Final Year Design Project System Requirements Specification

E-Masjid System

# Software Design Specification Document

by

| | |
|---|---|
| **Dawood Ahmed** | **089264 (2022-KS-158)** |
| **Haris Ehsan** | **089301 (2022-KS-190)** |

**(Evening)**

Project Advisor:

**Muhammad Kamran**

Faculty of Computing & Information Technology,

University of the Punjab, Lahore, Pakistan.

(20xx)

**E-Masjid System**

## Executive Summary

We have made this design document which provide the technical design for our project E-Masjid system. It is a web based platform which help mosque to manage their daily operation digitally. This system is design in a way to solve the basic problems of manual record keeping and lack of transparency by giving features for donation tracking, prayer time management, event organization and online nikah booking service.

We are using modern web technologies including React.js for the user interface, Node.js for the server, and MongoDB for data storage. The design includes secure payment processing with Stripe, proper user authentication, and responsive design for mobile compatibility. This document serves as a complete guide for our development team to build the system correctly and efficiently.

## Table of Contents

## List of Tables

## Tables of Figures

**No table of figures entries found.**

## System Design

### Product Perspective

Our E-Masjid System is a complete web based platform that will be accessible through any modern web browser. The system is designed to serve mosque administration who manages operations and community members who use these services.

### Dependencies

1. Stable internet connection for all users
2. Stripe service availability for payment processing
3. Modern web browsers supporting React.js features
4. MongoDB database server for data storage

### Interaction with Other Systems

1. **Stripe Payment Gateway:** It is used for making online donations securely
2. **Email Service:** It is used for sending password reset links
3. **Internet Connection:** It is required for all users to access the system

### Design Constraints

1. **Performance Requirements:** Prayer times page loads within 3 seconds, handles 100+ users during Friday prayers
2. **Usability Requirements:** Simple interface with large buttons, works on mobile and computer, elderly friendly design
3. **Security Requirements:** Encrypted passwords, secure payments through Stripe, admin access protection
4. **Technical Constraints:** MERN stack technology, responsive design, automatic weekly backups

## 1. Design Considerations

### Assumptions

Following are the assumptions:

1. Mosque administrators have basic computer knowledge
2. Users have internet access and email accounts
3. The mosque has at least one computer for admin use

4. Religious scholars can use basic web applications
5. Community members can use web browsers on phones or computers

**Dependencies**

Following are the dependencies:

1. Stable internet connection for all users
2. Stripe payment service available at any time
3. Web browsers supporting modern JavaScript
4. MongoDB database running properly

**Limitations**

Following are the limitations:

1. Cannot work without internet connection
2. No SMS notifications for announcements
3. No mobile app version
4. Payment system requires card payments only
5. Cannot handle offline data entry

**Risks**

Following are the risks:

1. Payment security issues
2. System downtime during prayer times
3. Elderly users finding the system difficult
4. Data loss from system crashes

## 2. Requirements Traceability Matrix

*Table 1 Requirements Traceability Matrix*

| **Requirement ID** | **Requirement Description** | **Design Specification** |
|---|---|---|
| FR-1 | The system will allow users to register and login with email and password, with different access levels for admin and community members | Component "User Authentication System" |

| | | |
|---|---|---|
| FR-2 | The system will show donation records, expense details, and financial reports so people can see both income and spending | Component "Financial Reporting Module" |
| FR-3 | The mosque admin will be able to record cash donations with donor name, amount, date, and donation type | Component "Donation Management System" |
| FR-4 | The admin will be able to add, update, or remove events and announcements such as islamic classes, community programs, and eid prayers. Users can view them on the main page | Component "Event & Announcement Manager" |
| FR-5 | Community members will be able to book nikah registrar for nikah ceremonies by selecting date and providing contact details | Component "Nikah Booking System" |
| FR-6 | The admin will be able to set and update daily prayer times including special timings for Jummah and Ramadan | Component "Prayer Times Manager" |
| FR-7 | Community members will be able to make donations online through the website by entering amount and personal details and process real payments using Stripe and see payment confirmation with receipt | Component "Online Payment Processor" |
| FR-8 | If users forget their password, they can reset it using their email address | Component "Password Recovery System" |
| FR-9 | Users can see if their Nikah service request is pending, accepted, or rejected | Component "Booking Status Tracker" |
| FR-10 | Admin can add where mosque money is spend like for repairs, electricity etc | Component "Expense Management System" |
| FR-11 | Admin can create special accounts for religious scholars who perform Nikah | Component "User Account Manager" |

## 3. Design Models

This section describes the various design models used to represent the E-Masjid System architecture. We use UML diagrams to visualize the system structure, data relationships, and component interactions to ensure clear understanding.

### 1.1 Architectural Design

Our E-Masjid System follows the MVC architecture pattern. This separates the system into three main parts.

- **Model:** Handles data and business logic
- **View:** User interface that users see
- **Controller:** Processes user requests and connect Model and View

**System Architecture Components**

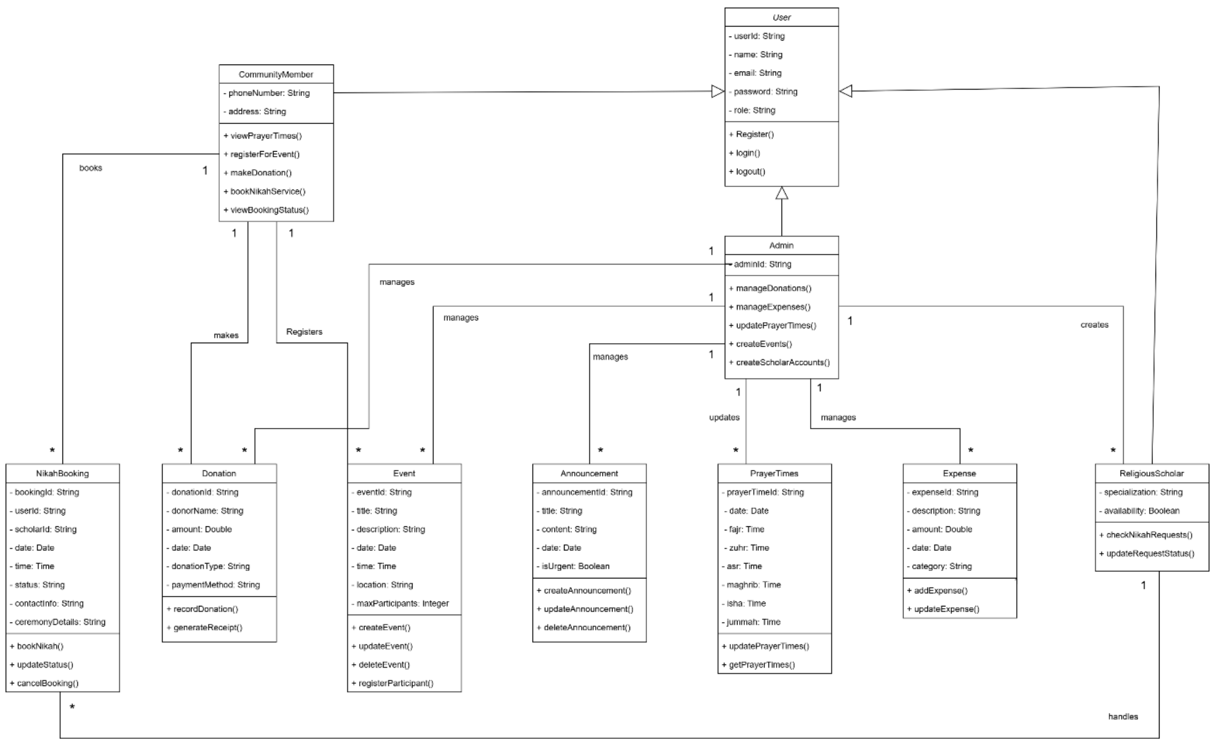
**Frontend Layer:**
- User Interface Components
- Mobile-responsive design
- Client-side validation
- Real-time updates

**Backend Layer:**
- API routes and controllers
- Business logic processing
- Authentication and authorization
- Payment processing

**Data Layer:**
- User data and profiles
- Donation and expense records
- Prayer time schedules
- Event and announcement data
- Nikah booking requests

**External Services:**
- Stripe Payment Gateway
- Email Service for password resets

## Class Diagram



## 1.2 Data Design

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed, and organized. This includes the dataset/ database design and/or other design of other data structures used.

### 1.2.1 Data Dictionary

- **Alphabetical List of System Entities or Major Data with Types and Descriptions:**

  - Prepare an alphabetical list of all system entities or significant data. For each entry, include the **data type** and provide a brief **description** of its purpose within the system.
  - Present this information in a **two-column format**:
    - The **first column** will contain the **terminology** (e.g., object name, attribute, method, or method parameter).
    - The **second column** will provide the **description**, including data types, roles, or any other relevant information

- **Structured Approach: Functions and Function Parameters:**

  - For systems following a structured programming paradigm, list **all functions** along with their **function parameters**. For each function, provide a description detailing its functionality, and for each parameter, specify the **data type** and its role within the function.

- **Object-Oriented (OO) Approach: Objects, Attributes, Methods, & Method Parameters:**

  - For systems using the Object-Oriented (OO) approach, list the **objects** and their **attributes**, followed by a description. Additionally, list the **methods** associated with each object, along with their **method parameters**.

## 1.3 User Interface Design

Describe overview of the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user. [1]

### 1.3.1 Screen Images

Display few screenshots showing the interface from the user's perspective. These can be hand-drawn, or you can use Prototyping tool like Canva/ Figma. Just make them as accurate as possible. [2]

[1] [User interface design will be presented in the Deliverable 4 - Prototype]

[2] Please don't forget to add the caption of each figure (Use References -> Insert Caption option in MS Word)

#### 1.3.2 Screen Objects and Actions

A discussion of screen objects and actions associated with those object for two use cases.

### 1.4 Behavioural Model

**Interaction Diagrams**: Includes diagrams (Sequence or Collaboration diagram) that illustrate interactions among components.[3]

**State Diagrams**: Shows state transitions within the software, especially useful for systems with complex workflows or lifecycle states.[4]

## 2. Design Decisions

Highlight the design choices while designing your system (e.g. choosing the Object oriented design pattern, choosing between the algorithmic approaches, normalization level of the database etc). Clearly specify which approach you have used and why you have made a specific design choice.

## 3. Summary

Summary of the key design ideas discussed, design refinement and decisions taken in the chapter. It may also reiterate chapter's importance in addressing the projects objectives

[3] . Make sure that objects & methods mentioned in the interaction diagram are consistent with the class diagram

[4] Make sure that each diagram has assigned a proper caption, so that table of figures can be created.

## References

List any documents or other resources to which this SRS refers, if any. These might include user interface style guides, standards, system requirements specifications, interface specifications, or the SRS for a related product. The following are a few examples of different resources.

**Book**
Author(s). Book title. Location: Publishing company, year, pp.
Example:
W.K. Chen. Linear Networks and Systems. Belmont, CA: Wadsworth, 1993, pp. 123-35.

**Article in a Journal**
Author(s). "Article title". Journal title, vol., pp, date.
Example:
G. Pevere. "Infrared Nation." The International Journal of Infrared Design, vol. 33, pp. 56-99, Jan. 1979.

**Articles from Conference Proceedings (published)**
Author(s). "Article title." Conference proceedings, year, pp.
Example:
D.B. Payne and H.G. Gunhold. "Digital sundials and broadband technology," in Proc. IOOC-ECOC, 1986, pp. 557-998.

**World Wide Web**
Author(s)*. "Title." Internet: complete URL, date updated* [date accessed].
M. Duncan. "Engineering Concepts on Ice. Internet: www.iceengg.edu/staff.html, Oct. 25, 2000 [Nov. 29, 2003].