EASTERN INTERNATIONAL UNIVERSITY

CSW 303: SOFTWARE ENGINEERING

QUARTER 3: 2024-2025

GROUP MEMBER

ONG VAN THO 2331200111

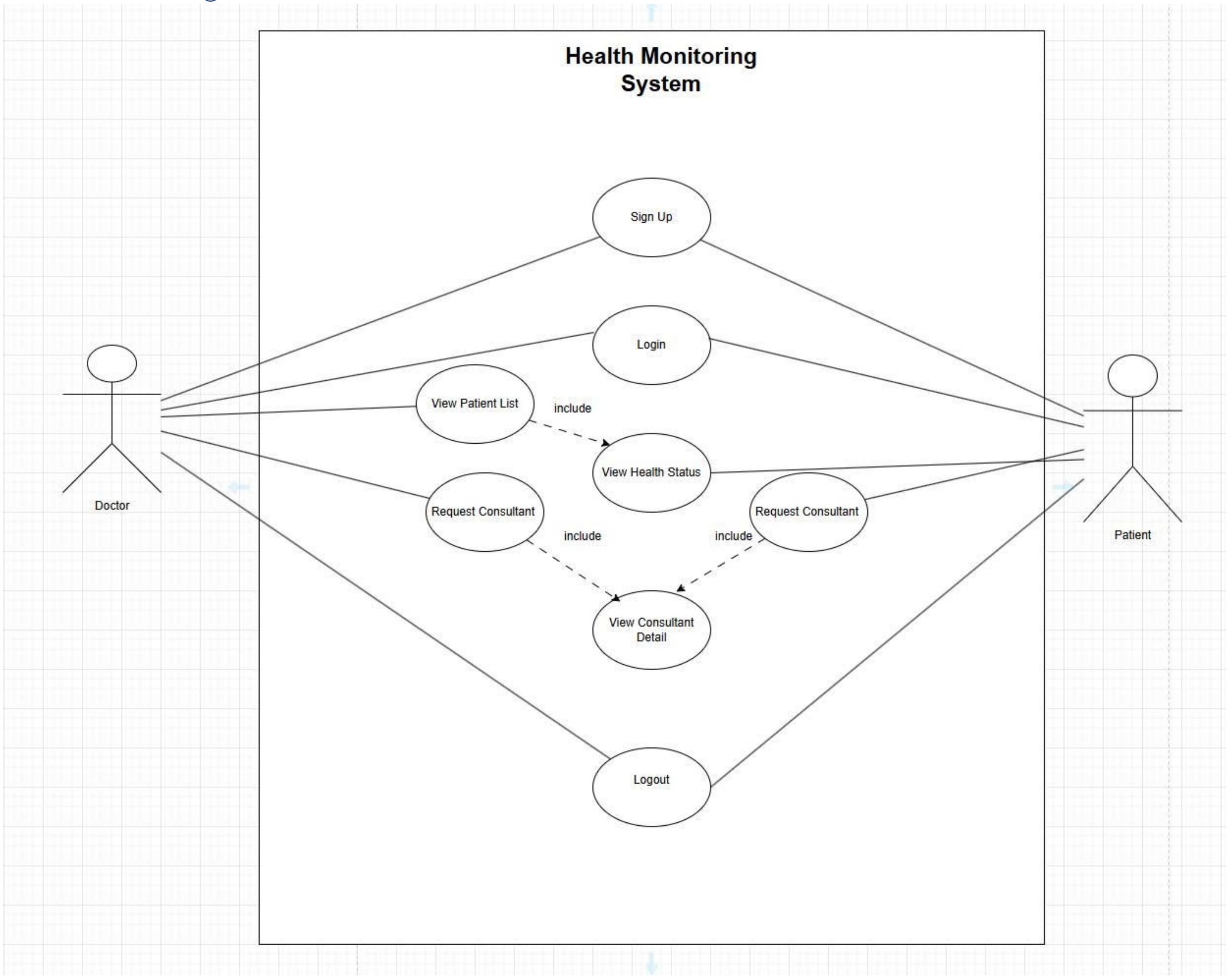
NGUYEN HOANG DUY KHANG

I. Objectives

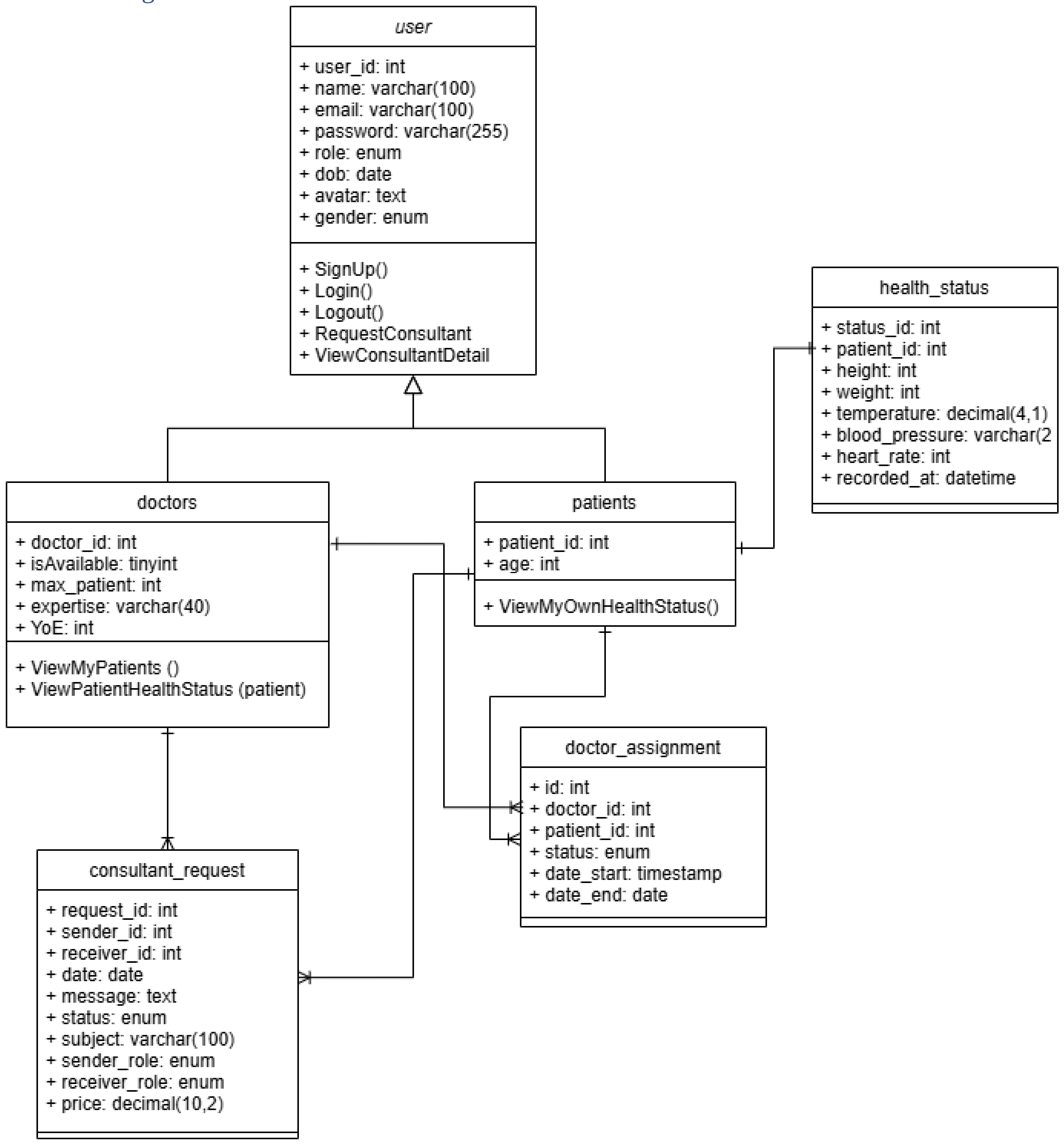
1) **Enable Efficient Communication Between Patients and Doctors**
Provide a platform where patients can quickly send consultant requests and messages to their assigned doctors for timely support and care.
2) **Facilitate Health Data Management**
Allow users (both patients and doctors) to track, access, and update health-related information such as symptoms, statuses, or treatments securely.
3) **Support Role-Based Access and Personalization**
Differentiate functionalities and dashboards based on the user's role (e.g., patient or doctor), ensuring relevant features and data are shown to each.
4) **Ensure Secure Authentication and Session Management**
Implement secure login and user session tracking using JWT and cookies to protect sensitive health and personal data.

II. System Modeling Diagram

Use Case Diagram:



Class Diagram:



III. Functional & Non-Functional

1) Functional

- **User Authentication and Role-Based Access**
  - Users (patients/doctors) can log in securely.
  - Role-based redirection and dashboard customization based on user type.
- **Health Status Dashboard**
  - Patients and doctors can view or update health-related information.
  - Each role has access to different views/data depending on permissions.
- **Consultant Request System**
  - Patients can send consultation requests to their assigned doctors with date, subject, and message.
  - Doctors can view and respond to these requests.
- **My Doctor / My Patient View**
  - Patients can view information about their assigned doctors.
  - Doctors can see a list of patients they're responsible for.
- **Logout and Session Handling**
  - logout function that clears authentication cookies.
- **Responsive Sidebar and Navigation**
  - Dynamic menu items change based on role (doctor or patient).
  - Active state management for current pages (in progress).

2) Non-Functional

- **Reliability**
  - Error handling for both frontend (try-catch) and backend (status codes, custom error messages).
- **User-friendly UI**
  - Notifications / Alert for every user's actions (confirm, delete, success alert,…).
- **Reusable Components**
  - Try to create components that can be reuse many times.

IV. Programming Languages, Software Tools, and Databases Used

1) Programming Languages:
   - **Javascript**
   - **HTML**
   - **CSS**
   - **SQL**: For interacting with the MySQL database

2) Software Tools & Frameworks:
   - **React.js**: Frontend framework for building interactive UIs.
   - **Next.js (App Router)**: Provides structure, routing, and server-side rendering.
   - **Express.js**: Backend framework for handling API requests.
   - **Axios**: For making HTTP requests between frontend and backend.
   - **bcrypt**: For password hashing and security.
   - **jsonwebtoken (JWT)**: For user authentication and session handling.
   - **draw.io**: For creating class diagrams and other software models.
   - **VS Code**: Development environment.

IV. Software Architecture

Architecture Type: Multi-Tier (Three-Tier Architecture)

This project follows a Three-Tier Architecture, which separates concerns into:

1. **Presentation Layer (Frontend/UI)**
   - Built using **React.js** with Next.js.
   - Handles user interaction and displays data from the backend.
2. **Application Layer (Backend/API)**
   - Built using **Node.js with Express.js**.
   - Manages business logic, user sessions, and communication between frontend and database.
3. **Data Layer (Database)**
   - Uses **MySQL** for storing persistent data (users, roles, consultant records).
   - Interacts with backend through SQL queries.