



**MAJMAAH UNIVERSITY  
COLLEGE OF SCIENCE  
DEPT. OF COMPUTER SCIENCE & INFORMATION**

# **Pilgrim Tracing System**

**BY**

Hessa Jasir ALshalani  
351205118

**Supervised by**

Dr. Sarah Mustafa Eljack

**A REPORT SUBMITTED TO  
UNIVERSITY OF MAJAMAAH**

**In partial fulfillment of the requirements  
For the degree of  
BACHELOR OF COMPUTER AND INFORMATION SCIENCE**

1439-1440 AH

## **Abstract:**

Pilgrim tracing system it is an Android application for the pilgrim and supervisor campaign

Making it easy for them to know each other's place.

a survey was conducted and its results were many including the loss of pilgrim in Hajj and do not know Mansak places.

This application facilitates communication with the supervisor, knowing each other's place and Tracing each other and can search for the Mansak places.

Keyword: pilgrim, track, loss, place.

## **مقدمه:**

نظام تتبع الحاج تطبيق اندرويد للحاج ومشرف الحمله في الحج. التطبيق يقوم بعده خدمات. يمكن للمشرف معرفه اماكن الحجاج و ارشادهم للطريق الصحيح او الذهاب له ويمكن للحاج ايضاً معرفه مكان المشرف.

كما يمكن للحاج الاتصال المباشر بالمشرف اذا استدعى الامر.

يمكن للحاج البحث عن مكان منسك محدد عن طريق الخريطة داخل التطبيق.

**Dedication:**

To the light that illuminates my path of success ... My Father

To who taught me to endure no matter how the circumstances change... My Mother

To all who taught me and illuminated the way for me

**Acknowledgment:**

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank D. Sara Mustafa Eljack

To provide all the support and guidance that made me complete the project as required. I am grateful to her for her support and guidance.

**MAJMAAH UNIVERSITY,  
COLLEGE OF SCIENCE AL ZULFI,  
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION**

**(CERTIFICATE BY STUDENT)**

This is to certify that the project titled “**Pilgrim Tracing System**” submitted by me  
(**Hessa jaser AlShalani, 351205118**) under the supervision of **Dr.Sarah M. Eljack** for  
award of Bachelor degree of the Majmaah University carried out during the Semester  
1, 2018-19 embodies my original work.

Signature in full: -----

Name in block letters: **Hessa Jasir ALShalani**

Student ID: 351205118

Date: 21/11/2018

## Table of contents:

Abstract .....	i
Dedication .....	ii
Acknowledgment .....	iii
Certificate .....	iv
Table of Content.....	v
List of figure.....	vii
List of Table .....	ix
1 introduction:	
1.1 Introduction .....	1
1.1.1 Abstract system description .....	2
1.2 problem definition .....	2
1.2.1 Goals .....	3
1.2.2 Objectives .....	3
1.2.3 Critical success factors .....	3
1.2.4 Organization chart and responsibilities .....	4
1.3 General rules .....	4
1.4 literature survey.....	5
1.5 feasibility study .....	9
2 System analysis:	
2.1.1 Introduction .....	11
2.1.2 description of DFD .....	11
2.2.1 Context diagram .....	12
2.2.2 DFD level0 .....	13
2.2.3 Detailed DFD .....	14
2.3 Entity Relationship Diagram.....	15
2.3.1 Description of Entities .....	15

2.3.2 Description of Relation .....	15
2.3.3 ERD .....	16
2.4 Class Diagram .....	17
2.5 Object Diagram .....	18
2.6 Use Case diagram .....	19
2.7 Activity diagram .....	20
2.8 Sequence Diagram .....	22
2.9 State diagram .....	23
3 System design:	
3.1 Description of procedures and function :.....	24
3.2 Relation database schema :.....	25
3.3 Hardware and Software requirement:.....	26
3.4 Initial Interfaces: .....	28
4 Implementation and Testing	
4.1 Introduction:.....	33
4.2 Procedure:.....	33
4.3 Layout:.....	37
4.4 Report Layout:.....	39
4.5 Report:.....	55
4 conclusion:.....	59
Reference .....	60
Appendix A .....	61
Appendix B .....	66

## List of figure:

Figure (1.2.4.1) General element in system .....	4
Figure (1.2.4.2) Function of element .....	4
Figure (2.2.1) Context diagram .....	12
Figure (2.2.2) DFD level0 .....	13
Figure (2.2.3) Detailed of DFD .....	14
Figure (2.3.3) ER diagram .....	16
Figure (2.4) Class diagram .....	17
Figure (2.5) Object diagram .....	18
Figure (2.6) Use Case diagram .....	19
Figure (2.7.1) Supervisor Activity diagram .....	20
Figure (2.7.2) Pilgrim Activity diagram .....	21
Figure (2.8) Sequence diagram .....	22
Figure (2.9) State diagram .....	23
Figure(3.4.1) Registration interface .....	28
Figure (3.4.2) Login interface .....	29
Figure (3.4.3) Admin interface .....	30
Figure (3.4.4) Supervisor interface .....	31
Figure (3.4.5) Pilgrim interface .....	32
Figure (4.3.1) sign up interface.....	37
Figure (4.3.2) sign in interface .....	37
Figure(4.3.3) supervisor interface.....	37
Figure(4.3.4) Admin interface.....	37
Figure(4.3.5)Pilgrim interface.....	38
Figure(4.3.6)pilgrim's location.....	38
Figure(4.3.7)Supervisor location.....	38
Figure(4.4.1) sign in interface.....	39
Figure(4.4.2) sign in Arabic language.....	40
Figure(4.4.3) sign up interface .....	41



Figure(4.4.4) Admin interface .....	42
Figure(4.4.5) Admin Add Supervisor.....	43
Figure(4.4.6) Supervisor interface.....	44
Figure(4.4.7) Supervisor Add pilgrims.....	45
Figure(4.4.8) Supervisor Add Event.....	46
Figure(4.4.9) Supervisor Delete pilgrims.....	47
Figure(4.4.10) Supervisor Modify pilgrim.....	48
Figure(4.4.11) Supervisor Show pilgrims location .....	49
Figure(4.4.12) Pilgrim interface.....	50
Figure(4.4.13) Pilgrim Show events.....	51
Figure(4.4.14) Macca places.....	52
Figure(4.4.15) Supervisor location.....	53
Figure(4.4.16) Pilgrim Call supervisor.....	54
Figure (4.5.1) Pilgrim Tracing System.....	55
Figure (4.5.2) Admin-Supervisor database.....	55
Figure (4.5.3) Supervisor-Pilgrim database.....	56
Figure (4.5.4) User database.....	56
Figure (4.5.5) Event database.....	57
Figure (4.5.6) Location database.....	57

**List of Table:**

Table (1.4.4) Comparison table .....8

Table (3.2.1) Users database .....25

Table (3.2.2) Event database .....25

Table (3.2.3) Location database .....25

Table (3.3.1) Hardware requirement .....26

Table(4.5.1) User database .....58

Table(4.5.2) Event database.....58

Table(4.5.3) Location database .....58

## **chapter1: INTRODUCTION**

### **1.1 Overview**

Many millions of Muslims come to Mecca in Saudi Arabia to perform pilgrim (Hajj) every year from all around the world. They used to dress white clothes that represent the real of human equality. Young and old ages, poor and rich people, and famous and simple people all together come to this holy place seeking the forgiveness from ALLAH. The Hajj is one of the five pillars in Islam. It is a must for each Muslim if he/her can afford the physical and financial ability.

Muslims follow the Islamic lunar calendar. However, the month of Hajj represents the 12th month of the Islamic lunar year and is called 'Dhul-Hijjah'. Early in Hajj month, Muslims start coming and gathering in Mecca preparing for this great event. Hajj main rites take place from 8th to 12th days of 'Dhul-Hijjah'.

The Prophet Muhammad, peace be upon him, said that the first ten days of 'Dhul-Hijjah' represent a special time for devotion. In these days, preparations are on the run for pilgrims undertaking the pilgrimage were the real pilgrimage rites happen. The day of 'ARAFAT' which is the day number 9 in the month shows the great of Hajj by gathering all millions of pilgrims on ARAFAT mounting seeking the forgiveness from God (ALLAH). The mount of ARAFAT represents the place where our prophet, peace be upon him, gave his farewell sermon. This day is followed by day number 10 which represents 'Eid al-Adha' day where all Muslims around the world celebrate this festival. All places in Mecca were equipped with all needed facility to control and monitor Hajj process. In addition, the parties in charge are ready to introduce the assistance to pilgrims and to guide them step by step to accomplish this holy journey. (Al-Akhras.A.2017)

The current provided services and all these developed facilities in Mecca that facilitates the performing of pilgrim are very appreciated. Still, as the number of pilgrims is huge, different languages, variety of ages, cultures and needs, each pilgrim might need an assistance to keep him safe and to guide him/ her in performing Hajj rites. Thereby, there is a need to develop a communication system that facilitates the communication between the pilgrim and the campaign manager to handle pilgrim's lost and to introduce the needed help very fast.

### **1.1.1 Abstract system description:**

Pilgrim's application eases the communication with campaign manager, the application provides a variety of services. If the pilgrim wants to go to the place of the next Mansak, the pilgrim simply opens the application and looks to the map that provides him/ her with the details about any location inside Mecca. If the pilgrim has lost the road, both pilgrim and campaign supervisor can open the map inside the application and locate the location of each other which can be utilized for bringing back the lost pilgrims. The application provides a direct call feature to campaign Supervisor in case the pilgrim could not be able to use the map. and supervisor Add events about this day and sent to pilgrims.

### **1.2 Problem Statement:**

The pilgrims come from different places with many languages to Mecca to perform Hajj. As most of them do not know the exact rites locations, the time the rite starts and ends, the movement between rites places and the inability to contact their campaign manager in case of lost or sick, many problems may occur. The most difficult case is the loss of pilgrim between this huge number of pilgrims and the inability to communicate with his/ her campaign manager. This study seeks the implementation of mobile applications that overcome the mentioned problems and facilitates the process of Hajj.

### **1.2.1 Goals:**

This project is building to

- To facilitate the communication between the pilgrim and the campaign manager
- To ease the determination of pilgrim location and the location of his/ her campaign manager
- To know the place of the next rite and to provide details about that rite
- To provide a mobile application that address all issues related to pilgrims

### **1.2.2 Objectives:**

- Minimize the loss of pilgrim
- Finding the lost pilgrim quickly
- Assist campaign in finding the lost pilgrim
- Direct the pilgrims to the rites in an optimal way

### **1.2.3 Critical success factors:**

- Each pilgrim has a unique identifier to ease the process of tracking on the map
- Each pilgrim and campaign manager must have internet connection

### 1-2-4 Organization chart and responsibilities:

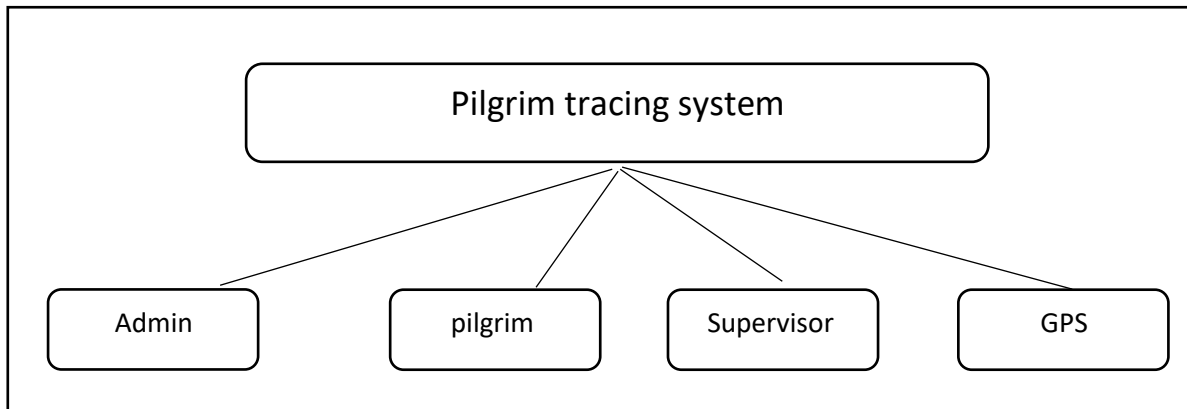


Figure1.2.4.1: general element in system

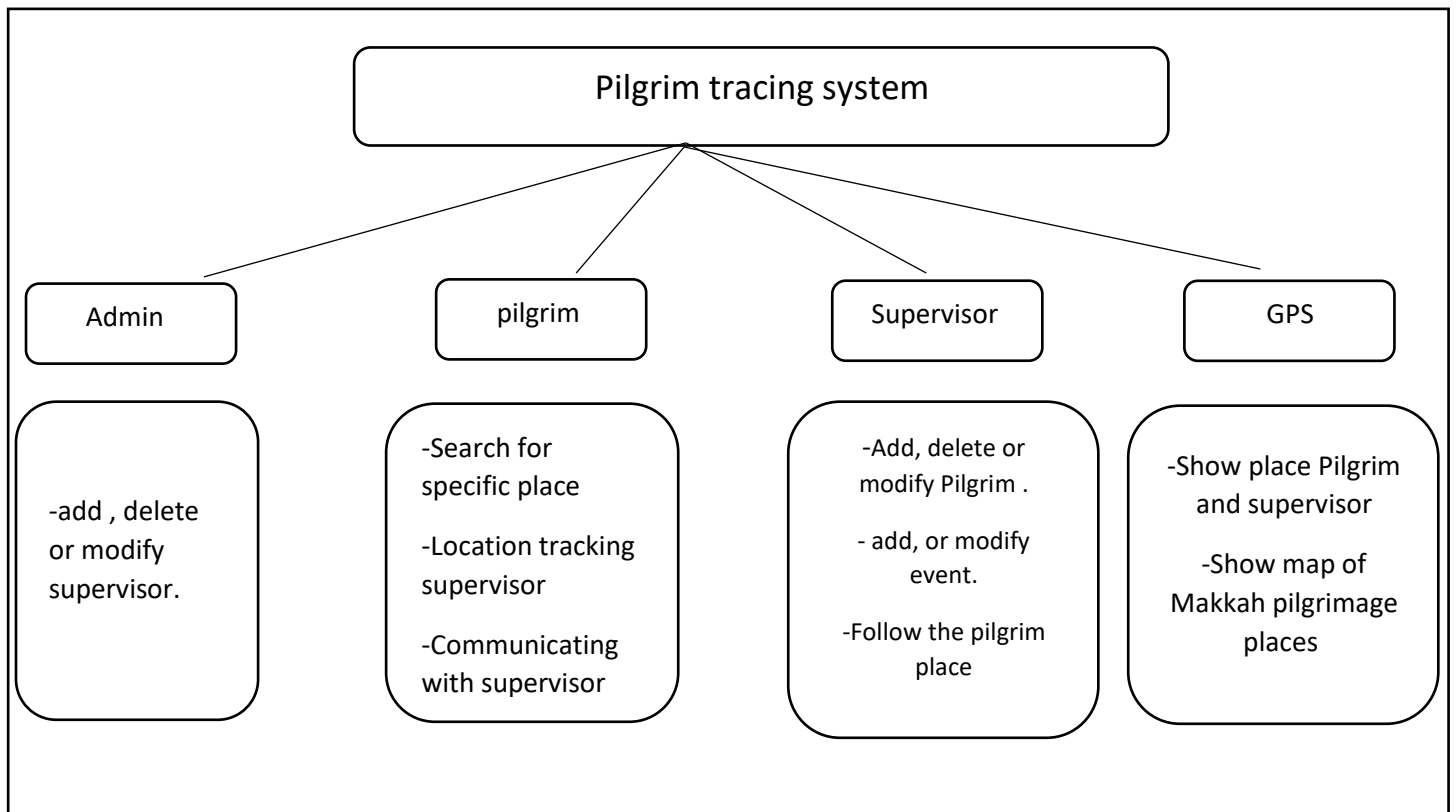


Figure1.2.4.2: function of elements

### 1.3 General rules (assumptions):

- Each campaign should have a secret-unique ID that passed to pilgrims by their campaign manager.
- Each pilgrim and campaign manager must have internet connection
- Android Operating System

## 1.4: literature survey:

- 1-Manasikana
- 2-Familo locator
- 3-space time

### 1.4.1 Manasikana:



Relied As: <https://itunes.apple.com/us/app/manasikana/id1119024075?mt=8>

#### **-simple Description:**

-

An application that offers Hajj services in seven languages, application on maps of the places Hajj, the direction of the Kaaba, and your current location, and provides the important numbers and emergency numbers that Haj may need.

#### **-Advantages:**

- Easy to use
- Multilanguage
- Maps provide all important places in Hajj.

#### **- problems:**

- there is no connection between the pilgrim and her group.

### 1.4.2 familo locator:



Relied As : <https://www.familo.net/ar/index.html>

#### **-Simple Description:**

Application to track and locate your kids and family by locating them using a GPS tracker, and communicating with them directly through the application.

#### **-Advantages:**

-It can be used in the event of an emergency and this through the provision of emergency button in the application works to determine the whereabouts and send information to the second party.

- provides more safety for parents.

#### **-problems:**

-The person can decide to send your place or not.



### 1.4.3 Space Time:



Relied As: <https://itunes.apple.com/us/app/space-time/id508723489?mt=8>

#### **-Simple Description:**

Application to watch friends and family by sending a message to the person (where is your place Click here to enter your place) If you press the button (here) the map of your place is opened and can follow you any time.

#### **-Advantages:**

-it not necessary to download the Application for the second person to become a service provider.

#### **-problems:**

- Application is not available for Android.
- The Application version is old and there is no new version.

#### 1.4.4 table of comparison

<b>App</b>	<b>Manasikana</b>	<b>Familo locator</b>	<b>Space Time</b>	<b>The proposed project</b>
<b>Languages</b>	<b>7 languages</b>	<b>English</b>	<b>English</b>	<b>English</b>
<b>Possibility of Registration</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>Possibility to trace the map</b>	<b>No</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>Possibility to direct communication</b>	<b>Yes</b>	<b>Yes</b>	<b>No</b>	<b>Yes</b>
<b>Speed</b>	<b>Slow</b>	<b>fast</b>	<b>slow</b>	<b>Fast</b>

Table (1.4.4): comparison table

## **1.5 feasibility study:**

### **1.5.1: Data collection:**

In this project data collected using questionnaire method. A random sample was chosen from people who had completed Hajj and those who intend to perform Hajj in the future was selected to identify their views on the project idea.

the total number of the sample was (45). questionnaire was distributed through Google Drive in general the result of the questionnaire was that there are people who are afraid of loss in Hajj, and loss affects their performance of Hajj and they have no knowledge of the MANASIK in the Hajj (see appendix).

### **1.5.2 Financial Feasibility:**

Being a mobile application, Pilgrim tracking App will not contain hosting cost. The app will follow Android software standards. No cost will be charged from the potential customers. Bug fixes and maintaining tasks will have an associated cost. At the initial stage the potential market space will be ministry of *Hajj*, Hajj service-provider companies and pilgrims. Beside the associated cost, there will be many benefits for the government, pilgrims and their supervisors. Especially the less effort that is associated with pilgrim's lost and pilgrim guidance will be significantly improved while the effort to provide a full guiding app, since pilgrim locations will be monitored in real time environment. From these it's clear that the project of Pilgrim (Hajj)tracking App is financially feasible.

### **1.5.3 Technical Feasibility:**

The project of Pilgrim Guiding App is a complete mobile application. The main technologies and tools that are associated with the App are:

- Android software standards which includes:
  1. Android software development kit (SDK)
  2. Programming languages such as Java, C++
  3. Third party tools such as AIDE, Android: Build, Corona SDK, Delphi
- Diagram drawing tools (Creatly software)

Each of the technologies are freely available and the technical skills required are manageable. Time limitations of the product development and the ease of implementing using these technologies are managed. Initially the web site will be hosted in a free web hosting space, but for later implementations it will be hosted

in a paid web hosting space with a sufficient bandwidth. In addition, the hardware requirements for running the application are:

- Android devices with Android 5+
- Minimal of 2 GB Ram
- Intel core i3 clocked at nearly 2 GHZ +

From these it's clear that the Pilgrim tracking App is technically feasible.

#### **1.5.4 Resource feasibility:**

The required resources for the Pilgrim Guiding App include:

- Programming device (windows emulators)
- Programming tools (freely available)
- Programming individuals

That's it, it's clear that the Pilgrim tracking App has the required resource feasibility.

#### **1.5.5 Does the environment make use of a database or repository?**

- This is a database-oriented app that will use MYSQL. Are all the software tools integrated with one another? Main deliverables will be packaged under a single project. All the stake holders will have a single login page.
- Process issue risks: Pilgrim Guiding App will follow the Agile software development process. This provides the flexibility to accommodate changing software requirements of Pilgrim Guiding App.
- Technical issue risks Are specific conventions for code documentation defined and used? Software code will be available upon request in coordination with our college and the code documentation will be provided.
- Technology risks: Is the technology to be built new? All the technologies are very well established and old enough (but not obsolete).

#### **1.5.6 Social/Legal Feasibility:**

- Pilgrim Guiding App uses freely available development tools.
- Since this new app eliminates the effort to track, guide and support pilgrim as needed, it will have a great impact in Hajj season.

#### **1.5.7 Maintainability:**

- Pilgrim Guiding App is designed using the best practices of Agile and OOP. Since every single segment in the App is very well structured, the
- App is highly maintainable.

## **Chapter 2: System Analysis**

### **2.1 introduction:**

“The systems analyst plays a key role in information systems development projects. The systems analyst works closely with all project team members so that the team develops the right system in an effective way. Systems analysts must understand how to apply technology to solve business problems. In addition, systems analysts may serve as change agents who identify the organizational improvements needed, design systems to implement those changes, and train and motivate others to use the systems”.  
(BARBARA,H.2008,pp8)

### **2.2 description of Data Flow Diagram(DFD):**

Also known as BUBBLE CHART, a data flow diagram(DFD) is a graphical representation of the flow of data through an information system , modeling its process aspects.

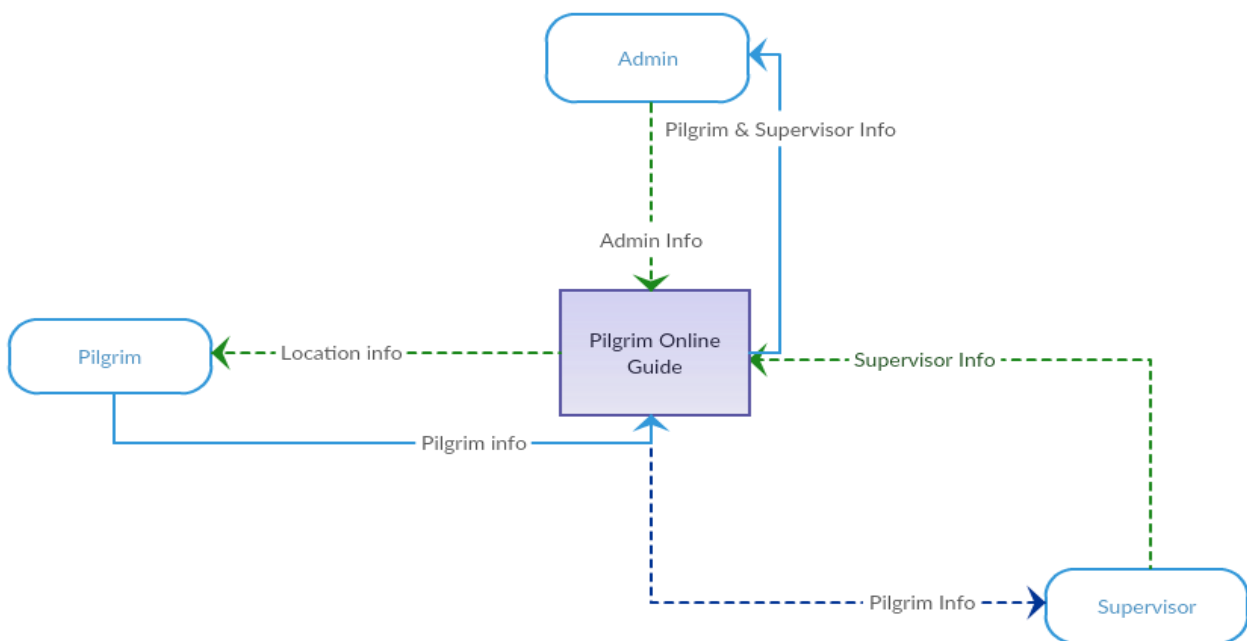
DFD shows what kind of information will be input to and output from the system. From where the data will come and where the data will be stored.

DFD does not show information about the timing of processes or information about whether processes will operate in sequence or in parallel which is shown on the Flowchart. (Bharath.P.2012)

### 2.2.1 Context diagram:

Context diagram is used to establish the context and boundaries of the system to be modeled: which things are inside and outside of the system being modeled, and what is the relationship of the system with these external entities. A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. (Bharath.P.2012)

The diagram figure 2.2.1 three main components which are system, actors and inputs. the system has three main actors Admin, Supervisor, and Pilgrim respectively. Each one of them will interact with the system to perform some tasks. Their interaction with the system occur using inputs.

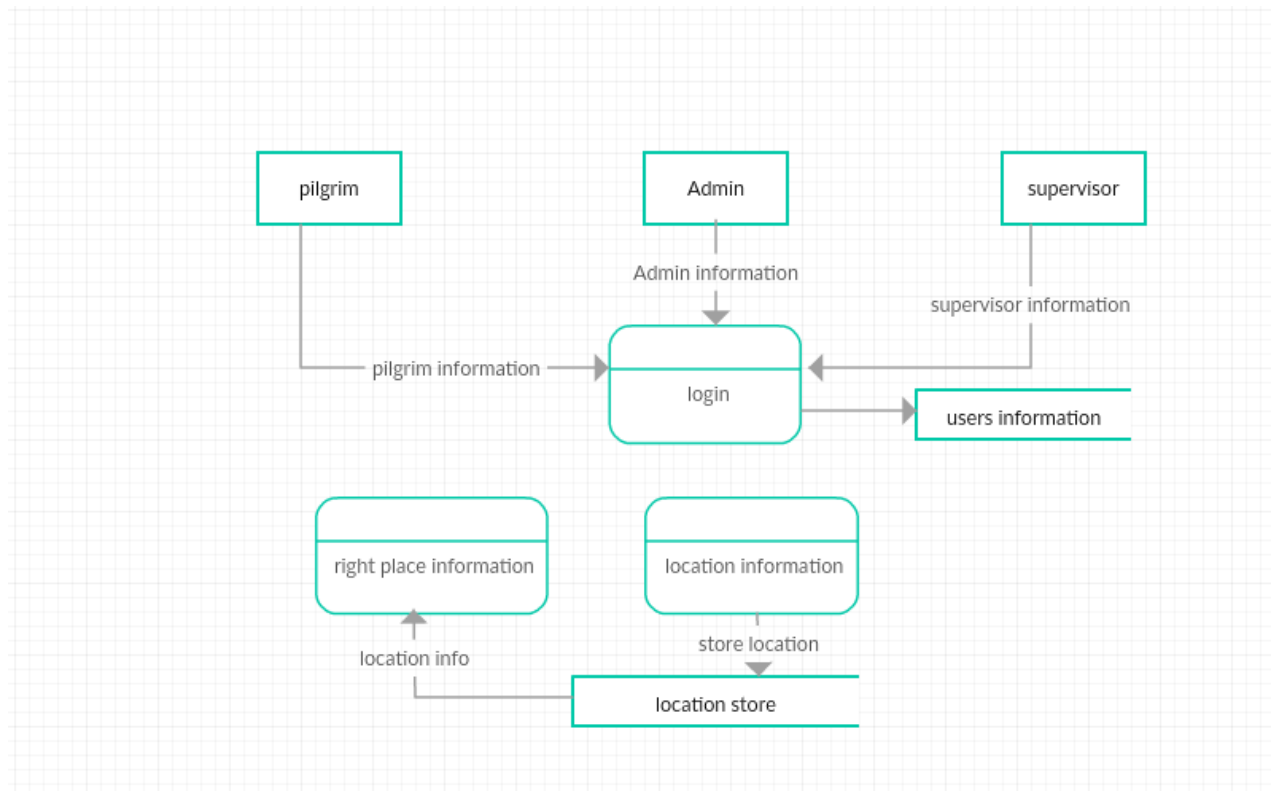


figure(2.2.1): context diagram

### 2.2.2 overview diagram(level0):

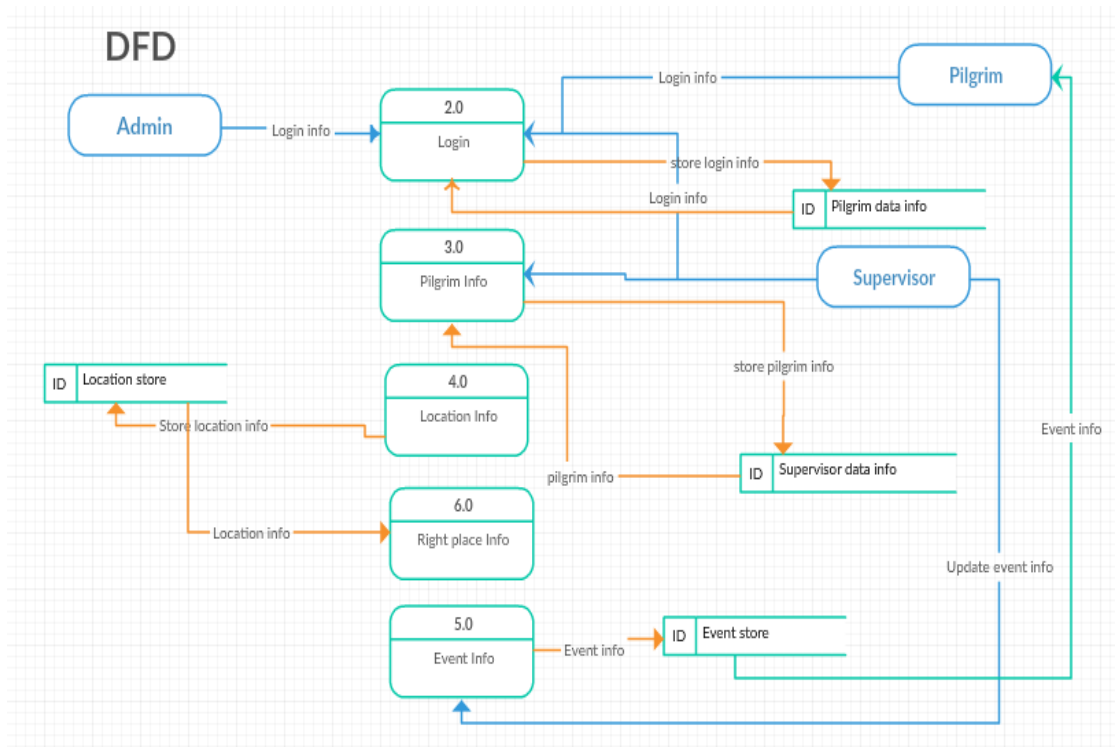
In figure(2.2.2) three actors Admin, Supervisor and pilgrim .

Show the process of each actors and data store for each information.



Figure(2.2.2): DFD level0

### 2.2.3 Detailed DFDs:



Figure(2.2.3): detailed DFD



## **2.3 Entity Relationship Diagram(ERD):**

An entity-relationship diagram (ERD) is a graphical representation of an information system that shows the relationship between people, objects, places or concepts within that system. An ERD is a data modeling technique that can help define business processes and can be used as the foundation for a relational database.

### **2.3.1 Description of Entities:**

-Admin entity which represent app administer and have the following attributes (User Name, Password, Mobile).

-pilgrim entity which represent app users and have the following attributes (User Name, password, Mobile) .

-supervisor entity which represent app trip mentor and have following attributes (User Name, password, Mobile).

### **2.3.2 Description of relations:**

- Every supervisor can add or delete or modified each pilgrim.

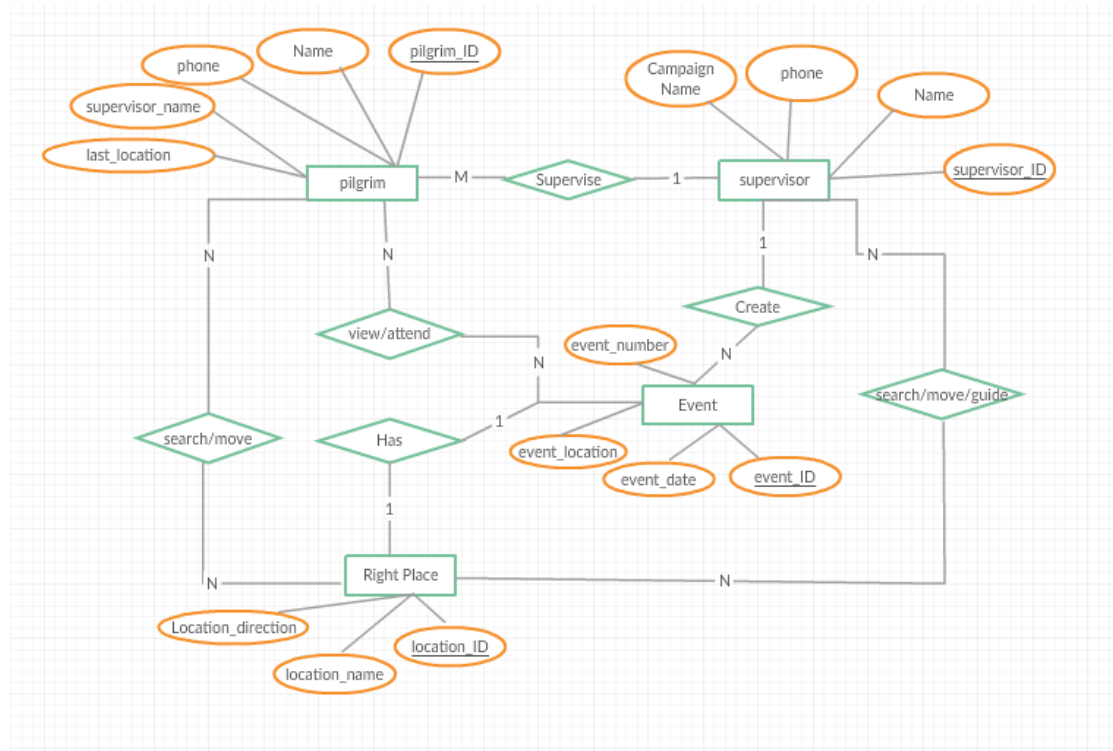
- Every supervisor in the application can communication with one or more pilgrims and pilgrim communicate with one supervisor

- Every supervisor can show location one or more pilgrims and pilgrim can show location one supervisor.

- Every supervisor can add or modified one or more event.

-Every pilgrims can search one or more of a right place.

### 2.3.3: ER Diagrams:

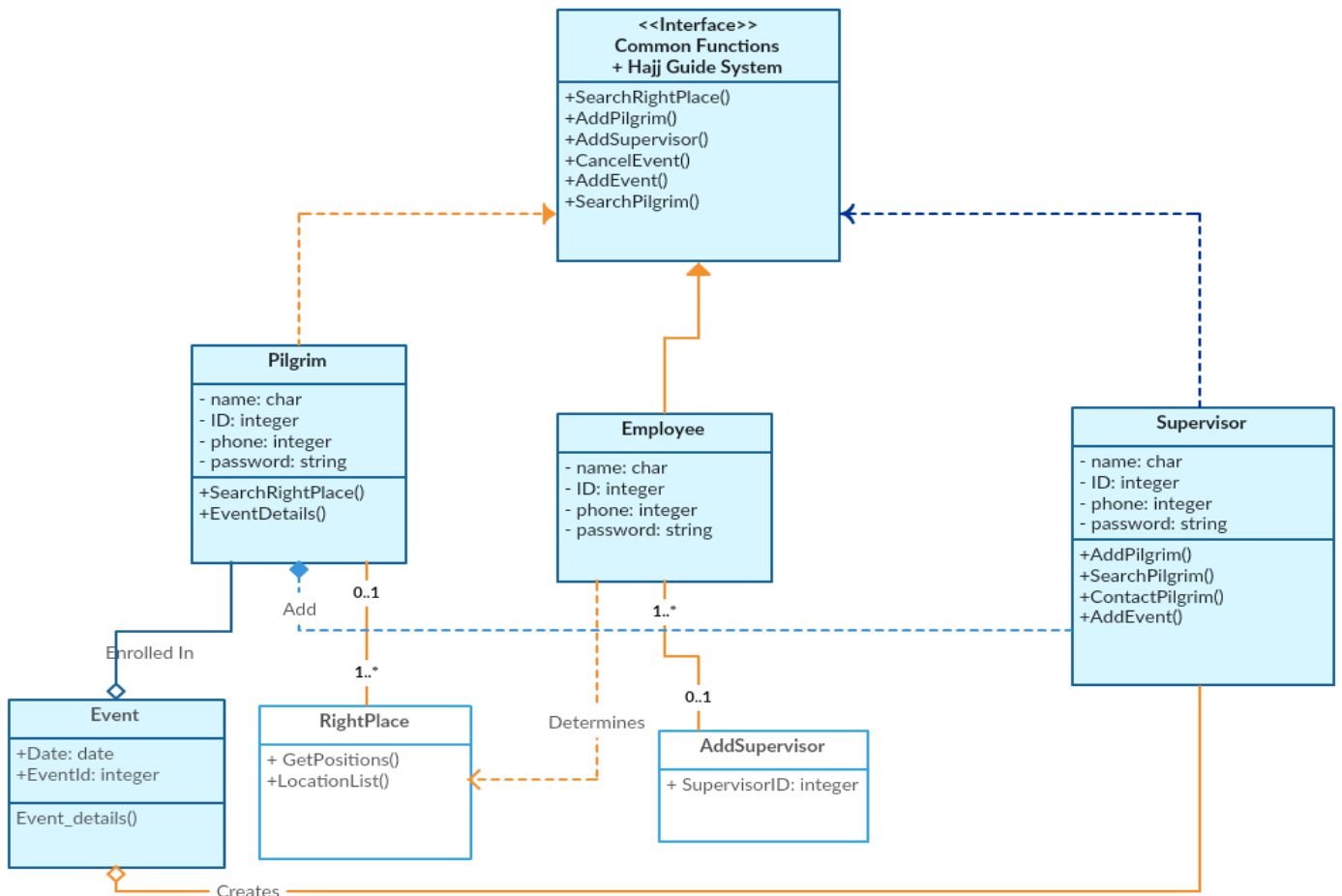


figure(2.3.3): ER diagram

## 2.4 Class Diagram:

A class describes a group of objects with similar properties (attributes), common behavior (operations), common relationships to other objects, and common meaning (“semantics”). (Bharath.P.2012)

- AddPilgrim() Add pilgrims by supervisor
- SearchPilgrim() Search for pilgrim location
- ContactPilgrim() Call the pilgrim and check his/ her status
- AddEvent() Supervisor can add, update, delete or call for an event to be attended by pilgrims
- SearchRightPlace() Search and request right location to visit
- AddSupervisor() Admin can add one or more supervisors
- EventDetails() Pilgrims and supervisors can view and explore event details

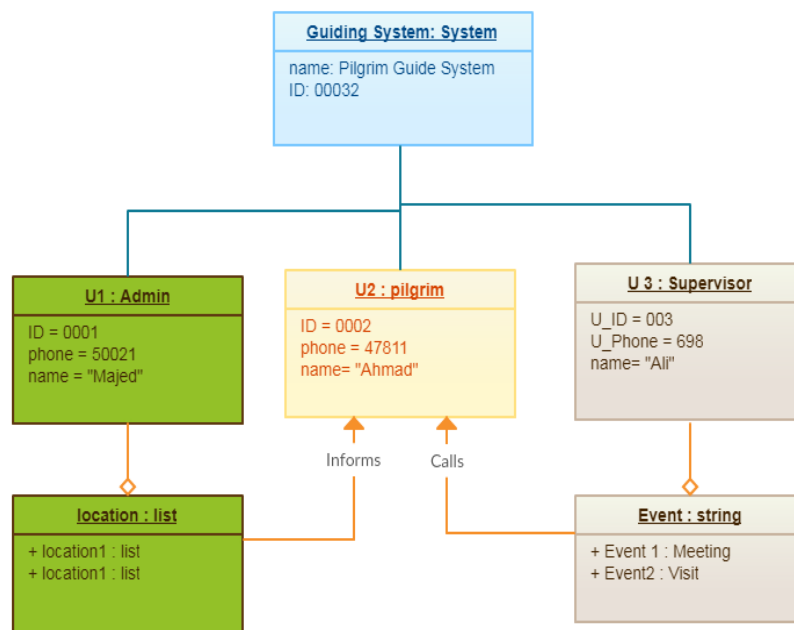


Figure(2.4): class diagram

## 2.5 Object Diagram:

Model the instances of things described by a class. Each object diagram shows a set of objects and their interrelationships at a point in time. Used to model a snapshot of the application. Each object has an optional name and set of classes it is an instance of, also values for attributes of these classes. (Bharath.P.2012)

Figure 2.5 shows the object diagram of the system that demonstrates the relation between the instantiated classes and the defined class, and the relation between these objects in the system. Objects (Admin, Supervisor, and Pilgrim) are an instance of a moments in runtime, including objects and data values. It may be considered a special case of a class diagram or a communication diagram.



Figure(2.5): object diagram

## 2.6 Use Case Diagram:

A use case diagram can be defined as a graphical description of the interactions between system elements .A use case is a methodology used in system analysis to identify, explain, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML, a standard notation for the modeling of real-world objects and systems.(Bharah.P.2012)

Actor in my system are Admin, Supervisor and Pilgrim.

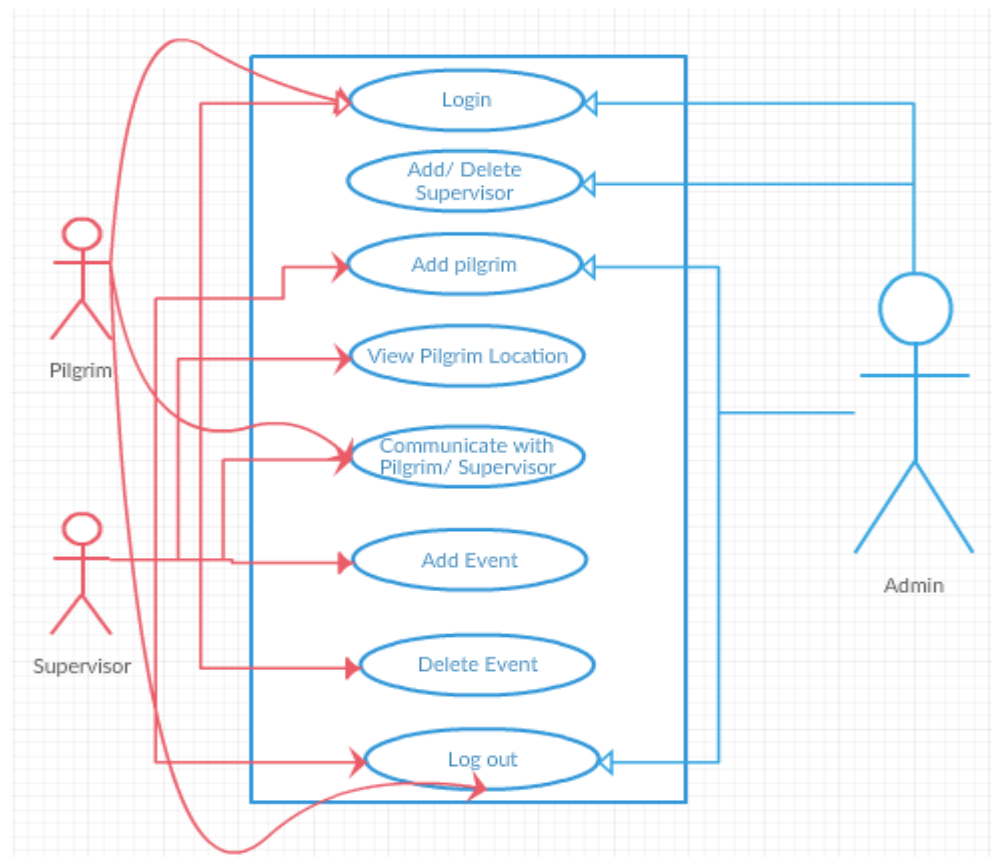
Use cases in my system are login (admin, supervisor, pilgrim),

admin is (add/delete supervisor,) admin and supervisor are ( add pilgrim),

supervisor is (View pilgrim location), pilgrim is (view supervisor location),

communication with pilgrim/ supervisor, supervisor is ( add event, delete event) and

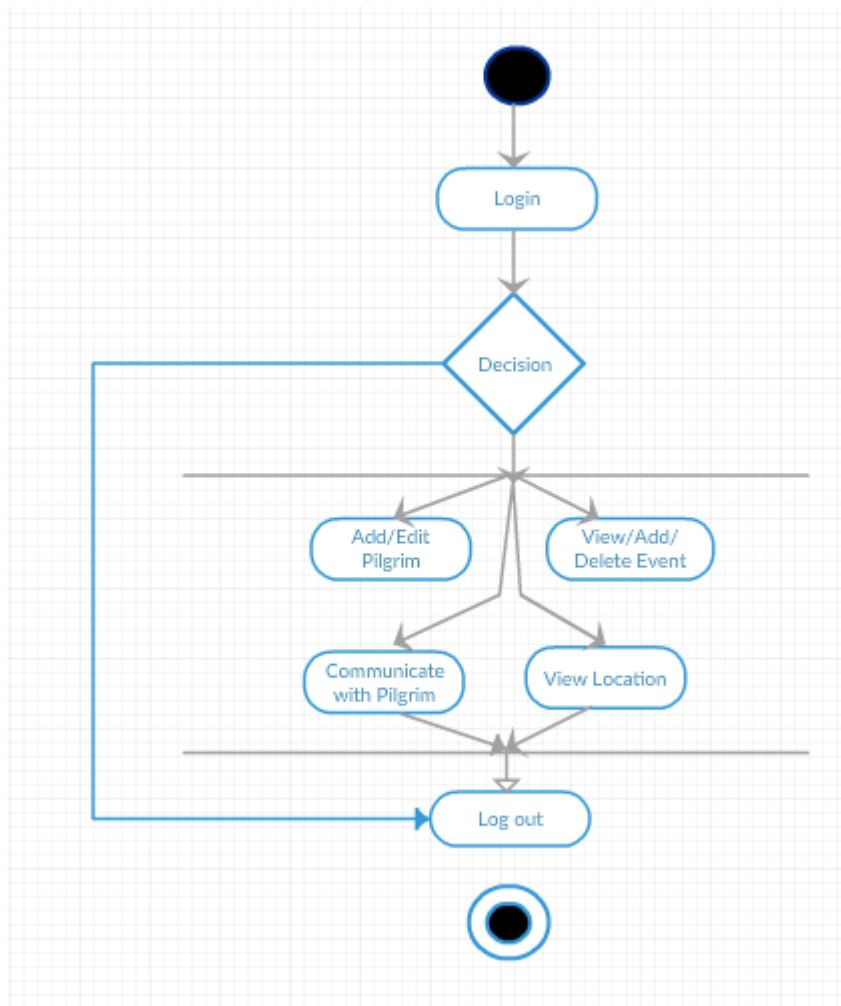
log out(admin, supervisor, pilgrim).



Figure(2.6): use case diagram

## 2.7 Activity Diagram:

in UML can be defined as a graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level. (Bharah.P.2012)



Figure(2.7.1): supervisor object diagram

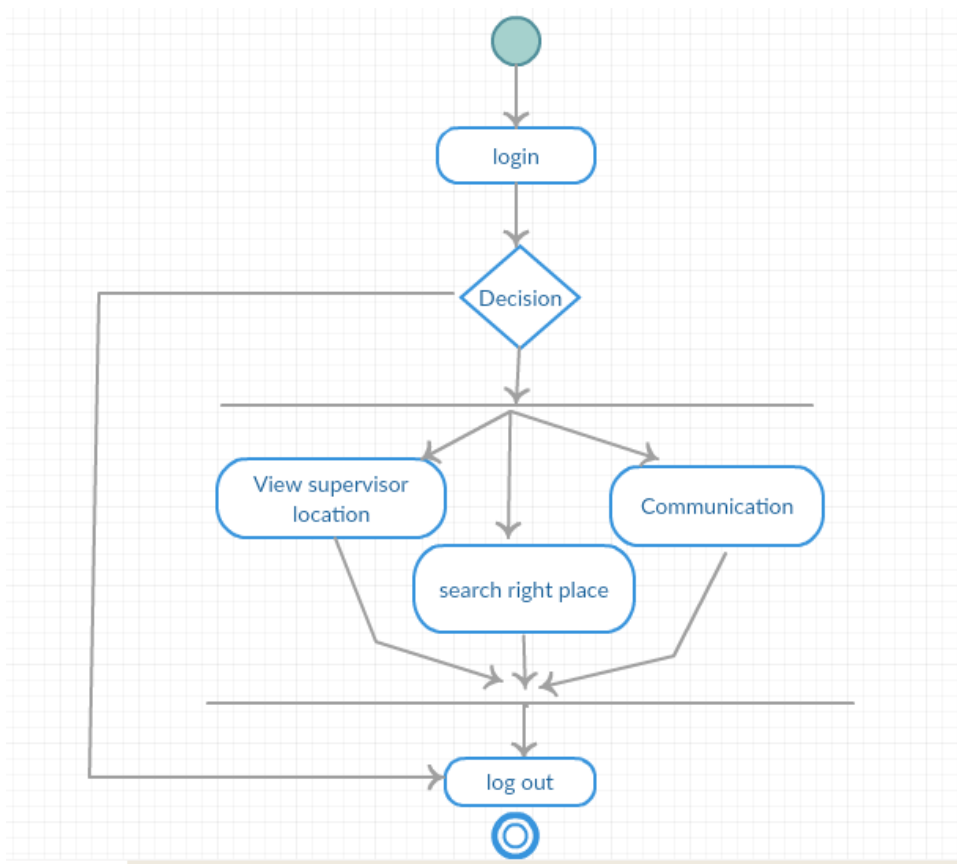


Figure 2.7.2pilgrim object diagram

## 2.8 Sequence Diagram:

“Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when”(visual-paradigm. n.d. para1)

Figure 2.8 The supervisor initiates the system trying to add new pilgrims, update right places list, add, update or delete an event. The information of any function made by the supervisor will be added and updated in the database. Then, the system will react to any query or data entry responding to supervisor’s request. In this sequence diagram, the supervisor for example request a location. Then the system checks location directions, select the right place and send it back to supervisor and to pilgrim as well.

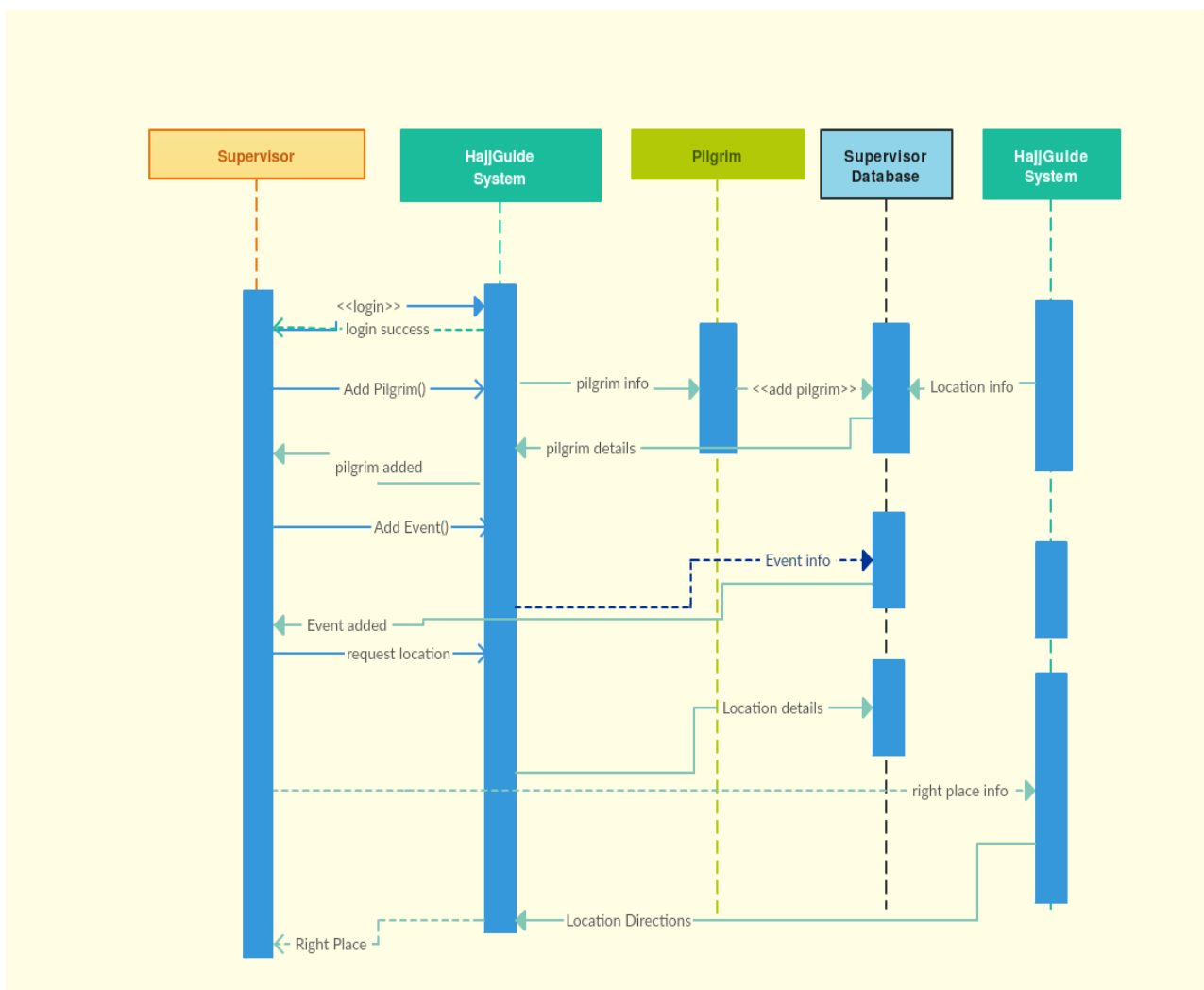


Figure 2.8 sequence diagram



## 2.9 state diagram:

Figure 2.9 explains the state diagram that contains the following components:

- Initial state: it shows the starting point or first activity of the flow.
- Final state: it represents the end of state diagram where no action is to be taken (all ended functions will go to final state).
- State: it represents the instances in the system (add pilgrim, search right place, search for pilgrim, add event)
- Transition: an arrow indicating the object to transition from one state to the other.

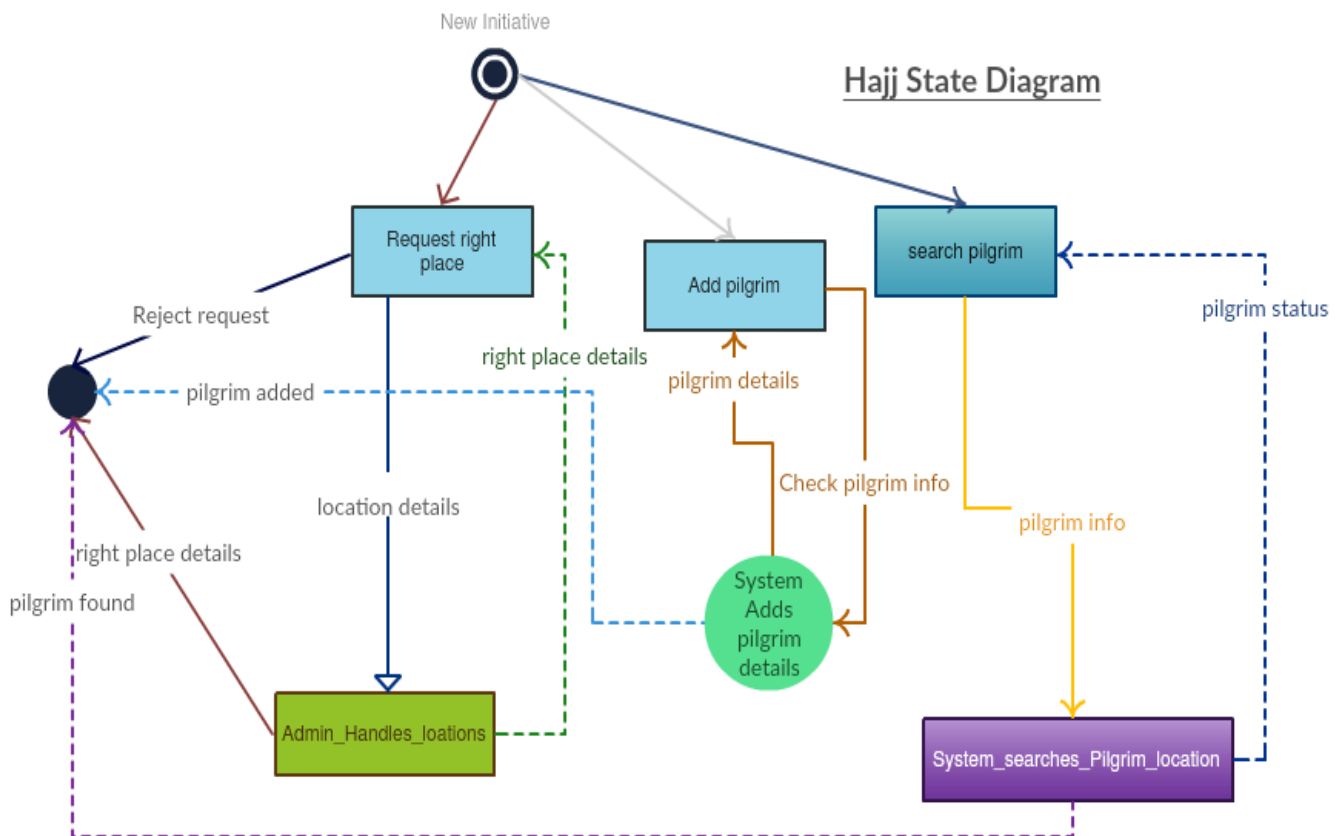


Figure 2.9 state diagram

## **Chapter 3 system design:**

### **3.1: Description of procedures and function:**

This section explains the procedures and functions of the online Pilgrim Guiding System. The functionality of the system is divided into system administration functions and system user functions. The details description of the mentioned functionalities is listed below:

#### **3.1.1 System Functionality**

The functions of the system are divided into two main categories, functions provided to the system administrator and functions provided to the users.

##### **System Administration Functions**

###### **- Add A New User (Pilgrim or Supervisor)**

It allows the system administrator to add a new user by identifying the basic information (ID, name, phone number, registration date... etc.) and identify the type of user (pilgrim or Supervisor).

###### **- Modify User Information**

This feature allows the admin to modify user information by entering user ID and then displays the information that can be modified.

- **Delete User:** Allow to delete specific user by entering user ID.

- **Show All Users:** It allows the system administrator to view all the user information by selecting the type of user and year of registration.

- **Add Event:** This feature allows the administrator to create an event, modify the event or delete that event.

#### **3.1.2 User Functions**

The online pilgrim guiding system provides the following functions to the supervisor.

###### **- Safe range Distance**

It allows the supervisor to identify the safe range distance with the pilgrims.

###### **- Notify the supervisor**

Notify the supervisor of all pilgrims (belong to him) about any pilgrim who exceeds the safe distance and show their location on map.

###### **- Pilgrim Tracking**

Track all pilgrims and show their location on map.

###### **- Event Call**

This feature allows the supervisor to call pilgrims to attend an event and to guide them to event location.

###### **- Send help request**

The system allows pilgrims to send help requests to their supervisors when they need any assistance.

### 3.2:Relation database schema:

Table (3.2.1): User database

Field Name	Date type	Properties
uId	String	Foreign Key
Email	String	Primary key
Admin name	Character	
Phone number	Integer	
Password	String	
privilge	Integer	
superv	String	

Table (3.2.2): Event database

Field Name	Data type	properties
body	String	
evDate	Integer	
evTime	Integer	
evLoc	String	
sub	Character	
pilgname	Character	
superme	String	
userId	String	

Table (3.2.3): Location database

Field Name	Data type	Properties
lat	float	
Lng	float	
mysuper	string	
username	character	

### 3.3: Hardware and Software requirements:

#### 3.3.1 Hardware requirement:

The following table the minimum and recommended hardware requirements for deploying pilgrim tracing. (MT.HOOD community college. 208)

Table 3.3.1 Hardware requirement

Component	Minimum	Recommended
Processor	2.5 gigahertz (GHz)	Dual processors that are each 3 GHz or faster
RAM	1 gigabyte (GB)	2 GB or more
Disk	NTFS file system–formatted partition with a minimum of 3 GB of free space	NTFS file system–formatted partition with 3 GB of free space plus adequate free space for your Web sites
Drive	DVD drive	DVD drive or the source copied to a local or network-accessible drive
Display	1024 × 768	1024 × 768 or higher resolution monitor
Network	56 kilobits per second (Kbps) connection between client computers and server	56 Kbps or faster connection between client computers and server

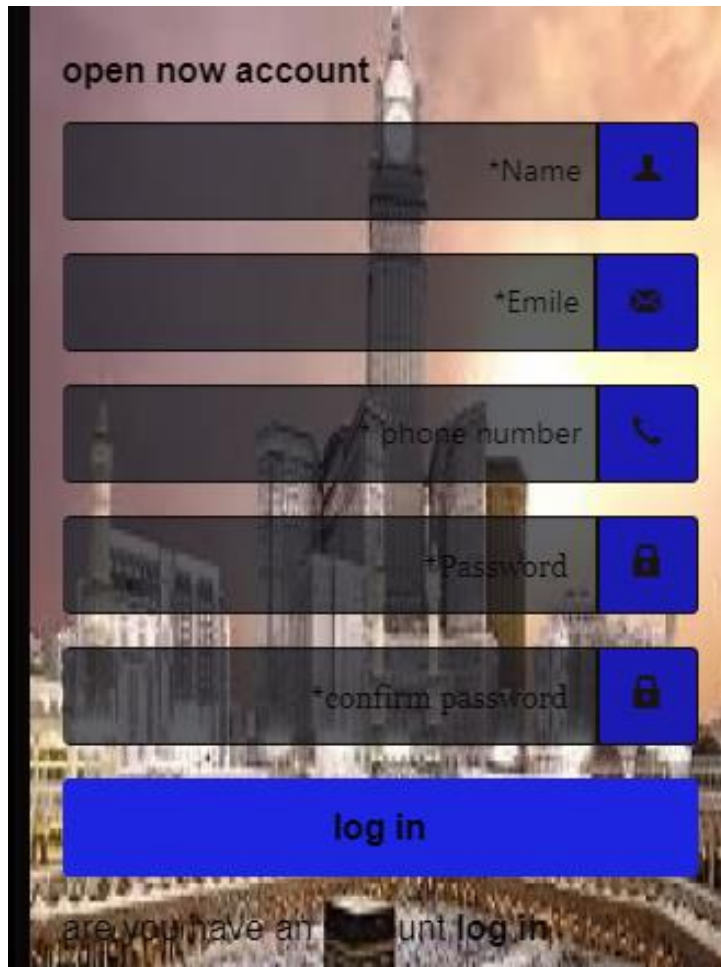
### **3.3.2: software requirement:**

The project of Pilgrim tracking System is android application. The main technologies and tools that are associated with the system are:

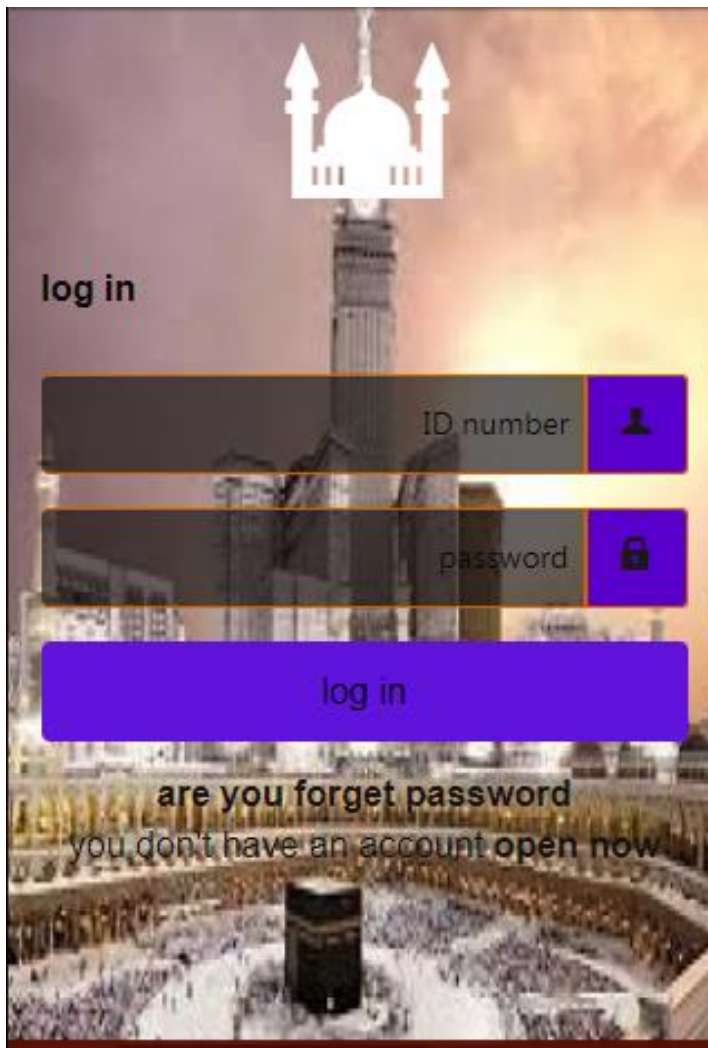
- Android studio
- Java.
- Xml.
- Firebase Database.
- Diagram drawing tools (Creatly software)

Each of the technologies are freely available and the technical skills required are manageable. Time limitations of the product development and the ease of implementing using these technologies are managed. Initially the web site will be hosted in a free web hosting space, but for later implementations it will be hosted in a paid web hosting space with a sufficient bandwidth.[9]

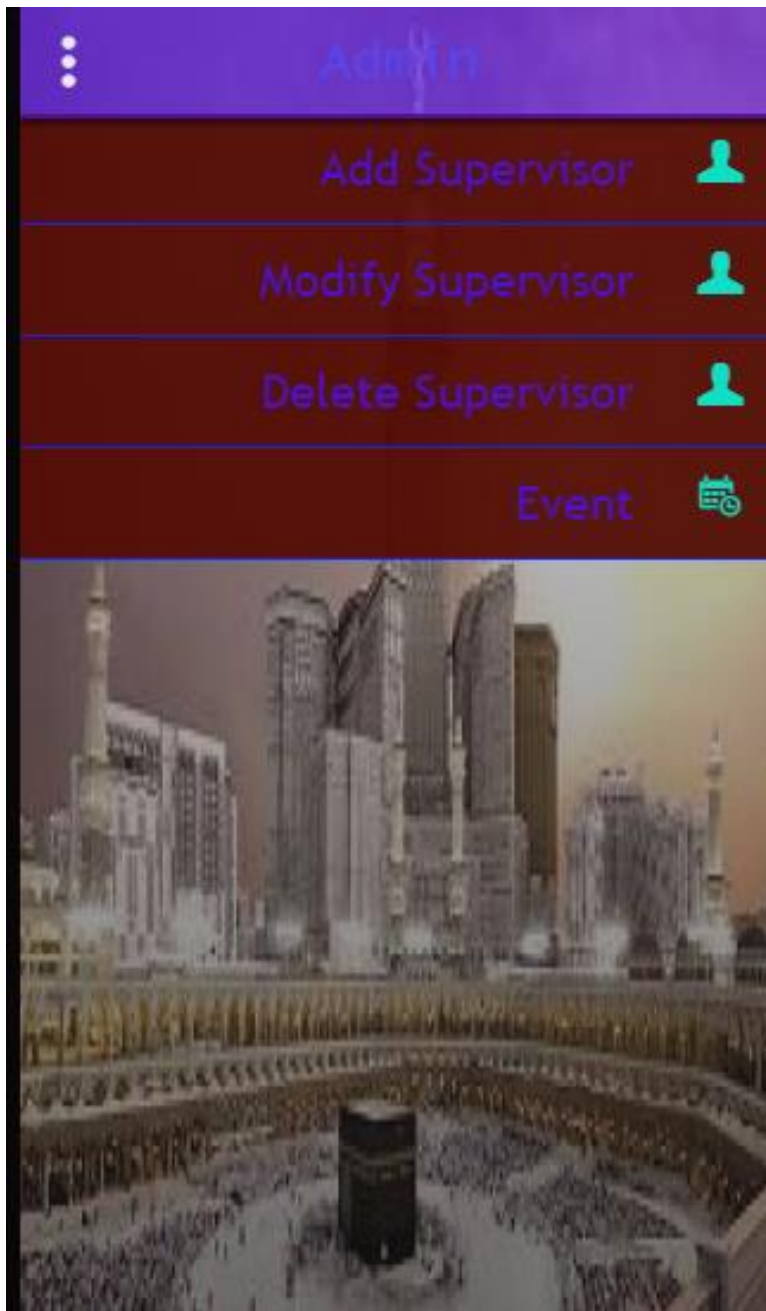
### 3.4:Initial Interfaces:



Figure(3.4.1): registration interface

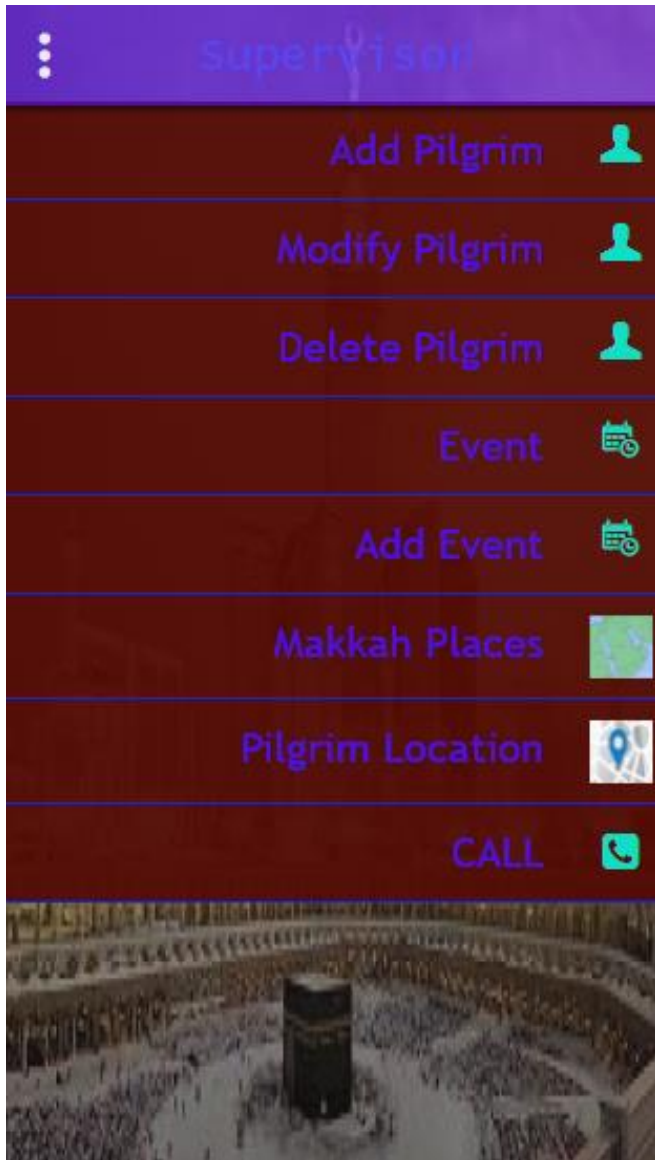


Figure(3.4.2): login interfaces



Figure(2.4.3): Admin interface





Figure(3.4.4): supervisor interface

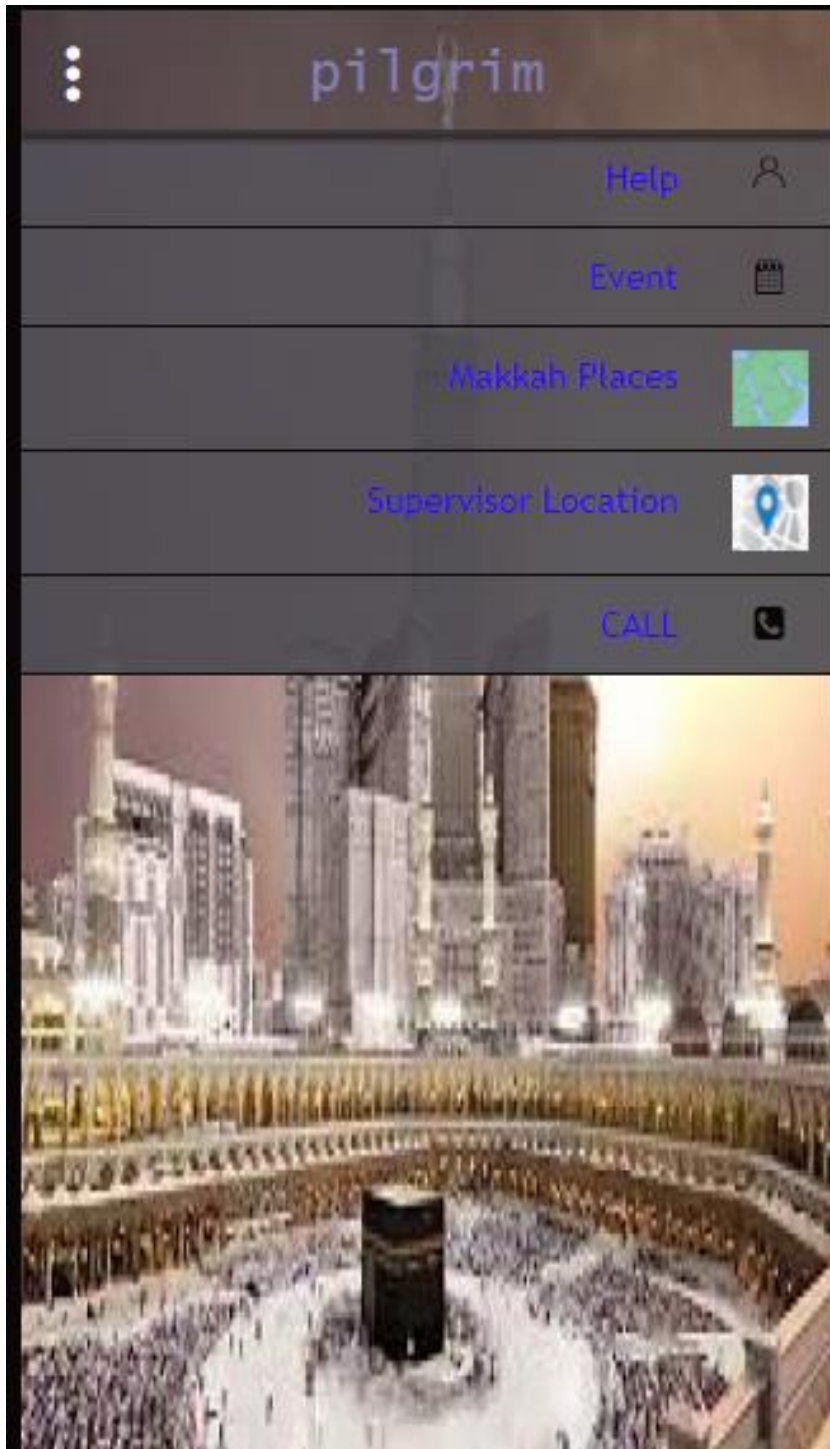


Figure (3.4.5): pilgrim interface

## Chapter 4: implementation and Testing:

### 4.1: Introduction:

A programming language implementation is a system for executing the program.

We use Android studio as editor and Firebase as Database.

### 4.2: procedures:

#### Add Supervisor Function:

```
private void writeNewSupPilg(String userId, String Pilgrimname, String
Phonenumber, String pakagename,String Pilgrim_num) {
// Create new post at /user-posts/$userid/$postid and at
// /posts/$postid simultaneously
String key = FirebaseDatabase.child("Pilgrim").push().getKey();
Pilgrim post = new Pilgrim(userId, Pilgrimname, Phonenumber,
pakagename,Pilgrim_num);
Map<String, Object> postValues = post.toMap();

Map<String, Object> childUpdates = new HashMap<>();
childUpdates.put("/Pilgrim/" + key, postValues);
childUpdates.put("/Admin-Supervisor/" + userId + "/" + key, postValues);

mDatabase.updateChildren(childUpdates);
} // add supervisor in database
```

#### Add Pilgrim Function:

```
private void writeNewPilg(String userId, String Pilgrimname, String
Phonenumber, String pakagename,String Pilgrim_num) {
// Create new post at /user-posts/$userid/$postid and at
// /posts/$postid simultaneously
String key = FirebaseDatabase.child("Pilgrim").push().getKey();
Pilgrim post = new Pilgrim(userId, Pilgrimname, Phonenumber, pakagename,
Pilgrim_num);
Map<String, Object> postValues = post.toMap();

Map<String, Object> childUpdates = new HashMap<>();
childUpdates.put("/Pilgrim/" + key, postValues);
childUpdates.put("/Supervisor-Pilgrim/" + userId + "/" + key,
postValues);

mDatabase.updateChildren(childUpdates);
} // add pilgrim in database
```

#### Modify Function:

```
private void updatePilg(DatabaseReference postRef) {
postRef.runTransaction(new Transaction.Handler() {
@Override
public Transaction.Result doTransaction(MutableData mutableData) {
Pilgrim p = mutableData.getValue(Pilgrim.class);
```

```

        if (p == null) {
            return Transaction.success(mutableData);
        }

        p.Pilgrim_name=mName.getText().toString();// update the name
        p.Phone_number=mPhone.getText().toString();//update the phonenum

        mutableData.setValue(p);
        return Transaction.success(mutableData);
    }

```

### Delete Function:

```

private void deletePilg(final DatabaseReference postRef) {
    postRef.runTransaction(new Transaction.Handler() {
        @Override
        public Transaction.Result doTransaction(MutableData mutableData) {
            Pilgrim p = mutableData.getValue(Pilgrim.class);
            if (p == null) {
                return Transaction.success(mutableData);
            }

            if (p.stars.containsKey(getUid())) {
                // Unstar the post and remove self from stars
                p.starCount = p.starCount - 1;
                p.stars.remove(getUid());

                String priv= User.getpublicpriv();
                String uid= p.Pilgrim_num;
                if(priv.equals("0")) {
                    mDatabase.child("Pilgrim").child(postRef.getKey()).removeValue();
                    mDatabase.child("Admin-
Supervisor").child(User.getpublicid()).child(postRef.getKey()).removeValue(); // delete

                    mDatabase.child("users").child(uid).removeValue();

                }

                else if(priv.equals("1")) {
                    mDatabase.child("Pilgrim").child(postRef.getKey()).removeValue();// delete pilgrim from data table
                    mDatabase.child("Supervisor-
Pilgrim").child(User.getpublicid()).child(postRef.getKey()).removeValue();
                }

            } else {
                p.starCount = p.starCount + 1;
                p.stars.put(getUid(), true);
            }

            mutableData.setValue(p);

```

```

        return Transaction.success(mutableData);
    }

    @Override
    public void onComplete(DatabaseError databaseError, boolean b,
        DataSnapshot dataSnapshot) {
        //Transaction completed
        Log.d(TAG, "postTransaction:onComplete:" + databaseError);
    }
});
}

```

## Tracing Function:

```

        private void getFromDbLng() throws InterruptedException {

// This method fetches the data

        DatabaseReference mDatabase =
        FirebaseDatabase.getInstance().getReference();

        final String userId =
        FirebaseAuth.getInstance().getCurrentUser().getUid();

        mDatabase.child("Pilgrim-
location").orderByChild("mysuper").equalTo(userId).addListenerForSingleValue
Event(new ValueEventListener() {

            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Log.e("Count ", "" + dataSnapshot.getChildrenCount());
                f=new String[(int) dataSnapshot.getChildrenCount()][3];
                int i=0,j=0;
                for (DataSnapshot postSnapshot:
dataSnapshot.getChildren()) // reorder the data in class format
                {
                    Locat c = postSnapshot.getValue(Locat.class);
//stroe the data and the name of coordinates on the map
                    f[i][0]=c.lat;
                    f[i][1]=c.lng;
                    f[i][2]=c.username;
                    Log.v(TAG, " add to list "+ f[i][0]+" "+f[i][1]+"
"+f[i][2]);
                    i++;
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });

    }

    private String getLng(String lng){

        longitude =lng;
        return lng;
    }
}

```

```

    }

    private String getlat(String lat){
        latitude =lat;
        return lat;
    }// view the coordinates of the pilgrims who follow a particular
supervisor in matrix

public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
    try {
        TimeUnit.SECONDS.sleep(5); // download data stop
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    for(int i=0;i<f.length;i++) { //matrix element of loop of the pilgrim

        lon = Double.parseDouble( f[i][1] ); // extraction of coordinates of
pilgrims
        lat = Double.parseDouble( f[i][0] );

        LatLng location = new LatLng( (lat), (lon) );
        Log.i( "Location", location.toString() +" "+f[i][2]);
        mMap.addMarker( new MarkerOptions().position( location ).title(
f[i][2]+" Pilgram Location" ) ); // Assign the name of the pilgrim
stored in the matrix to the mark
        mMap.animateCamera( CameraUpdateFactory.newLatLngZoom( location, 15.0f )
);

    }

}

```

### 4.3 Layouts:

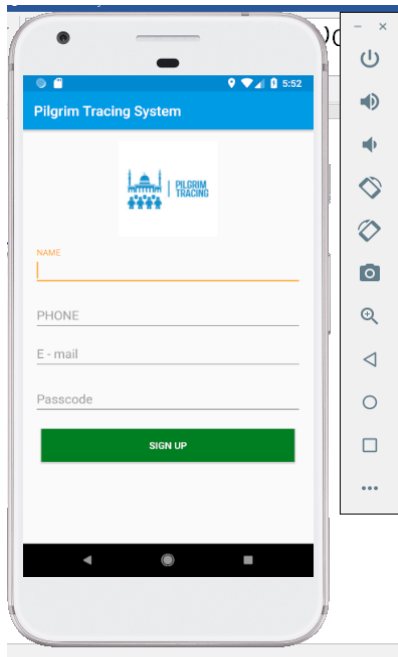


Figure 4.3.1 sign up interface

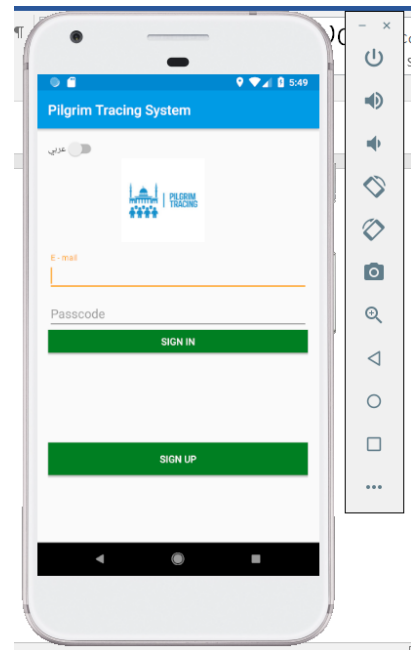


Figure 4.3.2 sign in interface

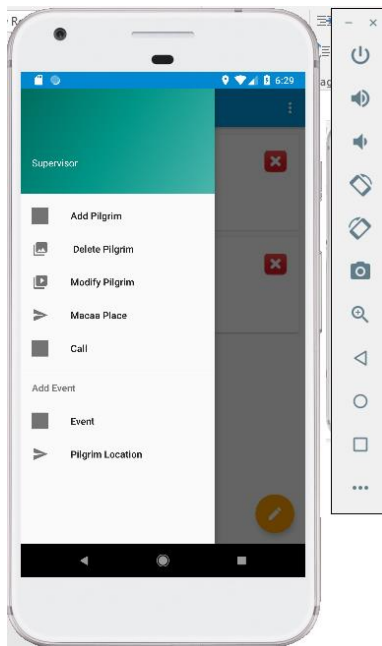


Figure 4.3.3 supervisor interface

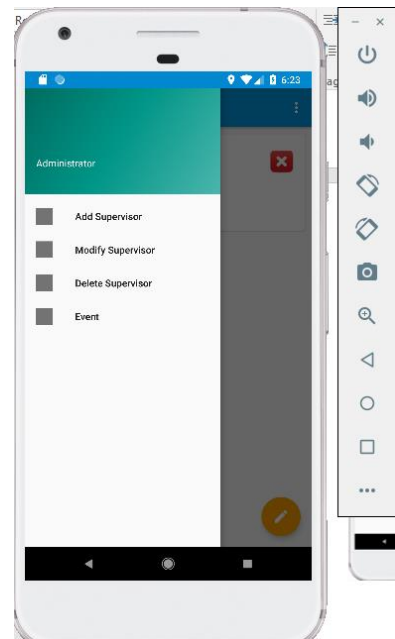


Figure 4.3.4 Admin interface

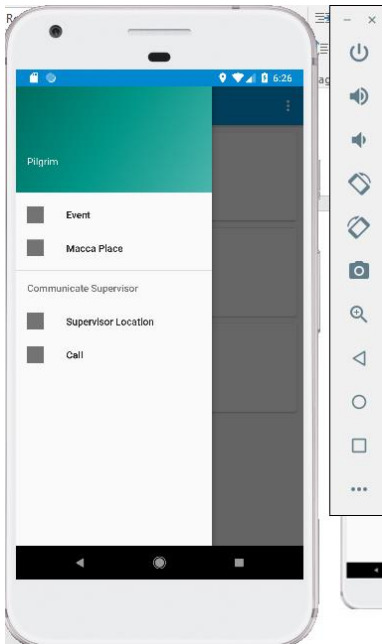


Figure 4.3.5 pilgrim interface

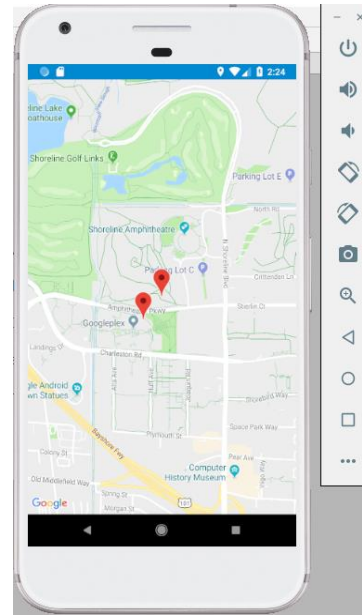


Figure 4.3.6 pilgrim's location

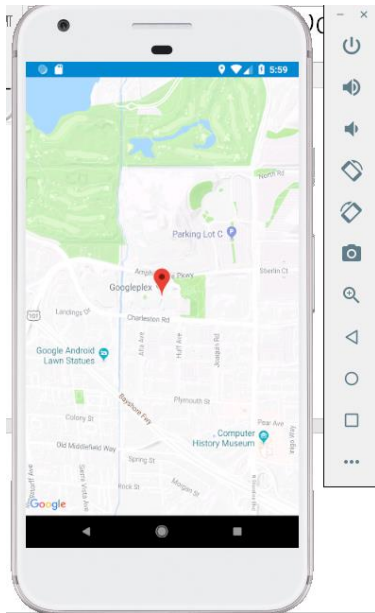


Figure 4.3.7 supervisor location



#### 4.4 Report Layouts:

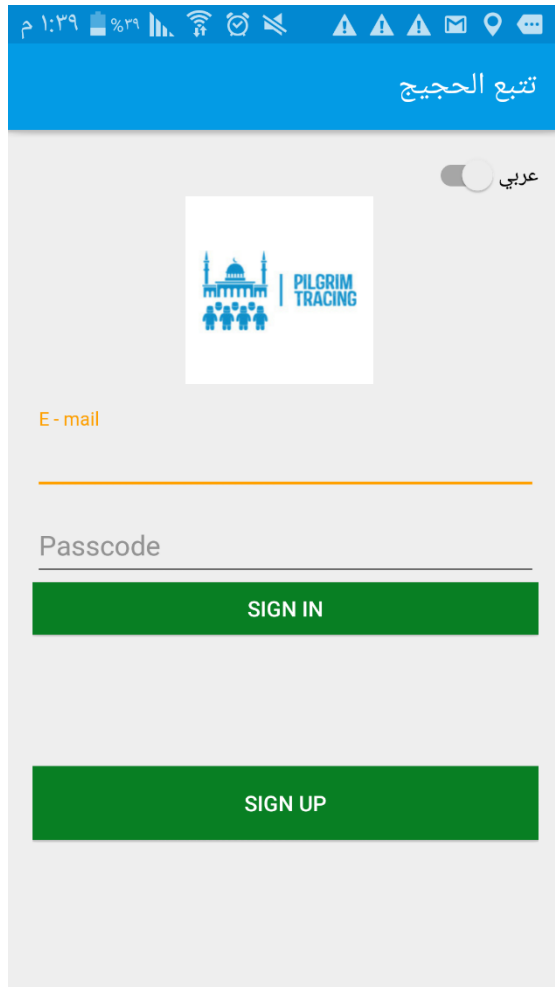


Figure 4.4.1 sign in interface

Login site for all users If you are not registered, click (Register) and there is an option to switch Arabic or English languages. Here we chose the English language.

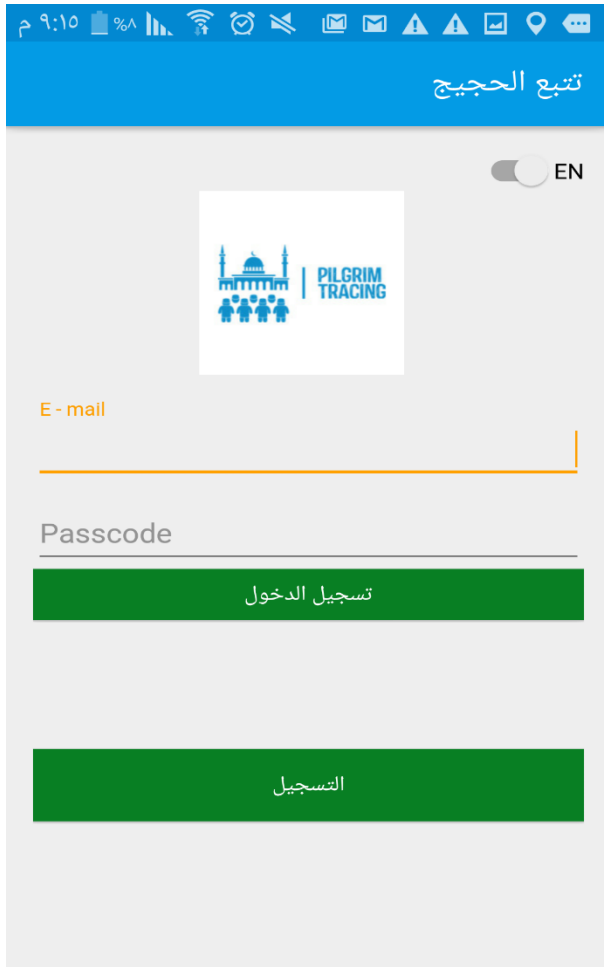


Figure 4.4.2 Sign in Arabic Language.

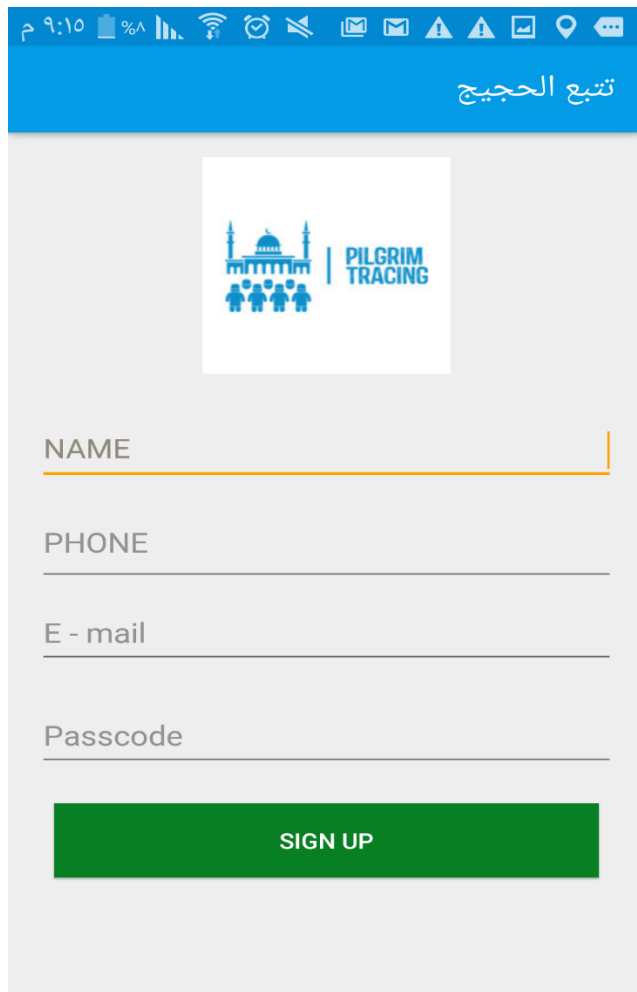


Figure 4.4.3 sign up interface

Sign Up interface for Admin of Campaign. Then Admin Can Add Supervisors.

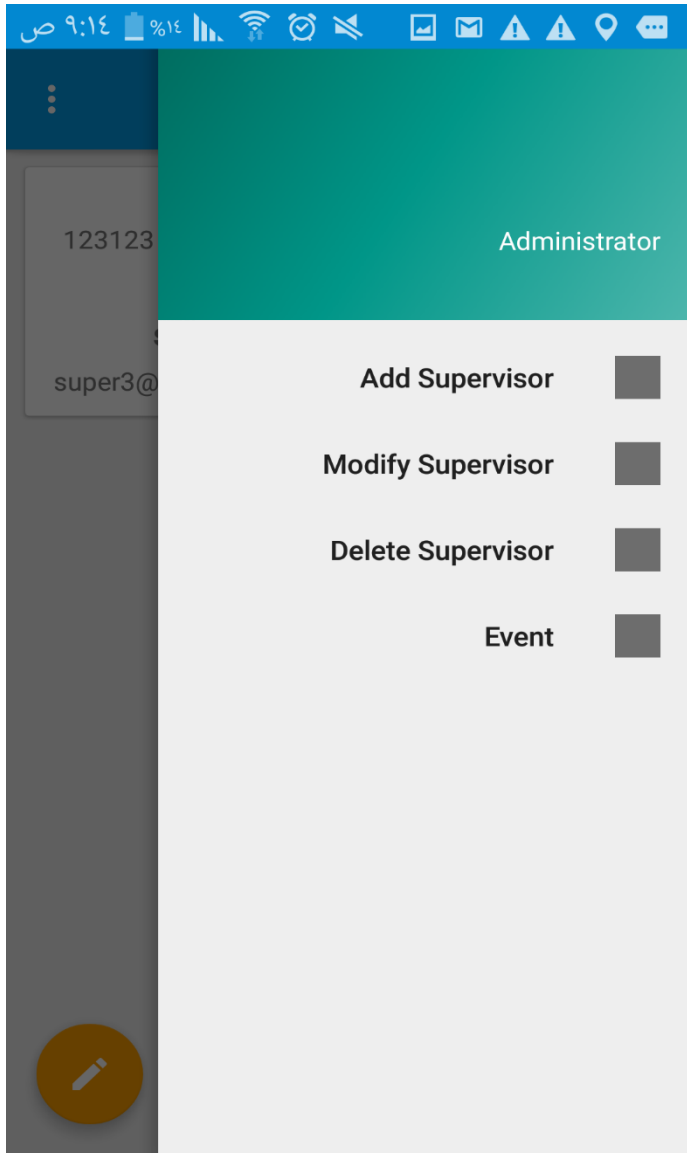


Figure 4.4.4 Admin interface

Admin of Campaign interface. Can (Add, Modify, Delete) Supervisors and show the Event.

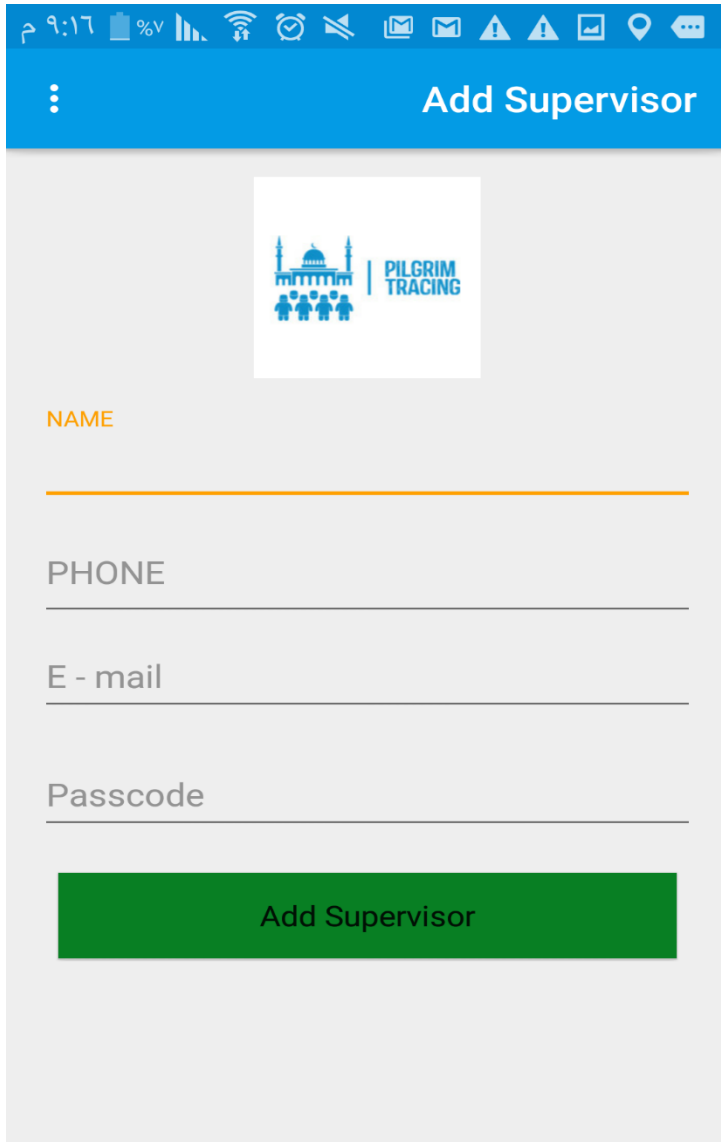


Figure 4.4.5 Admin Add Supervisors.

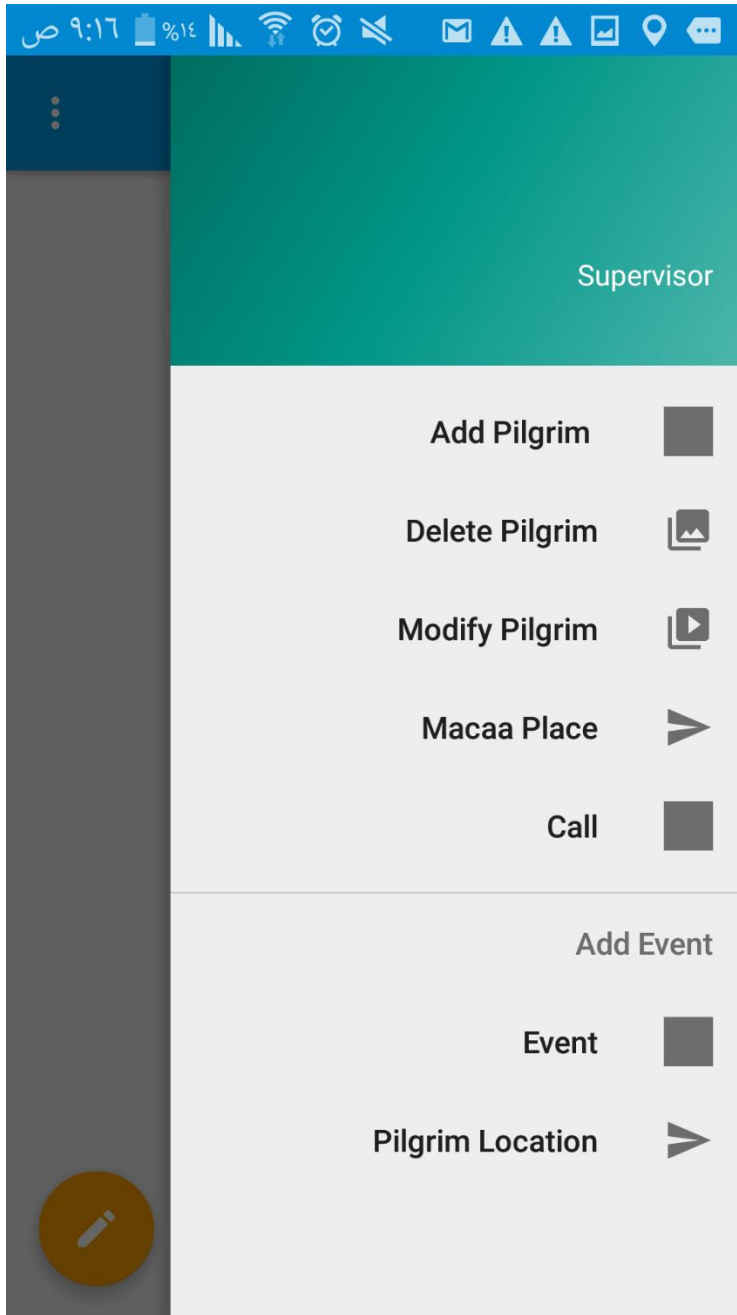



Figure 4.4.6 Supervisor interface.

Supervisor can (Add, Delete, modify) pilgrim, Add new event, Call pilgrim and Show pilgrim location

1:41 م 28% 3G 4G 5G Wi-Fi Alarm Do Not Disturb Location Services Alerts Mail Calendar Messages

تتبع الحجيج



NAME

---

PHONE

---

E - mail

---

Passcode

---

**ADD PILGRIM**

Figure 4.4.7 Supervisor Add Pilgrims.

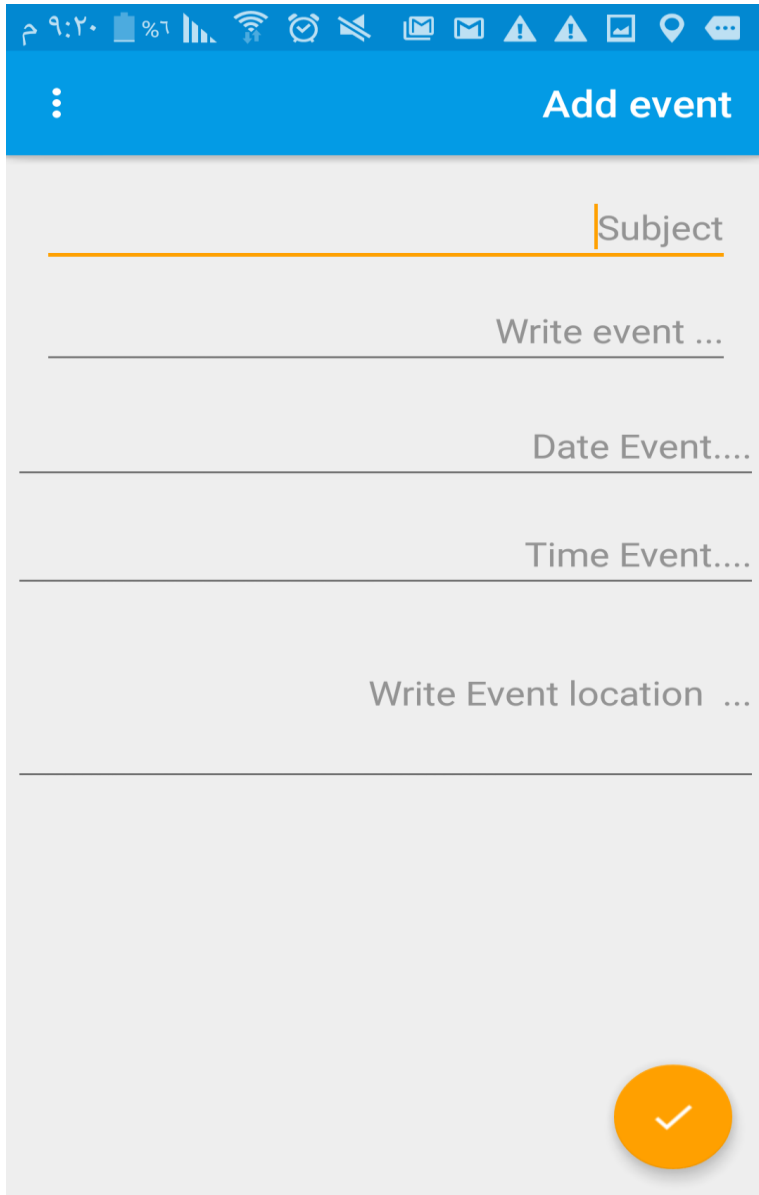


Figure 4.4.8 Supervisor Add events.



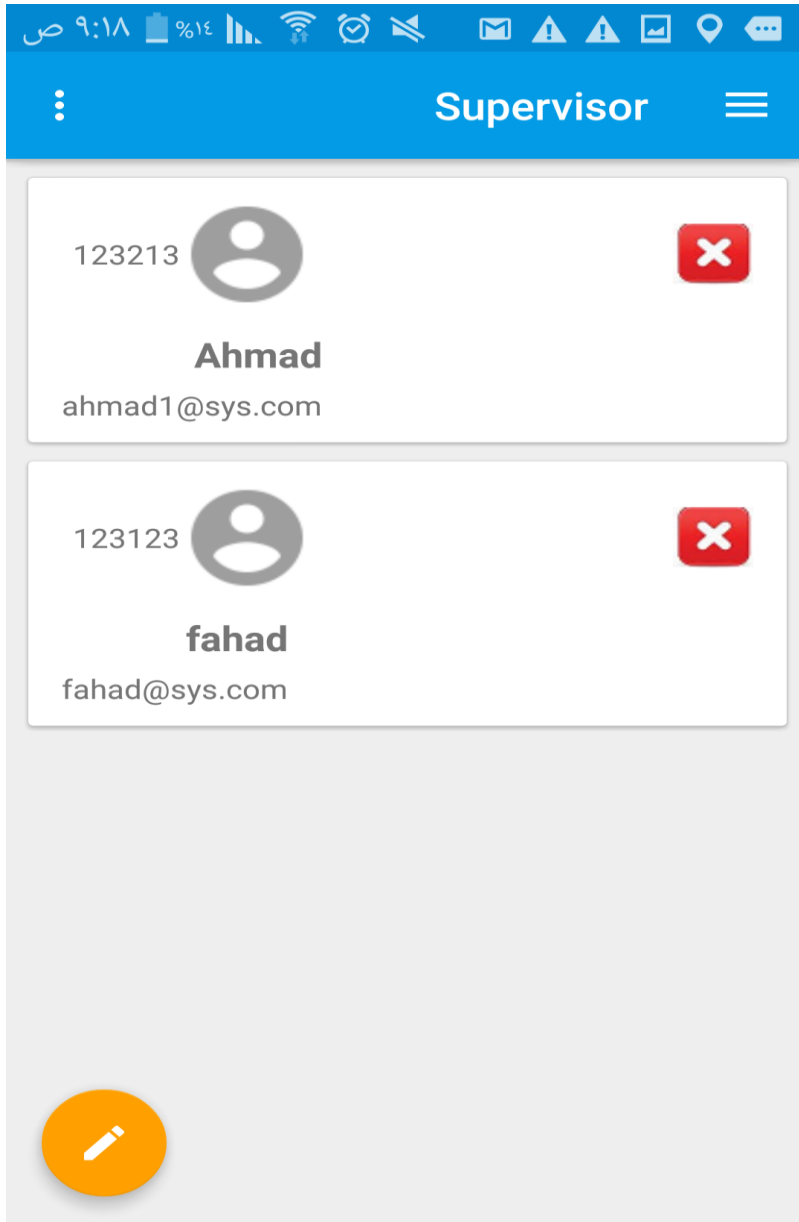


Figure 4.4.9 Supervisor Delete Pilgrims.

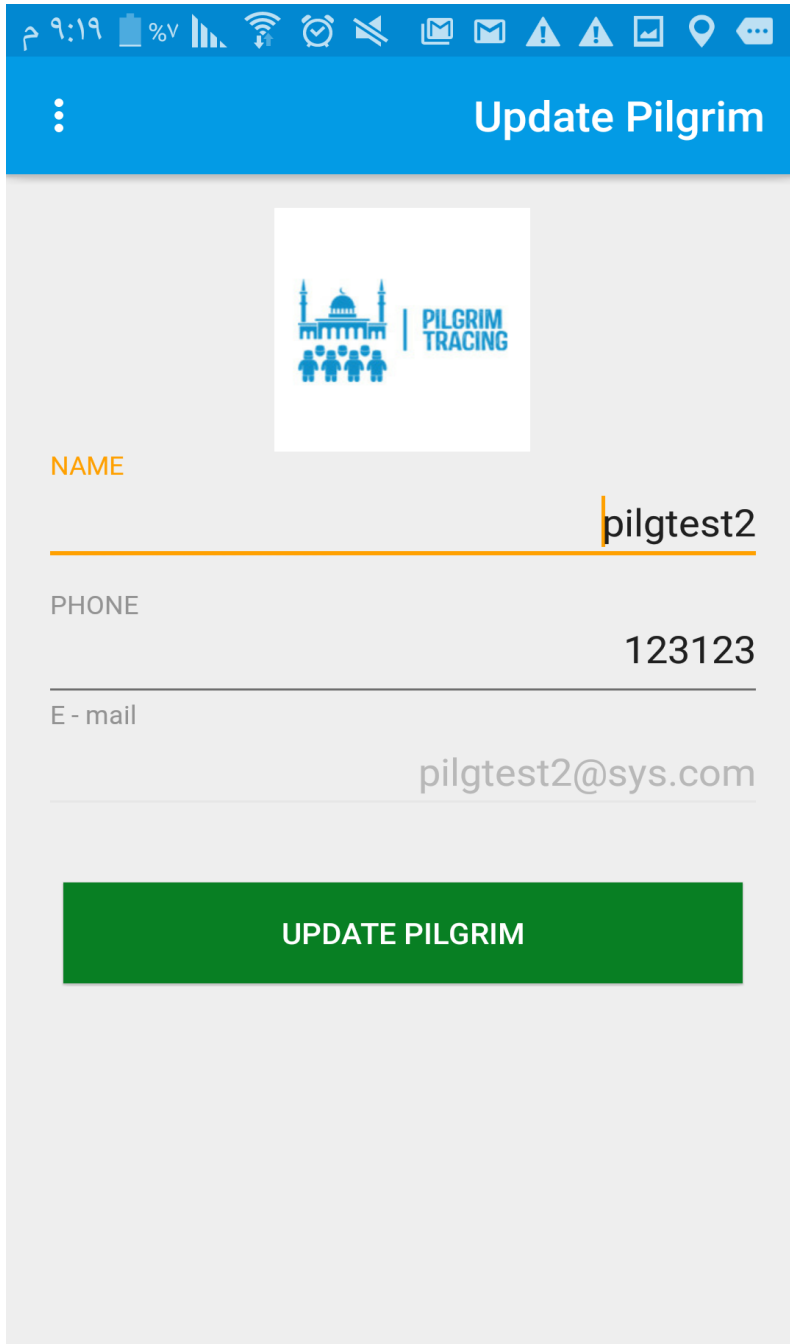


Figure 4.4.10 Supervisor Modify Pilgrims.

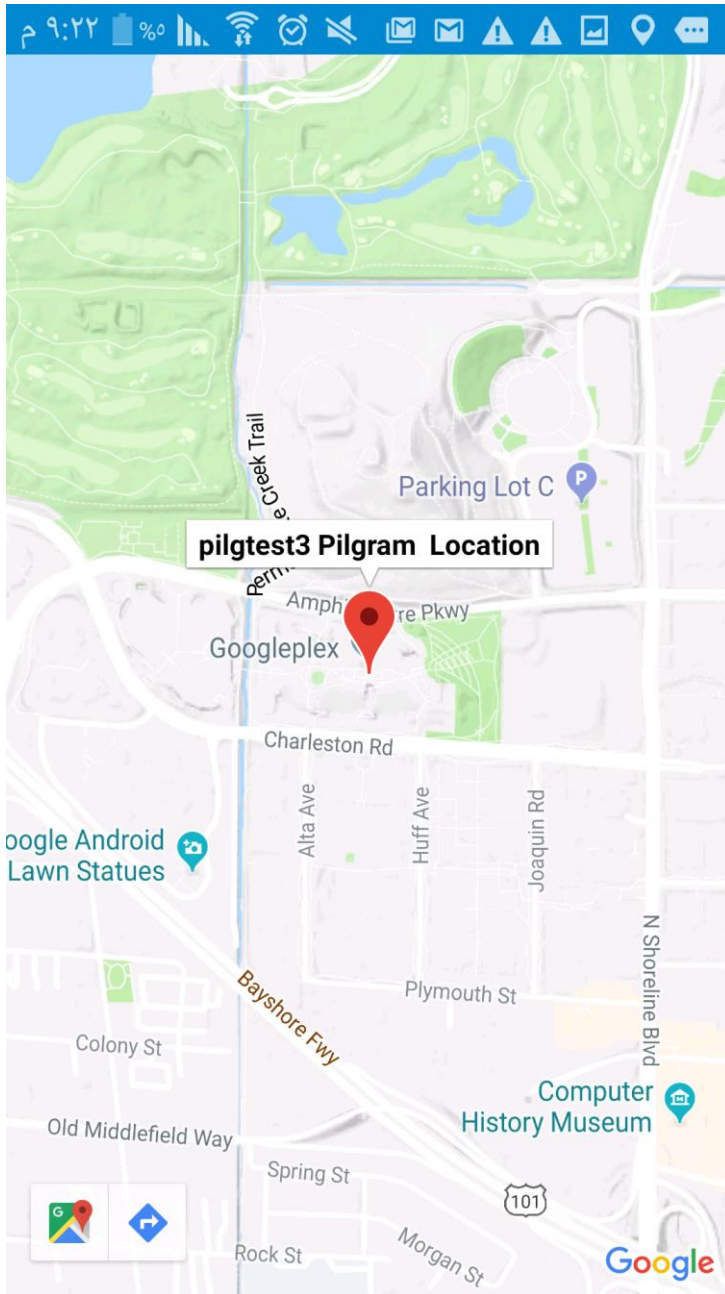


Figure 4.4.11 Supervisor Show Pilgrim Location.

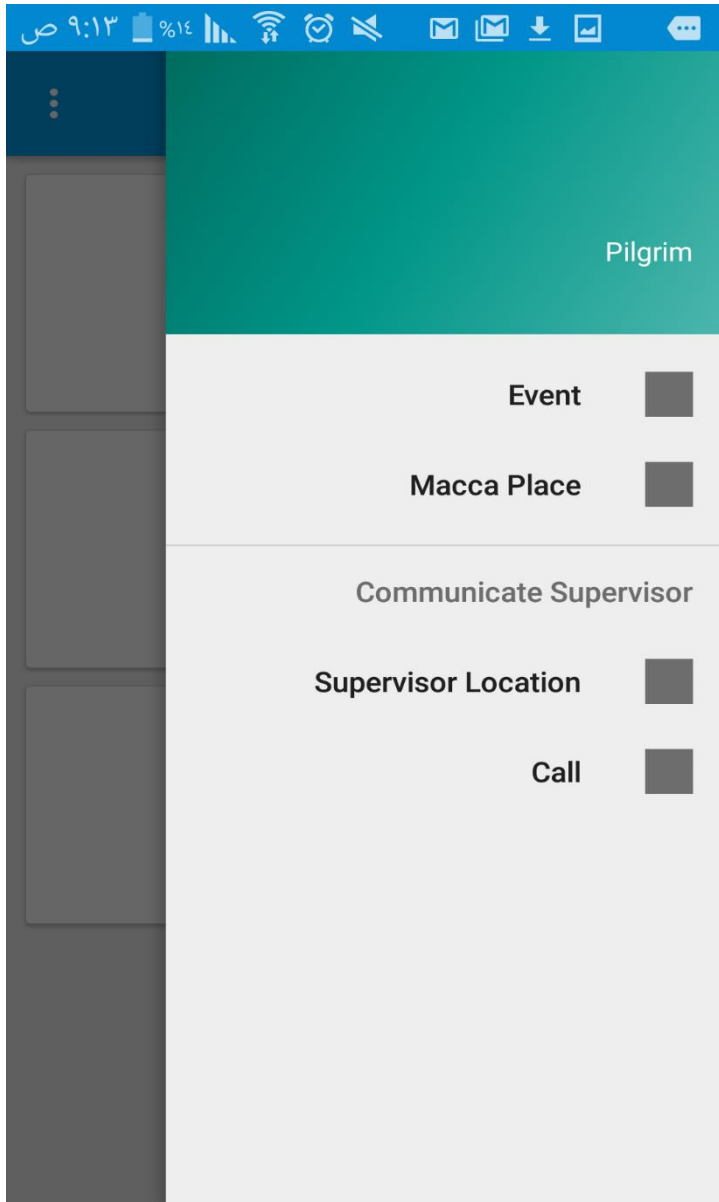


Figure 4.4.12 Pilgrims Interface.

Pilgrim can show the Event, Mecca places, supervisor location and Call the Supervisor.

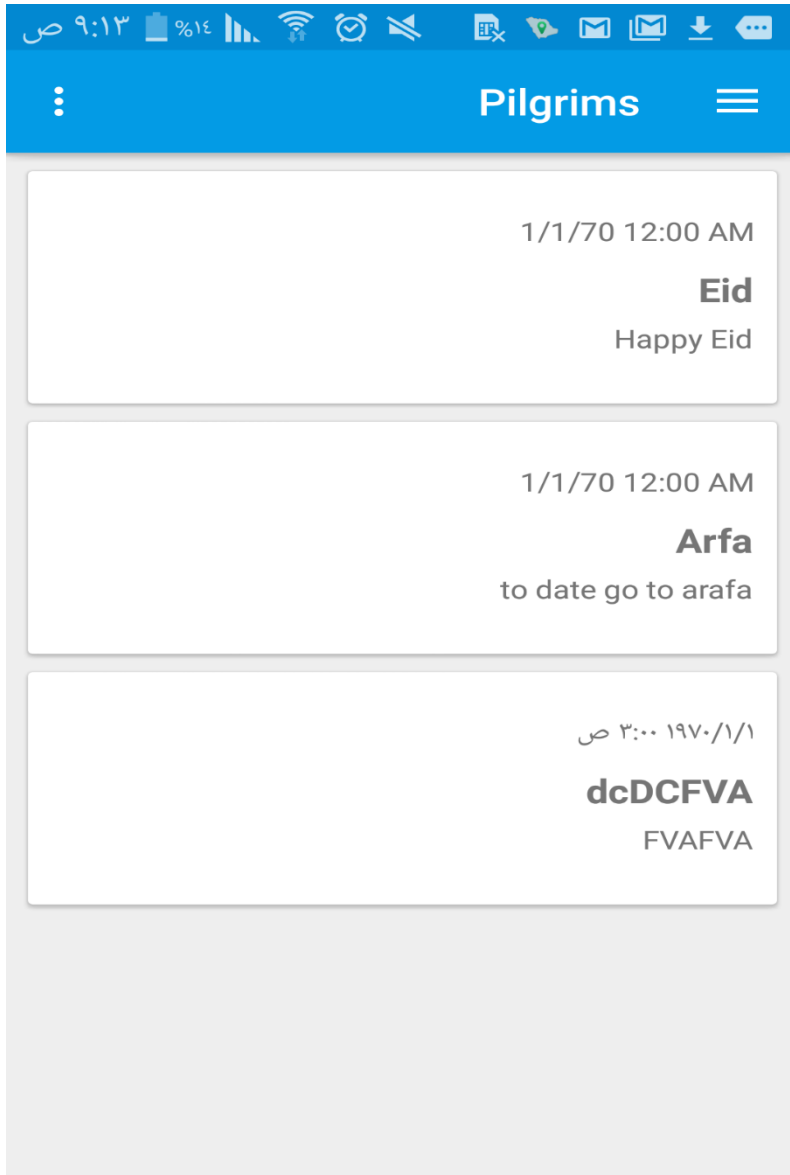


Figure 4.4.13 Pilgrim Show events.



Figure 4.4.14 Macca Location.



Figure 4.4.15 Supervisor Location.

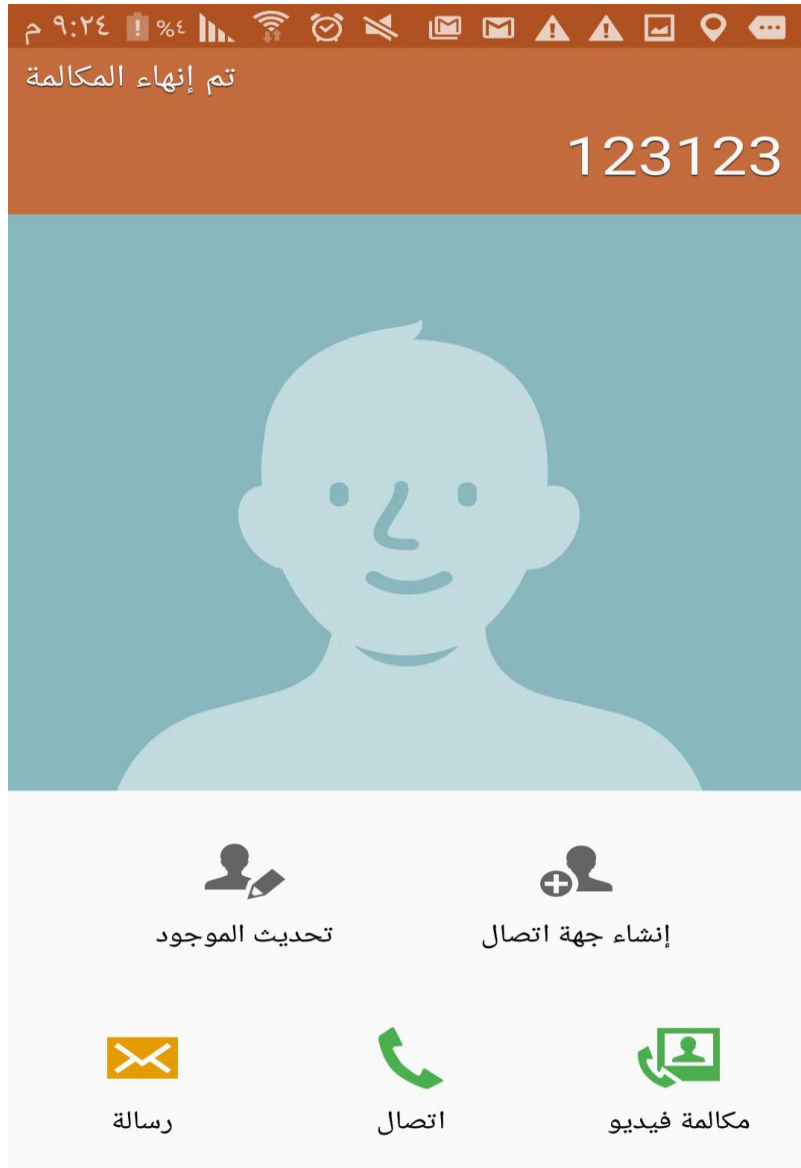


Figure 4.4.16 Pilgrim Call Supervisor.



## 4.5: Reports:

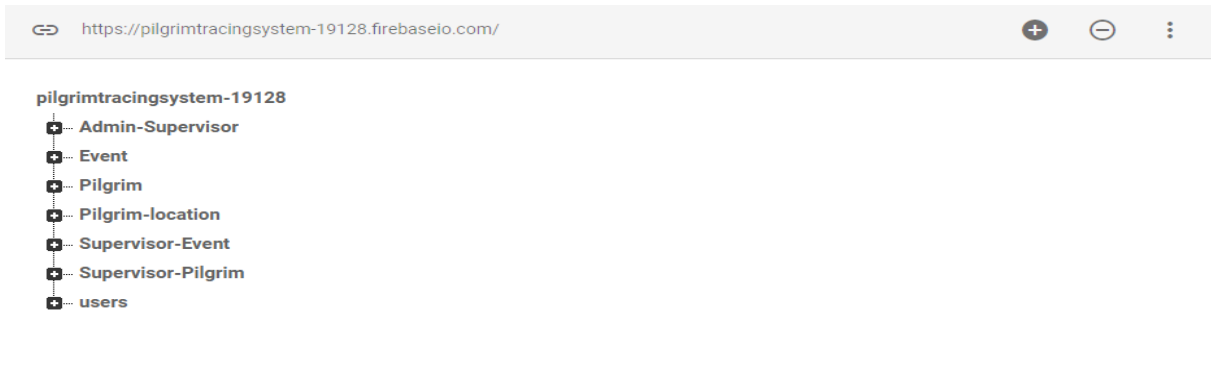


Figure 4.5.1 Pilgrim Tracing System Database

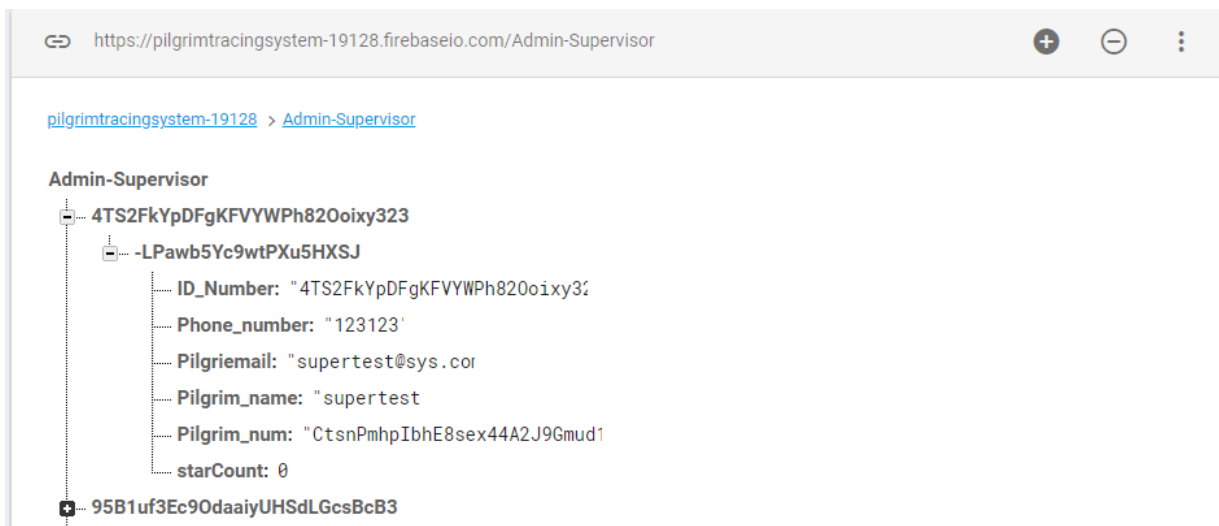


Figure 4.5.2 Admin-Supervisor database.

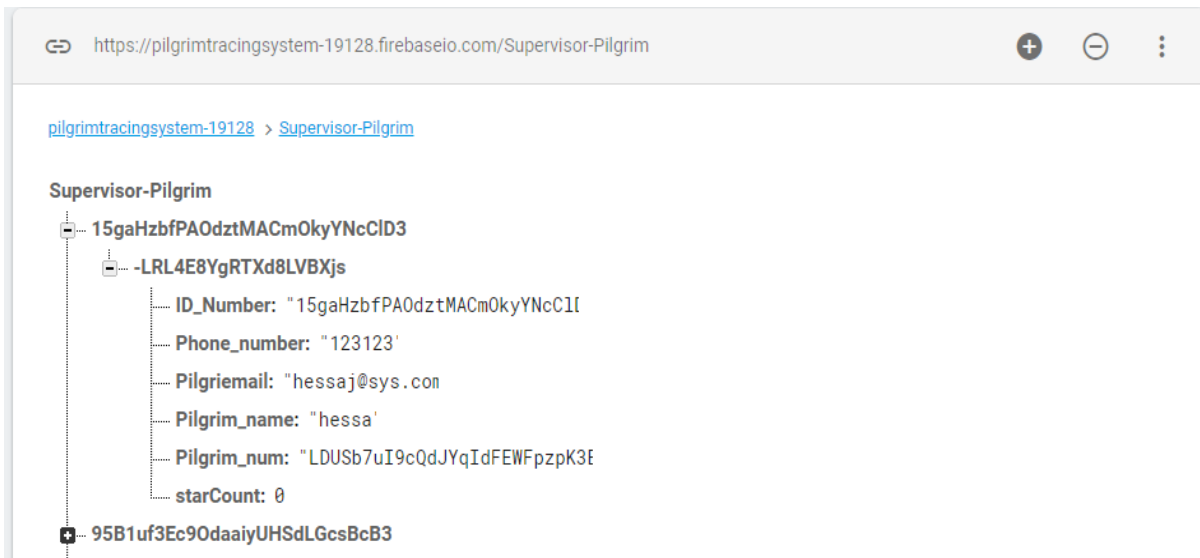


Figure 4.5.3 Supervisor-Pilgrim database

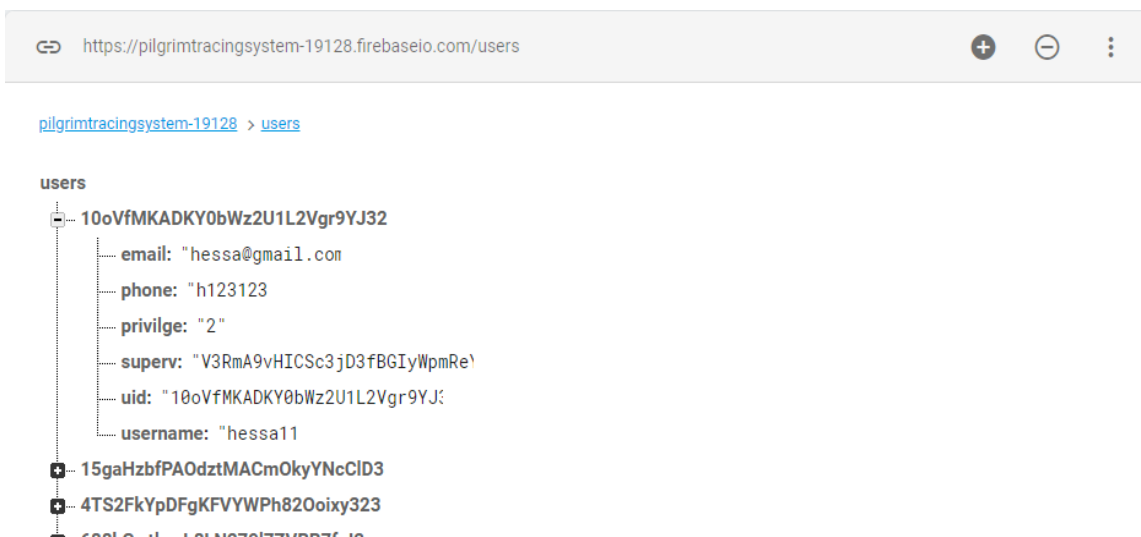


Figure 4.5.4 User (Admin, Supervisor, Pilgrim) database

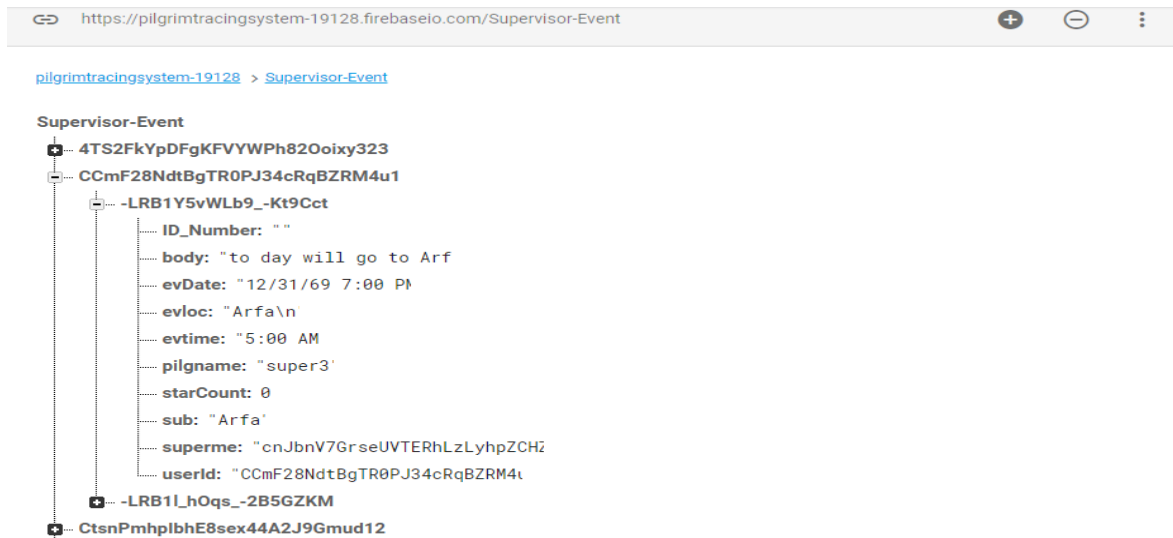


Figure 4.5.5 Event Database

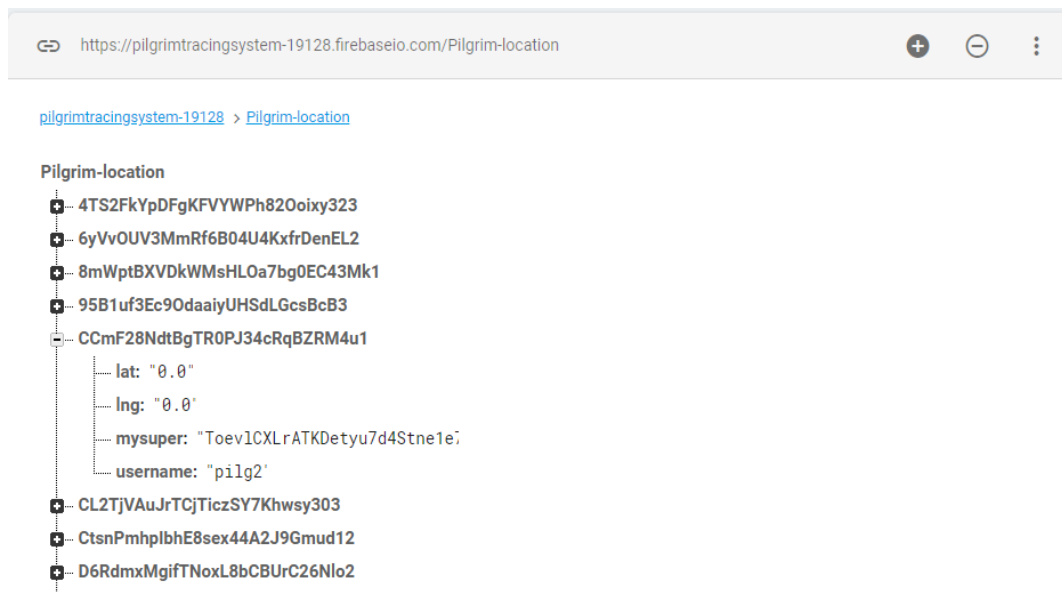


Figure 4.5.6 Location Database

Email	phone	privilge	superv	uid	username
hessa@gmail.com	512498531		2 V3RmA9vHICSc3jD3fBGlyWpmReY2	10oVfMKADKY0bWz2U1L2Vgr9YJ32	hessa11
mhamad@ssys.com	524892370		1 VtJxTdNiy5e9c8ENQnC TU9gOem13	15gaHzbfPAOdztMACmOkYnCID3	mhamad
admin@sys.com	55529826		0 " "	4TS2FkYpDFgKFVYWPh82Ooixy323	admin
supertest@sys.com	505693184		1 4TS2FkYpDFgKFVYWPh82Ooixy323	CtsnPmhplbhE8sex44A2J9Gmud12	syptest
pilgtest4@sys.com	55531938		2 CtsnPmhplbhE8sex44A2J9Gmud12	Y Tgte0GbxQcc9JBrxXIP2ifkzc2	pilgtest4

Table 4.5.1 User database

Body	evDate	evLoc	evTime	pilgname	sub	superme	uesrid
to day go to Arfa	01/09/18	Arfa	6:15Am	supertest	Arafa	4TS2FkYpDFgKFVYWPh82Ooixy323	CtsnPmhplbhE8sex44A2J9Gmud12
happy Eid	02/09/18	muzdalefa	8:00Am	supertest	Eid	4TS2FkYpDFgKFVYWPh82Ooixy323	CtsnPmhplbhE8sex44A2J9Gmud12
there is symposium in the mosqu	30/08/18	Mena	9:00Bm	super11	sympsiuim	HRhpA5w9YahRYJxATINgeibrdoX2	zwZE2eUSFiR0ZNjzPFSGXhEgdYb2

Table 4.5.2 Event Database

lat	lng	mysuper	username
37.422	122.084	HRhpA5w9YahRYJxATINgeibrdoX2	super11
26.30386	44.80952662	4TS2FkYpDFgKFVYWPh82Ooixy323	supertest
37.422	122.408	CtsnPmhplbhE8sex44A2J9Gmud12	pilgtest3
0	0	CtsnPmhplbhE8sex44A2J9Gmud13	pilgtest4

Table 4.5.3 Location Database

## **Chapter 5 Conclusions:**

- 1- After reviewing the current study and studying it thoroughly, the system was analyzed according to it and work on establishing an application that manages supervisor and pilgrim communication.
- 2- Through the use of the application, the supervisor is able to rely on it to manage and control pilgrim data, as well as to manage its location and states.
- 3- Through the use of the application, the supervisor can manage the lost Pilgrim roadmap and guide him/her to the right way.
- 4- By using the application, the system administrator can add or remove supervisor .
- 5- Pilgrim can knows his Manask roadmap easily.

## **Future works:**

Project Development.

Add message chatting between Pilgrims And Supervisor.

## References:

Ahmad Al-Akhras. 2017. *Hajj Exemplifies Equality Before God*. Retrieved from <https://www.thoughtco.com/hajj-exemplifies-equality-before-god-2004307>

Alexander. S (2017). *Structural specification: beyond class diagrams*. 21W80 Software specification and architecture.

BARBARA,H. (2008). *SYSTEM ANALYSIS AND DESIGN(5 ed pp.8)*. Don Fowley.

Bharath. P (2012). *UNIFIED MODELING LANGUAGE (UML) OVERVIEW*. Principles of Software Engineering.

MT. HOOD Community College (2018). *Hardware and Software Requirements*.

visual-paradigm. n.d. *What is sequence diagram*. Retrived from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram>

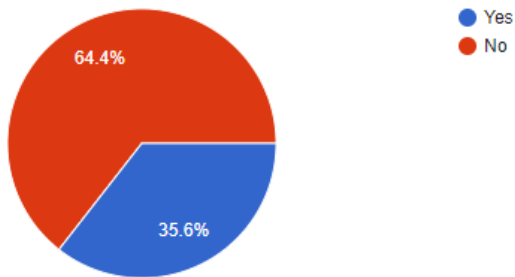
## Appendix A:

### Data collection (questionnaire method):

---

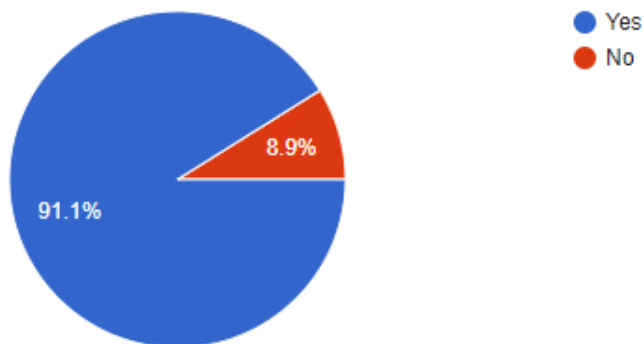
Have you ever been in Hajj?

٤٥ ردًا



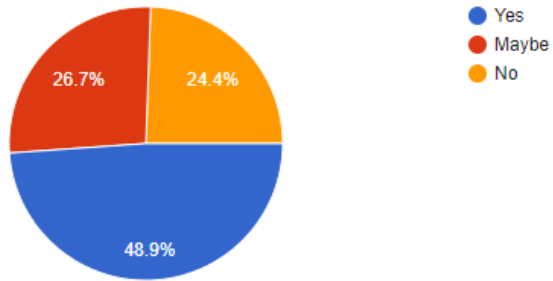
?if not, are you planning to perform Hajj in the future

٤٥ ردًا



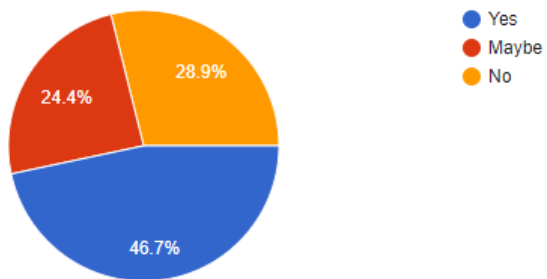
?Have you ever felt scared while doing your Hajj

٤٥ رڤا



?Did your fear of losing your way affect your Hajj

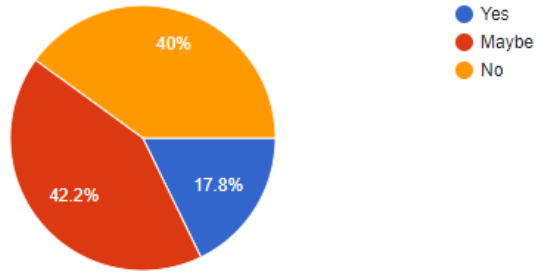
٤٥ رڤا





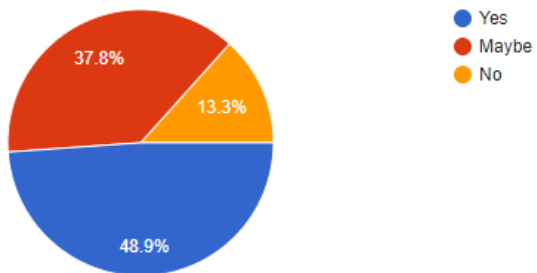
?Do you have full knowledge of Msnasek places

ردًا ٤٥



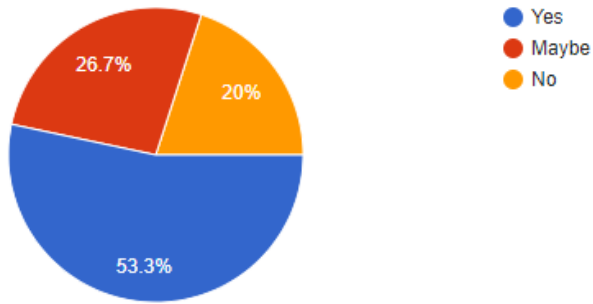
? Is the communication with the trip supervisor when needed easy

ردًا ٤٥



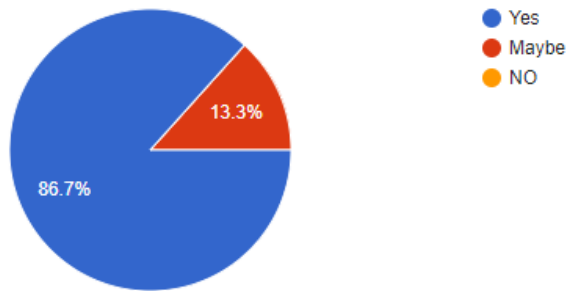
?Is the trip mentor has/have a defined place to go back to

٤٥ ردًا



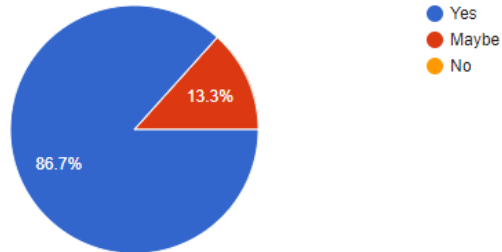
Are you feeling safe when the trip mentor watch your movements during  
?Hajj

٤٥ ردًا



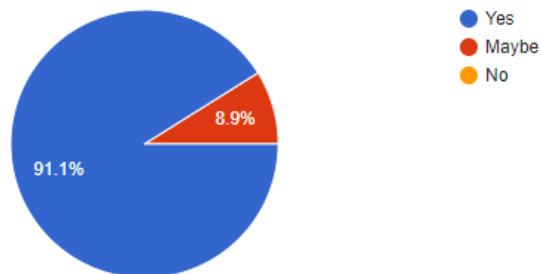
Do you agree that that supervision of the trip mentor minimize the ?number of lost cases during Hajj

٤٥ ردًا



Do you agree that having a mobile application would benefit the ?communication between pilgrims and their trip mentor during Hajj

٤٥ ردًا



## Appendix B:

### Some code:

#### Add Event:

```
private void writeNewEvent(String userId, String sub, String body, String
evDate,String evtime,String evloc,String id) {
    // Create new post at /user-posts/$userid/$postid and at
    // /posts/$postid simultaneously
    String username,superme;
    superme=User.getpublicsuperv();
    username=User.getpublicusername() ;
    String key = FirebaseDatabase.child("Event").push().getKey();
    Event event = new Event(userId, sub,
body,evDate,evtime,evloc,id,username,superme);
    Map<String, Object> postValues = event.toMap();

    Map<String, Object> childUpdates = new HashMap<>();
    childUpdates.put("/Event/" + key, postValues);
    childUpdates.put("/Supervisor-Event/" + userId + "/" + key, postValues);

    FirebaseDatabase.updateChildren(childUpdates);
}
```

#### Sign In :

```
package com.CP.Pilgrim.quickstart.PilgrimTracing;

import android.Manifest;
import android.app.AppOpsManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.Toast;

import com.CP.Pilgrim.quickstart.PilgrimTracing.models.User;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class SignInActivity extends BaseActivity implements
```

```

View.OnClickListener {
    private String LANG_CURRENT = "en";
    private static final String TAG = "SignInActivity";
    public static final int PERMISSION_ALL = 200;
    private DatabaseReference mDatabase;
    private FirebaseAuth mAuth;

    private EditText mEmailField;
    private EditText mPasswordField;
    private Button mSignInButton;
    private Button mSignUpButton;
    private Switch h;
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(com.CP.Pilgrim.quickstart.PilgrimTracing.R.layout.activity_si
gn_in);

        mDatabase = FirebaseDatabase.getInstance().getReference();
        mAuth = FirebaseAuth.getInstance();

        // Views
        mEmailField =
        findViewById(com.CP.Pilgrim.quickstart.PilgrimTracing.R.id.field_email);
        mPasswordField =
        findViewById(com.CP.Pilgrim.quickstart.PilgrimTracing.R.id.field_password);
        mSignInButton =
        findViewById(com.CP.Pilgrim.quickstart.PilgrimTracing.R.id.signinbtn);
        mSignUpButton = findViewById(R.id.signupbtn2);
        h=findViewById(R.id.swlang);
        String[] PERMISSIONS = {
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.RECEIVE_BOOT_COMPLETED, Manifest.permission.CALL_PHONE,
            Manifest.permission.INTERNET, Manifest.permission.ACCESS_NETWORK_STATE,
            Manifest.permission.RECEIVE_BOOT_COMPLETED,
            Manifest.permission.PACKAGE_USAGE_STATS,Manifest.permission.READ_PHONE_STATE
        };

        if (!hasPermissions(this, PERMISSIONS)) {
            ActivityCompat.requestPermissions(this, PERMISSIONS,
            PERMISSION_ALL);
        }

        if (Build.VERSION.SDK_INT > 20) {

            AppOpsManager appOps = (AppOpsManager)
                getSystemService(Context.APP_OPS_SERVICE);
            int mode = 0;
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                mode = appOps.checkOpNoThrow("android:get_usage_stats",
                    android.os.Process.myUid(), getPackageName());
            }
            boolean granted = mode == AppOpsManager.MODE_ALLOWED;

        }

        findViewById(R.id.swlang).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

```

```

        if (LANG_CURRENT.equals("en")) {
            changeLang(SignInActivity.this, "ar");
            User.setpubliclang( "en" );
            h.setText("EN");

        } else {
            changeLang(SignInActivity.this, "en");
            h.setTextOn("عربي");
            h.setText("عربي");
            User.setpubliclang( "en" );
        } //switch languages
        finish();
        startActivity(new Intent(SignInActivity.this,
SignInActivity.class));
    }
});
// Click listeners

    mSignInButton.setOnClickListener(new View.OnClickListener() {
@Override
        public void onClick(View view) {
            signIn();
        }
    });

    mSignUpButton.setOnClickListener(new View.OnClickListener() {
@Override
        public void onClick(View view) {
            startActivity(new
Intent(SignInActivity.this, SignUpActivity.class));
        }
    });

@Override
    public void onStart() {
        super.onStart();

        // Check auth on Activity start
        if ( mAuth.getCurrentUser() != null) {
            onAuthSuccess(mAuth.getCurrentUser());
        }
    }

    public static boolean hasPermissions(Context context, String...
permissions) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && context !=
null && permissions != null) {
            for (String permission : permissions) {
                if (ActivityCompat.checkSelfPermission(context, permission)
!= PackageManager.PERMISSION_GRANTED) {
                    return false;
                }
            }
        }
        return true;
    }
    private void signIn() {
        Log.d(TAG, "signIn");
        if (!validateForm()) {
            return;
        }

        showProgressDialog();

```

```

String email = mEmailField.getText().toString();
String password = mPasswordField.getText().toString();

 mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
task.isSuccessful());
        Log.d(TAG, "signIn:onComplete:" +
        hideProgressDialog();

        if (task.isSuccessful()) {
            onAuthSuccess(task.getResult().getUser());
        } else {
            Toast.makeText(SignInActivity.this, "Sign In
Failed",
                                Toast.LENGTH_SHORT).show();
        }
    }
});
}

private void onAuthSuccess(FirebaseUser user) {
    String username = usernameFromEmail(user.getEmail());

    // Write new user
    // writeNewUser(user.getUid(), username, user.getEmail());

    User.setpublicid(user.getUid());

    chickexist(User.getpublicid());
    // Go to MainActivity

    // startActivity(new Intent(SignInActivity.this,
MainActivity.class));
    //startActivity(new Intent(SignInActivity.this, SendService.class));
}

private String usernameFromEmail(String email) {
    if (email.contains("@")) {
        return email.split("@")[0];
    } else {
        return email;
    }
}

private boolean validateForm() {
    boolean result = true;
    if (TextUtils.isEmpty(mEmailField.getText().toString())) {
        mEmailField.setError("Required");
        result = false;
    } else {
        mEmailField.setError(null);
    }

    if (TextUtils.isEmpty(mPasswordField.getText().toString())) {
        mPasswordField.setError("Required");
        result = false;
    } else {
        mPasswordField.setError(null);
    }

    return result;
}

```

```

    private void chickexist(String uid) {

        DatabaseReference postRef =
        FirebaseDatabase.getInstance().getReference().child("users").child(uid);

        postRef.addListenerForSingleValueEvent(new ValueEventListener() {

            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                User d = dataSnapshot.getValue(User.class);
            }

            try {

                if(d.privilege.equals("0")){ // if privilege 0?
                    startActivity(new Intent(SignInActivity.this,
                    MainAdminActivity.class));
                    //MainAdminActivity interface
                    User.setpublicpriv( d.privilege );
                    User.setpublicsuperv( d.superv );
                    User.setpublicusername( d.username );
                    stopService(new
                    Intent(SignInActivity.this, SendService.class));

                    // Intent serviceIntent = new
                    Intent(SignInActivity.this, SendService.class); //
                    startService(serviceIntent);
                    finish();
                    //bus number exists in Database
                } else if(d.privilege.equals("1")){//privilege 1?
                    startActivity(new Intent(SignInActivity.this, Main2Activity.class));
                    // supervisor interface
                    // Intent serviceIntent = new
                    Intent(SignInActivity.this, SendService.class);
                    stopService(new
                    Intent(SignInActivity.this, SendService.class));
                    User.setpublicpriv( d.privilege );
                    User.setpublicsuperv( d.superv );
                    User.setpublicusername( d.username );
                    // startService(serviceIntent);
                    finish();

                }

                else if(d.privilege.equals("2")){
                    startActivity(new Intent(SignInActivity.this,
                    MainPilgrimActivity.class)); // pilgrim interfac
                    // Intent serviceIntent = new
                    Intent(SignInActivity.this, SendService.class);
                    stopService(new
                    Intent(SignInActivity.this, SendService.class));

                    // startService(serviceIntent);
                    User.setpublicpriv( d.privilege );
                    User.setpublicsuperv( d.superv );
                    User.setpublicusername( d.username );
                    finish();
                    //bus number doesn't exists.

                }

            }

        }

        catch (Exception e) {

        }

    }

    @Override

```



```

        public void onCancelled(DatabaseError databaseError) {

            });
    }

    @Override
    protected void attachBaseContext(Context newBase) {

        SharedPreferences preferences =
        PreferenceManager.getDefaultSharedPreferences(newBase);
        LANG_CURRENT = preferences.getString("Language", "en");

        super.attachBaseContext(MyContextWrapper.wrap(newBase,
        LANG_CURRENT));
    }

    public void changeLang(Context context, String lang) {
        SharedPreferences preferences =
        PreferenceManager.getDefaultSharedPreferences(context);
        SharedPreferences.Editor editor = preferences.edit();
        editor.putString("Language", lang);
        editor.apply();
    }
    @Override
    public void onClick(View v) {
        int i = v.getId();
        if (i == com.CP.Pilgrim.quickstart.PilgrimTracing.R.id.signinbtn) {
            signIn();
        }
    }
}

```