Final Project Report for Healthcare Appointments System

Use Case Diagrams:

1. Register User (Patient/Doctor):

Use Case Name: Register user
Actor: Patient, Doctor
Description: The patient or doctor provides personal information to create an account on the system.
Trigger: The user wants to use the healthcare appointment system.
Normal Course: 1. The user selects Sign up. 2. User fills in personal details, including email, password, and role 3. The system validates the information. 4. The system verifies data and creates an account. 5. An account is created, and a confirmation message is displayed.
Postconditions: The user's account is successfully created and stored.

2. Log in User:

Use Case Name: Login user
Actor: Patient, Doctor
Description: The patient or doctor logs into the system to access their respective dashboard.
Trigger: The user wants to manage appointments or availability.
Normal Course: 1. User selects "Login." 2. User enters credentials (email, password). 3. The system verifies credentials. 4. The user is redirected to the dashboard
Postconditions: The user is logged in.

3. Reset Password:

Use Case Name: Reset Password
Actor: System, Patient, Doctor
Description: The user resets their account password.
Trigger: The user forgets their password.
Normal Course: 1. User selects "Forgot Password." 2. User enters their registered email.

3. The system verifies the email and allows the user to set a new password. 4. Password is updated in the system
Postconditions: The user's password is reset.

4. View Doctor Dashboard:

Use Case Name: View Doctor Dashboard
Actor: Doctor
Description: The doctor views their profile, availability, and appointment history.
Trigger: The doctor logs in and navigates to their dashboard.
Normal Course: 1. Doctor logs into the system. 2. The system displays the doctor's profile, manage availability options, and appointment history. 3. Doctor can add, cancel availability, or update profile.
Postconditions: The doctor’s information and actions are accessible and manageable.

5. View Patient Dashboard:

Use Case Name: View Patient Dashboard
Actor: Patient
Description: The patient views their profile, upcoming appointments, and history.
Trigger: The patient logs in and navigates to their dashboard.
Normal Course: 1. Patient logs into the system. 2. The system displays user information, upcoming appointments, and appointment history. 3. Patient can book or cancel appointments and update their profile.
Postconditions: The patient's information is displayed, and appointment actions are accessible.

6. Update User Profile:

Use Case Name: Update User Profile
Actor: Patient, Doctor
Description: The user updates their personal information.
Trigger: The user wants to edit their profile details.
Normal Course: 4. User selects "Edit Profile." 5. User updates the desired fields. 6. The system validates and saves the changes.
Postconditions: Profile is successfully updated.

7. Book an Appointment:

Use Case Name: Book an Appointment
Actor: Patient
Description: The patient books an appointment with a doctor based on availability
Trigger: The patient selects a doctor and a suitable time slot.
Normal Course: 1. The patient selects a doctor from the search results. 2. The patient selects an available time slot. 3. The system confirms the booking.
Postconditions: Appointment is created and saved in the system.

8. Cancel an Appointment:

Use Case Name: cancel an appointment
Actor: Patient, Doctor
Description: The patient or doctor cancels an existing appointment.
Trigger: The user wants to change or cancel an appointment.
Normal Course: 1. User selects cancel an appointment from the dashboard. 2. User selects an appointment from the dashboard 3. User confirms cancellation. 4. The system updates the appointment status to canceled.
Postconditions: Appointment is updated or canceled.

9. Manage Doctor Availability:

Use Case Name: Manage Doctor Availability
Actor: Doctor
Description: The doctor adds or cancels availability slots.
Trigger: The doctor needs to update their schedule.
Normal Course: 1. Doctor selects "Add Availability" or "Cancel Availability." 2. Doctor specifies the date and time. 3. The system updates the availability records..
Postconditions: Availability is successfully managed.

10. Generate Completed Appointments Report:

Use Case Name: Generate Completed Appointments Report

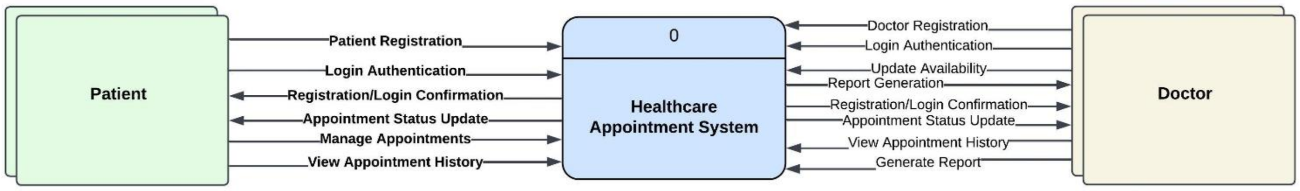
Actor: Doctor
Description: The doctor generates a PDF report of their completed appointments.
Trigger: The user wants to see past and upcoming appointments.
Normal Course: 1. The doctor logs into their dashboard. 2. The doctor clicks the **"Get Reports"** button. 3. The system fetches all completed appointments from the database. 4. The system generates a PDF report that includes: • Patient Name • Gender • Date of Birth 5. Appointment Date and Time 6. Total number of completed appointments 7. The report is saved in the designated directory.
Postconditions: The doctor wants to document or review their completed appointments.

11. View Appointment History:

Use Case Name: View Appointment History
Actor: Patient, Doctor
Description: The user views a history of their appointments.
Trigger: The user wants to see past and upcoming appointments.
Normal Course: 8. User can see "Appointment History" on their dashboard. 9. The system retrieves and displays appointment details
Postconditions: Appointment history is displayed.

Data Flow Diagrams:

Context-Level Data Flow Diagram (DFD) for Healthcare Appointment System:



The context-level Data Flow Diagram (DFD) provides an overview of the **Healthcare Appointment System**, illustrating the interaction between the primary external entities **Patients**, **Doctors,** and the central system. The system acts as the intermediary, processing and managing the flow of information between these entities.

1. Patient Interactions

- **Input to the System**:
  - **Patient Registration**: Patients provide their details to register with the system.
  - **Login Authentication**: Patients submit credentials for authentication.
  - **Manage Appointments**: Patients request actions such as booking, modifying, or canceling appointments.
- **Output from the System**:
  - **Registration/Login Confirmation**: The system confirms successful registration or login.
  - **Appointment Status Update**: Notifications regarding the status of appointments (e.g., confirmed, canceled).
  - **View Appointment History**: Patients can retrieve records of their past appointments.

2. Doctor Interactions

- **Input to the System**:

    - **Doctor Registration**: Doctors submit their details for registration.
    - **Login Authentication**: Doctors provide login credentials for access.
    - **Update Availability**: Doctors update their availability to accept appointments.
- **Output from the System**:
    - **Registration/Login Confirmation**: The system confirms successful registration or login.
    - **Appointment Status Update**: Notifications about appointment requests or changes.
    - **View Appointment History**: Doctors can access details of their completed appointments.

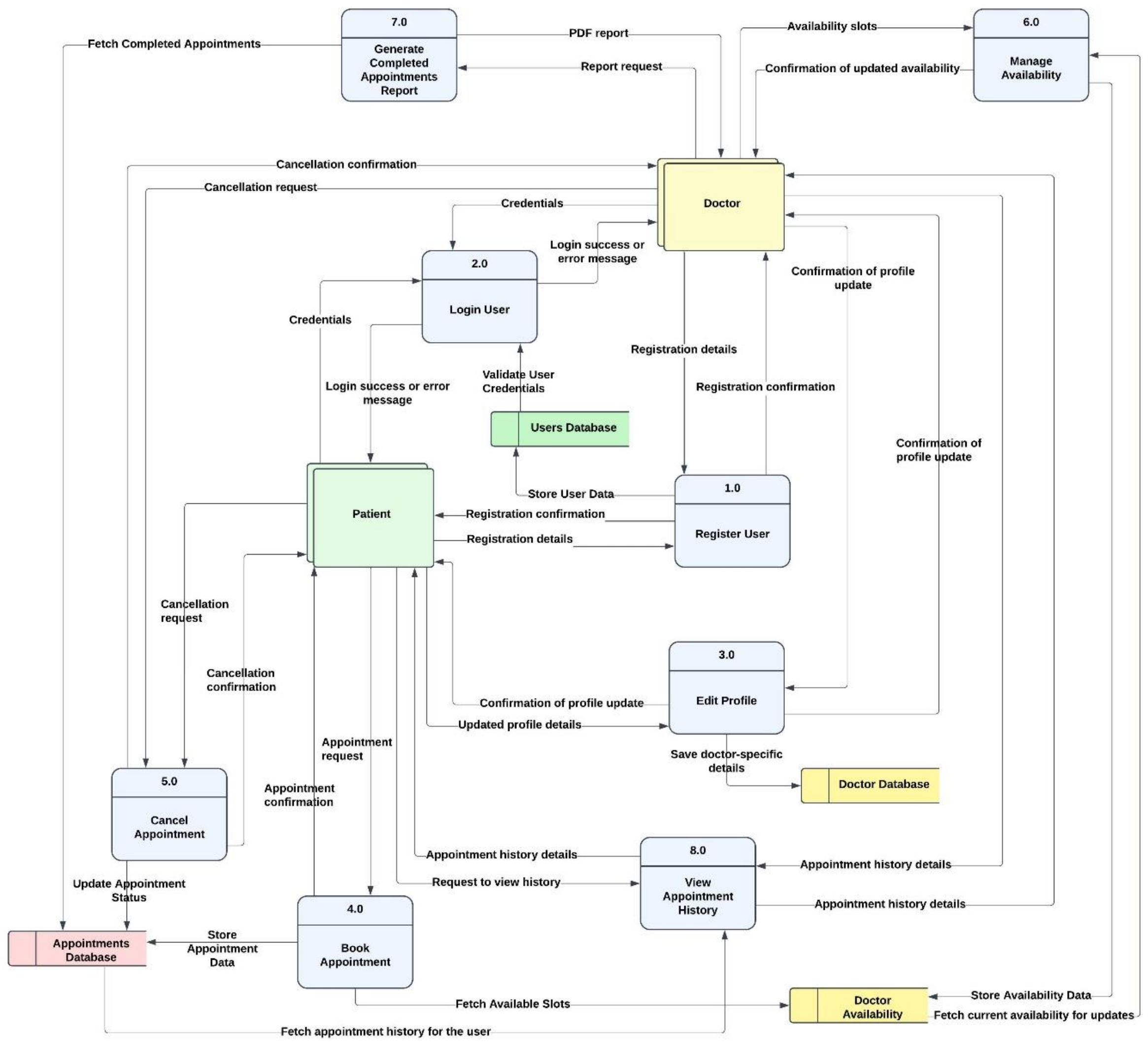
3. Healthcare Appointment System

The system is the central hub that:

- Facilitates **secure data exchange** between patients and doctors.
- Handles **registration, authentication, and appointment management** processes.
- Maintains and provides access to **appointment history** for both patients and doctors.

Level-0 Data Flow Diagram (DFD) for Healthcare Appointment System:

The **Level-0 DFD** provides a high-level view of the **Healthcare Appointment System**, illustrating the major processes, data flows, and interactions between the **Patient**, **Doctor**, and the system. This diagram represents the overall functionality of the system and highlights the data exchanges required to perform key tasks.


7.0
Generate Completed Appointments Report
6.0
Manage Availability
Fetch Completed Appointments
PDF report
Report request
Availability slots
Confirmation of updated availability
Cancellation confirmation
Cancellation request
Doctor
Credentials
Login success or error message
2.0
Login User
Confirmation of profile update
Credentials
Registration details
Validate User Credentials
Login success or error message
Registration confirmation
Users Database
Confirmation of profile update
Patient
Store User Data
1.0
Register User
Registration confirmation
Registration details
Cancellation request
Cancellation confirmation
3.0
Edit Profile
Confirmation of profile update
Updated profile details
Save doctor-specific details
Doctor Database
Appointment request
Appointment confirmation
5.0
Cancel Appointment
Appointment history details
8.0
View Appointment History
Appointment history details
Request to view history
Appointment history details
Update Appointment Status
4.0
Book Appointment
Store Appointment Data
Appointments Database
Fetch Available Slots
Doctor Availability
Store Availability Data
Fetch current availability for updates
Fetch appointment history for the user

Entities

- **Patient:** Registers, logs in, books appointments, edits profile, and views appointment history.
- **Doctor:** Registers, logs in, manages availability, views appointment history, cancels appointments, and generates reports.

Data Stores:

- **Users Database:** Stores user credentials and profile information.
- **Appointments Database:** Contains details of all booked, canceled, and completed appointments.
- **Doctor Database:** Maintains doctor-specific information such as specialization and availability.
- **Doctor Availability:** Tracks and updates the availability slots provided by doctors.

Key Processes and Data Flows:

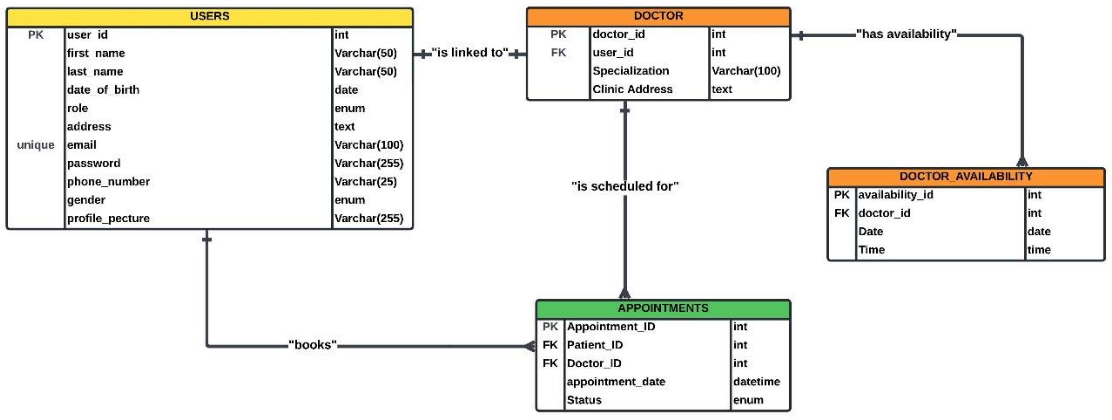
1. **Process 1.0: Register User**
    - **Input**: Registration details from Patient or Doctor.
    - **Process**: Validates the input and stores the user data in the **Users Database**.
    - **Output**: Sends a registration confirmation to the respective user.
2. **Process 2.0: Login User**
    - **Input**: User credentials from Patient or Doctor.
    - **Process**: Validates credentials against the **user database**.
    - **Output**: Sends a success or error message based on login status.
3. **Process 3.0: Edit Profile**
    - **Input**: Updated profile details from Patient or Doctor.
    - **Process**: Saves changes to the respective database
    - **Output**: Sends confirmation of the profile update.
4. **Process 4.0: Book Appointment**
    - **Input**: Appointment request details from Patient.
    - **Process**: Stores appointment data in the **Appointments Database** and fetches available slots from **Doctor Availability**.
    - **Output**: Sends an appointment confirmation to the Patient.
5. **Process 5.0: Cancel Appointment**
    - **Input**: Cancellation request from Patient or Doctor.
    - **Process**: Updates the **Appointments Database** to reflect the cancellation.
    - **Output**: Sends cancellation confirmation to the user.
6. **Process 6.0: Manage Availability**

   - **Input**: Availability slots from doctor.
   - **Process**: Updates availability data in the **Doctor Availability** database.
   - **Output**: Sends confirmation of updated availability to the Doctor.

7. **Process 7.0: Generate Completed Appointments Report**
   - **Input**: Report request from doctor.
   - **Process**: Fetches completed appointment data from the **Appointments Database**.
   - **Output**: Generates a PDF report and sends it to the Doctor.
8. **Process 8.0: View Appointment History**
   - **Input**: History request from Patient or Doctor.
   - **Process**: Fetches appointment history from the **Appointments Database**.
   - **Output**: Displays the appointment history to the user.

Entity-Relationship Diagram (ERD):

The **Entity-Relationship Diagram (ERD)** provides a high-level blueprint of the database design for the **Healthcare Appointment System**. It represents the core entities, their attributes, and relationships, enabling efficient data management and seamless integration of the system's functionalities.



Key Entities and Their Roles:

1. Users: Represents all users of the system, including both patients and doctors.

Attributes:

- user_id (Primary Key): Unique identifier for each user.
- first_name, last_name, date_of_birth: Basic personal information.
- role: Defines whether the user is a Patient or Doctor.
- email, password: Unique credentials for login and authentication.
- address, phone_number, gender, profile_picture: Additional profile details.

Purpose: Central table for storing user information.

2. Doctor: Contains information specific to doctors.

Attributes:

- doctor_id (Primary Key): Unique identifier for each doctor.
- user_id (Foreign Key referencing Users): Links doctor-specific data to the Users table.
- specialization: The area of expertise of the doctor.
- clinic_address: The clinic's location.

Purpose: Provides additional details about doctors, enabling features like specialization search.

3. Appointments: Tracks all appointments between patients and doctors.

Attributes:

- appointment_id (Primary Key): Unique identifier for each appointment.
- patient_id (Foreign Key referencing Users): Identifies the patient.
- doctor_id (Foreign Key referencing Doctor): Identifies the doctor.
- appointment_date: Scheduled date and time for the appointment.
- status: Enum field indicating the appointment's status (e.g., booked, canceled, completed).

Purpose: Central table for managing all appointment-related data.

4. Doctor Availability: Stores the availability schedule for doctors.

Attributes:

- availability_id (Primary Key): Unique identifier for each availability record.
- doctor_id (Foreign Key referencing Doctor): Links availability data to a specific doctor.

- date, time: Specifies when the doctor is available.

Purpose: Ensures appointment booking aligns with the doctor's availability.

Relationships Between Entities:

Users and Doctor:

- **Type: One-to-One**
- **Description:** Each doctor is also a user in the system, but not all users are doctors. This relationship ensures that doctor-specific data is linked to the general user information.

Users and Appointments:

- **Type: One-to-Many**
- **Description:** A single patient (user) can book multiple appointments in the system. Each appointment, however, is linked to only one patient.

Doctor and Appointments:

- **Type: One-to-Many**
- **Description:** A doctor can have multiple appointments scheduled with different patients. Each appointment references a specific doctor.

Doctor and Doctor Availability:

- **Type: One-to-Many**
- **Description:** Each doctor can have multiple availability slots, allowing patients to book appointments based on the doctor’s available times.