

Crowd Management

“TAAKAD Application”

Name	ID
1. Abdulmajeed Mohammed Alamri	43803419
2. Ahmed Ali Al-Duais	43812958
3. Faris Mohammed AL-Otaibi	43812546
4. Majed Abdullah AlGhamdi	43706829
5. Meshal Bandar Hadhrawi	43807307

Supervisor:

Dr.Abdulrahman ALQahtani.

Department of Computer Science

College Computers and Information Technology

Taif University, KSA

Contact Information

Fall 2022

Authors Information		
Name	Number	Email
1. Abdulmajeed Mohammed Alamri	0551756288	abdulmajeed.m.alamri@gmail.com
2. Ahmed Ali Al-Duais	0532178042	ahmedalduais18@gmail.com
3. Faris Mohammed AL-Otaibi	0547677046	farisksa79@gmail.com
4. Majed Abdullah AlGhamdi	0555045899	paradonrex@gmail.com
5. Meshal Bandar Hadhrawi	0555702500	meshal.b.hadhrawi@gmail.com

Supervisor

Dr. Abdulrahman ALQahtani

Department of Computer Science

Students' Property Right Declaration

I hereby declare that the work in this capstone project at Taif University is my own except for quotations and summaries which have been duly acknowledged. This work with the title "Crowd Management Recommendation System :Using Machine Learning Approach" has not been accepted for any degree and is not concurrently submitted for award of other degrees. It is the sole property of Taif University, and it is protected under the intellectual property right laws and conventions.

Authors:

Name:

Abdulmajeed Mohammed Alamri

Signature:



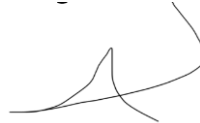
Date:

14 May 2022

Name:

Ahmed Ali Al-Duais

Signature:




Date:

14 May 2022

Name:

Faris Mohammed AL-Otaibi

Signature:



Date:

14 May 2022

Name:

Majed Abdullah ALGhamdi

Signature:



Date:

14 May 2022

Name:

Meshal Bandar Hadhrawi

Signature:



Date:

14 May 2022

Supervisor:

Name:

Abdulrahman ALQahtani.

Signature:

Date:

Students Anti-Plagiarism Statement

I hereby declare this report is my own work except for properly referenced quotations, and contains no plagiarism; it has not been submitted previously for any other assessed unit on this or other degree courses.

I have read and understood the School's rules on assessment offences which are available in the Taif University Handbook

انا الممضي أسفله أشهد أن هذا التقرير هو عملي الخاص انا و مجموعة الطلبة المذكورة أسماؤهم بأول هذا التقرير ما عدا ما هو مذكورة مصادره صراحة و أنه لا يحتوي على محتويات منقولة بدون عزوها لكتابتها الأصلي. و أشهد أن هذا العمل لم يسبق أن استخدم كعمل رسمي بمقررات اخرى بهذه الكلية أو غيرها. أشهد أنني اطلعت على قوانين الكلية الخاصة بتقييم الطلبة الموجود بالكتاب التقديمي للجامعة

Authors:

Name:

Abdulmajeed Mohammed Alamri

Signature:



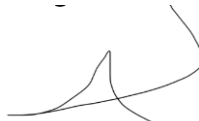
Date:

14 May 2022

Name:

Ahmed Ali Al-Duais

Signature:



Date:

14 May 2022

Name:

Faris Mohammed AL-Otaibi

Signature:



Date:

14 May 2022

Name:

Majed Abdullah AlGhamdi

Signature:



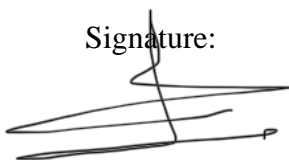
Date:

14 May 2022

Name:

Meshal Bandar Hadhrawi

Signature:



Date:

14 May 2022

Dedication

First and foremost, we thank God Almighty, our supervisor, Dr.Abdulrhman ALQahtani, for efficiently guiding us through this project, and our families. And to all of our friends for their continuous encouragement. We also want to express our gratitude to each and every participant in our project.

Crowd management is a public practice where large crowds are managed. It used to ensure the control of gatherings and crowded places. The goal of that process is to facilitate the movement between places, reduce risks, prevent disasters, and track the correct flow of movement and traffic. Malls and public places are constantly considered as crowded places, sometimes there are visitors with a number greater than the capacity of the place, which leads to overcrowding and congestions, and this causes them to be prevented from entering or unwanted waiting and loss of time. In the literature review, there are some studies and systems that have discussed crowd management in distinct and intelligent ways. However, we have found that their studies do work to know the peak times but do not provide suggestions to users for similar places. This project provides a solution for showing crowds in a form of mobile application. The solution would use historical data to provide suggestions to visitors and identify the peak times in public places and to facilitate access to these places, it would also help in reducing overcrowding and reducing waiting times in public places, which support decision makers to decide and people as well to decide about their trips and other activities.

ملخص مقترح البحث (عربي)

إدارة الحشود هي ممارسة عامة حيث تتم إدارة حشود كبيرة. كانت تستخدم لضمان السيطرة على التجمعات والأماكن المزدحمة. الهدف من هذه العملية هو تسهيل التنقل بين الأماكن وتقليل المخاطر ومنع الكوارث والتدفق الصحيح للحركة. تعتبر المولات والأماكن العامة على الدوام أماكن مزدحمة، ففي بعض الأحيان يكون هناك زوار بعدد أكبر من سعة المكان مما يؤدي إلى الازدحام، أو يؤدي إلى منعهم من الدخول أو الانتظار غير المرغوب فيه وضيق الوقت. في مراجعة الدراسات السابقة، هناك بعض الدراسات والأنظمة التي ناقشت إدارة الحشود بطرق متميزة وذكية. ومع ذلك، وجدنا أن دراساتهم تعمل لمعرفة أوقات الذروة، ولكنها لا تقدم اقتراحات للمستخدمين لأماكن مماثلة. سيقتراح هذا المشروع حلاً لإدارة الحشود من خلال تعزيز تكنولوجيا التعلم الآلي المطورة في شكل تطبيق للهاتف المحمول. يوفر هذا المشروع حلاً لإظهار الحشود في شكل تطبيق للهاتف المحمول. سيستخدم الحل البيانات التاريخية لتقديم اقتراحات للزوار وتحديد أوقات الذروة في الأماكن العامة وتسهيل الوصول إلى هذه الأماكن، كما أنه سيساعد في تقليل الازدحام وتقليل أوقات الانتظار في الأماكن العامة، مما يدعم صانعي القرار والأشخاص بشأن رحلاتهم والأنشطة الأخرى.

Table of Contents

Title Page	i
Contact Information	ii
Students' Property Right Declaration	iii
Students Anti-Plagiarism Statement	iv
Dedication	v
Abstract	vi
Table of Contents	vii
List of Tables.....	viii
List of Figures	ix
CHAPTER 1: INTRODUCTION	1
1.1 Overview and Problem Background	1
1.2 Problem Statement	1
1.3 Objectives.....	1
1.4 Scope	2
1.5 Contrubutions/Sigificanse of Project	2
1.6 Activity Plan.....	2
CHAPTER 2: LITERATURE REVIEW	3
2.1 Introduction	3
2.2 Related Work.....	3
2.3 Comparsion of Related Work.....	4
2.4 Counclusion.....	5
CHAPTER 3: SYSTEM ANALYSIS AND DESIGN	6
3.1 Introduction	6
3.2 Methodology	6
3.3 Requirement Analysis and Definition	9
3.3.1 Functional Requirement	9
3.3.2 Non-Functional Requirements	11
3.4 System Architecture	12
3.5 Description of The Main Use Cases.....	13
3.6 Sequence Diagram.....	21
3.7 Counclusion.....	24
CHAPTER 4: IMPLEMENTATION AND TESTS	25
4.1 Introduction	25
4.2 Hardware and software specifications.....	25
4.3 Class Diagram	28
4.4 ER Diagram.....	30
4.5 Firebase Initialzation	30
4.6 Suggested Time Implementation.....	31
4.7 QR-Code Scanner Implementation	32
4.8 Application Interface	33
4.9 Test case	42
4.10 Future work	46
4.11 Conclusion.....	46
REFERENCES	47
APPENDIX	49

Table 1.1: Activity Plan for the Project	2
Table 2.1: Comparison of Related Works.....	4
Table 2.2: Comparison of Related Works.....	5
Table 3.1: Description of Register Activity	13
Table 3.2: Description of Login Activity.....	14
Table 3.3: Description of Manage User Activity.....	15
Table 3.4: Description of Check Place Activity	16
Table 3.5: Description of Scan QR-Code Activity	17
Table 3.6: Description of Logout Activity.....	17
Table 4.1: Hardware Specification	25
Table 4.2: Software Specification.....	25

Figure 3.1:Waterfall Modle	7
Figure 3.2:System Architecture	12
Figure 3.3:Use Case Diagram for User.....	18
Figure 3.4:Use Case Diagram for Admin	19
Figure 3.5:Use Case Diagram for System.....	20
Figure 3.6 :Sequense Diagram of Register	21
Figure 3.7 :Sequense Diagram of Login	22
Figure 3.8 :Sequense Diagram of Scan QR-Code	23
Figure 3.9 :Sequense Diagram of Logout	24
Figure 4.1:Collection (Place_Info)	27
Figure 4.2:Collection (Visitors).....	27
Figure 4.3:Class Diagram	29
Figure 4.4:ER Diagram.....	30
Figure 4.5:Firebase Initialization.....	30
Figure 4.6:Suggested Time Code.....	31
Figure 4.7:Leave Time Collocation Code.....	31
Figure 4.8:QR-Code Scanner Code	32
Figure 4.9:Start Interface	33
Figure 4.10:Registration Interface	34
Figure 4.11:Login Interface	35
Figure 4.12:Home Interface	36
Figure 4.13:Time Interface	37
Figure 4.14:Date Interface	37
Figure 4.15:Bar Chart Interface	38
Figure 4.16:Maps Interface	39
Figure 4.17:SignIn to Location	40
Figure 4.18:Scan QR-Code.....	40
Figure 4.19:Successfully Entry	41

Figure 4.20:Login Page Test.....	43
Figure 4.21:Register Page Test	43
Figure 4.22:Home Page Test.....	44
Figure 4.23:Camera Page Test.....	44
Figure 4.24:Checkout Page Test	45
Figure 4.25: Test Result.....	45

1.1 Overview and Problem Background

Crowd management is a public practice where large crowds are managed. Effective crowd management is about managing expected and unexpected crowd occurrences.^[1] Crowd crushes can cause many hundreds of fatalities.^[2] It used to ensure the control of gatherings and crowded places. The goal of that process is to facilitate the movement between places, reduce risks, prevent disasters, and the correct flow of movement and traffic. Malls and public places are constantly considered as crowded places, sometimes there are visitors with a number greater than the capacity of the place.

1.2 Problem Statement

The problem that we are trying to present in this project is that the excessive numbers of people in public places and malls leads to congestions, and also to prevent them from entering or unwanted waiting and loss of time, this study, our solution for managing crowd in public areas in a form of a mobile application technology, our solution would use historical data to provide suggestions to visitors and identify the peak times in public places and to facilitate access to these places, it would also help in reducing overcrowding and reducing waiting times in public places, which support decision makers to decide and people as well to decide about their trips and other activities.

1.3 Objectives

In this project, we aim to achieve the following objectives:

1. Implement a mobile application that assists the user in making excellent decisions.
2. To design a system that assist the collection and processing of data and information about the visitor movement and location
3. Provide a suitable suggestion time for the visitors.
4. Make it easy to access, use and rely on.

1.4 Scope

The target users of the system are the people who would like to go to public places such as malls to find out the times that suit them, which aims to facilitate access to them and know more about the places they would like to visit.

1.5 Contributions/Significance of the Project

By the end of this project, we expect to obtain the following benefits:

1. Help identify peak times in public places
2. Supporting decision-makers to take the most appropriate course of action and facilitate movement
3. Facilitating access to public places and reducing overcrowding
4. Reducing unwanted waiting in public places and refusing to enter them

1.6 Activity Plan

Week	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Search for the idea															
Discuss ideas															
Idea selection															
Develop a plan of action															
Collection of information															
Writing project abstract															
Writing project proposal															
Work on report															
Project analysis and design															
Finalize the project															

Table 1.1 Activity Plan for the Project

2.1 Introduction

In this chapter, we will discuss among the many studies that dealt with the issue of crowd management, and that dealt with the issue of adding predictions and suggestions to the system, as well as the presence of applications used in crowd management, also we will review some studies that discuss this field. We will focus on some related work that is designed to manage crowds to make decisions easier, the characteristics of each system. Crowd management is a public practice of controlling big gatherings. Crowd management is the art of dealing with both expected and unforeseen crowds.^[1] In particular, crowd crushes can result in hundreds of deaths.^[2] It was once used to keep crowds and congested locations under control. The purpose of this method is to make it easier to move between locations, decrease hazards, prevent calamities, and maintain proper traffic flow. Malls and public areas are frequently described as crowded, and there are instances when the number of visitors exceeds the facility's capacity.

2.2 Related Works

Crowd-less Crowdless Designed during the height of lockdown, Crowd-less helps users find out how busy supermarkets and other food vendors are before visiting. As well as using already existing data from Google Maps, Crowd-less relies on users for insights and updates, asking them to confirm the accuracy of crowds upon arrival. Some of the disadvantages we looked for is the app does have many problems with connection issues, many major shops are not displayed, also it has a lot of missing information about places, and it has inaccurate system.^[3]

Skip The Crowds the app allows users to search nearby public places, with data in both real-time and based on historical information so that they can pre-plan their trips or go spontaneously. You can also create a favorites list, and monitor crowds regularly, allowing you the freedom to explore your most cherished spots, without the hassle. From our point of view, we believe that Skip the Crowds system has some disadvantages such as that it does not support all mobile platforms such as iOS, it works on Android only.^[4]

Crowd management COVID-19 the goal is to monitor and manage congestion levels in indoor spaces or points of interest (POI), especially shopping malls or stores. The solution is based on a (POI) recommendation system that suggests the closest safe options request a specific point of interest to be visited by the user.^[5]

Hajj crowd management the goal is to track the movement of pilgrims via RFID technology. A location aware mobile solution will also be integrated into this. This will be made available to pilgrims with smartphones to enhance the accuracy and tracking time of the pilgrims and provide them with location-based services for Hajj.^[6]

2.3 Comparison of Related Works

System	Objective of the system	Approach of the system
Crowd-less ^[3]	Crowd-less helps users find out how busy supermarkets and other food vendors are before visiting. ^[3]	Crowd-less relies on users for insights and updates, to collect from them the historical data, asking them to confirm the accuracy of crowds upon arrival and calculating the average time for the visit ^[3]
Skip The Crowds ^[4]	It is an application that allows users to search nearby public places and plan their visits. ^[4]	Using historical data in both real-time and based on historical information using the mobile application. ^[4]
Crowd management COVID-19 ^[5]	The goal is to monitor and manage congestion levels in indoor spaces. ^[5]	A recommendation system that recommends the most nearby safe possibilities requests that the user visit a specified point of interest. ^[5]
Hajj crowd management ^[6]	The idea is to use to track pilgrim movement. This will include a mobile solution that is aware of its position. ^[6]	Using RFID technology to enhance the accuracy and tracking time of the pilgrims. ^[6]

Table 2.1 Comparison of Related Works

Characteristics	Crowd-less ^[3]	Skip The Crowds ^[4]	Crowd management COVID-19 ^[5]	Hajj crowd management ^[6]	Crowd Management “TAAKAD Application”
Platform	Android/IOS	Android	N/A	Android/IOS	Android/IOS
Support decision making	Yes	Yes	Yes	Yes	Yes
Support Google maps	Yes	Yes	Yes	Yes	Yes
Provide suggestions	No	No	No	No	Yes
Provide visualizations	No	No	No	No	Yes

Table 2.2 Comparison of Related Works

2.4 Conclusion

We discussed several systems that are like our suggested system in this chapter. We'll go through the methodology in the next chapter, as well as the system analysis and design.

3.1 Introduction

A software process model is a simplified depiction of a software process. Each process model describes a process from a certain point of view and consequently gives just a portion of the information about that process. A process activity model, for example, may depict the activities and their order but not the responsibilities of the individuals involved. That is, we observe the process's structure but not the specifics of each activity. These generic models are not exact representations of software processes. Rather, they are process abstractions that may be used to describe various software development methodologies.^[7]

3.2 Methodology

There are many software development methodologies, such as waterfall model, rapid model. In this project, we choose a waterfall methodology, Because the system that we will build is clearly defined and there are no drastic changes that may occur during the project. The waterfall model This takes the fundamental process activities of specification, development, validation, and evolution and represents them as discrete development stages: specification, design, implementation, testing, and maintenance. In principle, one stage must be complete before progress to the next stage is possible. In practice, there is significant iteration between stages.^{[8][9]}

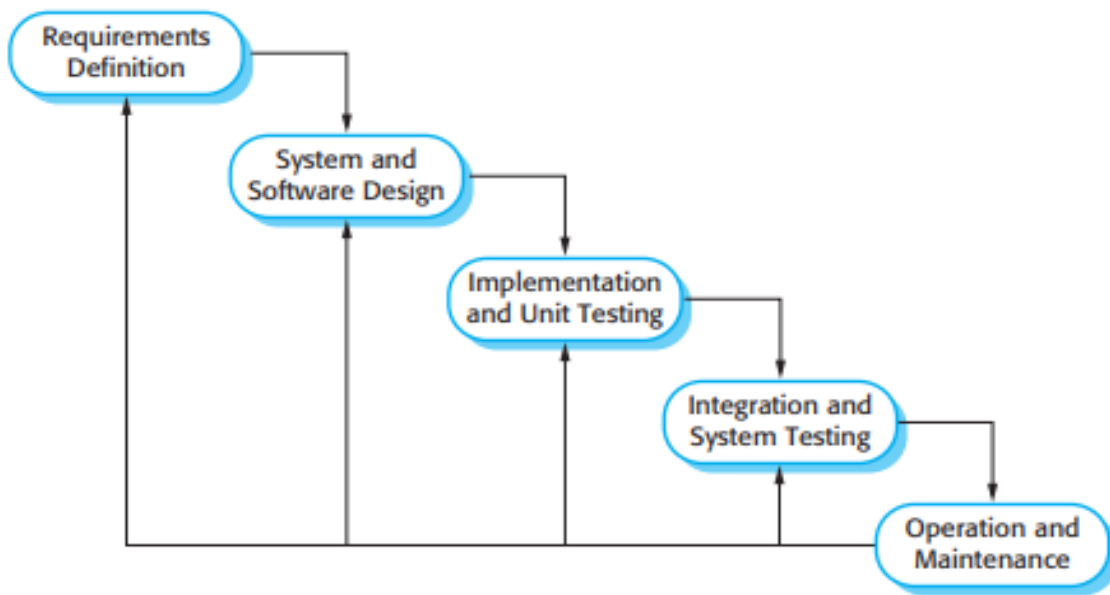


Figure 3.1: Waterfall Model.^[10]

The principal stages of the waterfall model directly reflect the fundamental development activities: ^[10]

1. Requirements analysis and definition the system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
2. System and software design the systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. Implementation and unit testing during this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. Integration and system testing the individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. Operation and maintenance normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

Advantages

The waterfall development process is simple to comprehend and manage due to its linear character. The waterfall technique is best suited for projects with defined objectives and predictable needs. The waterfall development process may be particularly beneficial to less experienced project managers and project teams, as well as teams whose composition changes regularly.

Disadvantages

Waterfall development method is often slow and expensive due to its strict structure and strict controls, and the inability to modify at some point in the project before the previous stage ends, these shortcomings can push waterfall users to explore other software development methodologies.

3.3 Requirements Analysis and Definition

The process of identifying needs and conditions to meet and implement what is to be built or modified, it includes all tasks to be performed keeping in mind the potential conflicting requirements of stakeholders. There are two types of requirements: the functional requirements and the nonfunctional requirements.^[11]

3.3.1 Functional Requirements:

User Functional Requirement

In our system, there are two types of accounts (Visitor and User). So that visitors are those who do not want to register in the program to preserve their privacy, but they don't have the same features that we will provide in the future work for our application.

1. The user can be able to register in the system.
 - The user can the ability to register in the system to use all the features of our application in the future work.
2. The user can log-in in the system.
 - If the user is not registered in the system, the user can use the application to check the specific place
3. The user should be check in.
 - The check-in activity gives the user or the visitor to sign into the chosen place.
4. The admin shall be able to modify information of the users.
5. The admin shall be able to delete users.
6. The admin shall be able to create users.

System Functional Requirement

In our system the functions are clear and defined based on what the system will do, and these functions will be performed automatically by the system features working through algorithms as described in the functional requirements.

1. The system shall be integrated with Google map.
 - The system is linked with the Google Maps service to give users access to the site through the application.
2. The system shall show the dashboard state to the user.
 - The system displays to the user, after specifying the place, time, and date of the visit, the statistics of the specified site and the number of current visitors.
3. The system shall be able to identify and analysis QR-CODE.
 - Among the capabilities of the system can analysis the QR-code of each specific location.
4. The system shall give suggestions and visualizations to the user.
 - After choosing the exact time and day to visit the place, the system gives the user the suggested time for the visit.
5. The system shall allow the users to communicate with technical support.
6. The system shall be able to send notification.

3.3.2 Non-Functional Requirements

Shows the number of used non-functional requirements that can present a software with high performance and security usage. Indeed, the non-functional requirements are considered as part of software characteristics which presented in the points of this section as the following:

Availability

1. The system shall provide the services without interoperation.
 - One of the characteristics of the system is that it provides the service without interruption in the service to ensure the quality of service in the application.

Environmental

2. The system should be connected to the internet.
 - To use Google Maps services, the system must be connected to the Internet.

Security

3. The system should be authenticating the user.
 - The system authenticates if the user is registered in the system to use all future features in the application.
4. The system shall protect the personal information.

Reliability

5. The system shall perform multiple functions in short time.
 - The short time means that the user can search for the location and show the user the current location data without having to wait for the new data to be shown.
6. The system shall respond to user commands smoothly.
7. The system shall be able to handle more than one request from multiple users.

3.4 System Architecture

In this Figure 3.2: The user uses the application to perform the main tasks in the system depicted in the drawing, as seen in the software architecture. The app is built with the Flutter framework and connected to the Firebase platform, which is a back end as a service that provides data storage using Firestore and user verification where the user sends the request.

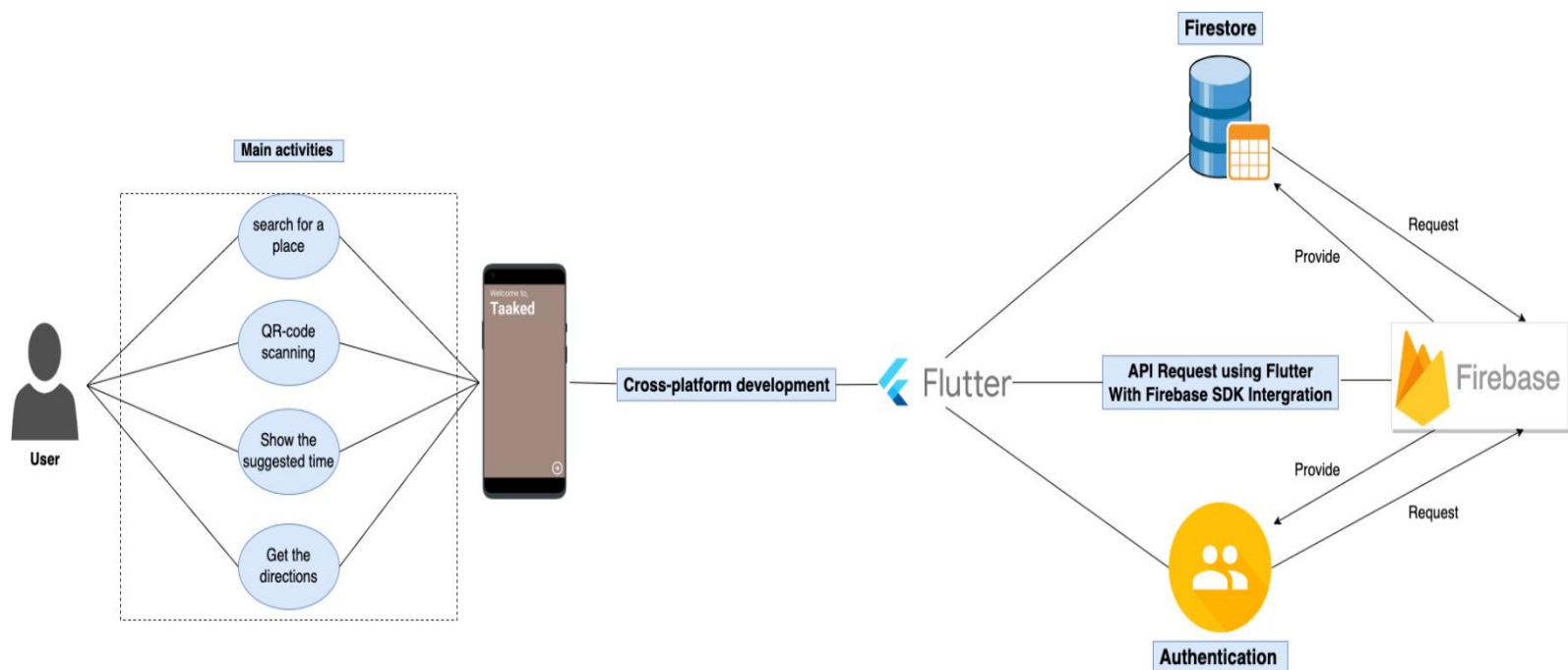


Figure 3.2: System Architecture

3.5 Description of the Main Use Cases

Register Activity	
Use case ID:	1
Description:	Admin and user fill the data needed to create a new account like full name, email, username, and password.
Actor:	Admin, User.
Precondition:	The admin and user must fill all the information requested by the registration form correctly.
Post condition:	Complete the registration and save the information.
Main Flow:	<ol style="list-style-type: none">1. Open the application.2. Click on create new account.3. The admin and user enter information in the registration form.4. Click on "Create account" button.5. The system checks the entered information.6. Save the entered information.7. Send message "Registration success".
Alternative Flow:	<p>5.some information has not been entered or entered incorrectly.</p> <ul style="list-style-type: none">- The system display error message.- The system returns to the flow step 3.

Table 3.1: Description of Register Activity

Login Activity	
Use case ID:	2
Description:	Admin and User can access their account by filling the login form with a correct email and password.
Actor:	Admin, User.
Precondition:	The admin or user has filled a correct email and password, system verify the information.
Post condition:	Login to the system.
Main Flow:	<ol style="list-style-type: none"> 1. Open the application. 2. The admin and user enter the needed information: email, password. 3. The system check the information. 4. Send verify message. 5. Use the verify message to login in the system.
Alternative Flow:	<ol style="list-style-type: none"> 3.Missing email and/or password: <ul style="list-style-type: none"> - The system display error message. - The system returns to the flow step 2. 5.The verify message is invalid: <ul style="list-style-type: none"> - The system display error message. - The system returns to the flow step 4.

Table 3.2: Description of Login Activity

Manage User Activity	
Use case ID:	3
Description:	Admin can create, read, update, and delete user accounts from the system.
Actor:	Admin.
Precondition:	You must be login with an admin account.
Post condition:	Ability to use admin privileges.
Main Flow:	<ol style="list-style-type: none"> 1. Open the application. 2. Admin make correct login. 3. System shows the screens for admin. 4. Admin click “Manage user accounts”. 5. Click (create, read, update, delete) button. 6. The system verifies that the function has been executed. 7. System message “Done successfully”.
Alternative Flow:	<ol style="list-style-type: none"> 2. Wrong login. <ul style="list-style-type: none"> - The system display error message. - The system returns to the flow step 2. 6.The function not excited. <ul style="list-style-type: none"> - The system display error message. - The system returns to the flow step 5.

Table 3.3: Description of Manage User Activity

Check Place Activity	
Use case ID:	4
Description:	User can search and know the peak time for the places
Actor:	User
Precondition:	User shall be login with a user account.
Post condition:	Ability to use user activities.
Main Flow:	<ol style="list-style-type: none"> 1. Open the application. 2. User login. 3. System shows the dashboard for user. 4. User can search for the places. 5. System give suggestions for similar places. 6. System give predictions depending on the user chosen place.
Alternative Flow:	<ol style="list-style-type: none"> 2. Invalid login. <ul style="list-style-type: none"> - The system displays error message. - The system returns to the flow step 2. 4. Invalid entry. <ul style="list-style-type: none"> - The system displays error message. - The system returns to the flow step 3.

Table 3.4: Description of Check Place Activity

Scan QR-Code Activity	
Use case ID:	5
Description:	User can scan the QR-code
Actor:	User
Precondition:	User shall be login with a user account.
Post condition:	Ability to use user activities.
Main Flow:	<ol style="list-style-type: none"> 1. Open the application. 2. User login. 3. System shows the QR-code icon for the user.
Alternative Flow:	<ol style="list-style-type: none"> 2. Invalid login. <ul style="list-style-type: none"> - The system displays error message. - The system returns to the flow step 2.

Table 3.5: Description of Scan QR-Code Activity

Logout Activity	
Use case ID:	6
Description:	Users wants to Exit from the system
Actor:	Admin, User
Precondition:	The admin or user shall be logged in the system
Post condition:	The admin or user logged out from the system
Main Flow:	<ol style="list-style-type: none"> 1. Open the application 2. Make correct login 3. Use the application 4. Click “logout” button 5. application message “successfully logged out”
Alternative Flow:	<ol style="list-style-type: none"> 2. Invalid logout. <ul style="list-style-type: none"> - The system displays network error message

Table 3.6: Description of Logout Activity

Use Case Diagram: is a type of representation that describes the interactions between the system's users and the system, the functional requirements can be summarized by a use case diagram (**Figure 3.3, Figure 3.4, Figure 3.5**).

• **Figure 3.3:** the diagram describes the user activities such as login and register activity which includes checking user information also checking places which includes search places and giving suggestions and scanning code and logout activity.

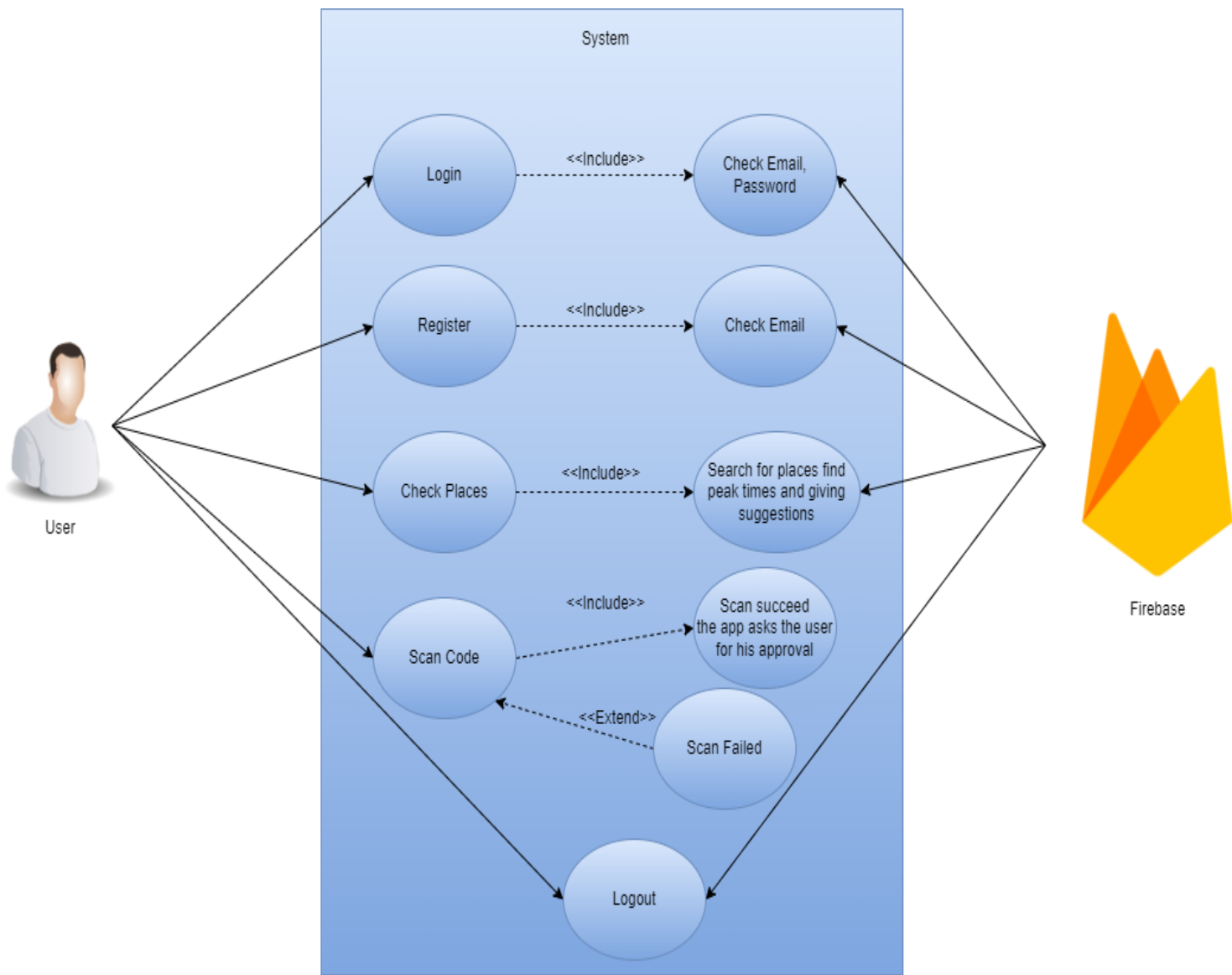


Figure 3.3 Use Case Diagram for User

- **Figure 3.4:** the diagram describes the admin activities such as login and register activity which includes checking user information also managing user include (create, update, etc.) and logout activity.

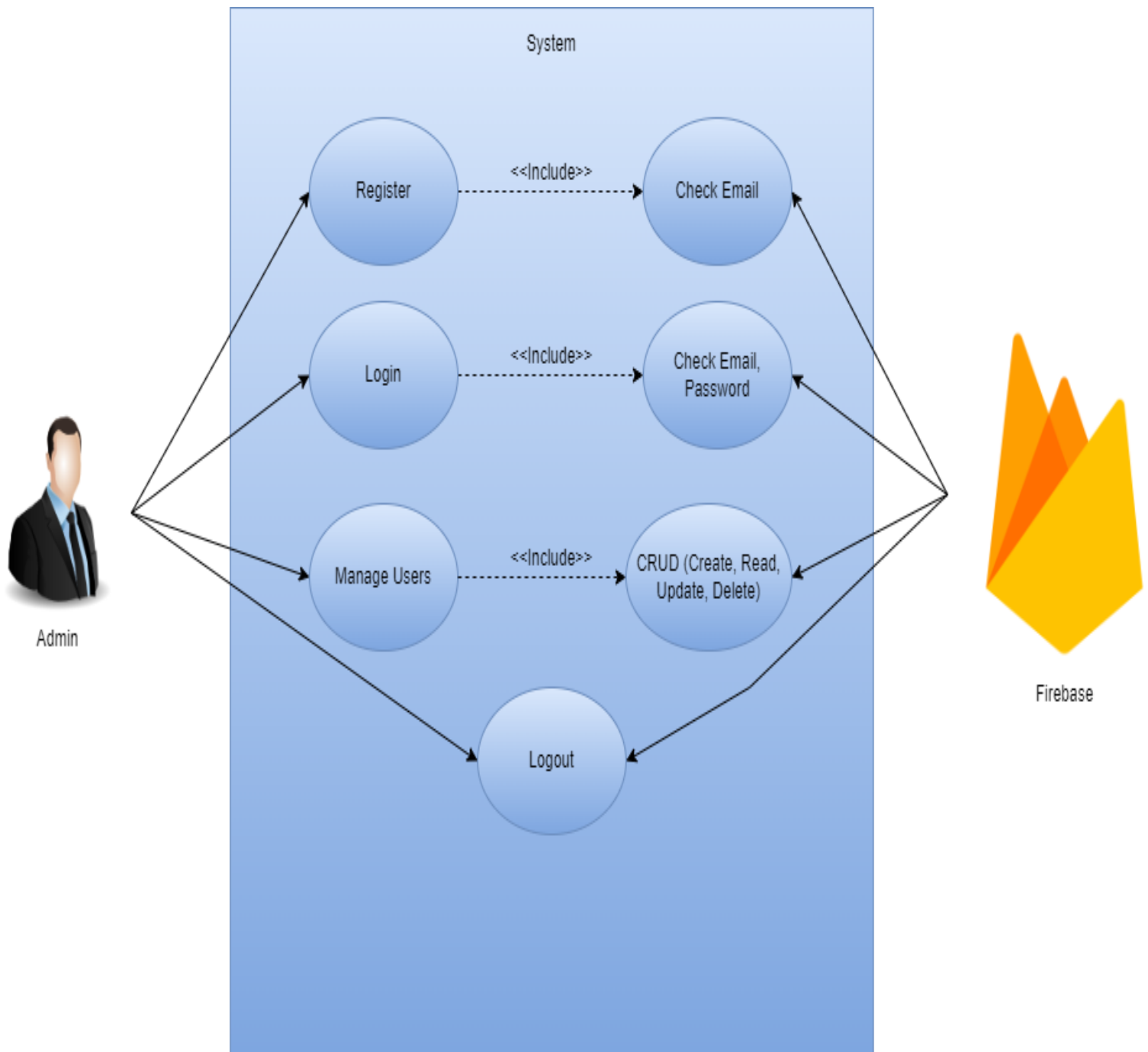


Figure 3.4 Use Case Diagram for Admin

- **Figure 3.5:** describes the admin and user activities that are the same as figure 3.3 & 3.4 in one diagram.

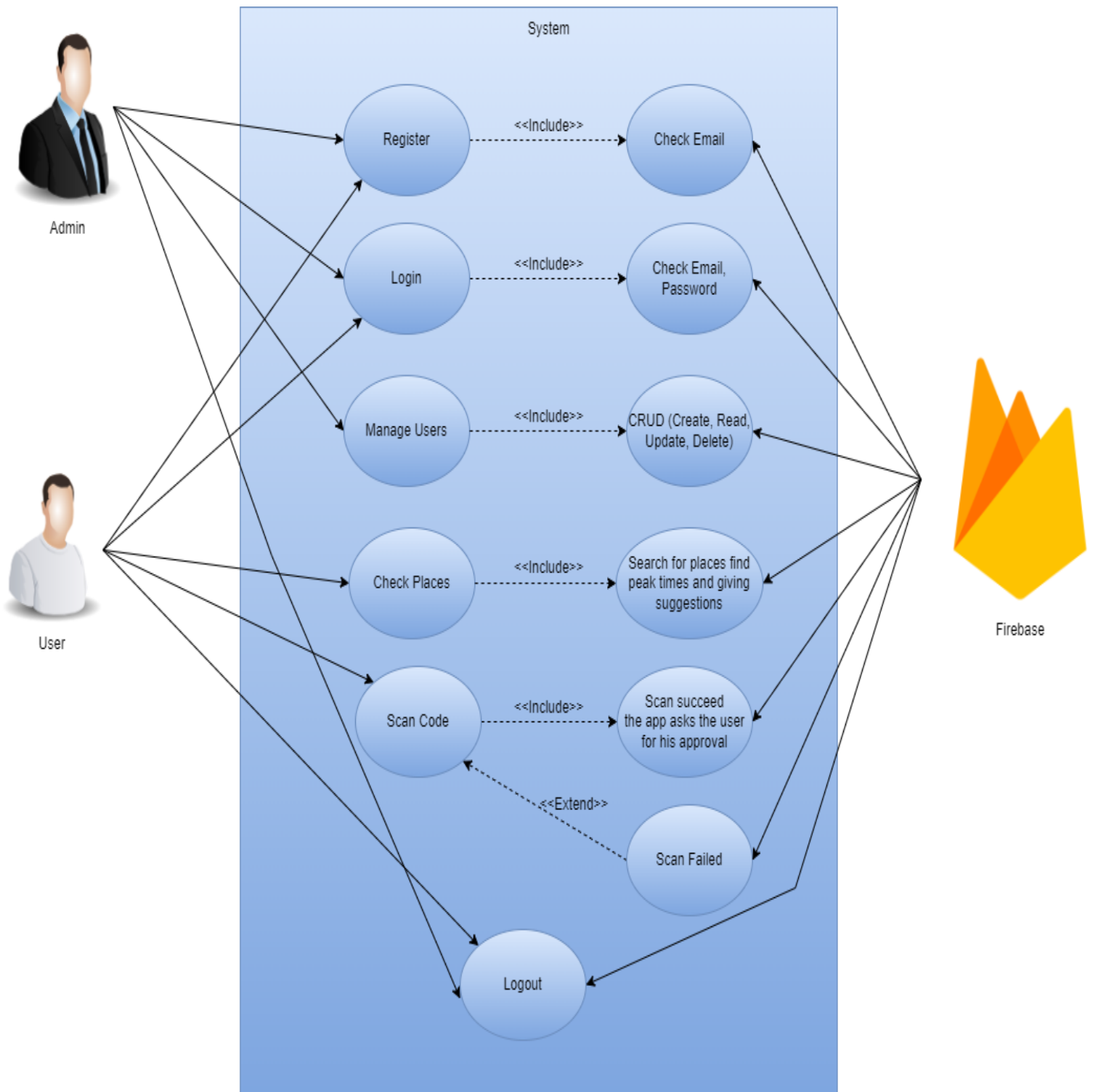


Figure 3.5 Use Case Diagram for System

3.6 Sequence Diagram

A sequence diagram for each scenario (Login, Register, Logout, Scan QR) of the user application. The sequence diagram related to create mobile application is the following:

- **Figure 3.6:** This diagram shows how the user to interact with the system to create a new account by entering his or her information like email, password, full name the visitor can open an account easily without issues.

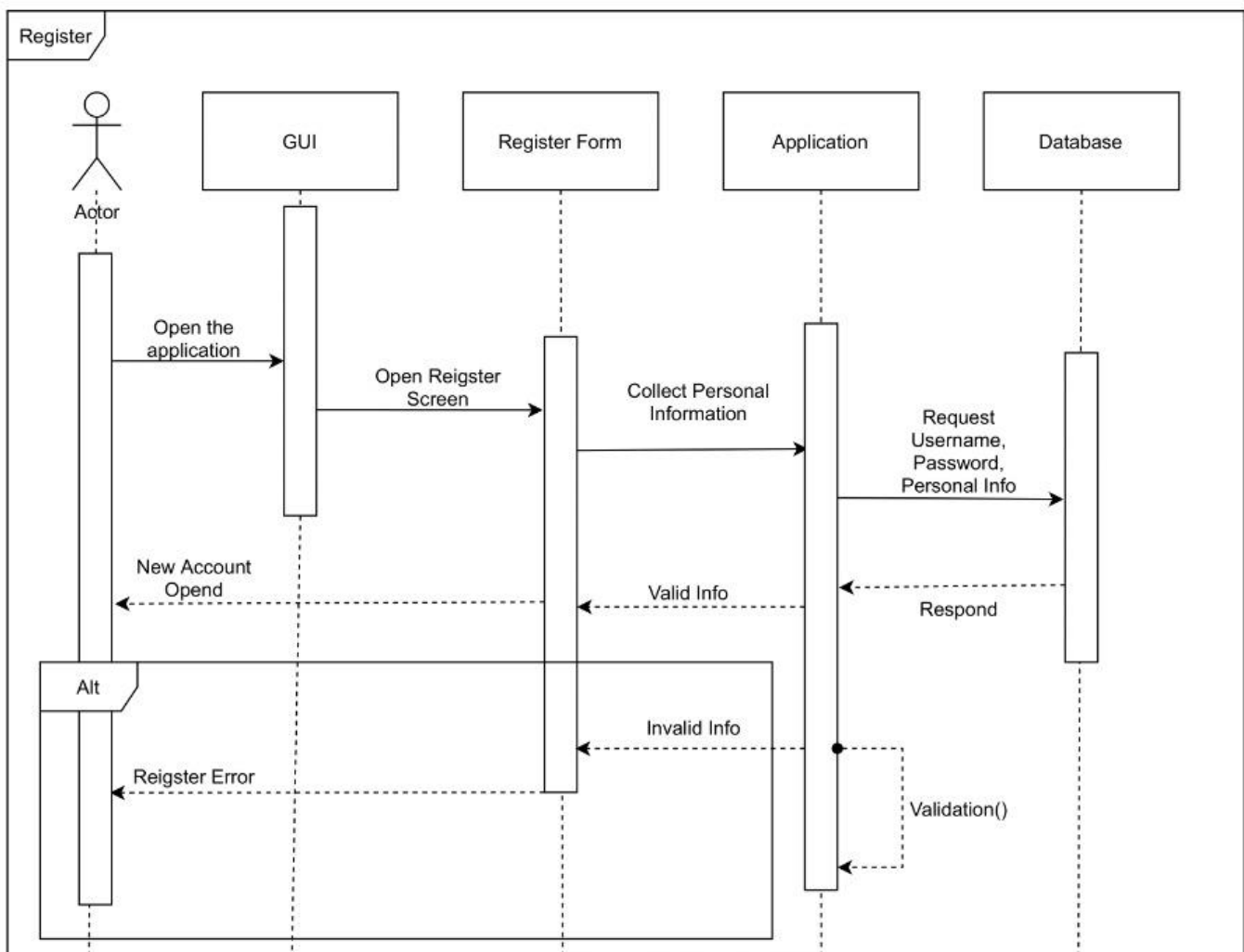


Figure 3.6: Sequence Diagram for Register

- **Figure 3.7:** this diagram shows how the user can interact with the system to login to his or her account using email address and password and check them against the database then return a message to welcome the user if the info is correct.

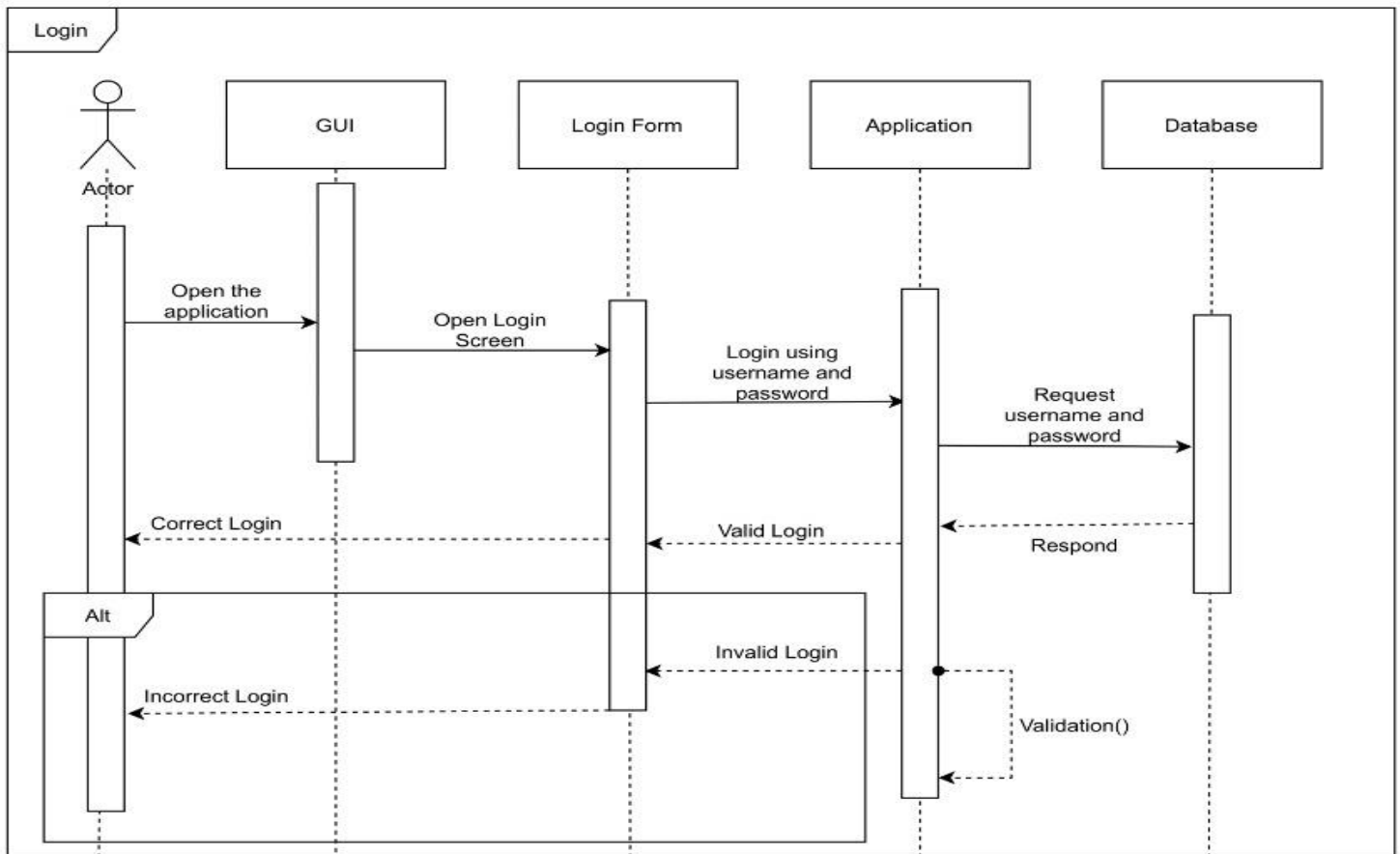


Figure 3.7: Sequence Diagram for Login

- **Figure 3.8:** This diagram shows how the user can scan the QR code and using average time of the previous visitors the system can return prediction and suggestion to the user and show it on the application.

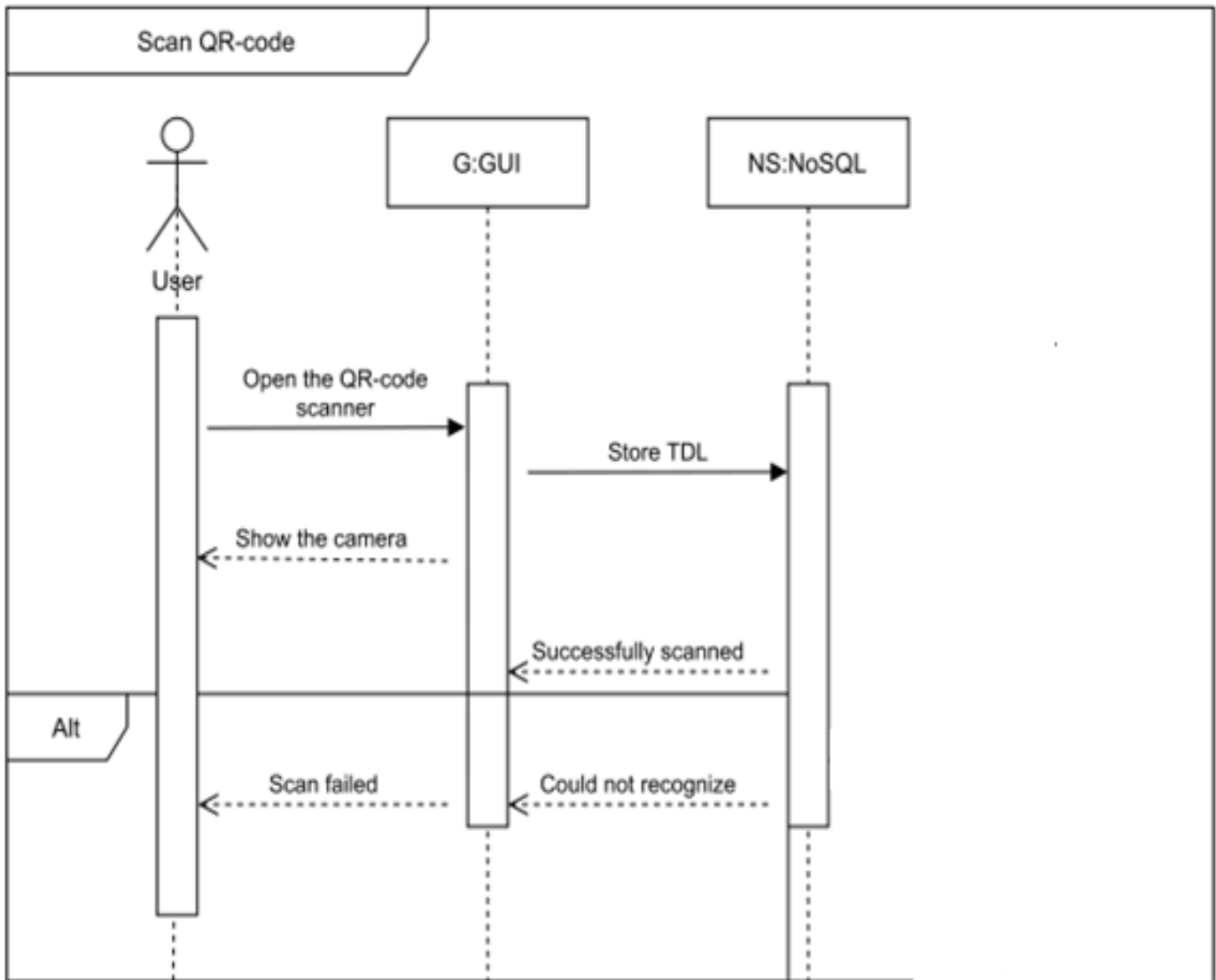


Figure 3.8: Sequence Diagram for Scan QR-Code

- **Figure 3.9:** This diagram shows how the user can logout from the application and how can the application timeout and kill the user session when needed.

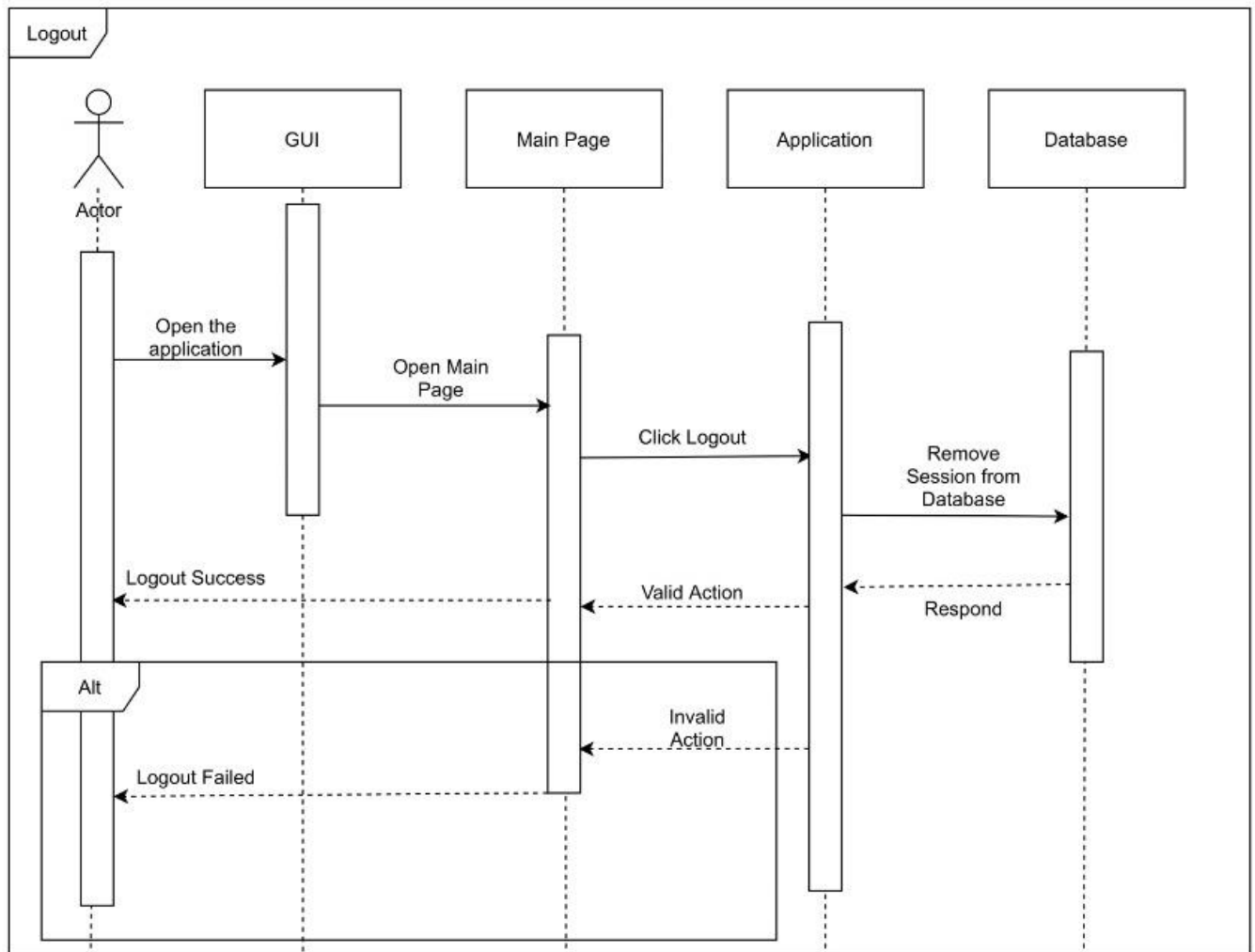


Figure 3.9: Sequence Diagram for Logout

3.7 Conclusion

We discussed the methodology type that we will used and analyzing the system requirement types to clarify the specification for the system moreover explain the main scenarios for the system.

4.1 Introduction

In this chapter, we will discuss in details the development tools, that are being used to implement the mobile application.

In addition, this chapter contains detailed explanation of the resulting interface supported by screenshots of all tasks accomplished by the application

4.2 Hardware and Software Specifications

4.2.1 Hardware Specifications

Item	Description
Computer device	For installing needed programs
Android / IOS mobile device	For testing the application
Internet connection	For downloading the needed programs

Table 4.1: Hardware Specification.

4.2.2 Software Specifications & Tools

Item	Description
Android studio	Android studio provides the fastest tools for building apps on every type of Android device
Android SDK	Android library used in developing application
Firebase	Firebase is a Backend-as-a-Service (BaaS). It provides developers with a variety of tools and services to help them develop quality apps. ^[12]
Flutter	Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. ^[13]
XCode	XCode is a MacOS app made by Apple for app development. It is the only officially supported way to develop iOS and other Apple OS apps. ^[14]

Table 4.2: Software Specification.

Main Program language:

“Dart” is a client-optimized programming language for creating apps on any platform. Its goal is to provide the most productive language for multi-platform development, along with a flexible execution runtime platform for app frameworks, it uses flutter to create user interfaces.^[15]

We choose to use Dart because:^[16]

1. It is flexible, we can create one code and run it on multi-platform
2. It has a wide range of libraries and frameworks that will help make our work easier
3. Uses flutter one of the famous mobile apps creating frameworks
4. Flutter leverages the use of widgets to create stunning UI
5. It is easier than learning multiple languages we can develop one app with one codebase and runs it on multi-platforms
6. Multiple tools IDE and text editors support it

Main Backend-as-a-Service:

“Firebase” is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps.

we are using Cloud Firestore it is a flexible, scalable database from Firebase and Google Cloud for mobile, web, and server development.

Firestore is a non-tabular database (also known as "not only SQL") store data differently than relational tables, Document, key-value, wide-column, and graph are the most common types. They have adaptable schemas and can handle large amounts of data and high user loads with ease.

In our Firestore Database we have two collection (**place_info**, **Visitors**) and each collection contains a set of documents, and each document contains a set of fields

Figure 4.1: shows a collection of “**places_info**” it contains five documents “five places” (city_walk , Jury_mall , , etc), the field contains all the information about this place.

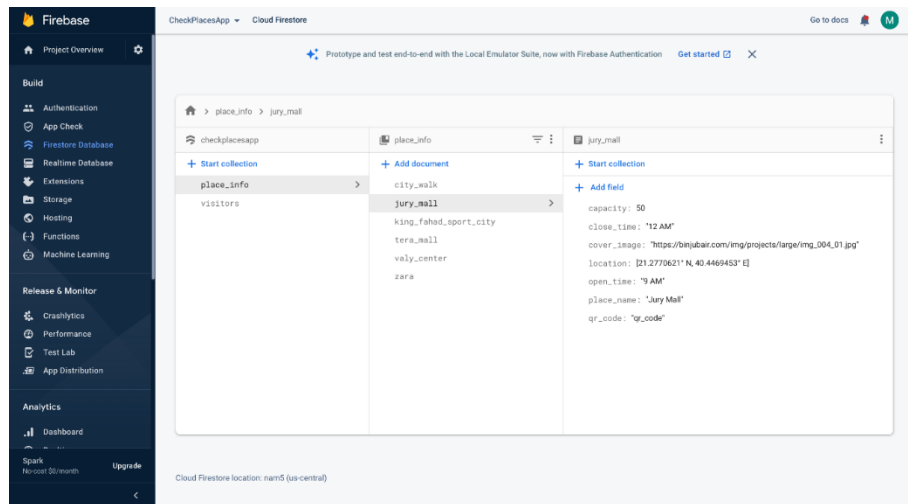


Figure 4.1: Collection (Place_Info).

Figure 4.2: shows a collection of “**Visitors**” it contains many documents each visitor has their own unique documents and document id, the field contains all the information about this place.

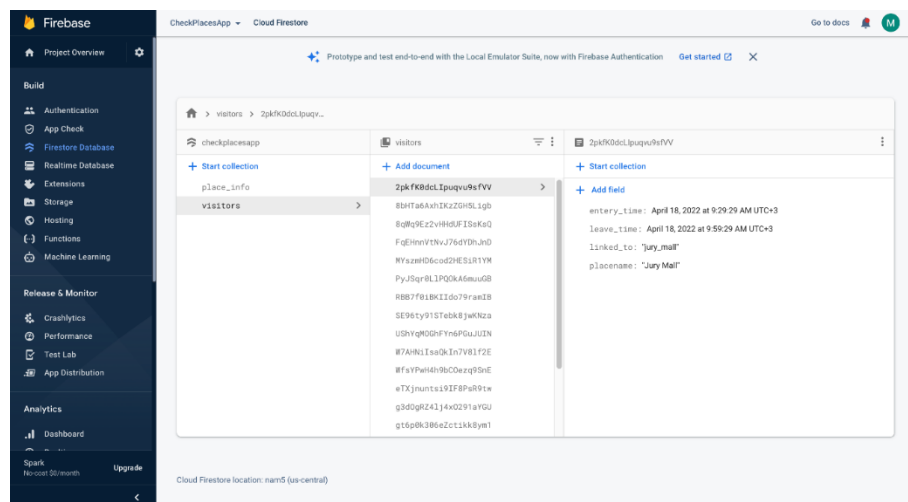


Figure 4.2: Collection (Visitors).

We choose to use Firebase because:^[17]

1. Hosting the application or website.
2. Instant synchronization between your software and the database.
3. API works on different platforms and platforms Android, iOS, Web.
4. Minimal effort from the programmer to get synchronization.
5. User authentication is easy and convenient via Firebase API.

4.3 Class Diagram

In this Fieger 4.3: In our Class Diagram there are five classes which are visitor, place, sub-place, user, and administrator:

- **Visitor:** it has 4 attributes, and it contains three methods and has two relationships, many to one, with place and sub-place.
- **Place:** it has 7 attributes, and it contains three methods and has three relationships, one to many, with the sub-place, visitors, and users.
- **Sub-place:** it has 1 attribute, sub-place it inherits all the attributes of the place class. Sub-place it contains one method and has two relationships one to many with visitor and user. And has one relationship many to one with place.
- **User:** it has 5 attributes, and it contains five methods and has two relationships, many to one, with place and sub-place.
- **Administrator:** has no attributes but it inherits all the attributes of the user class, and it contains five methods.

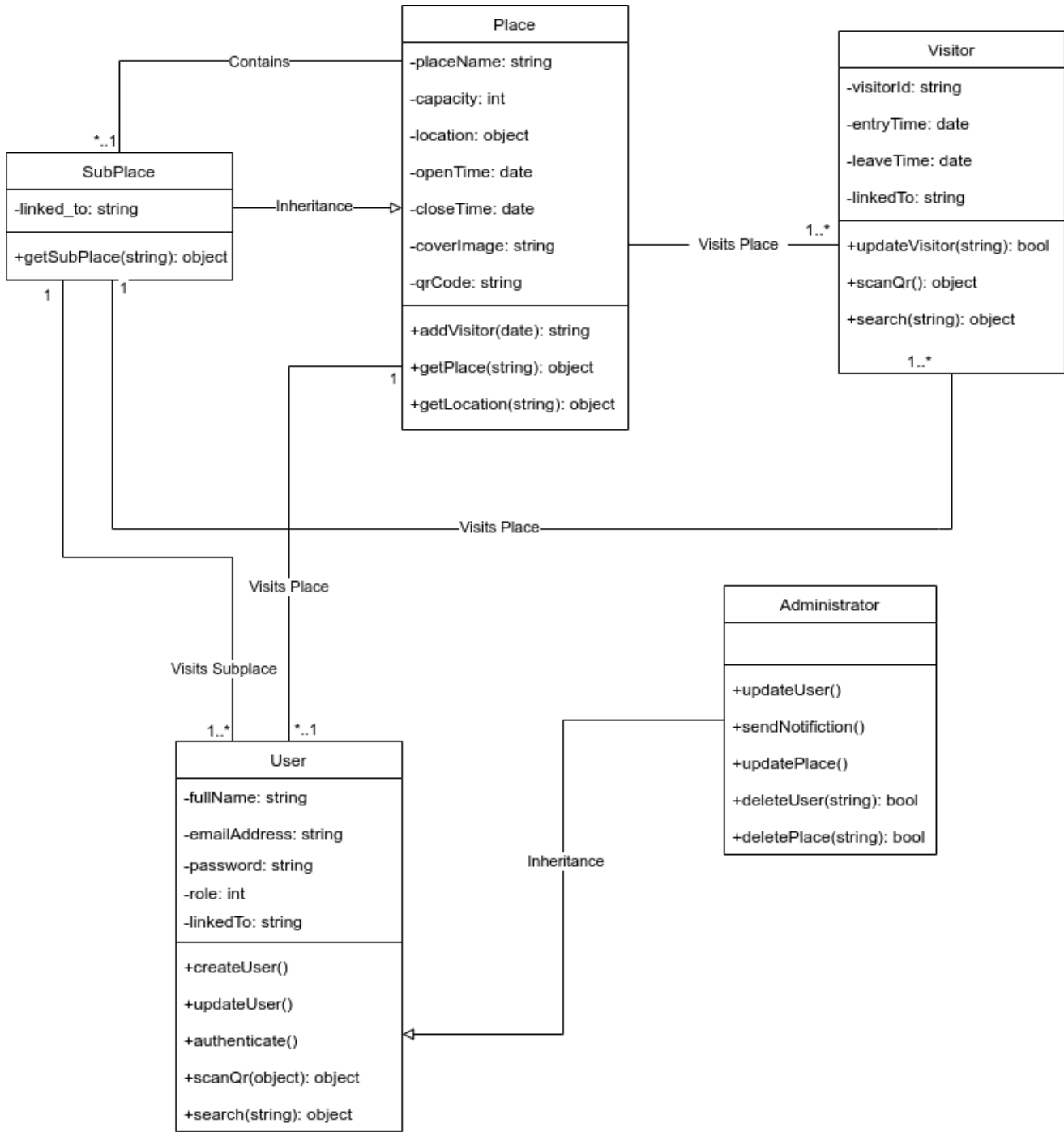


Figure 4.3: Class Diagram

4.4 ER Diagram

In this Figure 4.4: Because NoSQL instead of SQL we can't use ER Diagram so we implemented ER Diagram using class diagram.

We have three collocations that contains multiple fields:

- The first collocation is “place_info” contains nine fields and has one primary key and two foreign keys.
- The second collocation is “visitors” contains 4 fields and has one primary key. It has a relation many to one with place_info.
- The third collocation is “users” contains 4 fields and has one primary key. It has a relation many to one with place_info.

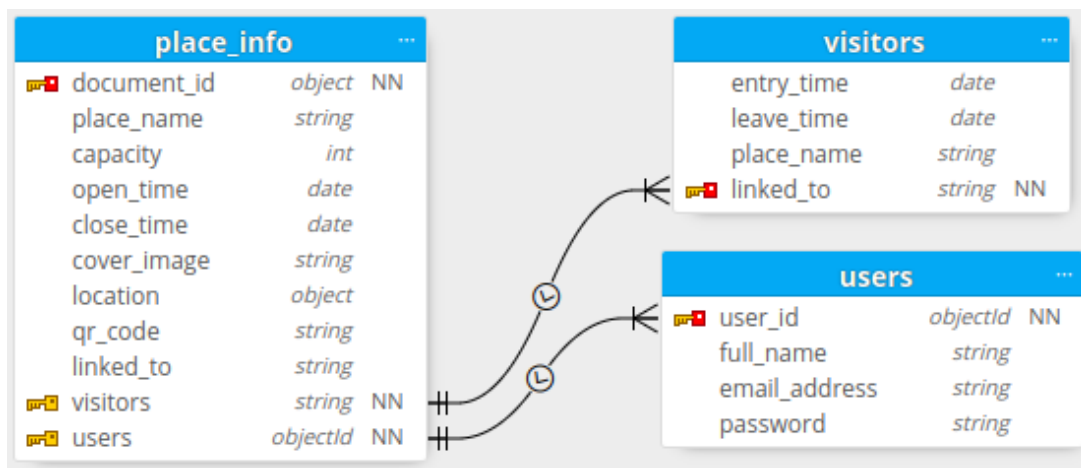


Figure 4.4: ER Diagram

4.5 Firebase Initialization

In this Figure 4.5: This code helped us link our application with Firebase Firestore service.

```
final FirebaseFirestore _firestore = FirebaseFirestore.instance;
```

Figure 4.5: Firebase Initialization Code.

4.6 Suggested Time Implementation

In this Figure 4.6&4.7: This is the function for getting the visitor suggested time. It's a basic function that collects the other visitors leaving time and then gets the average time for visitors and gives it to him.

This is a simple method until we collect enough data to make our code smart and we can convert our app to AI and ML.

```
Widget getSuggestedTime(List<int> leave) {
    if (leave.isNotEmpty) {
        return Text(
            "Suggested Time: ${DateFormat('hh:mm
a').format(DateTime.fromMicrosecondsSinceEpoch(leave.average.toInt()))}");
    } else {
        return Container();
    }
}
```

Figure 4.6: Suggested Time Code

```
List<QueryDocumentSnapshot> list = snapshot.data!.docs;

List<Visitors> v = [];

List<String> days = [];
List<String> addedDays = [];

for (var l in list) {
    String pn = l.get('linked_to');

    if (pn == place) {
        Timestamp time = l.get('entry_time');
        Timestamp? leave;

        days.add(DateFormat('EEE').format(time.toDate()));

        if(l.get('leave_time') != ''){
            leave = l.get('leave_time');
        }

        if(leave != null){
            leaveTime.add(leave.microsecondsSinceEpoch);
        }
    }
}
```

Figure 4.7: Leave Time Collocation Code

4.7 QR-Code Scanner Implementation

In this Figure 4.8: This code is the QR code scanner implementation; it reads the QR Content to retrieve the place name, then calls the database to add a new visitor using the place name, and finally triggers a notification schedule that runs every 10 minutes.

```
void _onQRViewCreated(QRViewController controller) {
  setState(() {
    this.controller = controller;
  });
  controller.scannedDataStream.listen((scanData) async {
    var data = scanData.code;
    Map<String, dynamic> user = {};

    if (data != null) {
      user = jsonDecode(data);
    }

    int count = await getCount(firestore, user['placename']);

    if (scanData.code != result?.code) {
      if (count < int.parse(user['visitorsNumber'])) {
        showModalBottomSheet(
          context: context,
          shape: const RoundedRectangleBorder(
            borderRadius:
              BorderRadius.vertical(top: Radius.circular(30))),
          builder: (context) =>
            Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text("Welcome to ${user['placename']}"),
                  const SizedBox(
                    height: 5,
                  ),
                  Text(
                    "You are Visitor number $count/${user['visitorsNumber']}",
                  ),
                  const SizedBox(
                    height: 10,
                  ),
                  ElevatedButton(
                    onPressed: () {
                      firestore.collection('visitors').add({
                        "placename": user['placename'],
                        "entry_time": Timestamp.now(),
                        "leave_time": '',
                        "linked_to": user['placename']
                          .toString()
                          .replaceAll(' ', '_')
                          .toLowerCase()
                      }).then((value) async {
                        NotificationApi.showScheduledNotification(
                          id: random.nextInt(500),
                          scheduled: DateTime.now()
                            .add(const Duration(minutes: 10)),
                          title: 'Are you out yet?',
                          body: 'Are you out yet?',
                          payload: value.id);
                      }).catchError((err) =>
                        showAlertDialog(
                          "Error",
                          err.toString(),
                          context,
                          () =>
                            Navigator.of(context,
                              rootNavigator: true)
                              .pop()));
                      Navigator.of(context).pop();
                    },
                    child: const Text("OK"))
                ],
              ),
            ),
          ),
        );
      result = scanData;
    }
  });
}
```

Figure 4.8: QR-Code Scanner Code

4.8 Application Interface

The application has two scenarios:

1. The first scene gives the ability the visitor to show the place location information and know the suitable time to visit the place and plan for the activity
2. The second scene the visitor has the ability to scan the QR-code to check in to the place location and also, we set a reminder for the visitor to know if the visitor still in the place or not and the reason of that to know the current number of the visitors to help us identify the peak time in the places, providing appropriate dynamic visualizations that reflect the real visitors in the places

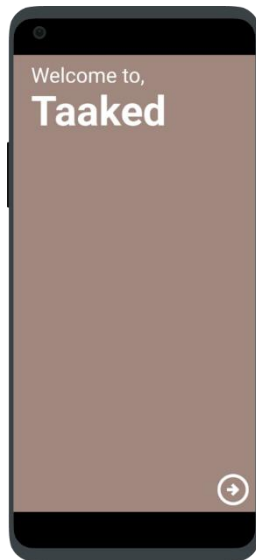


Figure 4.9: Start Interface

First scene

In this screen the visitor can create an account by entering the name, email, and password user can open an account easily to use our futures

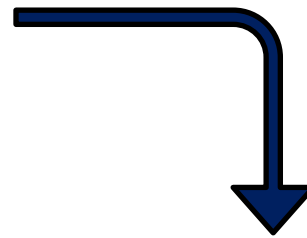
The image shows a smartphone screen displaying the registration interface of the 'Crowd Management App'. The app's title bar is brown with a back arrow, a camera icon, and the text 'Crowd Management App'. Below the title bar, there is a light gray rounded rectangle containing three white input fields with brown borders. The first field is labeled 'Enter your full name', the second 'Enter your email address', and the third 'Enter your password'. Below these fields are two brown buttons with white text: 'Register' and 'Login to your account'. The phone has a black bezel and a home indicator bar at the bottom.

Figure 4.10: Registration Interface

In this screen the visitor can login to the system by entering the correct email address and password the system will check it against the database and if not mandatory activity to use the application visitor can use it to scan the QR-code and check in to the place.

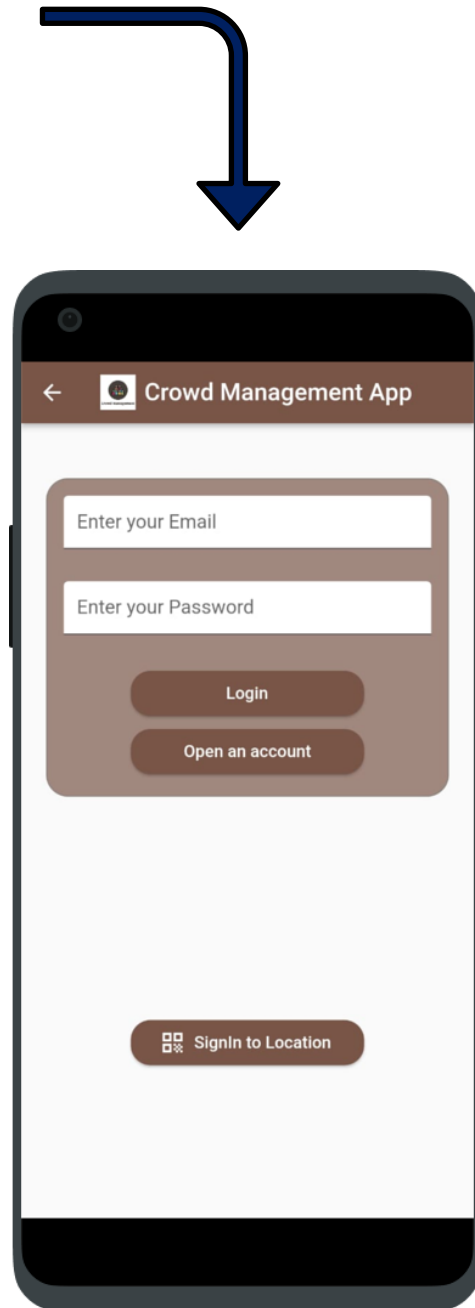


Figure 4.11: Login Interface

In this screen the visitor can search for a specific place to visit.

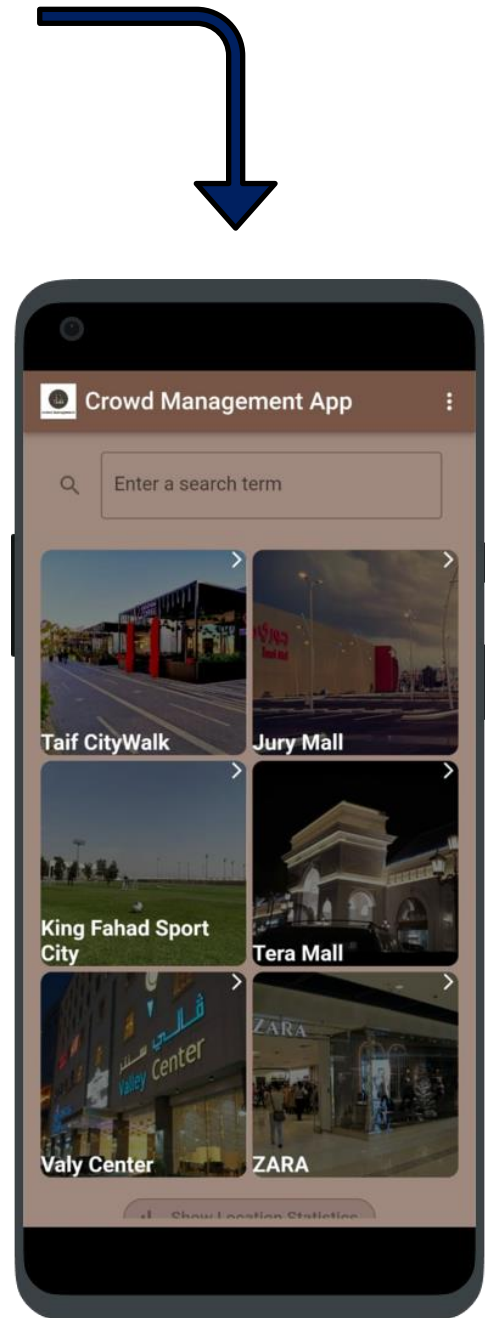


Figure 4.12: Home Interface

In this screen the visitor can choose the suitable time and date to plan and visit the location that have been chosen before and check the peak time for the place.

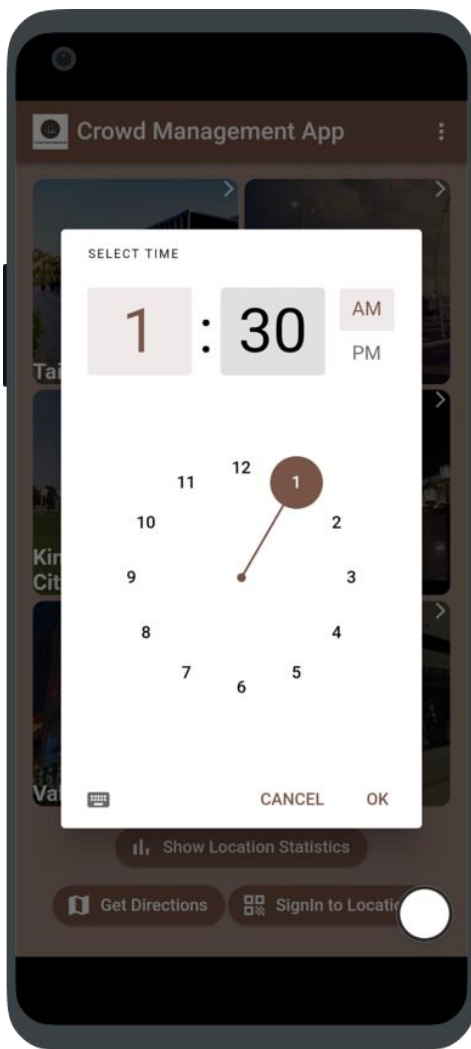


Figure 4.13: Time Interface

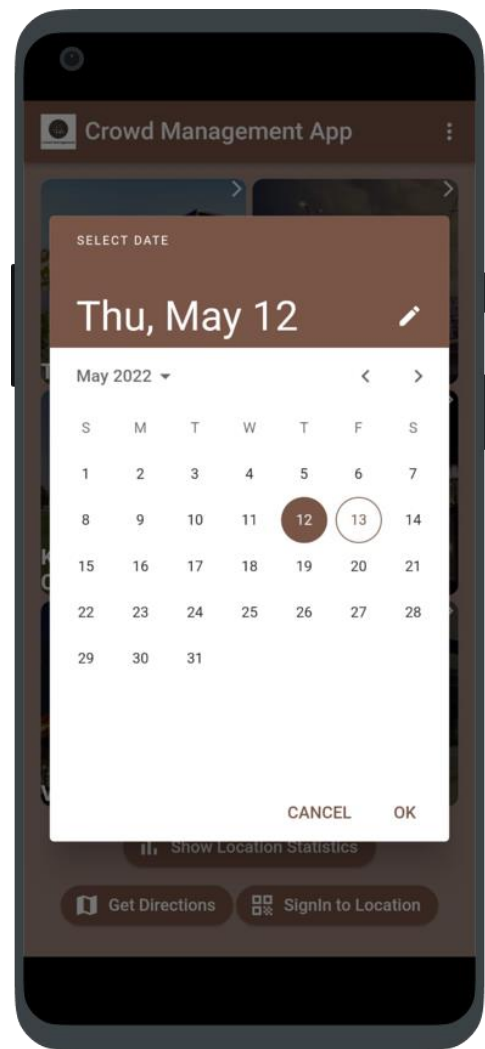


Figure 4.14: Date Interface

In this screen, the visitor can check the peak time and give the suggested time depending on the chosen time by the visitor and show visualizations, in the first chart above presents the current number of visitors for the main chosen place, and the second chart down presents the current number of visitors for the sub-place chosen from the main place

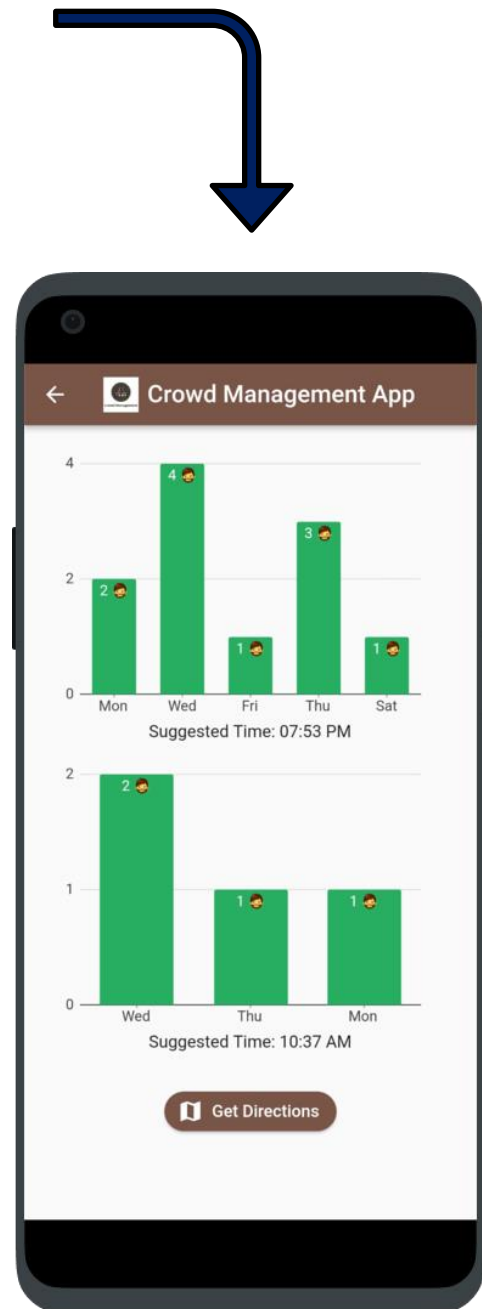


Figure 4.15: Bar Chart Interface

In this screen the visitor can reach to the direction of the chosen place to visit using Google Maps application.



Figure 4.16: Maps Interface

Second scene

In this screen the visitor can scan the QR-code and check in to the place. We use QR-code to collect data for our database to use it later.

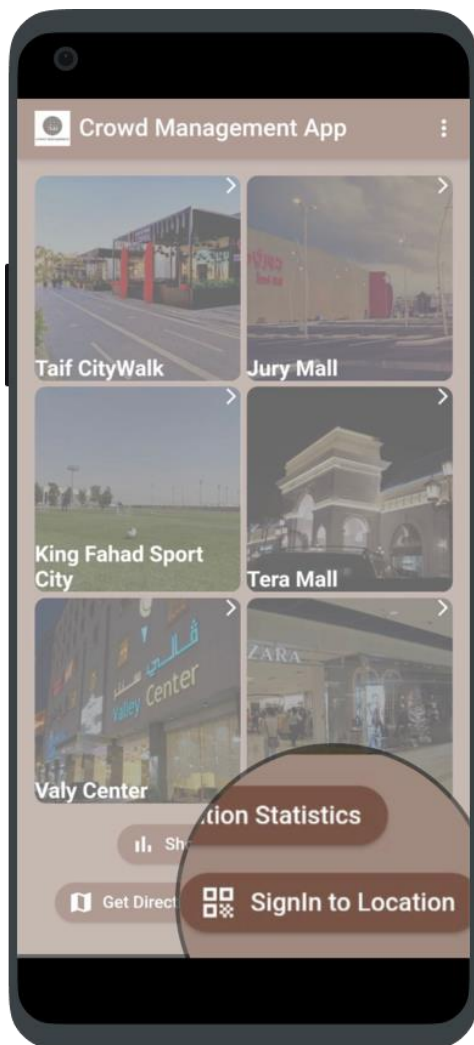


Figure 4.17: Sign-In to Location

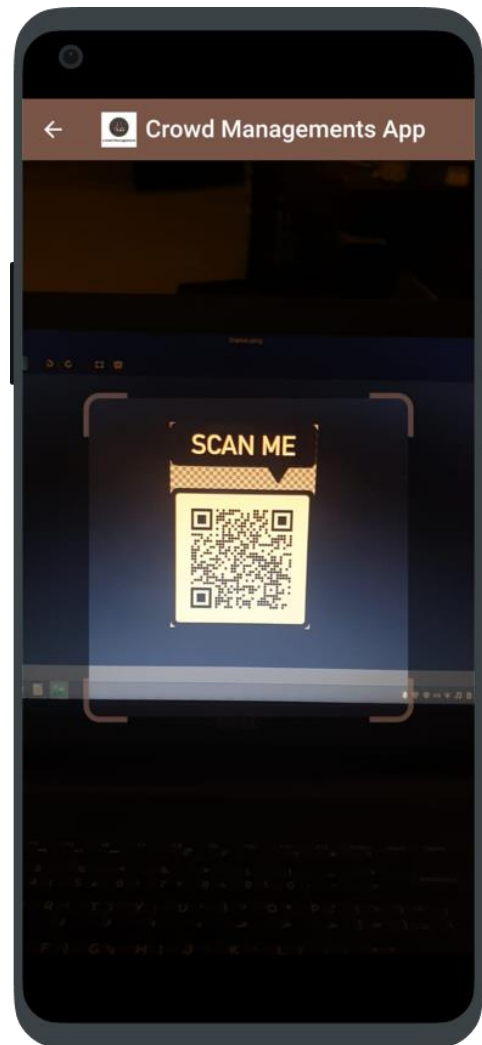


Figure 4.18: Scan QR-Code

In this screen the visitor will receive a notification after scanning the QR-code successfully and the app will show the current number of the visitors in the place.

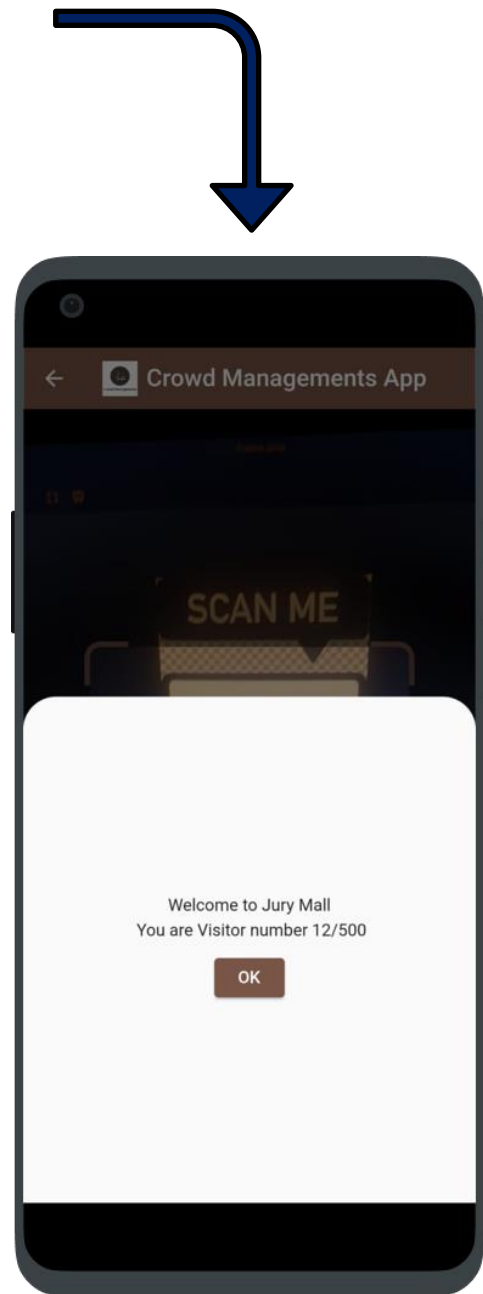


Figure 4.19: Successfully Entry

4.9 Test Case

Overview on Testing

- **Unit Testing**

The technique of testing program components such as methods or object classes is known as unit testing. The simplest sort of component is a single function or method. These methods should be called with different input parameters in your tests.

- **Component/Widget Testing**

Software components are often composite components composed of multiple interacting objects. The reconfiguration component, for example, in the weather station system includes objects that deal with each aspect of the reconfiguration. These objects' functionality is accessed via the defined component interface. As a result, testing composite components should concentrate on demonstrating that the component interface behaves as specified. You can presume that unit tests on the component's individual objects have been completed.

- **The Approach We Used**

We used component testing in our code to test the widgets of our application and we used unit testing to test some of application functionality. Our testing was white box, which is a method of program testing in which the tests are based on knowledge of the program's structure and components. White-box testing requires access to source code.

Login Page Test

In this Figure 4.20: This test case is for our Login screen it tries to search for the Login button if it finds then the screen works if it does not then the screen does not work.



```
testWidgets('Login Page Test', (WidgetTester tester) async {  
  Widget testWidget = const MediaQuery(  
    data: MediaQueryData(),  
    child: MaterialApp(home: LoginScreen())  
  );  
  
  await tester.pumpWidget(testWidget);  
  
  expect(find.text("Login"), findsWidgets);  
});
```

Figure 4.20: Login Page Test

Register Page Test

In this Figure 4.21: This test is for our register screen it tries to search for Register button if the widget found then the test works if it doesn't than it does not work.

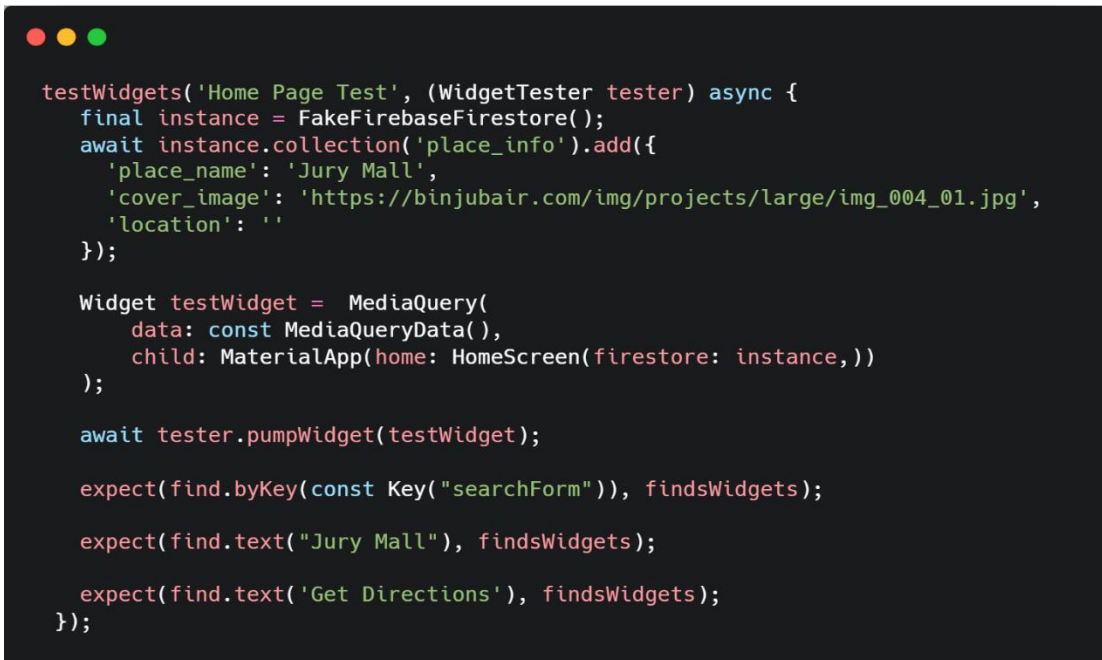


```
testWidgets('Register Page Test', (WidgetTester tester) async {  
  Widget testWidget = const MediaQuery(  
    data: MediaQueryData(),  
    child: MaterialApp(home: RegisterScreen())  
  );  
  
  await tester.pumpWidget(testWidget);  
  
  expect(find.text("Register"), findsWidgets);  
});
```

Figure 4.21: Register Page Test

Home Page Test

In this Figure 4.22: This the first test case it tests our home screen it tries to find 3 widgets [Search Form, Places Widgets, and Get Location Button] if this test case succeeds than our home page works if it doesn't then it does not work.

A screenshot of a code editor with a dark background and light-colored text. At the top left, there are three colored circles (red, yellow, green) representing window controls. The code is a Flutter widget test for the Home Page. It starts with a `testWidgets` function call, followed by a `WidgetTester` parameter and an `async` block. Inside the block, it creates a `FakeFirestore` instance, adds a document to the `place_info` collection, and then creates a `MediaQuery` widget with a `MaterialApp` child. The `MaterialApp` has a `home` property set to `HomeScreen`. The test then pumps the widget and uses `expect` to find three widgets: a search form, the text 'Jury Mall', and a 'Get Directions' button.

```
testWidgets('Home Page Test', (WidgetTester tester) async {  
  final instance = FakeFirestoreFirestore();  
  await instance.collection('place_info').add({  
    'place_name': 'Jury Mall',  
    'cover_image': 'https://binjubair.com/img/projects/large/img_004_01.jpg',  
    'location': ''  
  });  
  
  Widget testWidget = MediaQuery(  
    data: const MediaQueryData(),  
    child: MaterialApp(home: HomeScreen(firestore: instance,))  
  );  
  
  await tester.pumpWidget(testWidget);  
  
  expect(find.byKey(const Key("searchForm")), findsWidgets);  
  
  expect(find.text("Jury Mall"), findsWidgets);  
  
  expect(find.text('Get Directions'), findsWidgets);  
});
```

Figure 4.22: Home Page Test

Camera Page Test

In this Figure 4.23: This test is for the camera screen it tries to search for the Camera widget using "key" if it finds it then the test works if it doesn't then it does not work.

A screenshot of a code editor with a dark background and light-colored text. At the top left, there are three colored circles (red, yellow, green) representing window controls. The code is a Flutter widget test for the Camera Page. It starts with a `testWidgets` function call, followed by a `WidgetTester` parameter and an `async` block. Inside the block, it creates a `MediaQuery` widget with a `MaterialApp` child. The `MaterialApp` has a `home` property set to `CameraScreen`. The test then pumps the widget and uses `expect` to find a widget with the key 'Camera'.

```
testWidgets('Camera Page Test', (WidgetTester tester) async {  
  Widget testWidget = const MediaQuery(  
    data: MediaQueryData(),  
    child: MaterialApp(home: CameraScreen())  
  );  
  
  await tester.pumpWidget(testWidget);  
  
  expect(find.byKey(const Key('Camera')), findsWidgets);  
});
```

Figure 4.23: Camera Page Test

Checkout Page Test

In this Figure 4.24: This a test for the checkout page it tries to find the Checkout message if it finds it then the screen works if it doesn't then it does not work.

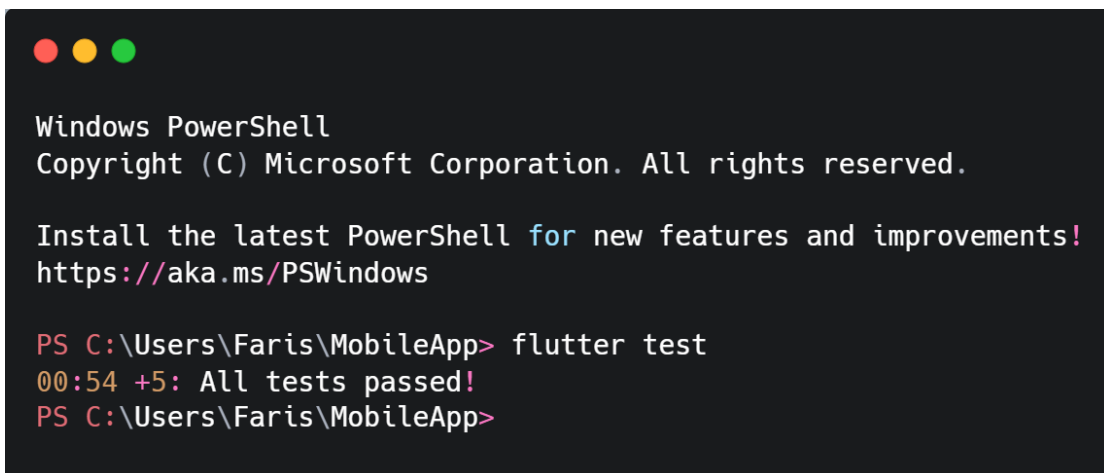
A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in a light-colored font and represents a Flutter test for the checkout page. It uses the 'testWidgets' function to create a test widget, pump it, and then expect a specific text to be found on the screen.

```
testWidgets('Checkout Page Test', (WidgetTester tester) async {  
  Widget testWidget = const MediaQuery(  
    data: MediaQueryData(),  
    child: MaterialApp(home: CheckoutScreen())  
  );  
  
  await tester.pumpWidget(testWidget);  
  
  expect(find.text("Are You Out Yet?"), findsWidgets);  
});
```

Figure 4.24: Checkout Page Test

Test Result

In this Figure 4.25: We found after running all test cases that all of them passed which means that our system works without issues, and it shows that our test cases are correct.

A screenshot of a Windows PowerShell terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows the output of running 'flutter test', which indicates that all tests passed successfully.

```
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Install the latest PowerShell for new features and improvements!  
https://aka.ms/PSWindows  
  
PS C:\Users\Faris\MobileApp> flutter test  
00:54 +5: All tests passed!  
PS C:\Users\Faris\MobileApp>
```

Figure 4.25: Test Result

4.10 Future Work

- Make the reminder notification dynamic depending on the place because some places have different average time than others.
- Developing machine learning algorithm to predict the crowds in a specific place.
- We will add influential events such as conferences and similar events that are close to the places that visitors can visit and affect the numbers and peak times and make them clear to the visitor.
- Try to add multiple charts and graphs depending on the multiple data types.

4.11 Conclusion

Crowd management is one of the problems that researchers have always been striving to find innovative solutions using technologies. In this report, our solution focusing on managing and facilitate the access also to know the peak times in public places in addition to contributing to reducing crowding and aiding decision-making, furthermore, talked about a brief review of similar systems and studies and the method that will be to develop the system and the tools used, in addition describing the application scenarios and the interfaces of the application in detail. We are planning to improve and add our future work ideas to enhance the application efficiency.

Reference

- [1] *What is crowd management?* (n.d.). Retrieved December 10, 2021, from <https://www.heras-mobile.com/crowdmanagement/wat-is-crowd-management>
- [2] *Hajj crush: how crowd disasters happen, and how they can be avoided* / Hajj / *The Guardian*. (n.d.). Retrieved November 14, 2021, from <https://www.theguardian.com/world/2015/oct/03/hajj-crush-how-crowd-disasters-happen-and-how-they-can-be-avoided>
- [3] Team, L. (2020, February). crowdlessapp. Retrieved from crowdless: <https://crowdlessapp.co/>
- [4] Skip The Crowds - Search businesses, avoid crowds. (2020, October 16). Retrieved from <https://play.google.com/store/apps/details?id=com.corewest.skipthecrowds>
- [5] Durán-Polanco, L., & Siller, M. (2021). Crowd management COVID-19. *Annual Reviews in Control*. <https://doi.org/10.1016/J.ARCONTROL.2021.04.006>
- [6] Mitchell, R. O., Rashid, H., Dawood, F., & Alkhalidi, A. (2013). Hajj crowd management and navigation system: People tracking and location based services via integrated mobile and RFID systems. *International Conference on Computer Applications Technology, ICCAT 2013*. <https://doi.org/10.1109/ICCAT.2013.6522008>
- [7] Arlow, J. and Neustadt, I. (2005). *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design* (2nd Edition). Boston: Addison-Wesley.
- [8] Petersen, Kai; Wohlin, Claes; Baca, Dejan (2009). Bomarius, Frank; Oivo, Markku; Jaring, Päivi; Abrahamsson, Pekka (eds.). "The Waterfall Model in Large-Scale Development". *Product-Focused Software Process Improvement. Lecture Notes in Business Information Processing*. Berlin, Heidelberg: Springer: 386–400. doi:10.1007/978-3-642-02152-7_29. ISBN 978-3-642-02152-7.
- [9] Benington, Herbert D. (1 October 1983). "Production of Large Computer Programs" (PDF). *IEEE Annals of the History of Computing*. IEEE Educational Activities Department. 5 (4): 350–361. doi:10.1109/MAHC.1983.10102. S2CID 8632276. Retrieved 2011-03-21. Archived July 18, 2011, at the Wayback Machine
- [10] Sommerville, I., Columbus, B., New, I., San, Y., Upper, F., River, S., Cape, A., Dubai, T., Madrid, L., Munich, M., Montreal, P., Delhi, T., São, M. C., Sydney, P., Kong, H., Singapore, S., & Tokyo, T. (2011). *SOFTWARE ENGINEERING Ninth Edition*.

- [11] Kotonya, Gerald; Sommerville, Ian (1998). Requirements Engineering: Processes and Techniques. Chichester, UK: John Wiley and Sons. ISBN 9780471972082.
- [12] What is Firebase(n.d.). Retrieved April 14, 2022, from <https://www.educative.io/edpresso/what-is-firebase>
- [13] Flutter Retrieved April 14, 2022, from <https://flutter.dev/>
- [14] What is Xcode and why do I need it - Zero To App Store. Retrieved April 14, 2022, from <https://www.zerotoappstore.com/what-is-xcode-and-why-do-i-need-it.html>
- [15] *Dart overview / Dart.* (n.d.). Retrieved December 10, 2021, from <https://dart.dev/overview>
- [16] *10 good reasons to learn Dart. If you already know C++, C#, or Java... | by Nafis Fuad | HackerNoon.com / Medium.* (n.d.). Retrieved December 10, 2021, from <https://medium.com/hackernoon/10-good-reasons-why-you-should-learn-dart-4b257708a332>
- [17] Crowe, T. (2022, may 11). what's new from Firebase at Google I/O 2022. Retrieved from Firebase: <https://firebase.blog/posts/2022/05/whats-new-at-google-io>

Appendix

- <https://github.com/farisc0de/MobileApp>

