SMART PARKING MANAGEMENT SYSTEM

Software Requirement Engineering

Final Term Project

Section D

Group 10

YASER, SAMIN	19-39442-1
RAFI, SADMAN	20-42708-1
BASAK, DIPON	19-39463-1

American International University - Bangladesh

LaTeX

Table of Contents

1. INTRODUCTION

1.1. Purpose

Due to the large automobile ownership rate in the United States, parking has become a source of contention and confusion for a number of individuals. The significant contrast between the constantly increasing number of cars and the restricted number of parking spaces in Bangladesh leads to the phenomena of ”difficult parking and disorderly parking,” which has severe effects on the quality of life of inhabitants and the operation of urban highways. Parking is a significant issue in metropolitan centers of both developed and developing nations. As a result of the fast expansion in automobile ownership, many cities suffer from a scarcity of parking spaces and an imbalance between parking supply and demand, which may be regarded the root cause of metropolitan parking issues. Parking issues in cities and metropolitan regions have been one of the most often debated concerns among both the general public and experts. The disparity between parking supply and parking demand has been identified as the primary cause of urban parking issues. In addition, the parking system plays a crucial function in the urban traffic system, and its absence is closely linked to traffic congestion, traffic accidents, and environmental pollution. Although an effective parking system may enhance urban mobility and the city environment in addition to enhancing the quality of life for inhabitants, the parking problem is an area of urban planning and transportation that is sometimes disregarded. Whether at an airport, transit station, or retail centre, parking issues are a daily occurrence. Lack of accessible parking may be detrimental to local companies and reduce inhabitants’ quality of life.

The proposed solution to the parking issue in Bangladesh is the development of an application that would aid cars in discovering the closest available parking spot based on their current location. Because there are so many buildings, malls, apartment complexes, and parking lots with parking spaces accessible for a limited time, the majority of car and bike owners do not always keep their vehicles in their garages. They must drive to the office, school, and other locations, thus their parking space in the garage is unoccupied the majority of the time. At that time, visitors who choose to park their vehicles in this location will be able to do so for an hourly fee, which will help alleviate the parking problem. The program will facilitate communication between the two users. In this manner, the parking space owner earns money by allowing other users to utilize it, while the other user avoids the hassle of searching for parking. This application will benefit both parties. The major objective here is to resolve the parking issue while simultaneously giving the garage owner a financial advantage. In addition, searching for a parking spot might waste a substantial amount of time and gasoline. If they utilize this application, they will also save time and petrol. Currently, there is no smart parking app accessible in Bangladesh. However, this sort of system is accessible in other nations. This project seeks to utilize those system characteristics and add a few new features based on Bangladesh's surroundings. This idea will alleviate the parking problem, reduce fuel usage, and give a financial benefit to users who rent out their parking space.

1.2. Intended Audience

Developers: It should ensure the developers are working on the correct features that meet the standards outlined in this document.

Testers: It must have a comprehensive list of the features and functionalities that must react in accordance with the specified requirements and illustrations.

Users: To familiarize themselves with the concept of the project and recommend further functional features.

1.3. Product Scope

Numerous individuals own automobiles. Nowadays, A vehicle is becoming a must. Every location is undergoing urbanization. There are several business offices, commercial complexes, hospitals, schools, and government agencies, among others. Therefore, these locations require parking spaces where individuals may park their automobiles simply and safely. Every parking lot must include a system that logs the information of cars using the facility. These systems may be automated or non-automated. With the aid of a computerized system, this project can provide excellent service to customers who wish to park on the grounds of any firm. A vehicle parking management system is an autonomous system that processes data at a very rapid rate and in a systematic fashion. The development of this technology is extremely beneficial to this profession.

2. REQUIREMENT DEVELOPMENT

2.1. Feasibility Study

2.1.1. Economic Feasibility

Economic feasibility aims to measure the expense of designing and implementing a new system against the advantages that would result from its implementation. This feasibility study provides the executive leadership with the financial basis for the new system. In this situation, a straightforward economic analysis that provides a direct comparison of costs and benefits is significantly more relevant. Additionally, this serves as a valuable reference point for comparing real expenses as the project advances.

This project is economically feasible since the funds that were previously spent on devices that were bought for developing the application will now be used to maintain and update the application and they do not need to be replaced annually. Since, for deploying and hosting mobile applications, it is preferable to use ‘Back-end as a Service’ solutions like ‘FireBase’, the company does not need to invest in servers. Therefore, a one-time investment in computers minimizes the company’s expenditures. In addition, a project is viable since all of the required software and its documentation are readily available on the internet, the company does not need to spend money on training the workforce.

2.1.2. Operational Feasibility

This project is operationally viable in the sense that it can be accomplished using existing Back-end solutions like ‘Firebase’ or serverless services like ‘AWS Lambda’. Thus, data are now more secure, and the danger of data loss is reduced. For deployment, using Docker and Kubernetes will lower the likelihood of errors in automatic updates. In addition, maintaining the program does not need a large number of computer experts, since the tools mentioned before automating various processes conveniently.

2.1.3. Technical Feasibility

Technical feasibility focuses on the current computer system’s (hardware, software, etc.) ability to support the planned addition. For instance, if the existing computer is functioning at 80 percent capacity - an arbitrary ceiling - then running another program

may cause the system to become overloaded or necessitate the acquisition of extra hardware. This necessitates budgetary considerations to allow technological upgrades. If the budget is a significant limitation, the project is deemed unfeasible.
The technical viability of the proposed system relates to the system's technology. It examines if the system's hardware and software are of the most recent technological generation. Using the Linux platform, React Native Framework makes the proposed system technically practical and sound. Because these technologies have been tried and tested by being on the market for a long time.

2.2. Elicitation

This is the first actual step in Requirement development. This section houses the following activities:

- Knowledge of the overall application domain for the parking systems.
- Understanding the specifics of the client problem to which the solution will be applied is necessary.
- System (the proposed parking application) interaction with external interfaces.
- In-depth analysis of user requirements.
- Define the system development restrictions.

2.2.1. Interviews

Information is gathered through interviews with users and an examination of existing materials. For the creation of the Parking management system, extensive study and feedback from a wide range of website and application users were required. Consequently, the following surveys were supplied to them, necessitating our application.

Questionnaires:

- What problems do you encounter with the current system?
- What all-new elements would you like to see added to the proposed system?
- What method do you use to store your information?
- Who are all of the system's users?

2.2.2. Brainstorming

It is a group discussion where it is meant to produce a large number of fresh ideas, therefore creating a forum for discussion. A facilitator, in this case, the Business Analyst with the help of the marketing team will manage group prejudice and group disputes. Extensive training is essential for this process to work. Finally, a rough document is composed of the list of needs and, if possible, their priority. This will come as a great help later when building **Software Requirement Specification**.

2.2.3. Analysis

Following elicitation, requirement analysis is an important and necessary process. Analyzing, refining, and examining the acquired requirements allows us to create unified and unambiguous requirements. This task examines all requirements and may produce a graphical representation of the complete parking system. After the completion of the analysis, it is possible that the project's clarity will increase substantially. In this case, we also leverage the engagement with the client to clarify areas of ambiguity and determine which needs are most crucial. In this section, we do the following:

- **Model the Application:** This procedure often involves graphical representations of the functions, data entities, external entities, and their relationships. The graphical approach can assist in identifying inaccurate, inconsistent, missing, and unnecessary criteria. Such models include the Data Flow diagram, the Entity-Relationship diagram, the Data Dictionary, and the State-transition diagram, among others. The diagrams are given in the Specification section of this chapter.
- **Finalise the Requirements:** After modelling the requirements, our knowledge of the system's behaviour will be enhanced. Identified and rectified errors and ambiguities. The data flow between various components has been examined. The operations of elicitation and analysis have yielded a deeper understanding of the proposed parking system. Now that the criteria have been finalized, the following step is to record them in a specified way or in **Software Requirement Specification** (SRS) format.

2.3. Specification

This section contains the SRS document. As its name implies, **Software Requirement Specification** or SRS Format is a comprehensive specification and description of software requirements that must be met for the effective development of a software system. Depending on the type of demand, these requirements may be functional or non-functional. The relationship between different consumers and the contractor is vital to completely comprehending the clients' demands.

2.3.1. Overall Description

2.3.1.1. Product Perspective

The product is made with the perspective that it will replace regular parking system, although it can be first be applied to small areas for testing and after being successful it might be implemented at a larger scale.

2.3.1.2. User Classes And Characteristics

Driver/User The customer will login and can search parking place, request for the parking, book parking place and check charges. Can pay the charges with online payment. Also, user or customer can update profile. Provider The provider can login in the site, manage parking space, manage parking fee, assign parking cars, accept all request, assign space. In the profile the system can manage payment, update profile, logout.

2.3.1.3. System Diagrams

The Figure 1 is about activity diagram. In this activity diagram we can see the activity of the smart parking app. where customer have to log in to the parking app then check login details and permission. After that customer can go to check permission of manage car, manage parking, manage parking slot, manage parking space manage parking fees. After that customer can log out from the app.

In figure 2 customer can search parking, allot parking and check charge and they can also login and logout. They can update their profile also check payment status and then book parking space. Here in the diagram system can manage parking space,

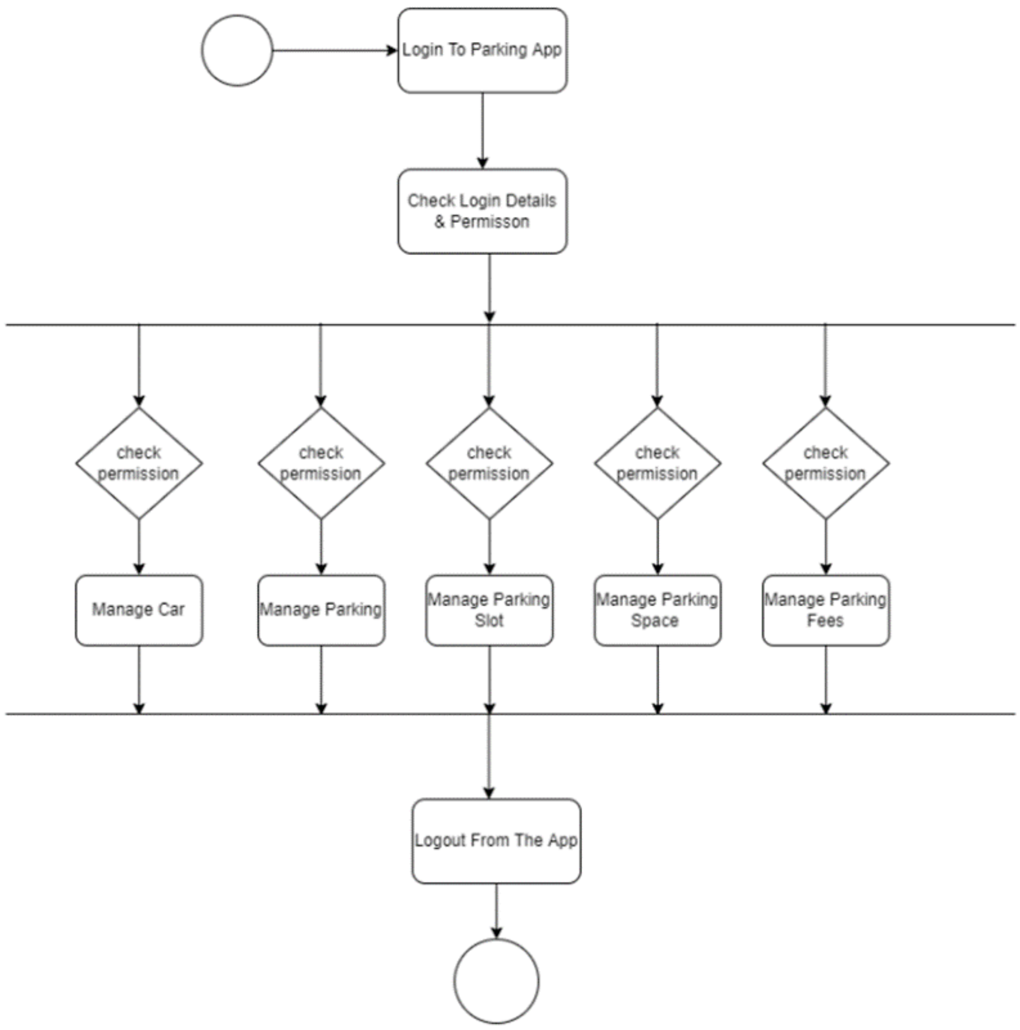


Figure 1 – Activity Diagram of Smart Parking System

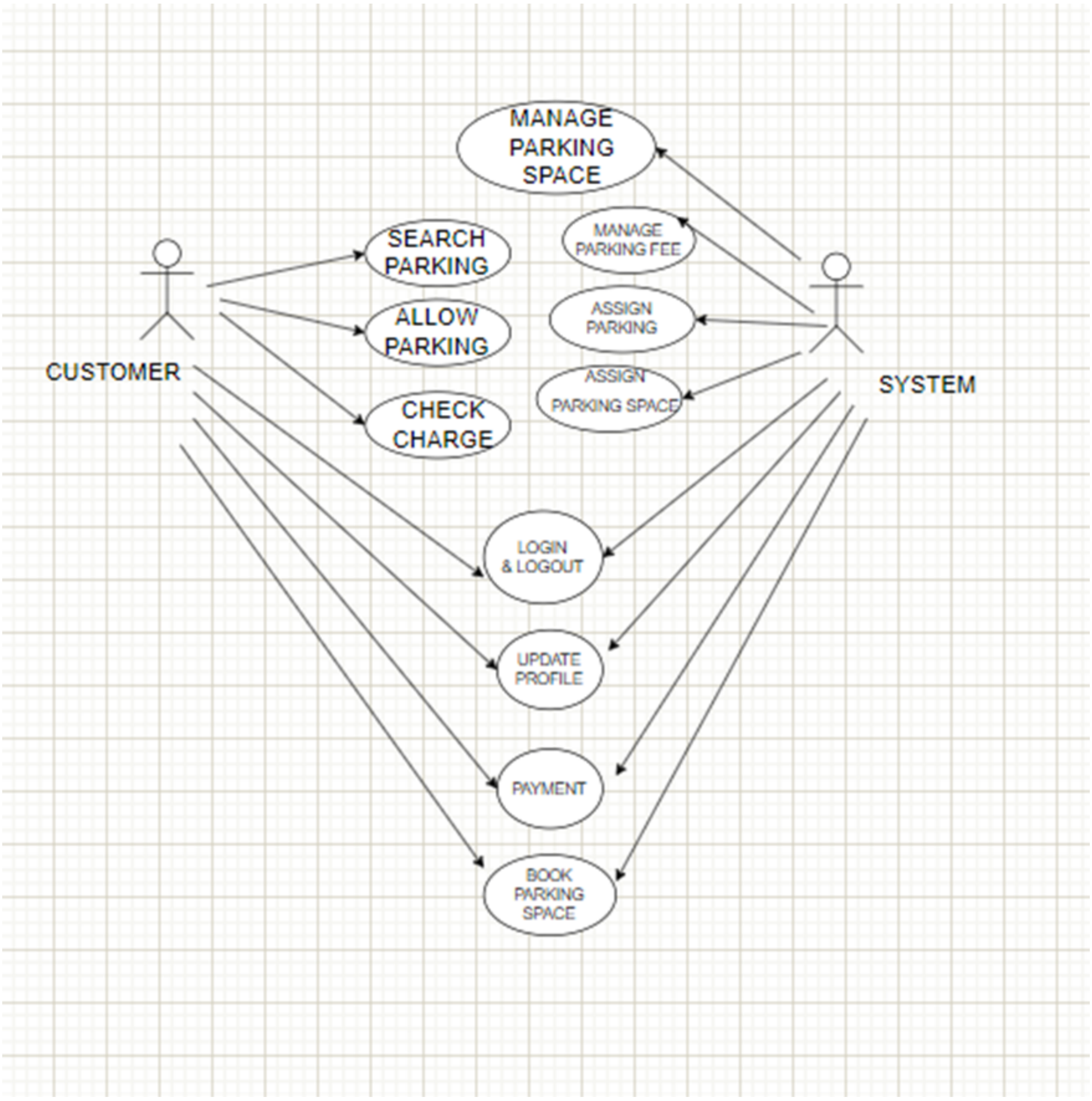


Figure 2 – Context Diagram of Smart Parking System

manage parking fee, assign parking, assign parking space, login and log out, update profile, payment and book parking space.

2.3.1.4. Design And Implementation Details

The application has been designed for large-scale public use therefore it was required to keep the design and implementation constraints as low as possible however a few of them could be:

1. GUI is in English and other languages.
2. The budget could be a major constraint, for example, if the customer wants more options and is low on budget then it becomes a major design constraint.
3. Any changes in the design demanded by the client in the latter stages of application development could bring great implementation troubles.
4. Users should have basic knowledge of how to install and to operate a mobile or desktop application

2.3.1.5. Operating Environment

The application can be run on any computer (machine), since it is a web application, it requires an internet connection. The application is supported by Windows, Linux, and macOS. No other hardware component is required other than a computer to access the application.

2.3.1.6. Assumptions And Dependencies

1. Roles and tasks are pre-defined.
2. The voting results will be managed and calculated by the Blockchain.
3. Administrator is created in the system already

2.3.2. External Interface Requirements

2.3.2.1. User Interfaces

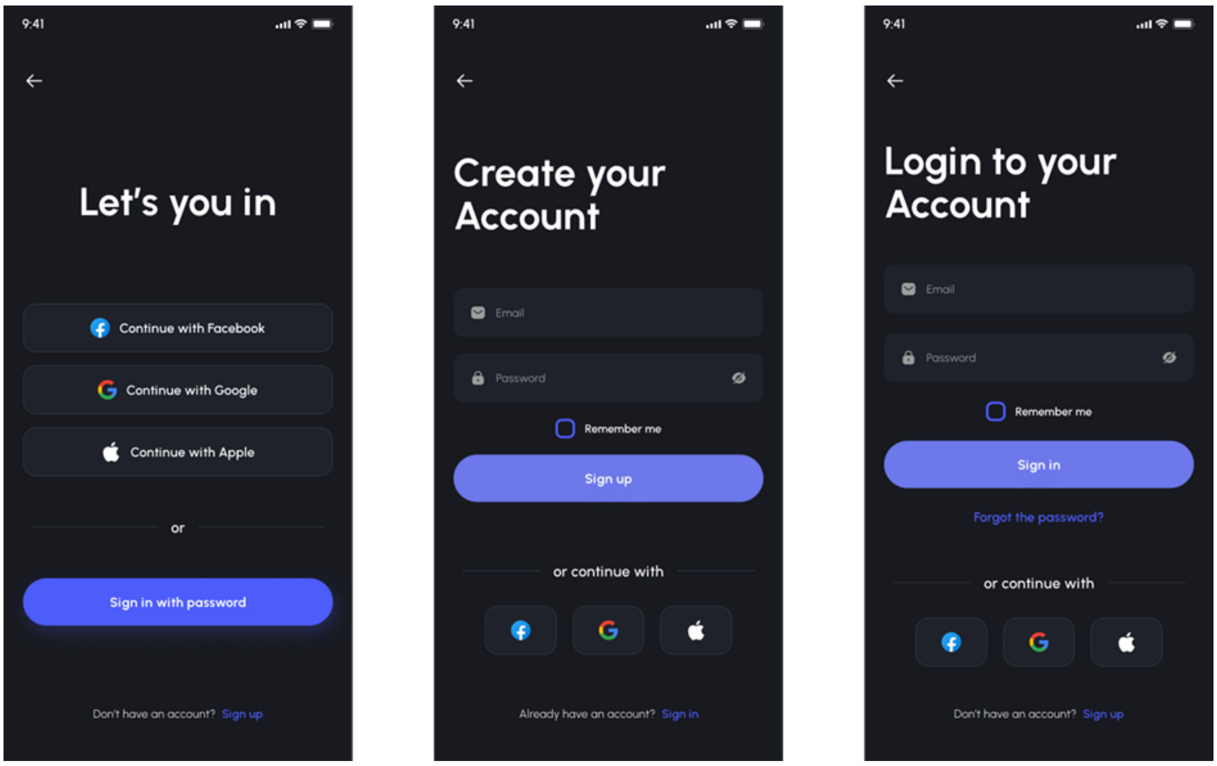
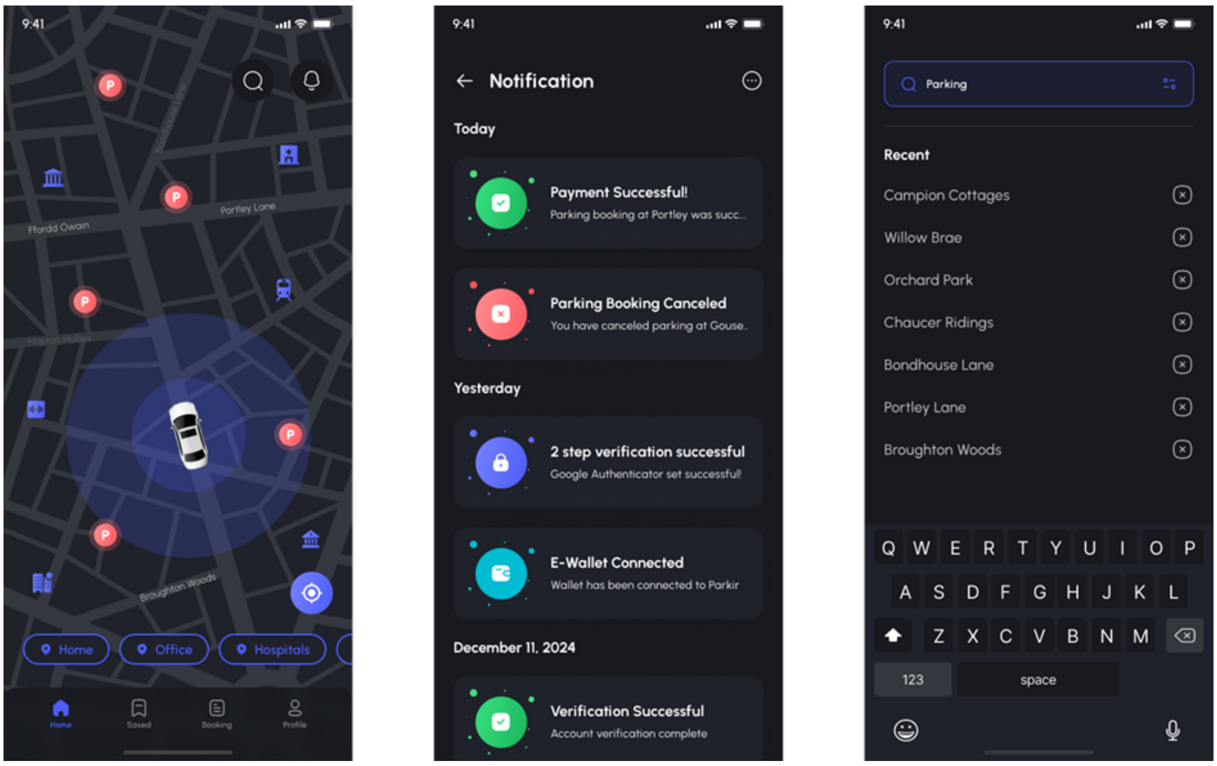


Figure 3 – User Interface for Login and Sign-up

This login interface (Figure 3 authenticates the users of the app before redirecting them to their home portal.



9:41

Notification

Today

Payment Successful!
Parking booking at Portley was succ...

Parking Booking Canceled
You have canceled parking at Gouse...

Yesterday

2 step verification successful
Google Authenticator set successful!

E-Wallet Connected
Wallet has been connected to Parkir

December 11, 2024

Verification Successful
Account verification complete

9:41

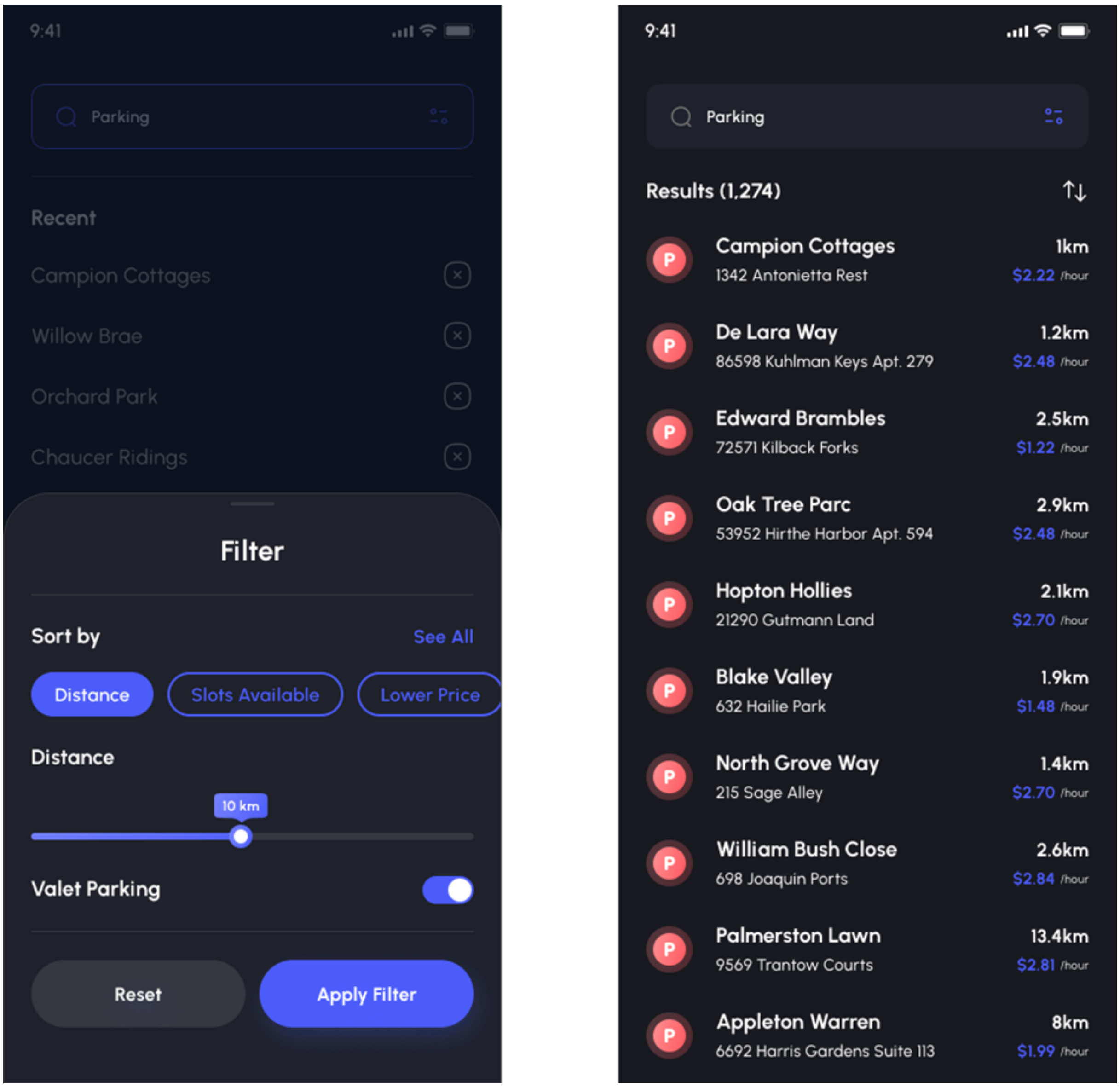
Parking

Recent

Campion Cottages
Willow Brae
Orchard Park
Chaucer Ridings
Bondhouse Lane
Portley Lane
Broughton Woods

Figure 4 – User Interface for Home Portal

This is home portal (Figure 4) where users can view their notifications, search for a parking spot and go there with their vehicle via GPS.


9:41

Parking

Recent

Campion Cottages

Willow Brae

Orchard Park

Chaucer Ridings

Filter

Sort by — See All

[✓ Distance] Slots Available Lower Price

Distance: 10 km

Valet Parking: on

Reset | Apply Filter

9:41

Parking

Results (1,274)

Campion Cottages	1342 Antonietta Rest	1km	$2.22 /hour
De Lara Way	86598 Kuhlman Keys Apt. 279	1.2km	$2.48 /hour
Edward Brambles	72571 Kilback Forks	2.5km	$1.22 /hour
Oak Tree Parc	53952 Hirthe Harbor Apt. 594	2.9km	$2.48 /hour
Hopton Hollies	21290 Gutmann Land	2.1km	$2.70 /hour
Blake Valley	632 Hailie Park	1.9km	$1.48 /hour
North Grove Way	215 Sage Alley	1.4km	$2.70 /hour
William Bush Close	698 Joaquin Ports	2.6km	$2.84 /hour
Palmerston Lawn	9569 Trantow Courts	13.4km	$2.81 /hour
Appleton Warren	6692 Harris Gardens Suite 113	8km	$1.99 /hour

Figure 5 – Search Parameters and Sorting Results

The figure 5 shows the interface for fine-tuning and filters available to the user while searching for a parking spot.

2.3.2.2. Hardware Interfaces

2.3.2.2.1 Arduino Uno Arduino Uno is the embedded machine in our system. The utilized microcontroller is an ATmega328P. Arduino Uno is an open-source development platform with strong community support. This may be programmed using the Arduino IDE, which includes a built-in function that reduces the user's work. In this system, this embedded device is used for reading license plate numbers using Optical Character Recognition (OCR) technology.

2.3.2.2.2 CCTV Camera CCTV (closed-circuit television) is a television system whose signals are not broadcast publicly but are monitored for surveillance and security reasons primarily. CCTV relies on the strategic positioning of cameras and the monitoring of camera inputs on monitors. Due to the fact that the cameras connect with monitors and/or video recorders through private coaxial cable lines or wireless communication links, they are referred to as ”closed-circuit” to signify that their material is intentionally restricted to only those who can view it. In this project, we use the images captured from CCTV cameras to read the license plate of the designated vehicle for verification.

2.3.2.3. Software Interfaces

2.3.2.3.1 OpenCV In the Hardware Interfaces section, we learn that CCTV or a webcam is utilized to take pictures of vehicles' license plates. In order to access and capture images from this camera, the Python library Open Computer Vision (OpenCV) is used. OpenCV is a freely available image processing library that includes a wide variety of useful tools, such as those for decoding, enhancing, converting color spaces, detecting and tracking objects, and more. The photograph is saved to the local computer system. With the help of the secret key supplied by the cloud services, the main source code of the system may access the recorded image that has been delivered there (see Figure 4 below).

2.3.2.3.2 Cloud Services Cloud services provide serverless hosting and powerful database queries which can help us to authenticate users, store and retrieve their information. To integrate these services with the proposed application, Development Kits or APIs will be used which will be provided by the cloud providers.

2.3.2.4. Communication Interfaces

2.3.2.4.1 Web Browser / Mobile Application The application can be accessed via a web browser to provide maximum compatibility with all existing devices. The web app can also be cross-compiled to Android and iOS as a native app via various technologies like React Native or Flutter.

2.3.2.4.2 Email Users of the app will get all important notifications, support, and policy updates via email. It is also the way to sign up for the app's services. Convenient one-click sign-in systems like "Sign with Google" will be provided which bypasses the process of manually entering the email address.

2.3.2.4.3 Cloud Synchronization Synchronization services shall be done via cloud databases like **FireStore** or **AWS S3**. These services have pay-per-use services which charge according to monthly use and are easy to scale as the load increases.

2.3.3. Requirements

2.3.3.1. Business Requirements

- **BR.1** Build an effective solution that will solve the vehicle parking problem in urban areas.
- **BR.2** Give the drivers a simple and easy to use application to find free parking pots.
- **BR.3** Give the garage owners or spot owners the to utilize their empty property to earn money.
- **BR.4** Build a platform that connects the drivers and providers.
- **BR.5** Develop a business model that is free to register but paid when the actual service is sought
- **BR.6** Incorporate premium features to the app for paid customers who want special services.

2.3.3.2. User Requirements

2.3.3.2.1 Driver Class The end-users of the application.
User Story: The driver wants an app that will work on both mobile and desktop platforms while synchronizing his/her personal data seamlessly. The driver can log in via his email and password. From the search prompt, the driver can search for a location. The application will show the driver all the available parking spots near that location. The driver can fine-tune his/her search by some filters and sort the search results by parameters like distance, cost, number of slots available, etc. After that, the driver can

select a spot according to his liking. The application will use a map to guide the driver to the parking spot. After the driver has parked his vehicle, he/she can check for the remaining time, and view the real-time video feed of his/her vehicle. Finally, the driver wants to pay via mobile banking services or cash facilitated via the app.

Requirements:

- **UR.Driver.1** The system shall have a registration portal for via email and password.
- **UR.Driver.2** The system shall provide a user-friendly UI and widely accessible mobile application.
- **UR.Driver.3** The system shall provide mobile banking integrations.
- **UR.Driver.4** The system shall provide a secure and low-latency video feed connection.
- **UR.Driver.5** The system shall facilitate the user with guidance with his/her parking spot.

2.3.3.2.2 Provider Class The user class that will provide the parking services.
User Story: The provider wants a similar sign-in process to the regular users. However, their registration process needs additional information like their NID, proof of ownership of the garage of parking spots, and pictures and videos of the parking spots. After a manual review, his/her registration will be approved. The company behind the application will provide all the necessary equipment needed and professionals who will set those up for a fee. After that, the provider's parking spot will be open for rent to the drivers. From his home screen, the provider can see his earnings and transfer his/her funds via mobile banking services.

Requirements:

- **UR.Provider.1** Registration portal for uploading required documents.
- **UR.Provider.2** Pay the registration fee via mobile banking integration.
- **UR.Provider.3** Display the monthly income in the home section.
- **UR.Provider.4** Provide functionality to transfer the funds

2.3.3.3. Functional Requirements

- **FR.Provider.1** The provider shall be able to Register his parking spot and provide personal and property data.
- **FR.Provider.2** The provider shall be able to define new parking spaces, specify a range of parking lots, the parking rate per minute/hour, and other pertinent information.
- **FR.Provider.3** The provider shall be able to change existing parking area data.
- **FR.Driver.1** Users shall be able to Register for the service and provide personal and vehicle data.
- **FR.Driver.2**The user shall be able to locate a parking space in the list of locations registered by parking administrators.
- **FR.Driver.3** The user shall be able to examine the information of a selected parking location, including its name, hourly rate, and total number of available spaces.
- **FR.Driver.4** The user shall be able to reserve a parking spot and define its term.
- **FR.Driver.5** The user shall be able to view live feed of his/her vehicle
- **FR.Data.1** The system shall have access to all registered parking lot information.
- **FR.Data.2** Before altering any sensitive data, the back-end management system shall be able to authenticate users and administrators.
- **FR.Auto.1** The back-end management system shall automatically modify parking lot status.
- **FR.Auto.2** The back-end management system shall be able to automatically cancel a reservation if the user is unsuccessful within the reservation time.
- **FR.Auto.3** The back-end service shall automatically verify the arrival of vehicle via CCTV footage.
- **FR.Auto.4** The back-end management system shall be able to generate a Session ID and deliver it to the shall for every reservation automatically .

- **FR.Auto.5** Once the driver/user check-in, the system shall be able to transmit the license plate number and reservation password (Session ID) to a central server for verification.
- **FR.Auto.6** At the time of checkout, the system shall be able to issue receipts to users automatically.
- **FR.Auto.7** The back-end management system shall accept parking lot reservations depending on availability.

2.3.4. Other Non-Functional Requirements

2.3.4.1. Data-Based Requirements

- **DR.1** Customers' web browsers must never display their passwords. It must always be repeated using special letters that represent typed characters.
- **DR.2** After retrieval from the database, a customer's credit card number should never be shown on their web browser. Always just the last four digits of the credit card number must be displayed.
- **DR.3** The back-end servers of the system shall never store a user's password in plain-text. The customer's password can be reset only if email account is verified.
- **DR.4** Back-end servers of the system shall only be available to authenticated administrators.
- **DR.5** The back-end databases of the system shall be encrypted.
- **DR.6** For storing private data of the users, the rules defined by GDPR shall be strictly followed.

2.3.4.2. Timing Requirements

- **TR.1** Any atomic action shall not take more than 100ms of computation time.
- **TR.2** The UI shall be responsive to user's interactions
- **TR.3** For long actions, the UI shall show the progression of the said task.
- **TR.4** Long searching actions, like finding a parking spot, shall time out after 30s seconds

- **TR.5** The UI shall show proper error messages to the user after the time-out period defined in *TR.4*.
- **TR.6** The application must be light-weight in network transactions so that it works in 3G network.
- **TR.7** The maximum time before the first input or output is expected after a task initiates shall be an average of 1 second.
- **TR.8** The system shall allow only one concurrent connection from one device.
- **TR.9** Premium users can interrupt or pre-empt others in finding parking spots

2.3.4.3. System Quality Attributes

- **Availability:** The app can handle continuous operation with a minimum availability rate of 98
- **Usability:** This website has a suitable user interface and sufficient information to help the user through the process of using the website. It shall have a user-friendly interface that allows users to interact with the product without difficulty.
- **Portability:** The app is portable and can be used on any device regardless of platform or operating system.
- **Performance:** High number of concurrent users or transactions is supported by the system, with an average of 1 second of response time.
- **Security:** This app provides user authentication so that only the authorized user may access the site.
- **Maintainability:** This website is capable of securing and retrieving data effortlessly.
- **Scalability:** The system is expandable a system so that future modifications to this system are possible without degrading its performance.

2.3.5. Validation And Verification

In this phase, we review all three previous phases. We develop acceptance tests like we have done in Specification section and discard unnecessary requirements.

2.3.5.1. Validation

Validation is the process of determining if our parking system application is up to par, or if it meets stringent standards. Usually, dynamic Testing is the good validation process.

The Smart Parking Management System can be validated by the following ways:

1. Blackbox testing
2. Smoke testing
3. Unit testing
4. Integration testing

2.3.5.2. Verification

Verification is the process of ensuring that software does its intended purpose without errors. It is the process of determining whether or not a created product is suitable. It determines whether the generated product meets our specifications. Verification refers to Static Testing.

Verification involves the following:

1. Alpha-beta test
2. Reviews
3. Walkthroughs

3. REQUIREMENT MANAGEMENT

The objective of requirements management is to assure the achievement of our smart parking system development objectives. It is a collection of strategies for documenting, evaluating, prioritizing, and agreeing on requirements so that the marketing team and project managers always have approved and current needs. Requirements management enables the avoidance of mistakes by monitoring changes in requirements and facilitating dialogue with stakeholders from the beginning of a project throughout its entire lifespan.

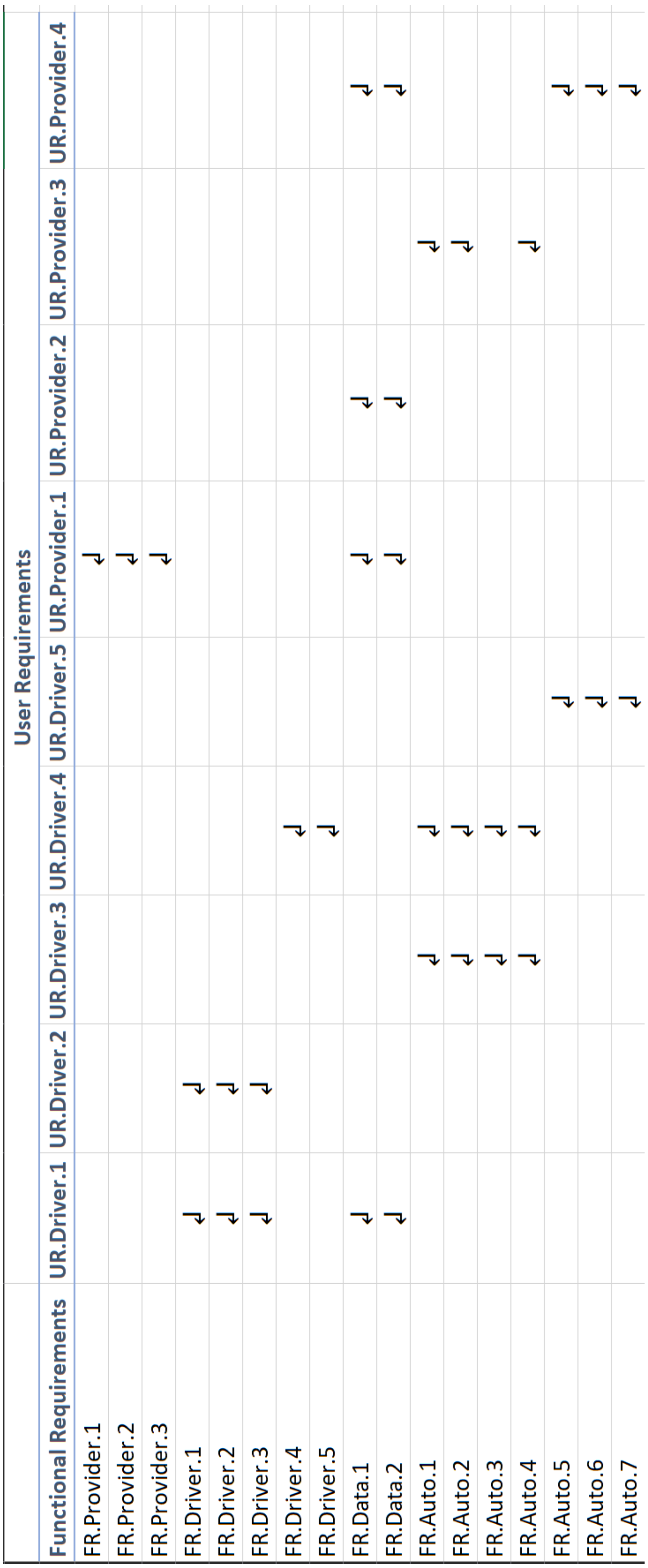
3.1. Change Management

Change management in software development projects involves identifying, planning, and executing software changes. Throughout the software development process, it is utilized. New needs and the necessity for change may appear out of nowhere and fluctuate often. If they are not handled appropriately, a project may be ruined. Change management is a systematic approach to dealing with transitions.

1. Developers and testers who will be affected by the change should be surveyed to establish the optimal implementation date.
2. Allow individuals to voice their concerns and guarantee they are addressed promptly and transparently as possible. Throughout the duration of the procedure, maintain contact with team members.
3. Develop a training program with sufficient sessions to familiarize users with the new methods.
4. Determine probable obstacles and how to overcome them if they emerge.
5. Monitor the process's development and be willing to make adjustments as necessary.
6. Conduct through impact analysis before incorporating any change to the requirements.

3.2. Requirement Traceability Matrix

Traceability matrices are used in software development to determine the completeness of a relationship by comparing any two baseline documents using a many-to-many relationship comparison. In this section, the relationship of the *functional requirement* and *user requirements* in the proposed application is explained through the traceability matrix.



	User Requirements								
Functional Requirements	UR.Driver.1	UR.Driver.2	UR.Driver.3	UR.Driver.4	UR.Driver.5	UR.Provider.1	UR.Provider.2	UR.Provider.3	UR.Provider.4
FR.Provider.1						✓			
FR.Provider.2						✓			
FR.Provider.3						✓			
FR.Driver.1	✓	✓							
FR.Driver.2	✓	✓							
FR.Driver.3	✓	✓							
FR.Driver.4				✓					
FR.Driver.5				✓					
FR.Data.1	✓					✓	✓		✓
FR.Data.2	✓					✓	✓		✓
FR.Auto.1			✓	✓				✓	
FR.Auto.2			✓	✓				✓	
FR.Auto.3			✓	✓					
FR.Auto.4			✓	✓				✓	
FR.Auto.5					✓				✓
FR.Auto.6					✓				✓
FR.Auto.7					✓				✓

Figure 6 – Requirement Traceability Matrix

4. PACKAGED SOLUTIONS

4.1. Application Development Frameworks

For developing the application itself, we can various existing frameworks. Since our application needs support for various platforms, it is preferable to use frameworks that can cross-compile to different platforms. This reduces the amount of work needed to port the application to different platforms and also the upkeep needed to maintain these different ports. The updates to the application can be streamlined without worrying about different platform quarks. Some of these frameworks are discussed below.

4.1.1. React Native

React Native is excellent for mobile applications. It gives a clean, fluid, and responsive user experience while lowering load time dramatically. Designing apps using React Native is also far faster and less expensive than building native apps, without sacrificing quality or functionality. React Native uses the popular programming language JavaScript for developing its components. So, developers do not need specialized personnel to code with React Native.

4.1.2. Flutter

Flutter is Google's portable UI toolkit for creating natively built mobile, web, and desktop applications from a single source of code. Flutter is compatible with current code, is utilized by developers and organizations worldwide, and is open source and free. It used Dart programming language. Although it is not as popular as JavaScript, the other developer-friendly features of Flutter makes it one of the best if not the best frameworks for building cross-platform application.

4.2. Back-end

4.2.1. Firebase

Firebase is a collection of hosting services for all applications. It provides NoSQL and real-time hosting of databases, content, social authentication, alerts, and services like a real-time communication server. Firebase offers comprehensive documentation and cross-platform SDKs for the development and distribution of Android, iOS, and web applications. The Firebase Realtime Database is a database stored in the cloud. The data is saved as JSON and is synced in real-time with all connected clients.

4.2.2. AWS

Amazon Web Services, Inc. is an Amazon company that offers on-demand cloud computing platforms and APIs to people, businesses, and governments on a pay-per-use basis. Through AWS server farms, these cloud computing web services provide distributed computing processing capability and software tools. AWS is designed to enable application providers, ISVs, and suppliers to swiftly and securely host any applications, whether they are new or old. AWS's application hosting infrastructure is accessible to developers via the AWS Management Console and well-documented web services APIs.

4.3. OCR Model

4.3.1. OpenCV

OpenCV (Open Source Computer Vision Library) is a free software library for computer vision and machine learning. OpenCV was created to offer a standard foundation for computer vision applications and to expedite the adoption of machine perception in commercial goods. In the proposed project, OpenCV can be used to develop a **Optical Character Recognition** model that specializes in reading the license plate of vehicle.

4.3.2. TFlite

TensorFlow Lite is a suite of tools that enables machine learning on-device by enabling developers to execute their models on mobile, embedded, and edge devices. It is tailored for on-device machine learning by addressing five important constraints:

1. **Latency:** No round trip to the server.
2. **Privacy:** No personal data leaves the device.
3. **Connectivity:** Internet link is not required.
4. **Size:** Reduced model and binary size

5. **Power Consumption:** Efficient inference and a lack of network connections

It also supports numerous platforms, including Android, iOS, embedded Linux, and micro-controllers. It supports several languages, including Java, Swift, Objective-C, C++, and Python. It has hardware acceleration and model optimization which improves performance. All these benefits are excellent for deploying our OCR model to Arduino Uno.