1.INTRODUCTION

* 1. SYSTEM OVERVIEW

The “**Heavy Vehicle Inspection and Rental Mobile Application**” developed using Flutter and SQLite, revolutionizes the way users interact with heavy vehicles and their operators. This comprehensive platform seamlessly integrates inspection and rental services for heavy vehicles, providing users with a one-stop solution for their transportation needs. Users can effortlessly find certified and reliable heavy vehicle operators for transportation tasks or inspection services, ensuring the efficiency and safety of their operations.

Through an intuitive user interface built with Flutter, the application allows users to browse and connect with a diverse network of skilled heavy vehicle operators. The integrated inspection feature ensures that users can assess the condition of the vehicles before committing to a rental, enhancing transparency and trust in the process. The application's backend, powered by SQLite, ensures robust data management and retrieval, enabling quick and accurate access to relevant information. Overall, this Heavy Vehicle Inspection and Renting Application streamlines the process of finding, inspecting, and renting heavy vehicles, fostering a more efficient and reliable ecosystem for both operators and users alike.

This application ensures farmers get their desired operators for their fields while cultivating crops, thus application makes those people to connect without hesitation. The application was developed to work with most heavy vehicles and their providers.

* 1. ABOUT THE ORGANIZATION

****

**Branch – Coimbatore**

Accent Techno Soft (ATS) provides a wide range of solutions in IT Consulting, technology and Operations space for our clients. To enhance the business value of our service offerings to our customers, we have formed strategic alliances with industry bodies, technologies vendors and system integrators. Through these partnerships we are able to deliver industry-best end –to-end solutions to our customers. ATS is an ISO Certified Company. Accent specializes in the planning, analysis, and management of business, infrastructure and natural resources. We serve clients who share our belief that high-quality; objective analysis is a prerequisite to resolving complex problems. ATS also provides consultancy services which are strongly committed to ensure that its student gets multiple placements opportunities in organizations. To keep up this commitment, ATS has strategic alliances with most of the leading organizations across the country which will help the students to get placed in premier organizations.

**2. SYSTEM STUDY AND ANALYSIS**

**2.1 EXISTING SYSTEM**

Before the advent of the Heavy Vehicle Inspection and Rental Mobile Application, the heavy vehicle management landscape grappled with inefficiencies and challenges in both inspection and rental processes. The conventional system often relied on fragmented approaches, with operators and users struggling to find a centralized platform for their heavy vehicle needs. Inspection services were disjointed and lacked a standardized framework, making it challenging for users to assess the condition of vehicles consistently. Additionally, the rental process was hindered by a lack of transparency and accessibility, as users had to rely on disparate channels to connect with operators, leading to delays and uncertainties.

2.1.1 DRAWBACKS

Fragmented Approach:

* The conventional system relied on fragmented processes for heavy vehicle management, lacking a centralized platform for both inspection and rental services.

Inspection Services:

* Inspection services were disjointed and lacked a standardized framework, making it difficult for users to consistently assess the condition of heavy vehicles.

Transparency Issues in Rental Process:

* The rental process suffered from a lack of transparency and accessibility. Users had to navigate through various channels to connect with operators, leading to delays and uncertainties.

Data Management:

* Data management was a significant bottleneck in the existing system, with manual record-keeping methods prone to errors and delays.

**2.2 PROPOSED SYSTEM**

The introduction of the "Heavy Vehicle Inspection and Rental Mobile Application" serves as a solution to overcome the challenges and inefficiencies present in the existing heavy vehicle management system. The system aims to revolutionize the way users interact with heavy vehicles and their operators by providing a comprehensive platform that seamlessly integrates inspection and rental services.

**2.2.1 OBJECTIVE**

The objective of the proposed system is to develop a comprehensive and user-friendly platform that integrates inspection and rental services for heavy vehicles. The system aims to revolutionize the way users interact with heavy vehicles and their operators, providing a one-stop solution for transportation needs.

* + 1. **ADVANTAGES**
* Our proposed Heavy Vehicle Inspection and Renting Application provides a unified and centralized platform, powered by Flutter, enabling users to seamlessly connect with certified heavy vehicle operators and access standardized inspection services in one location.
* The application prioritizes transparency in the rental process by offering a user-friendly interface that facilitates direct communication between users and operators. This ensures a clear and efficient rental experience and eliminating the uncertainties and delays present in the existing system.
* Leveraging the capabilities of SQLite, our solution ensures robust data management. This eliminates the manual record-keeping bottlenecks present in the current system and providing a secure and efficient database for storing and retrieving critical information.
* The proposed application is designed with the end-user in mind, offering a seamless experience through Flutter's user-friendly interface. This user-centric design enhances accessibility and usability, promoting a more efficient and satisfying heavy vehicle inspection and rental process for all stakeholders.
  1. **FEASIBILITY STUDY**

The system should undergo three different categories of feasibility studies

* Technical Feasibility
* Operational Feasibility
* Economic Feasibility
* Legal and Regulatory Feasibility

**2.3.1 TECHNICAL FEASIBILITY**

**Development Tools**

The choice of Flutter as the development framework is technically feasible as it allows for cross-platform development, reducing the effort required to build separate applications for different operating systems.

**Database Management**

The use of SQLite for the backend database is technically sound, offering a lightweight and efficient solution for mobile applications.

**2.3.2 OPERATIONAL FEASIBILITY**

**User Interface**

The intuitive user interface designed with Flutter contributes to operational feasibility by ensuring ease of use and navigation for users, even those with limited technical expertise.

**Network Integration**

The ability to connect with a diverse network of heavy vehicle operators enhances operational feasibility by providing users with a broad range of options.

**2.3.3 ECONOMIC FEASIBILITY**

**Cost of Development**

Flutter's open-source nature can contribute to cost savings in development. However, the overall economic feasibility would depend on the specific development and maintenance costs.

**Revenue Generation**

The application's potential to generate revenue through rental transactions and possibly subscription models for operators should be considered.

**2.3.4 LEGAL AND REGULATORY FEASIBILITY**

**Compliance**

The application must comply with relevant laws and regulations related to data privacy, user agreements, and potentially licensing or certification requirements for heavy vehicle operators.

**Contractual Agreements**

Legal feasibility also involves ensuring that contracts between users, operators, and the platform are legally sound and enforceable.

**3.DEVELOPMENT ENVIRONMENT**

**3.1 HARDWARE REQUIREMENT**

Processor : Intel(R) Core(TM) i5-6500T CPU @ 2.50GHz

Installed memory (RAM) : 8 GB

Hard Disk : 160 GB

**3.2 SOFTWARE REQUIREMENT**

Operating System : Windows 10

Front-End : Flutter

Back-End : SQLite

Tools : Android Studio, Visual Studio Code

Database : SQLite

**3.3 SOFTWARE DESCRIPTION**

**FLUTTER**

Flutter is an open-source UI (User Interface) software development framework created by Google. It is designed for building natively compiled applications for mobile, web, and desktop from a single codebase. Flutter was first introduced in 2017 and has gained significant popularity due to its unique approach to creating user interfaces and its ability to provide a consistent experience across different platforms.

Cross-Platform Development Flutter is a powerful framework for building cross-platform applications, meaning a single codebase can be used to create apps for both iOS and Android platforms. This eliminates the need for separate development teams or codebases for each platform, resulting in significant time and cost savings.

Widgets and UI Components Flutter&#39;s core building blocks are widgets, which are used to create the user interface (UI) of the app. Flutter provides an extensive library of pre-designed widgets for various UI elements such as buttons, text inputs, images, lists, and more. These widgets are highly customizable, enabling developers to create consistent and visually appealing interface.

**FEATURES**

**HANDSET LAYOUTS:**

The platform is adaptable to larger, VGA, 2D graphics library, 3D graphic library based on OpenGL ES 2.0 specification, and traditional smartphone layouts.

**STORAGE:**

SQLite, a lightweight relational database, is used for data storage purposes.

**CONNECTIVITY:**

Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, and Bluetooth.

**MESSAGING:**

SMS and MMS are available forms of messaging, including threaded text messaging and now Android Cloud To Device Messaging (C2DM) is also a part of Android push messaging service.

**WEB BROWSER:**

The web browser available in Android is based on the open-source Web kit layout engine, coupled with Chrome’s V8 JavaScript engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

**ADDITIONAL HARDWARE:**

Android can use video/still cameras, touch screens, GPS, accelerometers, gyroscopes, barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerator 2D bit blots (with hardware orientation, scaling, pixel format conversion) and accelerated 3d graphics.

**MULTI-TOUCH:**

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The features were originally disabled at the kernel level(possibly to avoid infringing Apple’s patents on touchscreen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.

**APPLICATIONS:**

Applications are usually developed in the Java language using the Android Software Development Kit, but other development tools are available, including a Native Development Kit for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers and various Cross-platform mobile web applications frameworks.

**SECURITY:**

Android applications run in a sandbox, an isolated area of the operating system that does not have access to the rest of the system’s resources, unless access permissions are granted by the user when the application is installed. Before installing an application, the Play Store displays all required permissions. A gain may need to enable vibration, for example, but should not need to read messages or access the phone book. After reviewing these permissions, the user can decide whether to install the application.

**Android SQLite**

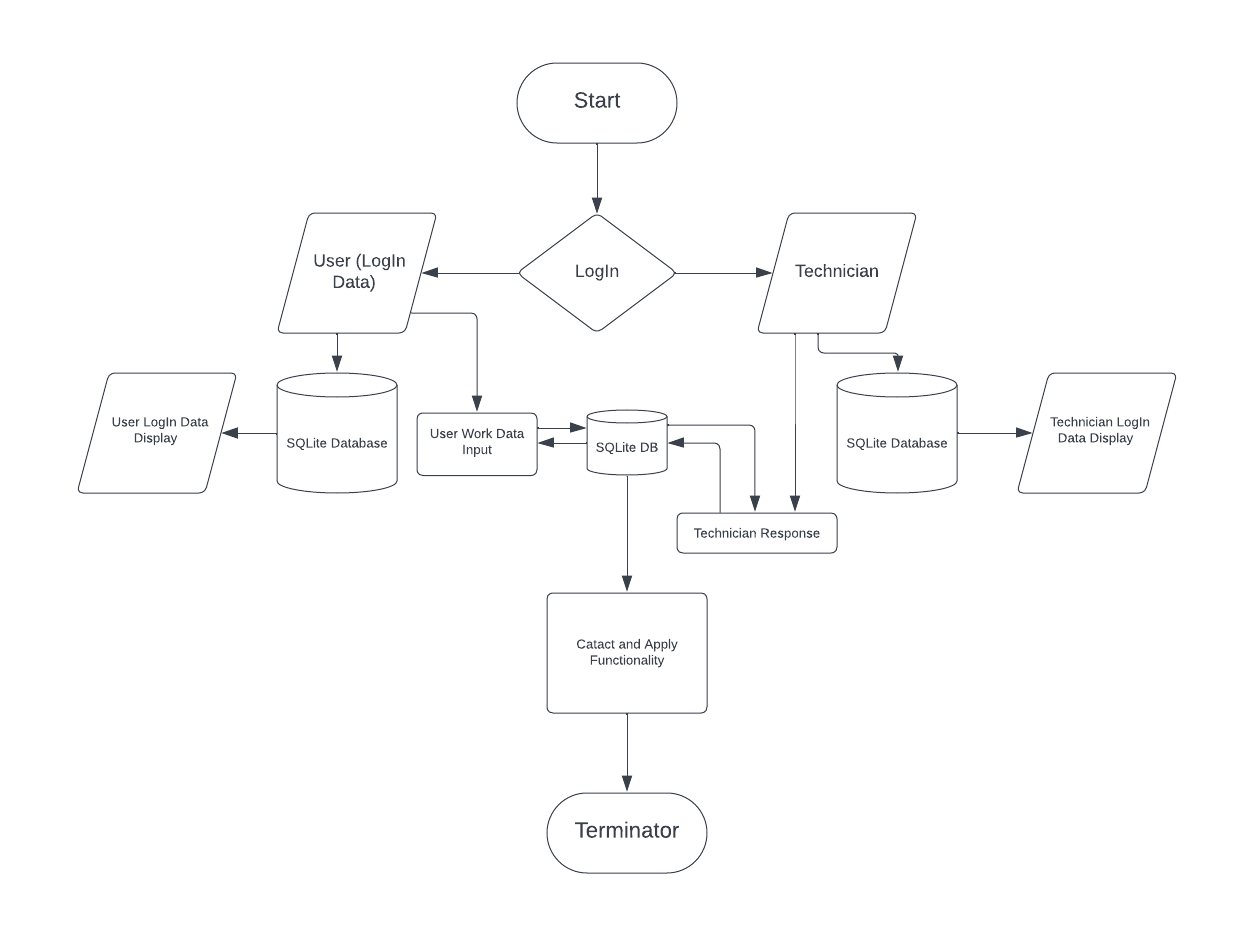
SQLite is an open-source relational database i.e. used to perform database operations on Android devices such as storing, manipulating, or retrieving persistent data from the database. It is embedded in Android by default. So, there is no need to perform any database setup or administration task. SQLite is not directly comparable to client/server SQL database engines such as MySQL, Oracle, PostgreSQL, or SQL Server since SQLite is trying to solve a different problem. Client/server SQL database engines strive to implement a shared repository of enterprise data. They emphasize scalability, concurrency, centralization, and control. SQLite strives to provide local data storage for individual applications and devices. SQLite emphasizes economy, efficiency, reliability, independence, and simplicity. SQLite does not compete with client/server databases. SQLite competes with open ().

If many client programs are sending SQL to the same database over a network, then use a client/server database engine instead of SQLite. SQLite will work over a network file system, but because of the latency associated with most network file systems, performance will not be great. Also, file-locking logic is buggy in many network file system implementations (on both UNIX and Windows). If file locking does not work correctly, two or more clients might try to modify the same part of the same database at the same time, resulting in corruption.

**4.SYSTEM DESIGN**

**4.1 LOGICAL DESIGN**

SYSTEM FLOW DIAGRAM



**4.2 DATABASE DESIGN**

**TABLE NAME :** Customer LOGIN

**DESCRIPTION :** This table stores customer login Details

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| ID | Int |  | It specified the customer id |
| Name | Varchar | 15 | Name of the User |
| E-mail | Varchar | 20 | It Specified the customer email |
| Phone Number | Varchar | 10 | Customer Phone number |
| Location | Varchar | 15 | Customer Location |
| Specialist | Varchar | 10 | Specify the customer specialist |

**TABLE NAME :** Customer Posting Data

**DESCRIPTION :** This table is used to post their required details.

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| User Name | Varchar, | 15 | Name of the user |
| From Date | Int | 20 | From date to start work |
| To Date | Int | 20 | To date to stop work |
| Location | Varchar, | 10 | Working location |
| Zip code | Int | 20 | Zip code of the posting user |
| Address | Varchar | 30 | Address of the posting user |
| Questions | Varchar | 30 | Questions for particular problems |

**TABLE NAME** **:** Technician Posting Data

**PRIMARY KEY :** ID

**DESCRIPTION :** This table is used to book their vehicles and also add their technician details for their user needs.

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| Job Name | Text | 15 | Name of their job |
| Person Name | Text | 15 | Name of the Technician |
| Contact Number | Int | 10 | Technician Contact number |
| Skills | Text | 20 | Skills of their Technician |
| Experience | Int |  | Working Experience |
| Specialist | Text | 20 | Specializing In Technician |
| Address | Text | 30 | Address of the Technician |

**4.3 INPUT DESIGN**

Input design is the process of converting an external user-oriented description of the input system into a machine-oriented format. The source document is prepared for the input of data to make the entry accurate and impact the source document was prepared. The data elements were sent out in a system that the data entry operator would easily follow. The input was designed with the following objectives

* User Registration and Profile Creation
* Search and Filtering Criteria
* Booking and Rental Information
* Vehicle Inspection Checklist
* Communication Channels

The Major activities that were done during the input design are

* The data are collected from its source.
* Data are converted to the mobile acceptable form.
* The converted data was verified.
  1. **4.4 OUTPUT DESIGN**

Output design is a process that involves designing necessary output that has to be given to various users according to their requirements, Efficient, intelligible output design will improve the system's relationship with the user and help indecision-making. Since the reports are directly required by the management to make decisions and to draw conclusions, they must be designed with care for the user. The options for the output and report are given in the system menu. When designing output, system analyst must accomplish the following

* Determine the information to present.
* Arrange the presentation of information acceptable format.
* Determine how to distribute the output.

**5. SYSTEM DEVELOPMENT**

**5.1 MODULE DESCRIPTION**

**System Modules**

The system comprises one major module with its sub-modules as follows:

**User:**

**Home Page:** It consists of suggestions where people can review their type of need in each suggestion

**Posting Page:** In this page users can input specific data for the need of technician according to their work

**Booking Page:** After going through various number of profile they have posted in the application, users can select the correct candidate for their required need

**Account Page:** The user login data can be shown over here

**Technician:**

**Home Page:** The posted data from user can be shown over here and technician can bid the required or correct position in the section

**Booking Page:** The booking page shows the selected posts

**Bids Page:** Technician can bid the work accordingly.

**Account Page:** The user login data can be shown over here

**6. SYSTEM TESTING**

Testing is vital to the success of the system. System Testing makes a logical assumption that, if all the parts of the system are correct, the goal will be achieved. Its basic function is to find the errors in the software by examining all possible loopholes. The goal of testing is to point out uncover requirements, design or coding errors or invalid acceptance or storage of data.

There are different types of system testing

* Unit Testing
* Widget testing
* Integration Testing

**6.1 UNIT TESTING**

Unit testing in Flutter involves testing individual units of code, such as functions, methods, or classes, in isolation to ensure they behave as expected. Flutter provides the test package for writing and executing unit tests. The goal is to ensure that each unit of code behaves correctly in isolation. By regularly running unit tests as part of your development workflow, you can catch issues early and maintain a high level of code quality in your Flutter application.

**6.2 WIDGET TESTING**

* 1. Widget testing in Flutter involves testing individual widgets and their interactions within a Flutter app. It's a type of testing that focuses on the visual and interactive aspects of your app's user interface. Widget tests are particularly useful for ensuring that your widgets render correctly, respond to user interactions as expected, and maintain their behavior across different app states. By writing and running widget tests regularly, you can catch visual and interactive issues early in the development process and maintain a stable user interface.
  2. **6.3 INTEGRATION TESTING**
  3. Integration testing in Flutter involves testing the interaction between different components, screens, or widgets of your app to ensure they work together as expected. Integration tests help you ensure that different parts of your app interact correctly and that your app's overall functionality is not compromised as you make changes or add new features. By regularly running integration tests, you can catch issues related to navigation, state management, and widget interactions early in the development process.
  4. **7. SYSTEM IMPLEMENTATION AND MAINTENCE**

**SYSTEM IMPLEMENTATION**

System Implementation is the stage of the project when the theoretical design is turned into practical system. After proper testing and validation, system implementation should be done. System implementation includes all those activities that take a place to convert an old system to the new system.

The "Heavy Vehicle Inspection and Rental Mobile Application" represents a groundbreaking advancement in the realm of heavy vehicle operations. Developed using Flutter and SQLite, this innovative platform seamlessly integrates inspection and rental services, presenting users with a comprehensive solution for their transportation needs. The application facilitates an effortless search for certified and reliable heavy vehicle operators, ensuring the efficiency and safety of transportation tasks and inspection services.

A key feature of the application is the integrated inspection functionality, allowing users to assess the condition of vehicles before committing to a rental. This not only enhances transparency but also fosters trust in the rental process.

The application's impact extends to the agricultural sector, where it plays a vital role in connecting farmers with operators for their fields during crop cultivation. By facilitating this connection without hesitation, the application contributes to the efficiency and success of agricultural operations. Furthermore, the application is designed to work seamlessly with a wide array of heavy vehicles and their providers, ensuring versatility and adaptability across the industry.

MAINTENANCE ACTIVITIES

This project involves ongoing efforts to ensure the system's efficiency, security, and adaptability. Regularly identify and address any software bugs or issues reported by users or discovered through testing. Continuously monitor the system's performance to identify and address any performance bottlenecks. Optimize code, database queries, and server configurations to maintain optimal speed and responsiveness.

Implement routine data backup procedures to prevent data loss in the event of system failures. Test data recovery processes to ensure their effectiveness and reliability. Maintain robust monitoring tools and logs to track system performance, errors, and user activities.

**8.CONCLUSION AND ENHANCEMENT**

**8.1 CONCLUSION**

Thus, the application is designed in a user-friendly manner. It is complemented by call functionality, signifies a pivotal shift towards modernizing and streamlining the inspection processes for heavy vehicles. Flutter's robust cross-platform capabilities ensure a seamless and consistent user experience on both iOS and Android devices, making the application accessible to a wide range of users, from vehicle inspectors to fleet managers. The choice of SQLite for data management provides a lightweight yet powerful solution for storing and retrieving critical inspection data. Incorporating call functionality within the application facilitates direct communication between inspectors, vehicle owners, and service providers, enhancing the clarity and speed of addressing identified issues. This level of integration not only speeds up the inspection process but also contributes to maintaining high safety standards by ensuring that vehicle issues are promptly and effectively communicated and resolved.

**8.2 FUTURE ENHANCEMENT**

Looking ahead, the heavy vehicle inspection application built with Flutter and SQLite, and featuring call functionality, is poised for significant future enhancements that could revolutionize the way heavy vehicle inspections are conducted. A major area for development is the integration of advanced diagnostic tools and Internet of Things (IoT) technology. By connecting the application directly with vehicle diagnostic systems, inspectors could automatically gather critical data on a vehicle's condition in real-time, streamlining the inspection process and enhancing the accuracy of assessments. IoT integration would also enable continuous monitoring of vehicle health, facilitating preventative maintenance and reducing the likelihood of critical failures on the road.

Another promising direction for future enhancement involves the application of artificial intelligence (AI) and machine learning (ML) algorithms. These technologies could be used to analyze historical inspection data, identifying patterns and predicting potential issues before they become serious problems. This predictive maintenance capability would not only improve vehicle safety and reliability but also help fleet managers optimize maintenance schedules, reducing downtime and operational costs. Additionally, implementing blockchain technology could enhance the security, transparency, and efficiency of inspection records management. By creating an immutable ledger of inspection data, stakeholders can ensure the authenticity and accuracy of vehicle histories, boosting trust among inspectors, vehicle owners, and regulatory bodies. These future enhancements would position the heavy vehicle inspection application at the forefront of digital transformation in the transportation sector. By leveraging cutting-edge technologies to improve the efficiency, accuracy, and predictive capabilities of vehicle inspections, the application will contribute to safer roads and more sustainable heavy vehicle operations, aligning with broader industry trends towards automation and data-driven decision-making.

**9.BIBLIOGRAPHY**

**BOOK REFERENCES**

* Das, S., and A. Dutta. Knowledge Extraction from Trans-portation Research Thesaurus. Presented at 97th AnnualMeeting of the Transportation Research Board, Washing-ton, D.C., 2018.33.
* Mehrotra, S., and S. Roberts. Identification and Validation of Themes from Vehicle Owner Complaints and Fatality Reports using Text Analysis. Presented at 97th AnnualMeeting of the Transportation Research Board, Washing-ton, D.C., 2018.
* Napoli, Marco L. Beginning Flutter: A Hands-On Guide to App Development. A press, 2020.
* NHTSA/ODI Databases. National Highway Traffic Safety Administration, Office of Defects,Investigation,Washington,D.C.http://www.odi.nhtsa.dot.gov/downloads/.Accessed July 27, 2018.4.

**WEBSITE REFERENCES**

* https://flutter.dev/docs
* https://flutter.dev/docs/development/ui/widgets
* https://pub.dev/
* https://dev.to/t/flutter

**ANNEXURE(S)**

**A.SAMPLE CODING**

**Home Page**

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:flutter application\_1/components/CustomerUserInputBox.dart';

import 'package:flutter\_application\_1/components/Techicianbottom.dart';

import 'package:flutter\_application\_1/components/bottom\_nav\_bar.dart';

import 'package:flutter\_application\_1/database/customerdatabases.dart';

import 'package:flutter\_svg/flutter\_svg.dart';

import 'package:gap/gap.dart';

import 'package:get/get.dart';

import 'package:google\_fonts/google\_fonts.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

class cususerlogin extends StatefulWidget {

const cususerlogin({super.key});

@override

State<cususerlogin> createState () => \_cususerloginState();

}

class \_cususerloginState extends State<cususerlogin> {

customerdb cuslogin = Get.put(customerdb());

TextEditingController controllername = TextEditingController();

TextEditingController controlleremail = TextEditingController();

TextEditingController controllerphonenum = TextEditingController();

TextEditingController controllerlocation = TextEditingController();

TextEditingController controllerspecialist = TextEditingController();

\_storeOnboardInfo() async{

print('Shared PRef called');

int isviewed = 0;

SharedPreferences prefs = await SharedPreferences.getInstance();

await prefs.setInt('onboard', isviewed);

print(prefs.getInt('onboard'));

}

@override

Widget build(BuildContext context) {

final screenSize = MediaQuery.sizeOf(context);

return Scaffold(

appBar: AppBar(automaticallyImplyLeading: false,),

body: SingleChildScrollView(

scrollDirection: Axis.vertical,

child: Column(

children: [

Text(

'Iron Knight',

textAlign: TextAlign.center,

style: GoogleFonts.poppins(

fontSize: 30,

fontWeight: FontWeight.bold,

color: Colors.blue),

),

Gap(25),

Column(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

Text(

'Register Your Account',

style: GoogleFonts.poppins(

fontSize: 25,

fontWeight: FontWeight.bold,

color: Colors.black),

),

],

),

Gap(3),

Row(

mainAxisAlignment: MainAxisAlignment.center,

children:

Text(

'Please Select Your User Type',

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.grey),

),

],

),

Gap(15),

UserInputbox(

txt: 'Name',

customerController: controllername,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

UserInputbox(

txt: 'Email',

customerController: controlleremail,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15)

UserInputbox(

txt: 'phonenumber',

customerController: controllerphonenum,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(10),

UserInputbox(

txt: 'location',

customerController: controllerlocation,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

UserInputbox(

txt: 'specialist',

customerController: controllerspecialist,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(35),

TextButton(

onPressed: () async {

Map<String, dynamic> login = {

'name': controllername.text,

'email': controlleremail.text,

'phonenumber': controllerphonenum.text,

'location': controllerlocation.text,

'specialist': controllerspecialist.text,

};

await cuslogin.insertdata(login);

},

child: Row(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Login',

textAlign: TextAlign.end,

style: GoogleFonts.poppins(

fontSize: 25,

fontWeight: FontWeight.bold,

color: Colors.black),

),

],

),

),

Gap(10),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Column(

children: [

InkWell(

onTap: () async{

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => BottomNavBar()),

);

HapticFeedback.lightImpact();

await \_storeOnboardInfo();

}

child: Container(

height: 50,

width: 180,

decoration: BoxDecoration(

color: Color.fromARGB(255, 213, 230, 237),

borderRadius: BorderRadius.circular(10),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

SvgPicture.network(

'https://www.svgrepo.com/show/512729/profile-round-1342.svg',

height: 15,

width: 15,

),

Text(

'Customer',

textAlign: TextAlign.center,

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Color.fromARGB(255, 2, 99, 177)),

),

],

),

),

),

],

),

Column(

children: [

InkWell(

onTap: () async{

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => Techicianbottom()));

HapticFeedback.lightImpact();

await \_storeOnboardInfo();

},

child: Container(

height: 50,

width: 180,

decoration: BoxDecoration(

color: Color.fromARGB(255, 213, 230, 237),

borderRadius: BorderRadius.circular(10),

),

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

SvgPicture.network(

'https://www.svgrepo.com/show/22829/soldier.svg',

height: 15,

width: 15,

),

Text(

'Techician',

textAlign: TextAlign.center,

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Color.fromARGB(255, 2, 99, 177)),

),

],

),

),

),

],

)

],

)

],

),

),

);

}

}

**Customer Data upload**

import 'package:flutter/material.dart';

import 'package:flutter\_application\_1/components/CustomerUserInputBox.dart';

import 'package:flutter\_application\_1/components/bottom\_nav\_bar.dart';

import 'package:flutter\_application\_1/constant/theme.dart';

import 'package:flutter\_application\_1/database/customerAI.dart';

import 'package:flutter\_svg/svg.dart';

import 'package:gap/gap.dart';

import 'package:get/get.dart';

import 'package:google\_fonts/google\_fonts.dart';

class UserDataUploadPage extends StatefulWidget {

const UserDataUploadPage({super.key});

@override

State<UserDataUploadPage> createState() => \_UserDataUploadPageState();}

class \_UserDataUploadPageState extends State<UserDataUploadPage> {

customerAI Customerdatatable = Get.put(customerAI());

TextEditingController controllerVechicle = TextEditingController();

TextEditingController controllerPersonName = TextEditingController();

TextEditingController controllerFromDate = TextEditingController();

TextEditingController controllerToDate = TextEditingController();

TextEditingController controllerLocation = TextEditingController();

TextEditingController controllerZipcode = TextEditingController();

TextEditingController controllerAddress = TextEditingController();

TextEditingController controllerQuestion = TextEditingController();

@override

Widget build(BuildContext context) {

final screenSize = MediaQuery.sizeOf(context);

return Scaffold(

appBar: AppBar(

leading: Padding(

padding: const EdgeInsets.all(14),

child: InkWell(

onTap: () {

Navigator.push(context,

MaterialPageRoute(builder: (context) => BottomNavBar()));

},

child: SvgPicture.network(

'https://www.svgrepo.com/show/533620/arrow-sm-left.svg',

),

),

),

title: Text(

'Create New Post',

style: GoogleFonts.poppins(

fontSize: 15, fontWeight: FontWeight.bold, color: Colors.black),

),

),

body: SingleChildScrollView(

scrollDirection: Axis.vertical,

child: Padding(

padding: const EdgeInsets.only(left: 10, right: 10),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Vechicle Description',

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.black),

),

Gap(15),

Row(

children: [

SvgPicture.network(

'https://www.svgrepo.com/show/427420/cement-concrete-concreting.svg',

height: 35,

width: 20),

Gap(10),

UserInputbox(

txt: 'Vechicle',

customerController: controllerVechicle,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

],

),

Gap(15),

Text(

'Appointment Name',

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.black),

),

Gap(10),

UserInputbox(

txt: 'Person Name',

customerController: controllerPersonName,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

Text(

'Prefered Data Range',

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.black),

),

Gap(10),

Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

UserInputbox(

txt: 'From Date',

customerController: controllerFromDate,

click: () {},

height: screenSize.height \* 0.04,

width: screenSize.width \* 0.4),

UserInputbox(

txt: 'To Date',

customerController: controllerToDate,

click: () {},

height: screenSize.height \* 0.04,

width: screenSize.width \* 0.4),

],

),

Gap(15),

Text(

'Equipment Location',

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.black),

),

Gap(10),

UserInputbox(

txt: 'Location',

customerController: controllerLocation,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

UserInputbox(

txt: 'ZipCode',

customerController: controllerZipcode,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

UserInputbox(

txt: 'Address',

customerController: controllerAddress,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8),

Gap(15),

Text(

'Specify Inspection Question',

textAlign: TextAlign.start,

style: GoogleFonts.poppins(

fontSize: 15,

fontWeight: FontWeight.bold,

color: Colors.black),

),

Gap(10),

UserInputbox(

txt: 'Question',

customerController: controllerQuestion,

click: () {},

height: screenSize.height \* 0.05,

width: screenSize.width \* 0.8,

),

],

),

),

),

floatingActionButtonLocation: FloatingActionButtonLocation.endFloat,

floatingActionButton: FloatingActionButton.extended(

label: Text(

'Post Detail',

textAlign: TextAlign.center,

style: GoogleFonts.poppins(

fontSize: 25,

fontWeight: FontWeight.w300,

color: cWhite,

),

),

onPressed: () async {

Map<String, dynamic> data = {

'Vechicle': controllerVechicle.text,

'personname': controllerPersonName.text,

'Fromdate': controllerFromDate.text,

'Todate': controllerToDate.text,

'Location': controllerLocation.text,

'Zipcode': controllerZipcode.text,

'Address': controllerAddress.text,

'Question': controllerQuestion.text

};

await Customerdatatable.insertdata(data);

},

backgroundColor: Colors.blue,

elevation: 0,

),

);

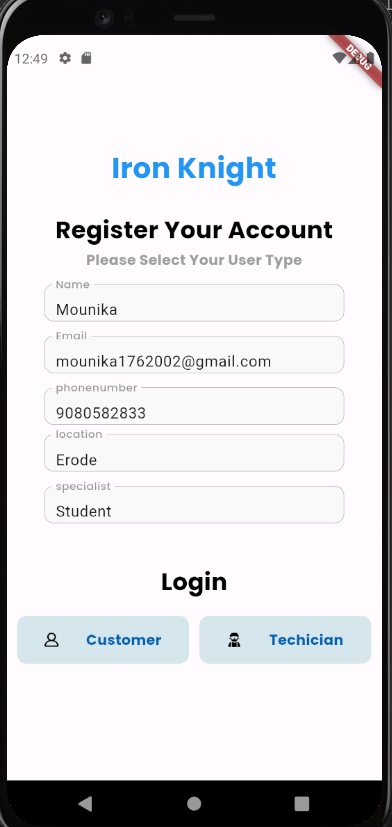
}

}

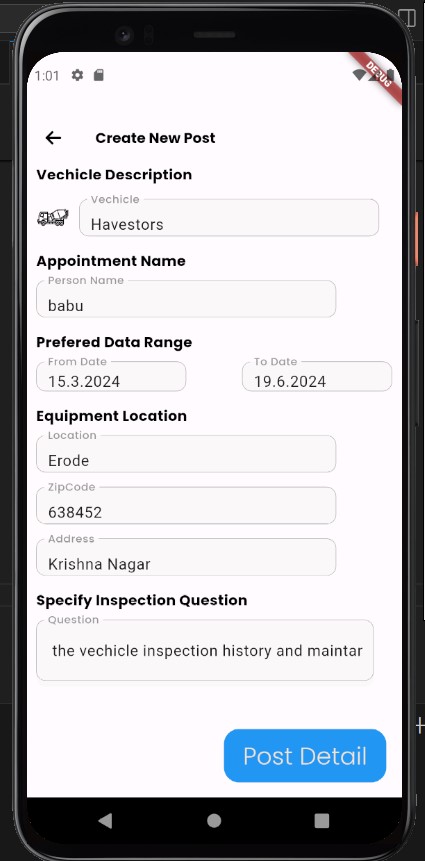
**B. INPUT /OUTPUT FORMS**

**INPUT FORMS**

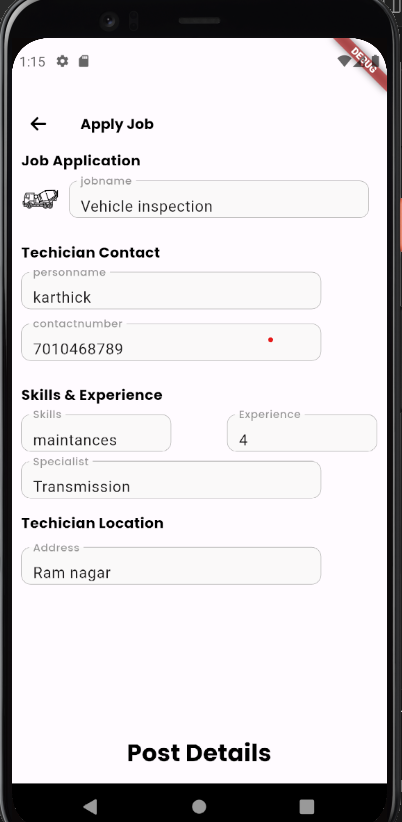
**LOGIN PAGE**

* 1. 

**USER DATA UPLOAD PAGE**

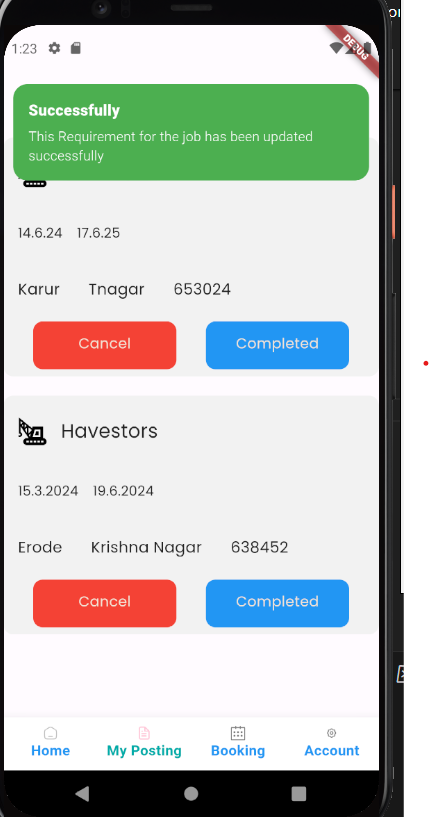


**TECHNICIAN BOOKING PAGE**

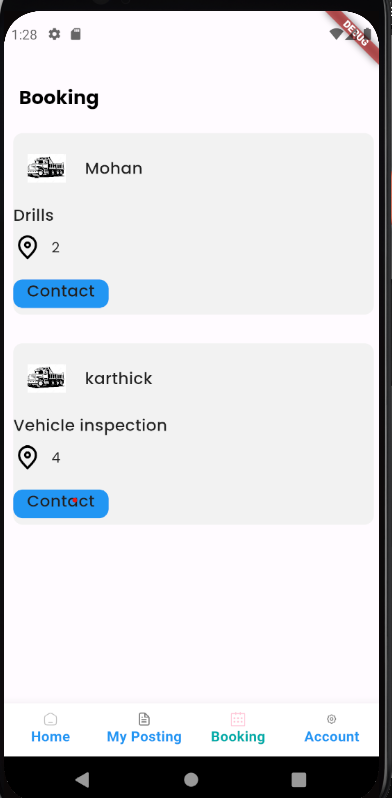
****

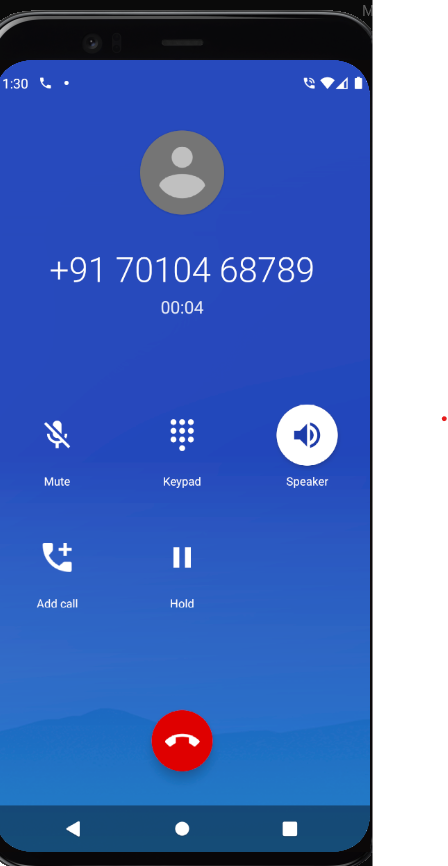
**OUTPUT FORMS:**

**Customer Posting Page**

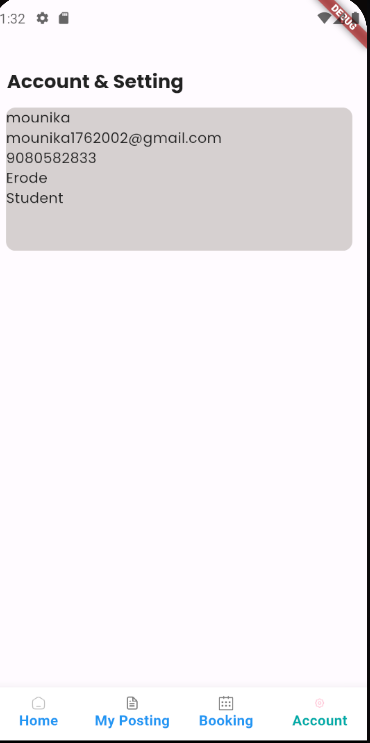


**Customer booking page**

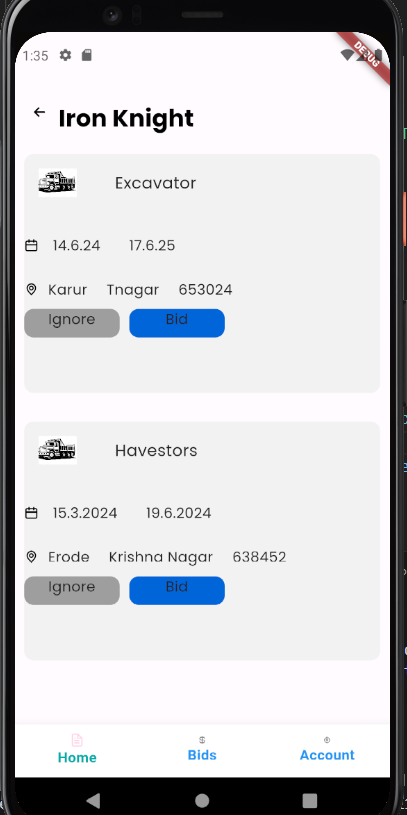
****

****

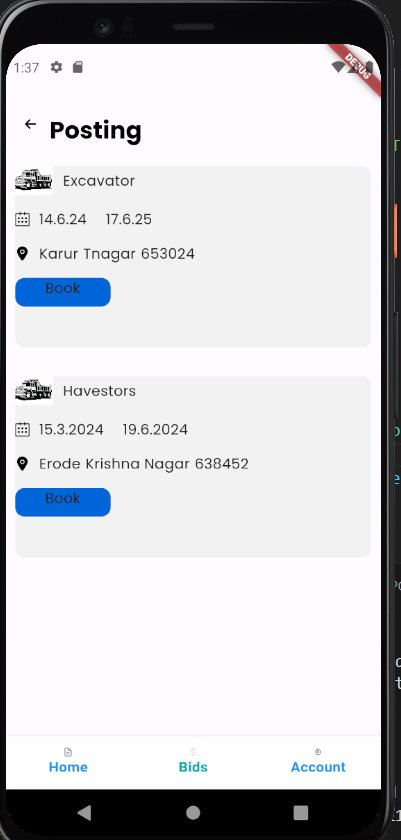
**CUSTOMER ACCOUNT PAGE**

****

**TECHNICIAN HOME PAGE:**

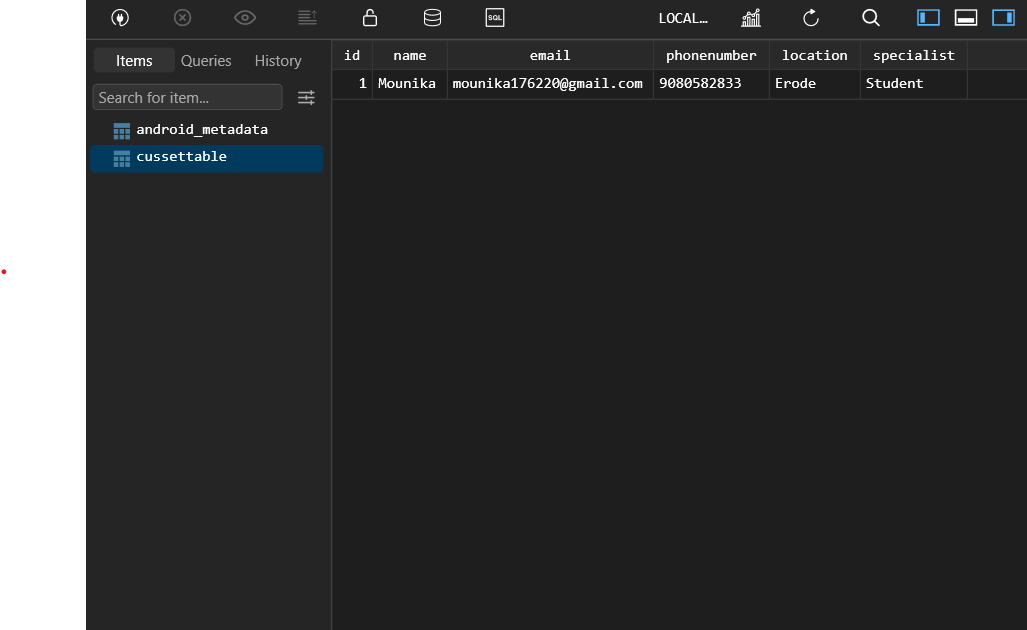
****

**TECHNICIAN BIDS:**

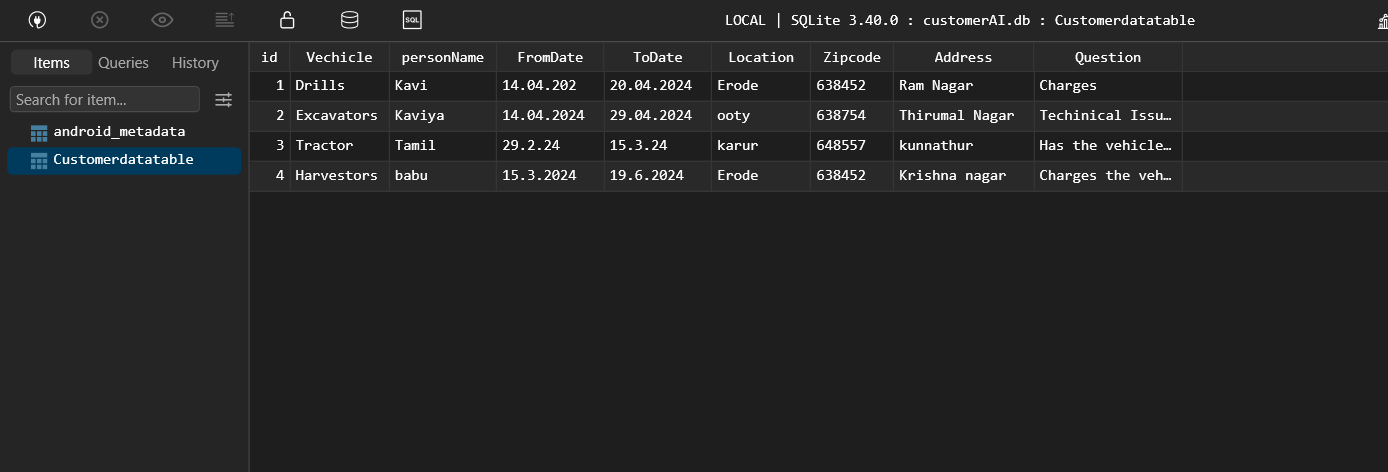
****

**REPORTS**

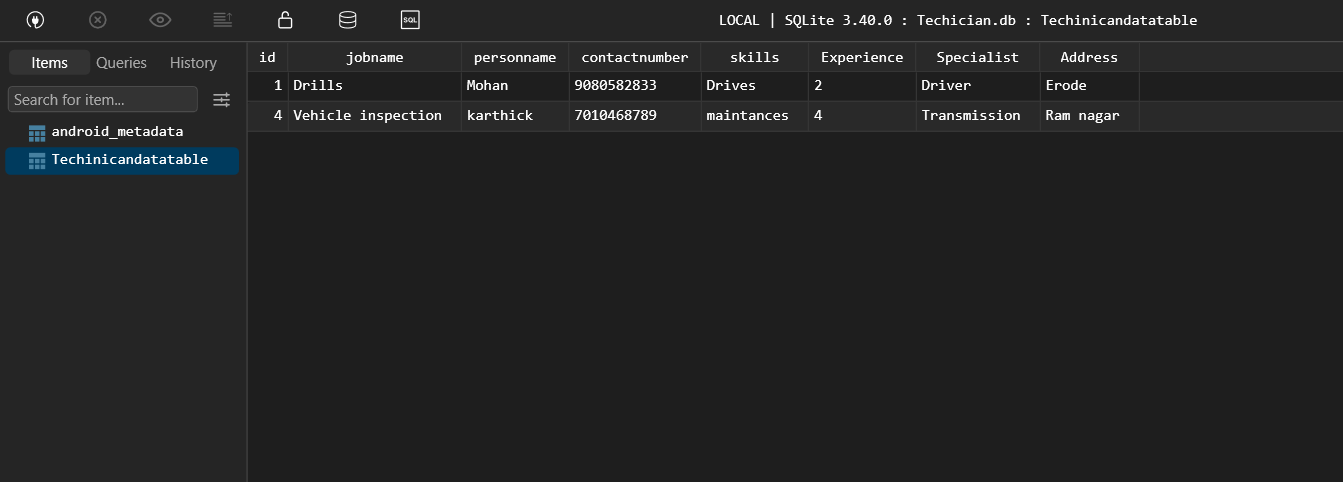
**LOGIN**

****

**CUSTOMER**

****

**TECHNICIAN**

****