

# **MIRINO**

## **Android Application**

### **A Project Report**

Submitted in partial fulfillment of the  
Requirements for the award of the Degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

Lareb Faizan

Seat No :- 3035033

**Under the esteemed guidance of**

**Prof. Shahid Ansari**

Assistance Professor, Maharashtra College

**Prof. Saima Shaikh**

Head Of I.T. Department, Maharashtra College



**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**MAHARASHTRA COLLEGE OF ARTS, SCIENCE AND COMMERCE**  
*(Affiliated to University of Mumbai)*

**MUMBAI, 400008**

**MAHARASHTRA**

**2018-2019**

**PROFORMA FOR THE APPROVAL PROJECT PROPOSAL**

**PNR No.: 2016016401282405.**

**Roll no: 321**

1. Name of the Student

LAREB FAIZAN

2. Title of the Project

MIRINO (Android Application).

3. Name of the Guide

Prof. Shahid Ansari

4. Teaching experience of the Guide \_\_\_\_\_

5. Is this your first submission?

Yes

☐

No

☐

Signature of the Student

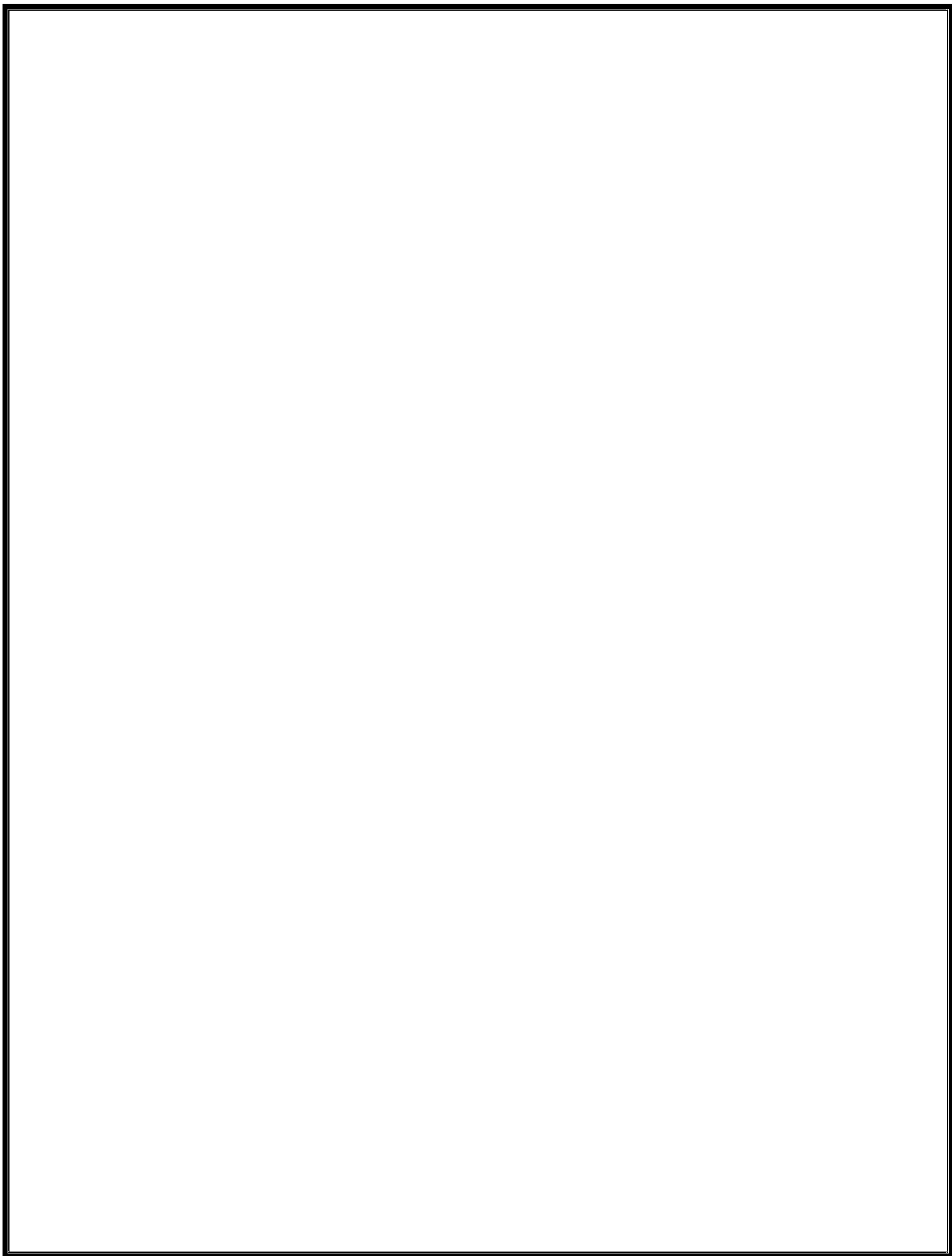
Signature of the Guide

Date: 18 -04 - 2019

Date: .....

Signature of the coordinator

Date: .....



## MAHARASHTRA COLLEGE OF ARTS, SCIENCE AND COMMERCE

*(Affiliated to University of Mumbai)*

**MUMBAI-MAHARASHTRA-400008**

## DEPARTMENT OF INFORMATION TECHNOLOGY



## CERTIFICATE

This is to certify that the project entitled, "**MIRINO** ", is bonafied work of **LAREB FAIZAN** bearing Seat.No: 321 submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

## Internal Guide

## Coordinator

## External Examiner

**Date:**

### College Seal

# **ABSTRACT**

In the field of Religious Services, the existing applications are discrete and broken modules that do not cover all aspects of this field. It have limited functionalities for searching and getting detail's of mosque or getting trust? committee contact. But none of the existing system provide a integrated model for all these functionalities.

This system (MIRINO) is an Android Application which is an absolute complete and integrated application with all the related services of the same field provided in a single proposed system. It can be used for searching Mosque and getting their details. All operation would be done correctly and it ensures that whatever information is coming from the database is accurate. The level of accuracy in the proposed system will be higher. The Proposed System is more dynamic and efficient than the existing system

# ACKNOWLEDGEMENT

I undersigned, have great pleasure in giving my sincere thanks to those who have contributed their valuable time in helping me to achieve the success in my project work. My heartfelt thanks to The Principle of the college **PROF.SIRAJUDDIN CHOUGLE** and the IT Department of College for helping in the project with words of encouragement and has shown full confidence in our abilities.

I would like to express my sincere thanks to **PROF.SAIMA SHAIKH** head of I.T Department for her constant encouragement, which made this project a success.

I am indebted and thankful to my Project Guide **PROF.SHAHID ANSARI** to Whom I owe his piece of knowledge for his valuable and timely guidance, co-operation, encouragement and time spent for this project work. I would also like to thank our IT staff for providing us sufficient information, which helped us to complete our project successfully. My sincere thanks to the Library staff for extending their help and giving me all the books for reference in a very short span of time.

I also thank MY PARENTS and all my family members for their continued Support, without their support this project would not be possible.

## DECLARATION

I here by declare that the project entitled, “**MIRINO**” done at **Maharashtra College Of Arts, Science and Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

**Lareb Faizan**

# TABLE OF CONTENTS

<b>Chapter 1: Introduction</b>	<b>01-05</b>
<b>1.1 Background</b>	<b>02</b>
<b>1.2 Objectives</b>	<b>03</b>
<b>1.3 Purpose and Scope</b>	<b>04-05</b>
1.2.1 Purpose	04
1.2.2 Scope	05
 <b>Chapter 2: System Analysis</b>	 <b>06-15</b>
<b>2.1 Existing System</b>	<b>07</b>
<b>2.2 Proposed System</b>	<b>07</b>
<b>2.3 Requirement Analysis</b>	<b>08-09</b>
<b>2.4 Functional Requirements</b>	<b>10</b>
<b>2.5 Hardware Requirements</b>	<b>11</b>
<b>2.6 Software Requirements</b>	<b>12</b>
<b>2.7 Justification of selection of Technology</b>	<b>13-15</b>
 <b>Chapter 3: System Design</b>	 <b>16-41</b>
<b>3.1 Gantt chart</b>	<b>17-20</b>
<b>3.2 Entity Relationship Diagram</b>	<b>21-22</b>
<b>3.3 Use Case Diagram</b>	<b>23-24</b>
<b>3.4 Sequence Diagram</b>	<b>25-27</b>
<b>3.5 Activity Diagram</b>	<b>28-33</b>
<b>3.6 Deployment Diagram</b>	<b>34-35</b>
<b>3.7 Class Diagram</b>	<b>36-38</b>
<b>3.8 Data Flow Diagram</b>	<b>39-40</b>
<b>3.9 Event Table</b>	<b>41</b>
 <b>Chapter 4: Implementation and Testing</b>	 <b>42-53</b>
<b>4.1 Code</b>	<b>44-45</b>
<b>4.2 Testing Approach</b>	<b>45-50</b>
4.2.1 Unit Testing	45-49
4.2.2 Integration System	50
<b>4.3 Test Cases, Test data And Test Result</b>	<b>51-53</b>



## **List of Tables**

<b>No.</b>	<b>Table Name</b>	<b>Page No.</b>
1.	Hardware Requirement Specification	11
2.	Software Requirement Specification	11
3.	Software Requirement (User)Specification	11
4.	Gantt Chart	20
5.	Components Of ER Diagram	21
6.	Components Of Use Case Diagram	23
7.	Components Of Sequence Diagram	25
8.	Components Of Activity Diagram	28
9.	Components Of Deployment Diagram	34
10.	Components Of Data Flow Diagram	39
11.	Event Table	41

## **List of Figures**

<b>No.</b>	<b>Diagram Name</b>	<b>Page No.</b>
1.	Entity Relationship Diagram	22
2.	Use Case Diagram	24
3.	Sequence(user) Diagram	26
4.	Sequence (Admin) Diagram	27
5.	LogIn Activity Diagram	29
6.	Registration Activity Diagram	30
7.	Search Mosque Activity Diagram	31
8.	Search Mosque Activity Diagram	32
9.	Admin Activity Diagram	33
10.	Deployment Diagram	35
11.	Class Diagram	38
12.	Data Flow Diagram	40

# **INTRODUCTION**

# **1.INTRODUCTION**

## **1.1 Background**

The project 'MIRINO' will be an application which runs on the Android Platform. The Name "MIRINO" Is an ITALIAN Word for Finder or Route Finder. This Application will provide all services related to Religious Places. No matter where the user is, the app will automatically detect the current location of User and provide services like details of Places such as Mosque (detail's of Mosque, including NAMAZ timing and Trustees /Committee detail's) .

These services will be choices in the app and the user will filter through choices to access a service. All possible Details like Contact Number, Timings of NAMAZ will be provided. Besides this, the User can see Mosque List. Thus, this Application will be a one stop for Users to avail all the Religious Places.

Also the user can set Recite and listen Quran. The existing Applications are broken modules and no such complete Application exist that cover all aspects of the field. The Existing Apps are only used to search all Places, we don't have any specific Application for this. also it is difficult to retrieve Information from existing systems(detail's like Namaz Timings, Quran Recite Listen, etc).

Sometimes User will want to add place in the Application , if User knows any of the place, which is not in MIRINO'S Database. User can request admin to add that place by filling the request form in the Application after sign in. after that Admin will send verification to the trust or that Place and after the Verification the particular requested place will be added to the MIRINO's database.

The System which Proposed is absolute complete application with all the Related services of the same field Provided in a single Proposed system. it can be used for Searching Mosque and Getting their details of NAMAZ timings.the level of accuracy in the Proposed system will be higher.

All Operation would be done correctly and it ensures that whatever Information is coming from the Database is Accurate.Making changes in Proposed System will be an easier task as compared to Existing Systems so that they can be Updated easily.

## **MIRINO (Android Application)**

The system should be easy to Operate and should be such that it can be Developed within a Short period and fit in the Limited budget of the user. Retrieving Information will be easier in Proposed system and it will be very user friendly.

### **1.2 Objectives**

As the use of Android mobile phone is drastically increasing and the people have an easy access to it so from that we found an idea to make an application on Android phone that will assist people in a better way to avail or look for Religious Places around them just on an Android phone.

The main objective is to provide ease to users to look for Religious Places around them.

This app can be very helpful to one who needs to find services like search for Mosque and the user can also contact the same organizations as the details like contact numbers and timings will be provided in application.

The application will have different filtering options that the user will choose and for all the filters corresponding services will be shown.

One of the objective is to Design and implement the Database that will contain details of all Mosque like Locations, Contact Numbers, etc.

Our focus is on providing the following details and services to user around him as per his/her location:

- Finding route for Mosque, and all other information related to that.
- Trust and Committee Detail's.
- Setting reminders for Namaz and Reciting Quran.

### **1.3 Purpose and Scope**

#### **1.3.1 Purpose**

The purpose to develop this application is to facilitate users with all the Mosque services on their fingertips.

As technology progressed, our phones are getting smarter and their computing power increased with a decline in their prices. We now have as much processing power in devices we carry in our hands that we had in huge computers a few years ago. And as we all know, with great power comes great responsibility. Now, we had to think of ways to consume that much processing power in creative ways to make our life easier and that too securely.

Therefore, I took this project of creating a Mobile Application for the most essential Places i.e. Mosque .

As operating systems like Android and iOS have become widespread, the development of apps became easier. We could now just build a single app for the platform that could be downloaded and installed into millions of phones across the world and for what purpose, well, our creativity is the limit. We now have a huge community of developers that for all the major platforms that develop app and earn their living. We have an open play store from where we can download the apps and extend the functionality of our mobile phones.

Since ,Android is the most widely used platform for mobile phones so, I chose to develop my application for an android platform so that maximum people can use this application.

### **1.3.2 Scope**

The proposed software product i.e. 'MIRINO' is an android application. Therefore, only android platform users will be able to use this software Application.

Also android is an open source platform and the most widely used platform for mobile phones hence, a wide range of people can use this Application.

This application will be developed using Google Maps and Places API, therefore will access location of your cell phone. So, the user needs to grant permissions to the app for accessing location so that app may run in their Android mobile phones.

Reasons for 'why our project is an android mobile application' are as follows:

- Most of the companies are working to improve mobile app experience.
- People are spending most of the time on Mobile nowadays.
- Mobile phone sizes are becoming Bigger, we work on mobile most of the time.
- Android Platform is developer friendly. Developers are free to build anything on Android platform.

As the application will have a user friendly GUI, a user with basic experience in use of android may be able to use the application with ease.

Main Functions of application include:

- Searching for Mosque.
- Getting these Mosque information for NAMAZ.
- You can Find A Trusted Person To Donate The Money TO the Committee or Trust.
- Allowing users to add information and authenticating the same to add it to the application. For example, a user may request to add details for a new Mosque recently opened around him. The admin will authenticate details and add the same to a database so that every user can access the same. (Future Update).
- Listen and recite Quran.

# **SURVEY OF TECHNOLOGIES**



## **2. SURVEY OF TECHNOLOGIES**

### **2.1 Existing System**

In the field of Religious Services, the existing applications are discrete and broken modules that do not cover all aspects of this field. It have limited functionalities for searching and getting detail's of mosque or getting trust? committee contact. But none of the existing system provide a integrated model for all these functionalities.

The existing systems are less dynamic as well as time consuming.

Also it is difficult to retrieve information from existing systems. Making changes is a difficult task due to which existing systems are not updated as per the requirements.

### **2.2 Proposed System**

The proposed system is an Android Application which is an absolute complete and integrated application with all the related services of the same field provided in a single proposed system.

It can be used for searching Mosque and getting their details.

All operation would be done correctly and it ensures that whatever information is coming from the database is accurate. The level of accuracy in the proposed system will be higher.

The Proposed System is more dynamic and efficient than the existing system

### **2.3 Requirement And Analysis**

#### **Step 1: Develop Requirements**

The first step is to gather, analyze and develop requirements from the Concept of Operations (CONOPS), stakeholder needs, objectives and other external requirement. Once requirements are documented, they are prioritized, de-conflicted, and validated with the stakeholders.

#### **Step 2: Write and Document Requirements**

The second step focuses on writing down the functional and performance requirements into the appropriate requirements documents; Initial Capabilities Document (ICD), Capability Development Document (CDD), and Capability Production Document (CPD). Requirements must be documented in order to establish a requirements baseline to start building a system and manage any changes. Requirements can be developed using the Capability Development Tracking and Manager (CDTM) tool for DoD programs.

#### **Step 3: Check Completeness**

The third step is to check that a complete set of requirements have been developed and documented that defines all system functions that are needed to satisfy the stakeholder needs with their associated performance, environmental, and other non-functional requirements. Requirement Tracing is a big tool in this step.

#### **Step 4: Analyze, Refine, and Decompose Requirements**

Requirements Analysis is the first major step in the Systems Engineering Process. This step examines each requirement to see if it meets the characteristics of a good requirement. Each requirement is then decomposed into a more refined set of requirements that are allocated to sub-systems and documented in the Weapons System Specification (WSS). Newly derived requirements are expected to emerge from this process, which continues until all requirements are defined and analyzed.

## **MIRINO (Android Application)**

### **Step 5: Validate Requirements**

In step five each requirement must be verified and validated to ensure that these are the correct requirements. This ensures that the requirements meet the overall objective of the system and all stakeholder needs.

### **Step 6: Manage Requirements**

In step six the requirements have been accepted and a baseline is established by the stakeholders. Any changes to the requirements are controlled using a Configuration Management process.

### **2.4 Functional Requirements**

The Software that we are making has the following Functional Requirements.

- Functions of application:
  - Provide list.
  - Provide Results.
  - Add Mosque.
  - Trace user activity.
  - Recite and Listen Quran.
  
- Functions of admin:
  - Delete User.
  - Add an Mosque.
  - Delete an Mosque.
  - Maintain database on server.
  - Update and maintain applications as per end user requirements
  
- User functionality :
  - Doing Request To add Mosque
  - Getting info of Mosque. ( Area, Namaz timings)
  - Getting Route for the selected mosque.
  - Setting Reminders to recite Quran.
  - Getting Namaz Timing.

### 2.5 Hardware Requirements.

Hardware requirements for development process.

Criterion	Description
OS version	Windows 7 or later
RAM	8 GB RAM recommended; plus 1 GB for the Android Emulator
Disk space	500 MB disk space for Android Studio, at least 1.5 GB for Android SDK, emulator system images, and caches
Screen resolution	1280×800 minimum screen resolution.

Table 2.1 :Hardware Requirement Specification

### 2.6 Software Requirements.

Java version	Java Development Kit (JDK) 8, use of bundled Open JDK (Version 2.2 and later) is recommended.
Android Studio	Android studio Version 3.x.

Table 2.2 :Software Requirement Specification

Hardware and software requirements for Implementation process. (Requirement for end user)

Smart phone	The smart phone should be an Android phone so that we can run .apk application Having Android Version Greater Than 4.4(kit-Kat)
-------------	---

Table 2.3 :Software Requirement(User) Specification

### 2.7 Justification of Platform

#### Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. [The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.2, which was released in September 2018.

Here we are using Android studio to develop our application because of the following reasons:

- Faster Deployment of Fresh Builds. Bringing incremental changes to an existing app code or resource is now easier and faster.
- More Accurate Programming.
- Faster Programming and Testing.
- Inclusive App Development.
- Better App Indexing.

## **MIRINO (Android Application)**

### **Android Platform**

Whether you like it or not, Android is the no. 1 mobile platform in the world, with 86 percent of smartphones and 65 percent of tablets running on this mobile platform. And since Android applications are built with Java, this level of market share establishes this technology as a mainstay in mobile development.

Java, the language and the platform, owes much of its fame and longevity to the libraries, frameworks and tools which together make up its ecosystem. No other programming language has been able to match the support that a rich ecosystem like the JVM has achieved.

Despite being over 20 years old, Java is still one of the most widely-used programming languages. Just look at the stats: according to the 2017 STACK Overflow Developer, Survey Java is the third most popular technology in the world.

The TIOBE index, which is a ranking based on the number of skilled engineers worldwide, courses, and third-party vendors, shows an even more impressive score, placing Java in the first position. Looking at the results of the past 15 years, Java has consistently ranked as either the first or second most popular language. Considering such massive popularity and the thousands of video tutorials, in-depth textbooks, online courses, and offline coding schools that offer free or affordable Java training to anyone willing to learn. Most importantly, Java offers a wide range of libraries that solve most of the common problems that enterprise applications need to solve. In many cases, there are a few good options to choose from when addressing a particular issue. And more often than not, these options are free and open source under a business-friendly license.

Google, Oracle, IBM, Philips, Facebook, Netflix, Spotify, eBay, and Uber are just a few of the larger players that utilize Java. And quite frankly, you'll have a hard time finding an enterprise that doesn't rely on Java for application development.

### Why JAVA?

Java provides backwards compatibility. First Sun, the original developer of Java, and then Oracle have dedicated special effort to make sure that code written for one iteration of Java will run unchanged on newer ones. This consistency makes Java very compelling for developers and enterprises alike. No one wants to take code that works perfectly fine and rewrite it every time a new language version comes out.

Java's readability, speed, and performance are hard to beat In this day and age, speed is everything. Just look at this Twitter case study by GoJava. Twitter wouldn't be able to handle 6,000 tweets per second if it hadn't migrated from Ruby on Rails to a JVM. Java's just-in-time compiler allows it to remain one of the fastest language/implementation combinations available today. If scalability and performance are your goals, Java is an obvious choice.

### SQL Lite:

I Have Selected SQL Lite Because It has Many Feature Which Suits my Application in All Manner:

- **Zero-Configuration:** SQLite does not need to be "installed" before it is used. There is no "setup" procedure. There is no server process that needs to be started, stopped, or configured. There is no need for an administrator to create a new database instance or assign access permissions to users. SQLite uses no configuration files. Nothing needs to be done to tell the system that SQLite is running. No actions are required to recover after a system crash or power failure. There is nothing to troubleshoot.
- **Serverless:** Most SQL database engines are implemented as a separate server process. Programs that want to access the database communicate with the server using some kind of interprocess communication (typically TCP/IP) to send requests to the server and to receive back results. SQLite does not work this way. With SQLite, the process that wants to access the database reads and writes directly from the database files on disk. There is no intermediary server process. Most SQL database engines are client/server based. Of those that are serverless, SQLite is the only one that this author knows of that allows multiple applications to access the same database at the same time.



### 2.8 Planning and Scheduling

Planning and scheduling are distinct but inseparable aspects of managing the successful project. The process of *planning* primarily deals with selecting the appropriate policies and procedures in order to achieve the objectives of the project. Scheduling converts the project action plans for scope, time cost and quality into an operating timetable. The translating of the project criteria for scope, time, cost, and quality and the requirements for human resources, communications, risk and procurement into workable “machinery” for the project team a critical interface juncture for the project team. Taken together with the project plan and budget, the schedule becomes the major tool for the management of projects. In addition, the integrated cost-time schedule serves as the fundamental basis for monitoring and controlling project activity throughout its life cycle.

This basic level paper addresses the integrated processes of planning and scheduling of multifacet / multidisciplinary programs. The paper presents a working level summary of the major Project Management topics involved in the planning process. The paper also details a systematic process for transforming the Project Plan into the Schedule and the use of the Project Schedule as a model for project control. Intended for the project management novice, the paper concludes with a suggested professional development scheme.


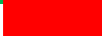














# **SYSTEM DESIGN**

## 3.SYSTEM DESIGN



















### 3.1 Gantt Chart

Sr. No.	Tasks	SEPTEMBER					OCTOBER					NOVEMBER			
		1 0	1 7	2 4	3 0	0 3	1 0	1 7	2 4	31		0 7	1 4	21	28
1.	Introduction														
1.1	Background														
1.2	Objectives														
1.3	Purpose, Scope .														
1.3.1	Purpose														
1.3.2	Scope														

## MIRINO (Android Application)

Sr. No.	Tasks	SEPTEMBER				OCTOBER					NOVEMBER			
		1 0	1 7	2 4	3 0	0 3	1 0	1 7	2 4	3 1	0 7	1 4	2 1	2 8
2.	<b>SURVEY OF TECHNOLOGIE</b>													
2.1.	Existing System	 												
2.2	Proposed System			 										
2.3	Requirements And Analysis				 									
2.4	Requirements Specification					 								
2.5	Software Requirements							 						
2.6	Hardware Requirements								 					
2.7	Justification Of Platform										 			
2.8	Planning And Scheduling												 	

## MIRINO (Android Application)

SR. NO	TASKS	DECEMBER				JANUARY				FEBRUARY			
		0 9	1 6	2 3	3 0	0 6	1 3	2 0	2 7	0 4	1 1	1 8	2 1
3	<b>SYSTEM DESIGN</b>												
3.1	Gantt Chart												
													
3.2	Entity Relationship Diagram												
													
3.3	Use Case Diagram												
													
3.4	Sequence Diagram												
													
3.5	Activity Diagram												
													
3.6	Deployment Diagram												
													
3.7	Class Diagram												
													
3.8	Data Flow Diagram												
													
3.9	Event Table												
													

## MIRINO (Android Application)

SR. NO	TASKS	DECEMBER				JANUARY				FEBRUARY			
		0 9	1 6	2 3	3 0	0 6	1 3	2 0	2 7	0 4	1 1	1 8	2 1
4	System Design												
4.1	Code												
4.2	Testing Approach												
4.2.1	Unit Testing												
4.2.2	Integration Testing												
4.3	Test Cases and Results												

Table 3.1 :Gantt Chart

### 3.2 Entity Relationship Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

#### The components of an ER diagram


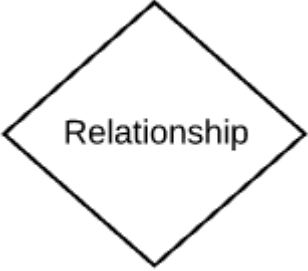

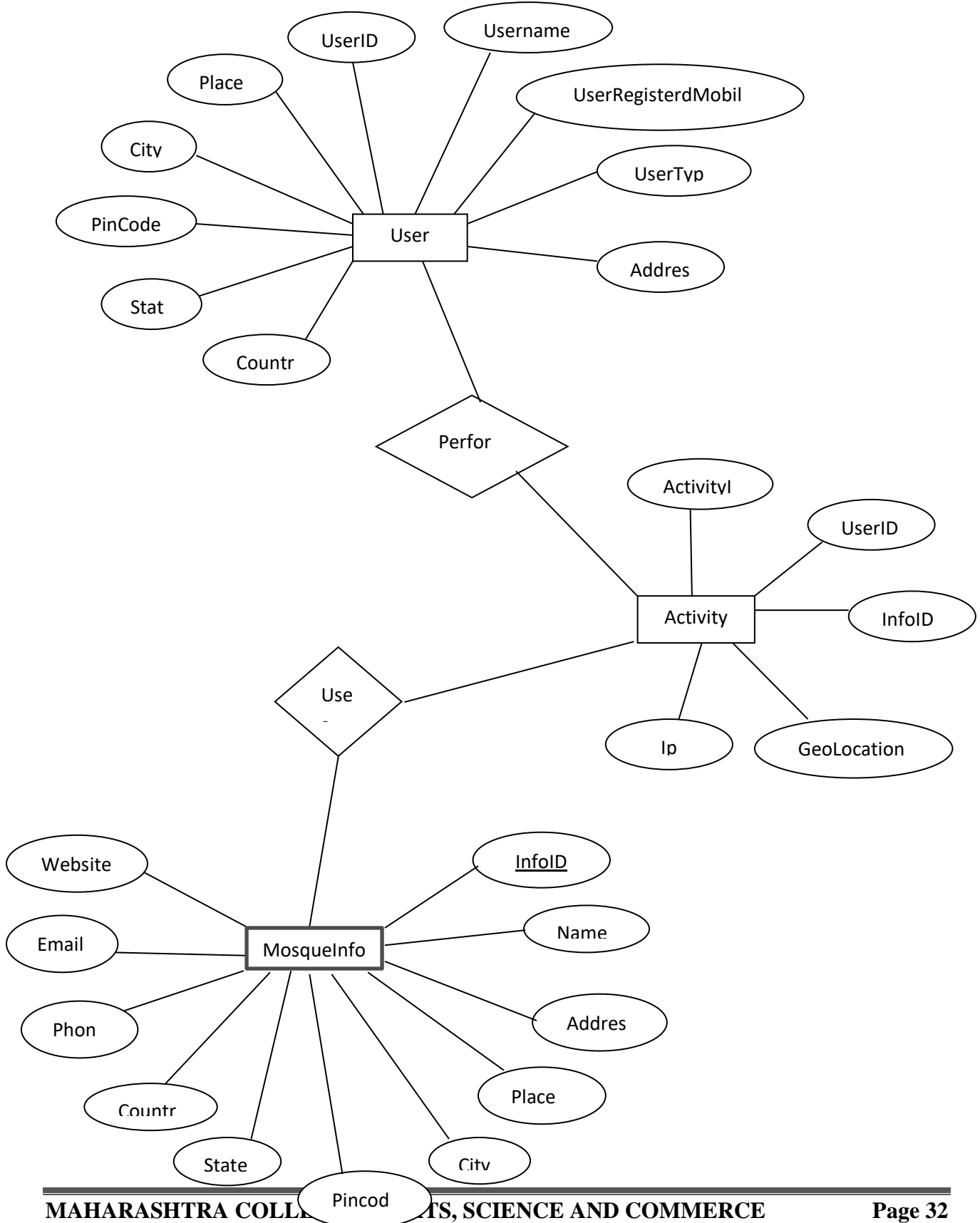
Component Name	Description	Symbol
Entity	A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.	
Relationship	How entities act upon each other or are associated with each other. Think of relationships as verbs. For example, the named student might register for a course. The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way. Relationships are typically shown as diamonds or labels directly on the connecting lines.	
Attribute	A property or characteristic of an entity. Often shown as an oval or circle.	

Table 3.2 :Component Of ER Diagram

## MIRINO (Android Application)

Fig 1. Entity Relationship Diagram





### 3.3 Use Case Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

#### The components of an Use Case diagram

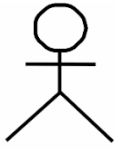


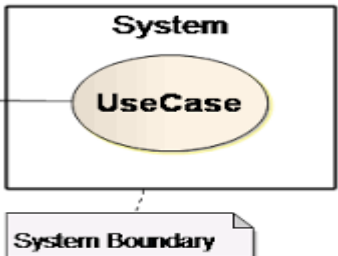
Component Name	Description	Symbol
Actor	The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.	 Actor
Use cases	Horizontally shaped ovals that represent the different uses that a user might have.	 Use Case
Associations	A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.	 Association
System Boundary	A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.	 System UseCase System Boundary

Table 3.3 :Components Of Use Case Diagram

## MIRINO (Android Application)

Use Case Diagram:-

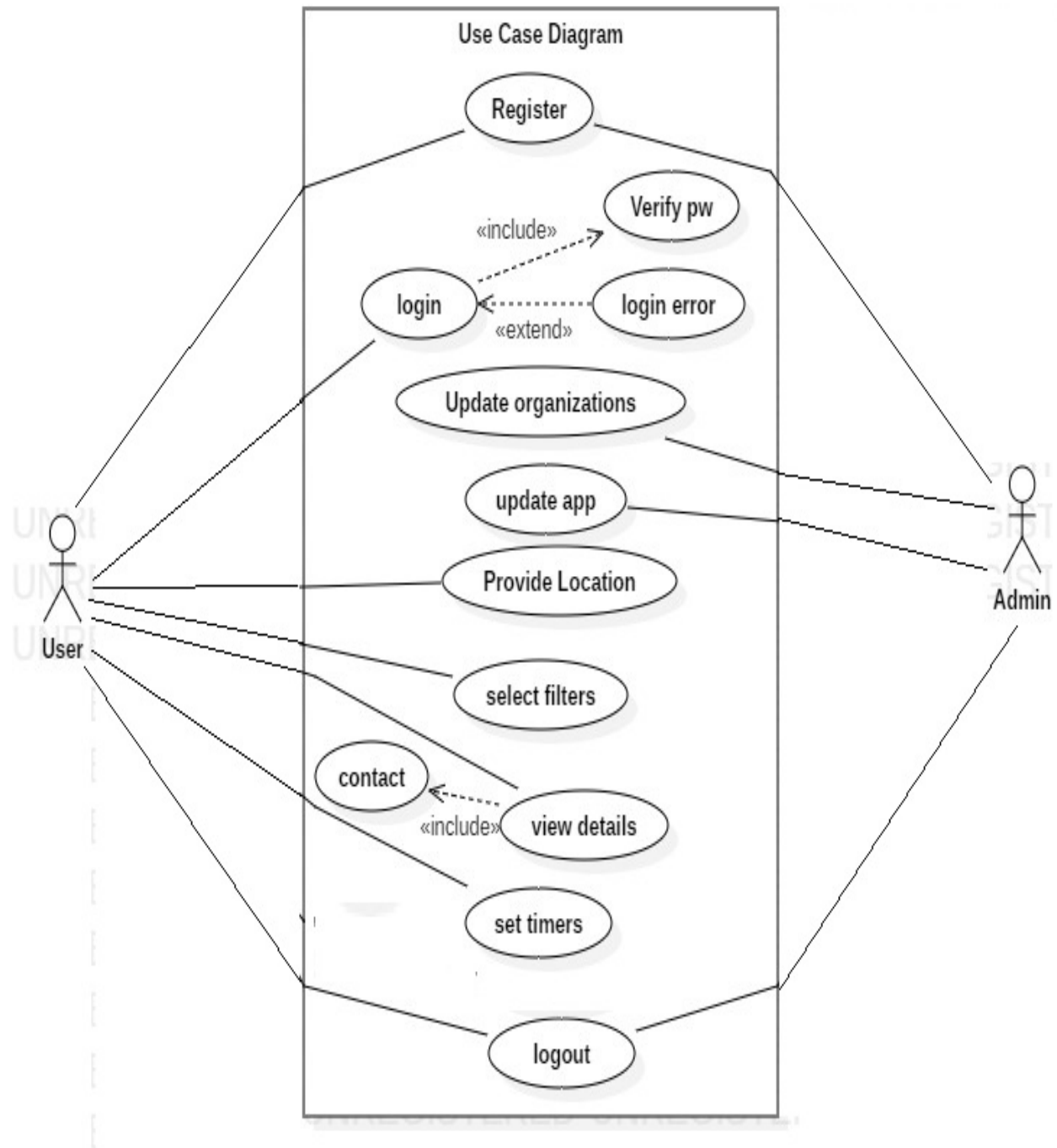


Fig 2. Use Case Diagram

### 3.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence diagrams can be useful references for businesses and other organizations.

#### The components of an Sequence diagram




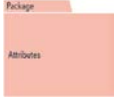

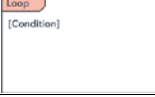

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.
	Package	Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.
	Lifeline	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.
	Option loop	Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.
	Alternative	Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

Table 3.4 :Components Of Sequence Diagram

## MIRINO (Android Application)

Sequence Diagram For User:-

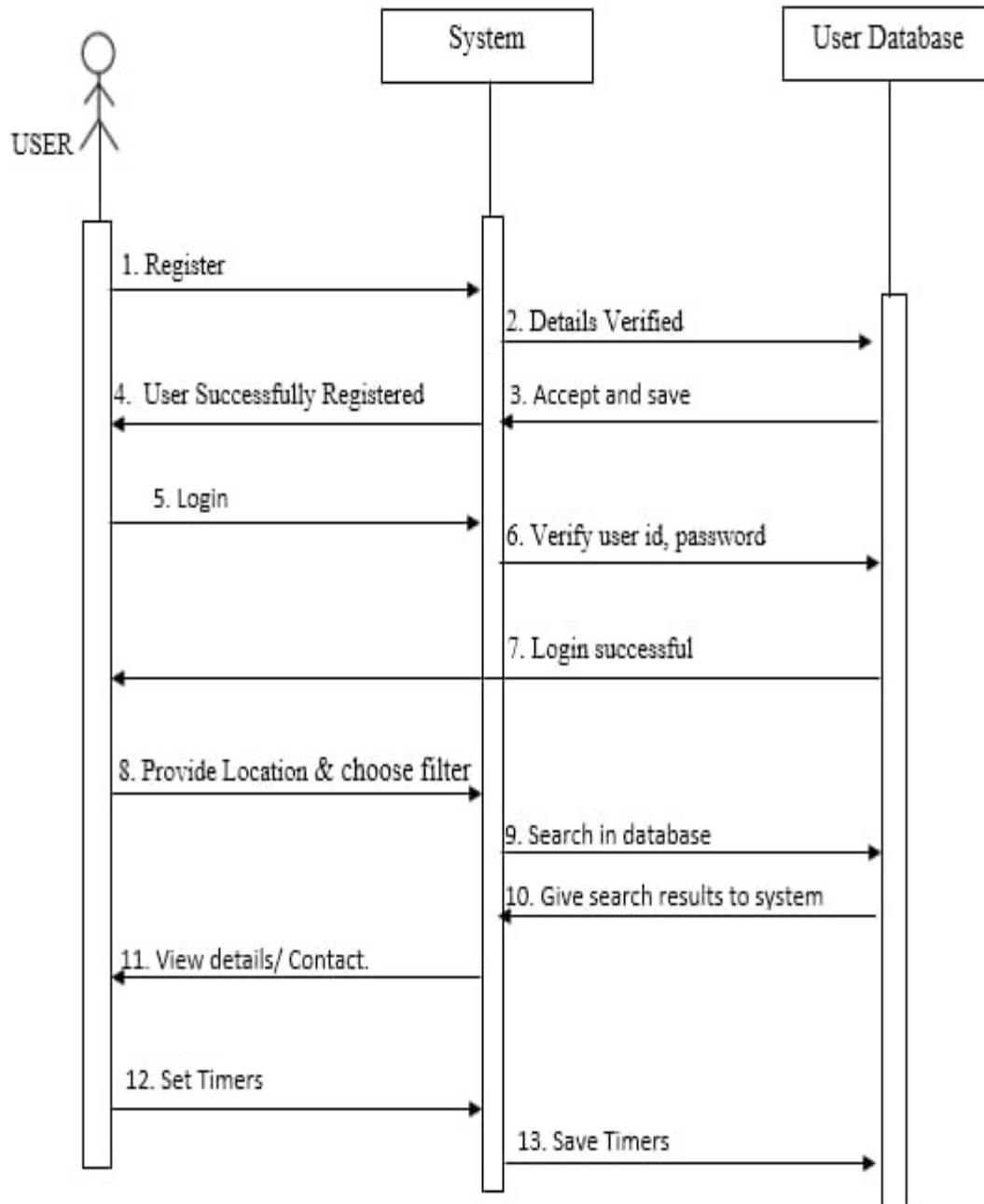


Fig 3.1. Sequence(user) Diagram

## MIRINO (Android Application)

Sequence Diagram For Admin-

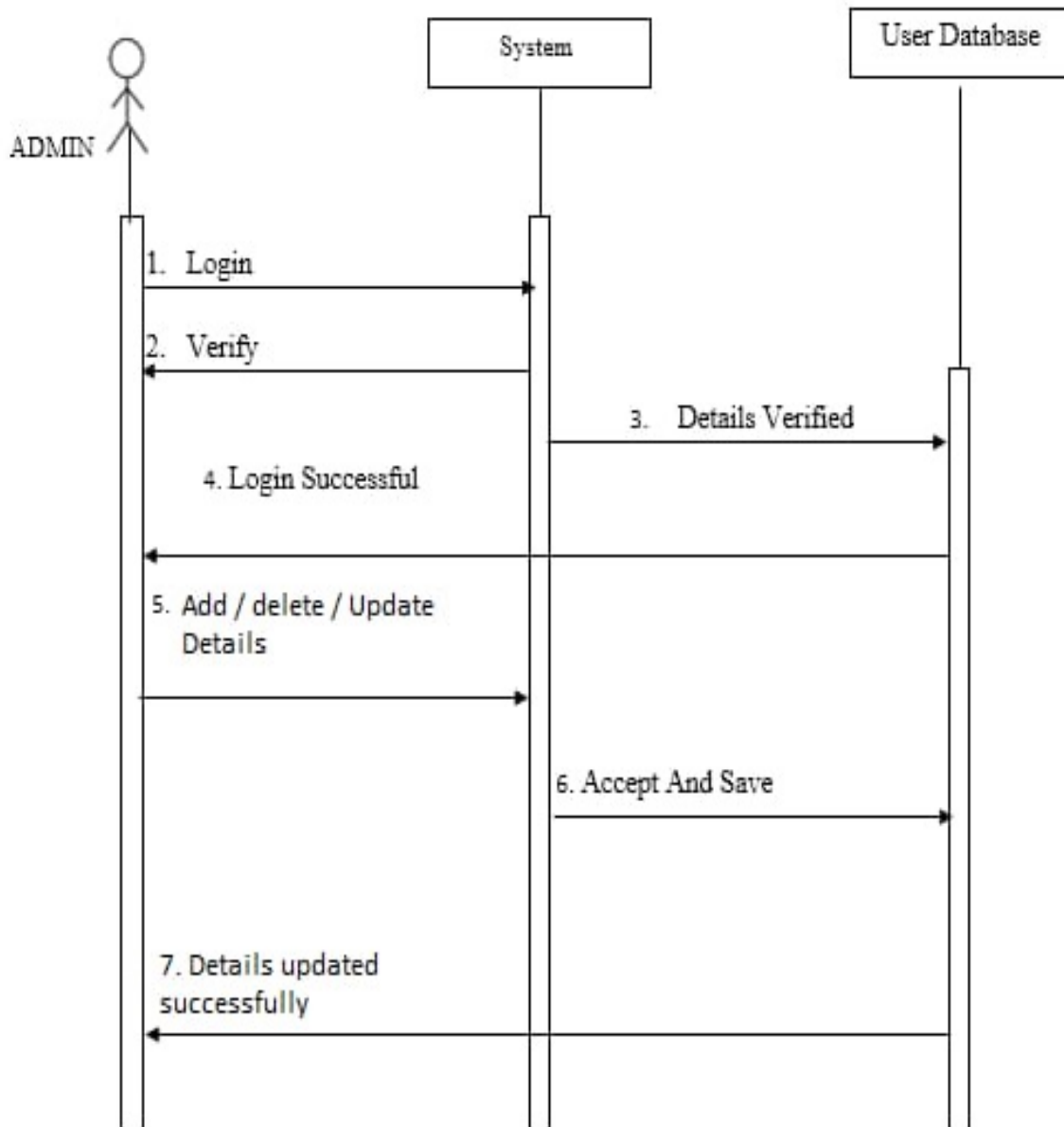


Fig 3.2. Sequence (Admin) Diagram

### 3.5 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

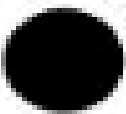

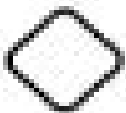
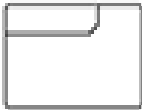
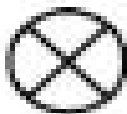

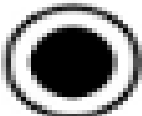
Symbol	Name	Description
	Start symbol	Represents the beginning of a process or workflow in an activity diagram. It can be used by itself or with a note symbol that explains the starting point.
	Activity symbol	Indicates the activities that make up a modeled process. These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram.
	Decision symbol	Represents a decision and always has at least two paths branching out with condition text to allow users to view options. This symbol represents the branching or merging of various flows with the symbol acting as a frame or container.
	Option loop symbol	Allows the creator to model a repetitive sequence within the option loop symbol.
	Flow final symbol	Represents the end of a specific process flow. This symbol shouldn't represent the end of all flows in an activity; in that instance, you would use the end symbol. The flow final symbol should be placed at the end of a process in a single activity flow.
	Condition text	Placed next to a decision marker to let you know under what condition an activity flow should split off in that direction.
	End symbol	Marks the end state of an activity and represents the completion of all flows of a process.

Table 3.5 :Components Of Activity Diagram

## MIRINO (Android Application)

### Login Activity.

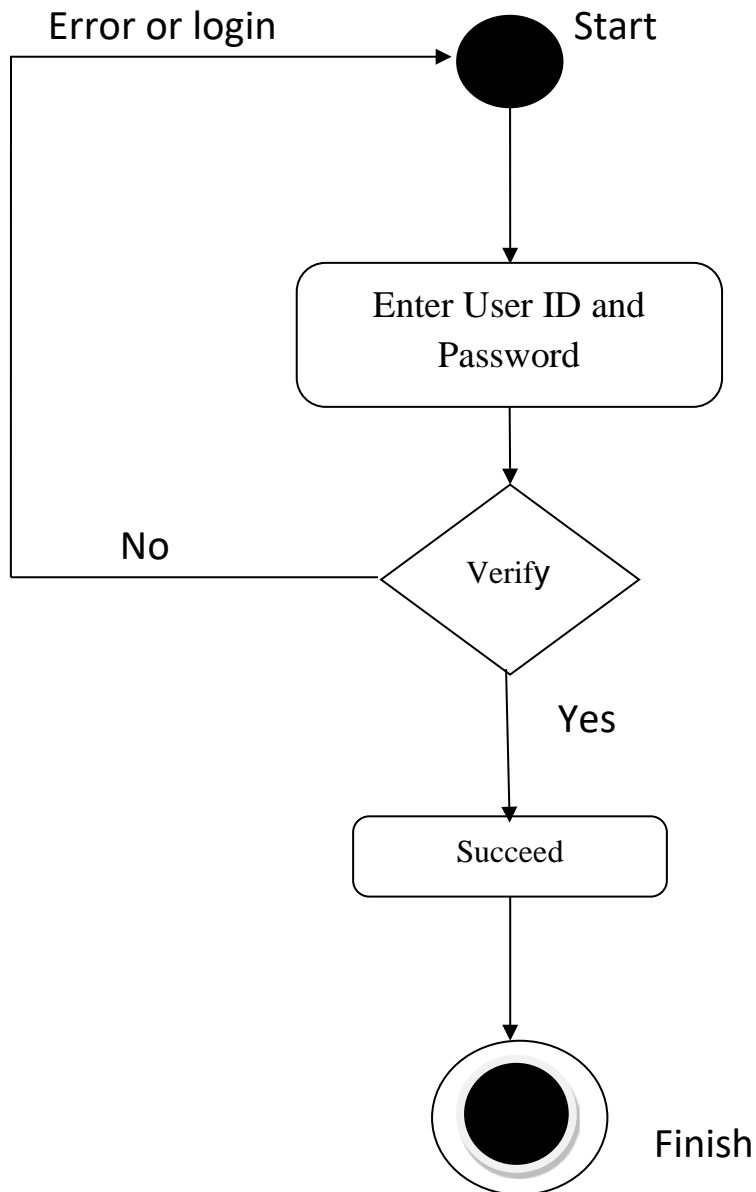


Fig 4.1. LogIn Activity Diagram

## MIRINO (Android Application)

### Registration activity

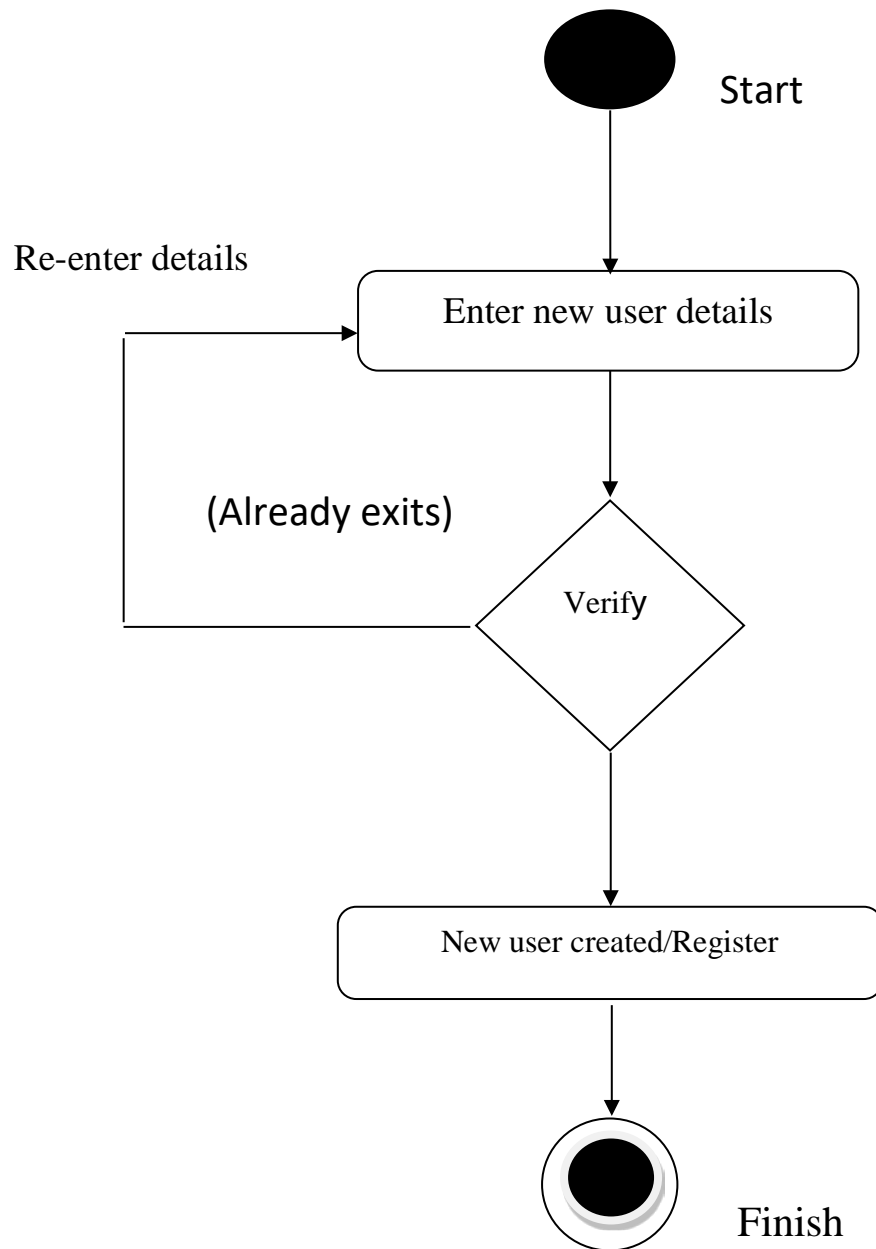


Fig 4.2. Registration Activity Diagram



**View Mosque Activity.**

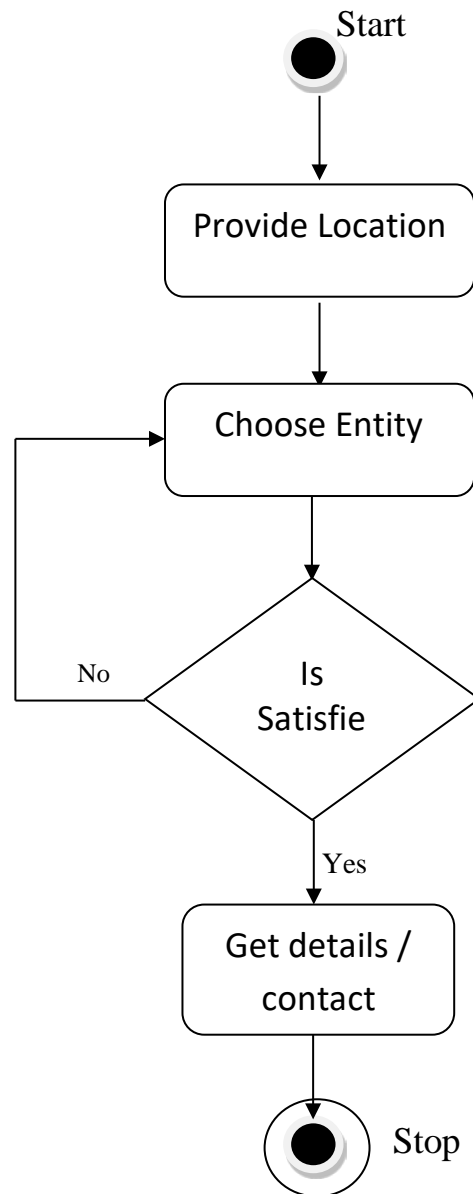


Fig 4.3. Search Mosque Activity Diagram

## MIRINO (Android Application)

### Admin Activity

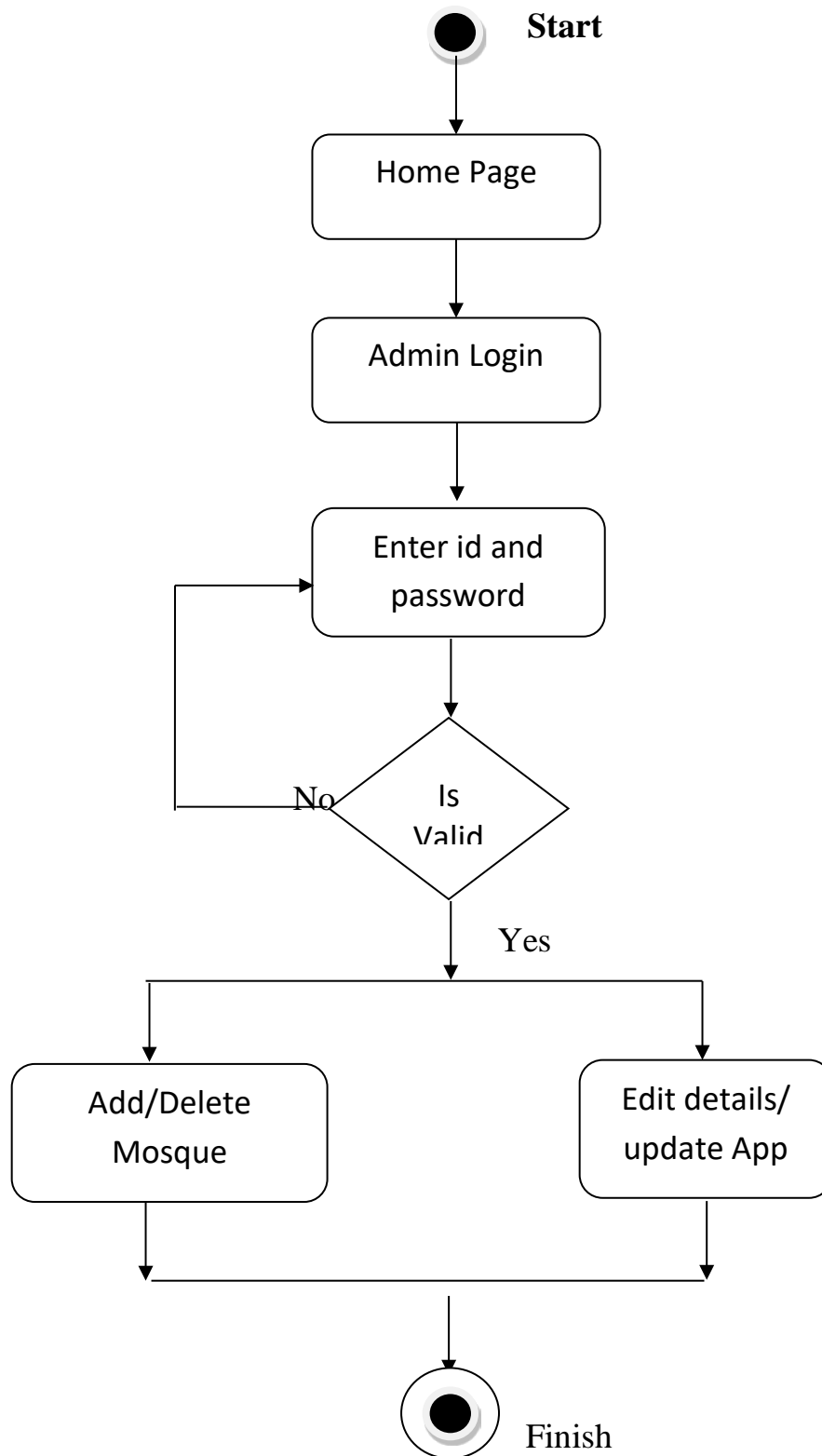


Fig 4.5. Admin Activity Diagram

### 3.6 Deployment Diagram

In the context of the Unified Modeling Language (UML), a deployment diagram falls under the structural diagramming family because it describes an aspect of the system itself. In this case, the deployment diagram describes the physical deployment of information generated by the software program on hardware components. The information that the software generates is called an artifact. This shouldn't be confused with the use of the term in other modeling approaches like BPMN.

Deployment diagrams are made up of several UML shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

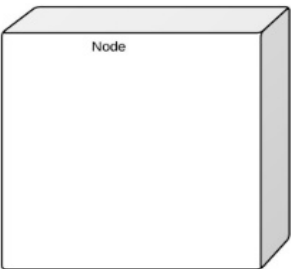


Symbol	Name	Description
	Nodes	There are two types of nodes in a deployment diagram: device nodes and execution environment nodes. Device nodes are computing resources with processing capabilities and the ability to execute programs. Some examples of device nodes include PCs, laptops, and mobile phones. An execution environment node, or EEN, is any computer system that resides within a device node. It could be an operating system, a JVM, or another servlet container.
	Database	Databases represent any data stored by the deployed system. In some instances, you'll see a database represented as just another node, but sometimes you will see this shape as a database.
	package	A file-shaped box that groups together all the device nodes to encapsulate the entire deployment.

Table 3.6 :Components Of Deployment Diagram

## MIRINO (Android Application)

### Deployment Diagram

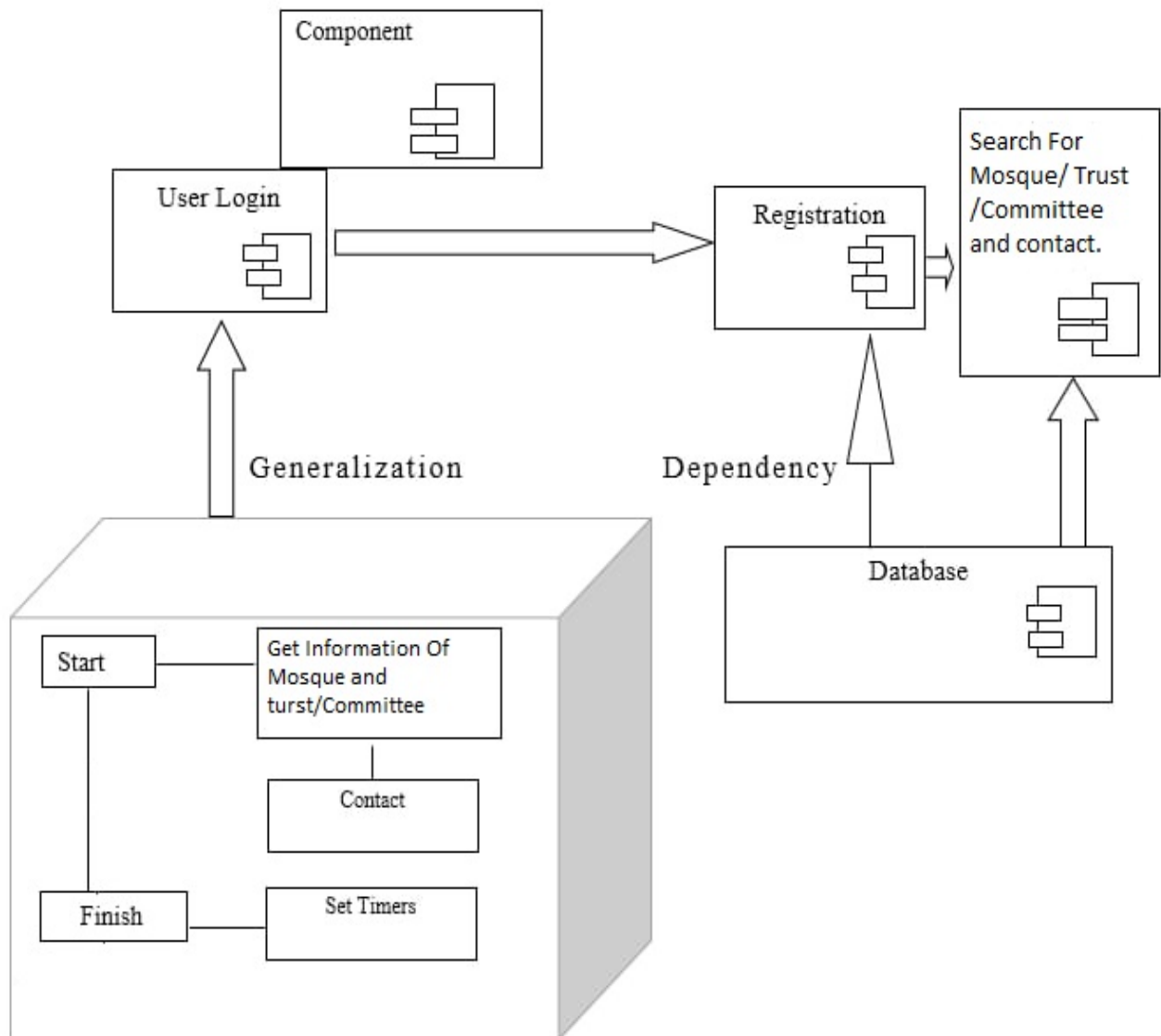


Fig 5. Deployment Diagram

### 3.7 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

#### Components Of Class Diagram.

The standard class diagram is composed of three sections:

- **Upper section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

#### Member access modifiers

All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:

- Public (+)
- Private (-)
- Protected (#)
- Package (~)
- Derived (/)
- Static (underlined)

## MIRINO (Android Application)

### Member scopes:

There are two scopes for members: classifiers and instances.

Classifiers are static members while instances are the specific instances of the class. If you are familiar with basic OO theory, this isn't anything groundbreaking.

### Additional class diagram components

Depending on the context, classes in a class diagram can represent the main objects, interactions in the application, or classes to be programmed. To answer the question "What is a class diagram in UML?" you should first understand its basic makeup.

- **Classes:** A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations. When you draw a class in a class diagram, you're only required to fill out the top row—the others are optional if you'd like to provide more detail.
  - **Name:** The first row in a class shape.
  - **Attributes:** The second row in a class shape. Each attribute of the class is displayed on a separate line.
  - **Methods:** The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.
- **Signals:** Symbols that represent one-way, asynchronous communications between active objects.
- **Data types:** Classifiers that define data values. Data types can model both primitive types and enumerations.
- **Packages:** Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
- **Interfaces:** A collection of operation signatures and/or attribute definitions that define a cohesive set of behaviors. Interfaces are similar to classes, except that a class can have an instance of its type, and an interface must have at least one class to implement it.
- **Enumerations:** Representations of user-defined data types. An enumeration includes groups of identifiers that represent values of the enumeration.
- **Objects:** Instances of a class or classes. Objects can be added to a class diagram to represent either concrete or prototypical instances.

## MIRINO (Android Application)

### Class Diagram

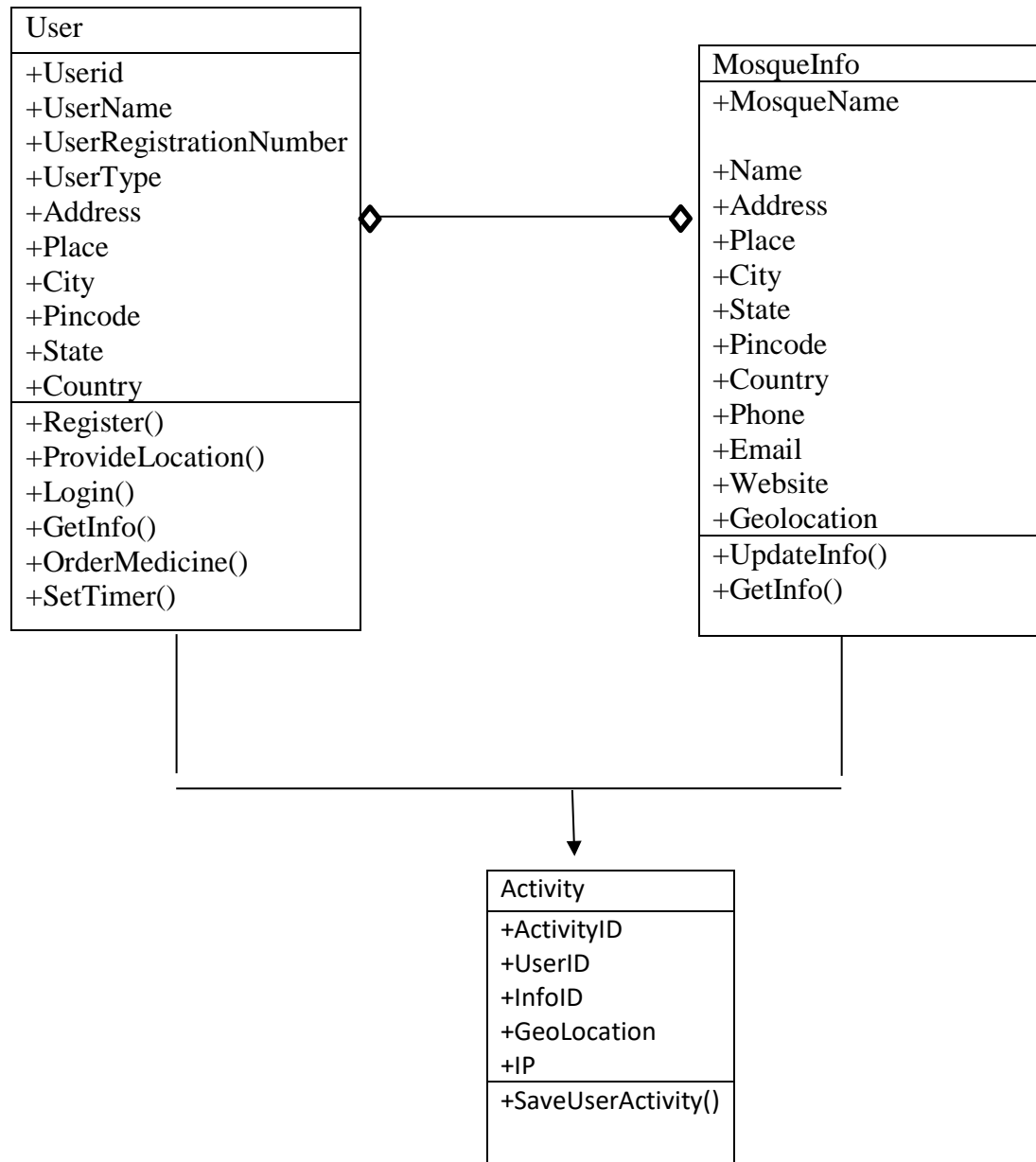


Fig 6. Class Diagram

### 3.8 Data Flow/DFD Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.


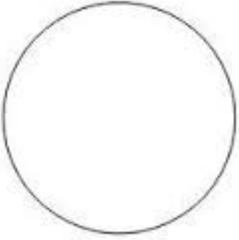


Symbol	Name	Description
	External entity	An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
	Process	Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
	Data Store	files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
	Data Flow	the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

Table 3.8 :Components Of Data Flow Diagram



## MIRINO (Android Application)

### Data Flow Diagram

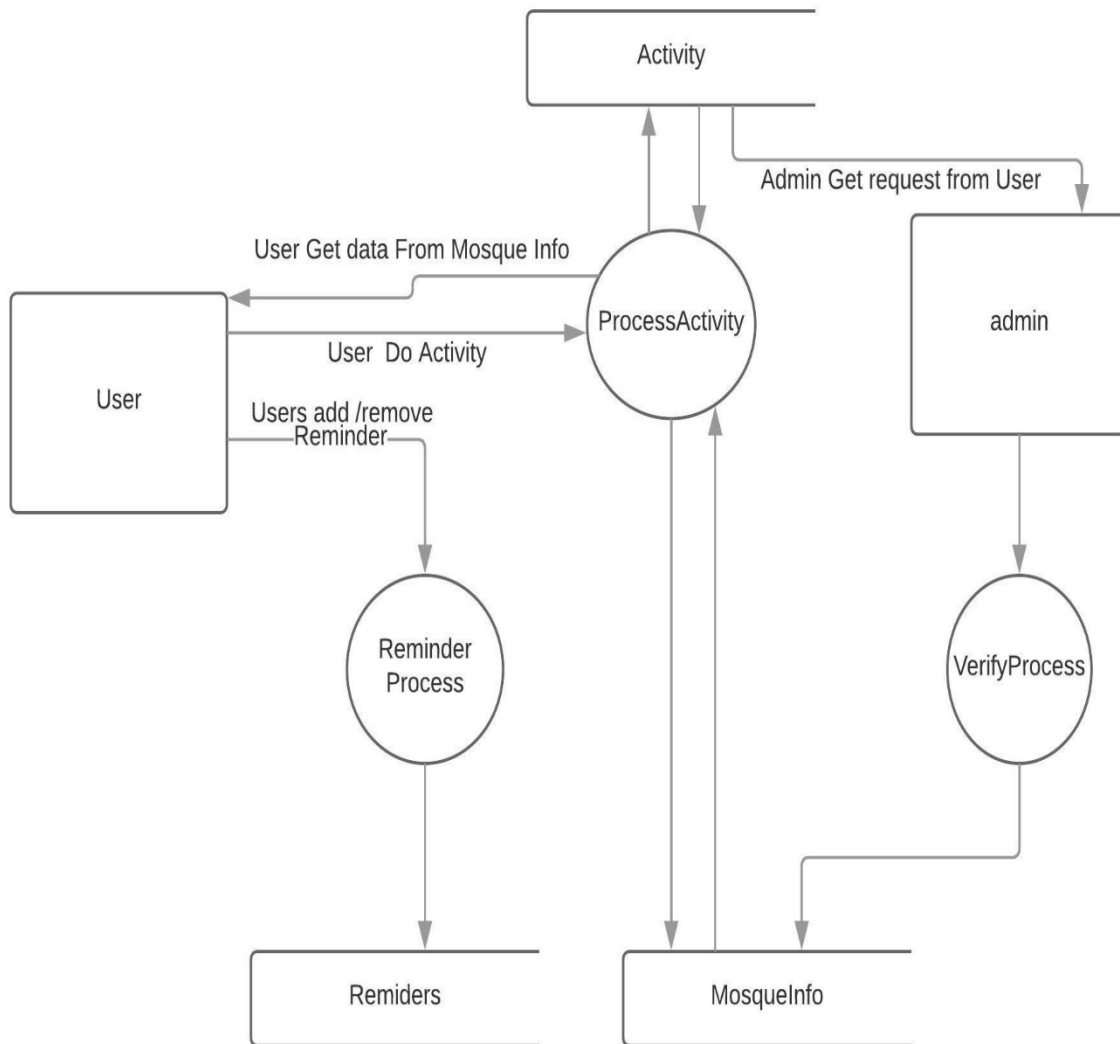


Fig 7. Data Flow Diagram

**3.9 Event Table**

<b>Events</b>	<b>Trigger</b>	<b>Source</b>	<b>Activity</b>	<b>Response</b>	<b>Destination</b>
User Login	Ask For Form	User	User Record	User Logged In	User
User registration	Ask For Form	User	Create New Record	User registered	User
Location	Allow Location Access	User	Provide Location For System	Location Received	User
Filter	Choose Filters	User	Selection Of Filter	Output Provide as per selection	User
View	View Details	User	View Details Of Selected Mosque	Details Shown	User
Timer	Update Timer	User	Create Or Edit Timer	Timer Update	User
Admin Login	Ask For Form	Admin	Admin Record	Admin Logged In	Admin
Admin Registration	Ask For Form	Admin	Create New Admin Record	Admin Registered	Admin
Update	Update Mosque	Admin	Add / Delete / Update	Updated	Admin

Table 3. :Event Table

# **IMPLEMENTATION AND TESTING**

## **4. IMPLEMENTATION AND TESTING**

### **Implementation:**

Implementation is the stage of the project where the theoretical design is turned in to a working system. The implementation state is a system project in its own right.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, training of staff in the changeover procedure and evaluation of change over methods.

Once the planning has been completed, the major efforts are to ensure that the program in the system is working properly.

At the same time concentrate on training user staff. When the staff has been trained a full system can carry out.

The various activities involved while implementing a project :-

- End user education and training
- Training on application software
- System testing
- Parallel run and change over to new system
- Post implementation review

### **Testing:**

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error.

A good test is one that has a probability of finding an yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is vital in the success of the system.

In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specification stating what the program should do and how it should perform under various conditions.

System testing does not test the software as a whole, but rather the integration of each module in the system. The primary concern is the compatibility of individual module. One has to find areas where modules have been design with different specifications of data lengths, type and data element name.

### 4.1 Code

#### Java the Programming Language

Programming languages, like regular languages, are different ways to communicate to a computer how you want it to act. Programming languages allow us to instruct a computer step-by-step how to manipulate data, collect input from users, and display things on a screen, among other things.

Way down on a microscopic level, the processor of a computer sends electrical signals back and forth that control how it operates. High level programming languages like Java mean that we can write these instructions in an abstract manner using words and symbols, and the computer will take care of translating these instructions that we can understand all the way down to electrical impulses that the processor can understand.

Not to get ahead of ourselves, but Java is a statically-typed, object-oriented language. Let's break this down:

- “Statically-typed” – Programming at its core is really about working with data. Pieces of data are stored as variables, which are basically containers that hold data. Statically-typed languages like Java require us to declare what type of data each variable (or container) will hold. So for example, if a variable is supposed to hold a number, we need to say so, and we won't be allowed to put something else like a letter in it. Statically-typed also means that all the variables will be checked before the program even runs, and we'll be presented with an error if we forget to declare a type of data or declare the wrong one.
- “Object-oriented” – An object-oriented language is one that is built around the concept of objects. In the physical world, take a look around the room and think of each thing as an object. For example, on my desk right now I have a mug. As an object, it's name is “mug” and it has properties about it like its color and how much liquid it will hold. Object-oriented languages allow us to define objects like mugs and access their properties in our code. We can also send messages to objects, so for my mug I might want to know “Is it empty?” We can then create and manipulate all sorts of objects to do different things in our app. For example, we can use the Camera object to take a photo. The Camera object represents the physical camera on an Android phone, but in a way that we can interact with in code.

## MIRINO (Android Application)

Extensible Markup Language (XML):

In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification<sup>[2]</sup> and several other related specifications—all of them free open standards—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services. Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. Here `<application>...</application>` tags enclosed the components related to the application. Attribute `android:icon` will point to the application icon available under `res/drawable-hdpi`. The application uses the image named `ic_launcher.png` located in the drawable folders

The `<activity>` tag is used to specify an activity and `android:name` attribute specifies the fully qualified class name of the Activity subclass and the `android:label` attributes specifies a string to use as the label for the activity. You can specify multiple activities using `<activity>` tags.

The **action** for the intent filter is named `android.intent.action.MAIN` to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named `android.intent.category.LAUNCHER` to indicate that the application can be launched from the device's launcher icon.

The `@string` refers to the `strings.xml` file explained below. Hence, `@string/app_name` refers to the `app_name` string defined in the `strings.xml` file, which is "HelloWorld". Similar way, other strings get populated in the application.

### **4.2 Testing Approach**

#### **White Box Testing**

WHITE BOX TESTING (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

Definition by ISTQB

- white-box testing: Testing based on an analysis of the internal structure of the component or system.
- white-box test design technique: Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.

Example:

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

Levels Applicable To

## MIRINO (Android Application)

White Box Testing method is applicable to the following levels of software testing:

- Unit Testing: For testing paths within a unit.
- Integration Testing: For testing paths between units.
- System Testing: For testing paths between subsystems.

However, it is mainly applied to Unit Testing.

➤ Advantages:

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

➤ Disadvantages:

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available.



### 4.2.1. Unit Testing

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

➤ Definition by ISTQB

- unit testing: See component testing.
- component testing: The testing of individual software components.

➤ Unit Testing Method

It is performed by using the White Box Testing method.

➤ When is it performed?

Unit Testing is the first level of software testing and is performed prior to Integration Testing.

➤ Who performs it?

is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.

## MIRINO (Android Application)

### Unit Testing Benefits:

- Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.
- Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.
- Development is faster. How? If you do not have unit testing in place, you write your code and perform that fuzzy ‘developer test’ (You set some breakpoints, fire up the GUI, provide a few inputs that hopefully hit your code and hope that you are all set.) But, if you have unit testing in place, you write the test, write the code and run the test. Writing tests takes time but the time is compensated by the less amount of time it takes to run the tests; You need not fire up the GUI and provide all those inputs. And, of course, unit tests are more reliable than ‘developer tests’. Development is faster in the long run too. How? The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.
- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the software is live.
- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.
- Codes are more reliable. Why? I think there is no need to explain this to a sane person.

### 4.2.2. Integration Testing.

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

➤ Definition by ISTQB

- integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.
- component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components.
- system integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet)

➤ When is it performed?

System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing.

➤ Who performs it?

Normally, independent Testers perform System Testing

### **4.3 Test Cases , Test Data & Test Result**

Test cases should be designed and written by someone who understands the function or technology being tested. A test case should include the following information –

1. Purpose of the test
2. Software requirements and Hardware requirements (if any)
3. Specific setup or configuration requirements
4. Description on how to perform the test(s)
5. Expected results or success criteria for the test

Designing test cases can be time consuming in a testing schedule, but they are worth giving time because they can really avoid unnecessary retesting or debugging or at least lower it.

Organizations can take the test cases approach in their own context and according to their own perspectives. Some follow a general step way approach while others may opt for a more detailed and complex approach.

It is very important for you to decide between the two extremes and judge on what would work the best for you.

Designing proper test cases is very vital for your software testing plans as a lot of bugs, ambiguities, inconsistencies and slip ups can be recovered in time as also it helps in saving your time on continuous debugging and re-testing test cases.

## MIRINO (Android Application)

### Test case 1

<b>Component Name :</b>	Login Module
<b>Test Condition :</b>	To check working of Login Module.
<b>Procedure :</b>	Enter a Registered User name And correct password in the required field and click login.
<b>Expected output:</b>	it should validate the users entered credentials with the database then it should log in otherwise return on login layout.

### Test case 2

<b>Component Name :</b>	Registration Module
<b>Test Condition :</b>	To check working of Registration Module.
<b>Procedure :</b>	Enter a valid detail's in the required fields and click register.
<b>Expected output:</b>	it should validate the users entered credentials with the with set pattern(i.e. email must consist of .@ and password must be 8 digit long) and If all the pattern matches with the details then user should get registered.

### Test case 3

<b>Component Name :</b>	View Module
<b>Test Condition :</b>	To check working of View Module.
<b>Procedure :</b>	Enter the details to search.
<b>Expected output:</b>	Get the details using like query from the database.

### Test case 4

<b>Component Name :</b>	Reminder Module
<b>Test Condition :</b>	To check working of Reminder Module.
<b>Procedure :</b>	Enter/view the previously stored reminders and change or add according to end user requirements.
<b>Expected output:</b>	If user set new reminder it should be settled and if it adds new reminder then new reminder should be created.

## **MIRINO (Android Application)**

### **Test case 5**

**Component Name :** Request Module

**Test Condition :** To check working of Request Module

**Procedure :** User can request admin to add the new mosque or delete the listed mosque .

**Expected output:** after doing request it should be display in the admin module.

### **Test case 6**

**Component Name :** Admin add/delete Module

**Test Condition :** To check working of Admin Mosque Add And Delete Module.

**Procedure :** If Admin Got Request From user It Should Displayed in the list and admin will verify it then it has to added in the final database .

**Expected output:** After verified by the admin changes should happen in the Database

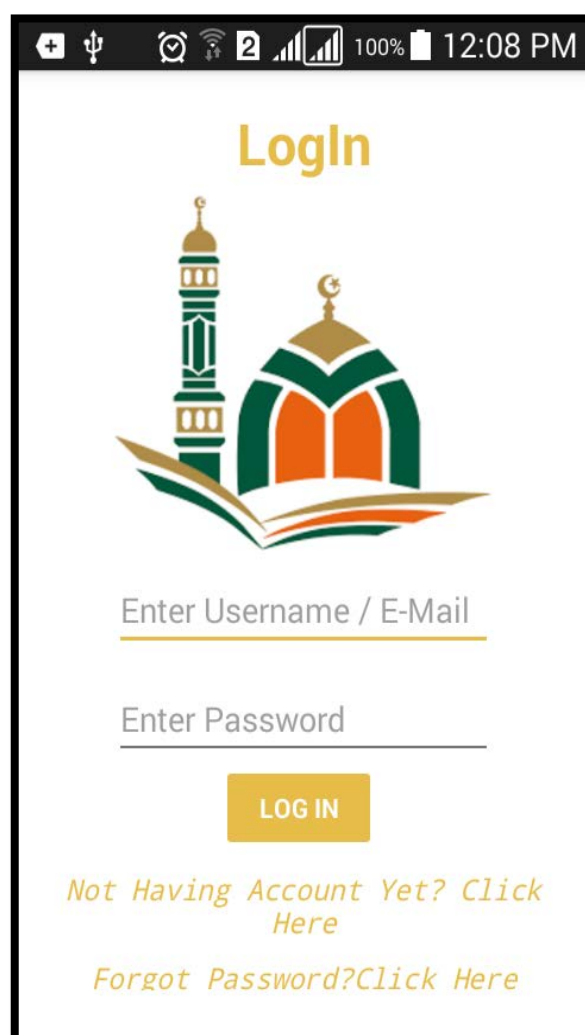
# **RESULTS AND DISCUSSIONS**

## 5.0 Results and Discussions

### 5.0.1 Results (Output Screens) :



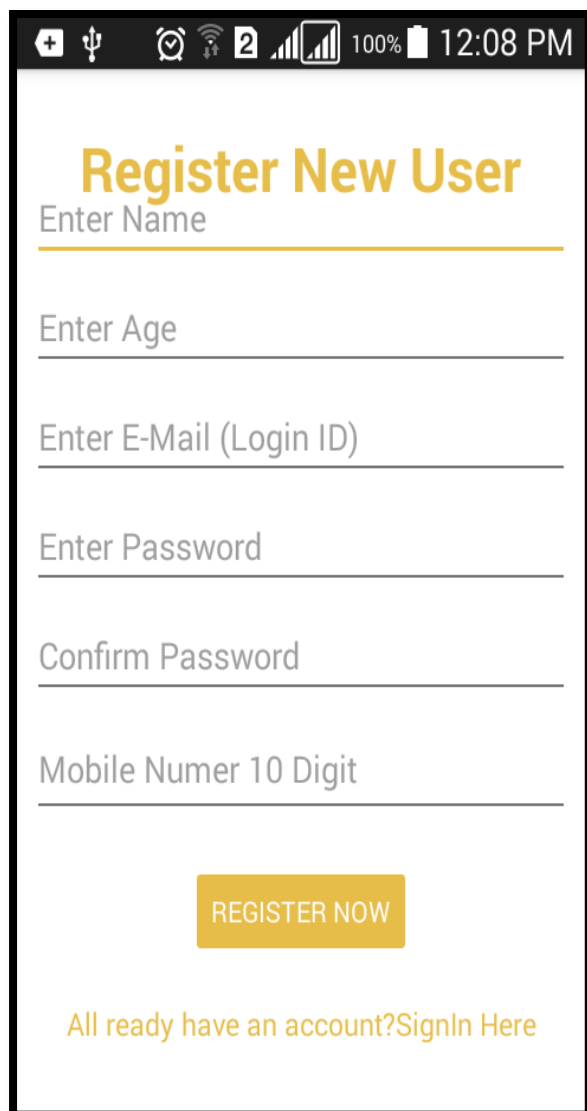
(Splash Layout)



(Login Layout)



## MIRINO (Android Application)



Register New User

Enter Name

Enter Age

Enter E-Mail (Login ID)

Enter Password

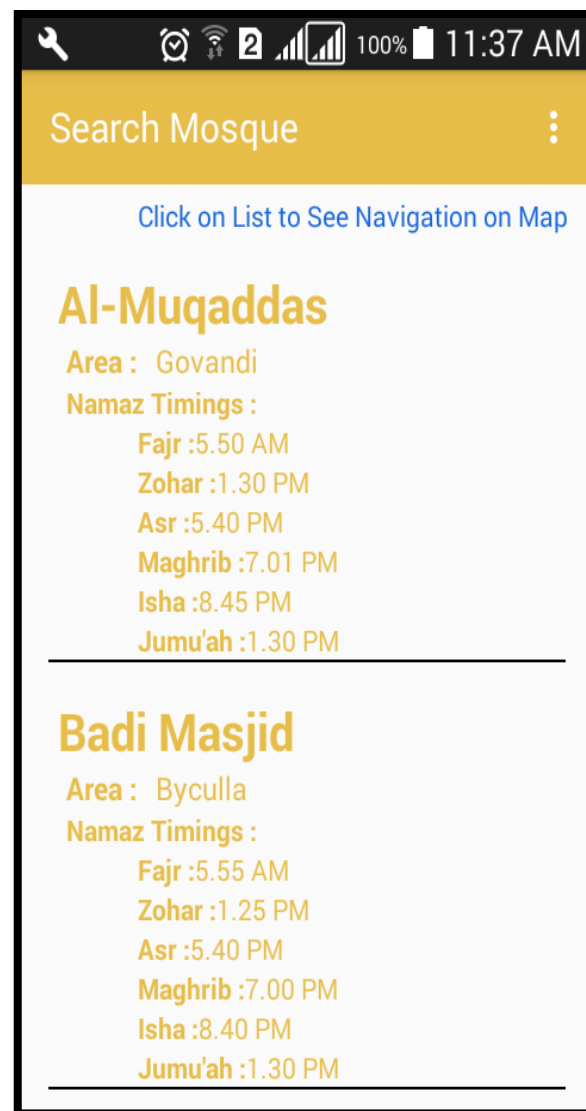
Confirm Password

Mobile Numer 10 Digit

REGISTER NOW

All ready have an account?SignIn Here

( Register Layout )



Search Mosque

Click on List to See Navigation on Map

**Al-Muqaddas**

Area : Govandi

Namaz Timings :

Fajr :5.50 AM

Zohar :1.30 PM

Asr :5.40 PM

Maghrib :7.01 PM

Isha :8.45 PM

Jumu'ah :1.30 PM

**Badi Masjid**

Area : Byculla

Namaz Timings :

Fajr :5.55 AM

Zohar :1.25 PM

Asr :5.40 PM

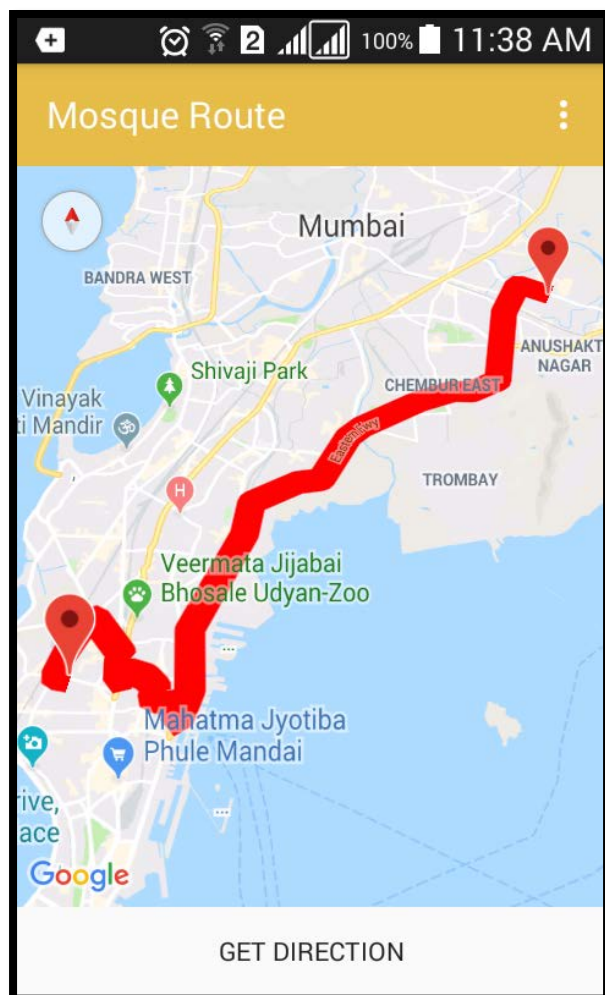
Maghrib :7.00 PM

Isha :8.40 PM

Jumu'ah :1.30 PM

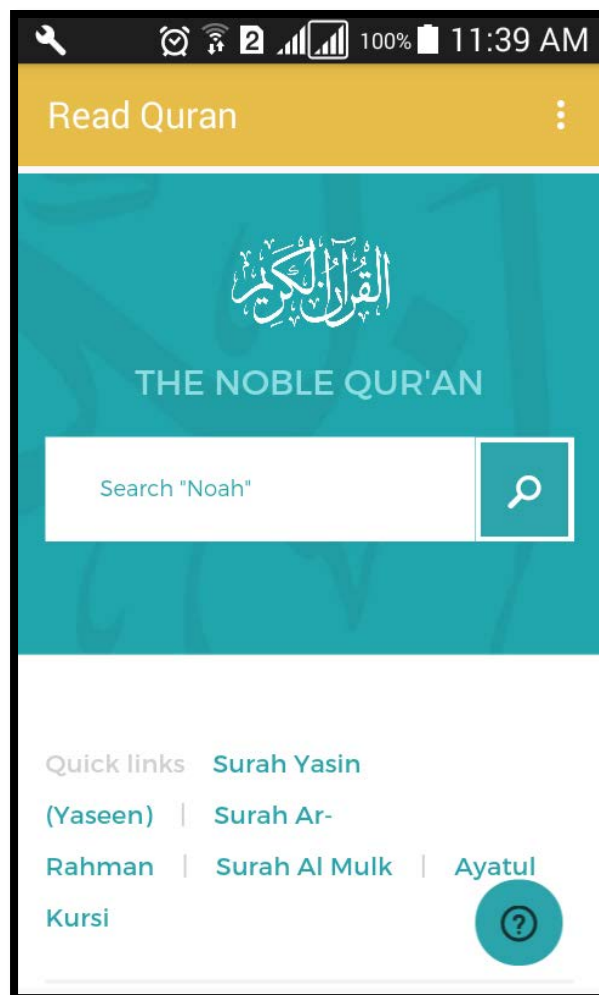
(Search Mosque Layout)

## MIRINO (Android Application)



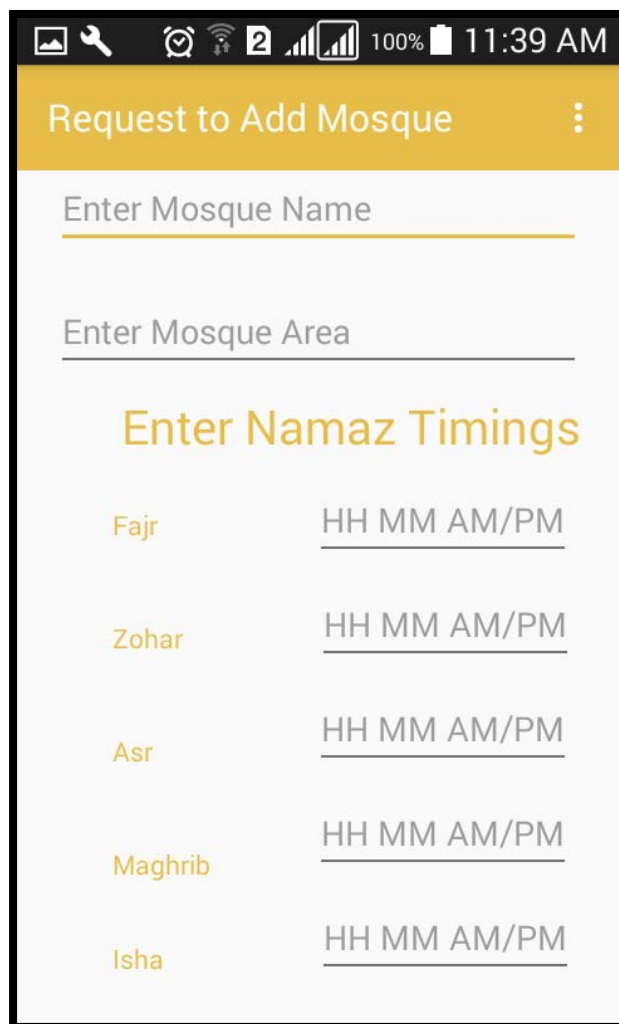
(Mosque Route Layout)

On Click of search mosque



(QURAN webview)

## MIRINO (Android Application)



The screenshot shows the 'Request to Add Mosque' form in the MIRINO app. The form has a yellow header bar with the title 'Request to Add Mosque' and a three-dot menu icon. Below the header, there are five input fields: 'Enter Mosque Name', 'Enter Mosque Area', 'Enter Namaz Timings' (in yellow text), and five individual fields for 'Fajr', 'Zohar', 'Asr', 'Maghrib', and 'Isha'. Each of these five fields is followed by a 'HH MM AM/PM' time format placeholder. The status bar at the top shows the time as 11:39 AM, 100% battery, and various connectivity icons.

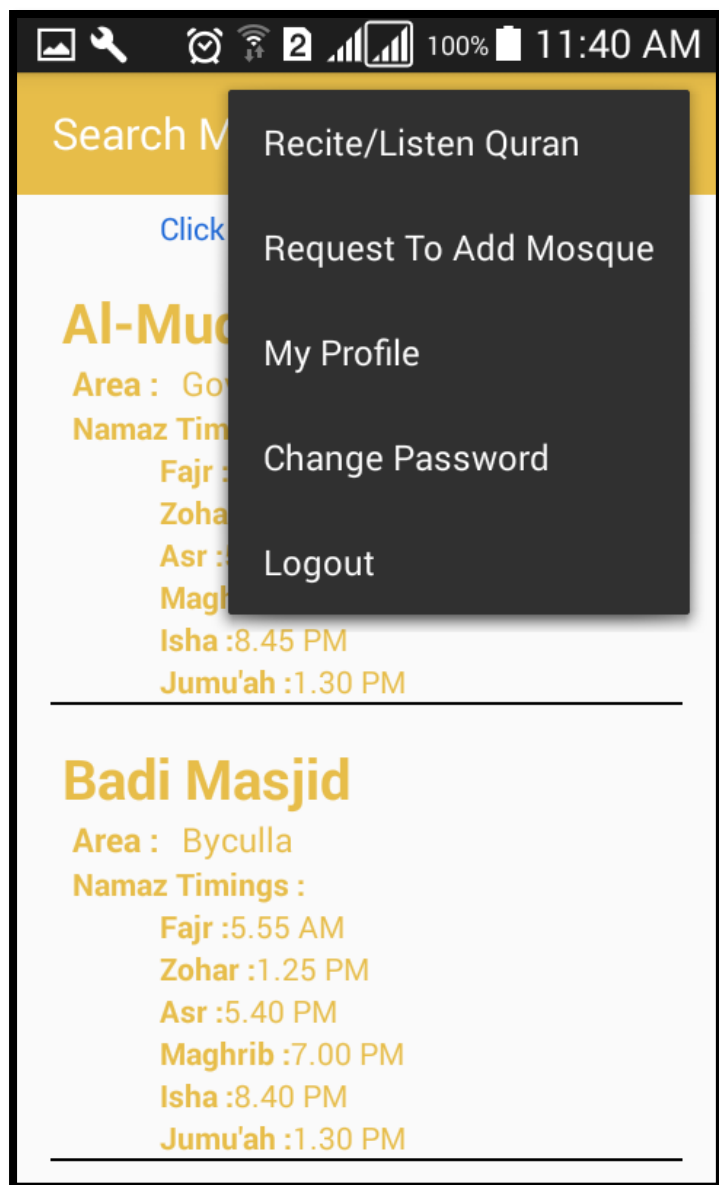
**(Request Add Layout)**



**(Mosque Location Layout)**

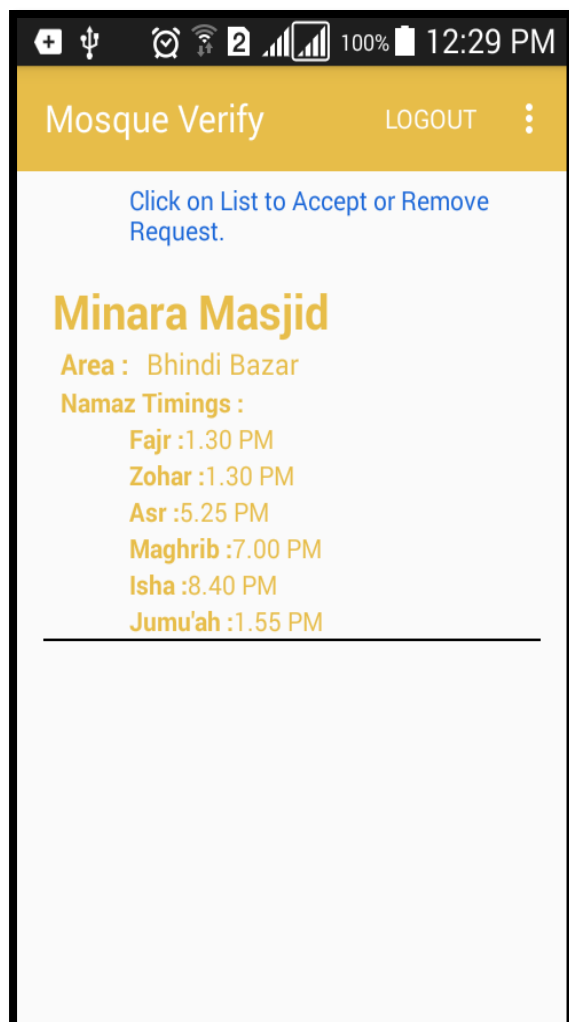
**Choose Lat Long**

## MIRINO (Android Application)

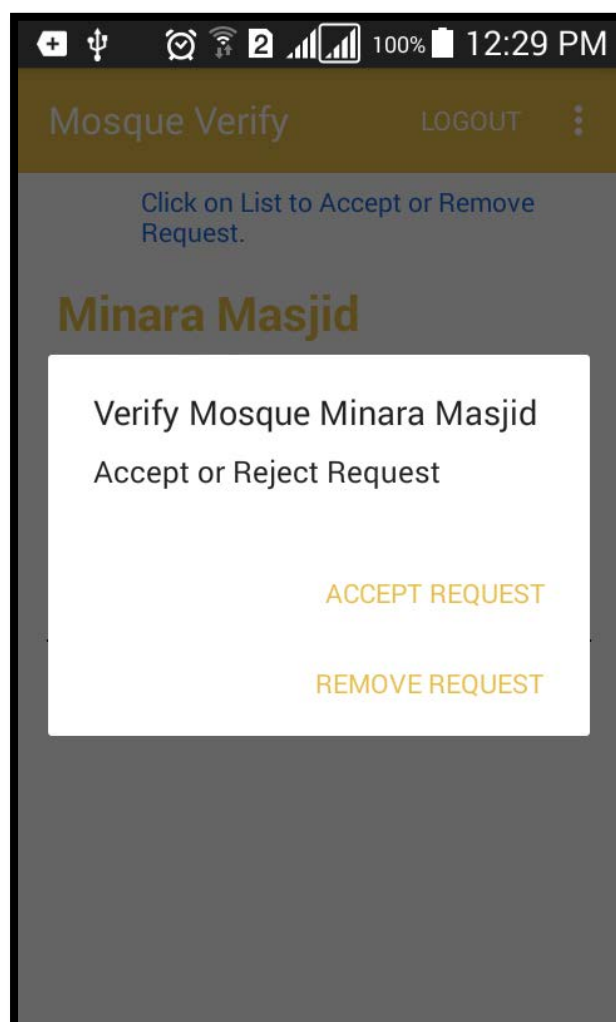


(User Menu Layout)

## MIRINO (Android Application)

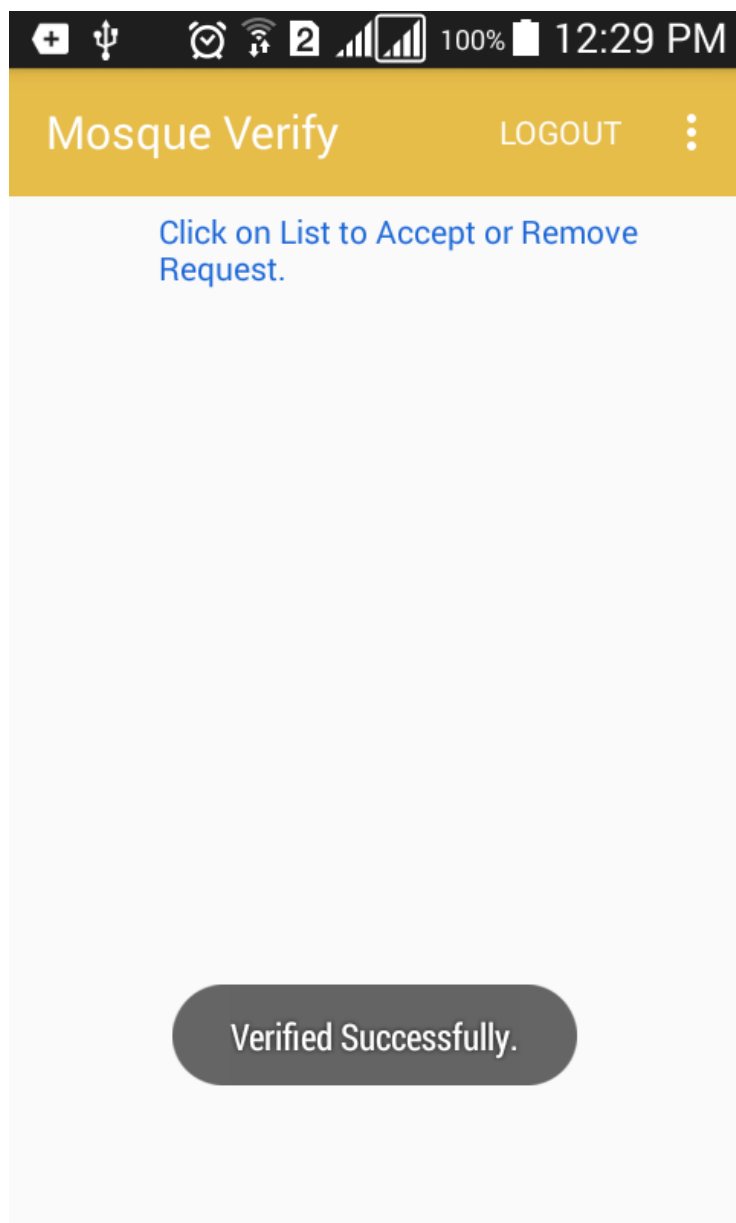


(Admin Activity)



(On Click Accept Reject)

## MIRINO (Android Application)



**(On Successful Clear List )**

## MIRINO (Android Application)

### 5.0.2 Discussion (Coding) :

#### LoginMainActivity

```
package com.larebshaikh.mirino;
import android.annotation.SuppressLint;
import android.content.DialogInterface;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Patterns;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

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 LoginMainActivity extends AppCompatActivity {

    TextView gotoregister, gotoForgotPassword;
    ProgressBar progressBar;
    EditText etUserName, etPassword;
    Button btnLogin;
```

## MIRINO (Android Application)

```
private FirebaseAuth mAuth;
private FirebaseDatabase firebaseDatabase;
private DatabaseReference databaseReference;
private Toolbar mtoolbar;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login_main);

    etPassword = findViewById(R.id.etPasslogin);
    etUserName = findViewById(R.id.etUsername);
    gotoregister=findViewById(R.id.tvGoToSignUp);
    gotoForgotPassword=findViewById(R.id.tvGoToForgotPassword);
    progressBar = findViewById(R.id.progressBar3);
    btnLogin = findViewById(R.id.btnLogin);

    mAuth= FirebaseAuth.getInstance();
    FirebaseUser user = mAuth.getCurrentUser();
    firebaseDatabase = FirebaseDatabase.getInstance();

    if(user != null){
        finish();
        startActivity(new Intent(LoginMainActivity.this, MosqueSearchActivity.class));
    }

    btnLogin.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick (View view){
            if(verifyEmail()){
                validate(etUserName.getText().toString(),etPassword.getText().toString());
            }
        }

    });
```



## MIRINO (Android Application)

```
        gotoregister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(LoginMainActivity.this,RegisterUser.class));
            }
        });

        gotoForgotPassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(LoginMainActivity.this,ForgotPassword2.class));
            }
        });
    }

    private void validate(String userName,String userPassword) {
        progressBar.setVisibility(View.VISIBLE);
        mAuth.signInWithEmailAndPassword(userName,
userPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {

firebaseDatabase.getReference().child("Users").child(mAuth.getCurrentUser().getUid());
                DatabaseReference
databaseReference=firebaseDatabase.getReference().child("Users").child(mAuth.getCurrentUser()
.getUid());
                databaseReference.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                        String Admin = dataSnapshot.child("isAdmin").getValue().toString();
                        if(Admin.equals("True")){
                            progressBar.setVisibility(View.GONE);
                            Toast.makeText(LoginMainActivity.this, "Login Successfull",
Toast.LENGTH_SHORT).show();
                            startActivity(new
Intent(LoginMainActivity.this,MosqueVerifyActivity.class));
                            finish();
                        }
                    }
                });
            }
        });
    }
}
```

## MIRINO (Android Application)

```
        }else if(Admin.equals("False")){
            progressBar.setVisibility(View.GONE);
            Toast.makeText(LoginMainActivity.this, "Login Successfull",
Toast.LENGTH_SHORT).show();
            startActivity(new
Intent(LoginMainActivity.this,MosqueSearchActivity.class));
            finish();
        }else{
            progressBar.setVisibility(View.GONE);
            Toast.makeText(LoginMainActivity.this, "Incorrect UserEmail or Password",
Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    }
    });
}
}
});
}

private Boolean verifyEmail(){

    String email=etUserName.getText().toString().trim();
    String Password=etPassword.getText().toString().trim();

    Boolean result = false;
    if (email.isEmpty()) {
        etUserName.setError(getString(R.string.input_error_email));
        etUserName.requestFocus();
        return false;

    } else if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        etUserName.setError(getString(R.string.input_error_email_invalid));
        etUserName.requestFocus();
        return false;
    }else if (Password.isEmpty()) {
        etPassword.setError(getString(R.string.input_error_password));
```

## MIRINO (Android Application)

```
        etPassword.requestFocus();
        return false;
    }else{
        result= true;
        return result;
    }
}

public void onBackPressed() {
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
    alertDialogBuilder.setTitle("Exit Application?");
    alertDialogBuilder
        .setMessage("Click yes to exit!")
        .setCancelable(false)
        .setPositiveButton("Yes",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    android.os.Process.killProcess(android.os.Process.myPid());
                    System.exit(1);
                }
            })
        .setNegativeButton("No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {

                dialog.cancel();
            }
        });
    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
}
```

## MIRINO (Android Application)

### MapFromTo.java

```
package com.larebshaikh.mirino;

import android.Manifest;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.net.Uri;
import android.provider.Settings;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.CameraPosition;
```

## MIRINO (Android Application)

```
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.MultiplePermissionsReport;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.DexterError;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.PermissionRequestErrorListener;
import com.karumi.dexter.listener.multi.MultiplePermissionsListener;
import java.util.List;

public class MapsFromTo extends UsersOptionsMenu implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener , OnMapReadyCallback,
TaskLoadedCallback{

    //variables for map and route

    private GoogleMap mMap;
    private MarkerOptions place1, place2;
    Button getDirection;
    private Polyline currentPolyline;
    private MapFragment mapFragment;
    private boolean isFirstTime = true;
    private Toolbar toolbar;

    String toLat,toLong;
    Double lat,lon;
```

## MIRINO (Android Application)

```
//variables for current location
private static final String TAG = "MainActivity";

private TextView tvLocation;
private GoogleApiClient mGoogleApiClient;
private Location mLocation;
private LocationRequest mLocationRequest;
private com.google.android.gms.location.LocationListener listener;
private long UPDATE_INTERVAL = 2 * 1000; /* 10 secs */
private long FASTEST_INTERVAL = 2000; /* 2 sec */

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps_from_to);
    toolbar=findViewById(R.id.usertoolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle("Mosque Route");

    //code for getting current location
    requestMultiplePermissions();

    getIncomingIntents();
    if(!toLat.equals(null)||!toLong.equals(null))
    {
        lat = Double.parseDouble(toLat);
        lon = Double.parseDouble(toLong);
    }else {
        Toast.makeText(this,"Didn't Got Any Longitude Latitude, Exiting
Activity",Toast.LENGTH_SHORT).show();
        startActivity(new Intent(MapsFromTo.this,MosqueSearchActivity.class));
    }
}
```

## MIRINO (Android Application)

```
}  
mGoogleApiClient = new GoogleApiClient.Builder(this)  
    .addConnectionCallbacks(this)  
    .addOnConnectionFailedListener(this)  
    .addApi(LocationServices.API)  
    .build();  
  
}  
  
public void getIncomingIntents(){  
    if(getIntent().hasExtra("latitude")&&getIntent().hasExtra("longitude")){  
        toLat=getIntent().getStringExtra("latitude");  
        toLong=getIntent().getStringExtra("longitude");  
        Toast.makeText(this,"Lat :"+toLat+"  
Long"+toLong,Toast.LENGTH_SHORT).show();  
    }  
}  
  
//code for drawing route  
  
@Override  
public void onMapReady(GoogleMap googleMap) {  
    mMap = googleMap;  
    mMap.clear();  
    Log.d("mylog", "Added Markers");  
    mMap.addMarker(place1);  
    mMap.addMarker(place2);  
  
    CameraPosition googlePlex = CameraPosition.builder()  
        .target(new LatLng(lat,lon))  
        .zoom(11)
```

## MIRINO (Android Application)

```
.bearing(0)
.tilt(45)
.build();

mMap.animateCamera(CameraUpdateFactory.newCameraPosition(googlePlex), 5000,
null);
}

private String getUrl(LatLng origin, LatLng dest, String directionMode) {
    // Origin of route
    String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
    // Destination of route
    String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
    // Mode
    String mode = "mode=" + directionMode;
    // Building the parameters to the web service
    String parameters = str_origin + "&" + str_dest + "&" + mode;
    // Output format
    String output = "json";
    // Building the url to the web service
    String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" +
parameters + "&key=" + getString(R.string.google_maps_key);
    return url;
}

@Override
public void onTaskDone(Object... values) {
    if (currentPolyline != null)
        currentPolyline.remove();
    currentPolyline = mMap.addPolyline((PolylineOptions) values[0]);
}
```



## MIRINO (Android Application)

//runtime permission method

```
private void requestMultiplePermissions(){
    Dexter.withActivity(this)
        .withPermissions(
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION )
        .withListener(new MultiplePermissionsListener() {
            @Override
            public void onPermissionsChecked(MultiplePermissionsReport report) {
                // check if all permissions are granted
                if (report.areAllPermissionsGranted()) {
                    Toast.makeText(getApplicationContext(), "All permissions are granted by
user!", Toast.LENGTH_SHORT).show();
                }

                // check for permanent denial of any permission
                if (report.isAnyPermissionPermanentlyDenied()) {
                    // show alert dialog navigating to Settings
                    openSettingsDialog();
                }

                @Override
                public void onPermissionRationaleShouldBeShown(List<PermissionRequest>
permissions, PermissionToken token) {
                    token.continuePermissionRequest();
                }
            }).
            withErrorListener(new PermissionRequestErrorListener() {
                @Override
                public void onError(DexterError error) {
```

## MIRINO (Android Application)

```
        Toast.makeText(getApplicationContext(), "Some Error! ",
Toast.LENGTH_SHORT).show();
    }
})
.onSameThread()
.check();
}
private void openSettingsDialog() {

    AlertDialog.Builder builder = new AlertDialog.Builder(MapsFromTo.this);
    builder.setTitle("Required Permissions");
    builder.setMessage("This app require permission to use awesome feature. Grant them in
app settings.");
    builder.setPositiveButton("Take Me To SETTINGS", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
            Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
            Uri uri = Uri.fromParts("package", getPackageName(), null);
            intent.setData(uri);
            startActivityForResult(intent, 101);
        }
    });
    builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
});
```

## MIRINO (Android Application)

```
        builder.show();

    }

    //methods for getting current location

    @Override
    public void onConnected(Bundle bundle) {
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //    public void onRequestPermissionsResult(int requestCode, String[] permissions,
            //                                            int[] grantResults)
            // to handle the case where the user grants the permission. See the documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }

        startLocationUpdates();

        mLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);

        if(mLocation == null){
            startLocationUpdates();
        }

        if (mLocation != null) {
```

## MIRINO (Android Application)

```
// mLatitudeTextView.setText(String.valueOf(mLocation.getLatitude()));
//mLongitudeTextView.setText(String.valueOf(mLocation.getLongitude()));
} else {
    Toast.makeText(this, "Location not Detected", Toast.LENGTH_SHORT).show();
}
}
```

@Override

```
public void onConnectionSuspended(int i) {
    Log.i(TAG, "Connection Suspended");
    mGoogleApiClient.connect();
}
```

@Override

```
public void onConnectionFailed(ConnectionResult connectionResult) {
    Log.i(TAG, "Connection failed. Error: " + connectionResult.getErrorCode());
}
```

@Override

```
protected void onStart() {
    super.onStart();
    if (mGoogleApiClient != null) {
        mGoogleApiClient.connect();
    }
}
```

@Override

```
protected void onStop() {
    super.onStop();
    if (mGoogleApiClient.isConnected()) {
```

## MIRINO (Android Application)

```
mGoogleApiClient.disconnect();
    }
}

protected void startLocationUpdates() {
    // Create the location request
    mLocationRequest = LocationRequest.create()
        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
        .setInterval(UPDATE_INTERVAL)
        .setFastestInterval(FATEST_INTERVAL);
    // Request location updates
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //   ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                           int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
        mLocationRequest, this);
    Log.d("reque", "--->>>>");
}
```

## MIRINO (Android Application)

```
@Override
public void onLocationChanged(Location location) {

    String msg = "Updated Location: " +
        Double.toString(location.getLatitude()) + "," +
        Double.toString(location.getLongitude());

    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();

    if(isFirstTime){
        //code to draw path on map

        getDirection = findViewById(R.id.btnGetDirection);
        getDirection.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                new FetchURL(MapsFromTo.this).execute(getUrl(place1.getPosition(),
place2.getPosition(), "driving"), "driving");
            }
        });

        place1 = new MarkerOptions().position(new LatLng(location.getLatitude(),
location.getLongitude())).title("Your Location ");
        place2 = new MarkerOptions().position(new LatLng(lat, lon)).title("Mosque
Location");

        mapFragment = (MapFragment)
getFragmentManager().findFragmentById(R.id.mapNearBy);
        mapFragment.getMapAsync(this);
        isFirstTime = false;

    }
}
```

## **MIRINO (Android Application)**

### **MosqueVerify.Java**

```
package com.larebshaikh.mirino;

import android.content.DialogInterface;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AlertDialog;
import android.os.Bundle;

import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.SearchView;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.firebase.ui.database.FirebaseListAdapter;
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
```

## MIRINO (Android Application)

```
import java.util.HashMap;
import java.util.Map;

public class MosqueVerifyActivity extends AdminsOptionsMenu {
    private Toolbar admintoolbar;
    private FirebaseDatabase firebaseDatabase;
    private DatabaseReference ref;
    FirebaseListAdapter adapter;
    RecyclerView mosquerecycleview;
    ListView mosquelistview;
    SearchView mosquesearchView;
    String latitude, longitude;
    private RecyclerView mPeopleRV;
    private DatabaseReference mDatabase;
    private FirebaseRecyclerAdapter<MosqueDetails,
MosqueVerifyActivity.MosqueViewHolder> mPeopleRVAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mosque_verify);
        admintoolbar=findViewById(R.id.usertoolbar);
        setSupportActionBar(admintoolbar);
        getSupportActionBar().setTitle("Mosque Verify");

        mDatabase = FirebaseDatabase.getInstance().getReference().child("TempMosque");
        mDatabase.keepSynced(true);
```



## MIRINO (Android Application)

```
mPeopleRV = (RecyclerView) findViewById(R.id.myRecyclerview);

DatabaseReference personsRef =
FirebaseDatabase.getInstance().getReference().child("TempMosque");
Query personsQuery = personsRef.orderByKey();

mPeopleRV.hasFixedSize();
mPeopleRV.setLayoutManager(new LinearLayoutManager(this));

FirebaseRecyclerOptions personsOptions = new
FirebaseRecyclerOptions.Builder<MosqueDetails>().setQuery(personsQuery,
MosqueDetails.class).build();

mPeopleRVAdapter = new FirebaseRecyclerAdapter<MosqueDetails,
MosqueVerifyActivity.MosqueViewHolder>(personsOptions) {
    @Override
    protected void onBindViewHolder(MosqueVerifyActivity.MosqueViewHolder
holder, final int position, final MosqueDetails model) {
        holder.setMosqueName(model.getMosqueName());
        holder.setMosqueArea(model.getMosqueArea());
        holder.setFajr(model.getFajr());
        holder.setZohar(model.getZohar());
        holder.setAsr(model.getAsr());
        holder.setMaghrib(model.getMaghrib());
        holder.setIsha(model.getIsha());
        holder.setJuma(model.getJuma());
        String Longitude=model.getMosqueLongitude();
        String Latitude=model.getMosqueLatitude();
```

## MIRINO (Android Application)

```
holder.mView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String mosquename=model.getMosqueName();
        final String mosquearea=model.getMosqueArea();
        final String fajr=model.getFajr();
        final String zohar=model.getZohar();
        final String asr=model.getAsr();
        final String maghrib=model.getMaghrib();
        final String isha=model.getIsha();
        final String juma=model.getJuma();
        final String latitude=model.getMosqueLatitude();
        final String longitude=model.getMosqueLongitude();

        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(MosqueVerifyActivity.this);
        alertDialogBuilder.setTitle("Verify Mosque "+mosquename);
        alertDialogBuilder
            .setMessage("Accept or Reject Request")
            .setCancelable(false)
            .setPositiveButton("Accept Request",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        Map<String, String> mosque = new HashMap<>();

                        mosque.put("MosqueName", mosquename);
                        mosque.put("MosqueArea", mosquearea);
                        mosque.put("Fajr", fajr);
                        mosque.put("Zohar", zohar);
                        mosque.put("Asr",asr);
                        mosque.put("Maghrib",maghrib);
```

## MIRINO (Android Application)

```
mosque.put("Isha",isha);
mosque.put("Juma",juma);
mosque.put("MosqueLongitude",longitude);
mosque.put("MosqueLatitude",latitude);

FirebaseDatabase.getInstance().getReference("Mosque")
    .child(mosquename)
    .setValue(mosque).addOnCompleteListener(new
OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if (task.isSuccessful()) {
            Toast.makeText(MosqueVerifyActivity.this, "Verified
Successfully.", Toast.LENGTH_LONG).show();

            DatabaseReference
dltempMosque=FirebaseDatabase.getInstance().getReference("TempMosque").child(mosque
name);

            dltempMosque.removeValue();
            Toast.makeText(MosqueVerifyActivity.this, "Removed
The Values from Temp Table",Toast.LENGTH_SHORT).show();

        } else {
            Toast.makeText(MosqueVerifyActivity.this,
(task.getException()).getMessage(), Toast.LENGTH_LONG).show();
        }
    }
});dialog.cancel();
})

.setNegativeButton("Remove Request", new
```

## MIRINO (Android Application)

```
DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int id) {  
  
        DatabaseReference  
dltempMosque=FirebaseDatabase.getInstance().getReference("TempMosque").child(mosque  
name);  
  
        dltempMosque.removeValue();  
        Toast.makeText(MosqueVerifyActivity.this,"Removed The Values  
from Temp Table",Toast.LENGTH_SHORT).show();  
  
        dialog.cancel();  
    }  
});  
  
AlertDialog alertDialog = alertDialogBuilder.create();  
alertDialog.show();  
  
}  
});  
  
}  
  
@Override  
public MosqueVerifyActivity.MosqueViewHolder onCreateViewHolder(ViewGroup  
parent, int viewType) {  
  
    View view = LayoutInflater.from(parent.getContext())  
        .inflate(R.layout.mosquecardholder, parent, false);  
  
    return new MosqueVerifyActivity.MosqueViewHolder(view);  
}
```

## MIRINO (Android Application)

```
};
```

```
mPeopleRV.setAdapter(mPeopleRVAdapter);  
}
```

```
@Override  
public void onStart() {  
    super.onStart();  
    mPeopleRVAdapter.startListening();  
  
}
```

```
@Override  
protected void onStop() {  
    super.onStop();  
    mPeopleRVAdapter.startListening();  
}
```

```
public static class MosqueViewHolder extends RecyclerView.ViewHolder{  
    View mView;  
    public MosqueViewHolder(View itemView){  
        super(itemView);  
        mView = itemView;  
    }  
    public void setMosqueName(String name){  
        TextView mosqueName = (TextView)mView.findViewById(R.id.mosqueName);
```

## MIRINO (Android Application)

```
        mosquename.setText(name);
    }
    public void setMosqueArea(String area){
        TextView mosquearea = (TextView)mView.findViewById(R.id.mosquearea);
        mosquearea.setText(area);
    }
    public void setFajr(String fajr){
        TextView fajrtime = (TextView) mView.findViewById(R.id.FajrTime);
        fajrtime.setText(fajr);
    }
    public void setZohar(String zohar){
        TextView zohartime=(TextView) mView.findViewById(R.id.ZoharTime);
        zohartime.setText(zohar);
    }
    public void setAsr(String asr){
        TextView asrtime=(TextView) mView.findViewById(R.id.AsrTime);
        asrtime.setText(asr);
    }
    public void setMaghrib(String maghrib){
        TextView maghribtime=(TextView) mView.findViewById(R.id.MaghribTime);
        maghribtime.setText(maghrib);
    }
    public void setIsha(String isha){
        TextView ishatime=(TextView) mView.findViewById(R.id.IshaTime);
        ishatime.setText(isha);
    }
    public void setJuma(String juma){
        TextView jumatime=(TextView) mView.findViewById(R.id.jumaTime);
        jumatime.setText(juma);
    }
}
```

## MIRINO (Android Application)

```
public void onBackPressed() {  
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);  
    alertDialogBuilder.setTitle("Exit Application?");  
    alertDialogBuilder  
        .setMessage("Click yes to exit!")  
        .setCancelable(false)  
        .setPositiveButton("Yes",  
            new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialog, int id) {  
                    moveTaskToBack(true);  
                    android.os.Process.killProcess(android.os.Process.myPid());  
                    System.exit(1);  
                }  
            })  
        .setNegativeButton("No", new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialog, int id) {  
                    dialog.cancel();  
                }  
            });  
    AlertDialog alertDialog = alertDialogBuilder.create();  
    alertDialog.show();  
}
```

## **6.0 Conclusion and Future Work**

### **6.0.1 Conclusion :**

This Project was Built to provide Religious Service to the user as per the need, before this there was no such Android Navigation Based Work was There in Industry.

- The Application is interactive and friendly.
- Entire Application is fully automatic to the clients and satisfies the clients request
- The Application is Highly Secure.
- The Application provide various automated features like root calculation ,Viewing Mosque Namaz time and Reciting Quran.
- The Application is More useful for People, who travel on daily bases to new places, by using the application they can find religious places.
- The Application Automation make user and admin's task easier.

### **6.0.2 Future Work :**

- The Application Can be Enhanced by Adding live tracking feature.
- It Can be Enhanced by adding other religions Religious places.
- It can be enhanced by adding search filter for the user and the admin.

#### **➤ References:**

- StackOverFlow.com (to Debug the Various Error).
- Google Maps References by Google.
- Firebase References by Firebase.



