Design Specification

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

1 Introduction

1.1 Goal

The goal of this design specification document is to provide a detailed framework for the development of the Gofies Hospital Management System (GHMS), ensuring it meets functional, scalability, security, and user experience requirements. It aims to define the system architecture, including components, inputs, outputs, and services, while ensuring seamless integration, data privacy, and maintainability. By incorporating comprehensive low-level designs such as class diagrams, sequence diagrams, and updated data flow diagrams, this document establishes a clear roadmap for implementing a robust, user-friendly, and efficient healthcare management platform.

2 System Architecture

2.1 System Architecture

The architecture of this project is designed to ensure a robust, scalable, and maintainable system that seamlessly integrates various services and components. It leverages containerization through **Docker Compose** to orchestrate deployment, manage configurations, and maintain health checks across the stack. The system is composed of the following components:

2.1.1 GitHub Repository

- Serves as the central hub for version control, holding the source code, configurations, and CI/CD workflows for deployment automation.

2.1.2 Orchestration Layer

- **Docker Compose** manages the lifecycle of all services, ensuring they are correctly deployed, networked, and monitored.
- Health checks are integrated at each level to ensure system reliability.
- Network module to insure seamless communication between different service containers while maintaining a layer of security from outside requests by blocking container ports

2.1.3 Frontend Service

- A user interface (UI) implemented to provide end-user functionality.
- **Health Check:** Verifies the responsiveness of the UI and the integrity of route mappings.

2.1.4 Nginx Proxy/server

- Acts as a reverse proxy, a load balancer, data encrypter, and data compressor. It routes requests to appropriate endpoints.
- **Health Check:** Ensures the server is up and functionalities are intact.

2.1.5 Backend API

- The core API responsible for processing business logic and communicating with databases.
- **Health Check:** Monitors API responsiveness and verifies endpoint functionality.

2.1.6 Database Layers

- **PostgreSQL (SQL Database):** Handles structured data and supports the Backend API with efficient queries.
- **MongoDB (NoSQL Database):** Stores unstructured data, enhancing system flexibility.
- **Health Checks:** Test database connections and ensure data accessibility.

2.1.7 AI Services

- **AI Backend API:** Facilitates communication with the Large language models.
- **AI Service:** runs medical expert systems for interaction with the users.
- **Health Checks:**
  - For the AI Backend API: Verify endpoints and response functionality.
  - For the AI Service: Ensure model loading and acceptable response times.

2.1.8 Communication and Data Flow

- The **Frontend Service** interacts with the user and displays available backend endpoint requests which if activated are routed via the **Nginx Proxy** to the **Backend API**.
- Backend services query structured and unstructured data through PostgreSQL and MongoDB.
- The AI Backend API invokes the AI Service for medical expert model interaction.

2.1.9 Continuous Integration and Continuous Deployment (CI/CD) Pipelines

To streamline the development and deployment process, a robust CI/CD pipeline is integrated into the architecture. This ensures efficient and reliable code delivery while maintaining high software quality. Key aspects of the CI/CD pipeline include:

- **Automated Testing:** Every code change triggers a suite of automated unit and integration tests, ensuring that new features do not introduce regressions or bugs.
- **Containerized Builds:** Docker images are automatically built and tagged during the CI process, ensuring consistency across all environments.
- **Version Control Integration:** The pipeline is integrated with a Git-based version control system, enabling developers to collaborate seamlessly and automatically trigger builds for every commit.
- **Staging Environment:** A staging environment is automatically provisioned to allow thorough validation of features and bug fixes before deployment to production.
- **Rolling Deployments:** The pipeline employs rolling deployments to minimize downtime and ensure that new versions of services are deployed incrementally, preserving availability.

2.1.10 Key Features of the Architecture

- **Modularity:** Components are containerized and loosely coupled to enable independent scaling and updates.
- **Health Monitoring:** Each service integrates Docker health checks to proactively detect and resolve issues.
- **Scalability:** Docker Compose ensures that additional instances of services can be deployed seamlessly.
- **Fault Tolerance:** The architecture is designed to minimize downtime and maintain service continuity.

The system is deployable on any machine since it utilizes docker containers. For our cloud provider, we are utilizing Google Cloud. Where any pushes/merges to the main branch of our repository would trigger the CD pipeline that will update the code on our virtual machine (VM) thus keeping our system up to date.

System Architecture Diagram is presented in Figure 1.

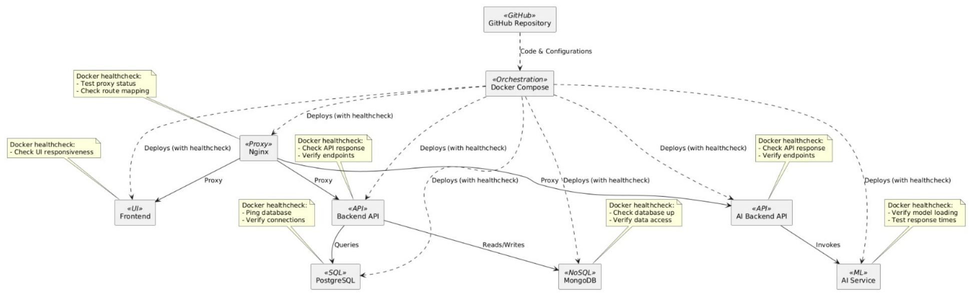


Figure 1: System Architecture Diagram

2.2 Component Diagram

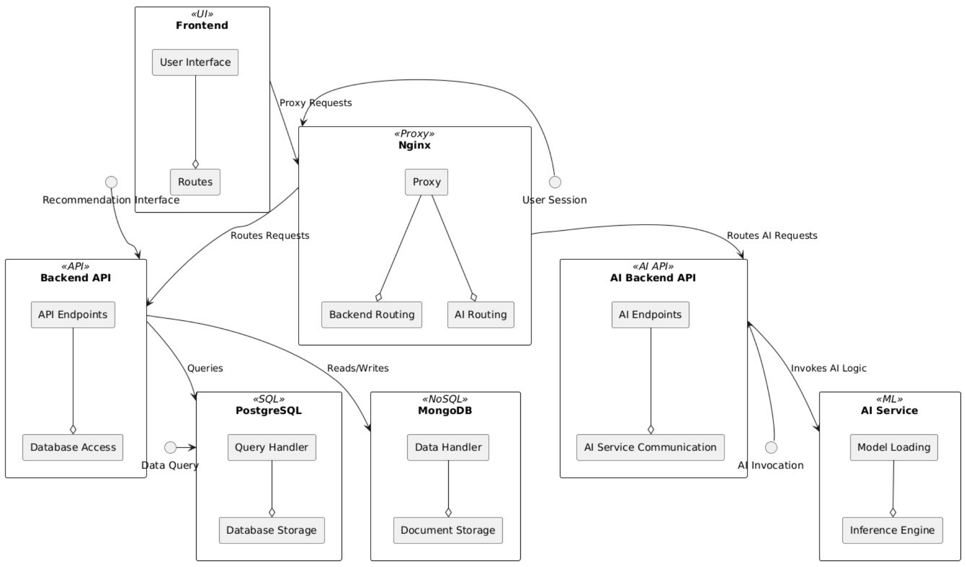


Figure 2: Component Diagram 1

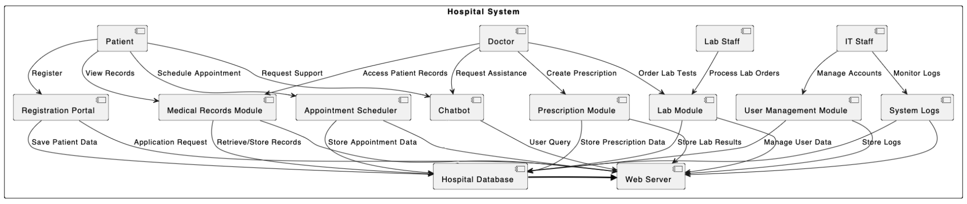


Figure 3: Component Diagram 2

3 Low Level Design

3.1 Class Diagrams

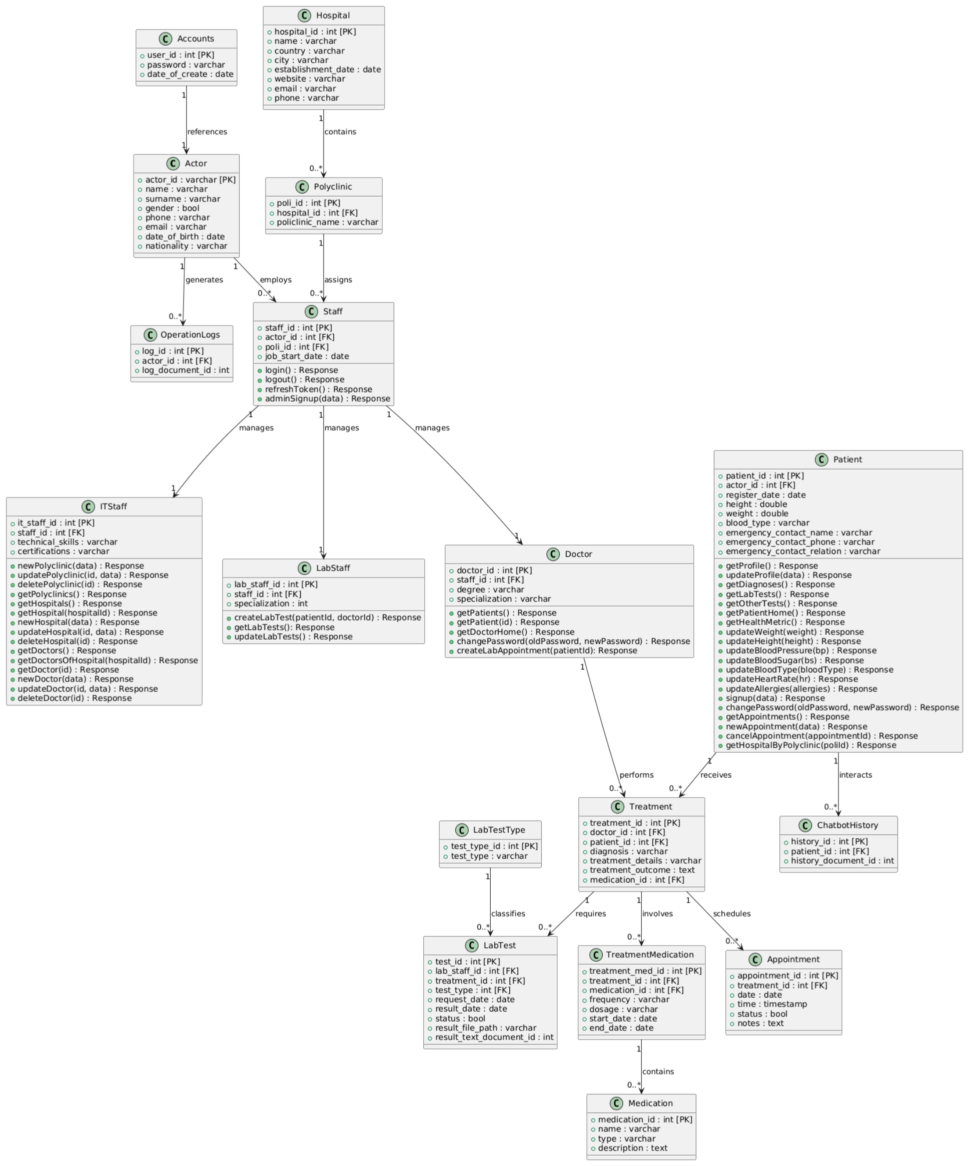


Figure 4: Class Diagram

3.2 Sequence Diagrams

Below are the abbreviations used in the sequence diagrams:

- **PMP:** Patient Management Panel
- **MRD:** Medical Records Database
- **LDB:** Lab Tests Database
- **NS:** Notification System

3.2.1 Sequence Diagram 1: IT Staff Account Management

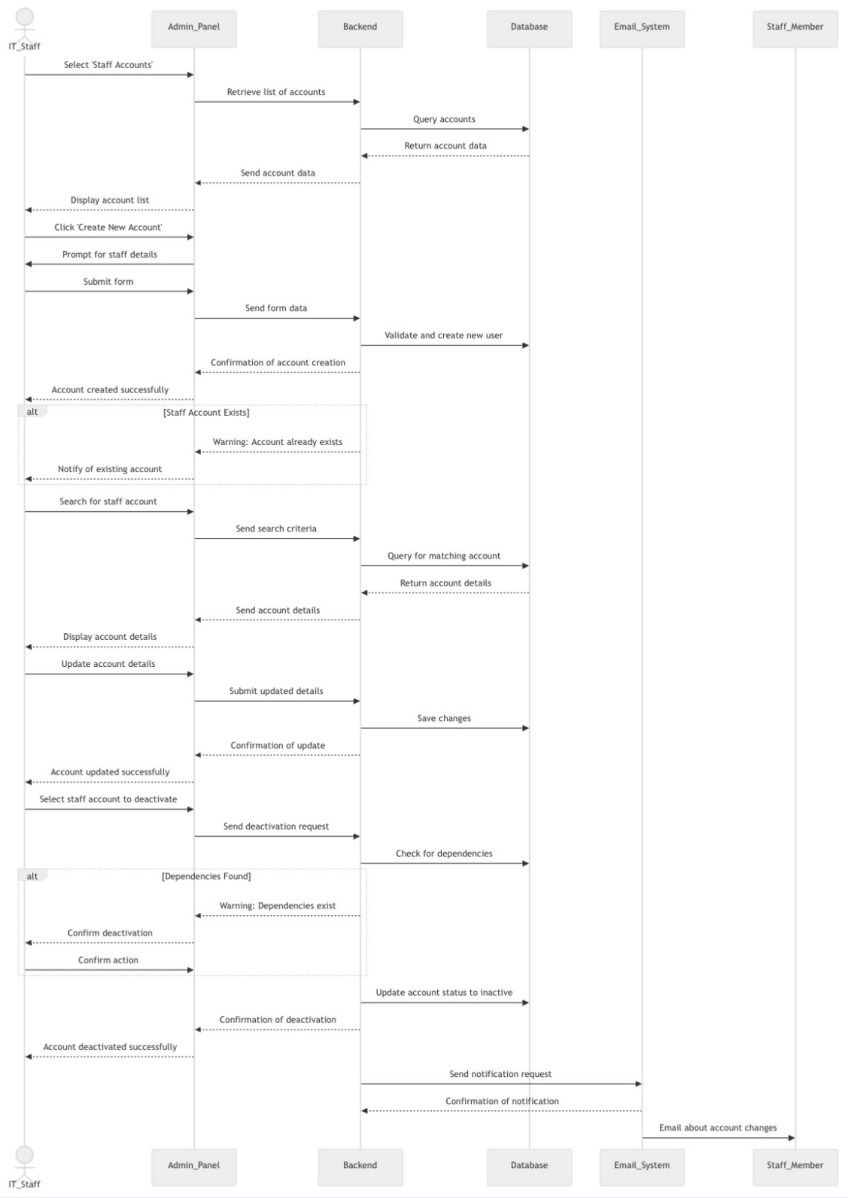

IT_Staff
Admin_Panel
Backend
Database
Email_System
Staff_Member
Select 'Staff Accounts'
Retrieve list of accounts
Query accounts
Return account data
Send account data
Display account list
Click 'Create New Account'
Prompt for staff details
Submit form
Send form data
Validate and create new user
Confirmation of account creation
Account created successfully
alt
[Staff Account Exists]
Warning: Account already exists
Notify of existing account
Search for staff account
Send search criteria
Query for matching account
Return account details
Send account details
Display account details
Update account details
Submit updated details
Save changes
Confirmation of update
Account updated successfully
Select staff account to deactivate
Send deactivation request
Check for dependencies
alt
[Dependencies Found]
Warning: Dependencies exist
Confirm deactivation
Confirm action
Update account status to inactive
Confirmation of deactivation
Account deactivated successfully
Send notification request
Confirmation of notification
Email about account changes

Figure 5: IT Staff Account Management

3.2.2 Sequence Diagram 2: System Logs

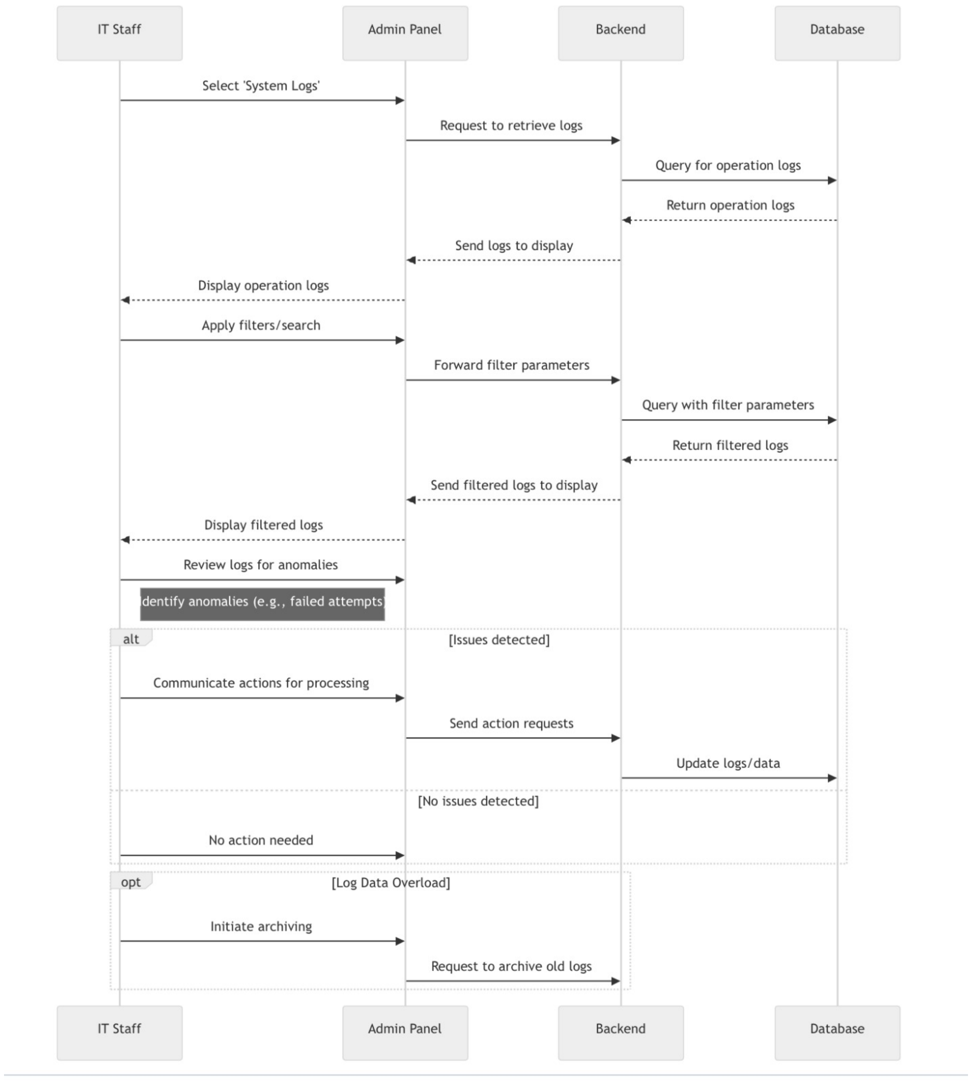
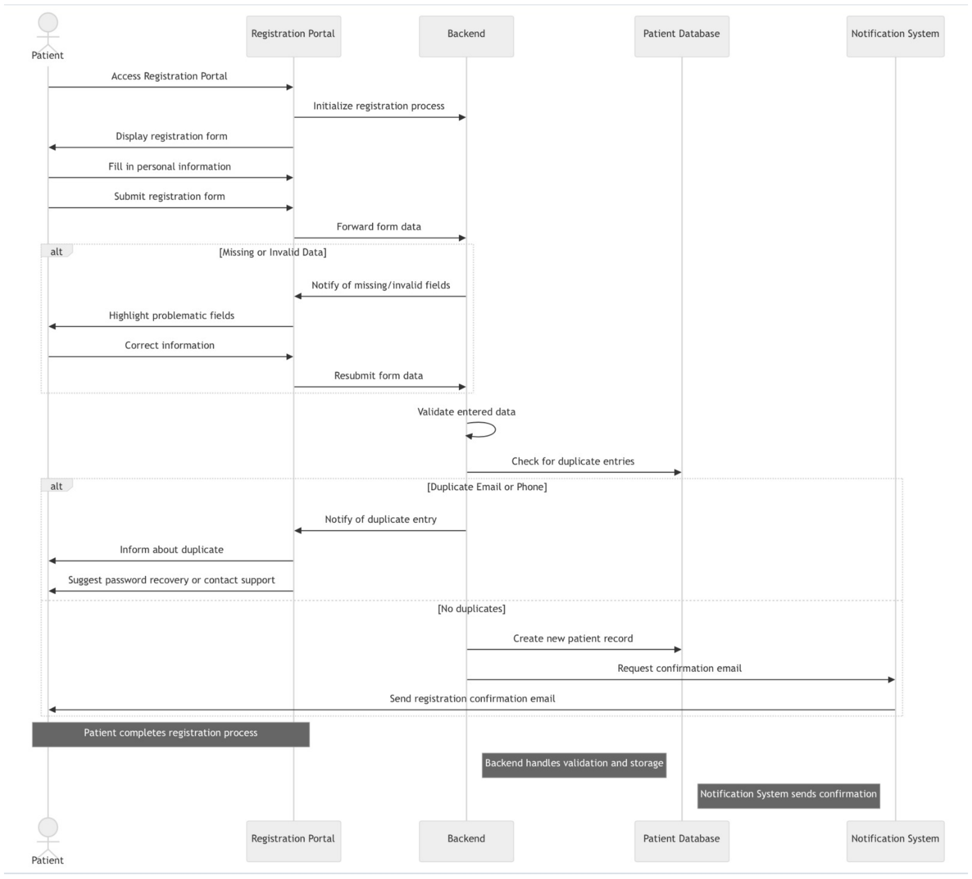


Figure 6: System Logs

3.2.3 Sequence Diagram 3: Patient Registration



Patient
Registration Portal
Backend
Patient Database
Notification System
Access Registration Portal
Initialize registration process
Display registration form
Fill in personal information
Submit registration form
Forward form data
alt
[Missing or Invalid Data]
Notify of missing/invalid fields
Highlight problematic fields
Correct information
Resubmit form data
Validate entered data
Check for duplicate entries
alt
[Duplicate Email or Phone]
Notify of duplicate entry
Inform about duplicate
Suggest password recovery or contact support
[No duplicates]
Create new patient record
Request confirmation email
Send registration confirmation email
Patient completes registration process
Backend handles validation and storage
Notification System sends confirmation

Figure 7: Patient Registration

3.2.4 Sequence Diagram 4: Appointment Scheduling

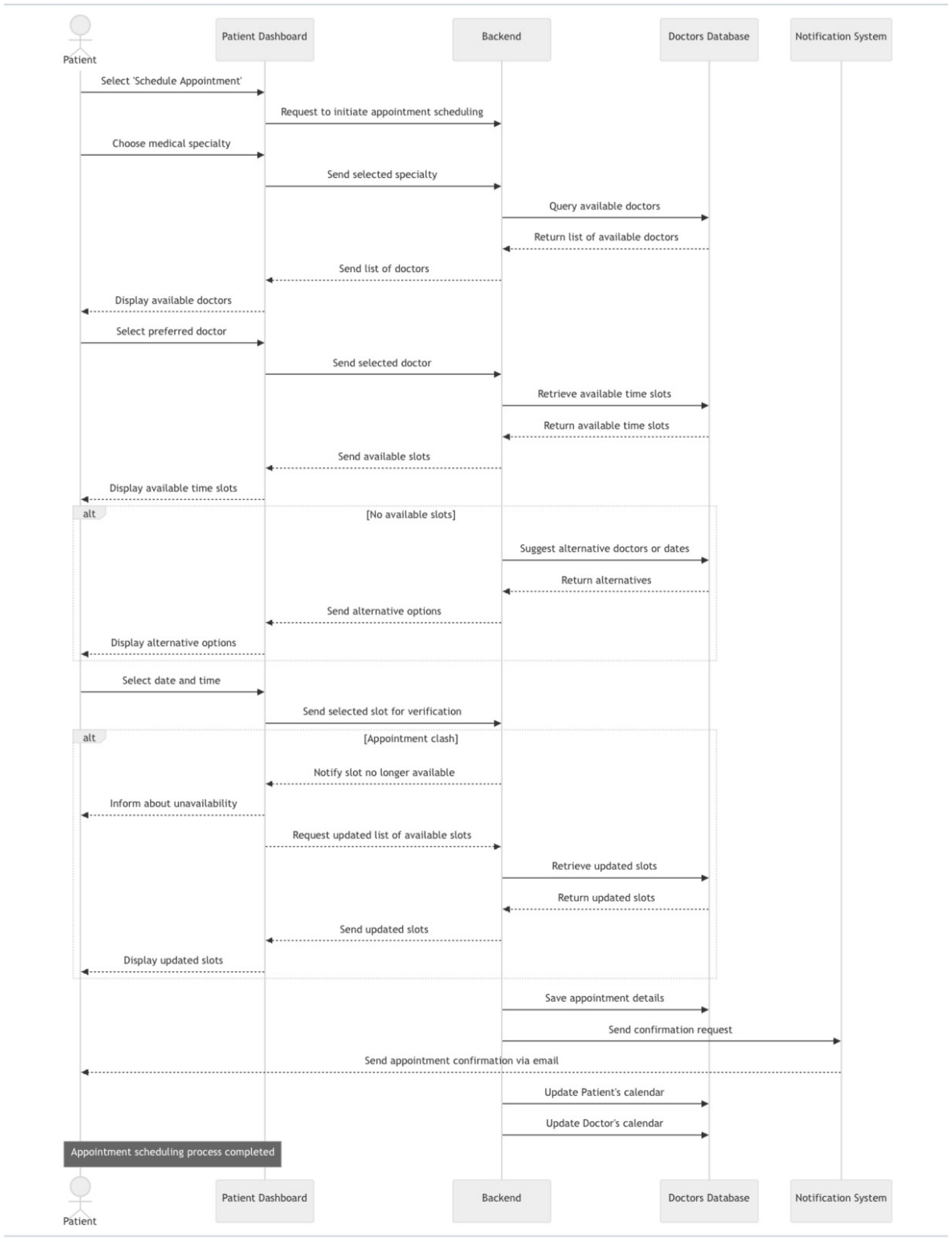

Patient
Patient Dashboard
Backend
Doctors Database
Notification System
Select 'Schedule Appointment'
Request to initiate appointment scheduling
Choose medical specialty
Send selected specialty
Query available doctors
Return list of available doctors
Send list of doctors
Display available doctors
Select preferred doctor
Send selected doctor
Retrieve available time slots
Return available time slots
Send available slots
Display available time slots
alt
[No available slots]
Suggest alternative doctors or dates
Return alternatives
Send alternative options
Display alternative options
Select date and time
Send selected slot for verification
alt
[Appointment clash]
Notify slot no longer available
Inform about unavailability
Request updated list of available slots
Retrieve updated slots
Return updated slots
Send updated slots
Display updated slots
Save appointment details
Send confirmation request
Send appointment confirmation via email
Update Patient's calendar
Update Doctor's calendar
Appointment scheduling process completed

Figure 8: Appointment Scheduling

3.2.5 Sequence Diagram 5: Medical Records

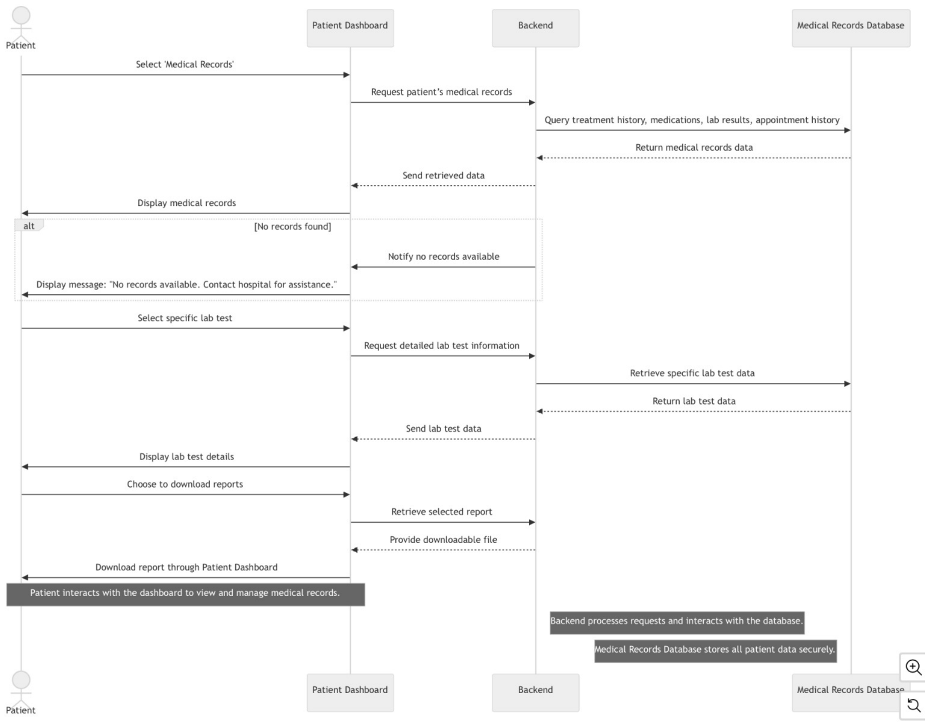

Patient
Patient Dashboard
Backend
Medical Records Database
Select 'Medical Records'
Request patient's medical records
Query treatment history, medications, lab results, appointment history
Return medical records data
Send retrieved data
Display medical records
alt
[No records found]
Notify no records available
Display message: "No records available. Contact hospital for assistance."
Select specific lab test
Request detailed lab test information
Retrieve specific lab test data
Return lab test data
Send lab test data
Display lab test details
Choose to download reports
Retrieve selected report
Provide downloadable file
Download report through Patient Dashboard
Patient interacts with the dashboard to view and manage medical records.
Backend processes requests and interacts with the database.
Medical Records Database stores all patient data securely.
Patient
Patient Dashboard
Backend
Medical Records Database

Figure 9: Medical Records

3.2.6 Sequence Diagram 6: Access Patient Medical Records

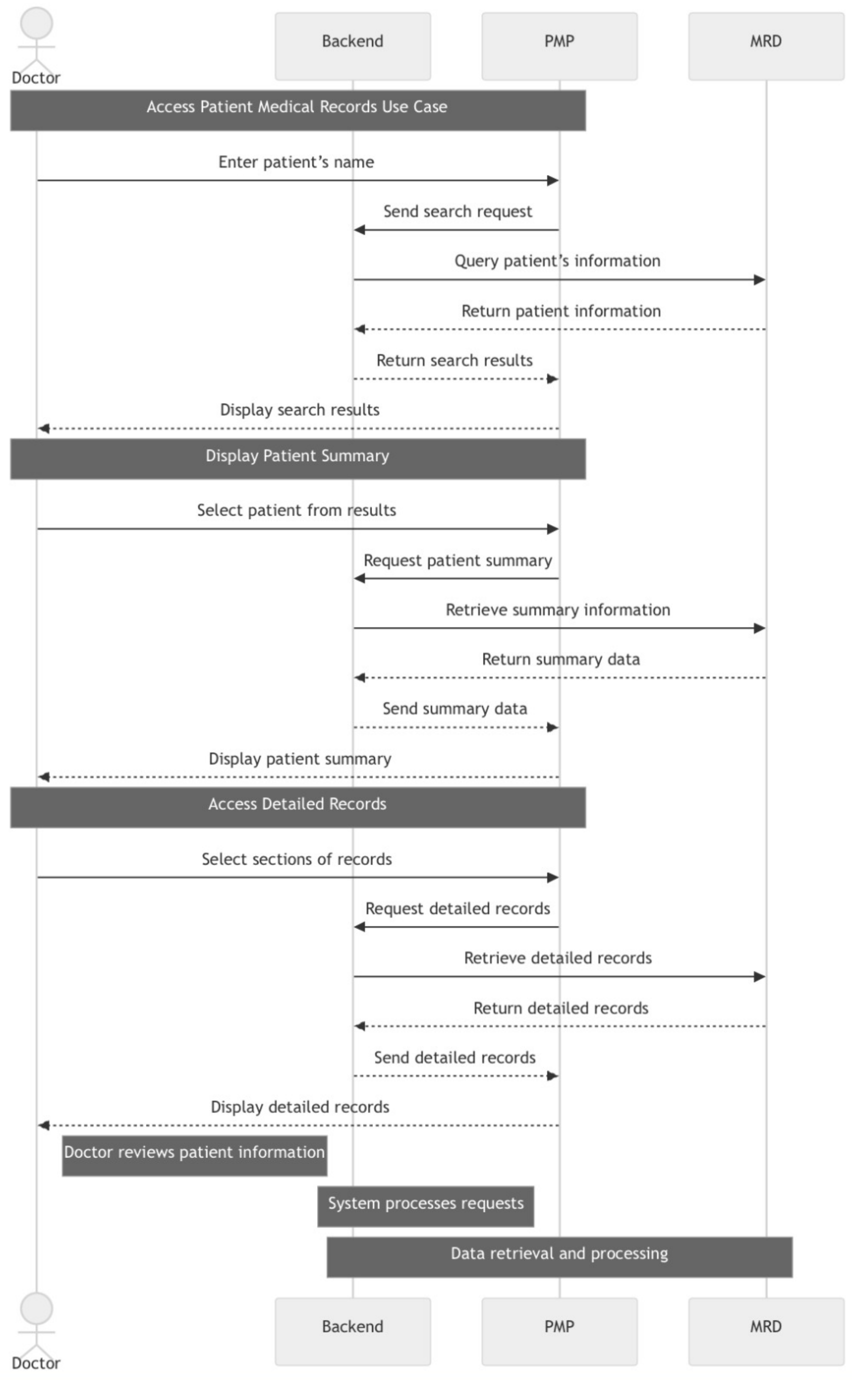


Figure 10: Access Patient Medical Records

3.2.7 Sequence Diagram 7: Lab Test Ordering

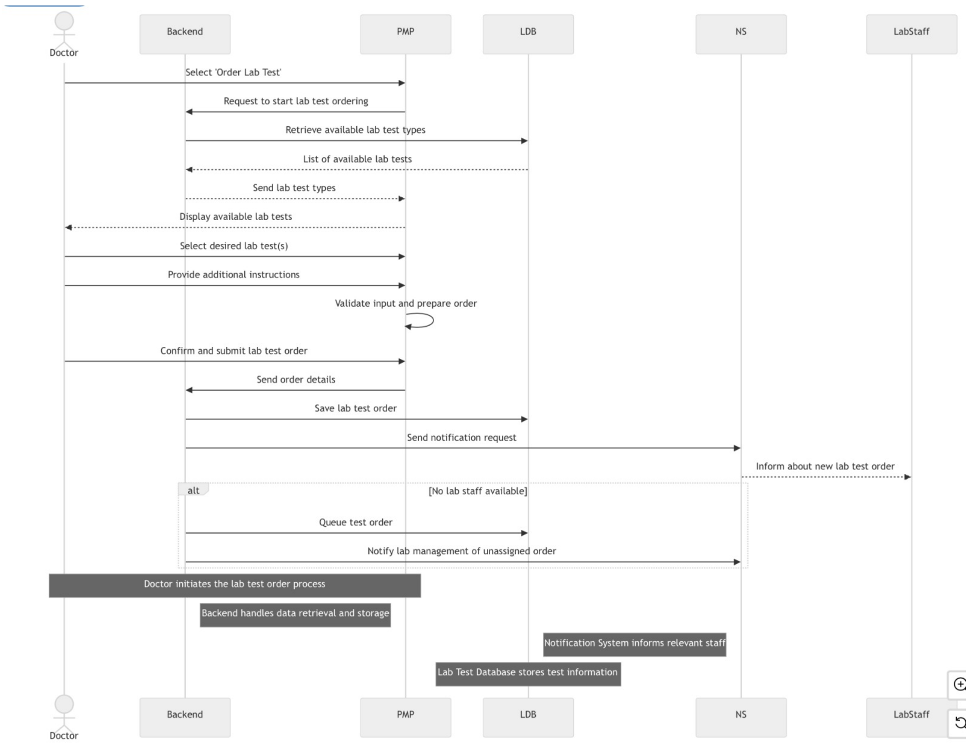

Doctor
Backend
PMP
LDB
NS
LabStaff
Select 'Order Lab Test'
Request to start lab test ordering
Retrieve available lab test types
List of available lab tests
Send lab test types
Display available lab tests
Select desired lab test(s)
Provide additional instructions
Validate input and prepare order
Confirm and submit lab test order
Send order details
Save lab test order
Send notification request
Inform about new lab test order
alt
[No lab staff available]
Queue test order
Notify lab management of unassigned order
Doctor initiates the lab test order process
Backend handles data retrieval and storage
Notification System informs relevant staff
Lab Test Database stores test information

Figure 11: Lab Test Ordering

3.2.8 Sequence Diagram 8: Prescription Management

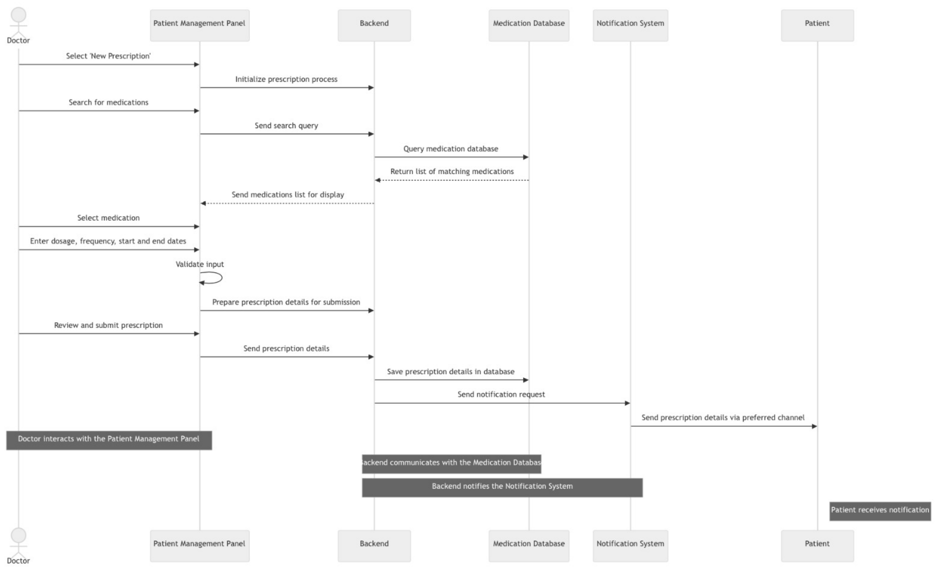

Doctor
Patient Management Panel
Backend
Medication Database
Notification System
Patient
Select 'New Prescription'
Initialize prescription process
Search for medications
Send search query
Query medication database
Return list of matching medications
Send medications list for display
Select medication
Enter dosage, frequency, start and end dates
Validate input
Prepare prescription details for submission
Review and submit prescription
Send prescription details
Save prescription details in database
Send notification request
Send prescription details via preferred channel
Doctor interacts with the Patient Management Panel
ackend communicates with the Medication Databas
Backend notifies the Notification System
Patient receives notification

Figure 12: Prescription Management

3.2.9 Sequence Diagram 9: Pending Lab Tests

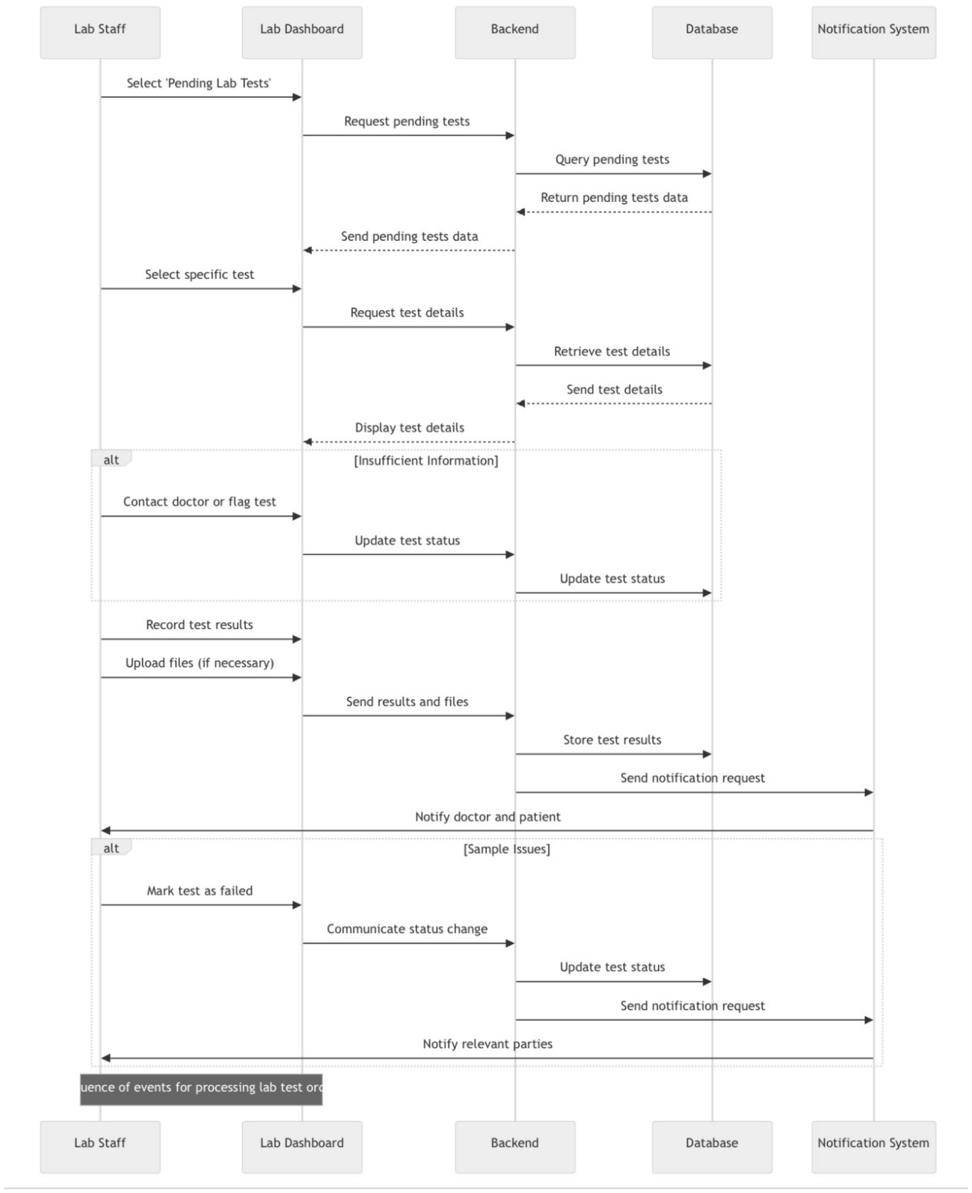


Figure 13: Pending Lab Tests

3.3 Data Flow Diagram

Since we did not make any changes to the Data Flow Diagrams included in the Requirements Specification document, they have not been added to this document. For reference, please refer to the Requirements Specification document. We believe that our implementation in that paper was up to the required standards from this class according to the slides shown in lectures.