EV COMPANION

MAIN PROJECT REPORT

In partial fulfilment of requirements for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

of

UNIVERSITY OF CALICUT

Submitted by

SAYANA T S (AIAVBCA014)

Under the supervision of

Ms. JABIN T H

March 2024

DEPARTMENT OF COMPUTER APPLICATION

M.E.S ASMABI COLLEGE , P. VEMBALLUR

(2021-2024)

M.E.S ASMABI COLLEGE, P. VEMBALLUR

Affiliated to University of Calicut

CERTIFICATE

This is certify that the project entitled "EV COMPANION" done by **SAYANA T S (AIAVBCA014)** during the year 2021-2024 in partial fulfillment of the requirement for the award of the degree of Bachelor of Computer Application - University of Calicut.

MS. JABIN T.H **MS JABIN T.H**

HEAD OF THE DEPARTMENT **FACULTY IN CHARGE**

Date:

Submitted for Examination on held on:……………………………………………….

Internal Examiner **External Examiner**

DECLARATION

Do hereby declare that the project entitled "**EV COMPANION**" submitted to the Department of Computer Application, **M.E.S. ASMABI College, P. Vemballur** is an original dissertation done by me under the supervision of **MS. JABIN T. H**

SAYANA T S (AIAVBCA014)

Date:

Place:

ABSTRACT

"EV COMPANION" is an application for to find nearest electric vehicle charging station and booking a slot. The application is used to insert vehicle’s details, find out the location of the nearest charging station and to book a slot. The electric vehicle industry is going a long way, and the industry is making diverse types of vehicles like scooters, cars, rikshaw, transport vehicles, etc. that run-on charging. All EV riders will go to the station to get their vehicle charged. keeping EV charging station finder and slot booking app It will help riders to find stations and book slots. In this rider can book a charging slot on a particular date or time. Riders can pay according to faster and slow charging

KEYWORDS: Charging Station, Slot Booking, Ev Companion, Electric Vehicles

INDEX

INTRODUCTION

1.Introduction

1.1 Introduction to EV Companion

EV COMPANION is an application to find the nearest electric vehicle charging station and book a slot. The application is used to insert vehicle’s details, find out the location of the nearest charging station and to book a slot.

Due to the limitation of electrical power distribution network, Electric Vehicles charging stations are limited and to find them is hard for new EV owners. To provide information to users about the charging stations and to help users to navigate, this application helps the EV owners with these processes. This Proposed EV Companion Application helps EV owners to locate a charging station near them and to book a slot. It also provides various payment methods for smooth transaction in the process of booking a slot.

1.2 Objectives of EV Companion

- To find a nearby charging station where, location is given as input effectively and efficiently
- To reserve a slot in advance for charging the Electric Vehicle
- Customers can prefer their own time slot according to their convenience
- To store the details of user electric vehicle
- To allow user to book a slot and make a payment in advance
- Customer can choose a charging station according to their preference by viewing ratings
- To provide details about charging stations

1.3 Project Overview

The transition towards electric vehicles (EVs) is gaining momentum worldwide, driven by concerns about environmental sustainability and energy efficiency. One of the critical challenges faced by EV owners is the availability and accessibility of charging stations. To address this, we propose the development of an Electric Vehicle Charging Station Finder and Slot Booking Application. This application aims to provide EV owners with a seamless experience for locating nearby charging stations and booking available slots in advance, thereby promoting the widespread adoption of electric vehicles.The application will utilize GPS technology to identify the user's current location and display nearby charging stations on an interactive map. Users can view essential details such as station type, availability, and supported charging standards.Users can browse through available time slots at charging stations and book them in advance. This feature ensures that users have guaranteed access to charging facilities during their planned trips, eliminating the uncertainty associated with finding available slots upon arrival.The application will require users to create accounts to access advanced features such as slot booking and personalized preferences. User profiles will store information like vehicle type, charging preferences, and payment details for a seamless booking experience.

BACKGROUNG STUDY

2.1 ARCHITECTURE

The architecture of the Electric Vehicle Charging Station Finder and Slot Booking Application comprises several interconnected components working together to provide a seamless user experience. Here's an overview of the architectural components:

Frontend: Flutter will be used to develop the client interface, providing a single codebase for both iOS and Android platforms.

Backend: Python Django serves as the backend framework, responsible for handling business logic, data processing, and communication with the client interface and database.

Database: The application uses SQLite database. It is an embedded database.The database stores user profiles, charging station details, booking information, and reviews in structured tables, accessible by the backend server.

Security: It uses authentication mechanism.

2.2 Drawbacks

- **Limited Coverage and Accuracy of Charging Station Data:** One drawback is the potential limited coverage and accuracy of charging station data, especially in remote or less populated areas. The application heavily relies on available data sources, and if the database lacks comprehensive and up-to-date information, users may struggle to find suitable charging stations.
- **Dependency on Internet Connectivity:** The application's functionality is heavily reliant on internet connectivity. Users may face challenges accessing the application or booking charging slots in areas with poor or no network coverage. Additionally, server downtimes or maintenance can disrupt the application's availability, impacting user experience
- **Reliability of Booking System:** The reliability of the slot booking system is crucial for user satisfaction. However, technical issues such as server errors, synchronization issues between frontend and backend systems, or payment gateway failures can lead to booking discrepancies or failed transactions, causing frustration among users.

- **User Frustration and Disappointment:** Users rely on the search bar as a primary tool for quickly finding relevant information. If it doesn't work properly, users may become frustrated and disappointed with the application's performance, leading to a negative overall experience.

SYSTEM ANALYSIS

3.1 Feasibility Study

Feasibility is defined as the practical extent to which a project can be performed successfully. Evaluate feasibility; a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software.

Information such as resources availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study.

The objective of the feasibility study is to establish the reasons for developing software that is acceptable to users, adaptable to change and conformable to established.

3.2 Technical Feasibility Study

Technical feasibility is a measure of how feasible the project is technically. The effort and technology included in the conventional system is not needed as the entire process is automated seems that our project being not that much to code and execute, it is sufficient to complete this project called “EV COMPANION” by duration of three months.

The technical feasibility study examined the feasibility of developing the Electric Vehicle Charging Station Finder and Slot Booking Application. It analyzed software tools, mapping service integration, authentication mechanisms, payment gateway integration, scalability, mobile platform compatibility, development team skills, resource availability, and cost considerations. Based on the study's results, decisions were made to address technical challenges and ensure successful application implementation.

3.3 Economical Feasibility Study

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. This project is Ecnomically feasible.

3.4 Operational Feasibility Study

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. This project is Operationally feasible.

Operational feasibility also performs the following tasks.

1. Determines whether the problems anticipated in user requirements are of high priority.

2. Determines whether the solution suggested by the software development team is acceptable.

3. Analyses whether users will adapt to new software.

4. Determines whether the organization is satisfied by the alternative solutions proposed by the software device

REQUIREMENT ANALYSIS AND SPECIFICATION

4.1 REQUIREMENTS ANALYSIS AND SPECIFICATION

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant, and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management.

Requirement analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or group of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mild user expectations to fit the requirements. Analysis is a team effort that demands a combination of hardware, software, and human factor engineering expertise as well as skill in dealing with people.

4.2 HARDWARE REQUIREMENTS

The hardware configuration is an important task related to software development. Insufficient random-access memory may affect adversely the speed of the entire system. The major hardware requirements are:

Processor	: Intel Core i3 or above
Memory	: 4GB of memory or above
Hard Disk /SSD	: 256GB or above

4.3 SOFTWARE REQUIREMENTS

A major element in building a system is the selection of compatible software. The selected software should be feasible to the system. The major software requirements that needed for developing the application are:

Operating system	: Windows 11
Database	: SQLite3
Front end	: Flutter
Back end	: Python Django
IDE	: Visual Studio

4.4 ABOUT SOFTWARE

4.4.1 Operating System

The fundamental goal of a computer system is to execute user programs and to make tasks easier. Various application programs along with hardware system are used to perform this work. An operating system is software which manages and controls the complete set of resources and effectively utilizes every part of a computer.

- **OS as a platform for application programs:** Operating system provides a platform, on top of which, other programs, called application programs can run These application programs help the users to perform a specific task easily. It acts as an interface between the computer and the user.
- **Managing input-Output unit:** Operating system also the computer to manage its own resources such as memory, monitor, keyboard, printer etc. Management of these resources are required for effective utilization.
- **Consistent user interface:** Operating system provides the user an easy-to-work user interface, so the user does not have to learn a different UI every time and can focus on the content and be productive as quickly as possible
- **Multitasking:** Operating system manages memory and allows multiple programs to run in their own space and even communicate with each other through shared memory.

4.4.2 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating system.

Features of Android Studio

- Android Studio's Apply Changes feature lets you push code and resource changes to your running app without restarting your app and, in some cases, without restarting the current activity.
- The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis.
- The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.
- Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager.
- Android Studio provides a robust static analysis framework and includes over 365 different lint checks across the entirety of your app.
- Android Studio offers build automation, dependency management. and customizable build configurations, You can configure your project to include local and hosted libraries, and define build variants that include different code and resources, and apply different code shrinking and app signing configurations.
- Android Studio provides a robust static analysis framework and Includes over 365 different lint chocks across the entirety of your app. Additionally, it provides several quick fixes that help address issues in various categories, such as performance, security, and correctness, with a single click.

.

4.4.3 Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

Features of VS Code

- Cross-platform: VS Code is available on Windows, macOS, and Linux, making it accessible to a wide range of developers.
- IntelliSense: Provides smart code completion, suggestions, and parameter info based on variable types, function definitions, and imported modules.
- Debugging: Built-in debugger with support for breakpoints, stepping through code, and variable inspection for various languages.
- Extensions: Extensible through a vast library of extensions for additional languages, themes, debuggers, and other functionalities.
- Version Control: Integrated Git support allows for seamless version control operations like committing, pulling, pushing, and diff viewing.
- Customization: Highly customizable with settings, themes, and key bindings to tailor the editor to individual preferences.
- Terminal Integration: Integrated terminal within the editor for executing commands, running scripts, and interacting with the system.
- Task Automation: Built-in task runner for automating repetitive tasks, such as building, testing, and deploying applications.
- Snippet Support: Supports code snippets for quick insertion of common code patterns, saving time and reducing typing effort.
- Language Support: Offers support for a wide range of programming languages through built-in features and extensions, including syntax highlighting, linting, and formatting.
- These features make VS Code a powerful and versatile tool for software development across different platforms and programming languages.

4.4.4 PYTHON Django

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

The Django web framework is a free, open-source framework that can speed up development of a web application being built in the Python programming language. Django—pronounced "Jango," named after the famous jazz guitarist Django Reinhardt—is a free, open-source framework that was first publicly released in 2005. Django facilitates "rapid development and clean, pragmatic design." The Django web framework, deployed on a web server, can help developers quickly produce a web frontend that's feature-rich, secure and scalable.

Features of python Django

- Django was intended to help developers make applications as fast as could be expected under the circumstances with less coding.
- Django deals with client validation, content organization, site maps, RSS channels, and many other tasks which become too easy with the use of this framework.
- Django pays attention to security and assists developers with keeping away from numerous normal security problems, like SQL injections, cross-site scripting, cross-site request forgery, and clickjacking. Its client verification framework gives safe gratitude to manage client records and passwords.
- Organizations, associations, and governments have utilized Django to make a wide range of things — from Content Administration to Interpersonal Organizations.

  Django gives an extension between the information model and the database motor and supports a huge arrangement of database frameworks including MySQL,Oracle,Postgres, and so forth.

4.4.5 Flutter

Flutter Tutorial provides basic and advanced concepts of the Flutter framework. Flutter is a UI toolkit for building fast, beautiful, natively compiled applications for mobile, web, and desktop with one programming language and single codebase. It is free and open-source. Initially, it was developed from Google and now manages by an ECMA standard. Flutter apps use Dart programming language for creating an app.

Features of Flutter

Flutter gives easy and simple methods to start building beautiful mobile and desktop apps with a rich set of material design and widgets. Here, we are going to discuss its main features for developing the mobile framework.

Open-Source: Flutter is a free and open-source framework for developing mobile applications.

Cross-platform: This feature allows Flutter to write the code once, maintain, and can run on different platforms. It saves the time, effort, and money of the developers.

Hot Reload: Whenever the developer makes changes in the code, then these changes can be seen instantaneously with Hot Reload. It means the changes immediately visible in the app itself. It is a very handy feature, which allows the developer to fix the bugs instantly.

Accessible Native Features and SDKs: This feature allows the app development process easy and delightful through Flutter's native code, third-party integration, and platform APIs. Thus, we can easily access the SDKs on both platforms.

Minimal code: Flutter app is developed by Dart programming language, which uses JIT and AOT compilation to improve the overall start-up time, functioning and accelerates the performance. JIT enhances the development system and refreshes the UI without putting extra effort into building a new one.

Widgets: The Flutter framework offers widgets, which are capable of developing customizable specific designs. Most importantly, Flutter has two sets of widgets: Material Design and Cupertino widgets that help to provide a glitch-free experience on all platforms.

4.4.6 SQLite3

A standalone command-line shell program called sqlite3 is provided in SQLite's distribution. It can be used to create a database, define tables, insert and change rows, run queries and manage an SQLite database file. It also serves as an example for writing applications that use the SQLite library.

SQLite IS an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly toordinary disk files. SQLite is a compact library. With all features enabled, the library size can be less than 600KiB, depending on the target platform and compiler optimization settings. (64- bit code is larger.

Features of SQLite3

Serverless: SQLite3 doesn't require a different server process or system to operate.

Flexibility: It facilitates you to work on multiple databases on the same session on the same time.

Cross-platform DBMS: You don't need a large range of different platforms like Windows, Mac OS, Linux, and Unix. It can also be used on a lot of embedded operating systems like Symbian, and Windows CE.

Provide large number of API's: SQLite provides API for a large range of programming languages. For example: .Net languages (Visual Basic, C#), PHP, Java, Objective C, Python and a lot of other programming language.

SYSTEM DESIGN

5.1 System Design

System design is the process of defining the elements of a system such as architectural, modules and components, the different interface of those components and the data that goes through that system.

It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well running system.

5.2 Input Design

Input facilitates the entry of data into the computer system. Input design involves the selection of the best strategy for getting data into the computer system at the right time and as accurately as possible. Effective input design minimizes the error made to data entry operators.

Our project "EV COMPANION" has got several inputs taken from the user. The input design of the Electric Vehicle Charging Station Finder and Slot Booking Application focuses on creating user-friendly interfaces for seamless interaction. It includes intuitive forms for user registration, search and filtering options, location selection, slot booking, user preferences, review submission, payment information, and error handling. The design prioritizes simplicity, clarity, and usability to enhance the overall user experience.

5.3 Output Design

Output from computer systems is required primarily to communicate the processing to the end users. The output of the system is designed in such a way that it provides opportunities, problems or warnings trigger an action and confirm an action.

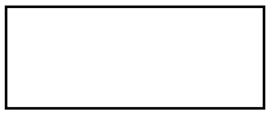
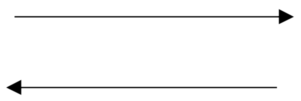
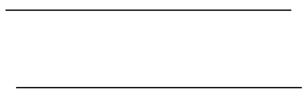
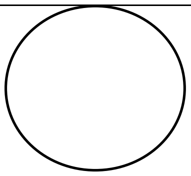
The user-friendly interfaces provide a clear output to the system. The output of this software system is an application.The output design of the Electric Vehicle Charging Station Finder and Slot Booking Application focuses on presenting information clearly and visually appealingly. It includes features like charging station listings, interactive maps, slot availability, booking confirmations, user profiles, review systems, and notifications. The design aims to prioritize clarity, usability, and accessibility for an enhanced user experience.

5.4 Database design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The general objectives are to make database access easy, quick, inexpensive and flexible for users. Normalization is due to get internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required and optimizing for updates.

5.5 Symbols of Dataflow Diagram

Name	Notation	Description
External entity		Represents the source or destination of data within a system.
Data flow		Represents the movement of data from its source to destination within a system.
Data source		Indicates the place for storing information within the system.
Process		Shows the transformation or manipulation of data within the system.

5.6 Data Flow Diagram

Dataflow diagram is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among those functions. The main merit of the DFD is that it can provide an overview of what data a system would process, what files are used, and where the result flow. Context level DFD was drawn first. Then the processes were decomposed into several elementary levels and were represented in the order of importance.

5.7 Table Structure

Table Name: Reg_Table (User Registration)

Sl no	Field name	Constrains	Data Type	Description
1.	Reg_id	Primary Key, Autoincrement	Text	Registration id of user
2.	Name	Not Null	Text	Name of user
3.	Email	Not Null	Text	Email of user
4.	Phone	Not null	Text	Phone number of user
5.	Password	Not null	Text	Password

Table Name: Charging Station

slno	Field Name	Constrains	Data Type	Description
1.	ID	Primary Key, Autoincrement	Text	user id
2.	Name	Not null	Text	Name of user
3.	Latitude	Not null	Integer	Latitude of charging station

4.	Longitude	Not null	Integer	Longitude of charging station
5.	Address	Not null	Text	Address of charging station
6.	Operating hours	Not null	Integer	Operating hours of charging station
7.	Photo	Not null	Long	Photo of charging station
8.	Contact info	Not null	Integer	Contact info of charging station

Table Name: Vehicle info

slno	Field Name	Constrains	Data Type	Description
1.	ID	Primary Key, Autoincrement	Text	user id
2.	Make	Not Null	Long	The company that made it
3.	Year	Not Null	Integer	The year that made it
4.	Battery capacity	Not Null	Text	The battery capacity of vehicle
5.	Charging time	Not null	Integer	Charging time of vehicle
6.	Vehicle image	Not null	Long	Image of vehicle

Table Name: Payment Integration

slno	Field Name	Constains	Data Type	Description
1.	Username	Not Null	Text	Username of user
2.	Payment id	Not null	Text	Payment id of user
3.	Order id	Not null	Text	Unique id given to the transaction
4.	Signature	Not null	Text	Signature of user
5.	Amount	Not null	Integer	Amount to pay
6.	Created at	Not null	Text	Payment gateway

SYSTEM TESTING

6.1 System Testing

Software testing is a critical element of software quality and represents the ultimate reviews of specification, design, and coding. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The project "**EV COMPANION**" undergoes system testing steps to carry it into desired output.

The testing steps are:

- Unit testing
- Integrated testing
- Validation testing
- Output testing
- Acceptance testing

Testing Result

All the test cases mentioned above passed successfully. No defects encountered.

6.2 Black Box Testing

The black box testing review for the Electric Vehicle Charging Station Finder and Slot Booking Application indicates satisfactory performance in functionality, usability, and reliability. The application successfully executes key features such as user registration, charging station search, slot booking, and review submission, offering users an intuitive and stable experience. Recommendations for enhancements include adding features like route optimization, exploring localization options, and conducting thorough security testing. Overall, the application shows promise in meeting user needs and expectations, with potential for further improvements.

6.3 White Box Testing

This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box testing. Internal software and code working should be known for this type of testing. The white box testing review for the Electric Vehicle Charging Station Finder and Slot Booking Application reveals a robust and well-structured codebase. Unit testing, integration testing, and adherence to coding best practices have contributed to the application's reliability, maintainability, and scalability. Recommendations for continuous improvement include implementing CI/CD pipelines, fostering a culture of code review, and conducting performance testing. Overall, the application demonstrates a high level of quality and readiness for deployment.

6.4 Unit Testing

The unit test evaluates the effectiveness of unit tests in verifying the functionality of individual components, functions, and modules within the application. Unit tests focus on isolating and testing specific units of code in isolation to ensure they perform as expected. For instance, a module such as the login is thoroughly tested by giving username and password.

6.5 Integration Testing

The integration test demonstrates effective evaluation of the interaction between different components, such as user authentication, payment processing, and database management. Through examples like the slot booking functionality test, the review ensures that these components work seamlessly together to achieve the desired functionality. This ensures the application's reliability and robustness in real-world scenarios.

6.6 Validation Testing

The validation test evaluates the application's ability to handle user input accurately and securely. Through examples like the user registration form test, the review ensures that the application appropriately validates input data to maintain integrity and prevent security vulnerabilities. This validation process enhances the overall reliability and security of the application.

6.7 Output Testing

The output test evaluates the accuracy and presentation of output data to users. Through comprehensive testing of features like charging station listings and booking confirmations, the review ensures that the application delivers relevant information in a clear, organized, and visually appealing manner. This ensures a positive user experience and fosters trust in the application's functionality and reliability.

6.8 User Acceptance Testing

Once the system tests have been satisfactorily completed, the system is ready for acceptance testing. The user acceptance test assesses its readiness for user adoption. By involving end-users in testing features such as searching for charging stations, booking slots, and submitting reviews, the review ensures that the application meets user expectations and effectively addresses their needs. This validation process is crucial for confirming that the application aligns with user requirements and delivers a satisfactory experience.

SYSTEM SECURITY

7.1 Security

As computing systems become more essential to our daily life, it becomes ever more important that the services they provide are available whenever we need them. We must also be able to rely on the integrity of systems, and thus the information they hold and provide. It protects information from unauthorized access.

The security of a computer system is a crucial task. It is a process of ensuring the confidentiality and integrity of the OS. Security is one of the most important as well as the major task to keep all the threats or other malicious tasks or attacks or program away from the computer's software system.

A system is said to be secure if its resources are used and accessed as intended under all circumstances, but no system can guarantee absolute security from several malicious threats and unauthorized access.

7.2 Backup

In information technology, a backup, or the process of backing up, refers to the copying and archiving of computer data so it may be used to restore the original after a data loss event. The verb form is to back up in two words, whereas the noun is backup. Backups have two distinct purposes

The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss can be a common experience for computer users. A 2008 survey found that 66% of respondents had lost files on their home PC.

The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.

Though backups represent a simple form of disaster recovery, and should be part of any disaster recovery plan, backups by themselves should not be considered a complete disaster recovery plan. One reason for this is that not all backup systems are able to reconstitute a computer system or other complex configuration such as a computer cluster. active directory server, or database server by simply restoring data from a backup. Since a backup system contains at least one copy of all data considered worth saving, the data storage requirements can be significant.

Organizing this storage space and managing the backup process can be a complicated undertaking. A data repository model may be used to provide structure to the storage. Nowadays, there are many distinct types of data storage devices that are useful for making backups. There are also many ways in which these devices can be arranged to provide geographic redundancy, data security, and portability.

7.3 Software Cost Estimation

Defining Cost Estimation

Cost estimation can be defined as the approximate judgment of the costs for a project. Cost estimation will never be an exact science because there are too many variables involved in the calculation for a cost estimate, such as human, technical, environmental, and political.

Furthermore, any process that involves a significant human factor can never be exact because humans are far too complex to be entirely predictable. Furthermore, software development for any fair-sized project will inevitably include a few tasks that have complexities that are difficult to judge because of the complexity of software systems.

Cost Estimation and Project Planning

Cost estimation is a crucial tool that can affect the planning and budgeting of a project. Because there are a finite number of resources for a project, all the features of requirements document can often not all are included in the final product. A cost estimate done at the beginning of a project will help determine which features can be included within the resource constraints of the project (e g., time). Requirements can be prioritized to ensure that the most notable features are included in the product.

The risk of a project is reduced when the most notable features are included at the beginning because the complexity of a project increases with its size, which means there is more opportunity for mistakes as development progresses. Thus, cost estimation can have a significant impact on the life cycle and schedule for a project.

SYSTEM IMPLEMENTATION

AND MAINTANANCE

8.1 System Implementation

Its design involves careful planning, investigation of the current system and its constraints on implementation, methods to achieve the changeover, an evaluation, of change over methods. Apart from planning, the major tasks of preparing the implementation are education and training of users. The more complex the system being implemented, the more involved will be the system analysis and the design effort required just for implementation.

For the successful implementation and cooperation of new systems the users must be selected, educated, and trained. Unless the users are not trained, the system will become complex, and it will feel like a burden for them. The selection of staff must take place at an early stage. The estimate of the numbers and the types of people required has been submitted to the management for approval. The people who will be affected are given a high priority in training and they are told at the earliest stage why the changes are necessary and how they will be affected. This communication was taken through their respective managers.

Implementation Plan

An implementation coordinating committee based on policies of individual organizations has been appointed. The implementation process begins with preparing a plan for implementation of the system. According to this plan activities are to be carried out, discussions made regarding the equipment and resources and additional equipment must be acquired to implement the new system.

Implementation is a final and important phase. The most critical stage in achieving a successful new system and in giving the users confidence that the system will work and be effective, the system can be implemented only after thorough testing is done and if it is found to be working according to the specification.

This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transactions while using the new system. At the beginning of the development phase a preliminary plan is created to schedule and manage the many different activities that must be integrated into the plan.

The implementation plan is updated throughout the development phase, culminating in a changeover for the operation phase. The major elements of the implementation plan are the test plan, equipment installation plan and a conversion plan. Implementing the sign language detecting application in deaf and mute communities, special schools, and within families involves tailored deployment strategies. Engage stakeholders across these settings to customize the app for their needs and provide comprehensive training and support.

Post Implementation Review

In the post-implementation phase of this application, the focus shifts to ongoing maintenance, user support, and continuous improvement. This involves collecting user feedback, monitoring application performance, addressing bugs, implementing new features, updating documentation, providing training and support, and promoting the application to attract and retain users. By staying responsive to user needs and market trends, the application can evolve and remain relevant in meeting user needs and achieving business goals.

8.2 System Maintenance

Maintenance means restoring something to its original condition. System maintenance conforms the system to its original requirements. It is an ongoing activity, which covers a wide variety of activities, including program and design errors, updating documentation and test data, and updating user support. For convenience, maintenance may be categorized into three classes, namely:

i. **Corrective Maintenance:** Enable user to carry out the repairing and correcting leftover problems.

ii. **Adaptive Maintenance:** Enable user to replace the functions of the programs.

iii. **Perfective Maintenance:** Enable users to modify or enhance the programs according to user's requirements and changing needs.

Ev Companion can improve user experience by station access, improving mapping accuracy, and exploring innovative solutions like simplified booking systems can significantly enhance the platform's utility. Integrating emerging technologies and fostering community engagement can further propel its impact, driving sustainable mobility and electric vehicle adoption in India. Expanding the EV Companion project to include a trip planning system based on the availability of electric vehicles on the road would be a significant advancement. This feature could provide users with valuable insights into routes with sufficient charging infrastructure, helping them plan their journeys efficiently.

CONCLUSION

9.1 Conclusion

In conclusion, the "**EV COMPANION**" Application represents an innovative solution to address the growing demand for convenient and efficient electric vehicle charging services. By leveraging technologies such as Flutter for the frontend, Python Django for the backend, and SQLite3 for the database, the application offers users a seamless experience in locating charging stations, booking slots, and managing their charging needs.

Throughout the development process, careful attention was paid to user feedback, industry trends, and best practices to ensure that the application meets the needs of its target audience. Rigorous testing, including unit tests, integration tests, and user acceptance tests, verified the reliability, functionality, and usability of the application.

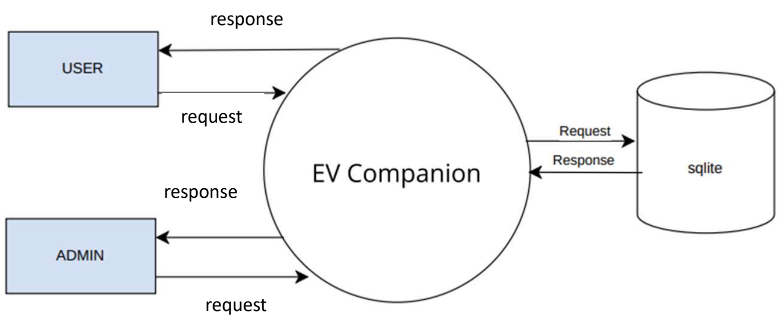
As the application moves into the post-implementation phase, the focus will remain on continuous improvement, user support, and promotion. By staying responsive to user feedback, addressing bugs, implementing new features, and maintaining a high level of performance and security, the application aims to remain a valuable tool for electric vehicle owners and charging station operators alike.

Overall, the Electric Vehicle Charging Station Finder and Slot Booking Application represents a significant step forward in promoting the adoption of electric vehicles and facilitating a cleaner, more sustainable transportation ecosystem. With its user-centric design and robust functionality, the application is poised to make a meaningful impact in the transition to electric mobility.

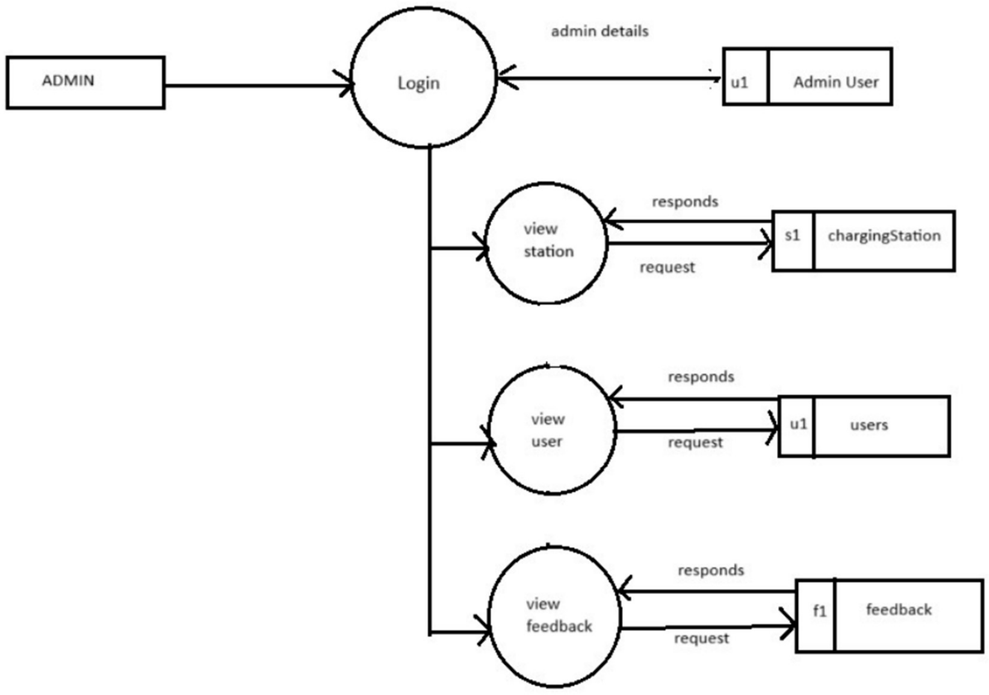
APPENDICES

10.1 Data Flow Diagram

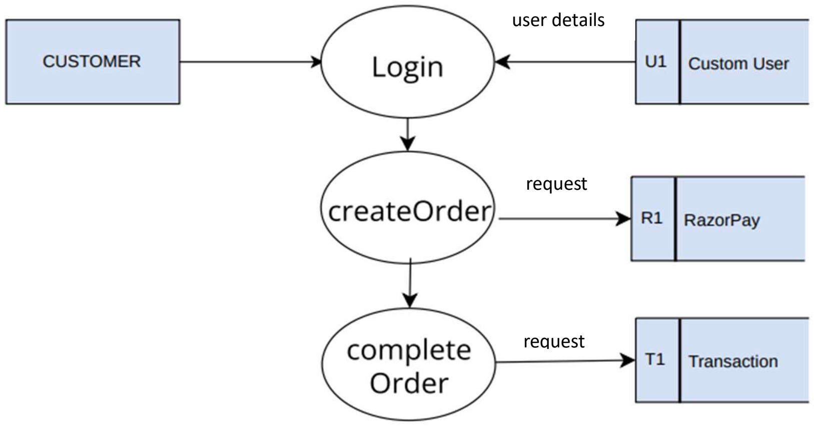
LEVEL 0



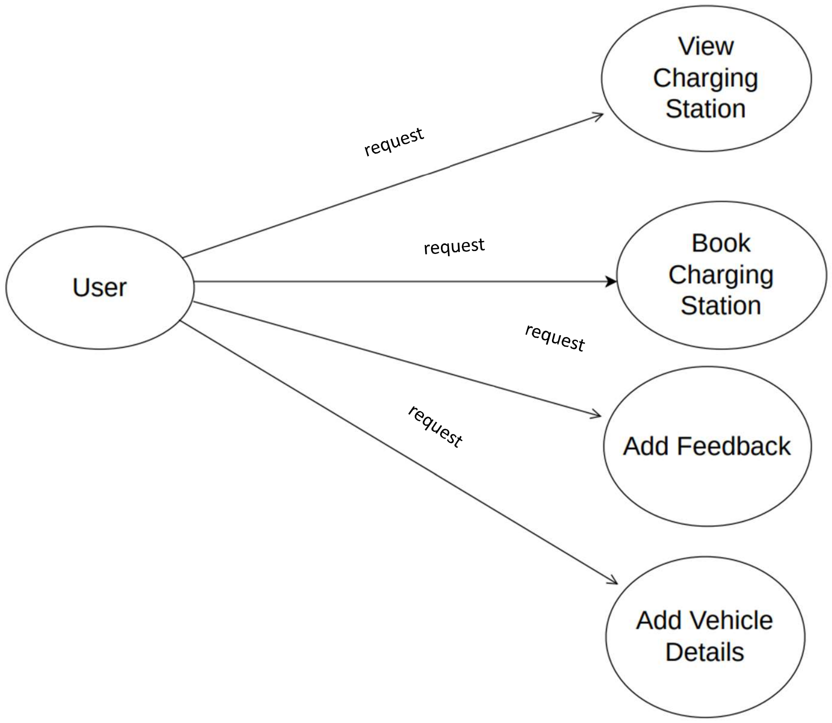
LEVEL 1



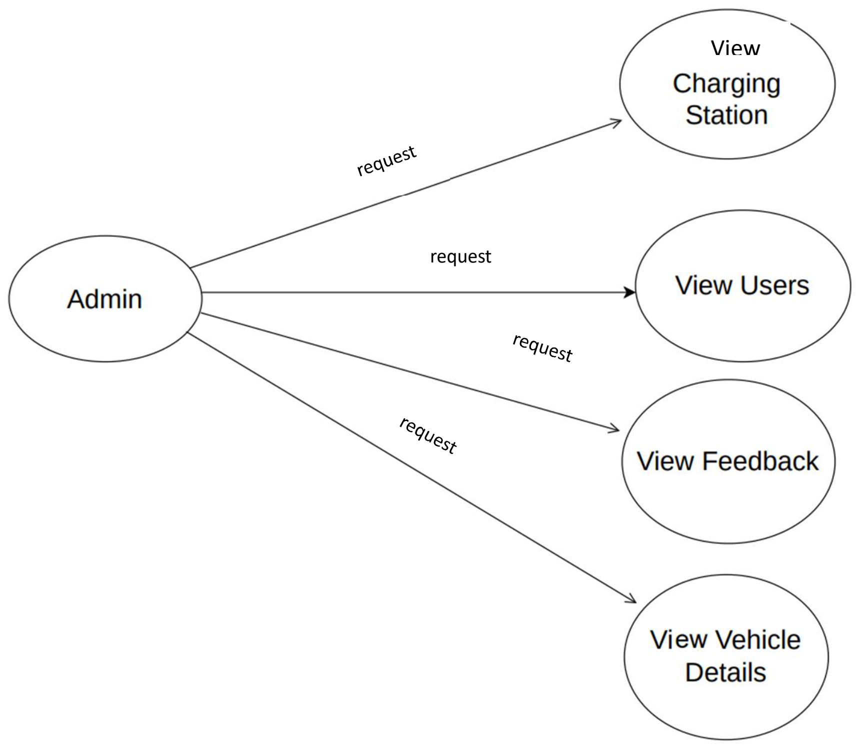
LEVEL 2



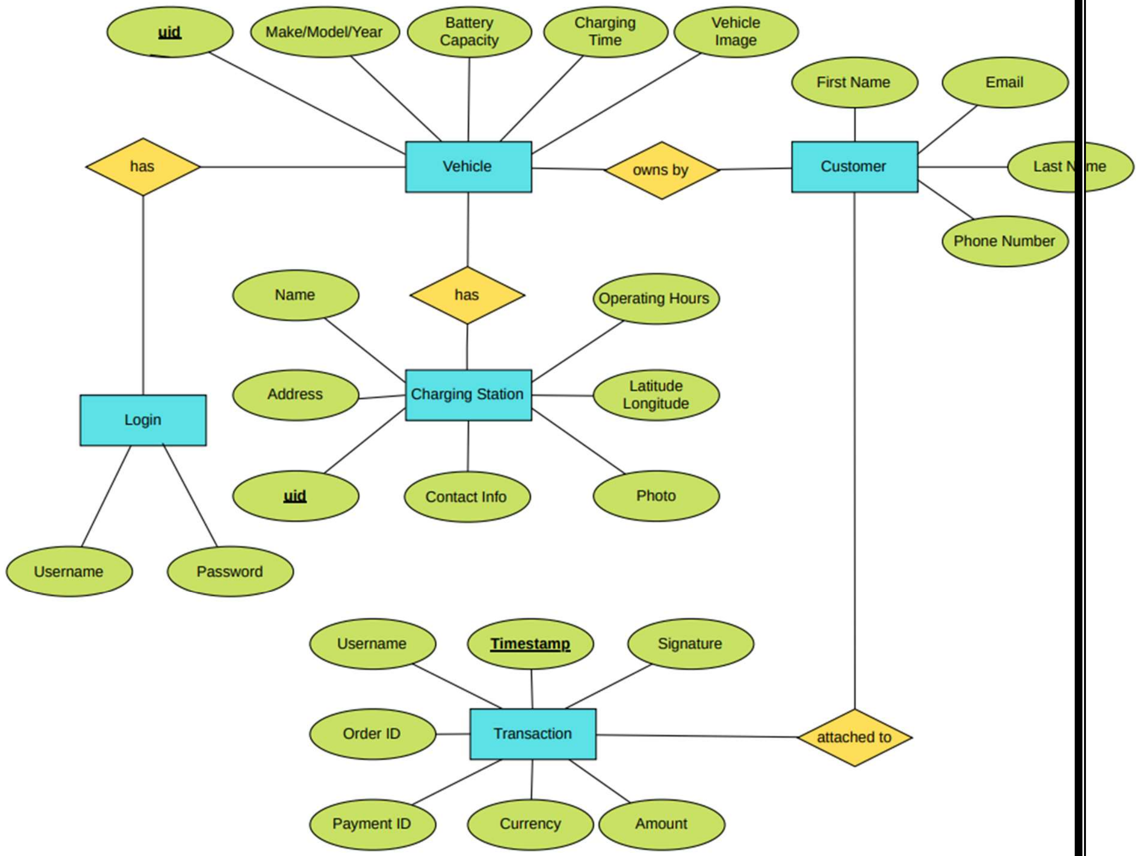
LEVEL 3



LEVEL 4

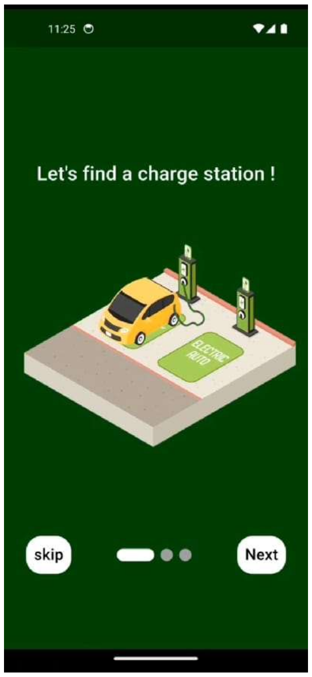

View
Charging
Station
request
request
Admin
View Users
request
request
View Feedback
View Vehicle
Details

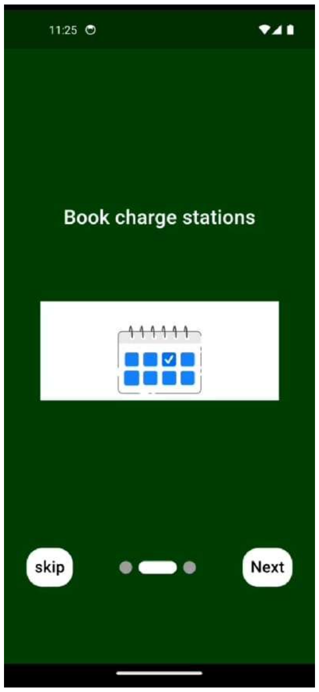
10.2 ER DIAGRAM

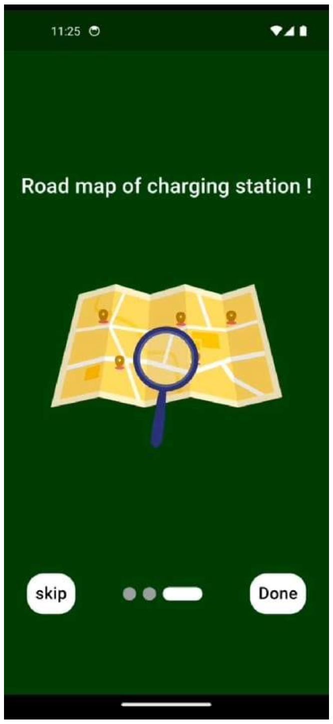


10.3 Screenshots

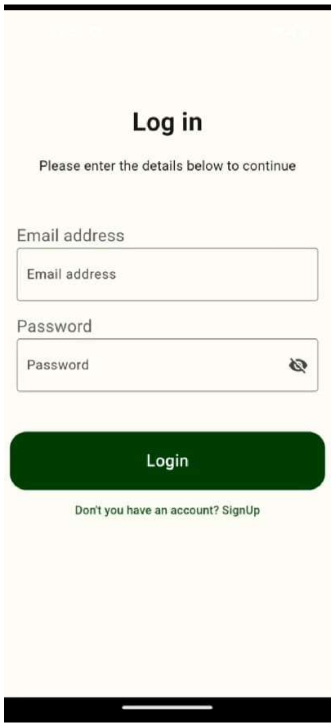
Splash Screen



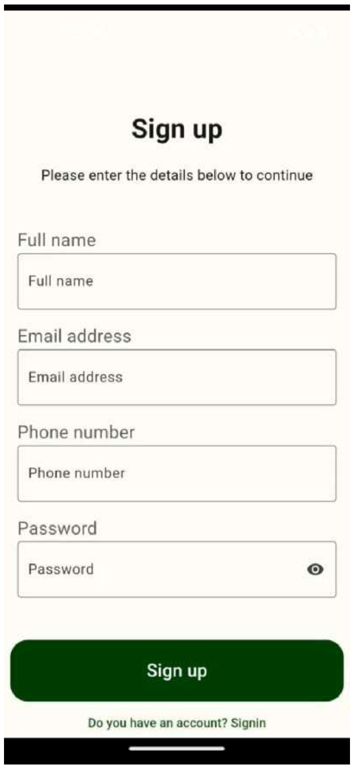



11:25
Road map of charging station !
skip
Done

Login Page


Log in
Please enter the details below to continue
Email address
Email address
Password
Password
Login
Don't you have an account? SignUp

Sign-Up Page


Sign up

Please enter the details below to continue

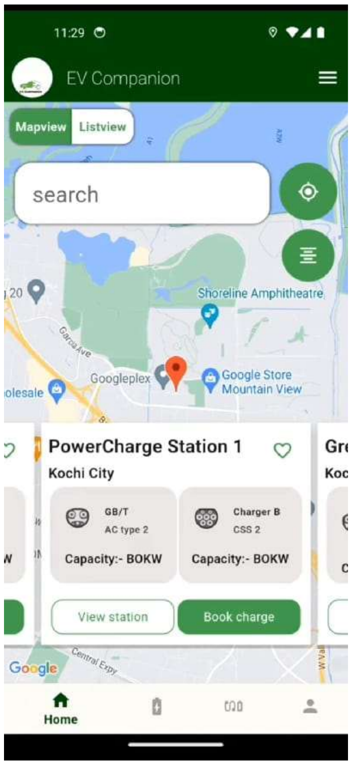
Full name: Full name

Email address: Email address

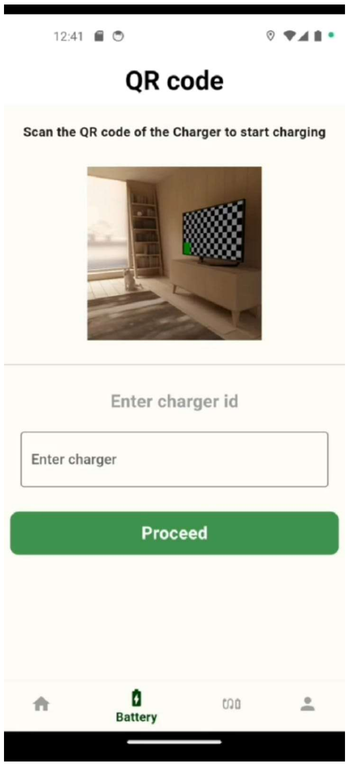
Phone number: Phone number

Password: Password

Home Page



Scan Page

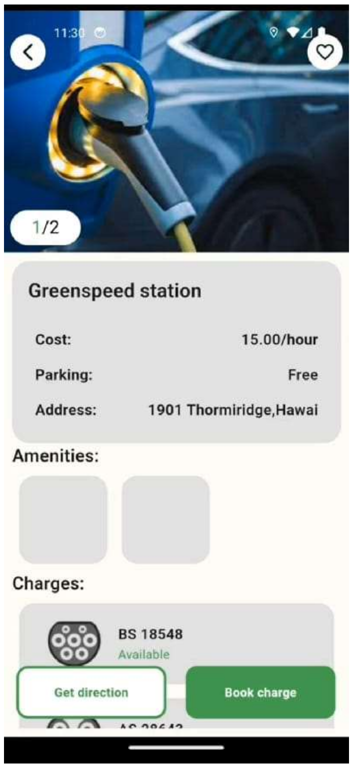

12:41

QR code

Scan the QR code of the Charger to start charging

Enter charger id

Charging Station Page


11:30

Greenspeed station

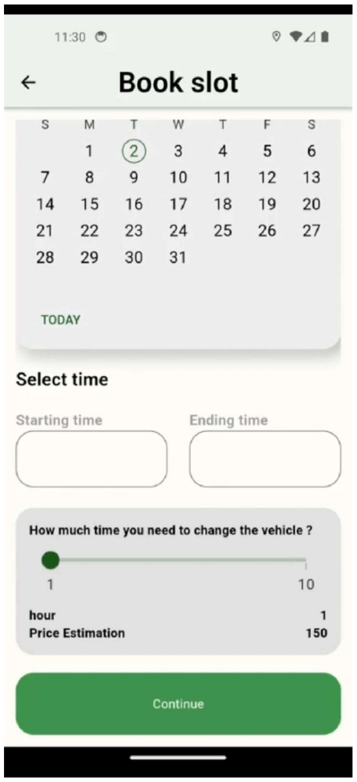
Cost:	15.00/hour
Parking:	Free
Address:	1901 Thormiridge,Hawai

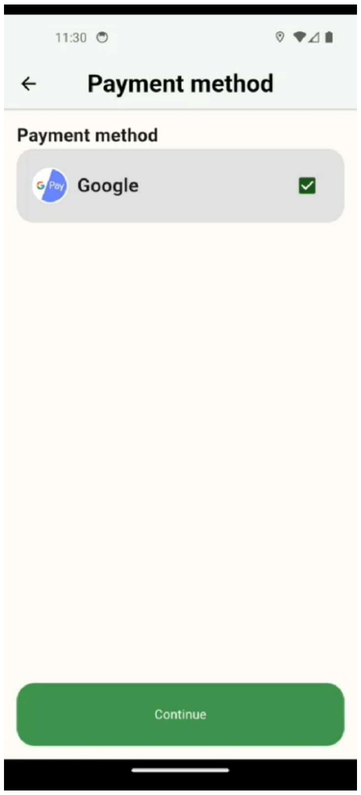
Amenities:

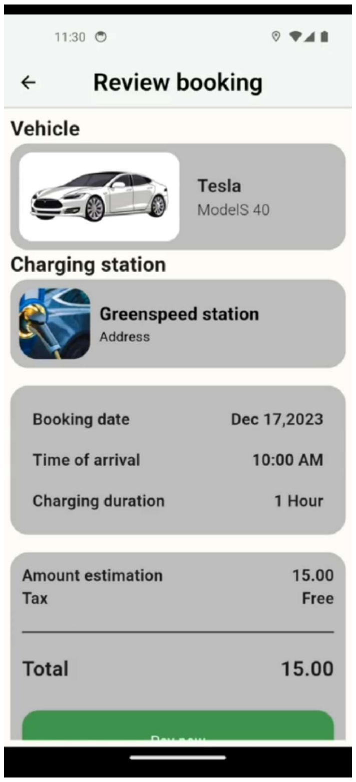
Charges:

BS 18548
Available

Booking Slot







11:30

Review booking

Vehicle

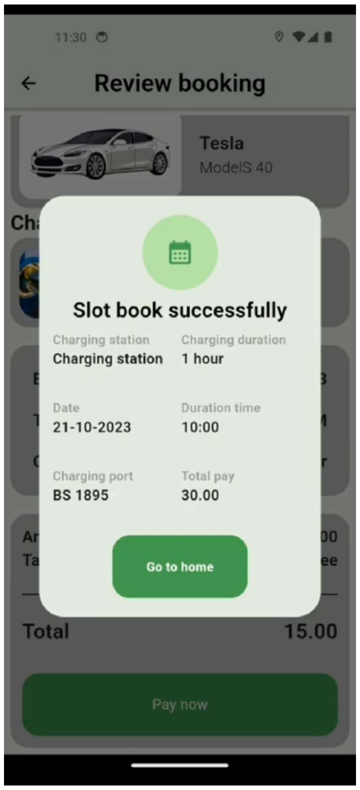
Tesla
ModelS 40

Charging station

Greenspeed station
Address

Booking date	Dec 17,2023
Time of arrival	10:00 AM
Charging duration	1 Hour

Amount estimation	15.00
Tax	Free
Total	15.00



11:30

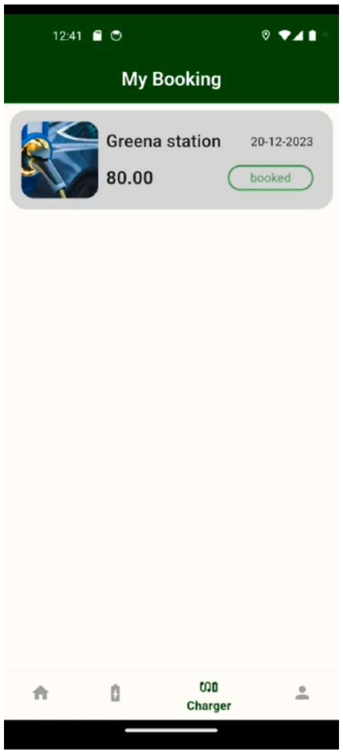
Review booking

Tesla
ModelS 40

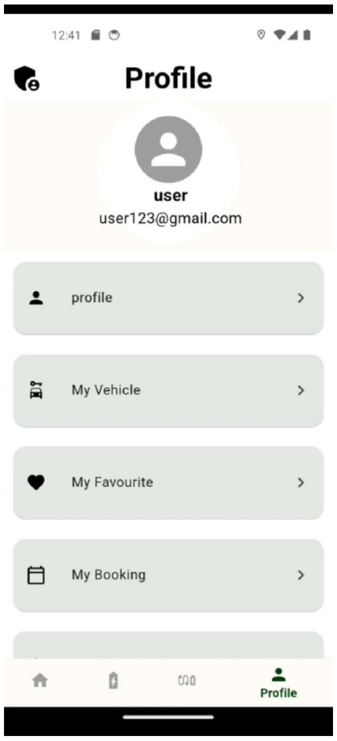
Slot book successfully

Charging station	Charging duration
Charging station	1 hour
Date	Duration time
21-10-2023	10:00
Charging port	Total pay
BS 1895	30.00

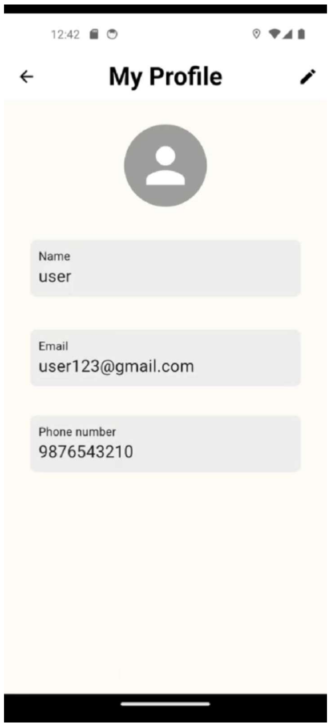
My Booking



Profile Page



My Profile

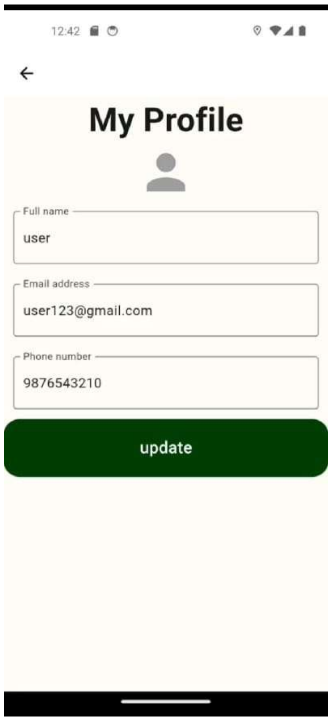


Name
user

Email
user123@gmail.com

Phone number
9876543210

Update Profile



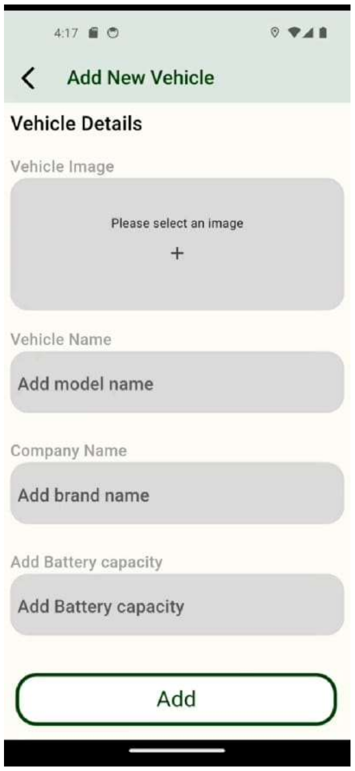
My Profile

Full name
user

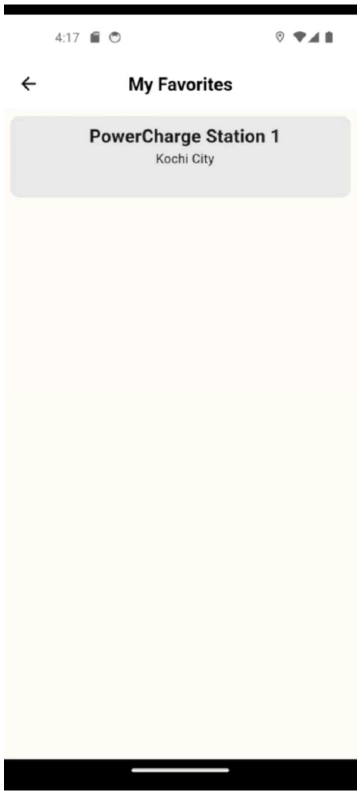
Email address
user123@gmail.com

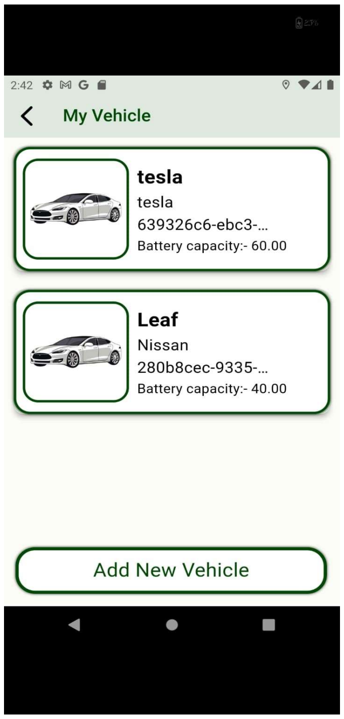
Phone number
9876543210

Add Vehicle

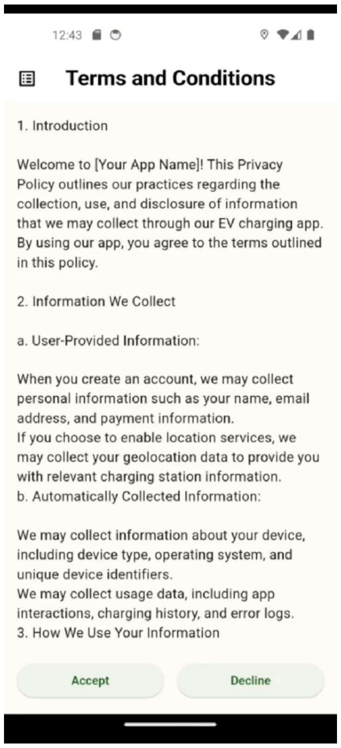


My Favourite





Terms&Condition


12:43

Terms and Conditions

1. Introduction

Welcome to [Your App Name]! This Privacy Policy outlines our practices regarding the collection, use, and disclosure of information that we may collect through our EV charging app. By using our app, you agree to the terms outlined in this policy.

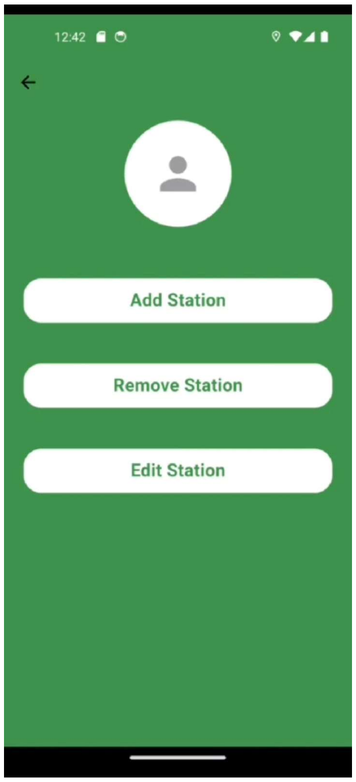
2. Information We Collect

a. User-Provided Information:

When you create an account, we may collect personal information such as your name, email address, and payment information.
If you choose to enable location services, we may collect your geolocation data to provide you with relevant charging station information.
b. Automatically Collected Information:

We may collect information about your device, including device type, operating system, and unique device identifiers.
We may collect usage data, including app interactions, charging history, and error logs.
3. How We Use Your Information

Admin Page



10.4 Code

Frontend Code of Application

Splash Screen

```
import 'dart:async';
import 'package:evcompanion2/presentation/view/bottom_nav_pages/bottom_nav_controller.dart';
import 'package:evcompanion2/presentation/view/feature_page/feature_page.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import '../utils/colorConstants.dart';
class SplashScreen extends StatefulWidget {
 const SplashScreen({super.key});
 @override
 _SplashScreenState createState() => _SplashScreenState();
}
class _SplashScreenState extends State<SplashScreen> {
 @override
 void initState() {
  Future.delayed(const Duration(seconds: 3), () {
   _checkData();
  });
  super.initState();
 }
 void _checkData() async {
  SharedPreferences sharedpreference = await SharedPreferences.getInstance();
  String storedusername = sharedpreference.getString('unamekey') ?? '';
  String storedpassword = sharedpreference.getString('passkey') ?? '';
  if (storedpassword.isNotEmpty && storedusername.isNotEmpty) {
   Navigator.pushReplacement(context, MaterialPageRoute(builder: (cnt) {
    return const BottomNavController();
   }));
   } else {
     Navigator.pushReplacement(context, MaterialPageRoute(builder: (cnt) {
       return const FeaturePage();
     }));
   }
  }
  @override
  Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: myappColor,
     body: SafeArea(
       child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
           const CircleAvatar(
               radius: 70, backgroundImage: AssetImage("assets/evcomp.jpg")
               //backgroundImage:AssetImage("assets/evcomp.jpg")
               ),
           SizedBox(
             height: 20.h,
           ),
           const CircularProgressIndicator(
             color: Colors.white,
           ),
          ],
        ),
       ),
     ),
   );
  }
}
```

Home Page

```
import 'dart:async';
import 'dart:convert';
import 'package:evcompanion2/presentation/view/homepage/filter.dart';
import 'package:evcompanion2/controller/ev_stations_service/ev_station_services.dart';
import 'package:evcompanion2/presentation/view/homepage/widgets/ev_list_view.dart';
import 'package:evcompanion2/presentation/view/homepage/widgets/ev_map_view.dart';
import 'package:evcompanion2/utils/colorConstants.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:location/location.dart';
import 'package:flutter_google_places/flutter_google_places.dart' as loc;
import 'package:google_maps_webservice/places.dart' as places;
import 'package:http/http.dart' as http;
import 'package:provider/provider.dart';
class Homepage extends StatefulWidget {
  const Homepage({super.key});
  @override
  State<Homepage> createState() => _HomepageState();
}
class _HomepageState extends State<Homepage> {
  TextEditingController searchController = TextEditingController();
  Location locationController = Location();
  static const LatLng _pApplepark = LatLng(37.346, -122.0090);
  static const LatLng _destination = LatLng(10.011104, 76.343877);
  final Completer<GoogleMapController> mapController = Completer();
  BitmapDescriptor customMarkerIcon = BitmapDescriptor.defaultMarker;
  bool viewCategoryCheck = true;
  List viewCategoryPages = [
    MapView(
      key: UniqueKey(),
    ),
    const EvListView(),
  ];
  int pageViewCategoryIndex = 0;
  void addCustomMarker() async {
    BitmapDescriptor.fromAssetImage(
            const ImageConfiguration(
              size: Size(13, 13),
            ),
            'assets/ev_location_marker.png')
        .then((icon) {
      setState(() {
        customMarkerIcon = icon;
      });
    });
  }
  LatLng? _currentP = null;
  @override
  void initState() {
    super.initState();
    addCustomMarker();
    getLocationupdaets();
  }
  @override
  Widget build(BuildContext context) {
    var evLocationController = Provider.of<EvStationsServices>(context);
    return Scaffold(
      appBar: AppBar(
        backgroundColor: myappColor,
        leading: const Padding(
          padding: EdgeInsets.only(
            left: 10,
          ),
          child: CircleAvatar(
            backgroundImage: AssetImage('assets/evcomp.jpg'),
      ),
     ),
     title: const Text('EV Companion'),
     titleTextStyle: const TextStyle(
      color: Colors.white,
      fontSize: 20,
      fontWeight: FontWeight.w200,
     ),
     actions: [
      IconButton(
       onPressed: () {},
       icon: const Icon(
        Icons.menu,
        color: Colors.white,
        size: 27,
       ),
      )
     ],
    ),
    body: _currentP == null
      ? const Center(
        child: Text('Loading...'),
       )
      : Stack(
        children: [
         viewCategoryCheck
           ? GoogleMap(
             key: widget.key,
             onMapCreated: (GoogleMapController mapcontroller) {
              mapController.complete(mapcontroller);
             },
             zoomControlsEnabled: false,
             initialCameraPosition:
              CameraPosition(target: _currentP!, zoom: 14),
          markers: {
            Marker(
                markerId: const MarkerId('_curr 47 on'),
                icon: BitmapDescriptor.defaultMarker,
                position: _currentP!),
            //evlocation markers
            for (var evLocation
                in evLocationController.stationData)
              Marker(
                markerId: MarkerId(evLocation["stationName"]),
                position: LatLng(evLocation["latitude"],
                    evLocation["longitude"]),
                infoWindow: InfoWindow(
                  title: evLocation["stationName"],
                  snippet: evLocation["location"],
                ),
              ),
          },
        )
      : Container(),
  //view category
  Column(
    children: [
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: SizedBox(
          height: 40,
          child: Row(
            children: [
              InkWell(
                onTap: () {
                  setState(() {
```

```
      viewCategoryCheck = !viewCategoryCheck;
      pageViewCategoryIndex = 0;
    });
  },
  child: Container(
    width: 70,
    height: 40,
    decoration: BoxDecoration(
      boxShadow: const [
        BoxShadow(
          color: Colors.black,
          blurStyle: BlurStyle.outer,
          blurRadius: 2,
        )
      ],
      borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(13),
        bottomLeft: Radius.circular(13),
      ),
      color: viewCategoryCheck
          ? Colors.green
          : Colors.white,
    ),
    child: Center(
      child: Text(
        'Mapview',
        style: TextStyle(
          color: viewCategoryCheck
              ? Colors.white
              : Colors.green,
          fontWeight: FontWeight.w600,
        ),
      ),
    ),
  ),
),
InkWell(
  onTap: () {
    setState(() {
      viewCategoryCheck = !viewCategoryCheck;
      pageViewCategoryIndex = 1;
    });
  },
  child: Container(
    width: 70,
    height: 40,
    decoration: BoxDecoration(
        boxShadow: const [
          BoxShadow(
            color: Colors.black,
            blurStyle: BlurStyle.outer,
            blurRadius: 2,
          )
        ],
        color: viewCategoryCheck
            ? Colors.white
            : Colors.green,
        borderRadius: const BorderRadius.only(
          bottomRight: Radius.circular(13),
          topRight: Radius.circular(13),
        )),
    child: Center(
      child: Text(
        'Listview',
        style: TextStyle(
          color: viewCategoryCheck
```

```
                      ? Colors.green
                      : Colors.white,
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
//search bar
viewCategoryCheck
    ? Padding(
        padding: const EdgeInsets.all(13),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Expanded(
              child: Padding(
                padding: const EdgeInsets.only(right: 10),
                child: Container(
                  // width: 290,
                  height: 70,
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(20),
                      color: Colors.white,
                      boxShadow: const [
                        BoxShadow(
                          blurRadius: 2,
                          color: Colors.black,
                        ),
                        BoxShadow(
                          color: Colors.black,
                        ),
                        BoxShadow(
                          color: Colors.black,
                        ),
                      ]),
                  child: Center(
                    child: Padding(
                      padding: const EdgeInsets.only(
                        left: 20,
                      ),
                      child: TextField(
                        controller: searchController,
                        onChanged: (value) {
                          if (value.isNotEmpty) {
                            searchPlaces();
                          }
                        },
                        style: const TextStyle(
                          fontSize: 25,
                          color: myappColor,
                          fontWeight: FontWeight.w400,
                        ),
                        decoration: const InputDecoration(
                          hintText: 'search',
                          isDense: false,
                          border: InputBorder.none,
                        ),
                      ),
                    ),
                  ),
                ),
              ),
            ),
            InkWell(
              onTap: () {
                if (_currentP != null &&
                    mapController.isCompleted) {
                  mapController.future.then((controller) {
                    controller.animateCamera(
                      CameraUpdate.newCameraPosition(
                        CameraPosition(
                          target: _currentP!,
                          zoom: 14,
                        ),
                      ),
                    );
                  });
                }
              },
              child: Column(
                children: [
                  const CircleAvatar(
                    radius: 33,
                    backgroundColor: Colors.green,
                    child: Center(
                      child: Icon(
                        Icons.gps_fixed_outlined,
                        color: Colors.white,
                      ),
                    ),
                  ),
                  SizedBox(height: 10,),
                  CircleAvatar(
                        radius: 30,
                        backgroundColor: Colors.green,
                        child: Center(
                          child: IconButton(onPressed: (){
                            Navigator.of(context).push(
                              MaterialPageRoute(builder:(context)=>Filter()));
                          },
                           icon: Icon(Icons.format_align_center_outlined,
                           color: Colors.white,)),
                        ),
                      )
                    ],
                  ),

                ),
              ],
            ),
          )
        : Container(),
      ],
    ),
    //list of pages in home
    viewCategoryPages[pageViewCategoryIndex],
  ],
),
);
}
Future<void> getLocationupdaets() async {
  bool serviceEnabled;
  PermissionStatus _permissionGranted;
  serviceEnabled = await locationController.serviceEnabled();
  if (serviceEnabled) {
    serviceEnabled = await locationController.requestService();
  } else {
    return;
  }
  _permissionGranted = await locationController.hasPermission();
  if (_permissionGranted == PermissionStatus.denied) {
    _permissionGranted = await locationController.requestPermission();
    if (_permissionGranted == PermissionStatus.granted) {
      return;
    }
  }
  locationController.onLocationChanged.listen((LocationData currentLocation) {
    if (currentLocation.latitude != null &&
        currentLocation.longitude != null) {
      setState(() {
        _currentP =
            LatLng(currentLocation.latitude!, currentLocation.longitude!);
        print(_currentP);
      });
    }
  });
}
void searchPlaces() async {
  places.Prediction? prediction = await loc.PlacesAutocomplete.show(
    context: context,
    apiKey: 'AIzaSyBkuIUpT3yTbQ8by32XKYuS2ggdORXhxNo',
    offset: 0,
    radius: 10000,
    strictbounds: false,
    mode: loc.Mode.overlay,
    language: "en",
    components: [places.Component(places.Component.country, "US")],
  );
```

```
    if (prediction != null) {
      final response = await http.get(
        Uri.parse(
'https://maps.googleapis.com/maps/api/place/details/json?place_id=${prediction.placeId}&key=AI
zaSyBkuIUpT3yTbQ8by32XKYuS2ggdORXhxNo'),
      );
      if (response.statusCode == 200) {
        final Map<String, dynamic> data = json.decode(response.body);
        if (data['status'] == 'OK') {
          final result = data['result'];
          double lat = result['geometry']['location']['lat'];
          double lng = result['geometry']['location']['lng'];
          LatLng selectedLocation = LatLng(lat, lng);
          mapController.future.then((controller) {
            controller.animateCamera(
              CameraUpdate.newCameraPosition(
                CameraPosition(
                  target: selectedLocation,
                  zoom: 14,
                ),
              ),
            );
          });
          setState(() {
            _currentP = selectedLocation;
          });
        }
      }
    }
  }
}
```

View Booking

```
import 'package:flutter/material.dart';
import 'package:evcompanion2/controller/bookstation_controller.dart';
import 'package:provider/provider.dart';
import 'package:evcompanion2/model/my_booking_model/my_booking_model.dart';
class ViewBookingPage extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  var provider = Provider.of<StationbookController>(context);
  return Scaffold(
   appBar: AppBar(
    title: Text('My Bookings'),
   ),
   body: ListView.builder(
    itemCount: provider.bookingList.length,
    itemBuilder: (context, index) {
     MyBookingModel booking = provider.bookingList[index];
     return
     Container(
        padding: const EdgeInsets.all(16.0),
   decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12.0),
    boxShadow: [
     BoxShadow(
      color: Colors.grey.shade300,
      blurRadius: 8.0,
      offset: const Offset(0, 2),
     ),
    ],
   ),
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
     Row(
      children: [
    ClipRRect(
      borderRadius: BorderRadius.circular(8.0),
      child: Image.asset(
        booking.image,
        width: 80,
        height: 80,
        fit: BoxFit.cover,
      ),
    ),
    const SizedBox(height: 12.0,width: 20,),
    Text(
      booking.user,
      style: TextStyle(
        fontSize: 20.0,
        fontWeight: FontWeight.bold,
      ),
    ),
  ],
),
const SizedBox(height: 8.0),
Text(
  booking.name,
  style: TextStyle(
    fontSize: 20.0,
    fontWeight: FontWeight.bold,
    color: Colors.black,
  ),
),
const SizedBox(height: 8.0),
Text(
  ('${booking.date} - ${booking.buttontext}'),
  style: TextStyle(
    fontSize: 14.0,
          color: Colors.grey,
         ),
        ),
        Text(
         booking.startingtime,
         style: TextStyle(
          fontSize: 14.0,
          color: Colors.grey,
         ),
        ),
        const SizedBox(height: 8.0),
        Text(
         booking.price,
         style: TextStyle(
          fontSize: 16.0,
          fontWeight: FontWeight.bold,
          color: Colors.green,
         ),
        ),
       ],
      ),
        );
       },
      ),
    );
  }
}
```

My profile

```
import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:evcompanion2/presentation/view/settings_page/edit_profile.dart';
```

```
class myProfile extends StatefulWidget {
 @override
 State<myProfile> createState() => _myProfileState();
}
class _myProfileState extends State<myProfile> {
 late SharedPreferences Preferences;
 String? tname;
 String? tuname;
 String? tphone;
 @override
 void initState() {
  super.initState();
  fetchData();
 }
 void fetchData() async {
  Preferences = await SharedPreferences.getInstance();
  setState(() {
   tname = Preferences.getString('namekey') ?? "";
   tuname = Preferences.getString('unamekey') ?? "";
   tphone = Preferences.getString('phonekey') ?? "";
  });
 }
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    elevation: 0,
    shadowColor: Colors.black,
    title: Text(
      "My Profile",
      style: TextStyle(fontWeight: FontWeight.bold, fontSize: 30),
    ),
    centerTitle: true,
    actions: [
      IconButton(
        onPressed: () {
          Navigator.of(context).push(
            MaterialPageRoute(builder: (context) => EditProfile()),
          );
        },
        icon: Icon(Icons.edit),
      ),
    ],
   ),
   body: Container(
    child: SingleChildScrollView(
      child: Container(
        height: MediaQuery.of(context).size.height,
        width: MediaQuery.of(context).size.width,
        child: Column(
          children: [
            SizedBox(height: 30,),
            CircleAvatar(
              backgroundColor: Colors.grey,
              radius: 50.0,
              child: Icon(
                Icons.person,
                size: 70,
                color: Colors.white,
              ),
            ),
            SizedBox(height: 20,),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Padding(
                  padding: const EdgeInsets.all(20.0),
                  child: Container(
                    width: MediaQuery.of(context).size.width / 1.2,
                    padding: EdgeInsets.all(10.0),
                    decoration: BoxDecoration(
                      color: Colors.grey[200],
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text("Name"),
                        Text("$tname", style: TextStyle(fontSize: 20)),
                      ],
                    ),
                  ),
                ),
              ],
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Padding(
                  padding: const EdgeInsets.all(20.0),
                  child: Container(
                    width: MediaQuery.of(context).size.width / 1.2,
                    padding: EdgeInsets.all(10.0),
                    decoration: BoxDecoration(
                      color: Colors.grey[200],
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text("Email"),
                        Text("$tuname", style: TextStyle(fontSize: 20)),
                      ],
                    ),
                  ),
                ),
              ],
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Padding(
                  padding: const EdgeInsets.all(15.0),
                  child: Container(
                    width: MediaQuery.of(context).size.width / 1.2,
                    padding: EdgeInsets.all(10.0),
                    decoration: BoxDecoration(
                      color: Colors.grey[200],
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text("Phone number"),
                        Text(
                          "$tphone",
                          style: TextStyle(fontSize: 20),
                        ),
                      ],
                    ),
                  ),
                ),
              ],
            ),
          ],
            ),
          ),
        ),
      ),
    );
   }
}
```

My Favourite

```
import 'package:evcompanion2/controller/favorite_controller/f_controller.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
class MyFavourites extends StatefulWidget {
  const MyFavourites({super.key});
  @override
  State<MyFavourites> createState() => _MyFavouritesState();
}
class _MyFavouritesState extends State<MyFavourites> {
  FavoriteController favoriteController = FavoriteController();
  @override
  Widget build(BuildContext context) {
    var favContro = Provider.of<FavoriteController>(context);
    return Scaffold(
      appBar: AppBar(
        title: Text(
          "My Favorites",
          style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
        ),
        centerTitle: true,
      ),
      body: Column(
        children: List.generate(favContro.favoriteList.length, (index) {
          return Padding(
            padding: const EdgeInsets.all(8.0),
            child: Container(
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(10),
                color: Color.fromARGB(255, 231, 231, 231),
              ),
              width: double.infinity,
              height: 100,
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  children: [
                    SizedBox(width: 10),
                    Expanded(
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Row(
                            mainAxisAlignment: MainAxisAlignment.spaceBetween,
                            children: [
                              Text(
                                favContro.favoriteList[index].stationName,
              style: TextStyle(
               fontSize: 20,
               fontWeight: FontWeight.bold,
              ),
             ),
             IconButton(
              onPressed: () {
               setState(() {
                favContro.favoriteList.removeAt(index);
               });
              },
              icon: Icon(Icons.delete),
             ),
            ],
           ),
           Text(favContro.favoriteList[index].location),
           SizedBox(
            height: 10,
           ),
           Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
           ),
          ],
         ),
        ),
       ],
      ),
                  ),
                ),
              );
            }),
          ),
        );
       }
      }
```

Admin page

```
import 'package:evcompanion2/presentation/view/login_screen/login_screen.dart';
import 'package:evcompanion2/presentation/view/settings_page/feedisplay.dart';
import 'package:evcompanion2/registration_page/reg_details.dart';
import 'package:flutter/material.dart';
import 'Admin_view_station.dart';
class AdminHome extends StatelessWidget {
 const AdminHome({super.key});
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   // appBar: AppBar(
   //   title: Text(
   //     "Admin Home",
   //     style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold, color: Colors.black),
   //   ),
   //   backgroundColor: Colors.transparent,
   //   centerTitle: true,
   // ),
body: Container(
  decoration: BoxDecoration(
    image: DecorationImage(
      image:
          AssetImage('assets/andreas-rasmussen-MBW3F1jEhh0-unsplash.jpg'),
      fit: BoxFit.cover,
    ),
  ),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.start,
    children: [
      Container(
        color: Colors.transparent,
        child: Padding(
          padding: const EdgeInsets.all(10),
          child: Row(
            children: [
              IconButton(onPressed: (){
                Navigator.pop(context);
              }, icon:Icon(Icons.arrow_back)),
              // Text(
              //   "Admin Home",
              //   style: TextStyle(
              //       fontSize: 30,
              //       fontWeight: FontWeight.bold,
              //       color: Colors.black),
              // ),
            ],
          ),
        ),
      ),
      Expanded(
        child: Container(
          color: Colors.transparent,
          child: ListView(
            padding: EdgeInsets.all(16),
            children: [
              _buildContainer("View Users", () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => ProfilePage()));
              }),
              _buildContainer("View Stations", () {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                      builder: (context) => StationsPage()),
                );
              }),
              _buildContainer("View feedback", () {
                Navigator.push(context, MaterialPageRoute(
                 builder: (context)=>DisplayFeedbackPage()));
              }),
                _buildContainer("Logout", () {
                  Navigator.push(context, MaterialPageRoute(builder: (context)=>LoginFire()));
                }),
              ],
            ),
          ),
        ),
      ],
    ),
  ),
);
}
Widget _buildContainer(String text, VoidCallback onTap) {
  return GestureDetector(
    onTap: onTap,
    child: Container(
      padding: EdgeInsets.all(16),
      margin: EdgeInsets.only(bottom: 16),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(12),
      ),
      child: Center(
        child: Text(
          text,
          style: TextStyle(
              fontSize: 20, fontWeight: FontWeight.bold, color: Colors.black),
          ),
        ),
      ),
    );
  }
}
```

BACKEND OF APPLICATION

Charging Station

```
from rest_framework.views import APIView

from rest_framework.response import Response

from .serializers import ChargingStationSerializer

from rest_framework import status

from rest_framework.permissions import IsAuthenticated

from rest_framework_simplejwt.authentication import JWTAuthentication

from django.db.models import Q

from django.core.paginator import Paginator

from UserAccounts.models import CustomUser

from ChargingStations.models import ChargingStation

# Create your views here.

class StationsAddView(APIView):

    # permission_classes=[IsAuthenticated]

    # authentication_classes=[JWTAuthentication]

    def get(self, request):

        try:

            vehicles=ChargingStation.objects.all().order_by('?')

            page_number=request.GET.get('page',1)

            paginator=Paginator(vehicles,2)

           serializer=ChargingStationSerializer(paginator.page(page_number),many=True)

            return Response({

                'data':serializer.data,

                'message':'Charging Staion details fetched successfully'

            }, status=status.HTTP_201_CREATED)
    except Exception as e:
        print(e)
        return Response({
                'data':{},
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
def post(self, request):
    try:
        current_user = CustomUser.objects.get(id=request.user.id)
        print(current_user)
        if not current_user.is_superuser:
            return Response({
                'message': 'You are not authorized to perform this action'
            }, status=status.HTTP_403_FORBIDDEN)
        data=request.data
        data['user']=request.user.id
        print(request.user)
        serializer=ChargingStationSerializer(data=data)
        if not serializer.is_valid():
            return Response({
                'data':serializer.errors,
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
        serializer.save()
        return Response({
                'data':serializer.data,
            'message':'Station added successfully'
          }, status=status.HTTP_201_CREATED)
      except Exception as e:
        print(e)
        return Response({
            'data':{},
            'message':'something went wrong'
          }, status=status.HTTP_400_BAD_REQUEST)
    def patch(self,request):
      try:
        data=request.data
        station=ChargingStation.objects.filter(uid=data.get('uid'))
        current_user = CustomUser.objects.get(id=request.user.id)
        if not station.exists():
          return Response({
            'data':{},
            'message':'Not a valid station id'
          }, status=status.HTTP_400_BAD_REQUEST)
        if not current_user.is_superuser:
          return Response({
            'data':{},
            'message':'You are not authorized'
          }, status=status.HTTP_400_BAD_REQUEST)
        serializer=ChargingStationSerializer(station[0],data=data,partial=True)
        if not serializer.is_valid():
          return Response({
            'data':{},
                    'message':'Something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
            serializer.save()
            return Response({
                    'data':serializer.data,
                    'message':'Successfully updated'
                }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                    'data':{},
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
    def delete(self,request):
        try:
            data=request.data
            station=ChargingStation.objects.filter(uid=data.get('uid'))
            current_user = CustomUser.objects.get(id=request.user.id)
            if not station.exists():
                return Response({
                    'data':{},
                    'message':'Not a valid station id'
                }, status=status.HTTP_400_BAD_REQUEST)
            if not current_user.is_superuser:
                return Response({
                    'data':{},
                    'message':'You are not authorized'
            }, status=status.HTTP_400_BAD_REQUEST)
          station[0].delete()
          return Response({
                'data':{},
                'message':'Successfully deleted'
            }, status=status.HTTP_201_CREATED)
        except Exception as e:
          print(e)
          return Response({
                'data':{},
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
```

Payment Integration

```
from rest_framework.views import APIView
from rest_framework import status
from  PaymentIntegration.serializers import RazorpayOrderSerializer, TransactionModelSerializer
from PaymentIntegration.main import RazorpayClient
from rest_framework.response import Response
rz_client = RazorpayClient()
class RazorpayOrderAPIView(APIView):
    """This API will create an order"""
    def post(self, request):
        razorpay_order_serializer = RazorpayOrderSerializer(
            data=request.data
        )
        print(razorpay_order_serializer)
        if razorpay_order_serializer.is_valid():
            order_response = rz_client.create_order(
                amount=razorpay_order_serializer.validated_data.get("amount"),
                currency=razorpay_order_serializer.validated_data.get("currency")
            )
            response = {
                "status_code": status.HTTP_201_CREATED,
                "message": "order created",
                "data": order_response
            }
            return Response(response, status=status.HTTP_201_CREATED)
        else:
            response = {
                "status_code": status.HTTP_400_BAD_REQUEST,
                "message": "bad request",
                "error": razorpay_order_serializer.errors
            }
            return Response(response, status=status.HTTP_400_BAD_REQUEST)
class TransactionAPIView(APIView):
    """This API will complete order and save the
    transaction"""
    def post(self, request):
        transaction_serializer = TransactionModelSerializer(data=request.data)
        if transaction_serializer.is_valid():
            rz_client.verify_payment_signature(
                razorpay_payment_id = transaction_serializer.validated_data.get("payment_id"),
                razorpay_order_id = transaction_serializer.validated_data.get("order_id"),
                razorpay_signature = transaction_serializer.validated_data.get("signature")
            )
            transaction_serializer.save()
            response = {
                "status_code": status.HTTP_201_CREATED,
                "message": "transaction created"
            }
            return Response(response, status=status.HTTP_201_CREATED)
        else:
            response = {
                "status_code": status.HTTP_400_BAD_REQUEST,
                "message": "bad request",
                "error": transaction_serializer.errors
            }
            return Response(response, status=status.HTTP_400_BAD_REQUEST)
```

User Accounts

```
from rest_framework.views import APIView
from rest_framework.response import Response
from .serializers import RegisterSerializer, LoginSerializer,EditSerializer
from rest_framework import status
from .models import CustomUser
from rest_framework.permissions import IsAuthenticated
from rest_framework_simplejwt.authentication import JWTAuthentication
# Create your views here.
class RegisterView(APIView):
    def post(self,request):
        try:
            data=request.data
            serializer=RegisterSerializer(data=data)
            if not serializer.is_valid():
                return Response({
                    'data':serializer.errors,
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
            serializer.save()
            return Response({
                    'data':{},
                    'message':'Account created'
                }, status=status.HTTP_201_CREATED)
        except Exception as e:
            return Response({
                    'data':serializer.errors,
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
class LoginView(APIView):
    def post(self,request):
        try:
            data=request.data
            serializer=LoginSerializer(data=data)
            if not serializer.is_valid():
                return Response({
                    'data':serializer.errors,
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
            response=serializer.get_jwt_token(serializer.data)
            return Response(response, status=status.HTTP_200_OK)
        except Exception as e:
            return Response({
                    'data':serializer.errors,
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
class EditProfile(APIView):
    # permission_classes=[IsAuthenticated]
    # authentication_classes=[JWTAuthentication]
    def get(self, request):
        try:
            user_details = CustomUser.objects.get(id=request.user.id)
            serializer=EditSerializer(user_details)
            return Response({
                    'data':serializer.data,
                    'message':'User details fetched successfully'
                }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                    'data':{},
                    'message':'something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
    def patch(self,request):
        try:
            data=request.data
            current_user = CustomUser.objects.get(id=request.user.id)
            print(current_user)
            if not current_user:
                return Response({
                    'data':{},
                    'message':'Not a valid user id'
                }, status=status.HTTP_400_BAD_REQUEST)
            if request.user != current_user:
                return Response({
                    'data':{},
                    'message':'You are not authorized'
                }, status=status.HTTP_400_BAD_REQUEST)
            serializer = EditSerializer(current_user, data=data, partial=True)
            if not serializer.is_valid():
                return Response({
                    'data':{},
                    'message':'Something went wrong'
                }, status=status.HTTP_400_BAD_REQUEST)
            serializer.save()
            return Response({
                    'data':serializer.data,
                    'message':'Successfully updated'
                }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                    'data':{},
                    'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
```

Vehicles Info

```
from rest_framework.views import APIView
from rest_framework.response import Response
from .serializers import VehicleSerializer
from rest_framework import status
from rest_framework.permissions import IsAuthenticated
from rest_framework_simplejwt.authentication import JWTAuthentication
from django.db.models import Q
from django.core.paginator import Paginator
from UserAccounts.models import CustomUser
from VehicleInfo.models import Vehicles
# Create your views here.
class VehiclesAddView(APIView):
    # permission_classes=[IsAuthenticated]
    # authentication_classes=[JWTAuthentication]
    def get(self, request):
        try:
            vehicles=Vehicles.objects.all().order_by('?')
            page_number=request.GET.get('page',1)
            paginator=Paginator(vehicles,2)
            serializer=VehicleSerializer(paginator.page(page_number),many=True)
            return Response({
                'data':serializer.data,
                'message':'Vehicle details fetched successfully'
            }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                'data':{},
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
    def post(self, request):
        try:
            # current_user = CustomUser.objects.get(id=request.user.id)
            # print(current_user)
            # if not current_user.is_superuser:
            #     return Response({
            #         'message': 'You are not authorized to perform this action'
            #     }, status=status.HTTP_403_FORBIDDEN)
            data=request.data
            # data['user']=request.user.id
            # print(request.user)
            serializer=VehicleSerializer(data=data)
            if not serializer.is_valid():
            return Response({
                'data':serializer.errors,
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
            serializer.save()
            return Response({
                'data':serializer.data,
                'message':'vehicle added successfully'
            }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                'data':{},
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
    def patch(self,request):
        try:
            data=request.data
            vehicle=Vehicles.objects.filter(uid=data.get('uid'))
            current_user = CustomUser.objects.get(id=request.user.id)
            if not vehicle.exists():
                return Response({
                    'data':{},
                    'message':'Not a valid vehicle id'
                }, status=status.HTTP_400_BAD_REQUEST)
            if not current_user.is_superuser:
                return Response({
                    'data':{},
                    'message':'You are not authorized'
                }, status=status.HTTP_400_BAD_REQUEST)
            serializer=VehicleSerializer(vehicle[0],data=data,partial=True)
            if not serializer.is_valid():
                return Response({
                    'data':{},
                    'message':'Something went wrong'
              }, status=status.HTTP_400_BAD_REQUEST)
            serializer.save()
            return Response({
                  'data':serializer.data,
                  'message':'Successfully updated'
              }, status=status.HTTP_201_CREATED)
        except Exception as e:
            print(e)
            return Response({
                  'data':{},
                  'message':'something went wrong'
              }, status=status.HTTP_400_BAD_REQUEST)
    def delete(self,request):
        try:
            data=request.data
            vehicle=Vehicles.objects.filter(uid=data.get('uid'))
            current_user = CustomUser.objects.get(id=request.user.id)
            if not vehicle.exists():
                return Response({
                  'data':{},
                  'message':'Not a valid vehicle id'
              }, status=status.HTTP_400_BAD_REQUEST)
            if not current_user.is_superuser:
                return Response({
                  'data':{},
                  'message':'You are not authorized'
              }, status=status.HTTP_400_BAD_REQUEST)
```

```
        vehicle[0].delete()
        return Response({
                'data':{},
                'message':'Successfully deleted'
            }, status=status.HTTP_201_CREATED)
    except Exception as e:
        print(e)
        return Response({
                'data':{},
                'message':'something went wrong'
            }, status=status.HTTP_400_BAD_REQUEST)
```

BIBLIOGRAPHY

11.1 Bibliography

Websites:

https://ijarsct.co.in

http://www.geeksforgeeks.org

www.wekipedia.com

http://www.tutorialspoint.com

Books:

”Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2" by Alessandro Biessek

“Django Unleashed" by Andrew Pinkham.