Hamdard University

Department of Computing

Final Year Project

A Website for DoC, Hamdard University

FYP-001/FL24

Software Design Specifications

Submitted by

SHAMEER ABID (2278-2021)

SYEDA NOREEN ZAHRA (2025-2021)

HAMZA SHAIKH (2595-2021)

Supervisor(s)

SIR AFZAL HUSSAIN

FALL 2024

Document Sign off Sheet

1.1.1 Document Information

Project Title	A website for Department of computing
Project Code	**FYP-001/FL24**
Document Name	Software Design Specifications
Document Version	1.0
Document Identifier	**FYP-001/FL24-**SRS
Document Status	Final
Author(s)	Syeda Noreen Zahra, Shahmeer Abid, Hamza Sheikh
Approver(s)	Sir Afzal Hussain
Issue Date	Date of issuance of this document

Name	**Role**	**Signature**	**Date**
SHAMEER ABID	Team Lead		
SYEDA NOREEN ZAHRA	Team Member 2		
HAMZA SHAIKH	Team Member 3		
SIR AFZAL HUSSAIN	Supervisor		
	Co-Supervisor		
	Project Coordinator		

Revision History

Date	Version	Description	Author
17/Jan/2025	1.0	Details of the Changes made	Syeda Noreen Zahra

Definition of Terms, Acronyms, and Abbreviations

Term	**Description**
DoC	Department of Computing
FYP	Final Year Project
CMS	Content Management System
LMS	Learning Management System
Dynamic Website	A website that updates its content dynamically based on user interaction or data changes.
Chatbot	An AI-based tool designed to provide automated responses to frequently asked questions.
Timetable	A schedule of classes and activities organized by section and semester.
Database	An organized collection of data stored electronically for easy access, management, and retrieval.
SQL	Structured Query Language, used for managing and querying relational databases.
Prototype Methodology	A software development approach focused on iterative refinement based on user feedback.
Admission Form	A form used to collect details from prospective students applying to the department.
Complaint Box	A feature allowing users to submit feedback or report issues for resolution.
Library Database	An online catalog providing information about books and resources available in the library.
Frontend	The part of a website that users interact with directly, including design and layout.
Backend	The server-side logic of a website that handles data storage, processing, and functionality.

2 Table of Contents

1 Introduction

1.1 Purpose of Document

The purpose of this Software Designs Specification (SDS) document is to describe everything needed to build a website for the Department of Computing at Hamdard University. This website will make it easy for students, faculty, and administrators to access information, perform tasks, and stay updated.
Here's why this document is important:

1. *Clarify What Needs to Be Built It explains all the features and tools the website will include, like subpages for programs (Computer Science, AI, etc.), faculty details, timetables, announcements, and automated forms.*
2. *Centralize All Requirements It gathers all the technical and functional details in one place, ensuring everyone working on the project understands the goals and how the website should function.*
3. *Improve Communication the website will allow:*

- *Students to access information easily.*
- *Faculty and administrators to make announcements and manage tasks efficiently.*
- *A chatbot to answer user questions, connected to the main university website.*

4. *Save Time with Automation It defines how various forms (admission, applications, fee concessions, complaints) will be submitted and automatically emailed to the relevant person. If there's no response in 24 hours, the issue will escalate to the DEAN.*
5. *Guide the Technical Team It specifies the tools, software, and hardware required to build the website, ensuring the development team knows what technology to use.*
6. *Define Roles and Responsibilities It documents who can do what on the website (e.g., only the DEAN and HOD can post announcements, while coordinators can update timetables).*

Intended Audience

The purpose of this project is to:

1. *Current students and faculty of computing department at Hamdard University.*
2. *Prospective students interested in the Department of Computing and its sub-disciplines.*
3. *Visitors seeking information about FEST and its computing department.*

Document Convention

***Font**: Calibri (Body)*
***Font Size**: 12*
***Diagrams**: UML diagrams are used to represent the system design visually.*

Project Overview

The Department of Computing website is designed to provide a centralized platform for managing academic information and processes. Key functionalities include:

- *Sub-web pages for Software Engineering, Computer Science, Artificial Intelligence, and Computer Systems Engineering.*
- *Announcement areas managed by authorized personnel.*
- *A chatbot to assist users with queries related to Hamdard University.*

The development employs modern web technologies such as HTML, CSS, JavaScript, React.js, Node.js, or Python, with a robust database backend (SQL lite or PostgreSQL).

Scope

***Department Overview Pages**: Comprehensive pages dedicated to Computer Science, and Software Engineering, Artificial Intelligence each detailing faculty members, batch advisors, coordinators, and the department head.*

Announcement System**: A portal where only the HOD and Dean can log in to make announcements to students and faculty**.

***Chatbot Integration**: A chatbot that connects to the Hamdard University website to provide responses to frequently asked questions and queries, leveraging the university's existing information.*

2 Design Considerations

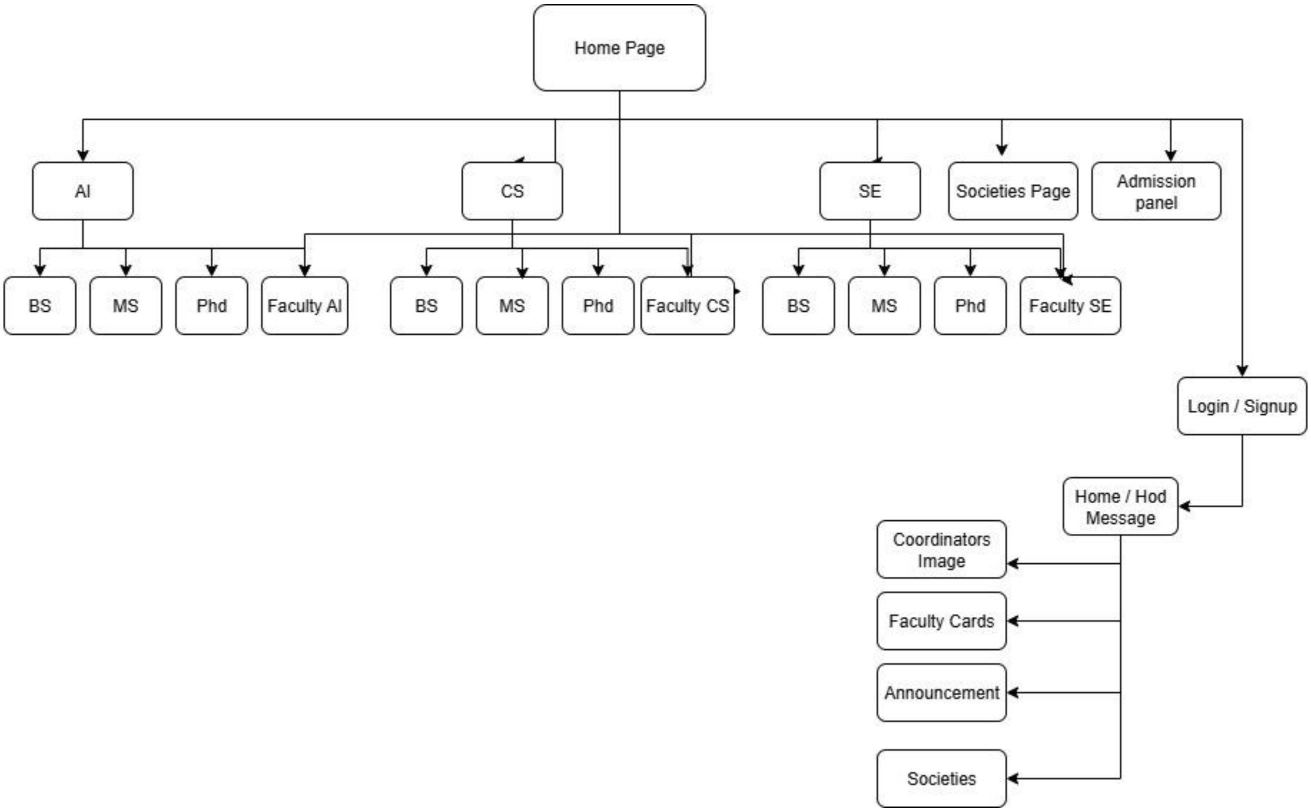
Assumptions and Dependencies

***Modular Architecture**: Ensuring seamless integration and testing in a scalable modular design.*
***Technology Compatibility**: Managing potential issues between frameworks like React.js, Node.js/Django, and databases.*
***UI/UX Challenges**: Creating accessible, responsive interfaces across devices per WCAG guidelines.*
***Database Optimization**: SQL lite*
***Concurrency Management**: Handling multiple users with efficient backend programming and load balancing.*
***Security Measures**: Implementing robust access control, encryption, and activity logging without performance loss.*
***Scalability**: Preparing the system for future enhancements like advanced AI and real-time notifications.*
***Error Recovery**: Designing systems for graceful failure handling and maintaining data integrity.*
***Testing Integration**: Ensuring components are testable during iterative development.*
***Resource Constraints**: Addressing limited developer resources with streamlined workflows and tool reliance.*

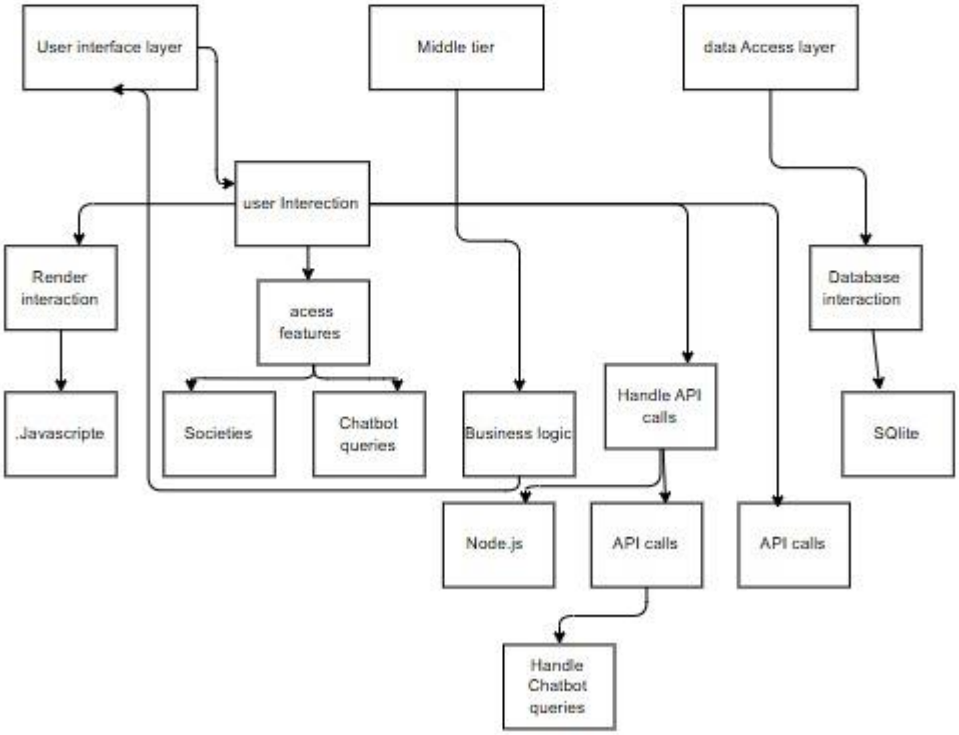
Risks and Volatile Areas

Requirement Changes: New feature requests may arise. Mitigation: Use modular design and agile methods.
Technology Risks: Updates to frameworks may cause compatibility issues. Mitigation: Use wellsupported tools and monitor trends.
Performance Bottlenecks: High user loads or database queries may slow the system. Mitigation: Conduct load testing and optimize queries.
Security Vulnerabilities: Risks of breaches or attacks. Mitigation: Enforce strict security protocols and conduct audits.
Resource Constraints: Limited skilled personnel could delay progress. Mitigation: Cross-train staff and outsource tasks as needed.
Integration Challenges: Issues with connecting APIs or external systems. Mitigation: Plan robust integrations and test early.
Data Inconsistencies: Outdated or incorrect data may affect usability. Mitigation: Automate validation and verify sources regularly.
Scalability Issues: Future needs might outgrow current design. Mitigation: Use scalable cloud-based and microservices architecture.

System Architecture



Software Architecture



Design Strategy

1. **Future System Extension or Enhancement**

- ***Design Strategy***:
  - *Modular architecture ensures that components can be added or replaced without significant impact on the overall system.*
  - *Use of APIs enables the integration of new features such as advanced analytics, mobile applications, or new user interfaces.*
  - *The system is designed to support additional sub-webpages for new departments or courses with minimal effort.*
  - *Advanced AI models can be incorporated into the chatbot in future iterations.*
- ***Reasoning***:
  - *Aligning with long-term scalability goals ensures the system remains relevant and adaptable to changing requirements.*
- ***Trade-offs***:
  - *Initial design complexity is increased to accommodate modularity.*
  - *Slightly higher development time due to the need for generalized and extensible components.*

2. System Reuse

- **Design Strategy**:
  - Use of reusable components such as:
    - ✦ React.js components for consistent UI elements.
    - ✦ RESTful APIs for handling data across multiple client applications (e.g., web and mobile).
    - ✦ Common utility modules for logging, error handling, and data validation.
  - Database schemas designed for flexibility and reusability across different modules.
- **Reasoning**:
  - Reusable components save time and resources during both initial development and future updates.
  - Encourages consistency across the system.
- **Trade-offs**:
  - Slightly higher upfront effort to ensure components are generic and reusable. o Potential underuse of some reusable components in the initial deployment phase.

3. User Interface Paradigms • Design Strategy:

o The user interface follows a **responsive and user-friendly design**:

- ✦ WCAG (Web Content Accessibility Guidelines) compliance ensures accessibility for all users, including those with disabilities.
- ✦ A mobile-first approach using frameworks like **Bootstrap** or **Material-UI** ensures compatibility across devices.
- ✦ Clear navigation structure reduces the learning curve for users.
- ✦ Contextual help such as tooltips and a chatbot improves user interaction.

- **Reasoning**:

  o A well-designed UI is essential for user satisfaction and adoption of the system.
  - Accessibility ensures the platform can be used by a broader audience.

- **Trade-offs**:
  - Increased development time for ensuring cross-platform compatibility and accessibility compliance.
  - Requires additional testing across a variety of devices and screen sizes.

4. Data Management (Storage, Distribution, Persistence)

- **Design Strategy**:
  - Hybrid database approach:

▪ **SQL lite** for structured relational data like timetables and user information.

  - Use of database replication and periodic backups to ensure high availability and data persistence.
  - APIs for efficient data distribution and retrieval.
  - Optimized indexing and query mechanisms to handle large datasets.

- ***Reasoning***: *o A hybrid database design allows the system to handle diverse data types efficiently.*
  - *Regular backups and replication ensure data security and availability.*
- ***Trade-offs***:
  - *Increased system complexity due to managing two database systems.*
  - *Higher resource requirements for maintaining database synchronization and backups.*

5. Concurrency and Synchronization • Design Strategy:

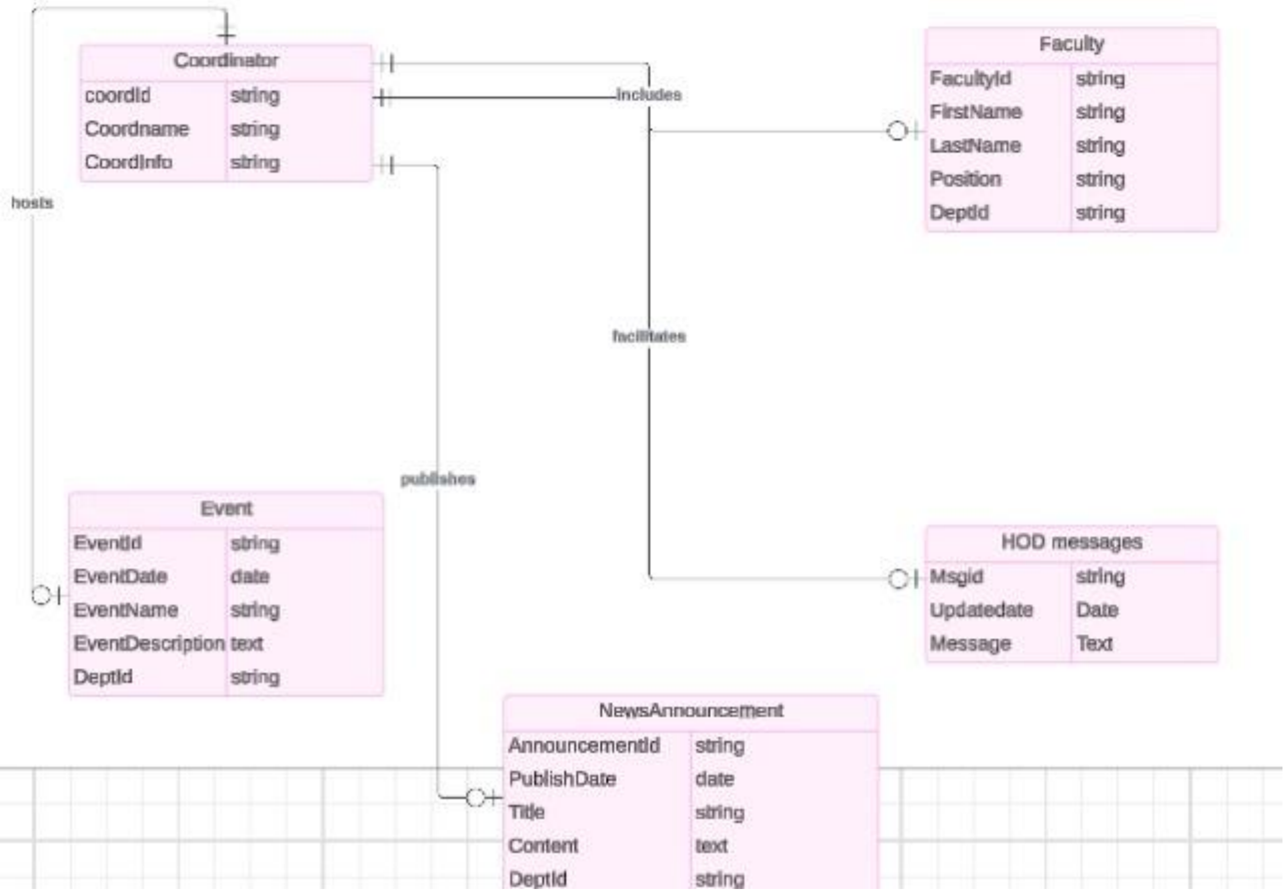
- *Use of asynchronous programming in **Node.js** or **Django** to handle multiple user requests simultaneously.*
- *Synchronization mechanisms such as:*
  - ***Optimistic concurrency control** to prevent conflicts in simultaneous database updates.*
  - *Distributed locks or similar mechanisms to ensure data consistency.*
- *Load balancing techniques to distribute requests evenly across servers.*

- ***Reasoning***:
  - *High concurrency support ensures smooth operation during peak usage times.*
  - *Synchronization mechanisms maintain data integrity.*
- ***Trade-offs***:
  - *Higher resource utilization due to concurrency management.*
  - *Slight increase in system latency when synchronization mechanisms are employed.*

Detailed System Design Design Class Diagram

Database Design

ER Diagram



Announcement Table

- ***Id:*** *Unique identifier for each announcement.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The announcement text.*
- ***DatePosted:*** *Date the announcement was posted.*

HOD Message

- ***Id:*** *Unique identifier for each message.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The message text.*

- ***DatePosted:*** *Date the message was posted.*

Coordinator image

- ***Id:*** *Unique identifier for each image.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The coordinator image.*
- ***DatePosted:*** *Date the image was posted.*

Faculty Card

- ***Id:*** *Unique identifier for each card.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The coordinator image.*
- ***DatePosted:*** *Date the card was posted.*

Events / Societies images

- ***Id:*** *Unique identifier for each image.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The event image.*
- ***DatePosted:*** *Date the imagewas posted.*

Login

- ***Id:*** *Unique identifier for each image.*
- ***UserId:*** *Foreign key referencing the User table.*
- ***Content:*** *The user will be loged in.*
- ***DatePosted:*** *Date the user logged in.*

Relationships:

- ***User*** *has a many-to-many relationship with* ***Announcement****, meaning two user can post multiple announcements.*
- ***User*** *has a one-to-many relationship with* ***Messages****, meaning one user can post multiple messages.*
- ***User*** *has a one-to-many relationship with* ***Faculty cards****, meaning one user can post multiple announcements.*
- ***User*** *has a one-to-many relationship with* ***coordinator image****, meaning one user can upload multiple coordinators.*
- ***User*** *has a many-to-many relationship with* ***societies/events image****, meaning two user can upload multiple images.*
- ***User*** *has a one-to-one relationship with* ***HOD login panel****, meaning one user can login to HOD panel.*

- ***User** has a one-to-one relationship with **coordinator's login**, meaning one user can login to coordinator's panel*

Data Dictionary

Entity	***Attribute***	***Description***	***Type***	***Constraints***
User	*user_id (PK)*	*Unique identifier for users*	*INT*	*Primary Key, Auto-increment*
	name	*Name of the user*	*VARCHAR(255)*	*Not Null*
	email	*Email address*	*VARCHAR(255)*	*Not Null, Unique*
	password	*Password for authentication*	*VARCHAR(255)*	*Not Null*
	role	*Role of the user (e.g., student, faculty, admin)*	*ENUM*	*Values: 'student', 'faculty', etc.*
Message	*message_id (PK)*	*Unique ID for complaints*	*INT*	*Primary Key, Auto-increment*
	description	*Description of the complaint*	*TEXT*	*Not Null*
	status	*Current status of the complaint*	*ENUM*	*Values: 'open', 'closed', etc.*
	created_at	*Timestamp for when the message was submitted*	*DATETIME*	*Not Null*
Society	*society_id (PK)*	*Unique ID for society*	*INT*	*Primary Key, Auto-increment*
	batch	*Batch associated with the timetable*	*VARCHAR(50)*	*Not Null*
	program	*Program name*	*VARCHAR(50)*	*Not Null*
Announcement	*announcement_id (PK)*	*Unique ID for announcements*	*INT*	*Primary Key, Auto-increment*

	author_id (FK)	*User who created the announcement*	*INT*	*Foreign Key to User.user_id*
	message	*Content of the announcement*	*TEXT*	*Not Null*
	created_at	*Timestamp for when the announcement was created*	*DATETIME*	*Not Null*

Data 1

Data 1						
Name	*Give primary name of the data or control item, the data store or an external entity.*					
Alias	*System User, Account Holder*					
Whereused/how-used	*Used in authentication processes (input to login).* *Referenced in complaints, announcements, and chatbot interactions.* *Acts as a control entity for role-based access.*					
Content description	*Represents all users (students, faculty, admin) interacting with the system.*					
Column Name	***Description of the Column***	***Type***	***Length***	***Null able***	***Default Value***	***Key Type***
user_id	***Unique identifier for users***	***INT***	***-***	***No***	***Autoincrement***	***PK***
name	***Name of the user***	***VARCHAR***	***255***	***No***	***-***	
email	***Email address***	***VARCHAR***	***255***	***No***	***-***	***UNIQUE***
password	***Password for authentication***	***VARCHAR***	***255***	***No***	***-***	

Data 2

4.2.2.4

Name	*Announcement* *News Update, Notification* *Stores announcements for the department.*				
Alias	*News Update, Notification*				
Where-used/howused	• *Created by admin or HOD for department-wide updates.* • *Accessible to students and faculty.*				
Content description	*Stores announcements for the department.*				
Column Name	*Description of the Column*	*Type*	*Length*	*Null able*	*Default Value*
announcement_id	*Unique ID for announcements*	*INT*	*-*	*No*	*Auto-increme*
author_id	*User who created the announcement*	*INT*	*-*	*No*	*-*
message	*Content of the announcement*	*TEXT*	*-*	*No*	*-*
created_at	*Timestamp of announcement creation*	*DATETIME*	*255*	*No*	*CURRENT_TI*

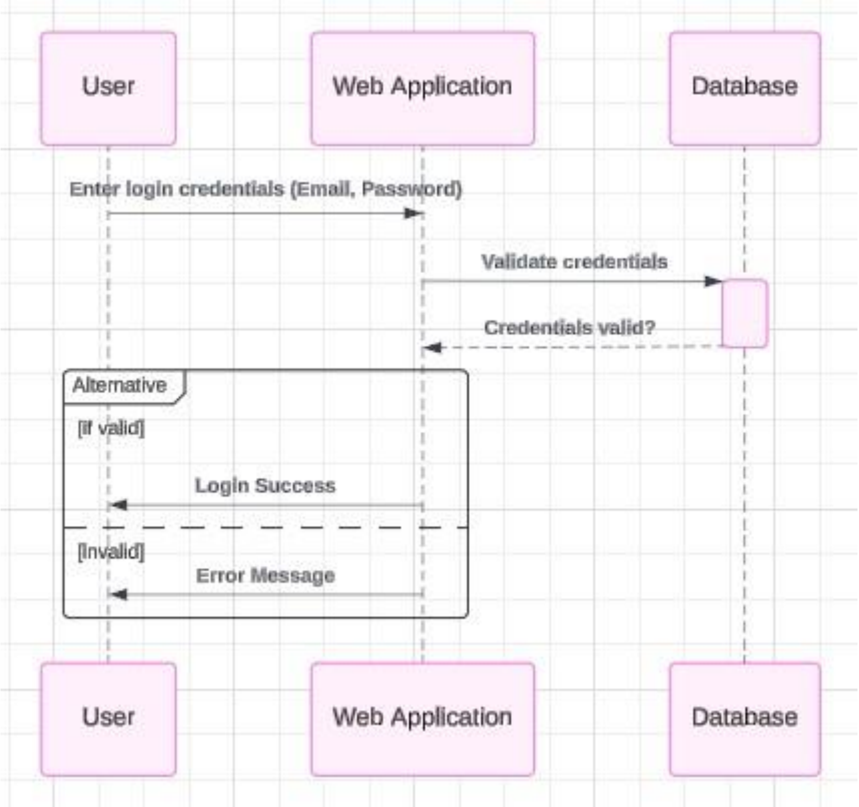
4.2.2.5
Data 5

Name	*Society*
Alias	*Society images*
Whereused/howused	• *Upload society events images*

Content description	*Upload societies event image .*					
Column Name	***Description of the Column***	***Type***	***Length***	***Null able***	***Default Value***	***Key Type***
query_id	*Unique ID for society images*	*INT*	*-*	*No*	*Auto-increment*	*PK*
user_id	*User interacting with the images*	*INT*	*-*	*No*	*-*	*FK*
query	*User's query*	*TEXT*		*No*	*-*	
timestamp	*Timestamp of interaction*	*DATETIME*	*-*	*No*	*CURRENT_TIMESTAMP*	

Application Design

<Sequence Diagram 1> User login process

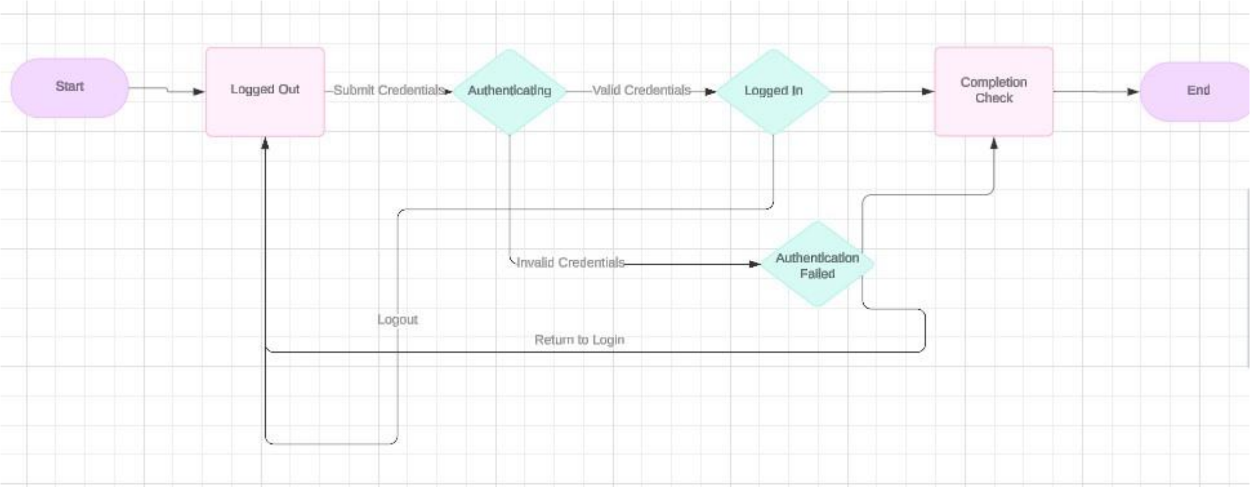


Explanation:

User enters login credentials (Email, Password)
Web Application sends credentials to Database for validation.
Database validates credentials.
Database sends validation result to Web Application.
If credentials are valid, the Web Application sends a "Login Success" message to the User. If credentials are invalid, the Web Application sends an "Error Message" to the User.

State Diagram

<State Diagram 1>User Authentication

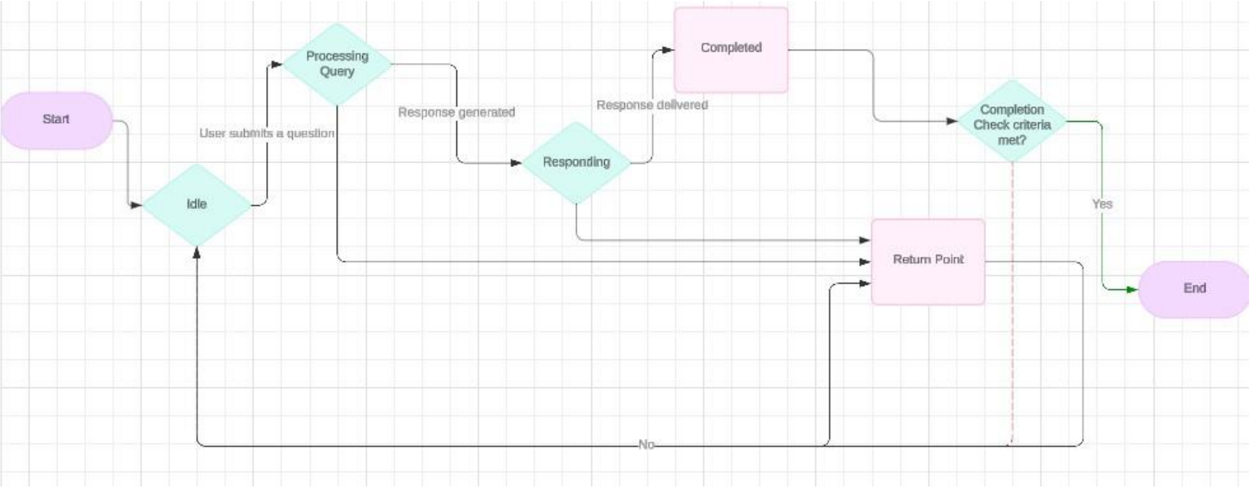


Explanation: Start: *The process begins.*
Logged Out: The user is not logged in.
Submit Credentials: The user provides their login credentials.
Authenticating: The system validates the provided credentials.

- *Valid Credentials: The credentials are correct. The process continues.*
- *Invalid Credentials: The credentials are incorrect. The user is redirected to the Login page.*

Logged In: The user is successfully logged in.
Completion Check: The system verifies the user's login status.
End: The process is complete.
Authentication Failed: If the login fails, the user is presented with an authentication failure message.
Logout: The user can log out of the system, which takes them back to the Logged Out state.
Return to Login: The user is redirected to the Login page after an unsuccessful login attempt.

<State Diagram > Chatbot Interaction



Explanation:

Start: *The process starts with a user submitting a question.*

Idle: *The system is in an idle state, waiting for a user query.*

Processing Query: *Once a question is submitted, the system moves into the "Processing Query" state, where the question is processed.*

Responding: *The system generates a response to the question and enters the "Responding" state.*

Completed: *The system delivers the response to the user and marks the query as "Completed".*

Completion Check Criteria Met? *The system checks whether all completion criteria have been met.*

End: *If the completion criteria are met, the process ends.*

Return Point: *If the completion criteria are not met, the process returns to the "Return Point" and continues to cycle through the process until the criteria are met.*

4.4 GUI Design

4.4.1 <Use Case Name - Mock Screen 1>

4.4.2 <Use Case Name - Mock Screen 2>

..

4.4.3 <Use Case Name - Mock Screen 3>

5 References

[This section should provide a complete list of all documents referenced at specific point in time. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained (This section is like the bibliography in a published book)].

6 Appendices

[Include supporting detail that would be too distracting to include in the main body of the document.]