MindSched: A Web-Based Mental Health Appointment System

1. Requirement Analysis

1.1. Project Objectives

The MindSched project aims to develop a three-tier web-based application to address the needs of patients and doctors in effective management of mental health appointments. The three tiers of the application include:

- **Patients**:

  Helping them to book, modify, and cancel appointments with mental health consultants.

- **Doctors(psychiatrist)**:

  Allowing them to effectively manage patient appointments, track their schedules, and access essential patient information.

- **Administrative Tracking**:

  Ensuring all actions performed within the system are stored effectively for future use and includes all changes such as inserts, updates, and deletes in the database.

This project focuses on creating an efficient and user-friendly system with the ability to scale for future use.

1.2. Functional Requirements

User Authentication:

- **Registration**:

  Both the patients and the psychiatrist can sign up on the system. For registration, few basic details like name, email, and password are required. The system asks for role during signup process that whether a person is a patient or a doctor(psychiatrist). The level of access of system depends on the role of a person

- **Login**:

  After successful registration, users can log in with their credentials that were saved during registration. The Flask-Login library is used for secure session management that makes ensure that after authentication users can only access their respective dashboards.

Appointment Management:

- **Booking Appointments**:

  Patients can check available doctors and book appointments based on their availability. When a patient selects a psychiatrist and time slot, the system records the appointment in the database and if two patients select the same psychiatrist at same time it tells that doctor is not available at this time.

- **Modifying and Cancelling Appointments**:

Patients can modify or cancel appointments through their dashboard. Any updates made to an appointment are shown in real-time.

- **Doctor(psychiatrist) Availability**:

  Doctors can set their availability on specific dates and times and helps patients to book only during those hours plus they can also delete or edit the appointments according to their free slots.

Audit Logging

- **Tracking Changes**:

  MySQL triggers are used to automatically locate every change in the database. Whenever a change like patient cancels an appointment and psychiatrist updates their schedule, the designed system records the action in the Audit Log Table and captures the related important information like the type of change and the timestamp.

Responsive UI

- The system is designed to be responsive and it works on both desktop and mobile devices. This ensures that patients and doctors can access the system from any device with ease.
- All design was implemented using custom CSS that makes ensure that it is completely flexible.

1.3. Non-Functional Requirements

Security

- Initially, the system stores passwords as plain text in the database but they can also be protected more efficiently using encryption decryption.

Scalability

- **Database**:

MySQL was chosen for database design because it ensures reliable and consistent transactions. MySQL also scales well as the number of users and appointments grows.

- **Indexing**:

Indexes are applied to frequently accessed fields such as doctor ID, patient ID, and appointment ID, which improves performance when querying large datasets.

Performance

- **Optimized SQL Queries**:

Every SQL query is written to reduce the complexity of system and optimize its performance. For example, joins are used efficiently to retrieve related data in a single query which reduces the need for multiple database hits.

1.4 Challenges and Risks

Challenge	Mitigation Strategy
Storing passwords	Implementation of bcrypt and other decryption hashing can be used in future versions to securely store passwords.
Booking appointments	Implementation of transaction locks to prevent multiple users from booking the same slot at the same time.
Mobile responsiveness	Custom CSS media queries were written to ensure the design adapts to both large screens and smaller mobile screens.
Handling appointment cancellations	Triggers in the MySQL database can be used to ensure that cancellations and updates are tracked and logged in real-time.

2. Backend Design

2.1. Logical Design

System Architecture

The system follows a three-tier architecture to improve overall efficiency of the system and to make it user friendly and responsie.

1. **Presentation Tier:**

   The HTML, CSS components are developed using VSCode that are responsible for rendering the user interface. This is the layer where users interact with the system.

2. **Business Logic Tier:**

   Flask handles routing, user authentication, and business logic. It processes the input from the frontend and communicates with the database to perform actions such as booking appointments, updating records, and retrieving data.

3. **Data Tier:**

   MySQL is used as the database to store all necessary information and also includes user credentials, doctor schedules, patient appointments and all database data.

Key Components

- **User Management**:

  This handles user roles and secure authentication. It ensures that only authorized users can access specific functionalities in the system. The unauthorized person cannot access any part of database.

- **Appointment System**:

  This component manages the entire booking process, including creating, updating, and deleting appointments based on user input. This is like the heart of system because it helps in collecting and dividing all data into respective booths.

- **Audit Logging**:

Automatically logs all database changes to ensure full accountability and track every action performed in the system. This helps in keeping track of whatever change is made in the database and helps in proper functioning of whole system.

2.2. Physical Design

Database Design

- **MySQL Database**:

The database schema was designed with scalability and efficiency in mind. Tables are normalized to avoid redundancy and ensure integrity.

- **Tables**:

  - **User Table**: Stores id, username, password, email, and role for each user.
  - **Doctors Table**: Stores id, name, email, and specialization for each doctor.
  - **Patients Table**: Stores id, name, appointment details, disease, and contact information.
  - **Audit Log Table**: Stores log entries for each action in the system, including the type of change and the user responsible.

Performance and Security

- **Indexing**:

Indexes are created on key columns such as patient ID, doctor ID, and audit log ID to improve data retrieval speed.

- **Triggers:**

MySQL triggers are used to log actions in the audit table automatically. This ensures that every action like appointments and cancellation is logged without additional manual intervention.

3. Business Logic – Middle Tier Design

3.1. UML Diagrams

Structural Design: Class Diagram

- The application uses Flask models and SQLAlchemy to interact with the database. The db.Model class is used to define the schema for each table. This database consists of different classes like patients, doctor etc. These UMLs are attached below.

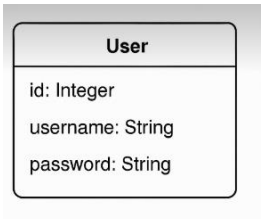

Figure 1: User's UML Diagram

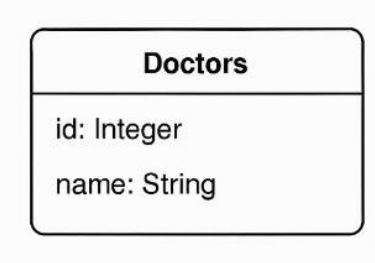

Figure 2: Doctor's UML Diagram

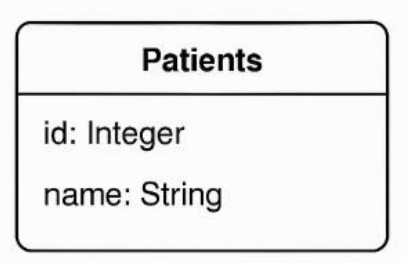

Figure 3: Patient's UML Diagram

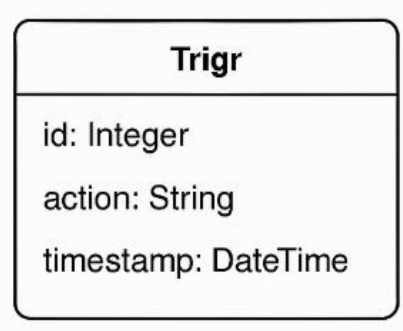

Figure 4: Triger's UML Diagram

Behavioural Design: Sequence Diagram

1. **When Patient books appointment**:

- **Frontend → Flask → MySQL**:

  Patient submits booking form on front end page and then Flask processes the request of patient and stores the data in the database and at the end triggers the creation of an audit log.

2. **How doctor views appointments**:

- **Frontend → Flask → MySQL**:

  Flask queries the appointments for the doctor from the Patients table and sends the data to the frontend for display.

4. Frontend Design

4.1. Tools

- **VSCode**:

The development environment where all frontend code was written.

- **HTML/CSS**:

The core technologies that are used to structure and style the pages. HTML provides the layout, while CSS is used to define the appearance of the system.

4.2. UI Components

Page	Functionality
Login/Signup	Role-based access control, users can sign in or register.
Patient Dashboard	Patients can view, book, modify, or cancel appointments.
Doctor Dashboard	Doctors can manage their schedules and view patient details.
Audit Logs	Displays actions and modifications made in the system (audit trail).

4.3. Custom CSS

- **Flexibility:**

Custom CSS was chosen for complete flexibility, as it allows the system to have a tailored and unique design without the limitations of external frameworks.

- **Responsiveness**:

Through media queries, the CSS adapts the layout for different screen sizes, ensuring the application is responsive and accessible on both desktop and mobile devices.

5. Output:

Complete working of system's video is uploaded on LMS.

6. Code:

Complete code of system is uploaded on LMS in form of zip file.