Requirements Specification Document

Group Name: Gofies

Date: November 17, 2024

Course Information

BLG 411E - Software Engineering

Project Title

Gofies Hospital Management System

Team Members

- Abdullah Shamout - Project Manager, DevOps Team Leader, AI Algorithms Engineer
- Berkay Gürsu - Frontend Team Leader, Backend Developer
- İbrahim Halil Marsil - Backend Team Leader, Frontend Developer
- Mustafa Can Çalışkan - Database Team Leader, Testing Developer
- Rasim Berke Turan - Backend Developer, Frontend Developer, Client Representative
- Sarper Öztorun - Frontend Developer, Testing Developer, Meeting Organizer
- Yusuf Emir Sezgin - UI/UX Team Leader, Frontend Developer
- Yusuf Şahin - Testing Team Leader, AI Algorithms Engineer

Contents

Change Log Table

Version	Date	Changes
v1.0	Nov 10, 2024	Initial draft
v1.1	Nov 17, 2024	User interface models and flow diagrams have been added.

1 Introduction

This document outlines the requirements for the Gofies Hospital Management System (GHMS) project. The goal is to meet a comprehensive set of functional and non-functional requirements set by the client and our development team, user scenarios, and use cases that guide the development of the hospital management system. The system aims to improve hospital workflows by providing an integrated platform for patient management, doctor services, AI-based expert systems, and administrative functions, all while ensuring high security, scalability, and real-time data synchronization.

2 System Requirements

2.1 Functional Requirements

2.1.1 Backend System:

- The system will use a monolithic architecture.
- It will provide a RESTful API for communication between the frontend and backend.
- Secure user access will be managed through JWT-based authorization.
- An AI-powered chatbot service will be integrated for user interaction.
- API rate limiting will be implemented to prevent system overload.

2.1.2 Database System:

- Role-based Access Control (RBAC) will manage user roles and permissions.
- Data will be encrypted both at rest and during transit to ensure security.
- The database will be designed for scalability and support horizontal partitioning for efficient data management.
- An audit log mechanism will be included to track and record user actions.
- Optimized queries.

2.1.3 Frontend System:

- The frontend will follow a modular, component-based architecture.
- State management will be implemented to handle data consistently across the application.
- The UI will receive real-time updates to stay in sync with backend changes.
- Lazy loading and code splitting will be used for optimal application performance.
- Error handling will be in place to display appropriate error messages when needed.
- UI animations and feedback will enhance the user experience.

2.1.4 UI/UX:

- The interface will support various themes, including night mode and a mode for color blindness.
- Multilingual support will be provided for accessibility to a global audience.
- The UI will be responsive and accessible across various devices, ensuring consistency in experience.

2.1.5 AI System:

- The system will integrate Large Language Models (LLMs) to enable chatbot capabilities.
- Specialized AI models will be included to support medical decision-making processes.
- A Doctor-Imitating Expert System will be developed to assist with medical consultations.

2.1.6 DevOps:

- A Continuous Integration and Continuous Deployment (CI/CD) pipeline will automate testing and deployment processes.
- Docker container orchestration will be used for scalability and ease of management.
- The system will be deployed on the cloud to ensure high availability and scalability.

2.1.7 Testing:

- All tests will be up-to-date.
- Testing scripts including all the edge cases will be provided.

2.2 Non-Functional Requirements

Performance: The system must handle high traffic and user load balance efficiently, with no performance degradation.

Scalability: The system must be able to scale horizontally to accommodate a growing number of users and increasing data.

Security: End-to-end security will be ensured through comprehensive encryption, authentication, and authorization mechanisms.

Usability: The interface should be intuitive, user-friendly, and accessible to ensure ease of use for all users.

Maintainability: The system should be designed for ease of maintenance, making it straightforward to update, scale, and troubleshoot.

3 Use Cases

3.1 User Types

IT Staff: Manages user accounts, monitors system activities, maintains hospital IT infrastructure, ensures data security, provides technical support, and oversees administrative tasks such as clinic, doctor, and patient management.

Doctor: Diagnoses patients, prescribes treatments, schedules and reviews lab tests, and manages patient treatment plans.

Lab Staff: Conducts medical tests, records and uploads test results, and communicates test outcomes to doctors or patients.

Patient: Accesses personal medical data, schedules appointments, views lab results, and communicates with doctors.

3.2 User Scenarios

3.2.1 IT Staff

Name: John Doe
Role: System Admin
Scenario: John logs into the system to oversee the hospital's IT operations. He reviews recent system activities to ensure everything is functioning securely and efficiently. He manages user accounts by creating new profiles, assigning roles, and updating permissions for existing users. To maintain security, John checks the activity logs for any unusual actions, such as multiple failed login attempts or unauthorized data access. He also performs system maintenance tasks, ensuring that all software and security measures are up-to-date.

3.2.2 Doctor

Name: Dr. Sarah Lee
Role: Doctor
Scenario: Dr. Lee begins her day by logging into the system to review her appointments and patient information. She examines each patient's medical history, including previous diagnoses and treatments. During consultations, she updates patient records with new symptoms, diagnoses, and prescribed treatments. She uses ChatBot to gather required information about the patient's background. If lab tests are required, she submits a request and ensures that urgent cases are prioritized. After reviewing the test results, she adjusts treatment plans accordingly and schedules follow-up appointments as needed.

3.2.3 Lab Staff

Name: Emily Clark
Role: Lab Staff
Scenario: Emily logs into the system to manage her workload for the day. She reviews pending test requests, prioritizes urgent cases, and conducts the necessary tests. Once completed, she uploads the test results and ensures that doctors are notified. In critical cases, she directly communicates with doctors to expedite test reviews. Emily also monitors the availability of testing supplies and reports any shortages to the admin for replenishment.

3.2.4 Patient

Name: Michael Smith
Role: Patient
Scenario: Michael logs into the hospital's patient portal to check his medical records. He reviews recent lab test results and compares them to past results to track his health progress. Concerned about some findings, he sends a secure message to his doctor for clarification. He uses ChatBot to get informed before seeing a doctor and have an idea about his situation. Michael also schedules a follow-up appointment and updates his personal information, such as contact details and insurance information, to ensure he receives timely notifications.

3.3 Use Case Diagram

This diagram shows the high-level interaction between users and system functionalities.

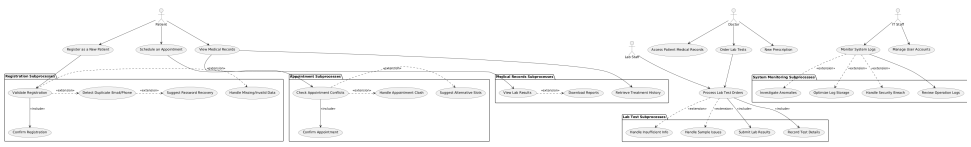

Figure 1: User Case Diagram

3.4 Use Cases

3.4.1 Register as a New Patient

Actors: Patient
Goal: To register in the hospital system to access medical services.
Preconditions:

- The patient has access to the hospital's registration portal.
- The patient is not already registered in the system.

Postconditions:

- A new patient record is created in the system.
- The patient receives confirmation of successful registration.

Main Success Scenario:

1. **Access Registration Portal:** The patient navigates to the hospital's registration page.
2. **Fill Registration Form:** The system displays a registration form requesting personal information: Name, surname, Gender, Phone, Email, Date of birth, Address.
3. **Submit Form:** The patient submits the completed form.
4. **Validate Information:** The system validates the entered data for correctness and completeness.
5. **Confirm Registration:** The system sends a confirmation email to the patient.

Extensions:

- **3a. Missing or Invalid Data:**
    1. The system highlights missing or invalid fields.
    2. The patient corrects the information and resubmits.
- **4a. Duplicate Email or Phone:**
    1. The system detects existing Actor with the same email.
    2. The system notifies the patient and suggests password recovery or contact support.

3.4.2 Schedule an Appointment

Actors: Patient
Goal: To schedule an appointment with a doctor.
Preconditions:

- The patient is registered and logged into the system.

Postconditions:

- The patient and doctor receive appointment confirmation.

Main Success Scenario:

1. **Access Appointment Scheduling:** The patient selects "Schedule Appointment" from the dashboard.
2. **Select Specialty:** The patient chooses a medical specialty or condition.
3. **View Available Doctors:** The system displays a list of doctors with the selected specialization.
4. **Choose Doctor:** The patient selects a preferred doctor.
5. **Select Date and Time:** The system shows the doctor's available time slots.

6. **Confirm Appointment Details:** The patient selects a suitable date and time.
7. **Appointment Confirmation:**
   (a) The system checks for any conflicts or double bookings.
   (b) Receive Confirmation: The system sends an appointment confirmation to the patient via email.
8. **Set Calendar:** The appointment is added to both patient and doctor calendars.

Extensions:

- **5a. No Available Slots:**
  1. The system suggests alternative doctors or dates.
  2. The patient selects a different option or modifies search criteria.
- **6a. Appointment Clash:**
  1. The selected slot is no longer available.
  2. The system informs the patient and returns to available slots.

3.4.3 View Medical Records

Actors: Patient
Goal: To view personal medical history, treatment plans, lab test results, and analysis.
Preconditions:

- The patient is logged into their account.
- The patient has existing medical records in the system.

Postconditions:

- The patient views up-to-date medical information.

Main Success Scenario:

1. **Navigate to Medical Records:** The patient selects "Medical Records" from the dashboard.
2. **Display Records:** The system retrieves and displays:
   - Treatment history
   - Medications
   - Lab test results
   - Appointment history
3. **View Lab Results:**
   (a) The patient selects a specific lab test.
   (b) Download Reports: The patient downloads lab reports or treatment summaries as needed.

Extensions:

- **2a. No Records Found:**
  1. The system informs the patient that no records are available.
  2. Suggests contacting the hospital for assistance.

3.4.4 Access Patient Medical Records

Actors: Doctor
Goal: To view a patient's medical history for informed diagnosis and treatment.
Preconditions:

- The doctor is logged into the system with appropriate permissions.
- The doctor is assigned to the patient or has authorised access.

Postconditions:

- The doctor reviews the patient's medical history.

Main Success Scenario:

1. **Search for Patient:** The doctor enters the patient's name.
2. **Display Patient Summary:** The system shows a summary of the patient's information.
3. **Access Detailed Records:** The doctor selects specific sections:
    - Disease history
    - Previous treatments
    - Lab results
    - Medications

3.4.5 Order Lab Tests

Actors: Doctor
Goal: To request lab tests for a patient as part of diagnosis.
Preconditions:

- The doctor is logged in and viewing a patient's treatment plan.

Postconditions:

- The system notifies lab staff of the new test order.

Main Success Scenario:

1. **Initiate Lab Test Order:** The doctor selects "Order Lab Test" within the patient's treatment plan.
2. **Select Test Type:** The system displays available lab test types.
3. **Specify Details:** The doctor selects the test(s) and provides any additional instructions.
4. **Submit Order:** The doctor confirms and submits the order.
5. **Notification:** The system notifies the relevant lab staff of the new test order.

Extensions:

- **6a. Assignment Failure:**
    1. No lab staff available for assignment.
    2. The system queues the test and notifies lab management.

3.4.6 New Prescription

Actors: Doctor
Goal: To prescribe medication as part of a patient's treatment plan.
Preconditions:

- The doctor is logged in.

Postconditions:

- The system notifies the patient of the new prescription.

Main Success Scenario:

1. **Access Medication Module:** The doctor selects "New Prescription" within the patient management.
2. **Search Medication Database:** The system provides a searchable list of medications.
3. **Define Dosage and Schedule:**
    - Enter dosage information.
    - Set frequency (e.g., twice daily).
    - Specify start and end dates.
4. **Confirm Prescription:** The doctor reviews and submits the prescription.
5. **Notify Patient:** The system sends prescription details to the patient.

3.4.7 Process Lab Test Orders

Actors: Lab Staff
Goal: To perform ordered lab tests and record results.
Preconditions:

- Lab staff is logged into the system.

Postconditions:

- Lab test results are recorded.

Main Success Scenario:

1. **View Pending Lab Tests:** Lab staff selects "Pending Lab Tests" from the dashboard.
2. **Access Test Details:** The system displays test information:
    - Patient details
    - Test type and instructions
3. **Record Results:**
    - Upload any necessary files.
    - Notify Doctor: The system notifies the ordering doctor and patient that results are available.

Extensions:

- **2a. Insufficient Information:**
    1. Lab staff finds missing information.
    2. Contacts the doctor or returns the test for clarification.
- **4a. Sample Issues:**
    1. Sample is contaminated or insufficient.
    2. Lab staff marks the test as failed and requests a new sample.

3.4.8 Manage User Accounts

Actors: IT Staff
Goal: To create, modify, and deactivate user accounts for system users.
Preconditions:

- IT staff member is logged in with administrative rights.

Postconditions:

- Accounts are updated.

Main Success Scenario:

1. **Navigate to User Management:** IT staff selects "Staff Accounts" from the admin menu.
2. **Create New Account:**
   (a) Clicks "Create New Account".
   (b) Enters staff details.
   (c) Submits the form.
3. **Modify Existing Account:**
   (a) Searches for a staff account.
   (b) Updates details.
   (c) Saves changes.
4. **Deactivate Account:**
   (a) Selects a staff account.
   (b) Changes account status to inactive.
   (c) Confirms deactivation.
5. **Notify User:** System sends an email to the staff about the account changes.

Extensions:

- **2a. Staff Account Exists:**
  1. The system warns the IT staff that the related account already exists.
- **4a. Dependency Check:**
  1. Deactivating an account linked to ongoing treatments.
  2. System warns of potential impact before proceeding.

3.4.9 Monitor System Logs

Actors: IT Staff
Goal: To review operation logs for security and auditing purposes.
Preconditions:

- IT staff member has access to monitor.

Postconditions:

- Potential issues are identified and addressed.

Main Success Scenario:

1. **Access Operation Logs:** IT staff selects "System Logs" from the admin panel.
2. **Filter Logs:**
   (a) IT staff can apply search and filter operations on the logs.

3. **Identify Anomalies:**
   (a) Looks for failed operations attempts.
4. **Take Action:**
   (a) Investigates issues.
   (b) Updates security protocols if necessary.
   (c) Documents findings.

Extensions:

- **3a. Log Data Overload:**
  1. System performance is affected by log size.
  2. IT staff archives old logs and optimizes log storage.
- **4a. Security Breach Detected:**
  1. Unusual activity is found.
  2. IT staff initiates security response procedures.

4 User Interface Model

4.1 Login

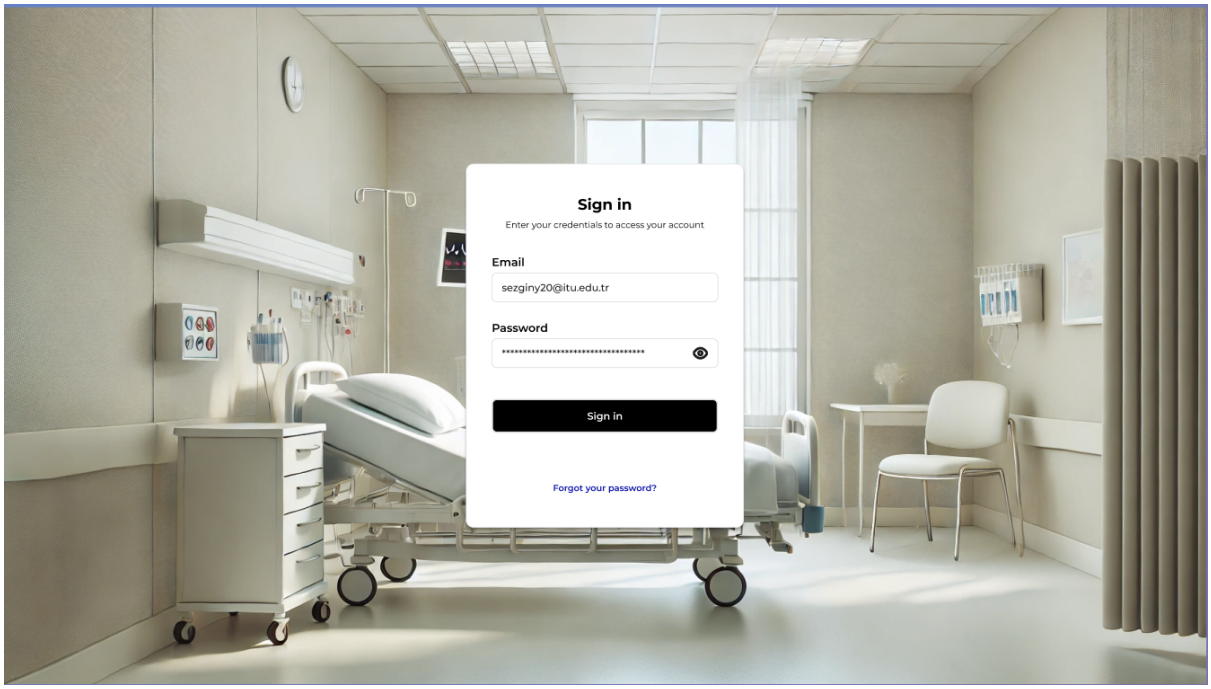

Figure 2: Login Page for All Users

4.2 Patient Homepage

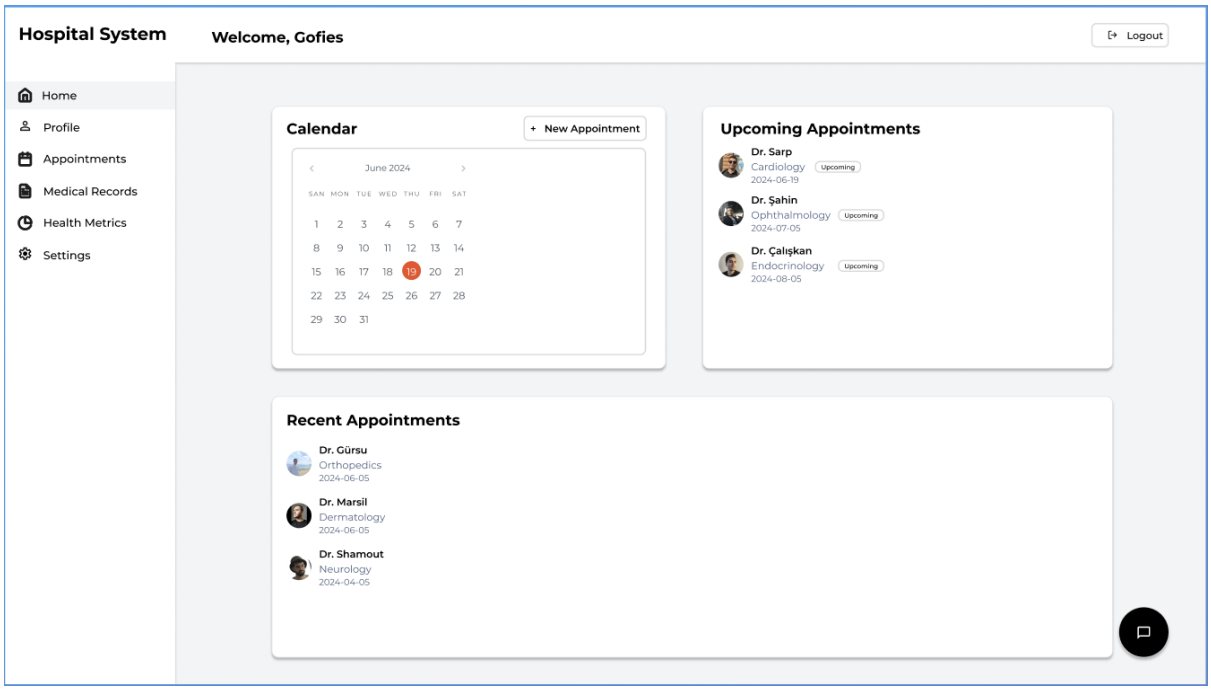

Figure 3: Patient Homepage Design (Scenario 3.4.2, 3.4.3)

4.3 Patient Profile

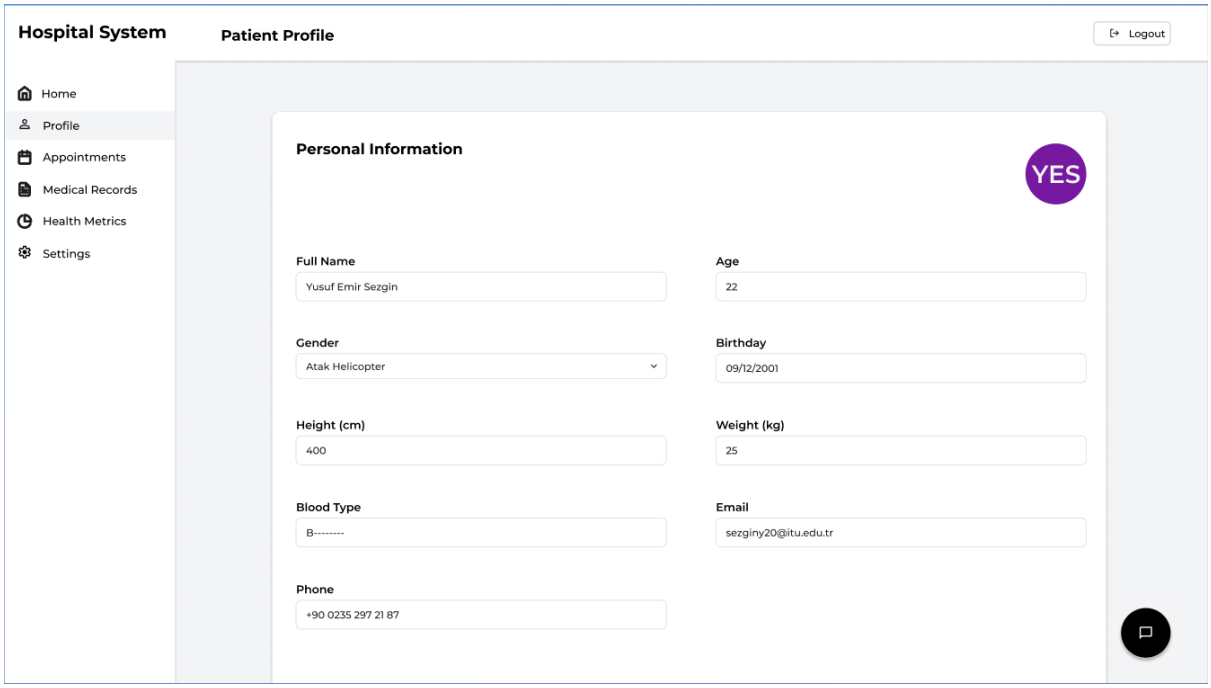

Figure 4: Patient Profile

4.4 Doctor Homepage

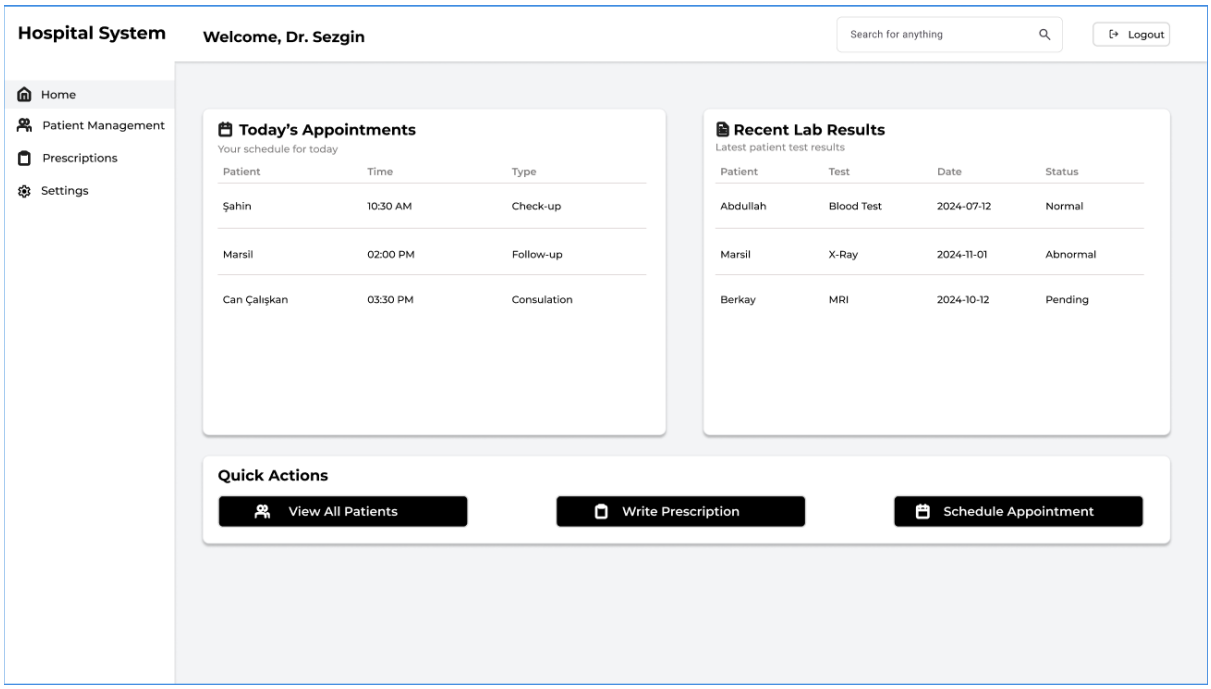

Figure 5: Doctor Homepage Design (Scenario: 3.4.4, 3.4.5, 3.4.6)

4.4.1 Patient Management

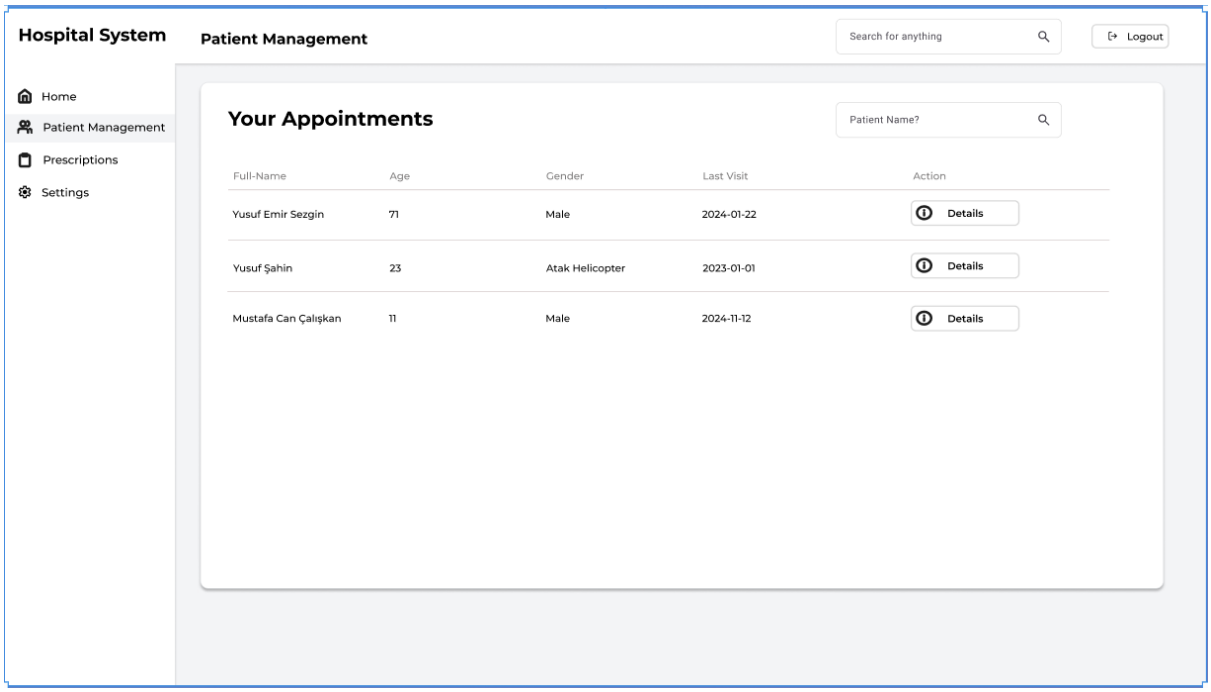

Figure 6: Patient Management for Doctors

4.4.2 Patient Details

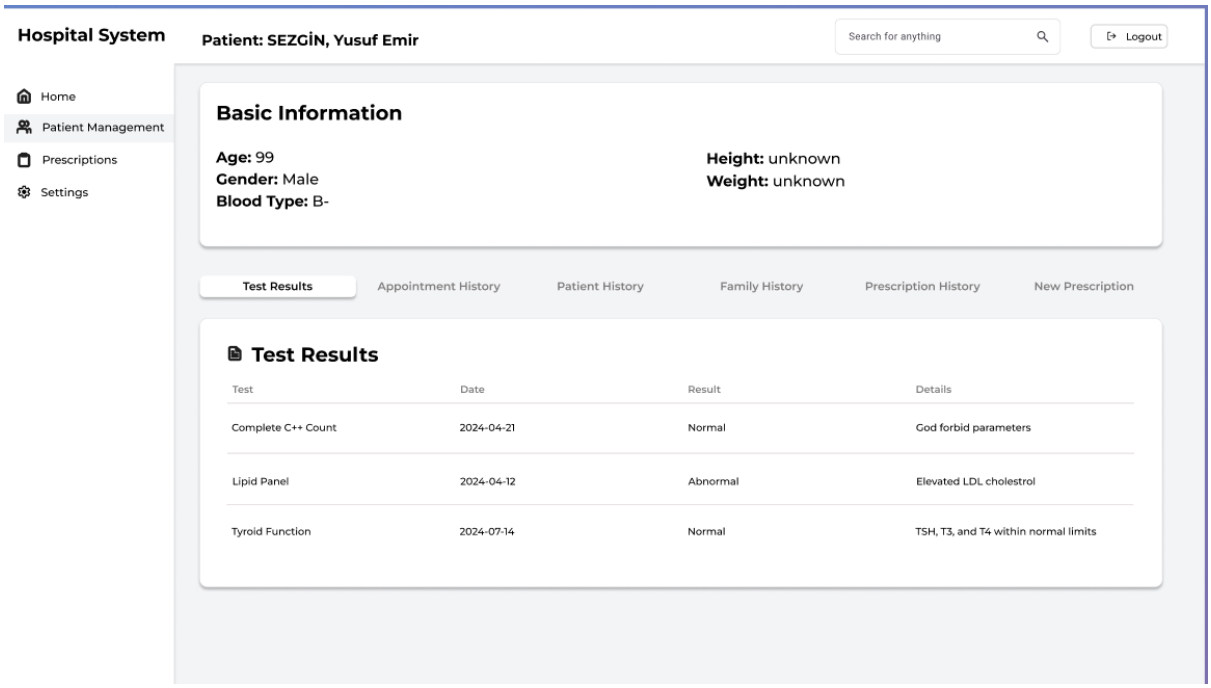

Figure 7: Patient Details for Doctors

4.5 Lab Staff Homepage

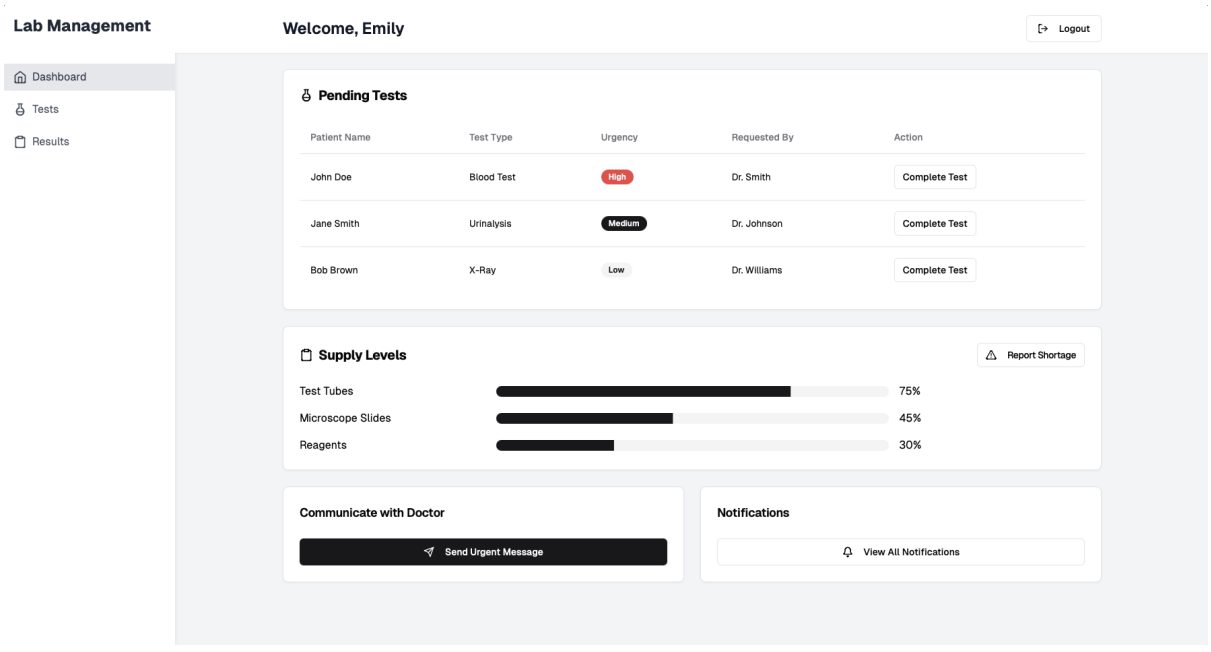

Figure 8: Lab Staff Homepage Design (Scenario: 3.4.7)

4.6 Admin - IT Homepage

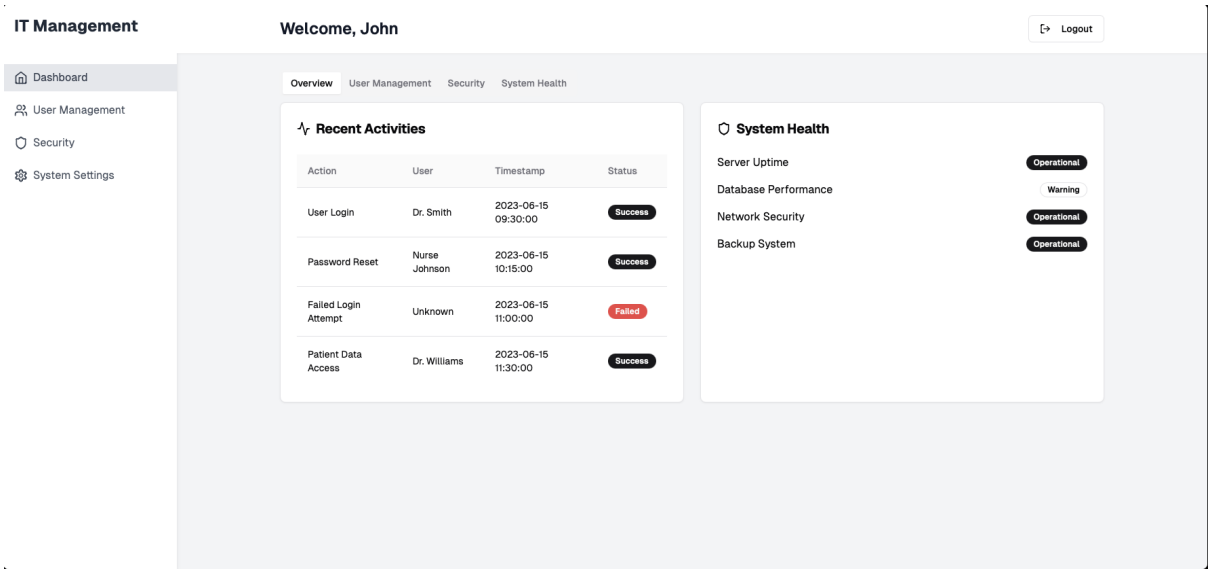

Figure 9: IT Homepage for Admins (Scenario: 3.4.8, 3.4.9)

4.6.1 User Management

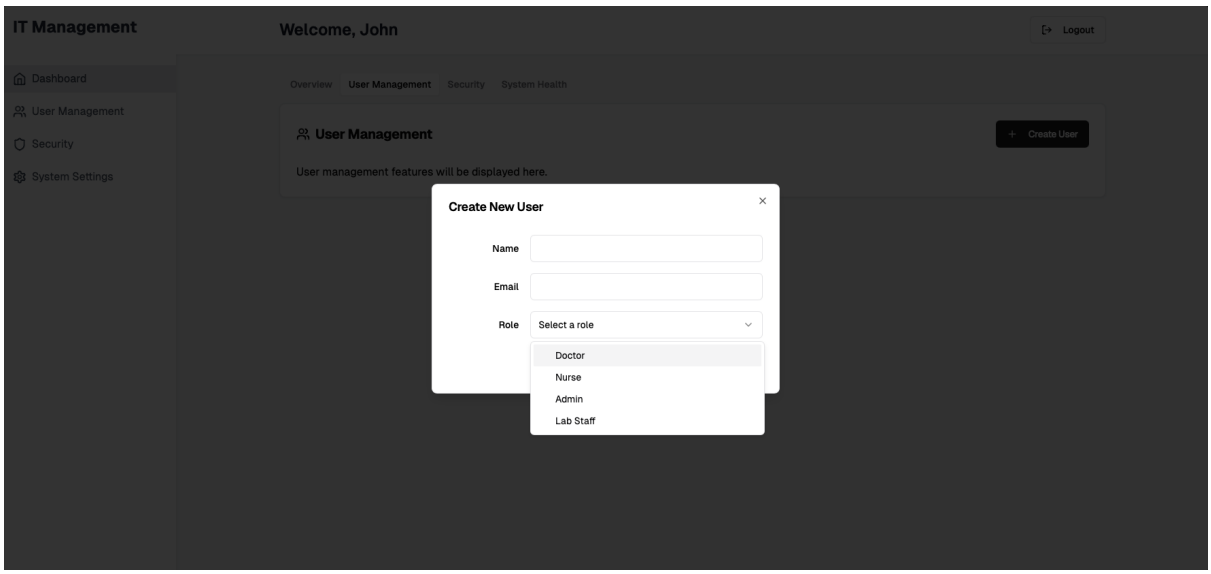

Figure 10: User Management for the GHMS

4.7 Settings

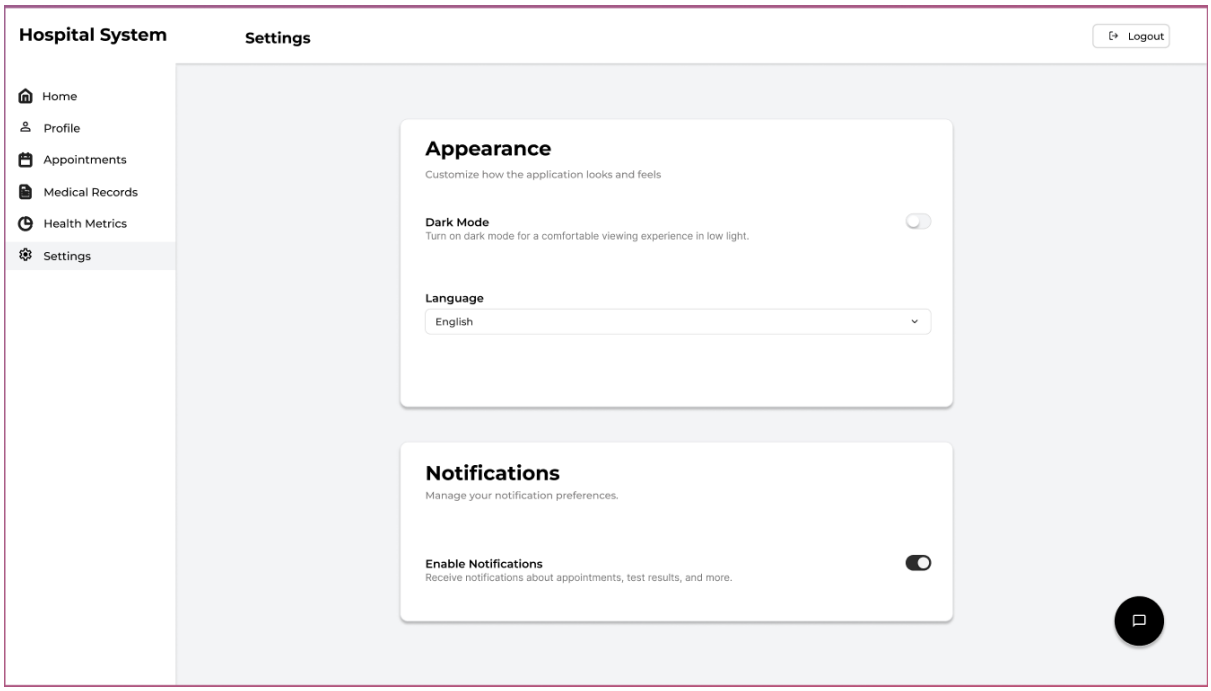

Figure 11: Settings (It will be variable for user type)

5 Flow Diagrams

5.1 General Data Model

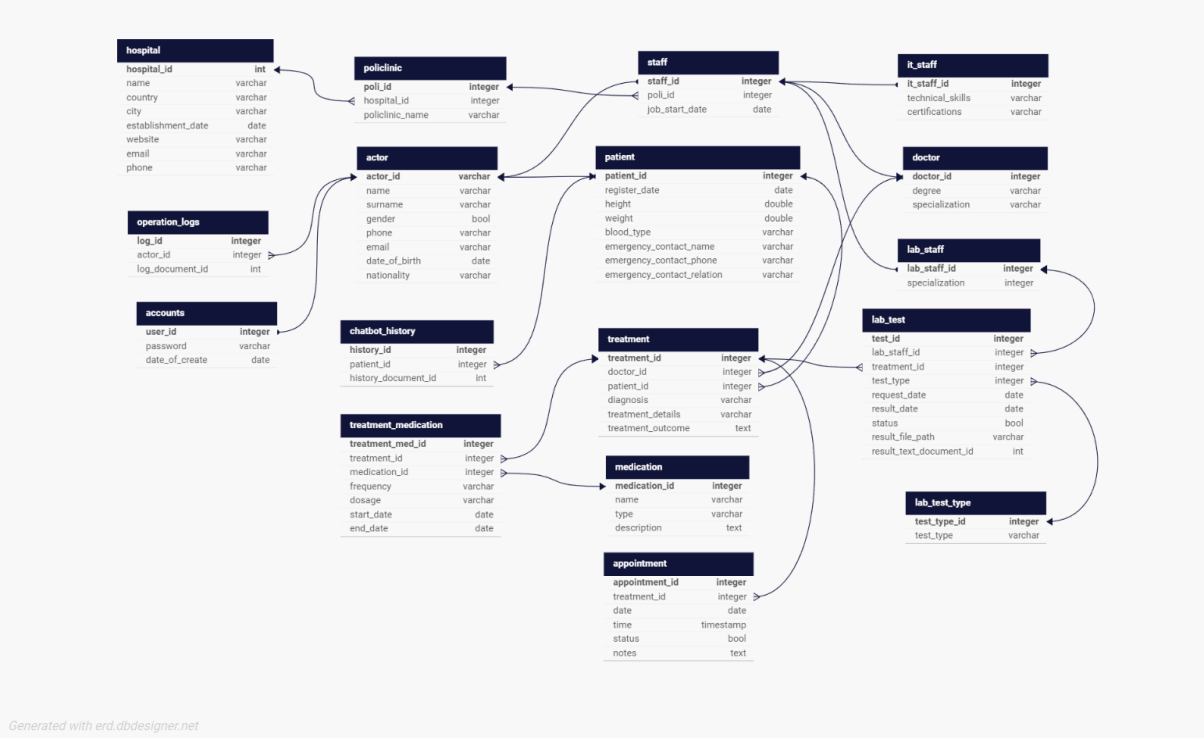

Figure 12: Entity Relationship Diagram (ERD)

5.2 Important Data Considerations

Due to its scalability and flexibility, we will store our history and log data in a NoSQL database using the JSON data format. The document IDs of data stored in JSON format will also be saved in the appropriate database fields for efficient referencing and organization. Additionally, files such as test results will be stored directly on the disk, with their file paths saved in the relevant fields within the database.

5.3 Data Flow Diagram

5.3.1 Level 0 (Context Diagram)

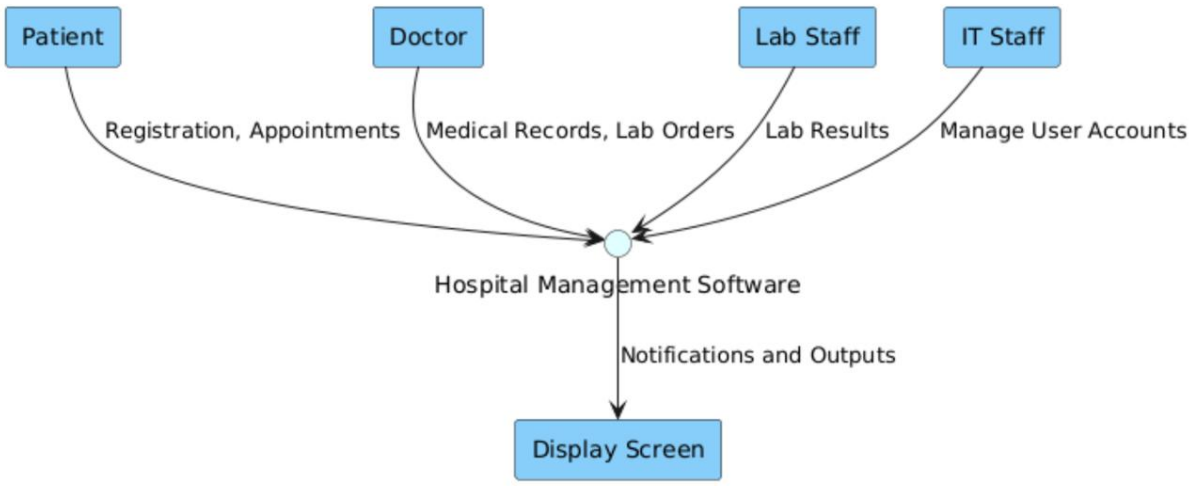

Figure 13: Level 0 Data Flow Diagram

Purpose: The context diagram gives an overview of the GHMS as a whole. It treats the system as a "black box" interacting with external entities.

Key Features:

1. **Entities:** Patients, doctors, lab staff, IT staff, and the display screen.
2. **System:**
   (a) Represented as a single process labeled "Hospital Management Software."
3. **Interactions:**
   (a) Patients registering and managing appointments.
   (b) Doctors accessing medical records and lab orders.
   (c) Lab staff uploading test results.
   (d) IT staff managing user accounts and system monitoring.
4. **Outputs:**
   (a) Notifications sent to the display for various stakeholders.

5.3.2 Level 1 (Expanded Process Overview)

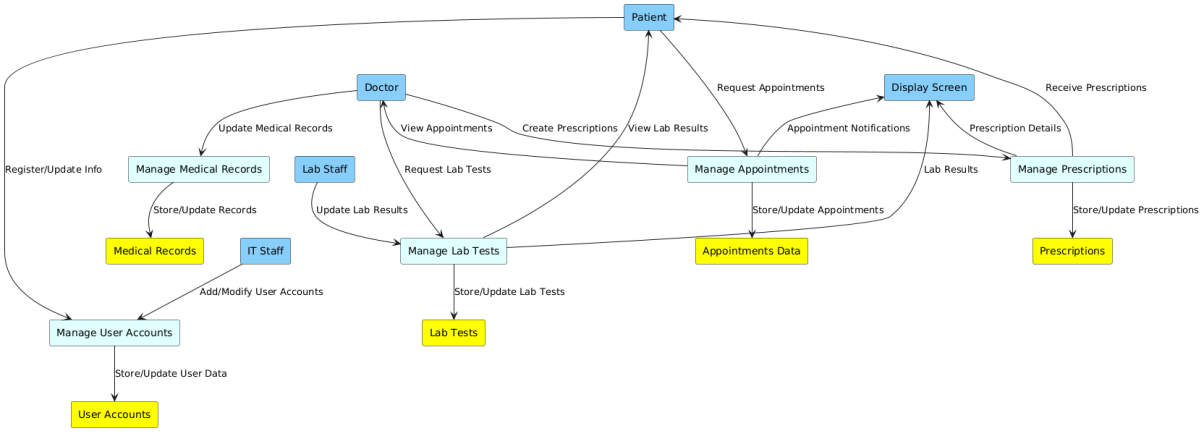

Figure 14: Level 1 Data Flow Diagram

Purpose: Decomposes the main system into major sub-processes that correspond to key functionalities.
Key Features:

1. **Data Stores:** User Accounts, Appointments, Medical Records, Lab Tests, and Prescriptions.
2. **Processes:**
   (a) "Manage User Accounts" (e.g., for IT staff).
   (b) "Manage Appointments" (e.g., for patients and doctors).
   (c) "Manage Medical Records" (e.g., for storing patient history).
   (d) "Manage Lab Tests" (e.g., for doctors and lab staff interactions).
   (e) Manage Prescriptions" (e.g., for doctors).
3. **Data Flows:** Tracks how entities interact with these processes and data stores:
   (a) Patients updating personal info and viewing prescriptions or lab results.
   (b) Doctors updating patient records and managing lab tests or prescriptions.
   (c) Lab staff updating test results.
   (d) IT staff managing user accounts and monitoring logs.

5.3.3 Level 2 (Detailed Process Flows)

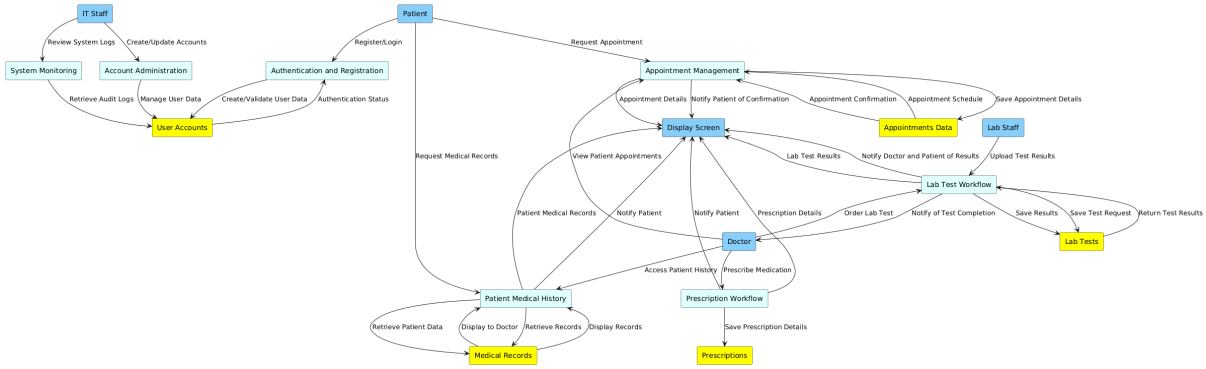

Figure 15: Level 2 Data Flow Diagram

Purpose: Provides a detailed breakdown of individual processes from Level 1, showing step-by-step interactions and flows.
Key Features:

1. **Processes:**
   (a) "Authentication and Registration": Handles login and user creation.
   (b) "Appointment Management": Manages requests, scheduling, and notifications.
   (c) "Patient Medical History": Retrieves and displays medical data.
   (d) "Lab Test Workflow": Handles lab test orders, results, and notifications.
   (e) "Prescription Workflow": Manages medication prescriptions.
   (f) "Account Administration" and "System Monitoring" for IT staff.
2. **Interactions:** Includes granular flows such as:
   (a) Registration validation and confirmation.
   (b) Appointment confirmation notifications.
   (c) Lab test orders flowing to lab staff, with results back to doctors and patients.
   (d) Prescriptions being saved and notified to patients.