T.C. MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE4053 Information Systems Analysis & Design

Project #4: Design

PARK IT

Ertuğrul SAĞDIÇ - 150116061

Ali ÇETİNKAYA - 150119606

1. System Acquisition Strategy

We are going to use the Custom development in-house strategy for developing user and admin mobile applications using Flutter/Dart framework. However, we are going to use the Purchase software package strategy for attaching the application with a server using Firebase.

We are going to create a system which allows them to keep track of the density of the parking lots, check in and check out the parking lot with a QR code system and keep track of payments which include monthly memberships. These functionalities are unique for the parking lot company.

Also, the development team has experience with the Flutter framework. However, Firebase is a prebuilt system which only requires creating and making connections with the application that we will develop. The team also has experience creating / connecting the database system using Firebase.

The project manager of the project is not highly skilled. However, for this project the methodology we are using is proven and the experience that the team has is enough to cover all the functionalities for developing the application.

The time to develop the application is not flexible. This is why we are using Firebase, which will enable us to focus on creating the mobile application on time.

Request for Proposal

We are going to create a system which allows them to keep track of the density of the parking lots, check in and check out the parking lot with a QR code system and keep track of payments which include monthly memberships for the parking lot company.

For this system, we need an admin mobile application for the company manager to use. Also, we need a user mobile application for the customers of the company. In order to make communications among these applications, we need an online database.

We are going to use the Flutter framework in order to develop these mobile applications. And we will use the Firebase as an online database system. By doing all these, the system will be ready to use.

The schedule for developing the applications is by 14th of June.

Alternative Matrix

Evaluation Criteria	**Relative İmportance (Weight)**	**Alternative 1**	**Score (1-5)***	**Weighted Score**	**Alternative 2**	**Score (1-5)***	**Weighted Score**
Technical Issues							
Developing admin mobile application	25	Developing application using Flutter	4	100	Developing application using Java	1	25
Developing user mobile application	25	Developing application using Flutter	4	100	Developing application using Java	1	25
Developing Database	25	Using prebuilt database	4	100	Creating database from scratch	1	25
Economic issues							
Cost	25	Flutter is free and Firebase is free up to some limitation	5	125	Server costs $20 per year	4	100
Total	100			425			175

* This denotes how well the alternative meets the criteria. 1 = poor fit; 5 = perfect fit.

Since we have experience with flutter and firebase and using these costs nothing, it will be wise to move with the alternative 1 which has the highest score among all alternatives.

2. Architecture

In the parking lot automation system we will have client mobile devices which handle the presentation logic and application logic, and database management system (Firebase) server which will handle our data access logic and data storage. Thus, Two-Tiered Client-Server architecture will be used. The client will be a Thick Client since most of the application logic will be on the client side.

We will not use any virtualization. We will use cloud computing for handling data access logic and data storage. However, Firebase handles these computations for us, so that we will not be concerned about developing data access logic and data storage.

Operational Requirements

- **Technical Environment**
    - The development environment must have Flutter Framework installed.
    - The development environment should have a virtual emulator.
    - The development environment should have always-on network connection permitting real-time database updates.

- **System Integration**
    - The system will read and write the density information of the parking lots to the database.
    - The system will use the camera in order for the customers to check in and check out the parking lot with a QR code system. In addition, this information will be read and written to the database.
    - The system will read and write the payments of the customers.

- **Portability**
    - The system must operate with Android and iOS mobile devices.

- **Maintainability**
    - The system should accommodate new membership types.
    - The system should accommodate new notification messages.

Performance Requirements

- **Speed**
    - The network transaction response time should be less than 4 seconds. (Depending on the network speed of the user)

- **Capacity**
    - Firebase has a free plan called Spark Plan.

    - Using this free plan, the system can handle a maximum of 100 active users at once.
    - Using this free plan, a maximum of 10GB data to be transferred within one month and store only a maximum of 1 GB of your data.

- **Availability and Reliability**
    - The system should perform 99% uptime performance.

Security Requirements

- **System Value Estimation**
    - A complete loss of all system data would cost $10,000.

- **Access Control**
    - The payments of all customers which include monthly memberships can be seen by only the managers of the parking lot company.
    - Check in and check out the parking lot with a QR code system should be made only by the customers of the parking lot company.
    - The payments of the user should be only seen by the user which made the payments.

- **Encryption and Authentication**
    - Firestore automatically encrypts all data before it is written to disk. Also, the data is automatically and transparently decrypted when read by an authorized user.
    - Firebase Authentication provides backend services, easy-to-use SDKs to authenticate users to the application.

- **Virus Control**
    - There will be no need for virus check since there are no files needed to upload to the database.

Cultural/Political Requirements

- **Multilingual**
    - The system will operate in English and Turkish.

- **Customization**
    - The managers can customize the notification messages that they are going to send.
- **Making Unstated Norms Explicit**
    - All payments will be done in Turkish liras.

- **Legal**
    - The personal information of the customers cannot be transferred to any third party organizations.

The admin and user mobile application will be developed with Flutter framework. Thus, the system will be able to run on the iOS device which has version 12 or more, and on the Android device which has the version of 5 or more.

3. Interfaces

Use Scenario: Check in into the parking lot
User wants to park his/her car into the parking lot by using the QR code system.

1. If a user does not have an account he/she needs to sign up to the system (1.1). If the user has an account already, users should login before using the system (1.2).
2. After signup or login, users should tab the QR code screen on the application in order to enter the parking lot (2.3).

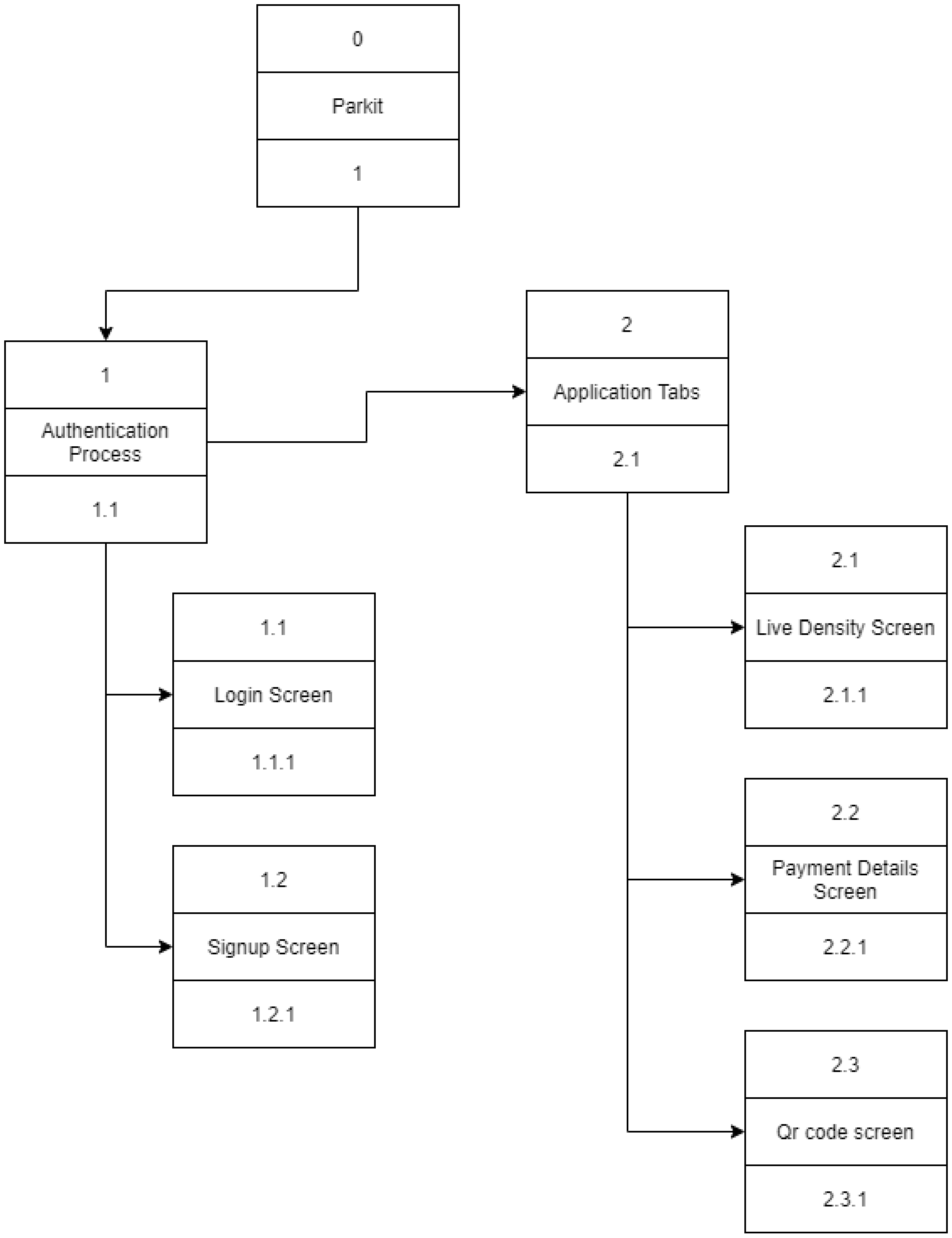
Use Scenario: Check out from the parking lot
User wants to leave the parking lot by using the QR code system.

1. Users should be already signed up to the system in order to park their cars into the parking lot. However, they need to login before using the system (1.2).
2. After login, users should tab the QR code screen on the application in order to leave the parking lot (2.3).
3. After system response, the user should be able to see the payment details from the payment details screen (2.2).

Use Scenario: Monitor the density of parking lots live
User wants to Monitor the density of the parking lots live.

1. If a user does not have an account he/she needs to sign up to the system (1.1). If the user has an account already, users should login before using the system (1.2).
2. After signup or login, users should tab the Live Density screen on the application in order to monitor the density information of the parking lots (2.1).

Interface Structure Diagram



Interface Standards

The interface metaphor is the QR code system in order to check in and check out to the parking lot. This is the core idea of the system to automate the parking lot entrance system.

In order to have a terminology that fits the users' expectations consistently, the interface objects and actions names are chosen carefully. For example, "Density" is chosen for the intensity of the parking lots.

For the interface Icons, the Flutter's icon package will be used in order to reach the standards within the icons. Also, the icons will be chosen with the best match of the word.

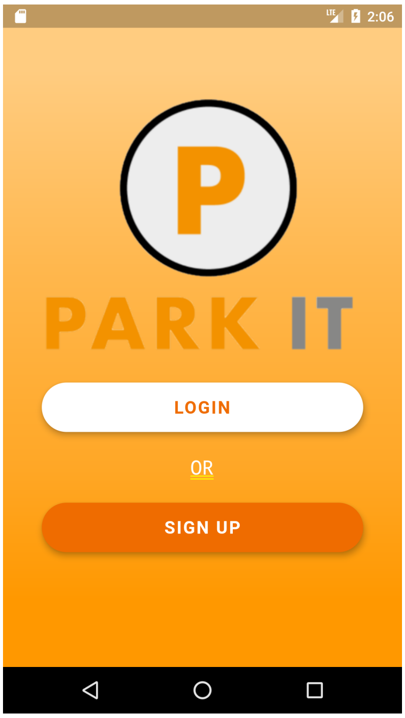
Template for the User Interfaces

Authentication screens template, consisting of login screen and signup screen, will have a basic form which asks for email and password from the user in order to authenticate the users.

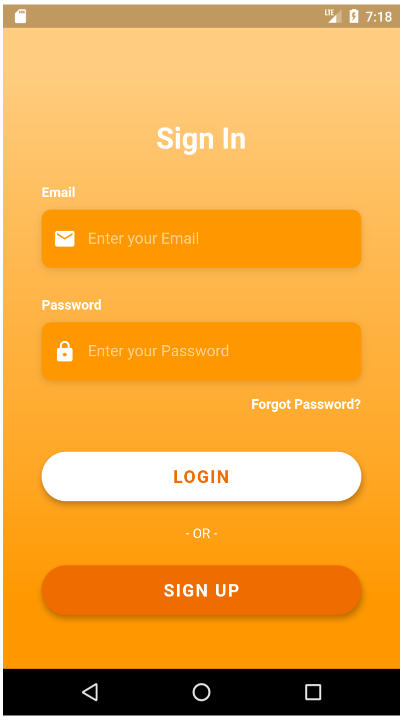
Available parking lots live screen template will have the box layout which will display the density information of the parking lots.

Check in/out with QR code screen template will have the camera in order to read QR code in the entrances of the parking lots.

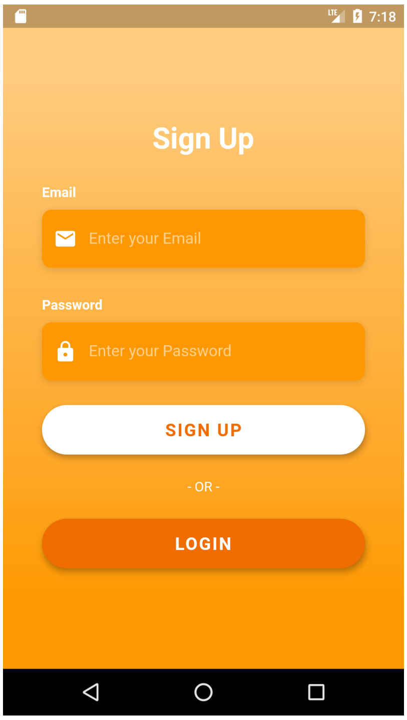
Payments details screen will have the box layout which will display the payment details of the customers.



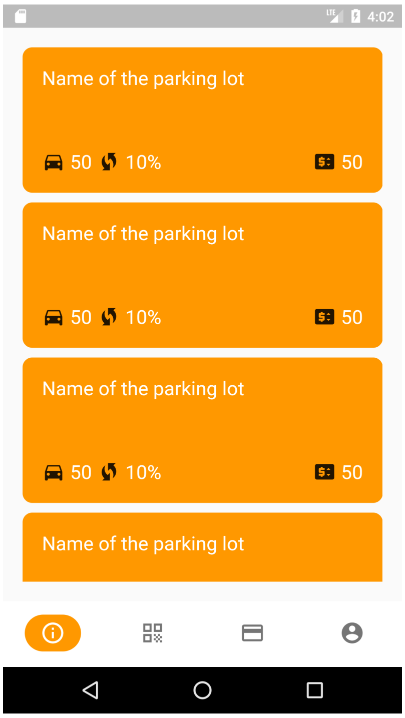
Welcome Screen



Login Screen



Sign up Screen



4:02

Name of the parking lot

50 10% 50

Name of the parking lot

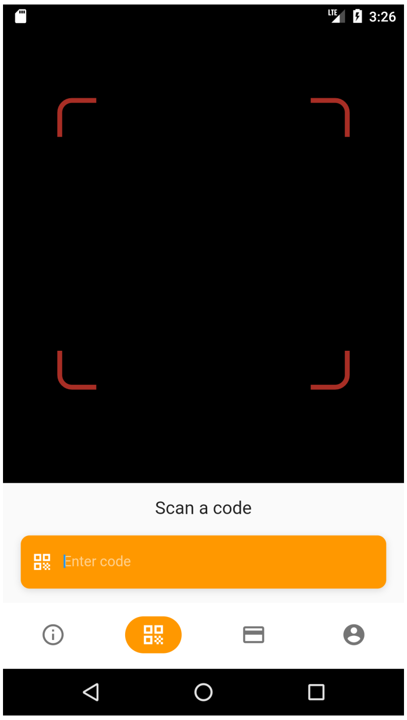
50 10% 50

Name of the parking lot

50 10% 50

Name of the parking lot

Live Density Screen

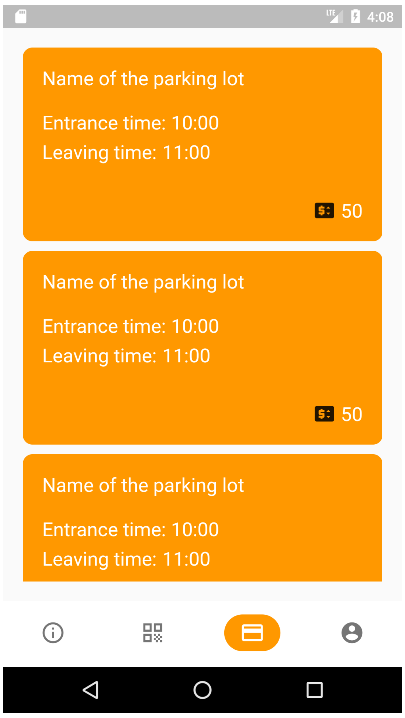


3:26

Scan a code

Enter code

Qr Code Screen



4:08

Name of the parking lot

Entrance time: 10:00
Leaving time: 11:00

50

Name of the parking lot

Entrance time: 10:00
Leaving time: 11:00

50

Name of the parking lot

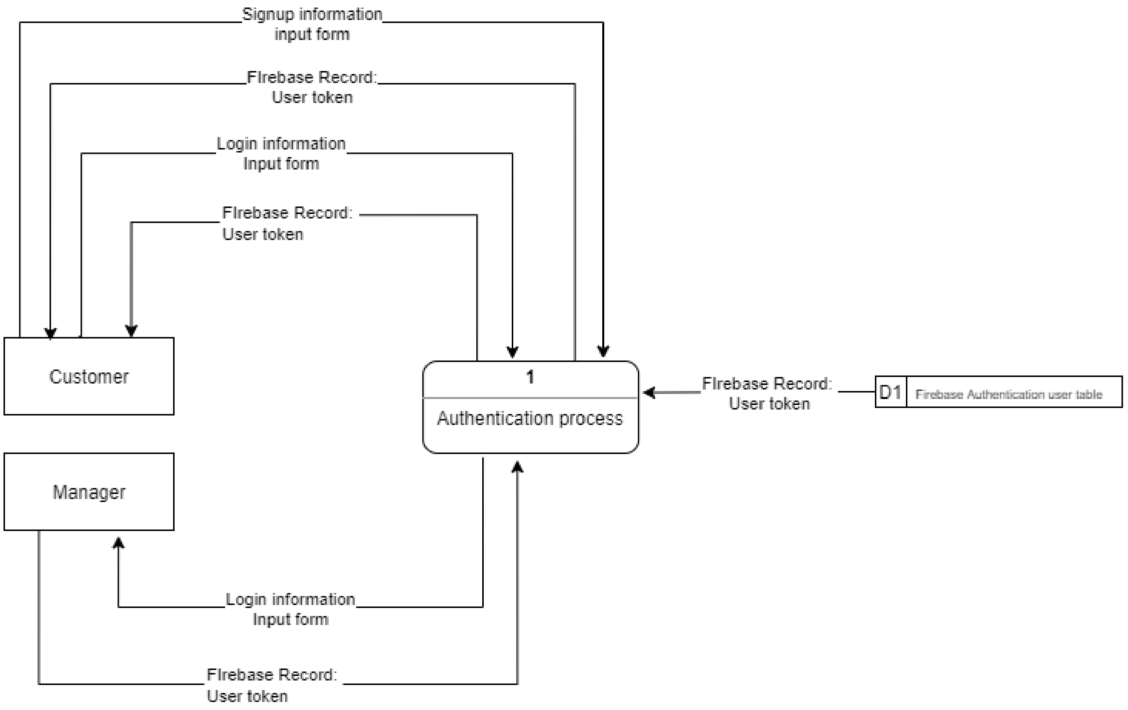
Entrance time: 10:00
Leaving time: 11:00

Payment Details Screen

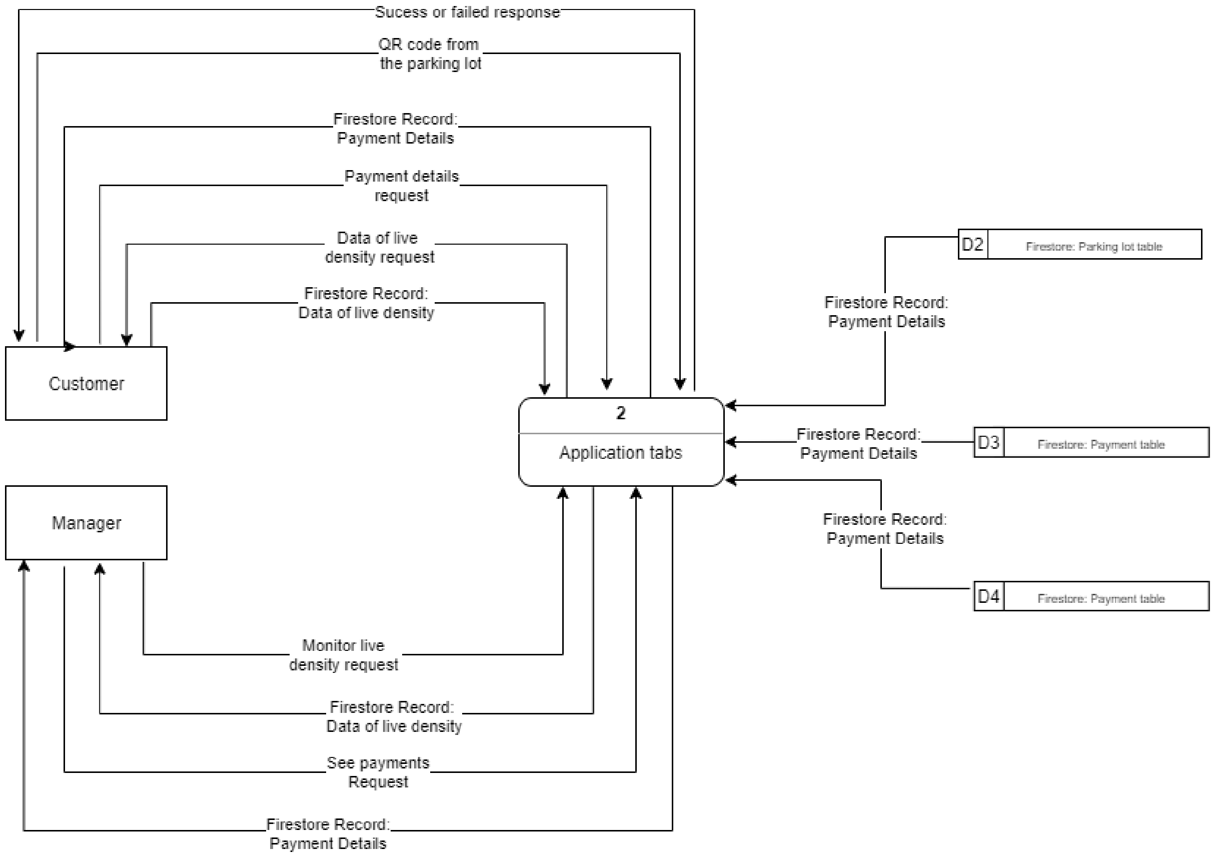
The interactive Evaluation method has been used in order to evaluate the user interfaces of the system. I developed the prototype in Flutter and we discussed each screen one by one in order to have what the customer wants.

4. Programs

Physical Models

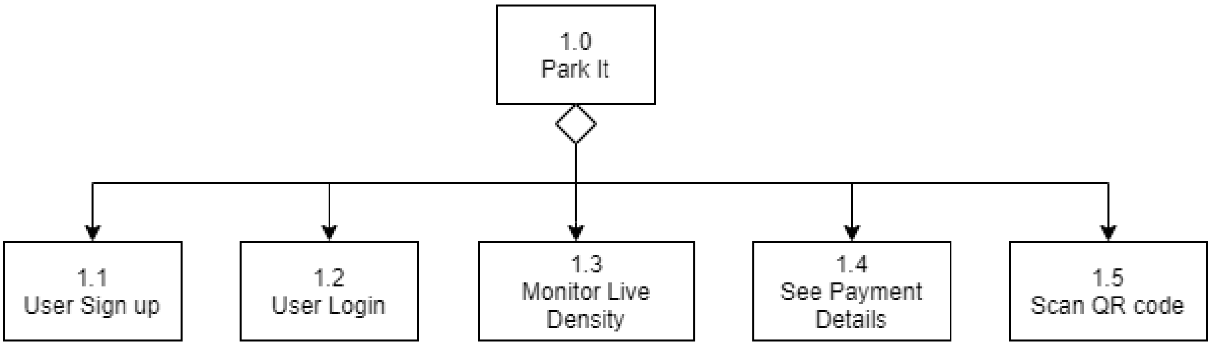


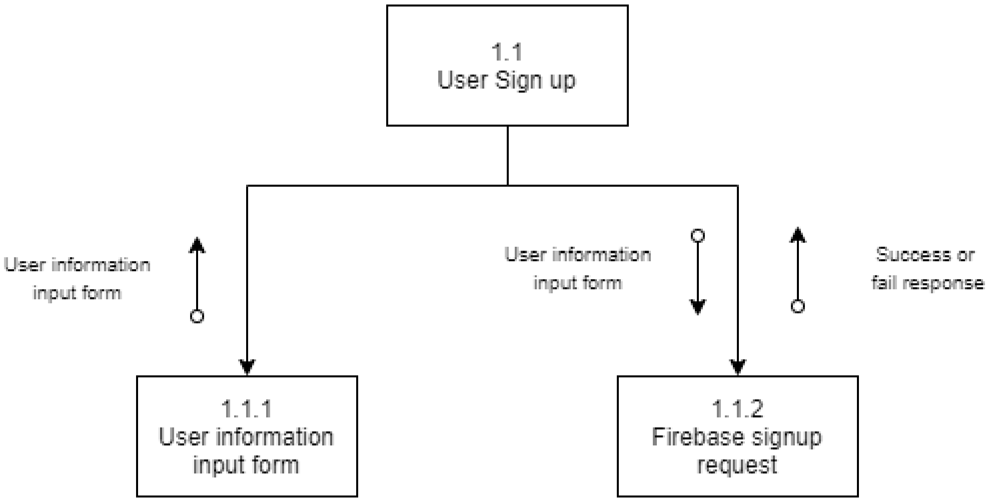
Signup information input form
FIrebase Record: User token
Login information Input form
FIrebase Record: User token
Customer
1
Authentication process
FIrebase Record: User token
D1
Firebase Authentication user table
Manager
Login information Input form
FIrebase Record: User token

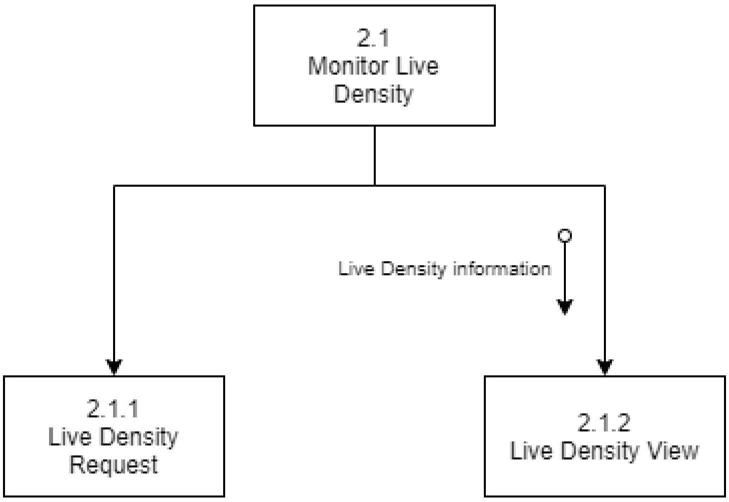


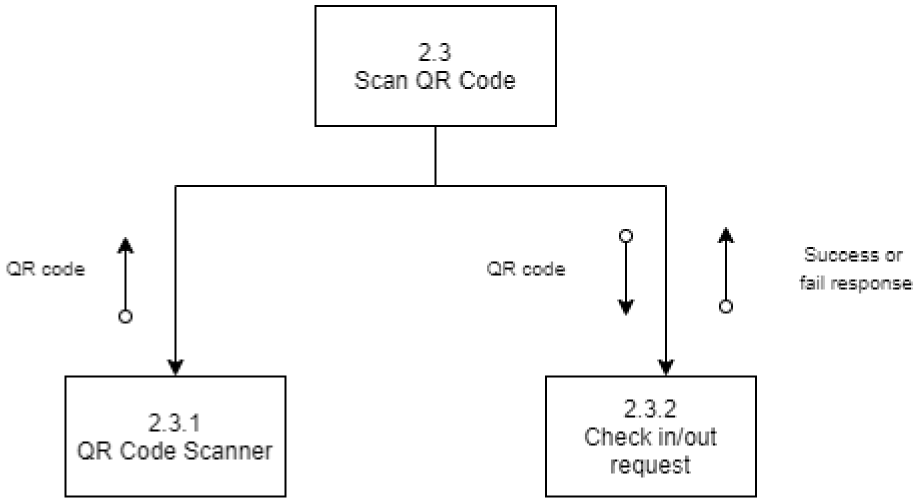
Sucess or failed response
QR code from the parking lot
Firestore Record: Payment Details
Payment details request
Data of live density request
Firestore Record: Data of live density
Customer
2
Application tabs
D2
Firestore: Parking lot table
Firestore Record: Payment Details
Firestore Record: Payment Details
D3
Firestore: Payment table
Firestore Record: Payment Details
D4
Firestore: Payment table
Manager
Monitor live density request
Firestore Record: Data of live density
See payments Request
Firestore Record: Payment Details

Program Structures




1.1
User Sign up
User information input form
User information input form
Success or fail response
1.1.1
User information input form
1.1.2
Firebase signup request





Program Specification

Module Name: userSignUp
Purpose: Sign up new user with user information which are email and password.
Programmer: Ertugrul Sagdic
Due: 14.06.2021
Programming Language: Dart
Events: Sign up user when sign up button is clicked.

Input Name	Type	Provided by
email	string	program 1.1.1
password	string	program 1.1.1

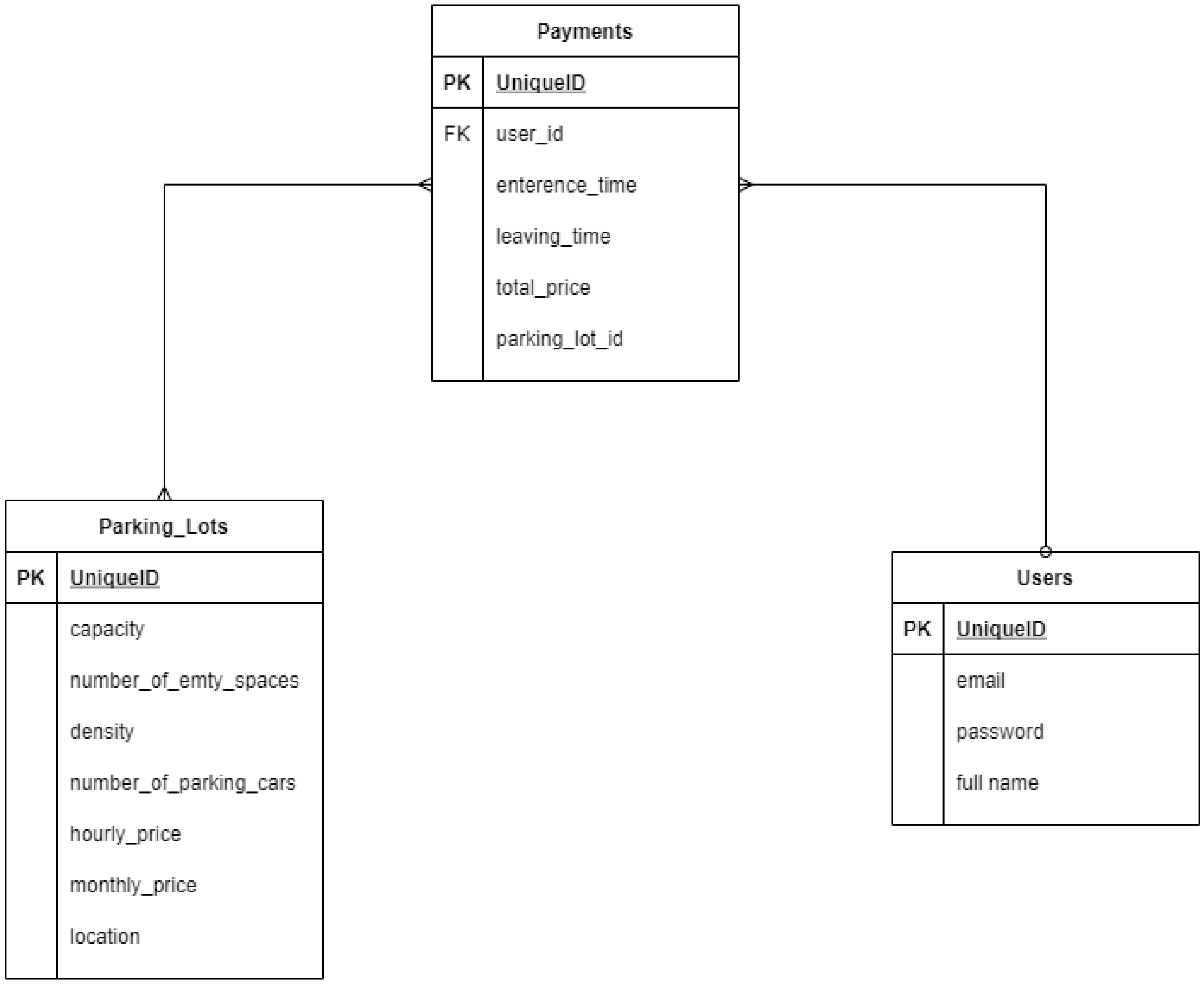
Output Name	Type	Used by	Note
response	boolean	program 1.1.1	Used to let user know if the process is successful or not

Pseudocode:

```
    get email and password from the sign up input form
    call the sign up request function of the firebase with email and password
    return response
```

5. Databases

A complete diagram for the physical data model



In the data model, we need a Users entity to hold the user information. We will be able to authenticate the user with email and password. Also, There will be a token which is basically a uniqueID of the user. Finally, the user's full name.

In addition, we have a Parking_Lots entity to identify parking lots of the company. We have capacity, number_of_empty_spaces attributes, and number_of_parking_cars to calculate density. With density information, users can monitor the live data. Hourly price and monthly price variables to calculate the payments. Finally the location of the parking lot.

Lastly, the Payments entity is to bill the user. We will show the payment details of the user and all payments for managers.

For the relation between Parking_Lots and Payments, there can be many Payments for many Parking_Lots.

For the relation between Users and Payments, there can be many Payments for a user