**2. System Analysis**

**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 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.

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.

**2.3.1 Functional Requirements**

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

* Functions of application:

|  |
| --- |
| * Provide filters. * Provide Search Results. * Provide Trust/Committee details. * Trace user activity. * Reminders. |

* Functions of admin:

|  |
| --- |
| * Updates details of Mosque. * Add an Mosque. * Delete an Mosque. * Maintain database on server. * Update and maintain applications as per end user requirements |

* User functionality :

|  |
| --- |
| * Searching Mosque around specified location. * Searching Trust/Committee around specified location * Doing Request To add Mosque * Getting info of these Trust/Committee. ( Contact number, website, timings) * Getting Route for the selected mosque. * Setting Reminders to recite Quran. * Getting Reminders For Namaz. |

**2.4 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. |

**2.5 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 2.x or 3.x. |

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 |

**2.6 Justification of Platform**

**Android Studio**

Android Studio is the official integrated development environment (IDE) for Google’s Android operating system, built on JetBrain’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.

**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.
* **Compact:** When optimized for size, the whole SQLite library with everything enabled is [less than 500KiB in size](https://www.sqlite.org/footprint.html) (as measured on an ix86 using the "size" utility from the GNU compiler suite.) Unneeded features can be disabled at compile-time to further reduce the size of the library to under 300KiB if desired.