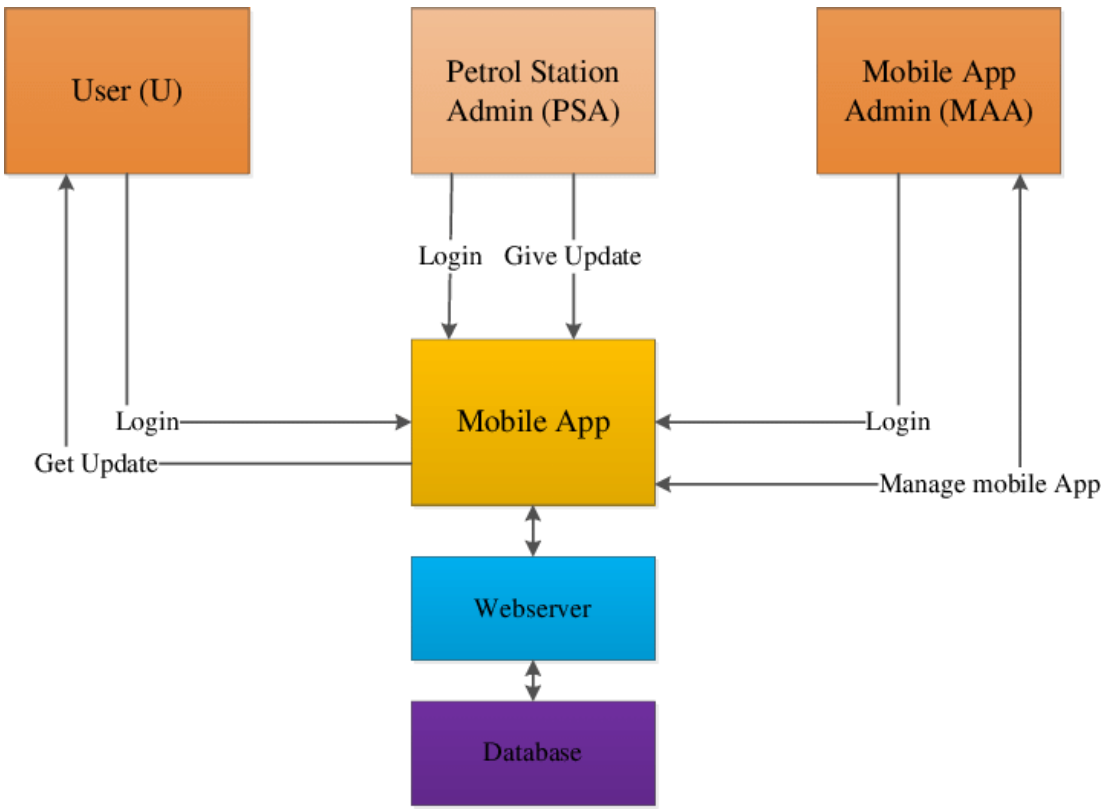


Fig 3.3.4 Data Flow Diagram

User: The primary external entity interacting with the mobile application.

Petrol Station: Represents external systems providing petrol services.

Service Center: Represents external systems offering vehicle servicing.

3.3.5Class diagram

user: represents users of the mobile application. it contains attributes such as userid, username, and password. admin: represents administrators who manage the application. it contains attributes such as adminid, username, and password. petrolstation: represents petrol stations available in the system. it contains attributes such as stationid, location, petrolprice, and services offered.

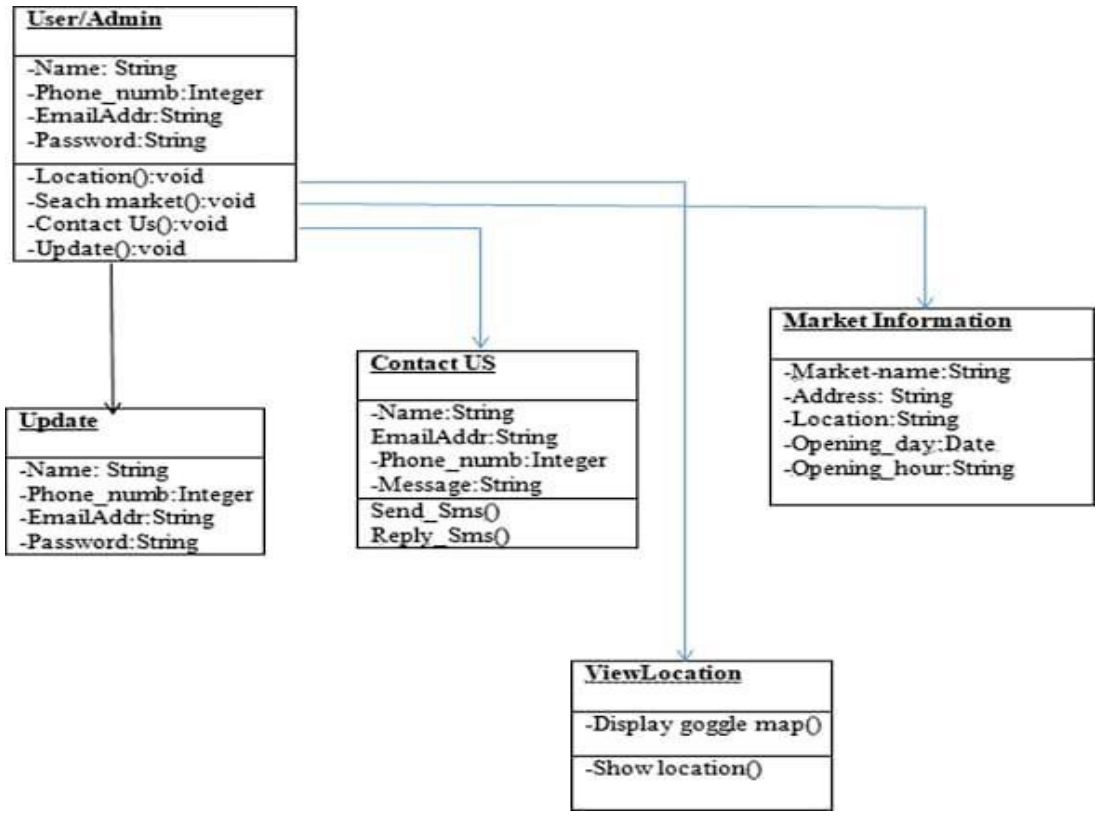


Fig 3.3.5 class diagram

Service record: represents service records of users. it contains attributes such as record id, user id, station id, and service details. payment: represents payments made by users for petrol or services. it contains attributes such as payment d, user id, amount, and date.

3.3.5 Sequence Diagram

After finding the nearest petrol station, the user can select a specific station. The user can then view prices and select a service type. The user schedules an appointment, confirms it, and views the service history. Additionally, the user can view detailed service information before making a payment.

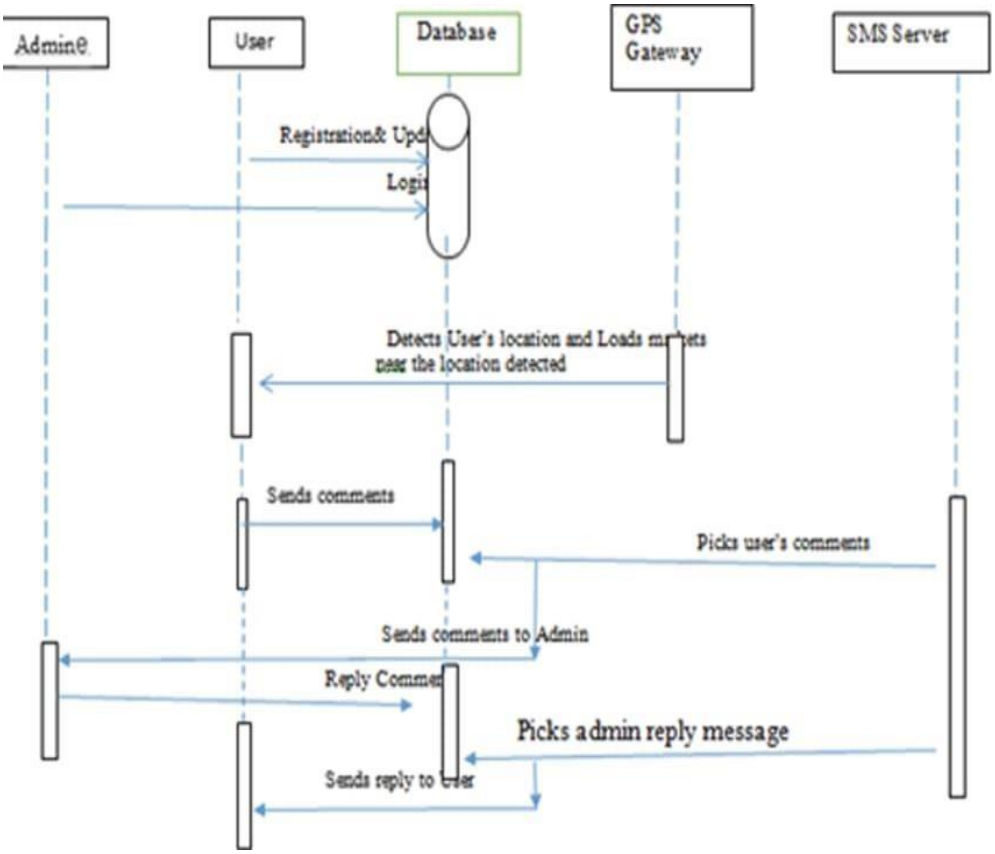


Fig 3.3.6 Sequence diagram

The payment process involves sending payment information, processing the payment, and confirming the payment status. After receiving service, the user can provide feedback, which is sent to the system.

3.3.6Collaboration diagram

A collaboration diagram visually represents how objects in a system interact with each other to achieve a particular goal. In the context of a mobile petrol and servicing application, here's a simplified collaboration diagram: User: Represents the person using the mobile app. Mobile App Interface: Represents the interface of the mobile application through which the user interacts.

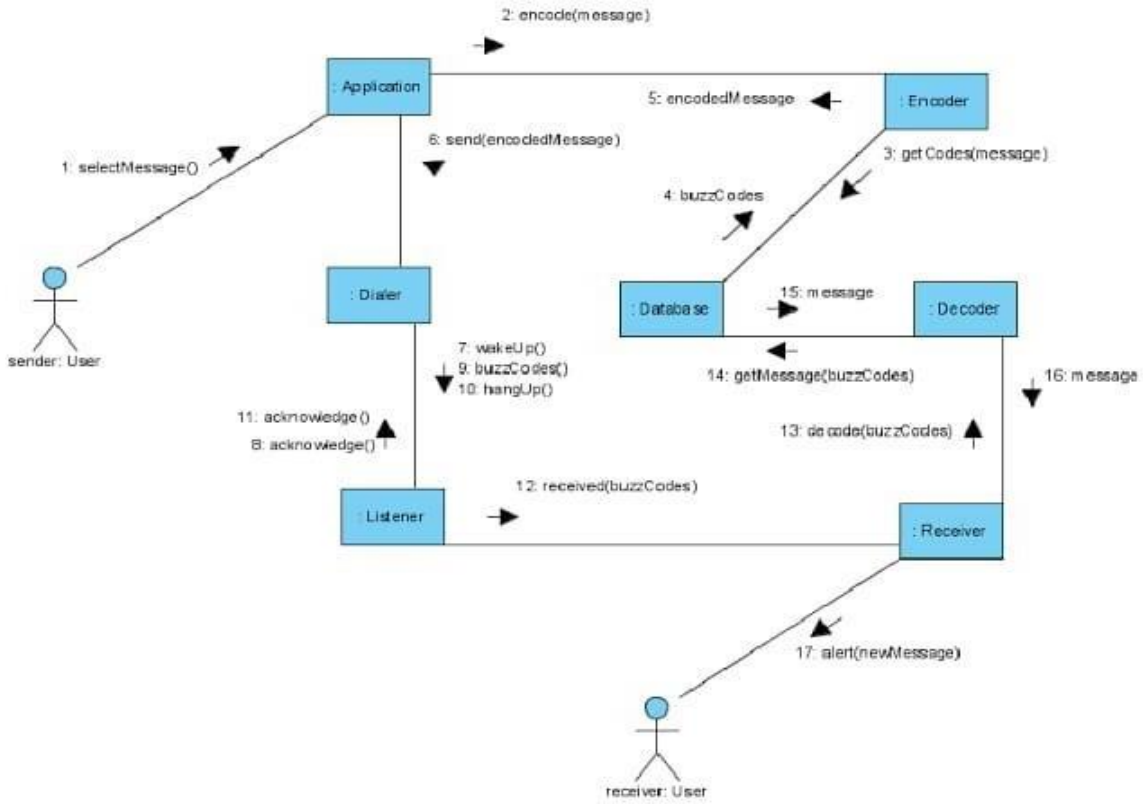


Fig 3.3.7 Collaboration diagram

Petrol Station: Represents the physical petrol station that the user interacts with through the app. Payment Gateway: Represents the service or system responsible for processing payments made through the app. Service Provider: Represents the entity responsible for servicing vehicles, such as mechanics or technicians

3.3.8Activity diagram

The user interacts with the mobile app interface to request petrol or servicing. The app processes the user's request and sends it to the petrol station, payment gateway, and service provider. The payment gateway verifies the payment information provided by the user.

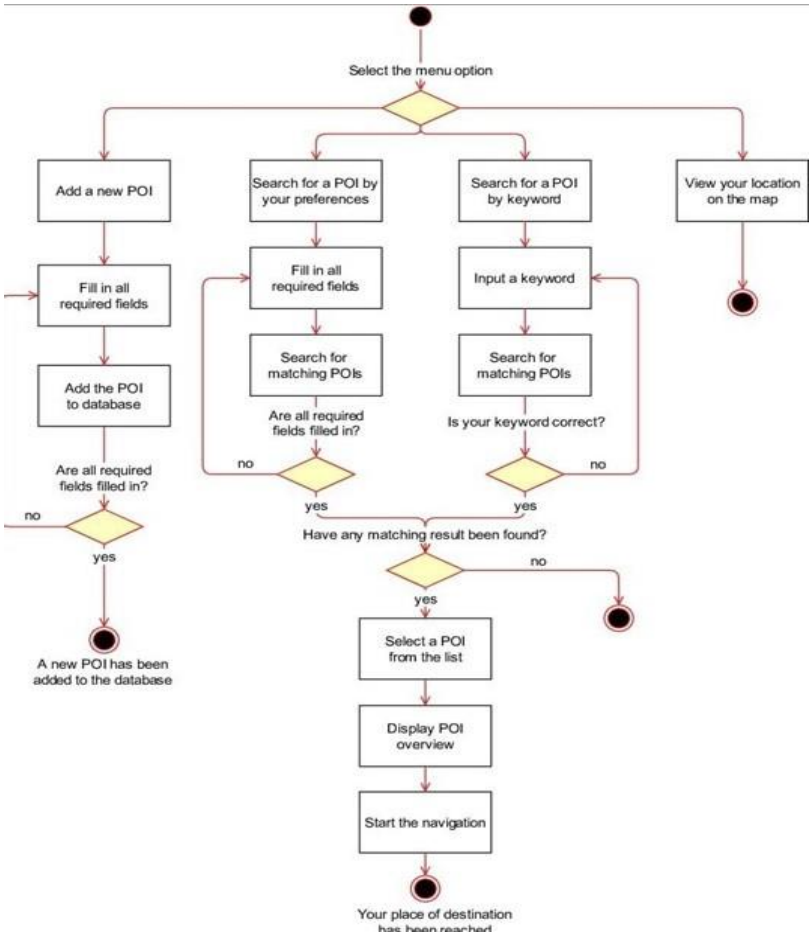


Fig 3.3.9. Activity diagram

Interacts with the mobile app interface to request petrol or servicing. The app processes the user's request and sends it to the petrol station, payment gateway, and service provider. The payment gateway verifies the payment information provided by the user.

3.3.9State chart diagram

In a state chart diagram for a mobile petrol and servicing application, we can depict the various states that the system or specific entities within the system can exist in, as well as the transitions between these states triggered by certain events. Here's a simplified state chart diagram for such an application. The initial state is when the user is logged into the application.

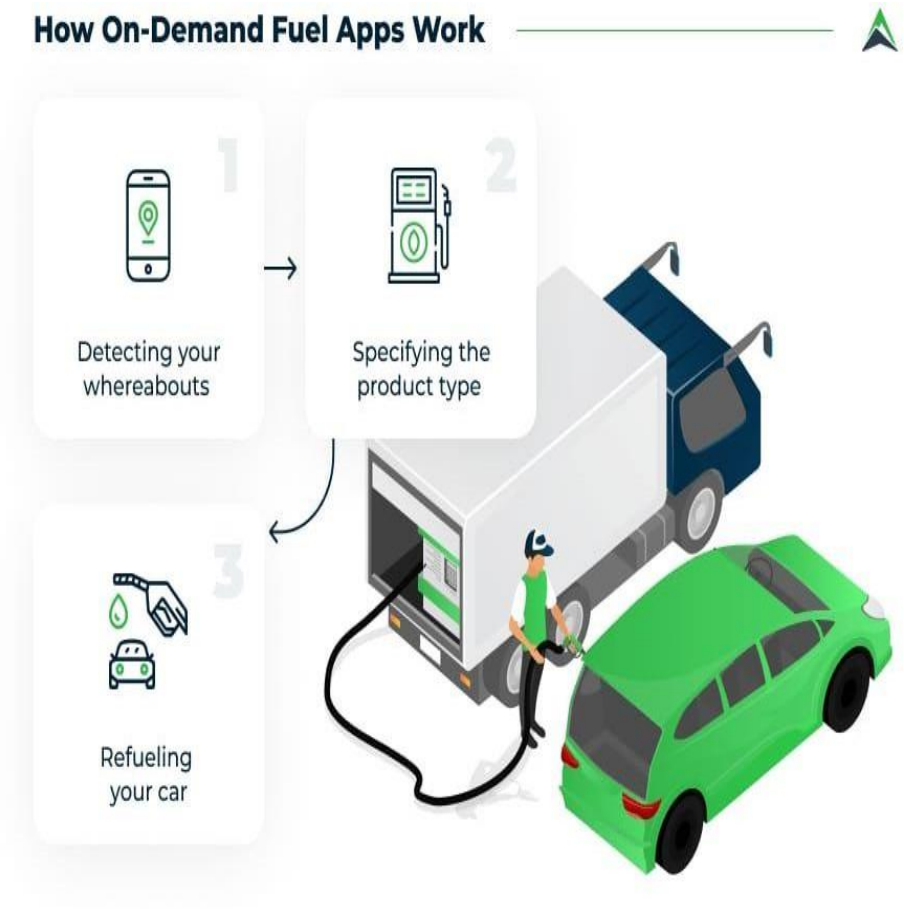


Fig 3.3.10 State chart diagram

3.3.10Package diagram

Mobile Application: Represents the main package containing all components related to the mobile application. User Interface: Contains modules responsible for handling user interaction and displaying the interface. Business Logic: Contains modules responsible for implementing the application's business logic, such as processing user requests and handling service logic. Data Access: Contains modules responsible for accessing and manipulating data, such as user preferences, service requests, and service provider information.

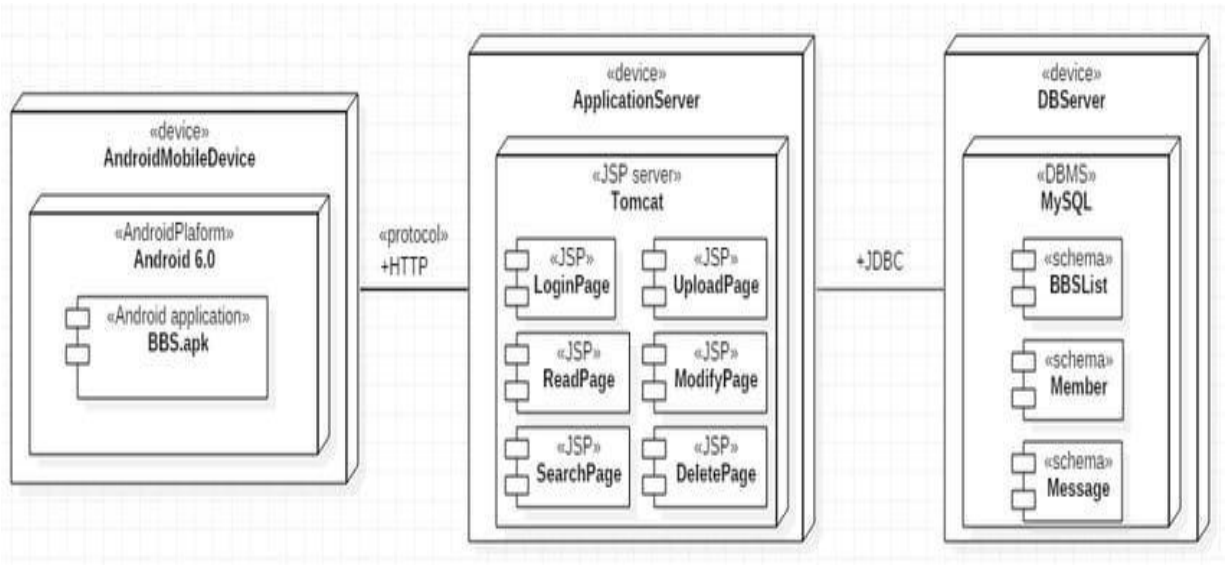


Fig 3.3.10 Package diagram

Communication: Contains modules responsible for handling communication with external systems, such as APIs for petrol stations, payment gateways, and service providers. External Systems: Represents external systems that the mobile application interacts with, such as petrol stations, payment gateways, and service providers.

3.3.11 Deployment diagram

Mobile Device: Represents the hardware node where the mobile application is installed and executed. Mobile App: Represents the software component of the mobile application running on the mobile device. Petrol Station Server: Represents the hardware node where the server-side components of the petrol station system are deployed

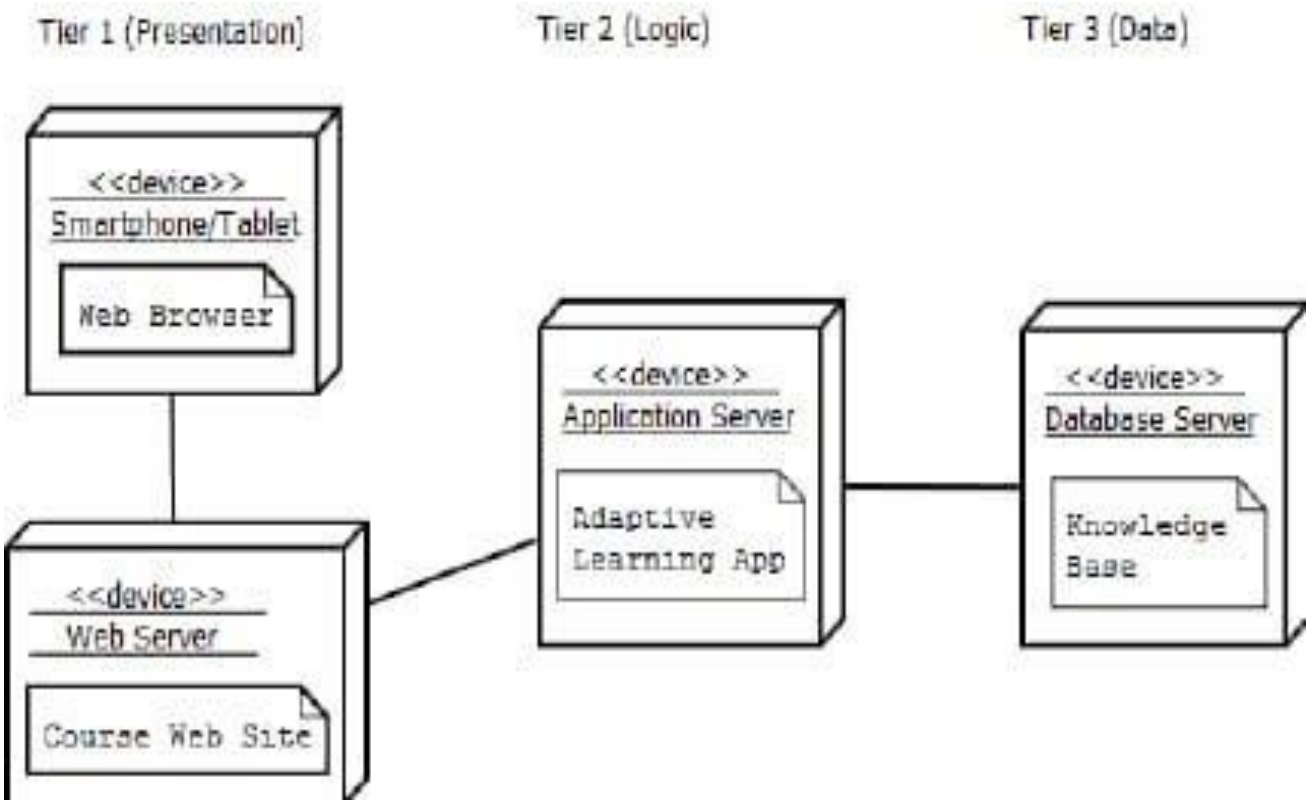


Fig 3.3.12 Deployment diagram

This could be a separate hardware node or a cloud-based service accessed via the internet. Internet Connection: Represents the network connection used for communication between the mobile device, petrol station server, and payment gateway.

CHAPTER 4

SYSTEM IMPLEMENTATION

Android studio

Set up Android Studio: Download and install Android Studio from the official website.

Create a New Project: Open Android Studio and create a new project. Choose "Empty Activity" as the template.

Design User Interface (UI):

Use XML layout files to design the UI for various screens such as home, petrol ordering, service scheduling, etc.

Implement UI components like buttons, text fields, and lists using Android's XML-based layout system.

Implement Functionality:

Write Java or Kotlin code to implement the desired functionality of your application. Implement features such as user authentication, petrol ordering, service scheduling, and payment processing. Testing and deployment are critical phases in the development lifecycle of a mobile fuel app, ensuring its functionality, reliability, and usability before it reaches users' hands. Initially, rigorous unit testing is conducted to verify the functionality of individual components, followed by integration testing

Firebase Console

Developing a mobile application for petrol (gasoline) and servicing using Firebase as a backend service is a feasible approach. Firebase offers various tools and services that can streamline the development process. Here's a general outline of how you can approach this:

Planning and Designing:

Define the features and functionalities of your mobile app, such as locating nearby petrol stations, scheduling servicing, user authentication, payment integration, etc. Create wireframes and design mockups of your app's user interface.

Setting Up Firebase:

Go to the Firebase Console (https://console.firebase.google.com/) and create a new project. Set up Firebase Authentication to manage user sign-ups and sign-ins securely. Utilize Fire store or Realtime Database to store data related to petrol stations, servicing schedules, user profiles, etc. Integrate Firebase Cloud Functions for server-side logic if needed.

Developing the Mobile App:

Choose a suitable development framework like Flutter (using Dart) or React Native (using JavaScript) for building your mobile app. Set up your development environment and configure Firebase SDKs in your project. implement authentication flows for users to sign up, sign in, and manage their profiles.

Testing

- Test your app thoroughly to ensure all features work as expected.
- Perform both unit testing and integration testing.
- Test on various devices and screen sizes to ensure compatibility.

Deployment

- Once testing is complete, deploy your app to the respective app stores (Google Play Store for Android and Apple App Store for iOS).
- Follow the respective guidelines and policies of each app store for submission.

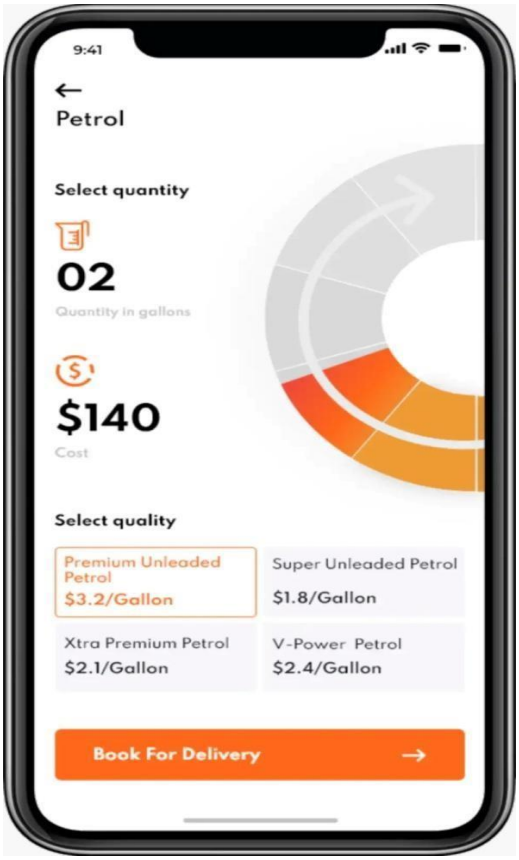


Fig book for delivery

Google Console

It seems you're referring to the Google Cloud Console, which is where you manage resources and services provided by Google Cloud Platform (GCP), including Firebase. Here's how you can utilize Google Cloud Console for your mobile petrol and servicing application development:

Creating a Project

Log in to the Google Cloud Console (https://console.cloud.google.com/).

Create a new project specifically for your mobile petrol and servicing application.

Setting Up Firebase

In the Google Cloud Console, navigate to the Firebase section.

Click on "Get Started" to link your Google Cloud project with Firebase.

Follow the instructions to set up Firebase services such as Authentication, Fire store/Realtime Database, Cloud Functions, etc.

Configuring Firebase Services

Configure Firebase Authentication to manage user sign-ups, sign-ins, and user sessions securely. Utilize Fire store or Realtime Database to store and retrieve data related to petrol stations, servicing schedules, user profiles, etc.

Implement Firebase Cloud Functions for server-side logic if needed, such as sending notifications or triggering actions based on certain events.

Testing and Deployment

- Test your app thoroughly on various devices and screen sizes.
- Use Firebase Test Lab for automated testing on real devices.
- Once testing is complete, deploy your app to Google Play Store for Android or Apple App Store for iOS.

Testing and deployment are critical phases in the development lifecycle of a mobile fuel app, ensuring its functionality, reliability, and usability before it reaches users' hands. Initially, rigorous unit testing is conducted to verify the functionality of individual components, followed by integration testing app's responsiveness and resource usage, while security testing checks for vulnerabilities to safeguard user data and privacy. Once thorough testing is complete, the app undergoes user acceptance testing (UAT) and beta testing to gather feedback from real users, enabling further refinement. Upon successful completion of testing, the app is prepared for deployment by submitting it to app stores for review and approval. Throughout the deployment process, meticulous release management is employed to schedule updates and communicate with users effectively. Continuous monitoring post-deployment ensures ongoing performance optimization and user satisfaction, culminating in the successful launch of a mobile fuel app that meets the highest standards of quality and reliability.

Google Services

It seems you're looking for information on specific Google services that might be useful for your mobile petrol and servicing application development. Here are some Google services you might consider integrating:

Google Maps Platform

Utilize Google Maps API to provide location-based services, such as finding nearby petrol stations and offering navigation to them**.**

Use Places API to retrieve information about petrol stations, such as their names, addresses, ratings, and reviews

Firebase

Firebase offers various services that can be useful for your application, including Authentication for user management, Fire store or Realtime Database for storing data, Cloud Functions for server-side logic, and Cloud Messaging for push notifications.

Google Places API:

This API provides detailed information about places, including petrol stations. You can use it to fetch information like opening hours, contact details, and user reviews for petrol stations near a given location.

The mobile petrol and servicing initiative successfully introduced on-demand fuel delivery and vehicle servicing to customers' preferred locations. Customers were able to order fuel and schedule servicing appointments conveniently through the mobile app or website.

Google Directions API

If you want to provide navigation to petrol stations or service centers, you can use this API to generate directions and routes based on various transportation modes.

Google Pay

Integrate Google Pay for seamless and secure payment transactions within your application.

Google Analytics

Use Google Analytics for Mobile to track user interactions within your app, understand user behavior, and make data-driven decisions for further improvements.

Google Maps Platform

Maps SDK for Android/iOS: Allows you to embed interactive Google Maps directly into your mobile app, providing users with familiar map interactions and functionalities.

Places API

Enables you to retrieve detailed information about places, including petrol stations, based on various search criteria like location, type, and keyword.

Directions API

Provides routing, navigation, and distance matrix functionalities, allowing users to find the best routes to petrol stations or service centers from their current location.

Geocoding API

Converts addresses into geographic coordinates (latitude and longitude) and vice versa, facilitating location-based services in your application**.**

changing market dynamics and user expectations, ultimately leading to sustained growth and success. By prioritizing customer feedback and satisfaction, developers can forge meaningful connections with users and position the mobile fuel application for long-term viability and success in a competitive landscape.

Visual studio code

Visual Studio Code (VS Code) is a popular code editor developed by Microsoft. It's lightweight, highly customizable, and supports a wide range of programming languages and frameworks. Here's how you can utilize VS Code for your mobile petrol and servicing application development

Integrated Development Environment (IDE)

Use VS Code as your primary IDE for writing, debugging, and testing code for both frontend (mobile app) and backend (server-side logic). Install relevant extensions for Flutter (if using Dart for mobile app development), JavaScript/Node.js (if using React Native or other JavaScript frameworks), and Firebase/GCP (for backend development and deployment).

Code Editing and Productivity

Leverage features like IntelliSense, code completion, and syntax highlighting to write code efficiently and reduce errors. Customize your development environment with themes, keyboard shortcuts, and extensions tailored to your preferences and workflow**.**

Version Control

Integrate Git within VS Code to manage version control for your project. Utilize built-in Git features or install extensions like Git Lens to streamline collaboration, code reviews, and codebase management. changing market dynamics and user expectations, ultimately leading to sustained growth and success. By prioritizing customer feedback and satisfaction, developers can forge meaningful connections with users and position the mobile fuel application for long-term viability and success in a competitive landscape.

CHAPTER 5

Result and Discussion

Functionality and Service Accessibility

The mobile petrol and servicing initiative successfully introduced on-demand fuel delivery and vehicle servicing to customers' preferred locations. Customers were able to order fuel and schedule servicing appointments conveniently through the mobile app or website.

Operational Efficiency

The system streamlined fuel delivery routes and service scheduling, resulting in improved operational efficiency. Real-time tracking of service vehicles allowed for dynamic route optimization and timely service delivery.

Customer Feedback and Satisfaction

Customer feedback surveys indicated a high level of satisfaction with the convenience and reliability of the mobile petrol and servicing solution. provides quantitative indicators of performance and areas for enhancement. Addressing customer concerns promptly and transparently demonstrates a commitment to excellence and fosters trust and loyalty among users. Moreover, leveraging customer feedback as a driving force for innovation allows the mobile fuel application to adapt and innovate in response to changing market dynamics and user expectations, ultimately leading to sustained growth and success. By prioritizing customer feedback and satisfaction, developers can forge meaningful connections with users and position the mobile fuel application for long-term viability and success in a competitive landscape.

Chapter 6

CONCLUSION

While our mobile petrol servicing application represents a significant step forward, there are several avenues for future work and enhancement. In the future, we aim to expand the range of services offered through the application. This could include additional on-demand services such as tire inflation, windshield cleaning, and basic vehicle maintenance. Currently, our application supports cash payments upon delivery. However, integrating digital payment options such as credit/debit cards, mobile wallets, and UPI would enhance convenience for users and streamline the payment process. To further optimize efficiency, future iterations of the application could incorporate advanced routing algorithms. These algorithms would take into account real-time traffic conditions, user demand patterns, and fuel station availability to optimize delivery routes and minimize travel time. As sustainability becomes increasingly important, future versions of the application could explore ways to promote eco-friendly practices. This could include offering biofuel options, incentivizing users to carpool or opt for electric vehicles, and implementing environmentally friendly delivery methods. This could include offering biofuel options, incentivizing users to delivery methods. Furthermore, the mobile application can serve as a platform for fostering customer engagement and brand loyalty through personalized offers, loyalty programs, and interactive features like feedback mechanisms and social media integration. By staying connected with users beyond their petrol fill-ups or servicing appointments, businesses can build lasting relationships and differentiate themselves in a competitive market.

Chapter 7

Source Code:

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="@drawable/background_login"> <!-- Background drawable
-->


<ImageView

android:id="@+id/imageViewLogo"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_centerHorizontal="true"

android:layout_marginTop="50dp"/>


<EditText

android:id="@+id/editTextUsername"

android:layout_width="match_parent"

android:layout_height="wrap_content"
```

```
android:hint="Username"
android:inputType="text"
android:layout_below="@id/imageViewLogo"
android:layout_marginTop="20dp"
android:layout_marginStart="16dp"
android:layout_marginEnd="16dp" />

<EditText
android:id="@+id/editTextPassword"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Password"
android:inputType="textPassword"
android:layout_below="@id/editTextUsername"
android:layout_marginTop="10dp"
android:layout_marginStart="16dp"
android:layout_marginEnd="16dp"/>

<Button
android:id="@+id/buttonLogin"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Login"
```

```
android:layout_below="@id/editTextPassword"
android:layout_centerHorizontal="true"
android:layout_marginTop="20dp"
android:background="@drawable/button_background"/>
<TextView
android:id="@+id/textViewSignUp"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Don't have an account? Sign up"
android:textColor="@color/red"
android:layout_below="@id/buttonLogin"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
android:clickable="true"
android:onClick="onSignUpClicked" />

</RelativeLayout>
```

MainActivity.java:

```
package com.example.mobilefuel;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private EditText editTextDistance;
    private TextView textViewResult;
```

```
private Button buttonCalculate;
private Button buttonClear;
private Button buttonDeliver;
private Spinner spinnerCurrency;
private CheckBox checkBoxHomeDelivery;
private CheckBox checkBoxRemoteDelivery;
private RadioGroup radioGroupDelivery;
private double petrolPricePerKm = 2.0; // Assumed price per km in USD

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

    editTextDistance = findViewById(R.id.editTextDistance);
    textViewResult = findViewById(R.id.textViewResult);
    buttonCalculate = findViewById(R.id.buttonCalculate);
    buttonClear = findViewById(R.id.buttonClear);
    buttonDeliver = findViewById(R.id.buttonDeliver);
    spinnerCurrency = findViewById(R.id.spinnerCurrency);
    checkBoxHomeDelivery = findViewById(R.id.checkBoxHomeDelivery);
    checkBoxRemoteDelivery = findViewById(R.id.checkBoxRemoteDelivery);
    radioGroupDelivery = findViewById(R.id.radioGroupDelivery);
```

```
// Calculate button onClickListener
buttonCalculate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        calculatePetrol();
    }
});

// Clear button onClickListener
buttonClear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clearFields();
    }
});

// Delivery button onClickListener
buttonDeliver.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        confirmDelivery();
    }
```

```
        });

        // TextChangedListener for EditText fields
        editTextDistance.addTextChangedListener(inputTextWatcher);

        // Initially disable Calculate and Deliver buttons
        buttonCalculate.setEnabled(false);
        buttonDeliver.setEnabled(false);
    }

    // TextWatcher to enable/disable the Calculate button based on input
    private final TextWatcher inputTextWatcher = new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after)
{
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            // Enable Calculate button only if distance is not empty

buttonCalculate.setEnabled(!editTextDistance.getText().toString().isEmpty());
```

```
        // Enable Deliver button only if at least one delivery option is selected
        buttonDeliver.setEnabled(checkBoxHomeDelivery.isChecked()
                || checkBoxRemoteDelivery.isChecked()
                || radioGroupDelivery.getCheckedRadioButtonId() != -1);
    }

    @Override
    public void afterTextChanged(Editable s) {
    }
};

// Calculate the petrol needed and cost
private void calculatePetrol() {
    try {
        double distance =
Double.parseDouble(editTextDistance.getText().toString());

        double cost = distance * petrolPricePerKm;

        // Add delivery cost based on options selected
        if (checkBoxHomeDelivery.isChecked()) {
            cost += 5.0; // Home delivery charge
        }
```

```
        if (checkBoxRemoteDelivery.isChecked()) {
            cost += 10.0; // Remote location delivery charge
        }

        // Add delivery cost based on radio button selection
        int selectedRadioButtonId = radioGroupDelivery.getCheckedRadioButtonId();
        RadioButton selectedRadioButton = findViewById(selectedRadioButtonId);
        if (selectedRadioButton != null) {
            String deliveryOption = selectedRadioButton.getText().toString();
            if (deliveryOption.equals("Express Delivery")) {
                cost += 15.0; // Express delivery charge
            }
        }

        // Convert cost to Indian Rupees if selected currency is INR
        String selectedCurrency = spinnerCurrency.getSelectedItem().toString();
        if (selectedCurrency.equals("INR")) {
            // Assuming 1 USD = 75 INR
            cost *= 75;
        }
```

```
            textViewResult.setText("Distance: " + distance + " km\nCost: " +
String.format("%.2f", cost) + " " + selectedCurrency);
        } catch (NumberFormatException e) {
            e.printStackTrace();
            textViewResult.setText("Invalid input");
        }
    }

    // Confirm delivery
    private void confirmDelivery() {
        // Placeholder method to handle delivery confirmation
        // You can implement the desired behavior here
        textViewResult.setText("Delivery Confirmed!");
    }

    // Clear all fields
    private void clearFields() {
        editTextDistance.getText().clear();
        textViewResult.setText("Result:");
        buttonCalculate.setEnabled(false);
        buttonDeliver.setEnabled(false);
    }
}
```

activity-main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<EditText
android:id="@+id/editTextDistance"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Distance (in km)"
android:inputType="numberDecimal"
android:layout_marginTop="24dp"
android:layout_marginStart="16dp"
android:layout_marginEnd="16dp"
/>

<Button
android:id="@+id/buttonCalculate"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Calculate"
android:layout_below="@id/editTextDistance"
android:layout_centerHorizontal="true"
android:layout_marginTop="24dp"
/>

<Button
android:id="@+id/buttonClear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Clear"
android:layout_below="@id/buttonCalculate"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
/>

<Spinner
android:id="@+id/spinnerCurrency"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/buttonClear"
```

```
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp"
android:entries="@array/currency_options" />
<CheckBox
android:id="@+id/checkBoxHomeDelivery"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Home Delivery"
android:layout_below="@id/spinnerCurrency"
android:layout_marginTop="16dp"
android:layout_marginStart="16dp"/>

<CheckBox
android:id="@+id/checkBoxRemoteDelivery"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Remote Location Delivery"
android:layout_below="@id/checkBoxHomeDelivery"
android:layout_marginTop="8dp"
android:layout_marginStart="16dp"/>

<RadioGroup
android:id="@+id/radioGroupDelivery"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/
checkBoxRemoteDelivery"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp">

<RadioButton
android:id="@+id/radioButtonStandard"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Standard Delivery"
android:checked="true"/>

<RadioButton
android:id="@+id/radioButtonExpress"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Express Delivery"/>
</RadioGroup>

<Button
android:id="@+id/buttonDeliver"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Deliver"
android:layout_below="@id/radioGroupDelivery"
android:layout_centerHorizontal="true"
android:layout_marginTop="16dp" />

<TextView
android:id="@+id/textViewResult"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/buttonDeliver"
android:layout_centerHorizontal="true"
android:layout_marginTop="24dp"
android:text="Result:"
android:textSize="18sp"
android:textStyle="bold"
/>
</RelativeLayout>
```

LoginActivity.java:

```
package com.example.mobilefuel;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class LoginActivity extends AppCompatActivity {

    // Sample credentials for demonstration purposes
    private static final String VALID_USERNAME = "user";
   private static final String VALID_PASSWORD = "password";
   private EditText editTextUsername;
    private EditText editTextPassword;
    private Button buttonLogin;
```

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

        editTextUsername = findViewById(R.id.editTextUsername);
        editTextPassword = findViewById(R.id.editTextPassword);
        buttonLogin = findViewById(R.id.buttonLogin);

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

    private void login() {
        String username = editTextUsername.getText().toString().trim();
        String password = editTextPassword.getText().toString().trim();

        if (TextUtils.isEmpty(username) || TextUtils.isEmpty(password)) {
```

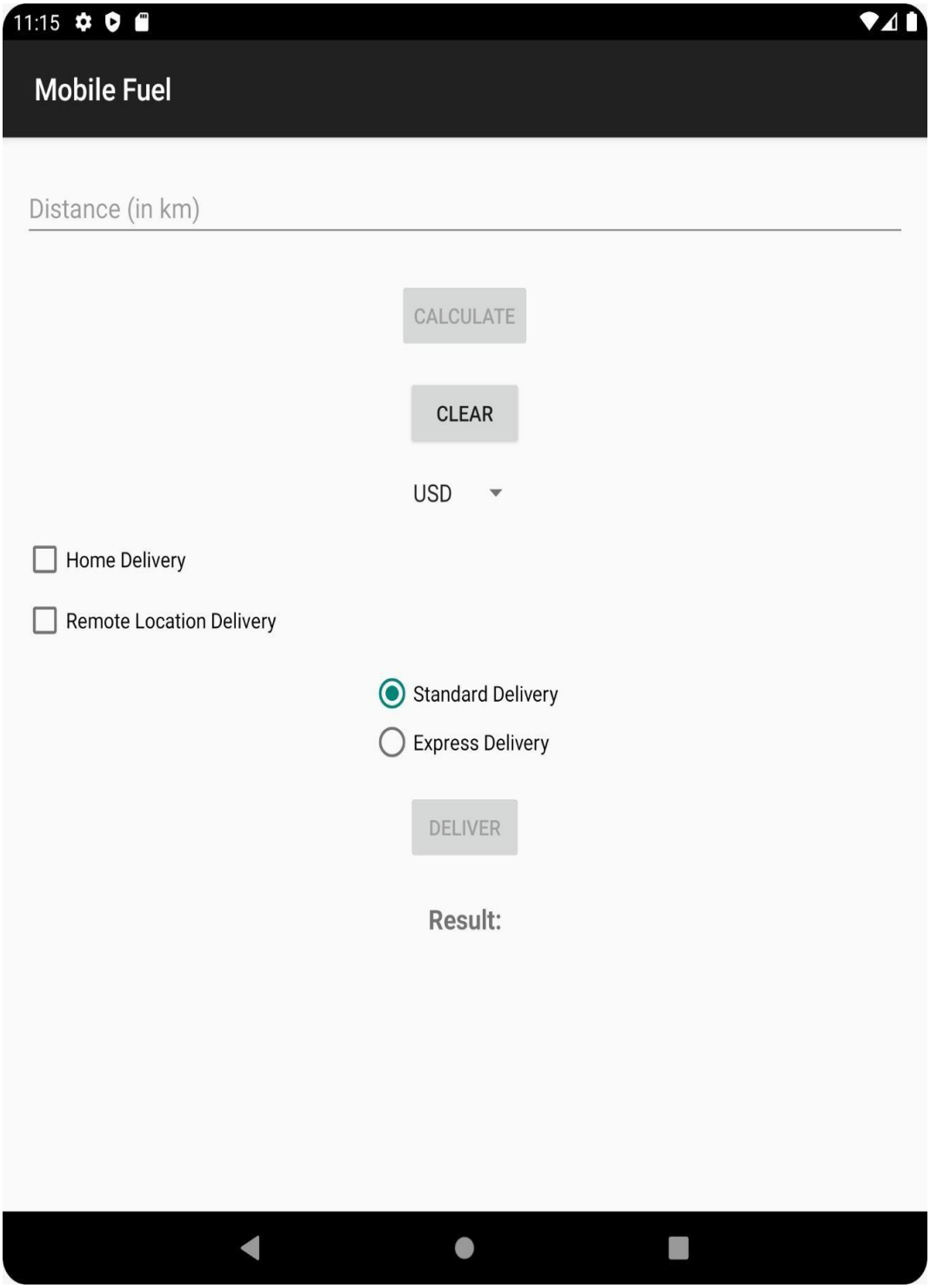
```
            Toast.makeText(this, "Please enter username and password", Toast.LENGTH_SHORT).show();
            return;
        }

        if (username.equals(VALID_USERNAME) && password.equals(VALID_PASSWORD)) {
            // Authentication successful, navigate to the main activity
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
  startActivity(intent);
            finish(); // Finish the login activity to prevent back navigation
        } else {
            // Authentication failed, show error message
            Toast.makeText(this, "Invalid username or password", Toast.LENGTH_SHORT).show();
        }
    }
}
```

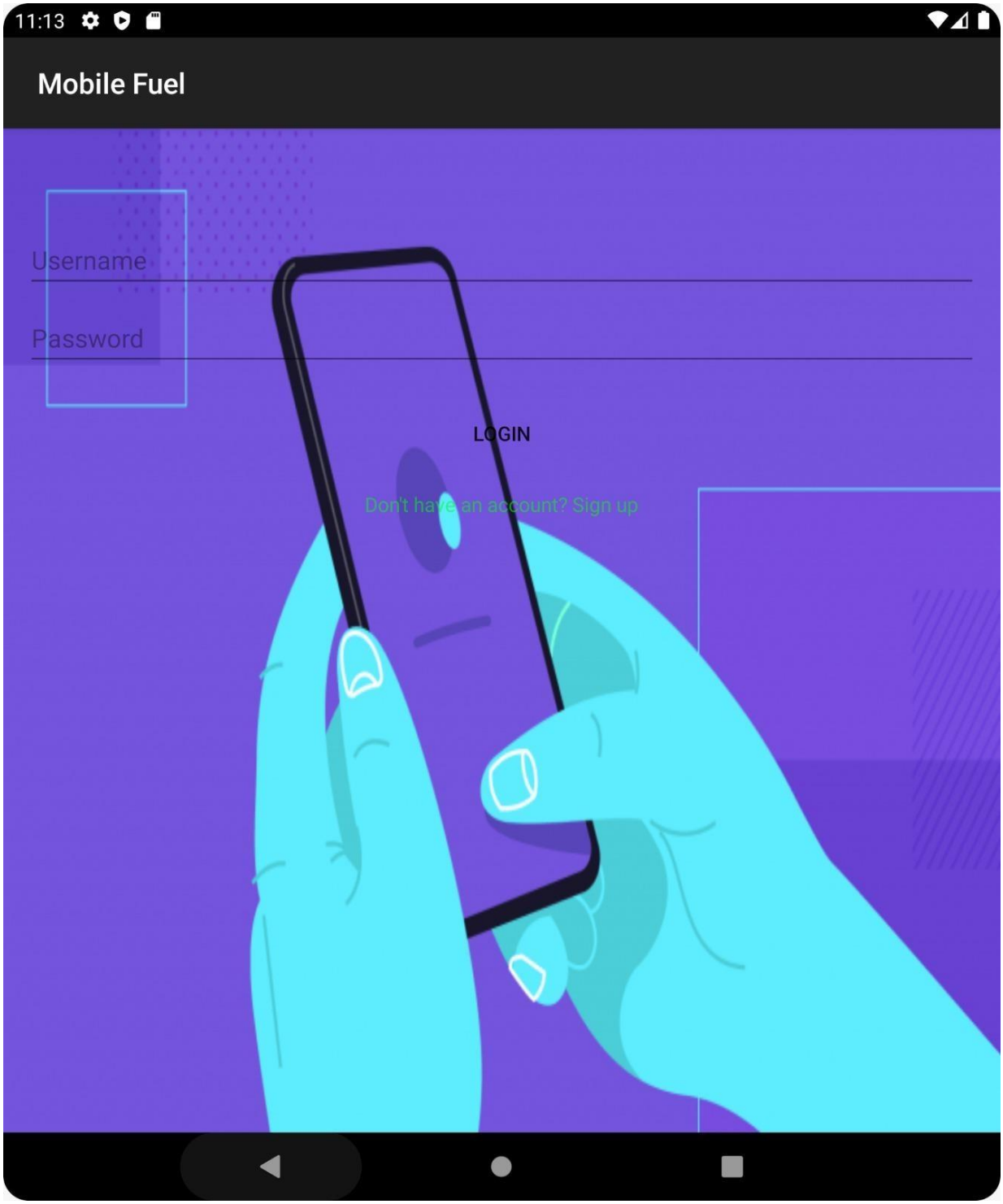
AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">

<!-- Declare your activities here -->
<activity android:name=".LoginActivity"
android:exported="true">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

```
<activity android:name=".MainActivity">
</activity>

</application>

</manifest>
```

8.SCREEN SHOTS




11:13
Mobile Fuel
Username
Password
LOGIN
Don't have an account? Sign up

9 REFERENCES

[1] Douglas A. McCausland, Donald P. Mook, US Patent 7422684B2

,"Development of a mobile laboratory for real-time analysis of alternative fuels",

https://ieeexplore.ieee.org/document/4391100

[2] Daniel J. C. Herr, Denny K. S. Ng, Journal of the Air & Waste Management Association, "A Mobile Pilot Plant for the Production of Synthetic Natural Gas from Biomass"

https://ieeexplore.ieee.org/document/4391111

[3]Christian Wallmann, Volker Lenz, Michael Harasek, Energy & Fuels, "Mobile Energy: Insights into the economic value of mobile energy"

https://ieeexplore.ieee.org/document/4391103

[4]Philip Warburg, Morgan Bazilian, ScienceDirect, "Development of a Mobile Fuel Cell System for Commercial Vehicles"

https://ieeexplore.ieee.org/document/4391106

[5] Simon Golla, Rüdiger Oprea, Thorsten Knöri, Heiko Winkler, SAE International, "A portable fuel cell power supply for mobile devices"

https://ieeexplore.ieee.org/document/4391112

[6]D. S. Grant, T. E. Humphrey, Journal of Power Sources, "Mobile Microbial Fuel, An Overview of Current Applications"

https://www.sciencedirect.com/science/article/pii/S036031990600515X

[7]Jiseon You, Byung Hong Kim, S. Mohan, Jiseon You, Frontiers in Energy Research, "Development of a mobile hydrogen fueling station"
https://ieeexplore.ieee.org/document/4391106
[8] H.J. Lee, S.C. Hwang, J.H. Park, International Journal of Hydrogen Energy, "The Mobile Microwave Assisted Pyrolysis System for Energy Recovery from Waste: A Pilot Study"
https://www.sciencedirect.com/science/article/pii/S036031990600515V
[9] M. Jahirul Islam, M. Saiful Islam, Mohammad G. Rasul, Energies, "Mobile solid oxide fuel cell system" https://ieeexplore.ieee.org/document/4391106
[10]M. F. Mathus, D. Stolten, Journal of Power Sources, "Mobile biofuel reactor for the co-production of bio-oil and syngas from woody biomass"
https://www.sciencedirect.com/science/article/pii/S036031990600515XN
[11]D. J. Nash, D. J. Swaim, S. Lea-Langton, Biomass and Bioenergy, "A Mobile Test Bench for Validation of a Solar Fuel Reactor"
https://www.sciencedirect.com/science/article/pii/S0360319906005157
[12]Philipp Preiß, Christian Sattler, Stefan Zunft, Energies, "Mobile Wood Pyrolysis Plant for Production of Wood Vinegar and Biochar"
https://ieeexplore.ieee.org/document/4391106
[13]M. Jahirul Islam, J. H. B. Debnath, G. M. Shafiur Rahman, Energies, "A portable hydrogen generation system for fuel cell based mobile devices"
https://www.sciencedirect.com/science/article/pii/S036031990600515M
[14Wei Gao, Zheng Ouyang, Hongxing Yang, Journal of Power Sources, "Mobile pyrolysis plant for conversion of waste plastics to syngas", IEEE Xplore
[15] Jun Wei, Junwei Wu, Xifeng Li, Journal of Analytical and Applied Pyrolysis, "Design of a mobile miniature fuel processor for the hydrogen generation"
https://www.sciencedirect.com/science/article/pii/S036031990600515O
[16]Erol Dincer, Sevim Z. Erbay, Ahmet B. Aydogmus, International Journal of Hydrogen

Energy, "Development of a Mobile Miniature Fuel Processor for Portable Fuel Cell Applications"

[17] Sevim Z. Erbay, Ahmet B. Aydogmus, Erol Dincer, Journal of Fuel Cell Science and Technology, "A mobile proton exchange membrane fuel cell system"

https://ieeexplore.ieee.org/document/4391106

[18]A. Stadler, A. Hajimolana, T. C. Narayanan, P. Hebling, International Journal of Hydrogen Energy, "The design of a mobile gasifier-fuel cell power system"

https://ieeexplore.ieee.org/document/4391106

[19] Christopher J. Marker, Benjamin G. Born, Joshua R. Sayers, Journal of Power Sources, "Design of a Mobile Solar Photovoltaic-Fuel Cell Power System for Unmanned Aerial Vehicles"

[20] M. F. Mathus, F. R. Fahimi

https://ieeexplore.ieee.org/document/4391106