FINAL PROJECT DOCUMENTATION

AI-Disease Predictor

Submitted By

Zeeshan Nadeem	20-Arid-1896
Muhammad Nouman Sarwar	20-Arid-1872

Supervisor

Mr. Ashar Javed

Lecturer

2024

Gujrat Institute of Management Sciences

PMAS-Arid Agriculture University, Rawalpindi

Members' Detail

Project ID	GIMS-BSIT-20207

AI-Disease Predictor			
Group Leader: Zeeshan Nadeem **Group Members:** 2			
Zeeshan Nadeem	20-Arid-1896	zeeshannadeem20arid1896@gmail.com	BSIT
Muhammad Nouman Sarwar	20-Arid-1872	muhammadnoumansarwar20arid1872@gmail.com	BSIT

_____

Mr. Awais Ilyas Baig

PMO, GIMS

Dated: _____

Approval

I Mr. Ashar Javed am willing to guide these students in all phases of the project titled "AI-Disease Predictor" as supervisor. I have carefully seen the Title and description of the proposal and believe that it is of an appropriate difficulty level for the number of students named above.

Supervisor

Mr. Ashar Javed
Lecturer, GIMS

Dated: _____

Abstract

The "AI-Disease Predictor" is an integrated healthcare platform aiming to revolutionize patient care and diagnosis by combining artificial intelligence and medical expertise. This multifaceted system encompasses various key functionalities to streamline the patient's healthcare journey. Its primary objective is to empower patients by offering a comprehensive suite of tools and services, ultimately enhancing disease prediction, facilitating doctor-patient interactions, and optimizing healthcare management. At its core, the system enables patients to input symptoms and undergo an AI-driven disease prediction process, providing potential diagnoses, detailed descriptions, and precautionary measures. Additionally, the platform incorporates a sophisticated machine learning algorithm for brain tumor detection via MRI analysis, presenting users with insights, albeit with acknowledgment of potential limitations in accuracy. The integration of these cutting-edge technologies into a user-friendly interface marks a significant advancement in predictive healthcare solutions, promising early detection and timely intervention. Moreover, the platform serves as a nexus for doctor-patient interactions, offering appointment scheduling, and facilitating communication through WhatsApp. Patients can access doctor profiles, receive tailored suggestions, manage appointments, and maintain a network of preferred physicians. For administrators, management tools for patient and doctor databases, appointment tracking, analytics, and overseeing system functionality are provided, ensuring efficient healthcare service delivery.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Chapter 1 Introduction

In an era shaped by technological advancements, healthcare has emerged as a frontier ripe for innovation. The "**AI-Disease Predictor**" project is a pioneering endeavor that fuses artificial intelligence with healthcare services, aiming to revolutionize patient care, diagnosis, and medical interactions.

The primary goal of this project is to empower individuals seeking healthcare solutions by providing an integrated platform that amalgamates AI-driven disease prediction, advanced MRI image analysis for brain tumor detection, streamlined appointment scheduling, and seamless doctor-patient communication. The intended beneficiaries are diverse, encompassing patients seeking accurate diagnoses, doctors aiming for enhanced patient interactions, and administrators striving for efficient healthcare management.

With a comprehensive scope, this project aims to address critical challenges in healthcare access, diagnosis, and communication. It endeavors to provide users with a user-friendly interface to input symptoms and receive AI-powered disease predictions while also allowing for the upload and analysis of brain MRI images for potential tumor detection. The approach blends sophisticated algorithms with user-centric design, ensuring accessibility and reliability while acknowledging the inherent limitations in medical prediction.

Assumptions underpinning this project revolve around the integration of advanced technologies into healthcare, the adherence to ethical data practices, and the understanding that while AI augments medical diagnosis, it does not replace professional medical advice.

Anticipated outcomes encompass early disease detection, improved doctor-patient interactions through virtual consultations, and streamlined healthcare administration through efficient appointment management and analytics. This project seeks to pave the way for a future where technology and healthcare converge to deliver more accurate, accessible, and personalized medical services.

1.1 Problem Statement

The challenge in healthcare extends to delayed disease prediction from symptoms, inadequate brain tumor detection tools via MRI, and barriers for international patients seeking medical advice in their native language or from family doctors. Quantitatively, there's a notable delay in diagnosing diseases and detecting brain tumors, impacting timely interventions. Qualitatively, international patients face hurdles in accessing personalized medical guidance, highlighting a global healthcare gap in catering to their linguistic and cultural needs.

This project targets the inefficiencies in disease prediction and brain tumor detection while aiming to bridge the gap for international patients. By leveraging advanced technologies, it seeks to provide accessible tools for early diagnosis and culturally sensitive medical advice. Ultimately, this initiative aims to revolutionize healthcare accessibility, ensuring that individuals, regardless of their location, receive timely and personalized care from their native language or family doctors, thereby advancing global healthcare inclusivity.

1.2 Significance

This project holds paramount significance in revolutionizing healthcare accessibility and patient-centric services. Its distinctiveness lies in its holistic approach, merging AI-driven disease prediction, MRI-based brain tumor detection, and enhanced doctor-patient interactions, particularly catering to international patients' needs for native language-speaking or family doctor consultations.

What sets this project apart is its comprehensive integration of cutting-edge technologies into a user-friendly healthcare platform. It offers unique AI-driven disease prediction tools and MRI analysis for brain tumor detection, aiming for early diagnosis and intervention. Moreover, its emphasis on facilitating doctor-patient interactions, especially for international patients, distinguishes it by addressing cultural and linguistic barriers that have been previously overlooked.

The project's contribution lies in its potential to significantly advance healthcare inclusivity on a global scale. Allowing individuals, irrespective of their geographic location, to access personalized medical advice in their native language or from their family doctors, fills a crucial gap in healthcare services. Policy implications stem from

the need for greater recognition and integration of technology-driven solutions in healthcare systems worldwide, emphasizing the importance of culturally sensitive and accessible patient care.

In essence, this project's significance lies in its potential to reshape healthcare services, fostering inclusivity, early intervention, and personalized care for patients globally. Its innovation and integration of AI and healthcare services mark a pivotal step towards bridging existing gaps in the healthcare system and setting new standards for patient-centric care.

1.3 Objectives

The "AI-Disease Predictor" project aims to create an integrated healthcare platform that utilizes advanced technologies to enhance disease prediction, facilitate early detection of brain tumors via MRI analysis, and improve doctor-patient interactions, especially for international patients seeking consultations in their native language or with their family doctors.

The objectives are as follows:

- ✓ **Develop an AI-driven disease prediction system:** Create an algorithm capable of analyzing symptoms input by users to accurately predict potential diseases with a high degree of reliability and specificity.
- ✓ **Implement an MRI-based brain tumor detection tool:** Build a machine learning algorithm that can analyze MRI images uploaded by users to detect the presence of brain tumors, providing insights and early indications while acknowledging potential accuracy limitations.
- ✓ **Enable doctor-patient interactions:** Design and integrate a user-friendly interface that facilitates appointment scheduling, virtual consultations, and communication via WhatsApp to bridge the gap for international patients seeking consultations in their native language or with their family doctors.
- ✓ **Measure and optimize accuracy and efficiency:** Establish metrics to evaluate the accuracy of disease prediction and brain tumor detection algorithms, aiming for continuous improvement and optimization of these features over time through feedback and data analysis.

These objectives form the basis for the project's activities and serve as measurable criteria for evaluating the success and impact of the developed healthcare platform. Ultimately, the project seeks to revolutionize healthcare accessibility and personalized patient care through the integration of advanced technologies and patient-centric features.

1.4 Limitations and Restrictions:

The "AI-Disease Predictor" system operates within specific limitations and restrictions, both internal and external, which are important to acknowledge for a realistic understanding of its functionalities:

- **Limited Symptom Dataset:** The system is constrained by a predefined set of 132 symptoms. It may not encompass every possible symptom, potentially limiting the scope of disease prediction for less common or atypical symptoms.

- **Restricted Disease Predictions:** The system can predict diseases within a predefined set of diseases. It does not cover all possible medical conditions, and users should be aware that certain diseases may not be included in the prediction capabilities of the system.

- **MRI Analysis Scope:** The machine learning algorithm for brain tumor detection is trained on a specific range of predictable diseases. The system may not provide accurate results for conditions outside this predefined range.

- **Predictive Accuracy:** The accuracy of disease predictions and brain tumor detections is influenced by the limitations of available data and the complexity of medical conditions. Users should recognize that the system's predictions are not infallible, and consultation with healthcare professionals is advisable for accurate diagnosis and treatment.

- **International Doctor-Patient Interactions:** The system facilitates international consultations but may not guarantee the availability of doctors proficient in all languages. Users are encouraged to verify the language capabilities of available doctors before scheduling appointments.

- **System Updates:** The system's performance and capabilities are subject to periodic updates and improvements. Users should be aware that advancements in

medical knowledge and technology may necessitate updates to enhance prediction accuracy and expand the system's capabilities.

These limitations and restrictions are integral to managing user expectations and ensuring a responsible and transparent use of the "AI-disease Predictor" system. Users are encouraged to approach the system as a supportive tool in healthcare decision-making, acknowledging its constraints and seeking professional medical advice when necessary.

1.5 Overview

Project Goal:

The overarching goal of the "AI-Disease Predictor" project is to pioneer a technologically advanced and user-centric healthcare platform that enhances disease prediction, enables early detection of brain tumors, and facilitates accessible and personalized doctor-patient interactions on a global scale. This goal aligns with the mission to revolutionize healthcare services by leveraging AI and machine learning, fostering inclusivity, and providing timely and accurate medical guidance.

This project aims to bridge the gap between technology and healthcare, aiming to:

- Develop an innovative system that predicts diseases based on symptoms, aiding in early diagnosis and intervention.
- Create a sophisticated tool capable of analyzing MRI images for the early detection of brain tumors within a limited range of predictable diseases.
- Establish a user-friendly platform that facilitates seamless doctor-patient interactions, particularly for international patients seeking consultations in their native language or with their family doctors.
- Cultivate a healthcare environment that emphasizes the integration of technology without compromising the importance of professional medical advice and patient-centric care.

Ultimately, the goal is to set new standards in predictive healthcare solutions, fostering a global community where individuals, regardless of geographical location, can access

timely, accurate, and culturally sensitive medical guidance, thereby enhancing healthcare accessibility and patient outcomes.

Type of project: ❑R&D ☑ Development

Project Success criteria:

The success of the "AI-Disease Predictor" project will be evaluated based on the following criteria:

- ✓ **Accuracy and Reliability:** The system's disease prediction accuracy should align closely with established medical standards. Evaluation metrics will measure the system's ability to predict diseases from symptoms accurately within the limitations of the predefined dataset.
- ✓ **MRI Analysis Performance:** The success criteria will include the system's ability to detect brain tumors accurately within the specified range of predictable diseases through MRI image analysis.
- ✓ **User Interaction and Satisfaction:** User feedback and satisfaction surveys will gauge the ease of use, effectiveness, and satisfaction levels with the platform's functionalities, especially for international patients seeking consultations in their native language or with their family doctors.
- ✓ **Improvement Over Time:** The success of the project will be measured by the system's continuous improvement in prediction accuracy and efficiency over iterations, indicating its adaptability and responsiveness to user feedback and technological advancements.
- ✓ **Stakeholder Satisfaction:** The satisfaction of stakeholders, including healthcare professionals, patients, and administrators, with the system's functionalities, accuracy, and contribution to enhancing healthcare services will be a key indicator of success.

Meeting these success criteria will signify the project's effectiveness in revolutionizing healthcare accessibility, advancing predictive healthcare solutions, and bridging the gap between technology and personalized patient care.

Risks of the Project:

Here are potential risks associated with the "AI-Disease Predictor" project and their estimated degrees of risk:

- ✓ **Technical Risk:**
  - **Data Limitations:** A limited dataset of symptoms and diseases could impact the system's accuracy and coverage. Expanding the dataset while maintaining accuracy might pose technical challenges.
  - **Algorithm Performance:** Uncertainty in the performance of AI algorithms for disease prediction and MRI analysis could affect the reliability of results. Rigorous testing and validation will mitigate this risk.
- ✓ **Timing Risk:**
  - **Development Delays:** Technical complexities or unexpected challenges during system development and testing might prolong the project timeline.
  - **Regulatory Compliance:** Adhering to healthcare regulations and ensuring ethical compliance could cause delays in deployment.
- ✓ **Budget Risk:**
  - **Resource Allocation:** Unexpected requirements or complexities may lead to increased resource allocation, impacting the budget.
  - **Technology Upgrades:** Incorporating advanced technologies for system enhancements might incur additional costs, but proper planning can mitigate this risk.

Managing these risks will require a proactive approach, including thorough testing, continuous monitoring, adapting to technological advancements, and having contingency plans to address any unforeseen challenges that may arise during the project implementation.

(Please mark ☑ where applicable)	**Low**	**Medium**	**High**
Technical risk		☑	

Timing risk			☑
Budget risk		☑	

Organization Details (if any):

N/A

Target End users:

The main end users benefiting from the "AI-Disease Predictor" project are:

- ✓ **Patients:**
  - Individuals seeking accurate and early disease predictions based on symptoms.
  - International patients looking for consultations in their native language or with their family doctors.
  - Those requiring early detection of brain tumors through MRI analysis.
- ✓ **Doctors/Healthcare Professionals:**
  - Medical practitioners using the system for supporting diagnoses and providing better-informed treatment plans.
  - Doctors catering to international patients or those seeking consultations in their native language.
- ✓ **Administrator (single in the whole system):**
  - Healthcare administrator managing appointments, patient records, doctors, and analytics within the system.

Platform:

☑ Web-based ❑ Distributed ❑ Setup Configurations
❑ Desktop based ❑ Android ❑ iOS
❑ Other_____

Project Supervisor: Mr. Ashar Javed

FOR OFFICE USE ONLY	
Approved	☐ Yes ☐ No

Approving Committee:

Date: _____

Mention the Reason(s) if not approved:

1.6 System Architecture

1.6.1 3-Tier Architecture

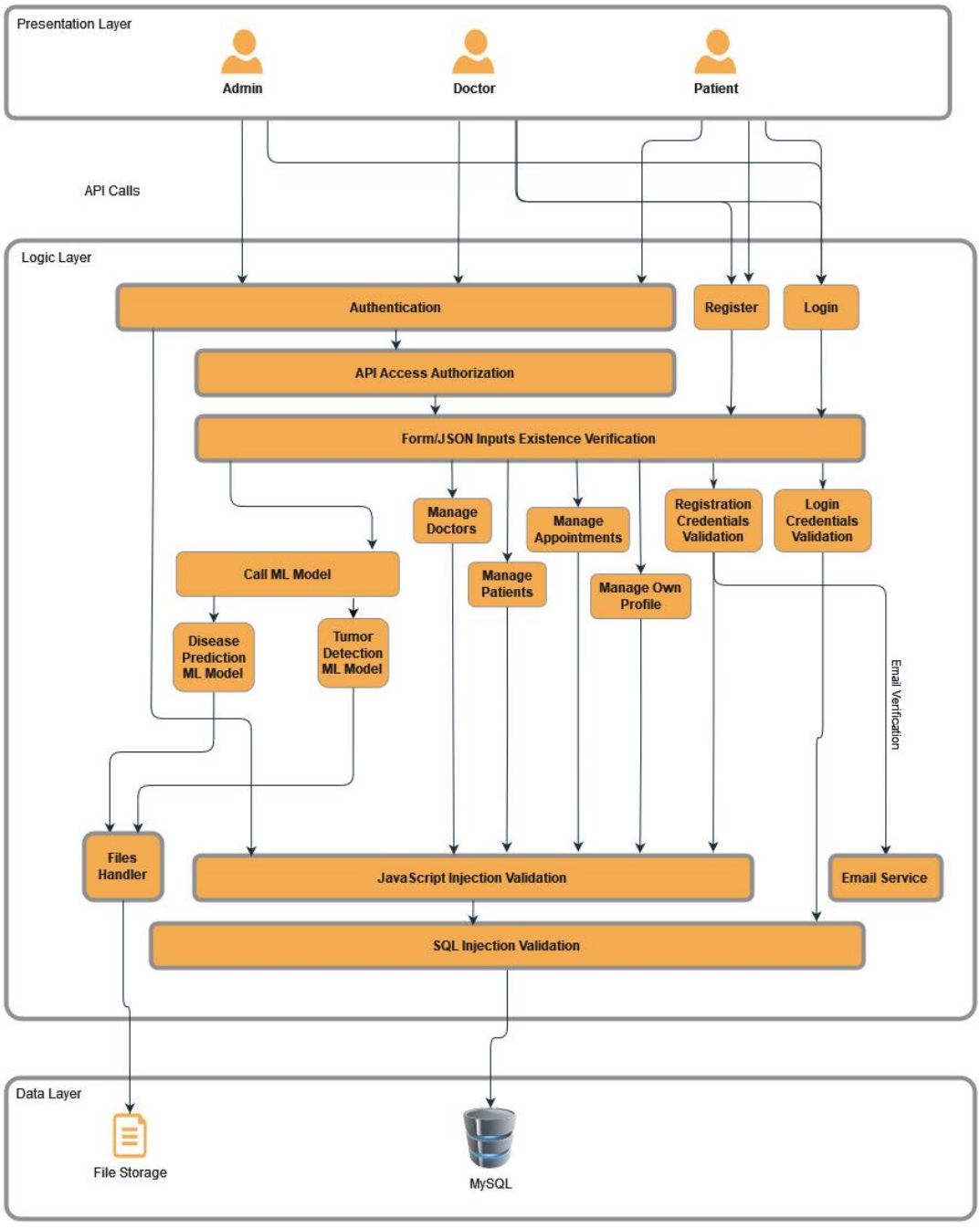

Figure 1.1: 3 Tier Architecture

1.6.2 Disease Prediction Machine Learning Model Architecture

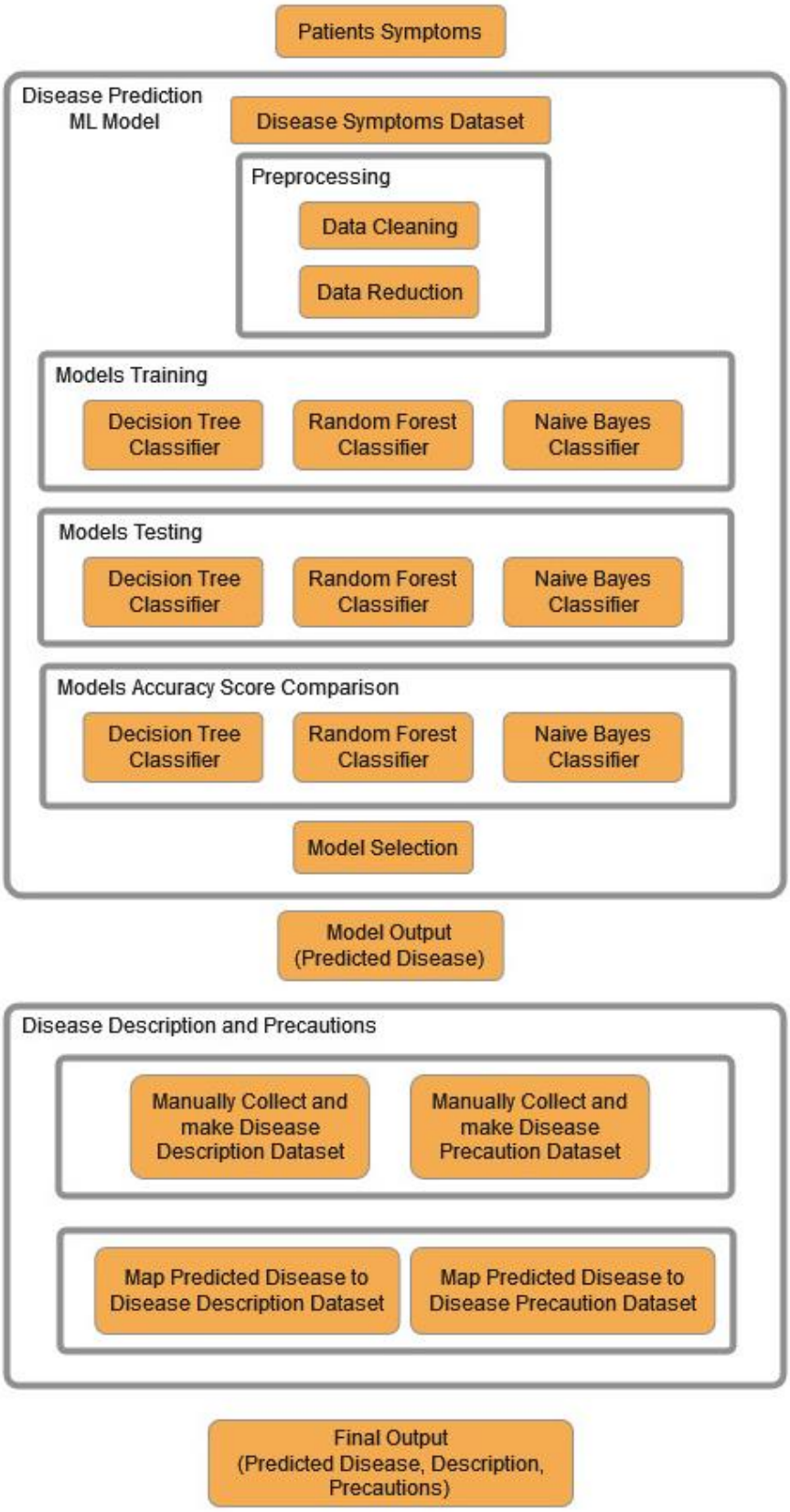

Figure 1.2: Disease Prediction Machine Learning Model Architecture

1.6.3 Brain Tumor Detection Machine Learning Model Architecture

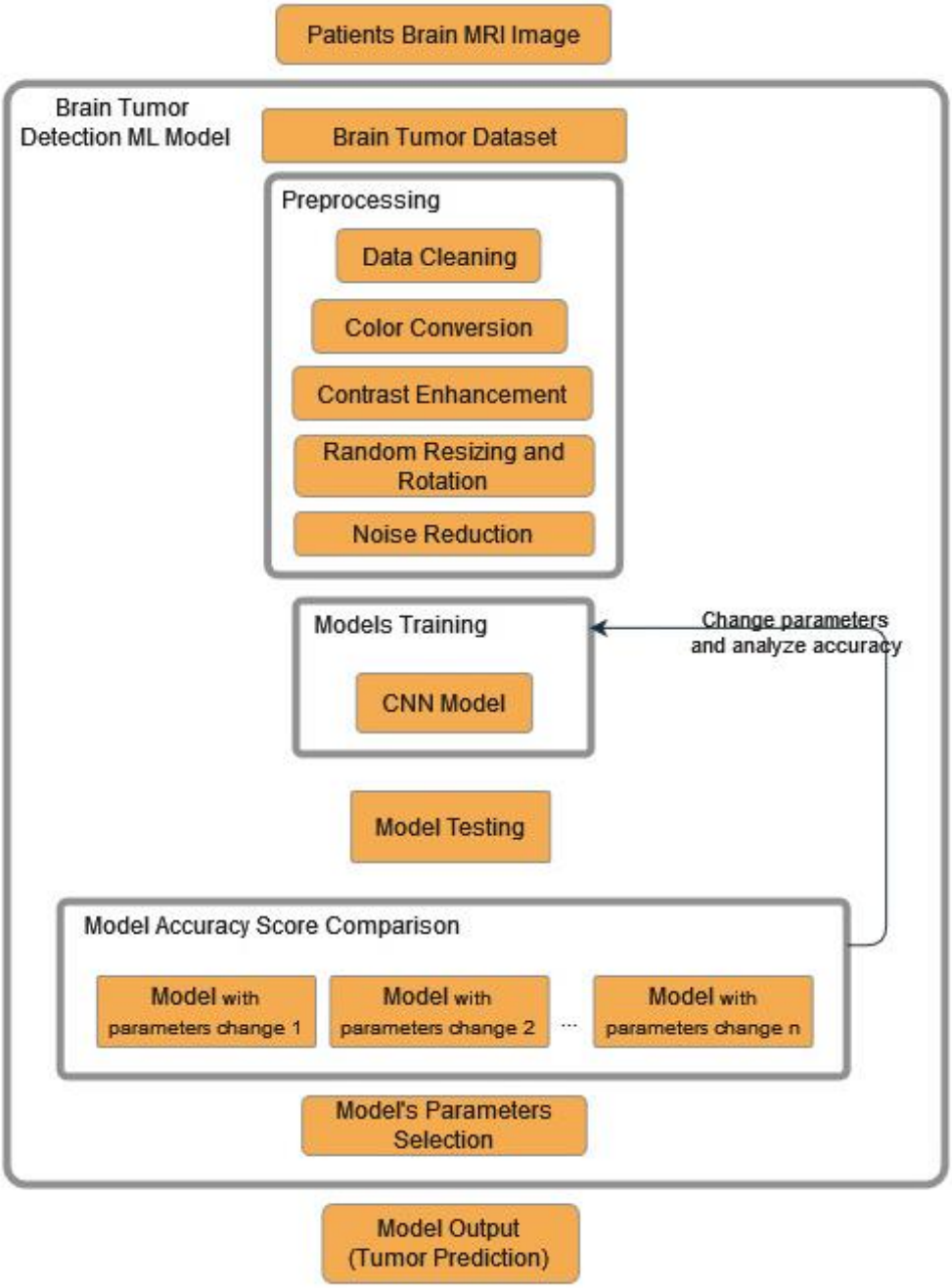

Figure 1.3: Brain Tumor Detection Machine Learning Model Architecture

1.7 Software/Hardware Requirements

Software Requirements:

- ✓ **Operating System:** Compatible with Windows 10, macOS, and Linux distributions.
- ✓ **Python:** Version 3.8 or later.
- ✓ **Flask Framework:** Required for the backend development.
- ✓ **Keras, CNN model(s):** Necessary for implementing convolutional neural networks for brain tumor detection.
- ✓ **ML model(s):** Utilizing ML algorithms/models are required for disease prediction.
- ✓ **Angular 17:** Framework for frontend development.
- ✓ **Tailwind CSS:** CSS framework for styling the user interface.
- ✓ **MySQL:** Database management system, preferably with XAMPP Server for local development.
- ✓ **Development Tools:** Visual Studio Code, PyCharm for coding, GitHub for version control and collaboration.

Hardware Requirements:

- ✓ **Processor:** Intel Core i7 or equivalent AMD processor.
- ✓ **RAM:** Minimum 8GB RAM for development purposes; 16GB or higher recommended.
- ✓ **Storage:** 256GB SSD or higher for faster development and data handling.
- ✓ **Graphics Card:** Specifically required for dealing with brain tumor detection CNN model(s).
- ✓ **Internet Connectivity:** Stable internet connection for accessing external resources and APIs during development and testing.

These specifications provide a baseline for software and hardware requirements needed to develop and test the "AI-Disease Predictor" system. Adjustments or additional resources might be necessary based on specific development needs and scalability considerations.

1.8 Implementation Tools and Technology

Frontend:

- ✓ **Angular 17:** For creating dynamic and responsive UI.
- ✓ **Tailwind CSS:** For efficient and flexible styling.

Backend:

- ✓ **Python with Flask:** For building the backend logic and APIs.
- ✓ **Keras with CNN** for brain tumor detection, **Decision Tree, Random Forest, and Naïve Bayse Classifiers** for disease prediction.

Database:

- ✓ **MySQL:** For managing and storing user data and medical information.

Version Control and IDEs:

- ✓ **Visual Studio Code, PyCharm:** For code editing and development.
- ✓ **GitHub:** For version control and collaboration among team members.

Reasoning:

- ✓ **Python:** Widely used for machine learning tasks and backend development.
- ✓ **Angular 17:** Offers robust frontend development capabilities.
- ✓ **Flask:** Lightweight and efficient for backend development in Python.
- ✓ **MySQL:** Popular and reliable for database management.
- ✓ **GitHub:** Essential for version control and collaborative development.

The choice of these tools and technologies is driven by their effectiveness, compatibility, popularity, and ability to cater to the specific requirements of the "AI-Disease Predictor" project, aiming for a comprehensive and scalable healthcare platform.

1.9 Implementation Plan

1.9.1 Deliverable Items

- ✓ Project Proposal
- ✓ Software Requirement Specification (SRS)
- ✓ Project Presentation and Final Documentation
- ✓ Executable Files
- ✓ CD
- ✓ User manual

1.9.2 Milestone Chart

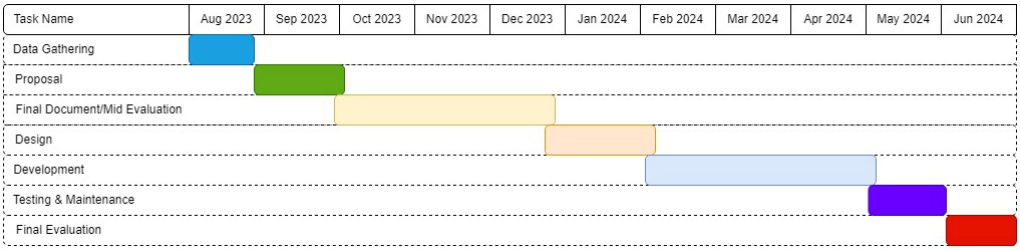

Figure 1.4: Gantt chart for milestones

Chapter 2
Requirement Analysis

Chapter 2 Requirement Analysis

2.1 Functional Requirements

2.1.1 For Administrator

(Single Admin in the whole system)

- ✓ **User login:**
  - Admin should be able to log in to the system using credentials to access the system features.
- ✓ **View Analytics:**
  - Admin should be able to view system analytics (limited).
- ✓ **Manage Patients:**
  - Admin should be able to view and update (only limited information) of the patients in the system.
  - Admin should be able to suspend/unsuspend a specific patient.
- ✓ **Manage Doctors:**
  - Admin should be able to view and update (only limited information) of the doctors in the system.
  - Admin should be able to suspend a specific doctor.
  - Admin should be able to approve/not approve a specific doctor.
- ✓ **View Appointments:**
  - Admin should be able to view all appointments in the system.
- ✓ **Absence Confirmation**
  - Admin should be able to approve or reject a doctor's request about a patient's absence. He can also view the recorded video uploaded by the doctor.
- ✓ **Doctor's Specialization**
  - Admin should be able to view, update, and delete the doctor's specialization.
- ✓ **Manage Languages**
  - Admin should be able to view, update, and delete languages.
- ✓ **Manage Transaction**
  - The admin should be able to mark withdrawal requests (sent by doctors) as completed or rejected.
  - The admin should be able to mark refund withdrawal requests (sent by patients) as completed or rejected.

- ✓ **Logout**
  - Admin should be able to log out from the system.

2.1.2 For Patients

- ✓ **User Registration and Login:**
  - Patients should be able to register and log in using their credentials to access the system's features.
- ✓ **User Password Reset:**
  - Patients should be able to reset their passwords by verifying their email addresses.
- ✓ **Manage Profile:**
  - Patients should be able to view their profiles.
  - Patients should be able to update their profiles according to the system's criteria.
- ✓ **Symptom-Based Disease Prediction**
  - Patients should be able to input their symptoms for disease prediction and should also be able to get potential predicted disease, and precautions for the predicted disease (unless it is not a critical disease).
- ✓ **Brain Tumor Detection**
  - Patients should be able to upload Brain MRI images for brain tumor detection and get results from it.
- ✓ **Manage Appointments:**
  - Patients should be able to view doctors' profiles.
  - Patients should be able to book an appointment with a doctor by going through a defined process.
  - Patients should be able to view or cancel upcoming appointments.
  - Patients should be able to view meeting times of the upcoming appointments.
- ✓ **Manage transactions**
  - Patient should be able to view previous transactions.
  - Patients should be able to withdraw the refunded amount.
- ✓ **Review**

- The patient should be able to provide a review after the appointment has been completed.

✓ **Reschedule appointment**

- Patient should be able to reschedule the appointment according to system rescheduling criteria.

✓ **Logout**

- The patient should be able to log in from the system.

2.1.3 For Doctors

✓ **Doctor Registration and Login:**

- Doctors should be able to register and log in using their credentials to access the system's features.

✓ **Doctors Password Reset:**

- Doctors should be able to reset passwords by going through a defined process.

✓ **Manage Profile:**

- Doctors should be able to view their profile.
- Doctors should be able to update their profiles.

✓ **Manage Appointments:**

- Doctors should be able to view past and upcoming appointments.
- Doctors should be able to cancel the upcoming appointments.
- Doctors should be able to mark the appointment as completed.
- Doctors should be able to get the patient's WhatsApp number they have booked the appointment with.
- Doctors should be able to send patient’s absence requests.
- Doctors should be able to delay the meeting.
- Doctors should be able to confirm that they were unable to join the meeting.

✓ **Manage transactions**

- Doctors should be able to view previous transactions.
- Doctors should be able to withdraw their refundable amount.

✓ **Send Approval Request to administration**

- Doctors should be able to send approval requests to admin.

- ✓ **Logout**
  - Doctors should be able to log in from the system.

2.2 Non-Functional Requirement

Here are the non-functional requirements for the "AI-Disease Predictor" system:

- ✓ **Usability**
  - **User-Friendly Interface:** The system should have an intuitive and easy-to-navigate interface for users, doctors, and administrators.
- ✓ **Reliability**
  - **System Availability:** The platform should be available 24/7, ensuring uninterrupted access for users globally.
  - **Data Security:** Ensure robust security measures to protect user data and maintain confidentiality.
- ✓ **Performance**
  - **Fast Response Time:** The system should respond promptly to user interactions, minimizing loading and processing times.
  - **Scalability:** Design the system to handle varying user loads and data volumes without compromising performance.
- ✓ **Design Constraints**
  - **Responsive Design:** The system should be responsive across **[4]** different devices and screen sizes, ensuring consistent user experience.
  - **Consistent Branding:** Maintain a consistent design language **[4]** and branding elements throughout the platform.
- ✓ **Portability:**
  - **Cross-Browser Compatibility:** Ensure compatibility across major web browsers (Chrome, Firefox, Safari, etc.) for seamless usage.
  - **Mobile Responsiveness:** Optimize the system to function effectively on both mobile and desktop devices.
- ✓ **Maintainability:**
  - **Code Maintainability:** Develop clean, modular, and well-documented code to facilitate future updates and maintenance.

- **Easy Upgrades**: Design the system architecture to allow for easy integration of future enhancements and updates (Note that there will always be a single administrator in the system).

✓ **Security Against Malware Code Injection:**

- **SQL Injections:** The system should take into account the possibilities of SQL injections and their possible preventions.
- **JavaScript Injections:** The system should also validate all the textual data provided by any Stakeholder for JavaScript code injection.

These non-functional requirements aim to ensure a highly usable, reliable, performant, and maintainable healthcare platform while upholding data security and compliance with legal agreements.

2.3 Use Cases

2.3.1 Sign-up Use Case

Table 2.1: Sign-up Use Case

Use Case Number	UC-1
Use Case Name	UC-Sign-up
Goal	The doctor or patient wants to create a new account in the system
Primary actor	Doctor, Patient
Level	User, System
Precondition	**PRE: 1** The user does not have an existing account in the system. **PRE: 2** The user has opened the login page.
Success end	The user successfully registers themselves into the system
Failure end condition	**1.** The user enters incorrect information in the form fields. **2.** The user enters an incorrect email verification code.
Main success scenario	**1.** The user clicks on the "Register" button. **2.** The system redirects the user to the registration page. **3.** The user enters his credentials and clicks on the "Register" button. **4.** The system validates and verifies the credentials from the database and redirects the user to the email verification page. **5.** The user enters the email verification code and clicks on the "Verify Email" button. **6.** The system redirects the user to their respective dashboard.
Extensions (error scenarios)	**4a.** If the user's email already exists in the database or enters invalid credentials. **4a.1.** The system shows a pop-up error message. **4a.2.** The user tries again with different credentials. **6a.** If the email verification code is incorrect. **6a.1.** The system shows an error message. **6a.2.** The user tries again with the correct verification

code.

2.3.2 Sign-in Use Case

Table 2.2: Sign-in Use Case

Use Case Number	UC-2
Use Case Name	UC-Sign-in
Goal	The user wants to sign in to the system
Primary actor	Admin, Doctor, Patient
Level	User, System
Precondition	**PRE: 1** The user must have valid login credentials/exist in the system **PRE: 2** The user must have opened the home page of the system.
Success end	The user successfully logs in to the system.
Failure end condition	The user enters invalid credentials.
Main success scenario	**1.** The user clicks on the "Login" button in the right top corner. **2.** The system redirects the user to the login page. **3.** The user enters his credentials and clicks the "Login" button. **4.** The system verifies their credentials from the database and redirects the user to their respective dashboard.
Extensions (error scenarios)	**4a.** If the user enters invalid credentials. **4a.1.** The system shows a pop-up error message. **4a.2.** The user tries again with valid credentials.

2.3.3 Forget Password Use Case

Table 2.3: Forget Password Use Case

Use Case Number	UC-3
Use Case Name	UC-Forget-Password
Goal	The user wants to reset their password
Primary actor	Doctor, Patient
Level	User, System
Precondition	**PRE: 1** The user account must exist in the system **PRE: 2** The user must have opened the login page
Success end	The user successfully resets their password.
Failure end condition	**1.** The user enters an invalid email address. **2.** The user enters an invalid email verification code. **3.** The user enters invalid new credentials.
Main success scenario	**1.** The user clicks on the “Forgot Password” link under the "Login" button on the login page. **2.** The system redirects the user to the forgot password page. **3.** The user enters his credentials and clicks on the “Get Verification Code” button. **4.** The system validates the credentials and redirects the user to the email verification page. **5.** The user enters their email verification code and clicks on the “Verify Email” button. **6.** The system verifies the email verification code and redirects the user to the reset password page. **7.** The user enters their new credentials and clicks on the “Update Password” button. **8.** The system updates the credentials and redirects the user to the login page.

Extensions (error scenarios)	**4a.** If the user's entered email address does not exist in the system. **4a.1.** The system shows a popup error message. **4a.2.** The user tries again with a valid email address. **6a.** If the user enters an incorrect email verification code. **6a.1.** The system shows a popup error message. **6a.2.** The user tries again with the correct email verification code. **8a.** If the user enters invalid new credentials. **8a.1.** The system shows a popup error message. **8a.2.** The user tries again to validate new credentials.

2.3.4 Update Profile Use Case

Table 2.4: Update Profile Use Case

Use Case Number	UC-4
Use Case Name	UC-Update-Profile
Goal	The user wants to update their profile details
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates their profile details.
Failure end condition	The user enters invalid credentials.
Main success scenario	**1.** The user clicks on the “Profile” button in the side panel. **2.** The system displays the user's profile. **3.** The user updates the profile details and clicks on the "Update" button. **4.** The system updates the credentials in the database and shows a popup confirmation message. **5.** The user clicks on the “Confirm Update” button. **6.** The system updates the profile and shows a pop-up success message.
Extensions (error scenarios)	**4a.** If the user enters invalid credentials. **4a.1.** The system shows a pop-up error message. **4a.2.** The user tries again with valid credentials.

2.3.5 Book Appointment Use Case

Table 2.5: Book Appointment Use Case

Use Case Number	UC-5
Use Case Name	UC-Book-Appointment
Goal	The patient wants to book an appointment.
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully books an appointment.
Failure end condition	**1.** The user already has one active appointment. **2.** The selected doctor has no empty appointment slot. **3.** The user got an appointment slot clash with another patient's appointment slot.
Main success scenario	**1.** The user clicks on the “Book an Appointment” button in the side panel. **2.** The system displays the list of all available doctors in the system. **3.** The user clicks on a doctor's profile according to their choice. **4.** The system displays the profile of the selected doctor. **5.** The user clicks on the "Book Appointment" button on the doctor's profile page. **6.** The system displays the appointment booking form. **7.** The user fills out the form and clicks on the "Book Appointment Now" button. **8.** The system displays the payment page for the appointment. **9.** The user clicks on the "Proceed to Pay" button. **10.** The system redirects the user to the payment gateway to perform the payment. **11.** The user performs the payment. **12.** The payment gateway redirects the user to his appointment page and the system shows up a pop-up success message.

Extensions(error scenarios)	**6a.** If there are no empty slots for the selected doctor. **6a.1**. The system shows up a pop-up error message. **6a.2.** The user tries again by selecting another doctor. **8a.** If the patient got appointment slot clashes with another patient's appointment. **8a.1.** The system shows up a popup warning message for rescheduling the appointment. **8a.2.** The user can reschedule the appointment.

2.3.6 Reschedule Appointment Use Case

Table 2.6: Reschedule Appointment Use Case

Use Case Number	UC-6
Use Case Name	UC-Reschedule-Appointment
Goal	The patient wants to reschedule their appointment
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment rescheduling option must be enabled for the current active appointment **PRE: 4** The doctor must have empty appointment slots.
Success end	The user successfully reschedules their appointment.
Failure end condition	The appointment rescheduling option has been disabled.
Main success scenario	**1.** The user clicks on the “Your Appointments” button in the side panel. **2.** The system displays the user's appointments. **3.** The user clicks on the "More Details" button against the current active appointment. **4.** The system shows all the details of the corresponding appointment. **5.** The user slots a new appointment slot and clicks on the "Reschedule Appointment" button. **6.** The system shows a popup confirmation message. **7.**The user clicks on the “Reschedule” button. **8.** The system reschedules the appointment and shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the rescheduling option has been disabled for the current active appointment. **6a.1.** The system shows all the details of the appointment except for the rescheduling option. **6a.2.** The user cannot reschedule the appointment.

2.3.7 Cancel Appointment Use Case

Table 2.7: Cancel Appointment Use Case

Use Case Number	UC-7
Use Case Name	UC-Cancel-Appointment
Goal	The patient wants to cancel their appointment
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment cancellation option must be enabled for a current active appointment.
Success end	The user successfully cancels their appointment.
Failure end condition	The appointment cancellation option has been disabled.
Main success scenario	**1.** The user clicks on the “Your Appointments” button in the side panel. **2.** The system displays the user's appointments. **3.** The user clicks on the "More Details" button against the current active appointment. **4.** The system shows all the details of the corresponding appointment. **5.** The user clicks on the "Cancel Appointment" button. **6.** The system shows a pop-up confirmation form. **7.** The user clicks on the "Cancel Appointment" button. **8.** The system cancels the appointment and shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the cancellation option has been disabled for the current active appointment. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user cannot cancel the appointment.

2.3.8 Submit Appointment Reviews Use Case

Table 2.8: Submit Appointment Reviews Use Case

Use Case Number	UC-8
Use Case Name	UC-Submit-Appointment-Reviews
Goal	The patient wants to submit their appointment reviews.
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment must have been completed. **PRE: 4** The appointment reviews must have not been already submitted.
Success end	The user successfully submits their appointment reviews.
Failure end condition	The submitted reviews do not satisfy the system's criteria.
Main success scenario	**1.** The user clicks on the “Your Appointments” button in the side panel. **2.** The system displays the user's appointments. **3.** The user clicks on the "More Details" button against the completed appointment. **4.** The system shows all the details (including the review form) of the corresponding appointment. **5.** The user fills out the review form and clicks on the "Submit Review" button. **6.** The system shows a pop-up confirmation form. **7.** The user clicks on the "Submit" button. **8.** The system submits the appointment reviews and shows a pop-up success message.

Extensions (error scenarios)	**6a.** If the submitted review does not satisfy the system's criteria. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user refills the review form while satisfying the system's criteria.

2.3.9 Submit Refundable Amount Withdrawal Request Use Case

Table 2.9: Submit Refundable Amount Withdrawal Request Use Case

Use Case Number	UC-9
Use Case Name	UC-Submit-Refundable-Amount-Withdrawal-Request
Goal	The patient wants to submit a withdrawal request against the refundable amount.
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment must have a refundable amount in the system. **PRE: 4** The user must not have any active refundable amount withdrawal request.
Success end	The user successfully submits an amount withdrawal request.
Failure end condition	The user already has submitted one withdrawal request that is currently active.
Main success scenario	**1.** The user clicks on the "Withdraw Refundable Amount" button in the side panel. **2.** The system displays the withdrawal request form. **3.** The user fills out the form and clicks on the "Submit Withdrawal Request" button. **4.** The system submits the withdrawal request and shows up a pop-up success message.
Extensions (error scenarios)	**2a.** If the refundable amount withdrawal request has already been submitted and is currently active. **2a.1.** The system displays the already submitted withdrawal request instead of the new withdrawal request form. **2a.2.** The user cannot submit a new withdrawal request until the other one has been completed or rejected.

2.3.10 Detect Brain Tumor Use Case

Table 2.10: Detect Brain Tumor Use Case

Use Case Number	UC-10
Use Case Name	UC-Detect-Brain-Tumor
Goal	The patient wants to detect a brain tumor in a brain MRI image.
Primary actor	Patient
Level	User, System
Precondition	The user must have opened the home page in the system.
Success end	The user successfully gets the detection results from the uploaded brain MRI image.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the "Brain Tumor" link in the navigation bar. **2.** The system redirects the user to the brain tumor detection page. **3.** The user selects the brain MRI image. **4.** The system automatically uploads the image and shows the results under the image selector field. The system also shows up a pop-up success message on the server response.
Extensions (error scenarios)	N/A

2.3.11 Detect Disease Use Case

Table 2.11: Detect Disease Use Case

Use Case Number	UC-11
Use Case Name	UC-Detect-Disease
Goal	The patient wants to detect their potential disease by providing their symptoms.
Primary actor	Patient
Level	User, System
Precondition	The user must have opened the home page in the system.
Success end	The user successfully gets the predicted disease name based on the provided symptoms.
Failure end condition	The user has not selected at least 3 symptoms.
Main success scenario	**1.** The user clicks on the “Disease Diagnosis” button in the navigation panel. **2.** The system redirects the user to the symptoms selection page. **3.** The user selects their symptoms and clicks on the "Predict Disease" button. **4.** The system detects the potential disease and redirects the user to the predicted disease results page showing the disease name and short description.
Extensions (error scenarios)	**4a.** If the user has not selected at least 3 symptoms. **4a.**1. The system shows a pop-up warning message. **4a.2.** The user selects at least 3 symptoms and tries again.

2.3.12 Get Doctor Suggestion Use Case

Table 2.12: Get Doctor Suggestion Use Case

Use Case Number	UC-12
Use Case Name	UC-Get-Doctor-Suggestion
Goal	The patient wants to get a doctor's suggestion based on their results from either a brain tumor detection model or disease prediction model.
Primary actor	Patient
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have run the brain tumor detection or disease prediction models and the system must have detected any disease. **PRE: 3** Disease must have been mapped to any doctor specialization category.
Success end	The user successfully gets the list of suggested doctors.
Failure end condition	The user is not logged in to the system.
Main success scenario	**1.** The user clicks on the "Book Specific Disease Specialist" button in the bottom right corner of the screen. **2.** The system redirects the user to the suggested doctor page showing the list of suggested doctors.
Extensions (error scenarios)	**2a.** If the user is not logged in the system. **2a.1.** The system redirects the user to the login page and shows a pop-up information message. **2a.2.** The user logs in to the system. **2a.3.** The system redirects the user to the suggested doctor page showing the list of suggested doctors. **2a.4.** The user can proceed to book an appointment with any of the suggested doctors.

2.3.13 Update Profile Use Case

Table 2.13: Update Profile Use Case

Use Case Number	UC-13
Use Case Name	UC-Update-Profile
Goal	The user wants to update their profile details
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates their profile details.
Failure end condition	The user enters invalid credentials.
Main success scenario	**1.** The user clicks on the “Profile” button in the side panel. **2.** The system displays the user's profile. **3.** The user updates the profile details and clicks on the "Update" button. **4.** The system updates the credentials in the database and shows a popup confirmation message. **5.** The user clicks on the “Confirm Update” button. **6.** The system updates the profile and shows a pop-up success message.
Extensions (error scenarios)	**4a.** If the user enters invalid credentials. **4a.1.** The system shows a pop-up error message. **4a.2.** The user tries again with valid credentials.

2.3.14 Submit Account Approval Request Use Case

Table 2.14: Submit Account Approval Request Use Case

Use Case Number	UC-14
Use Case Name	UC-Submit-Account-Approval-Request
Goal	The doctor wants to send an account approval request to the administration.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must not have his/her account already approved.
Success end	The user successfully sends the account approval request.
Failure end condition	The user does not provide a minimum number of documents or provides invalid document formats.
Main success scenario	**1.** The user clicks on the “Profile” button in the side panel of the dashboard. **2.** The system displays the profile of the user. **3.** The user clicks on the "Submit Account Approval Request" button. **4.** The system shows a pop-up submission form. **5.** The user selects their documents and clicks on the "Submit Request" button or "Submit New Request". **6.** The system deletes the previous account approval request (if any) and submits a new request. The system also shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the user does not provide the minimum number of documents or provides invalid document formats. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid and at least a minimum number of documents.

2.3.15 Download Uploaded Documents Use Case

Table 2.15: Download Uploaded Documents Use Case

Use Case Number	UC-15
Use Case Name	UC-Download-Uploaded-Documents
Goal	The doctor wants to download his/her uploaded documents for account approval requests.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must have submitted the account approval request.
Success end	The user successfully downloads their uploaded documents.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Profile” button in the side panel of the dashboard. **2.** The system displays the profile of the user. **3.** The user clicks on the question mark icon next to the account status. **4.** The system shows a pop-up account status information box. **5.** The user clicks on the "Download Your Documents" button. **6.** The system starts downloading the documents on the user's device and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.16 Get Patient's WhatsApp Number Use Case

Table 2.16: Get Patient's WhatsApp Number Use Case

Use Case Number	UC-16
Use Case Name	UC-Get-Patients-WhatsApp-Number
Goal	The doctor wants to get the WhatsApp contact number of the patient for a meeting.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must have an appointment with the patient.
Success end	The user successfully gets the WhatsApp contact number of the patient for a meeting.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the "Pending Appointments" button in the side panel of the dashboard. **2.** The system displays all the pending appointments of the user. **3.** The user copies or clicks on the WhatsApp number against a specific appointment. **4.** The system redirects the user to the WhatsApp web or App in case of clicking on the contact number.
Extensions(error scenarios)	N/A

2.3.17 Mark Appointment as Completed Use Case

Table 2.17: Mark Appointment as Completed Use Case

Use Case Number	UC-17
Use Case Name	UC-Mark-Appointment-Completed
Goal	The doctor wants to mark the appointment as completed.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The appointment must not already be completed.
Success end	The user successfully marks the appointment as completed.
Failure end condition	**1.** The user does not provide an appointment report. **2.** The user does not provide the correct appointment secret code.
Main success scenario	**1.** The user clicks on the “Pending Appointment” button in the side panel of the dashboard. **2.** The system displays all the pending appointments of the user. **3.** The user clicks on the "More Details" button against a specific appointment. **4.** The system displays all the actions that can be performed on the selected appointment. **5.** The user clicks on the "Mark as Completed" button. **6.** The system shows a pop-up appointment report submission form. **7.** The user fills out the form and clicks on the "Submit" button. **8.** The system submits the report and marks the appointment as completed. The system also shows a pop-up success message.
Extensions (error scenarios)	**8a.** If the user does not provide an appointment report. **8a.1.** The system shows a pop-up warning message. **8a.2.** The user tries again by providing the appointment report. **8b.** If the user does not provide the correct appointment secret code.

	8b.1. The system shows a pop-up error message. **8b.2.** The user tries again by providing the correct appointment secret code.

2.3.18 Delay Appointment Use Case

Table 2.18: Delay Appointment Use Case

Use Case Number	UC-18
Use Case Name	UC-Delay-Appointment
Goal	The doctor wants to delay the appointment.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must not have already submitted the delay request against the selected appointment.
Success end	The user successfully delays the appointment.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Pending Appointment” button in the side panel of the dashboard. **2.** The system displays all the pending appointments of the user. **3.** The user clicks on the "More Details" button against a specific appointment. **4.** The system displays all the actions that can be performed on the selected appointment. **5.** The user clicks on the "Ask to Delay" button. **6.** The system shows a pop-up appointment delay confirmation form. **7.** The user clicks on the "Delay Appointment" button. **8.** The system submits an appointment delay request and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.19 Mark Doctor's Absence Use Case

Table 2.19: Mark Doctor's Absence Use Case

Use Case Number	UC-19
Use Case Name	UC-Mark-Doctors-Absence
Goal	The doctor wants to mark his/her absence in the appointment meeting.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must not have already marked his/her own or patient's absence.
Success end	The user successfully marks their absence in the appointment meeting.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Pending Appointment” button in the side panel of the dashboard. **2.** The system displays all the pending appointments of the user. **3.** The user clicks on the "More Details" button against a specific appointment. **4.** The system displays all the actions that can be performed on the selected appointment. **5.** The user clicks on the "I Could Not Join" button. **6.** The system shows a pop-up doctor's absence confirmation form. **7.** The user clicks on the "Confirm My Absence" button. **8.** The system submits the user's absence and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.20 Mark Patient's Absence Use Case

Table 2.20: Mark Patient's Absence Use Case

Use Case Number	UC-20
Use Case Name	UC-Mark-Patients-Absence
Goal	The doctor wants to mark the patient's absence request in the appointment meeting.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must have been logged in to the system. **PRE: 2** The user must have opened the dashboard. **PRE: 3** The user must not have already marked his/her own or patient's absence.
Success end	The user successfully marks the patient's absence in the appointment meeting.
Failure end condition	The user does not provide appointment meetings or attempt screen recording videos.
Main success scenario	**1.** The user clicks on the “Pending Appointment” button in the side panel of the dashboard. **2.** The system displays all the pending appointments of the user. **3.** The user clicks on the "More Details" button against a specific appointment. **4.** The system displays all the actions that can be performed on the selected appointment. **5.** The user clicks on the "Patient Didn't Join" button. **6.** The system shows a pop-up patient's absence confirmation form. **7.** The user fills out the form and clicks on the "Submit Request" button. **8.** The system submits the request and shows a pop-up success message.
Extensions (error scenarios)	**8a.** If the user does not fill the required form fields. **8a.1.** The system shows a pop-up warning message. **8a.2.** The user tries again by filling in the required fields.

2.3.21 Cancel Appointment Use Case

Table 2.21: Cancel Appointment Use Case

Use Case Number	UC-21
Use Case Name	UC-Cancel-Appointment
Goal	The doctor wants to cancel their appointment
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment must not have already been canceled.
Success end	The user successfully cancels their appointment.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Pending Appointments” button in the side panel. **2.** The system displays the user's appointments. **3.** The user clicks on the "More Details" button against the current active appointment. **4.** The system displays all the actions that can be performed on the selected appointment. **5.** The user clicks on the "Cancel Appointment" button. **6.** The system shows up a pop-up confirmation form. **7.** The user clicks on the "Cancel Appointment" button. **8.** The system cancels the appointment and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.22 Submit Amount Withdrawal Request Use Case

Table 2.22: Submit Amount Withdrawal Request Use Case

Use Case Number	UC-22
Use Case Name	UC-Submit-Amount-Withdrawal-Request
Goal	The doctor wants to submit an amount withdrawal request.
Primary actor	Doctor
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard. **PRE: 3** The appointment must have a minimum of 5000 amount in the wallet. **PRE: 4** The user must not already have any withdrawal request active.
Success end	The user successfully submits an amount withdrawal request.
Failure end condition	The user's filled form does not satisfy the withdrawal request criteria.
Main success scenario	**1.** The user clicks on the “Withdraw Amount” button in the side panel. **2.** The system displays the withdrawal request form. **3.** The user fills out the form and clicks on the "Submit Withdrawal Request" button. **4.** The system submits the withdrawal request and shows a pop-up success message.
Extensions (error scenarios)	**4a.** If the user's filled form does not satisfy the withdrawal request criteria. **4a.1.** The system displays a pop-up warning message. **4a.2.** The user tries again by providing valid information.

2.3.23 Update Patient's Details Use Case

Table 2.23: Update Patient's Details Use Case

Use Case Number	UC-23
Use Case Name	UC-Update-Patients-Details
Goal	The admin wants to update a patient's details.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	If the user provides invalid new details.
Main success scenario	**1.** The user clicks on the “Patients” button in the side panel. **2.** The system displays all the patients in the system. **3.** The user clicks on the row against a specific patient in the data table. **4.** The system displays all the possible actions that can be performed on the patient. **5.** The user updates the details of the patient and clicks on the "Update" button. **6.** The system updates the details of the patient and shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the user enters invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details.

2.3.24 Update Doctor's Details Use Case

Table 2.24: Update Doctor's Details Use Case

Use Case Number	UC-24
Use Case Name	UC-Update-Doctors-Details
Goal	The admin wants to update a doctor's details.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	If the user provides invalid new details.
Main success scenario	**1.** The user clicks on the “Patients” button in the side panel. **2.** The system displays all the patients in the system. **3.** The user clicks on the row against a specific patient in the data table. **4.** The system displays all the possible actions that can be performed on the patient. **5.** The user updates the details of the patient and clicks on the "Update" button. **6.** The system updates the details of the patient and shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the user enters invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details.

2.3.25 Explore Associated Records Use Case

Table 2.25: Explore Associated Records Use Case

Use Case Number	UC-25
Use Case Name	UC-Explore-Associated-Records
Goal	The admin wants to explore/view associated records.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened the dashboard.
Success end	The user successfully explores associated data/records.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on any button/entity in the side panel. **2.** The system displays all the records of that entity. **3.** The user clicks on a row against a specific record in the data table. **4.** The system displays all the possible actions that can be performed on the selected record. **5.** The user clicks on any entity button to explore its related data. **6.** The system displays a new page showing all the records related to the selected record of the previous selection.
Extensions (error scenarios)	N/A

2.3.26 Accept/Reject Doctor Approval Request Use Case

Table 2.26: Accept/Reject Doctor Approval Request Use Case

Use Case Number	UC-26
Use Case Name	UC-Accept/Reject-Doctor-Approval-Request
Goal	The admin wants to accept or reject a doctor's account approval request.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their respective dashboard.
Success end	The user successfully accepts or rejects the doctor's account approval request.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Pending Doctor Actions” button in the side panel. **2.** The system displays all the account approval requests by doctors. **3.** The user clicks on a row against a specific doctor's details in the data table. **4.** The system displays all the possible actions that can be performed on the selected account approval request. **5.** The user selects a response for the request and clicks on the "Update" button. **6.** The system submits the response and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.27 Approve/Reject Patient's Absence Request Use Case

Table 2.27: Approve/Reject Patient's Absence Request Use Case

Use Case Number	UC-27
Use Case Name	UC-Approve/Reject-Patients-Absence-Request
Goal	The admin wants to approve or reject a patient's absence request submitted by a doctor.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully approves or rejects the patient's absence request.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Pending Appointment Actions” button in the side panel. **2.** The system displays all the requests in the system. **3.** The user clicks on a row against a specific request in the data table. **4.** The system displays all the possible actions that can be performed on the selected request. **5.** The user clicks on the "Approve" or "Reject" button. **6.** The system updates the request's status accordingly and shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.28 Accept Doctor's Withdrawal Request Use Case

Table 2.28: Accept Doctor's Withdrawal Request Use Case

Use Case Number	UC-28
Use Case Name	UC-Accept-Doctor's-Withdrawal-Request
Goal	The admin wants to accept the doctor's wallet money withdrawal request.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	If the user provides invalid transaction details.
Main success scenario	**1.** The user clicks on the "Doctor Withdrawal Requests" button in the side panel. **2.** The system displays all the pending money withdrawal requests by doctors. **3.** The user clicks on the row against a specific withdrawal request in the data table. **4.** The system displays all the possible actions that can be performed on the selected request. **5.** The user clicks on the "Mark as Completed" button. **6.** The system displays the transaction details form. **7.** The user fills out the form and clicks on the "Mark as Completed" button. **8.** The system displays a request completion confirmation form. **9.** The user clicks on the "Mark as Completed" button. **10.** The system updates the request details and marks it as completed. The system also shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the user enters invalid transaction details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid transaction details.

2.3.29 Reject Doctor's Withdrawal Request Use Case

Table 2.29: Reject Doctor's Withdrawal Request Use Case

Use Case Number	UC-29
Use Case Name	UC-Reject-Doctor's-Withdrawal-Request
Goal	The admin wants to reject the doctor's wallet money withdrawal request.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the "Doctor Withdrawal Requests" button in the side panel. **2.** The system displays all the pending money withdrawal requests by doctors. **3.** The user clicks on the row against a specific withdrawal request in the data table. **4.** The system displays all the possible actions that can be performed on the selected request. **5.** The user clicks on the "Reject" button. **6.** The system displays the request rejection form. **7.** The user fills out the form and clicks on the "Reject" button. **8.** The system shows a request rejection confirmation form. **9.** The user clicks on the "Reject Withdrawal Request" button. **10.** The system updates the request details and marks it as rejected. The system also shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.30 Accept Patient's Withdrawal Request Use Case

Table 2.30: Accept Patient's Withdrawal Request Use Case

Use Case Number	UC-30
Use Case Name	UC-Accept-Patient's-Withdrawal-Request
Goal	The admin wants to accept the patient's refundable money withdrawal request.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	If the user provides invalid transaction details.
Main success scenario	**1.** The user clicks on the "Patient Withdrawal Requests" button in the side panel. **2.** The system displays all the pending money withdrawal requests by patients. **3.** The user clicks on the row against a specific withdrawal request in the data table. **4.** The system displays all the possible actions that can be performed on the selected request. **5.** The user clicks on the "Mark as Completed" button. **6.** The system displays the transaction details form. **7.** The user fills out the form and clicks on the "Mark as Completed" button. **8.** The system displays a request completion confirmation form. **9.** The user clicks on the "Mark as Completed" button. **10.** The system updates the request details and marks it as completed. The system also shows a pop-up success message.
Extensions (error scenarios)	**6a.** If the user enters invalid transaction details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid transaction details.

2.3.31 Reject Patient's Withdrawal Request Use Case

Table 2.31: Reject Patient's Withdrawal Request Use Case

Use Case Number	UC-31
Use Case Name	UC-Reject-Patient's-Withdrawal-Request
Goal	The admin wants to reject the patient's refundable money withdrawal request.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the patient's details.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Patient Withdrawal Requests” button in the side panel. **2.** The system displays all the pending money withdrawal requests by patients. **3.** The user clicks on the row against a specific withdrawal request in the data table. **4.** The system displays all the possible actions that can be performed on the selected request. **5.** The user clicks on the "Reject" button. **6.** The system displays the request rejection form. **7.** The user fills out the form and clicks on the "Reject" button. **8.** The system shows a request rejection confirmation form. **9.** The user clicks on the "Reject Withdrawal Request" button. **10.** The system updates the request details and marks it as rejected. The system also shows a pop-up success message.
Extensions (error scenarios)	N/A

2.3.32 Create Specialization Category Use Case

Table 2.32: Create Specialization Category Use Case

Use Case Number	UC-32
Use Case Name	UC-Create-Specialization-Category
Goal	The admin wants to create a new specialization category for doctors.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully creates a new specialization category.
Failure end condition	**1.** If the user provides invalid specialization category details. **2.** If the specialization category already exists in the system.
Main success scenario	**1.** The user clicks on the “Specialization Category” button in the side panel. **2.** The system displays doctor's specialization categories page. **3.** The user clicks on the "Create New" button. **4.** The system displays a pop-up form to create a new specialization category. **5.** The user fills the form and clicks on the "Create" button. **6.** The system creates the specialization category and displays a pop-up success message.
Extensions (error scenarios)	**6a.** If the user provides invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details. **6b.** If the specialization category already exists in the system. **6b.1.** The system shows a pop-up warning message. **6b.2.** The user tries again with different specialization category details.

2.3.33 Update Specialization Category Details Use Case

Table 2.33: Update Specialization Category Details Use Case

Use Case Number	UC-33
Use Case Name	UC-Update-Specialization-Category-Details
Goal	The admin wants to update the details of a specific specialization category.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the details of a selected specialization category
Failure end condition	**1.** If the user provides invalid details for the specialization category. **2.** If another specialization category with the same details already exists in the system.
Main success scenario	**1.** The user clicks on the “Specialization Category” button in the side panel. **2.** The system displays doctor's specialization categories page. **3.** The user clicks on a row against a specific specialization category in the data table. **4.** The system displays all the possible actions that can be performed on the selected specialization category. **5.** The user updates the details and clicks on the "Update" button. **6.** The system updates the specialization category details and displays a pop-up success message.
Extensions (error scenarios)	**6a.** If the user provides invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details. **6b.** If the specialization category with the same details already exists in the system. **6b.1.** The system shows a pop-up warning message. **6b.2.** The user tries again with different specialization category details.

2.3.34 Delete Specialization Category Use Case

Table 2.34: Delete Specialization Category Use Case

Use Case Number	UC-34
Use Case Name	UC-Delete-Specialization-Category
Goal	The admin wants to delete a specific specialization category.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully deletes the selected specialization category
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Specialization Category” button in the side panel. **2.** The system displays doctor's specialization categories page. **3.** The user clicks on a row against a specific specialization category in the data table. **4.** The system displays all the possible actions that can be performed on the selected specialization category. **5.** The user clicks on the "Delete" button. **6.** The system displays a pop-up confirmation box. **7.** The user clicks on the "Delete Specialization" button. **8.** The system deletes the specialization category details and displays a pop-up success message.
Extensions (error scenarios)	N/A

2.3.35 Create Language Use Case

Table 2.35: Create Language Use Case

Use Case Number	UC-35
Use Case Name	UC-Create-Language
Goal	The admin wants to create a new language.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully creates a new language.
Failure end condition	**1.** If the user provides invalid language details. **2.** If the language already exists in the system.
Main success scenario	**1.** The user clicks on the “Languages” button in the side panel. **2.** The system displays doctor's languages page. **3.** The user clicks on the "Create New" button. **4.** The system displays a pop-up form to create a new language. **5.** The user fills the form and clicks on the "Create" button. **6.** The system creates the language and displays a pop-up success message.
Extensions (error scenarios)	**6a.** If the user provides invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details. **6b.** If the language already exists in the system. **6b.1.** The system shows a pop-up warning message. **6b.2.** The user tries again with different language details.

2.3.36 Update Language Details Use Case

Table 2.36: Update Language Details Use Case

Use Case Number	UC-36
Use Case Name	UC-Update-Language-Details
Goal	The admin wants to update the details of a language category.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully updates the details of a selected language.
Failure end condition	**1.** If the user provides invalid details for language. **2.** If another language with the same details already exists in the system.
Main success scenario	**1.** The user clicks on the “Languages” button in the side panel. **2.** The system displays the doctor's language page. **3.** The user clicks on a row against a specific language in the data table. **4.** The system displays all the possible actions that can be performed in the selected language. **5.** The user updates the details and clicks on the "Update" button. **6.** The system updates the language details and displays a pop-up success message.
Extensions (error scenarios)	**6a.** If the user provides invalid details. **6a.1.** The system shows a pop-up warning message. **6a.2.** The user tries again with valid details. **6b.** If the language with the same details already exists in the system. **6b.1.** The system shows a pop-up warning message. **6b.2.** The user tries again with different language details.

2.3.37 Delete Language Use Case

Table 2.37: Delete Language Use Case

Use Case Number	UC-37
Use Case Name	UC-Delete-Language
Goal	The admin wants to delete a specific language.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully deletes the selected language.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Languages” button in the side panel. **2.** The system displays the doctor's language page. **3.** The user clicks on a row against a specific language in the data table. **4.** The system displays all the possible actions that can be performed in the selected language. **5.** The user clicks on the "Delete Language" button. **6.** The system displays a pop-up confirmation box. **7.** The user clicks on the "Delete Language" button. **8.** The system deletes the language details and displays a pop-up success message.
Extensions (error scenarios)	N/A

2.3.38 Create Disease and Specialization Mapping Use Case

Table 2.38: Create Disease and Specialization Mapping Use Case

Use Case Number	UC-38
Use Case Name	UC-Create-Disease-And-Specialization-Mapping
Goal	The admin wants to create a new disease and specialization mapping.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully creates a new disease and specialization mapping.
Failure end condition	If all the diseases have already been mapped to any specialization category.
Main success scenario	**1.** The user clicks on the “Disease Prediction Mapping” button in the side panel. **2.** The system displays all the diseases in the specialization mappings list. **3.** The user clicks on the "Create New" button. **4.** The system displays a pop-up form to create a new mapping. **5.** The user fills out the form and clicks on the "Create" button. **6.** The system creates the mapping and displays a pop-up success message.
Extensions (error scenarios)	**4a.** If all the diseases have already been mapped to any specialization category. **4a.1.** The system shows a pop-up warning message. **4a.2.** The user cannot create new mapping until he deletes one first.

2.3.39 Delete Disease and Specialization Mapping Use Case

Table 2.39: Delete Disease and Specialization Mapping Use Case

Use Case Number	UC-39
Use Case Name	UC-Delete-Disease-And-Specialization-Mapping
Goal	The admin wants to delete a specific disease to specialization mapping.
Primary actor	Admin
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully deletes the selected disease and specialization mapping.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the “Disease Prediction Mapping” button in the side panel. **2.** The system displays all the disease-to-specialization mappings. **3.** The user clicks on a row against a specific mapping in the data table. **4.** The system displays all the possible actions that can be performed on the selected mapping. **5.** The user clicks on the "Delete" button. **6.** The system displays a pop-up confirmation box. **7.** The user clicks on the "Delete Mapping" button. **8.** The system deletes the selected mapping and displays a pop-up success message.
Extensions (error scenarios)	N/A

2.3.40 Sign-out Use Case

Table 2.40: Sign-out Use Case

Use Case Number	UC-40
Use Case Name	UC-Sign-out
Goal	The user wants to log in from the system.
Primary actor	Admin, Patient, Doctor
Level	User, System
Precondition	**PRE: 1** The user must be logged in to the system. **PRE: 2** The user must have opened their dashboard.
Success end	The user successfully logs out of the system.
Failure end condition	N/A
Main success scenario	**1.** The user clicks on the "Logout" icon in the bottom left corner of the side panel of the dashboard. **2.** The system displays a pop-up confirmation box. **3.** The user clicks on the “Confirm Logout” button. **4.** The system logs out and redirects the user to the login page.
Extension (error scenarios)	N/A

Chapter 3

Design

Chapter 3 Design

This chapter is all about object-oriented modeling and software design. In the previous chapter, an analysis of the system is completed. So, we understand the current situation of the problem domain. Now we are ready to strive for a solution for the problem domain by using an object-oriented approach. The following artifacts must be included in this deliverable

3.1 UML Diagrams

There are different UML diagrams are added to this project and are as follows:

1. Use a cases Diagram
2. Class diagram
3. Sequence Diagram
4. Data Model i.e. ERD

3.2 Use Case Diagram

3.2.1 Use-Case Diagram for Admin

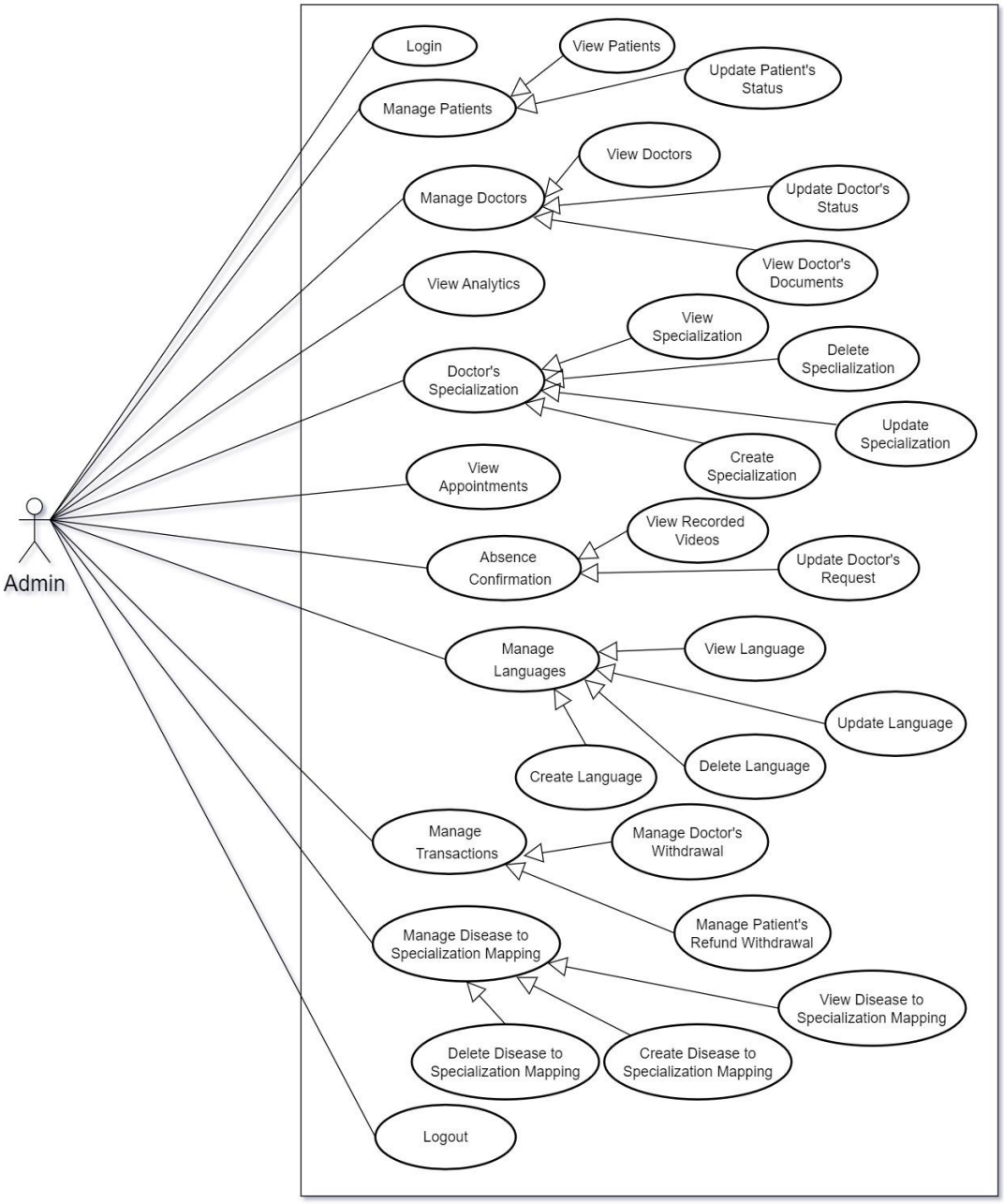

Figure 3.1:Use-Case Diagram for Admin

3.2.2 Use-Case Diagram for Patient

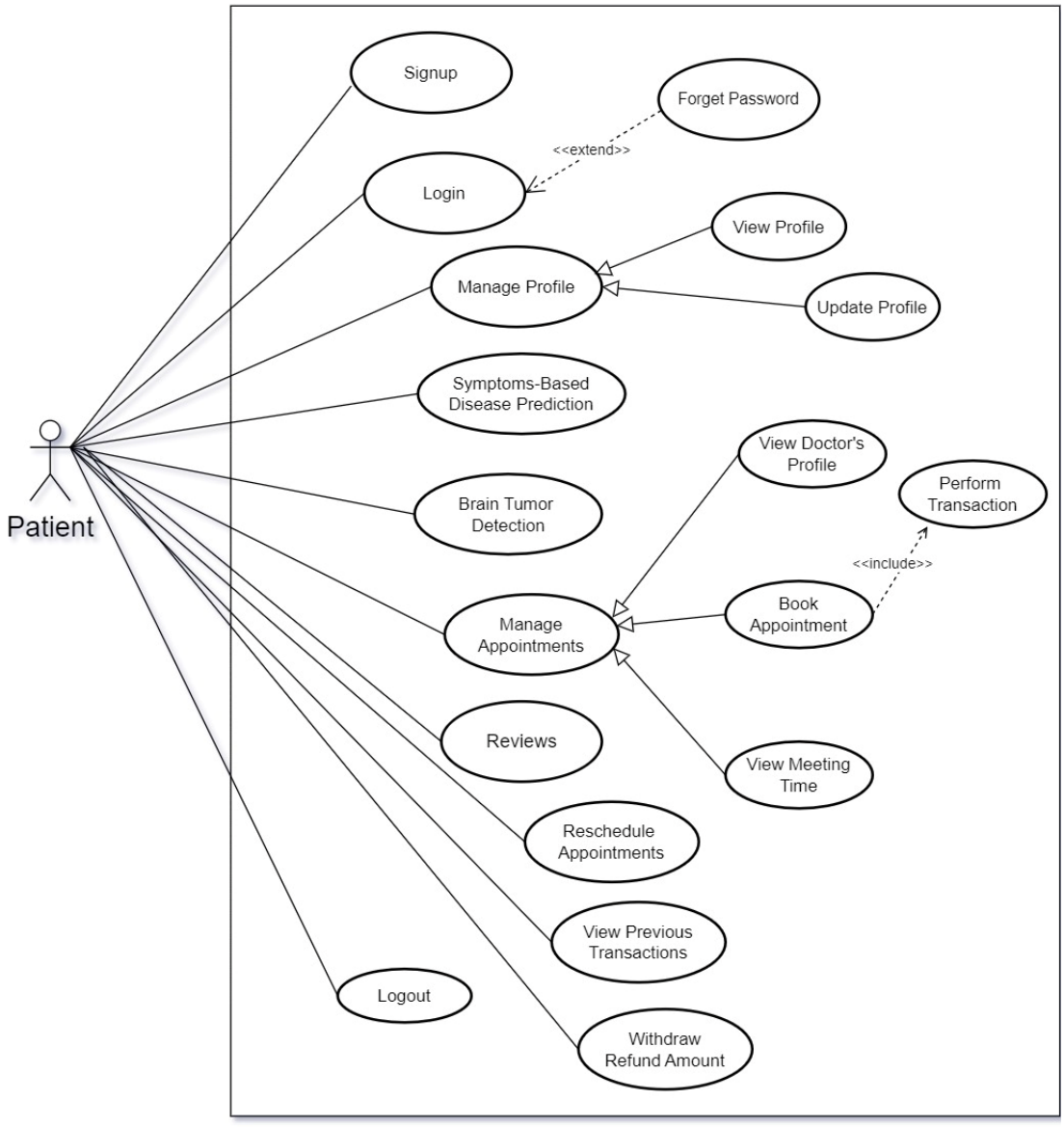

Figure 3.2: Use-Case Diagram for Patient

3.2.3 Use-Case Diagram for Doctor

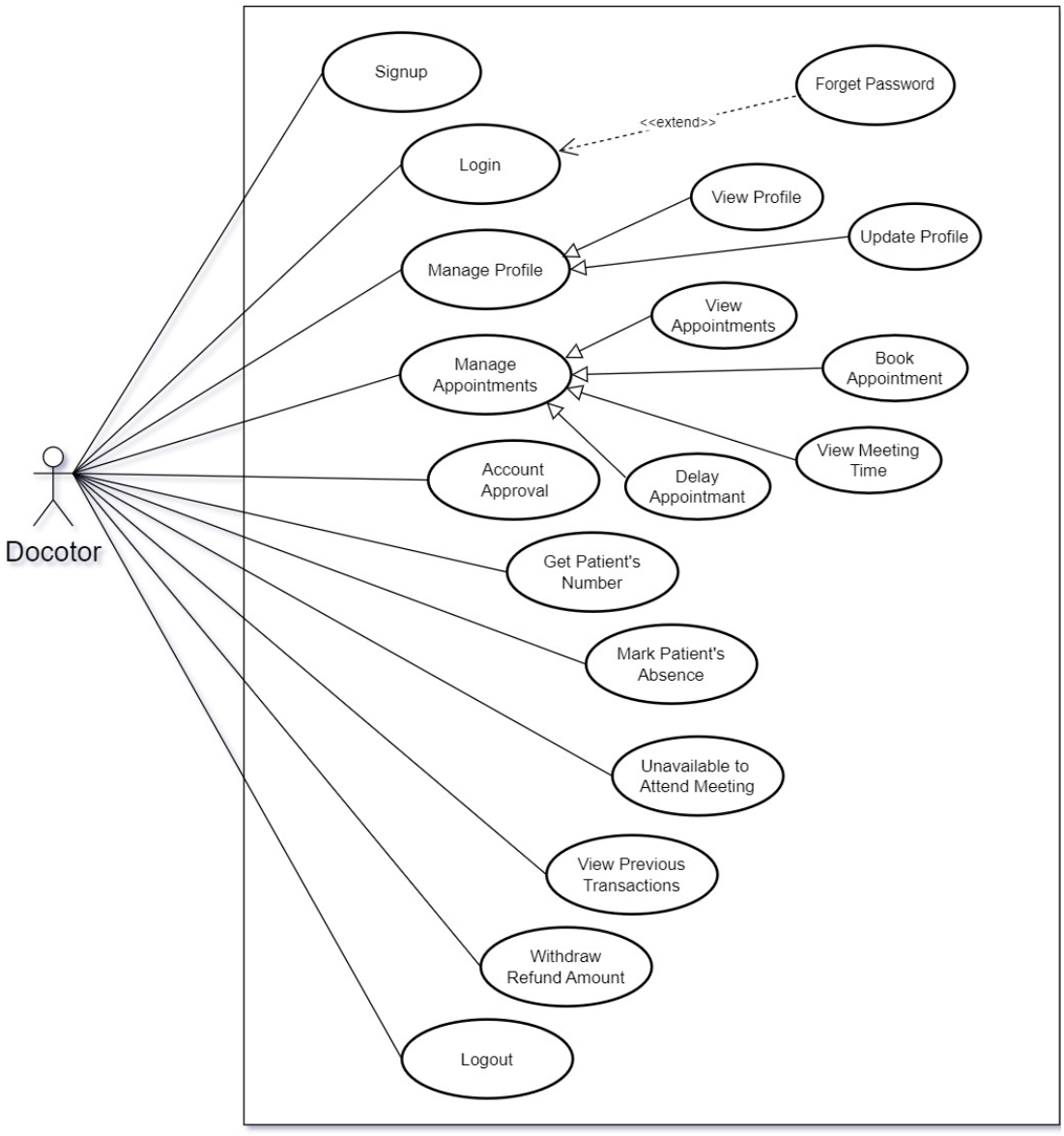

Figure 3.3: Use-Case Diagram for Doctor

3.3 Class Diagram

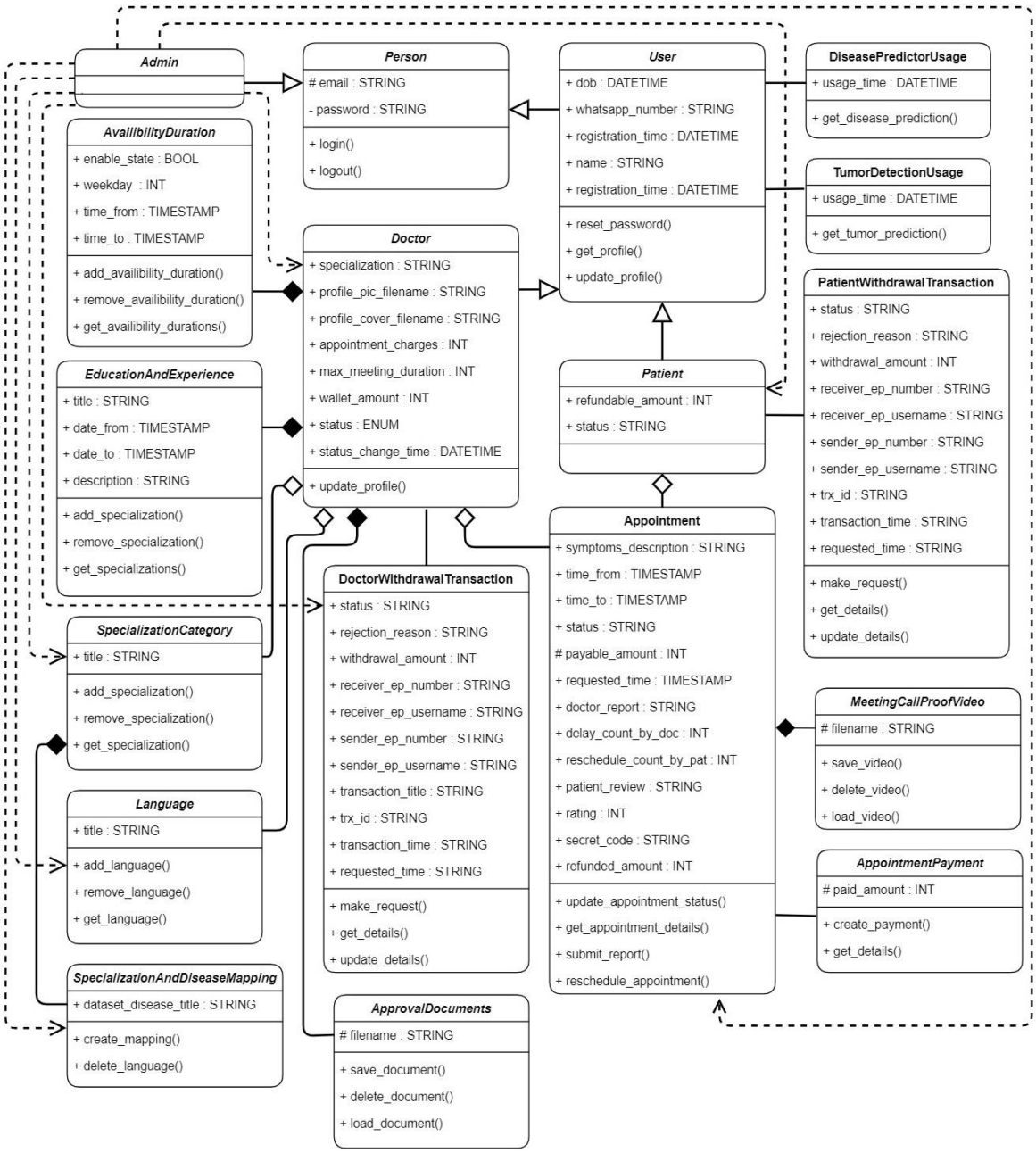

Figure 3.4: Class Diagram

3.4 Sequence Diagram

3.4.1 Sign Up

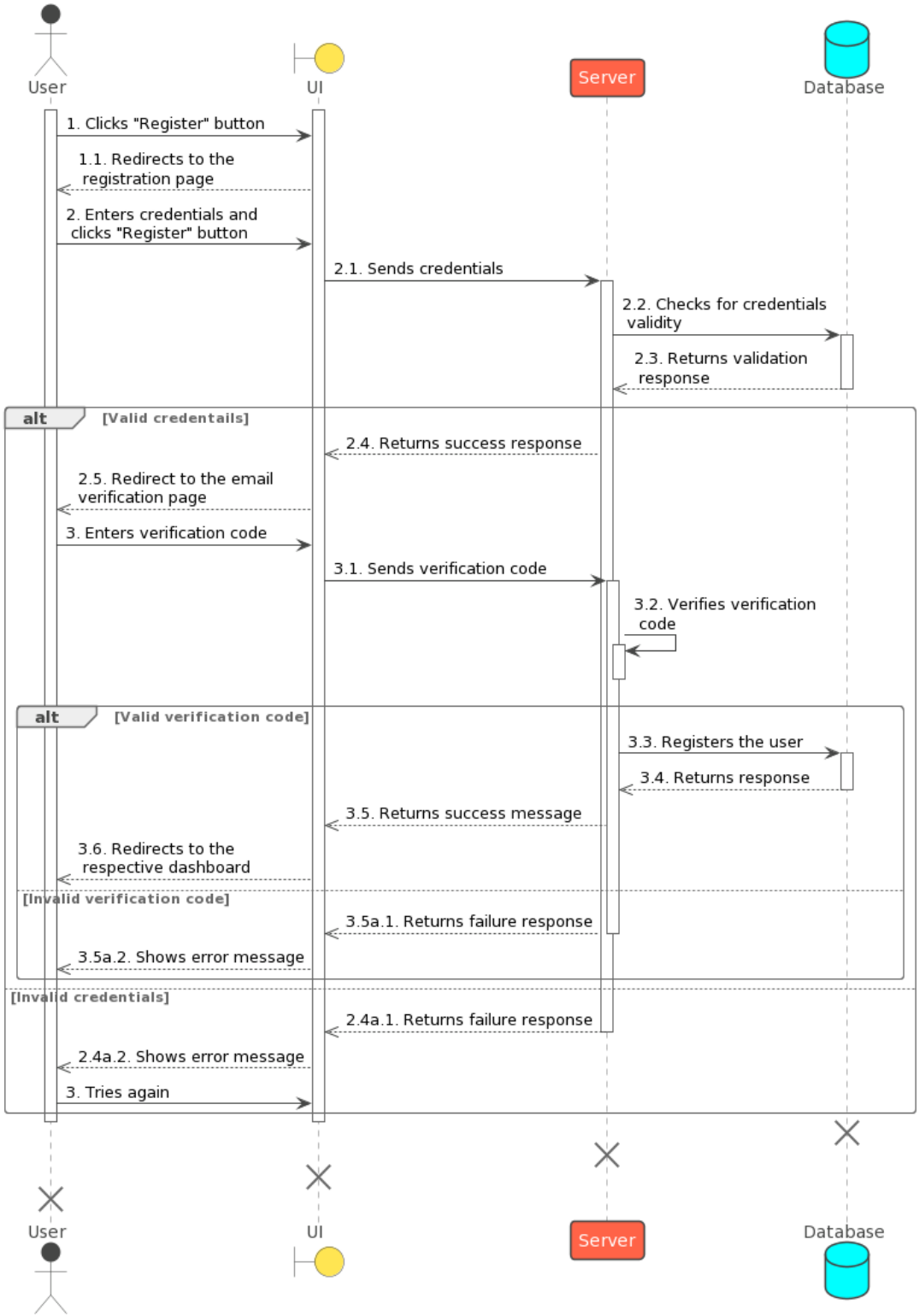

Figure 3.5: Sequence Diagram for Signup

3.4.2 Sign-in

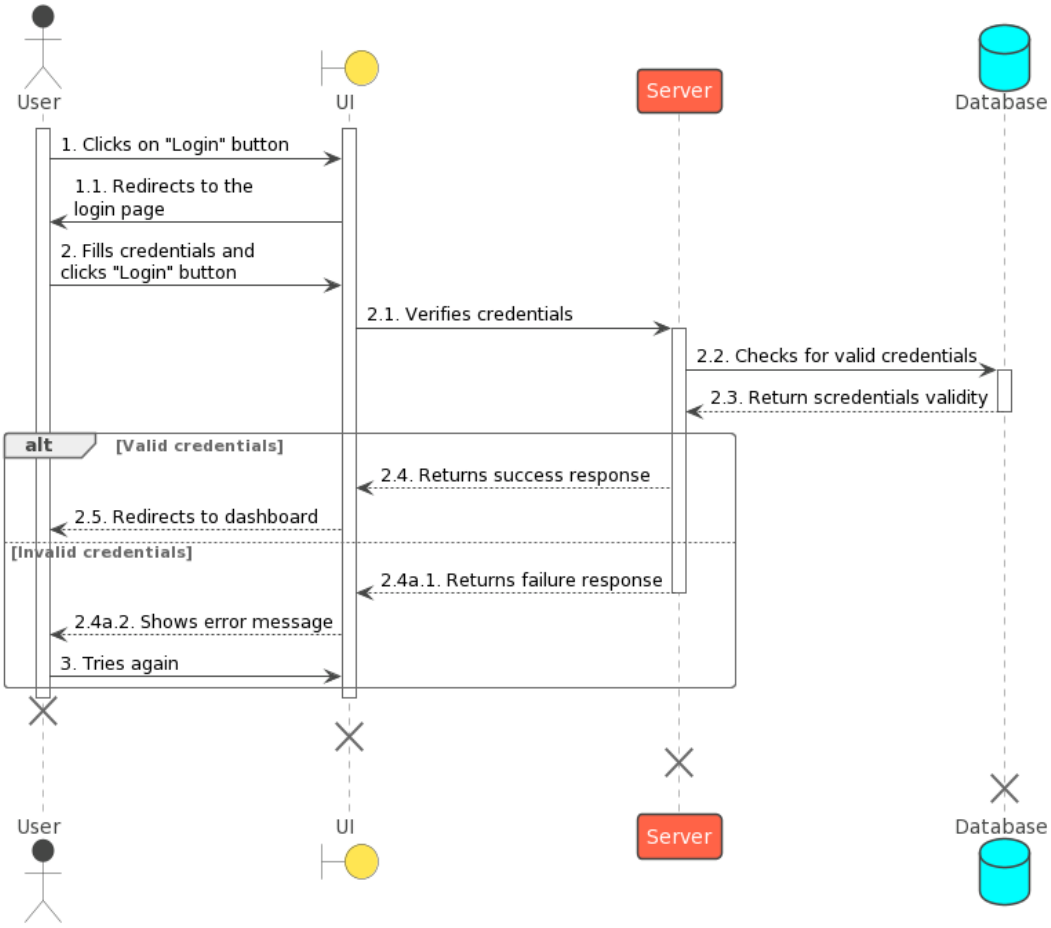

Figure 3.6: Sequence Diagram for Sign-in

3.4.3 Forget Password

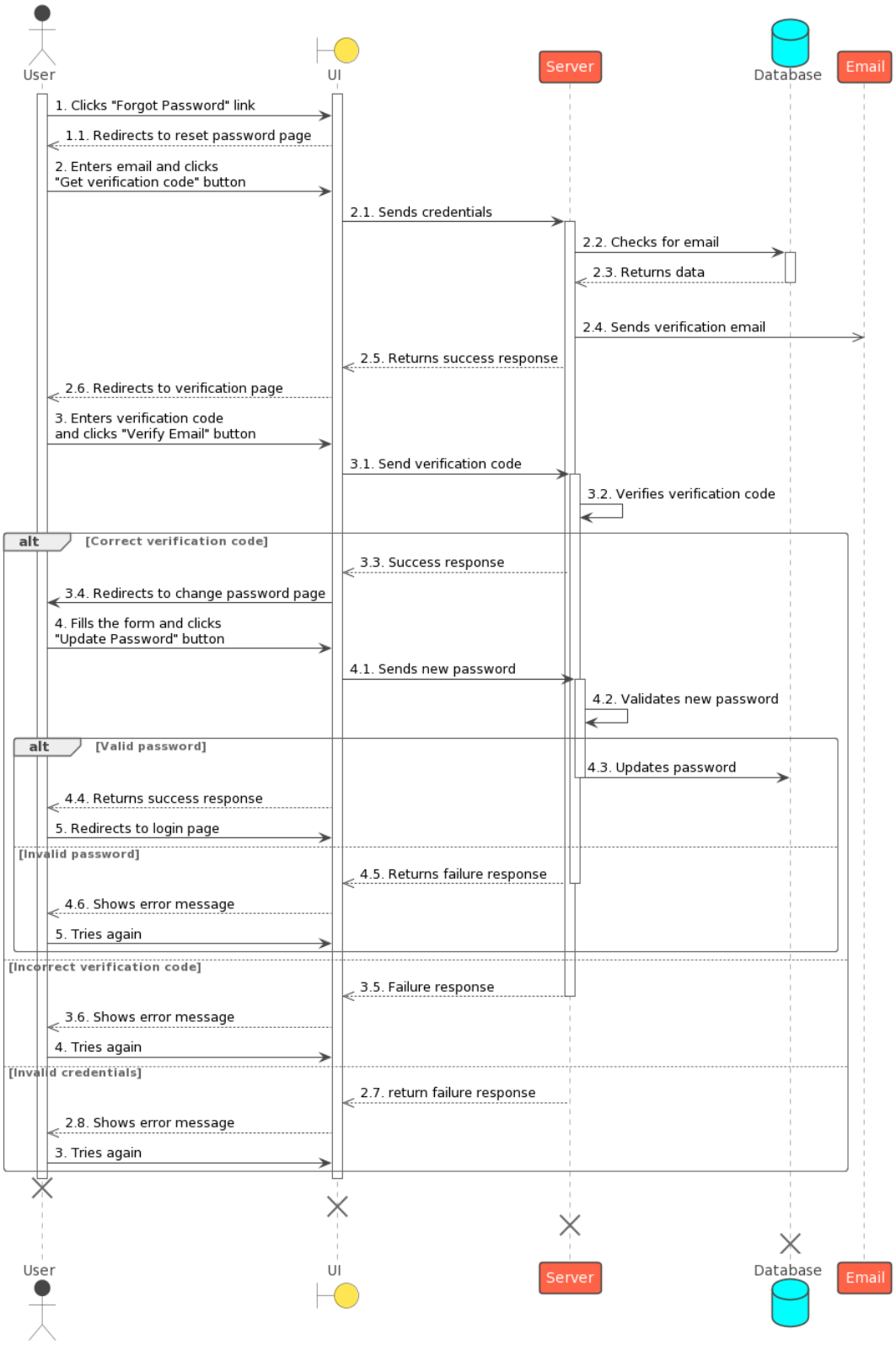

Figure 3.7: Sequence Diagram for Forget Password

3.4.4 Update Patient's Profile

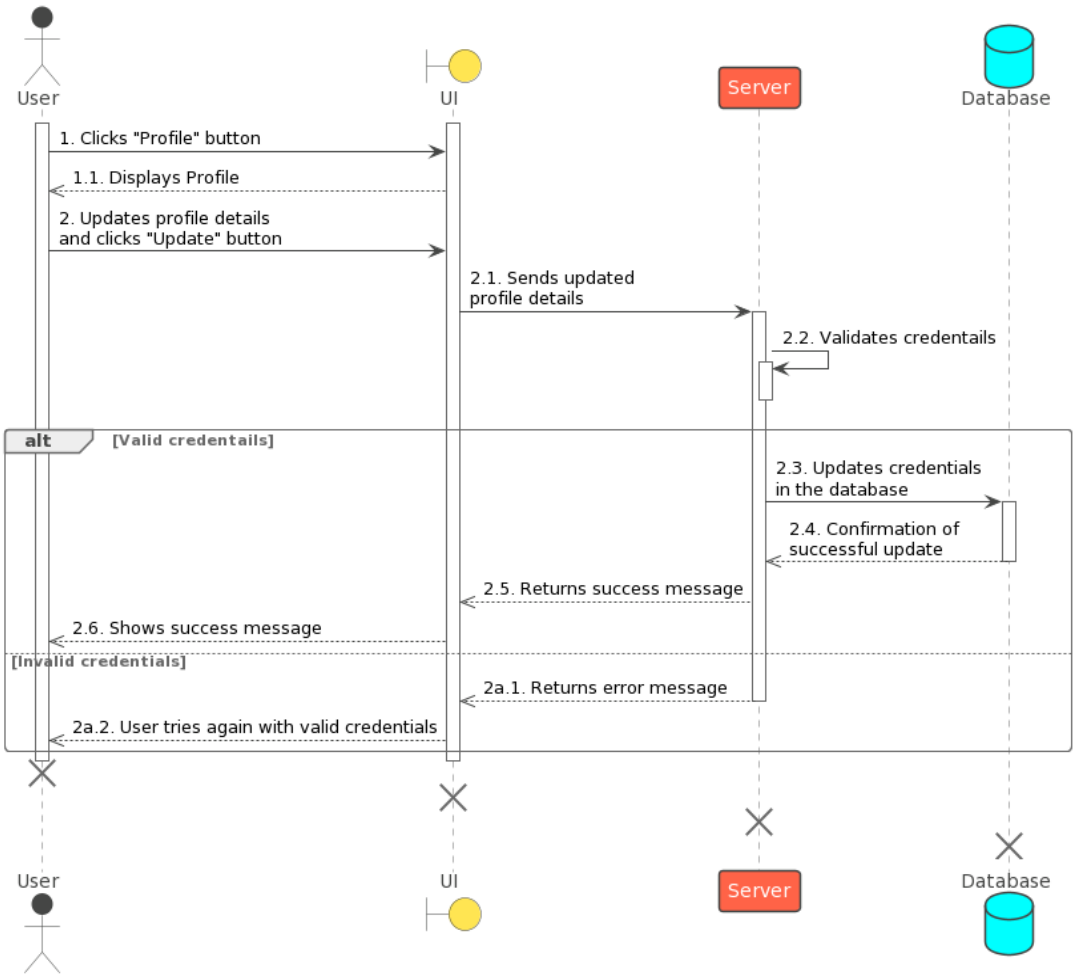

Figure 3.8: Sequence Diagram for Update Patient's Profile

3.4.5 Book Appointment

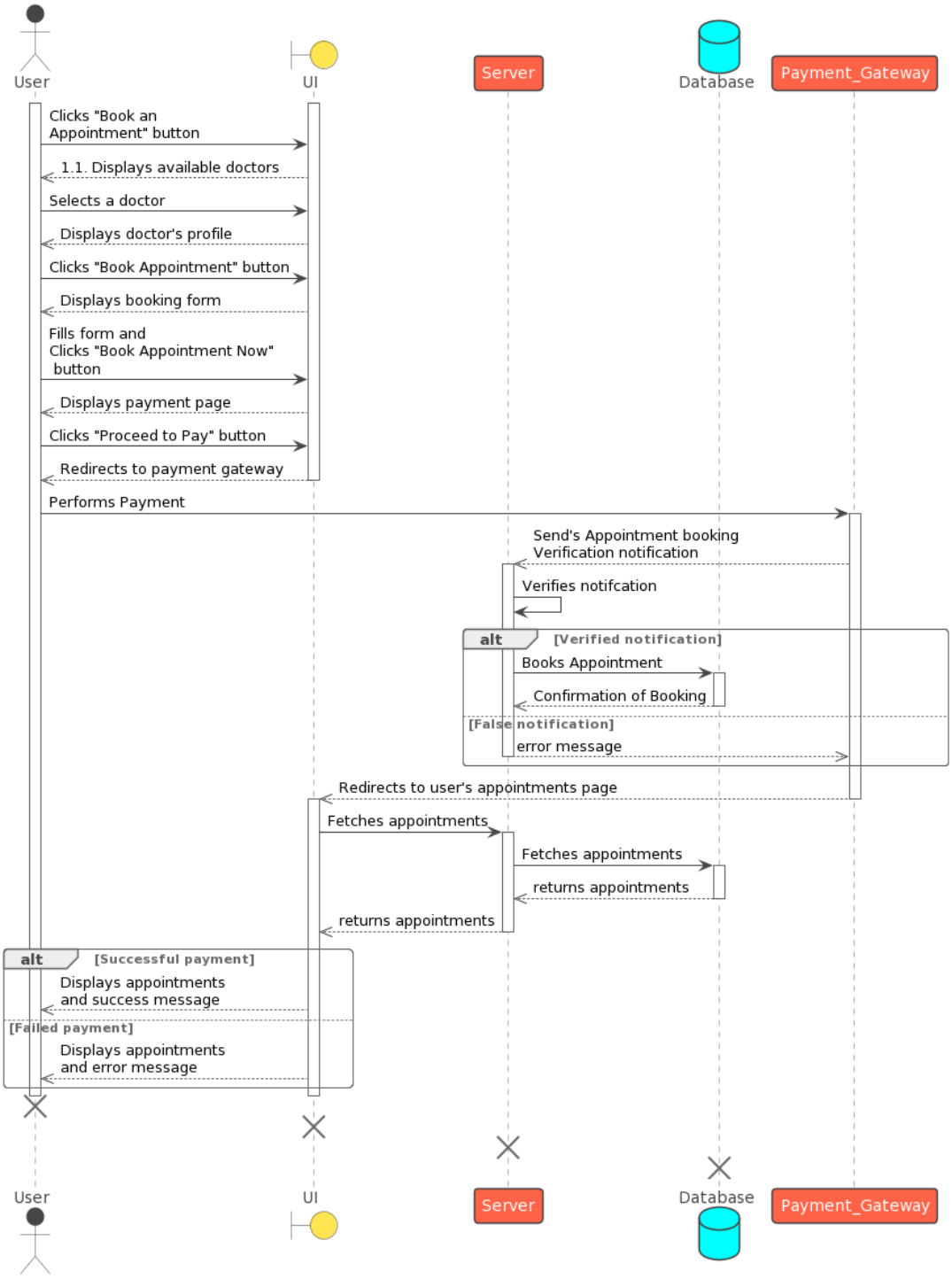

Figure 3.9: Sequence Diagram for Book Appointment

3.4.6 Reschedule Appointment

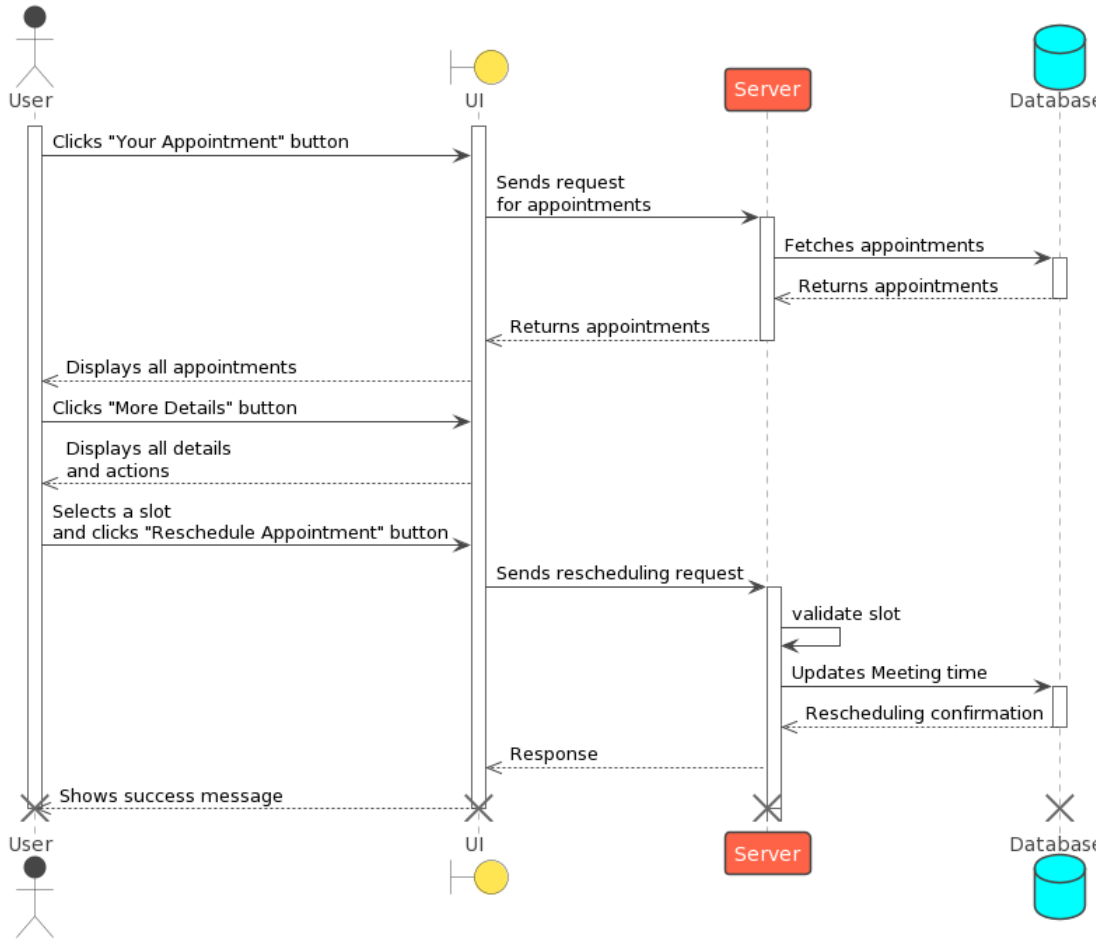

Figure 3.10: Sequence Diagram for Reschedule Appointment

3.4.7 Cancel Appointment

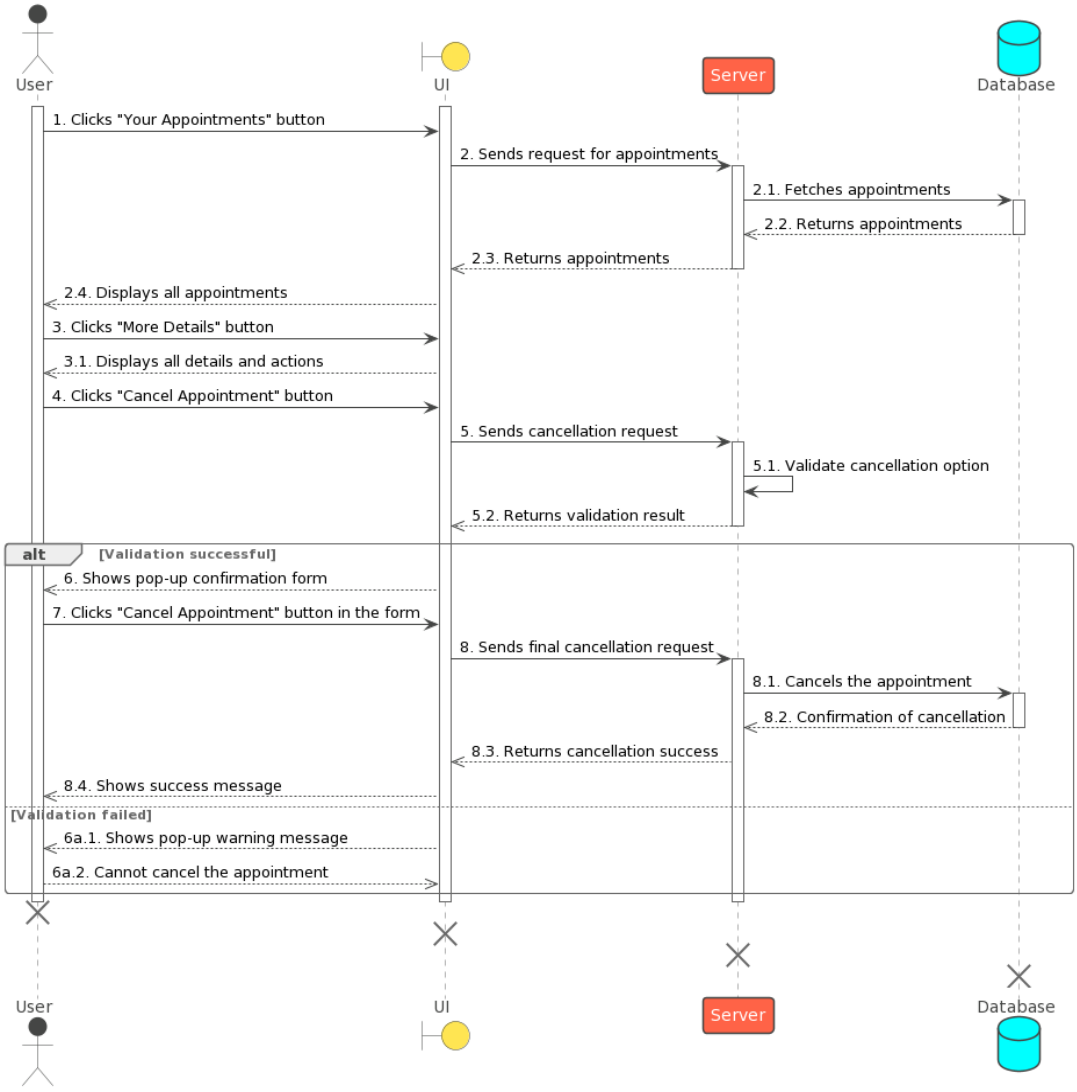

Figure 3.11: Sequence Diagram for Cancel Appointment

3.4.8 Appointment Reviews

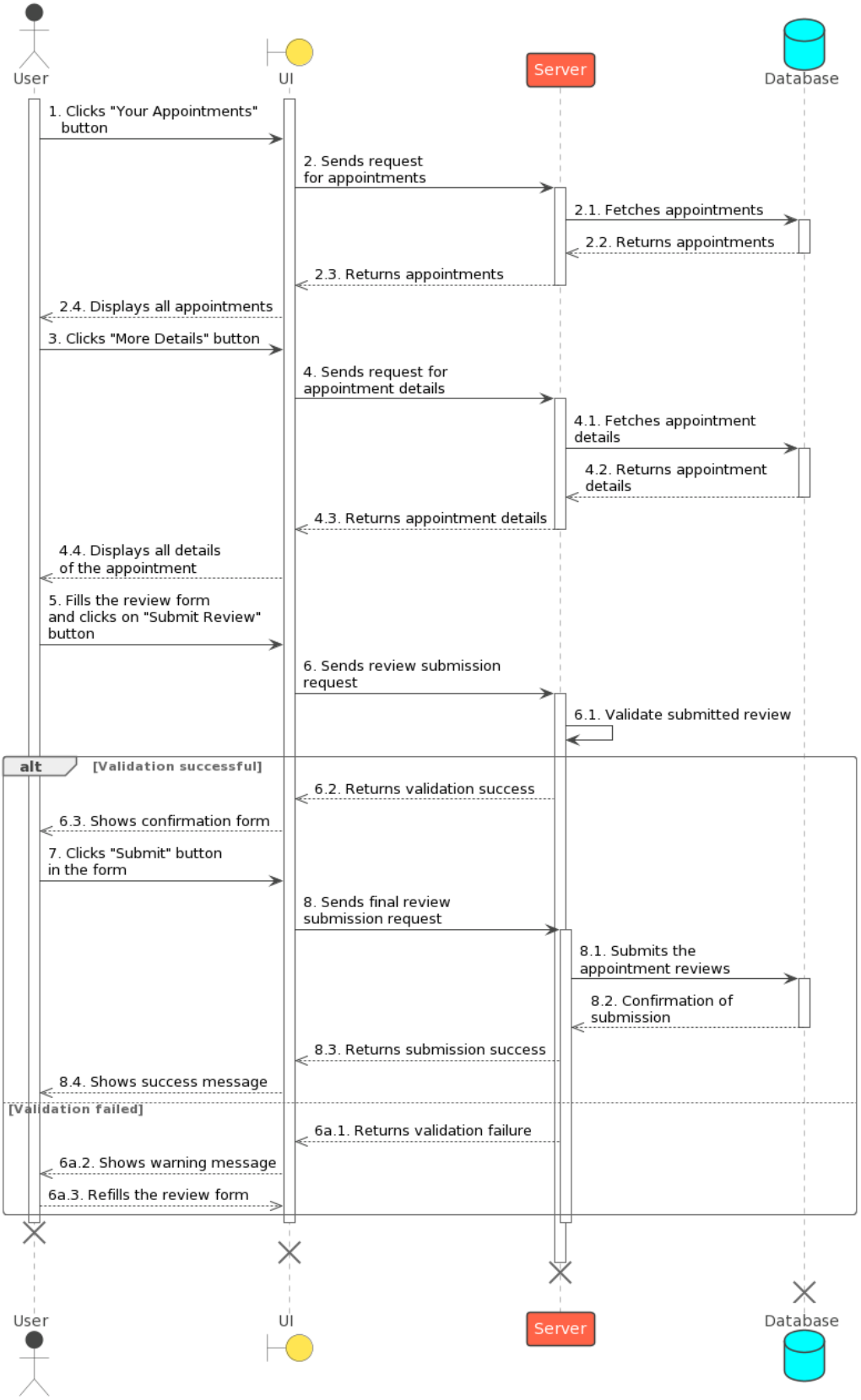

Figure 3.12: Sequence Diagram for Appointment Reviews

3.4.9 Withdraw Refund Amount

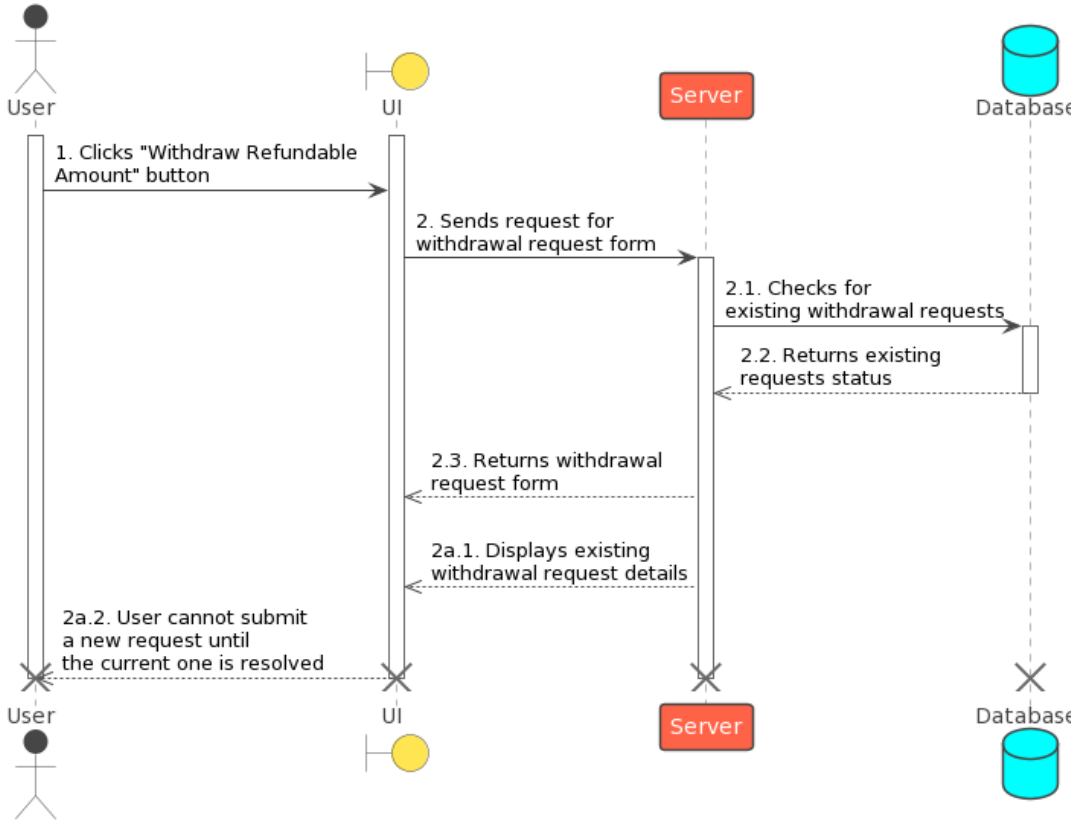

Figure 3.13: Sequence Diagram for Withdraw Refund Amount

3.4.10 Detect Brain Tumor

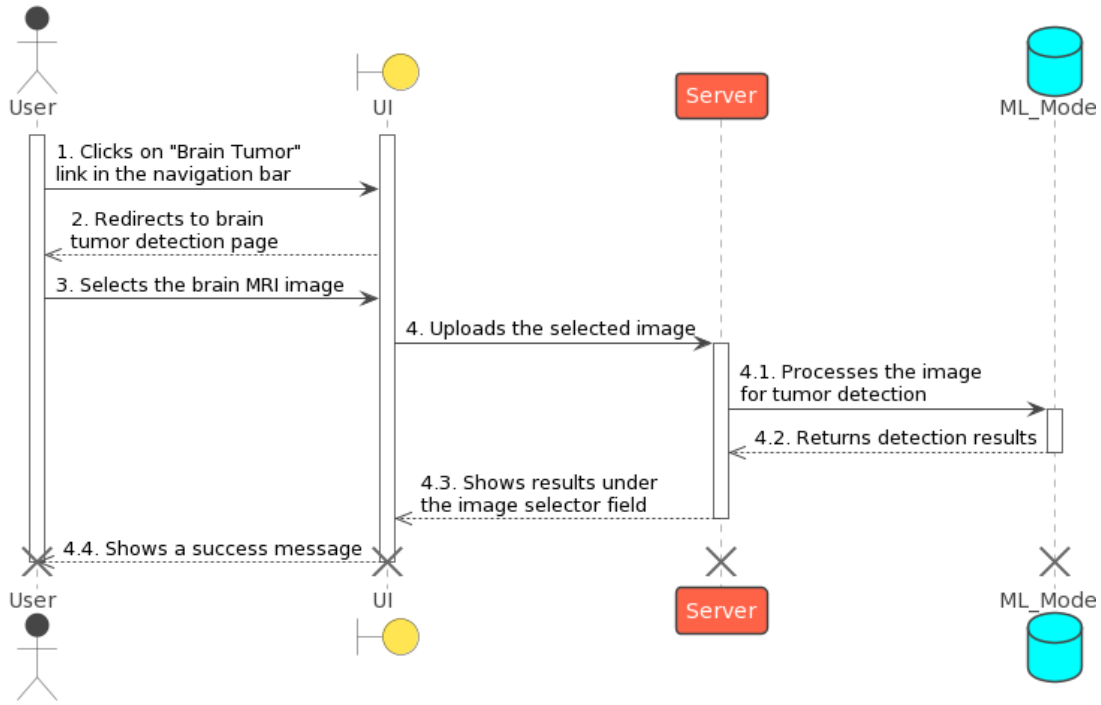

Figure 3.14: Sequence Diagram for Detect Brain Tumor

3.4.11 Detect Disease

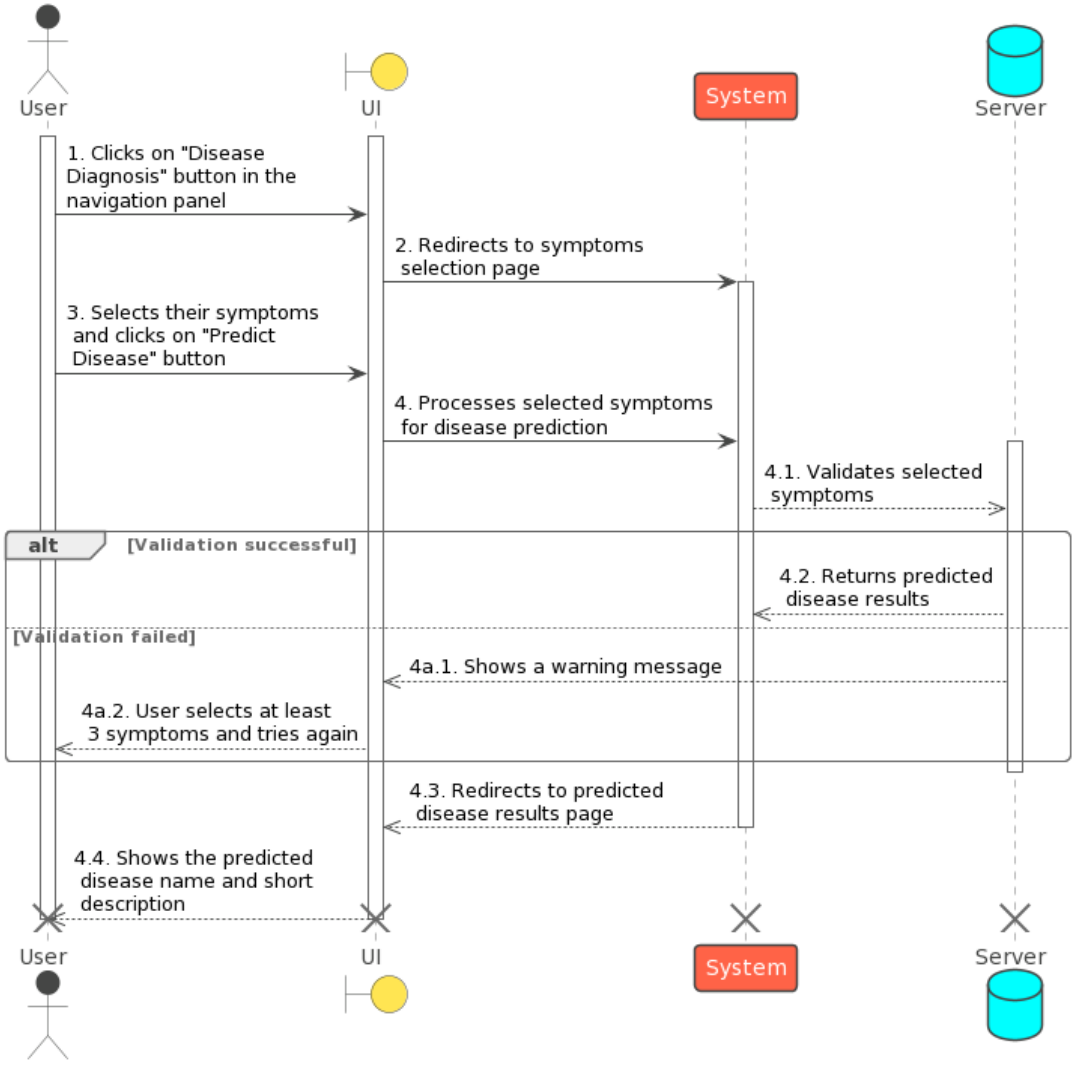

Figure 3.15: Sequence Diagram for Detect Disease

3.4.12 Get a Doctor Suggestion

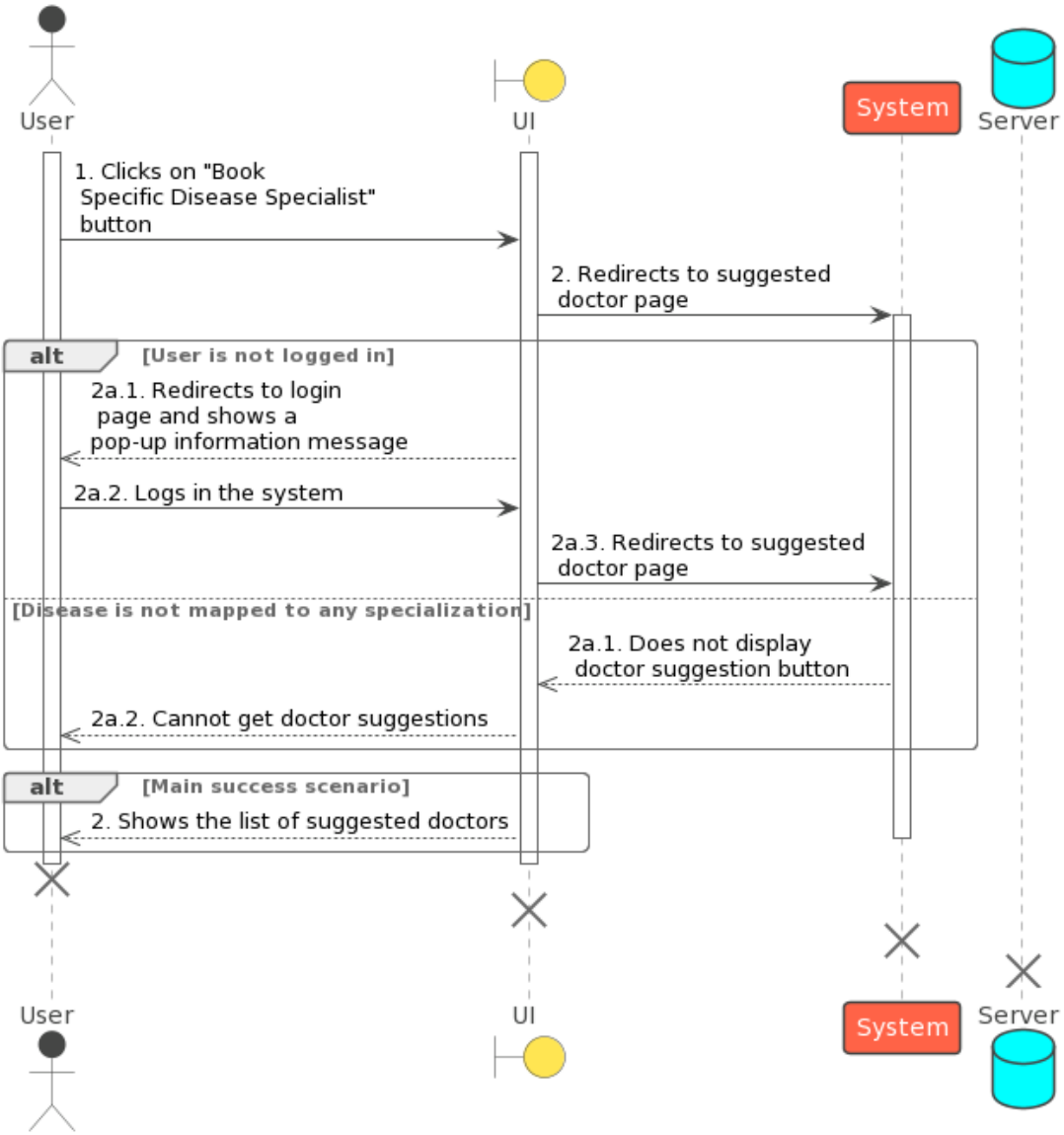

Figure 3.16: Sequence Diagram for Get Doctor Suggestion

3.4.13 Update Doctor's Profile

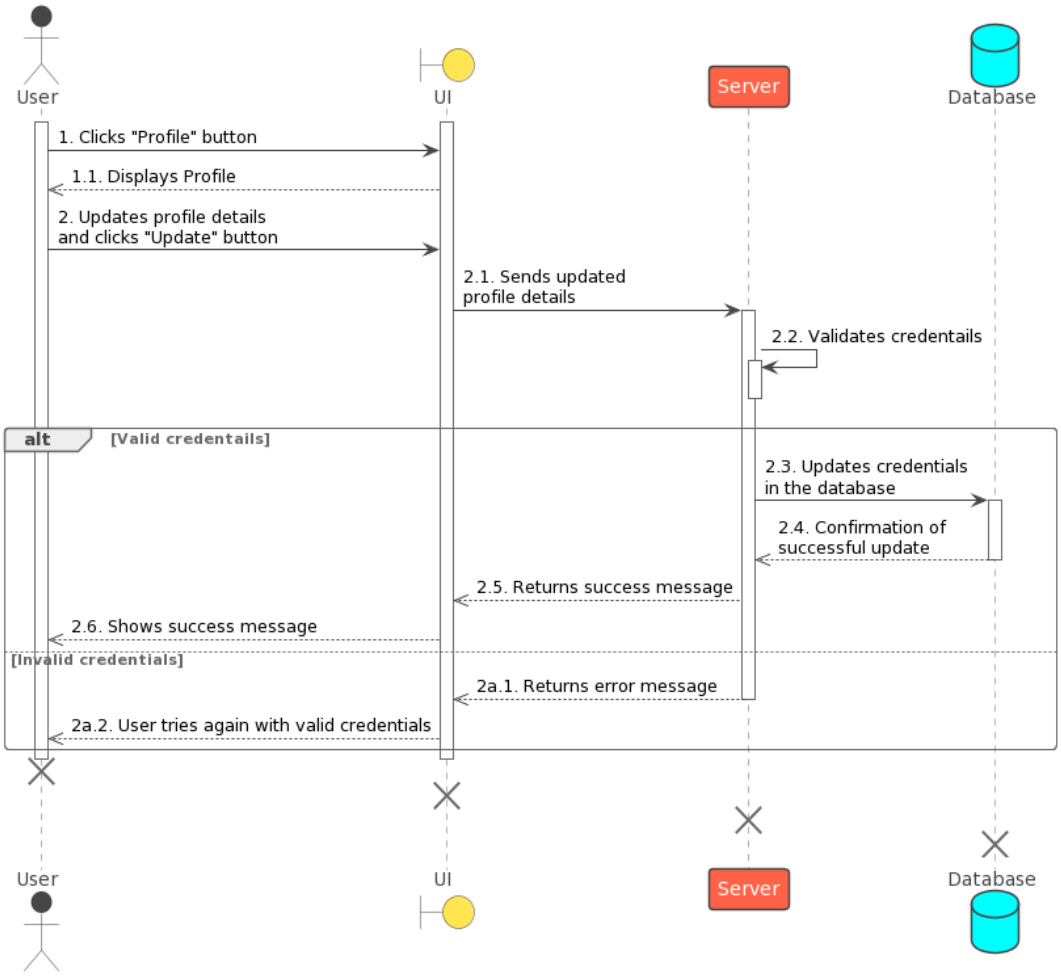

Figure 3.17: Sequence Diagram for Update Doctor's Profile

3.4.14 Account Approval Request

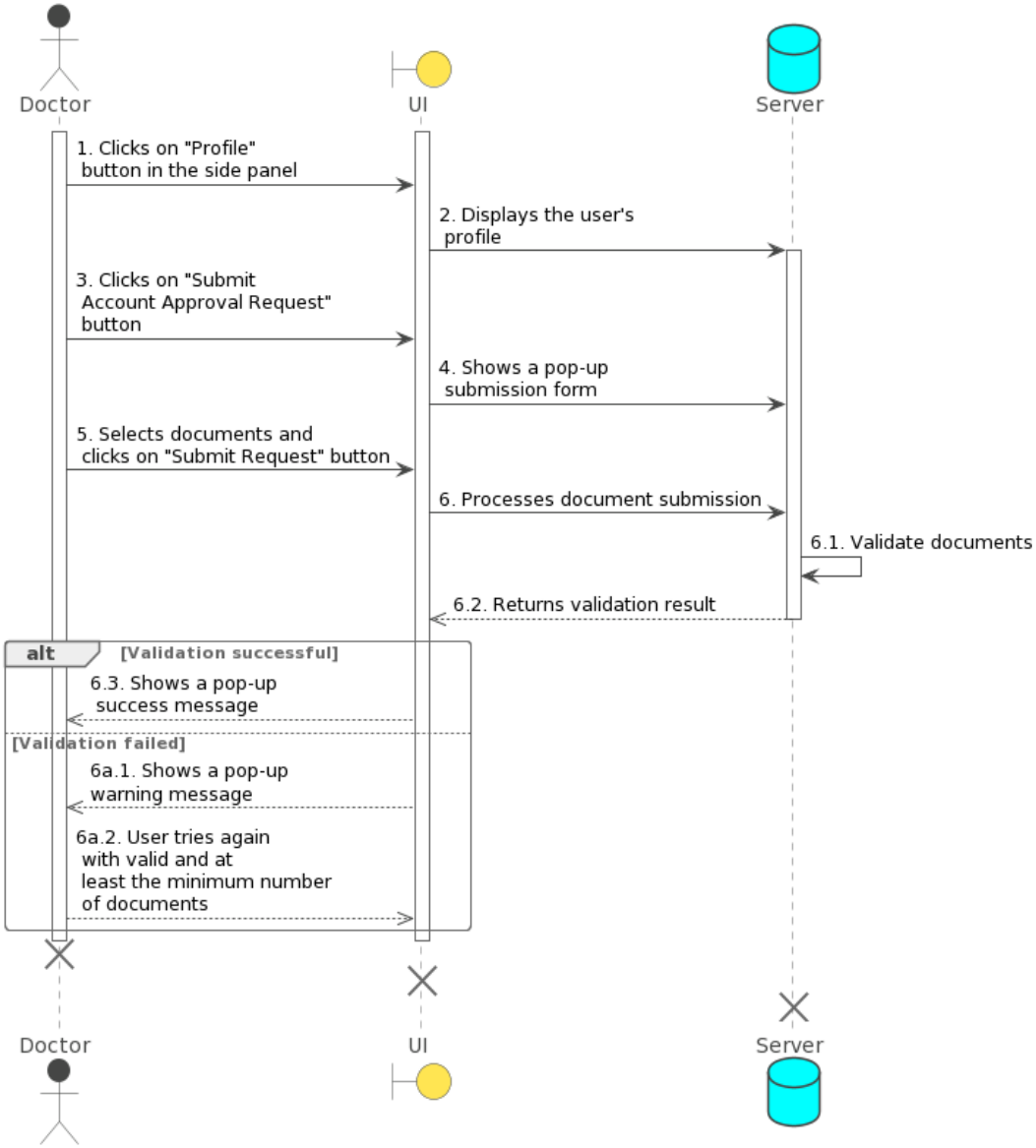

Figure 3.18: Sequence Diagram for Account Approval Request

3.4.15 Download Uploaded Documents

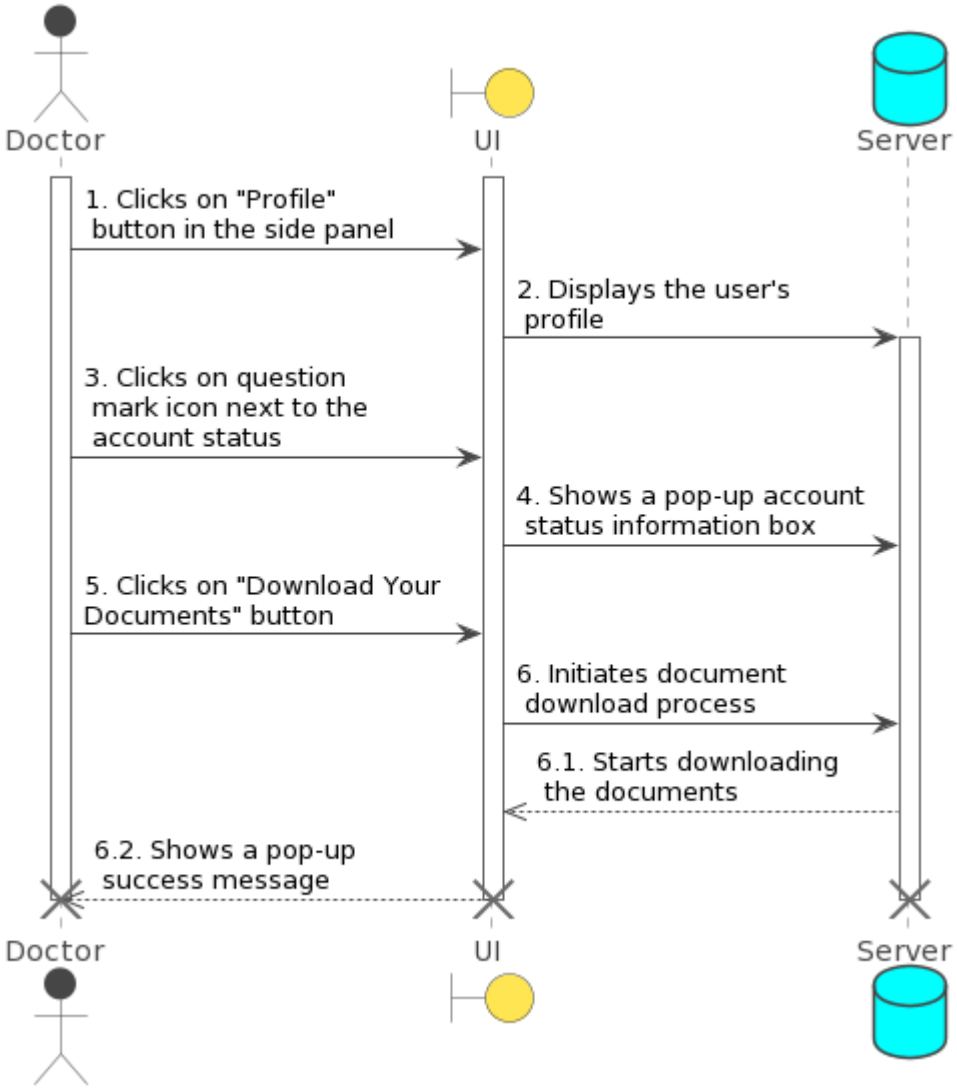

Figure 3.19: Sequence Diagram for Download Uploaded Documents

3.4.16 Get Patient's Number

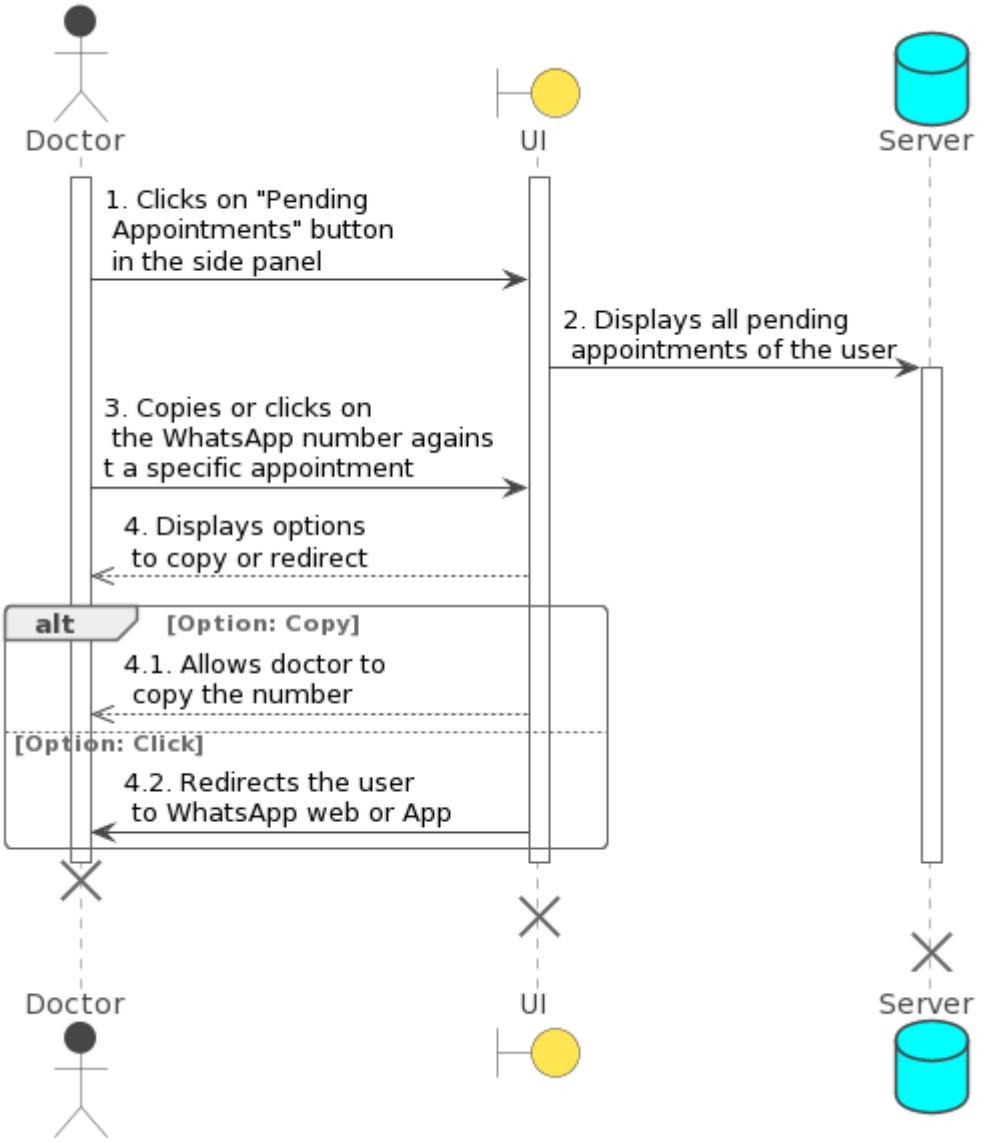

Figure 3.20: Sequence Diagram for Get Patient's Number

3.4.17 Mark Appointment as Completed

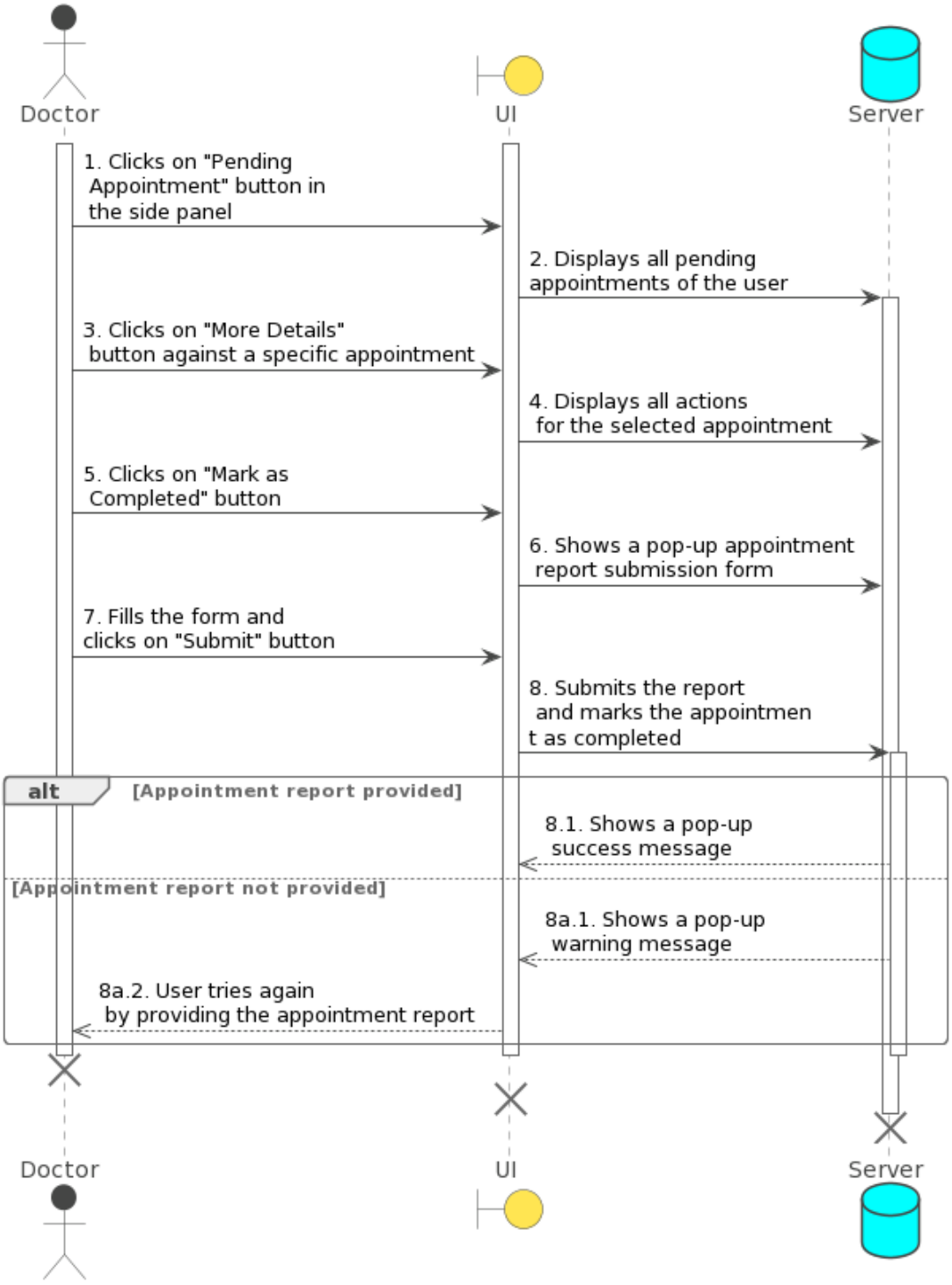

Figure 3.21: Sequence Diagram for Mark Appointment as Completed

3.4.18 Delay Appointment

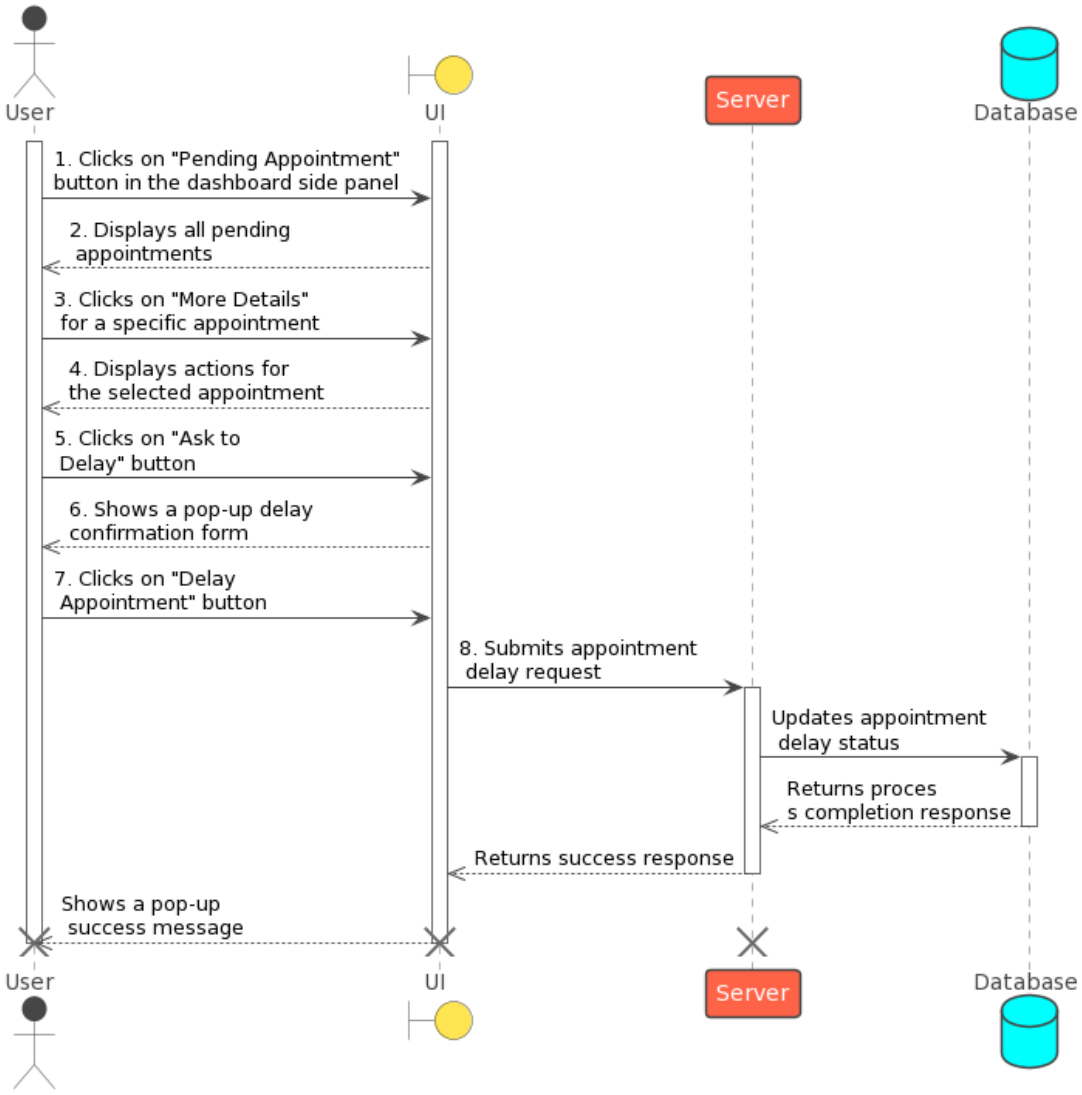

Figure 3.22: Sequence Diagram for Delay Appointment

3.4.19 Mark Doctor's Absence

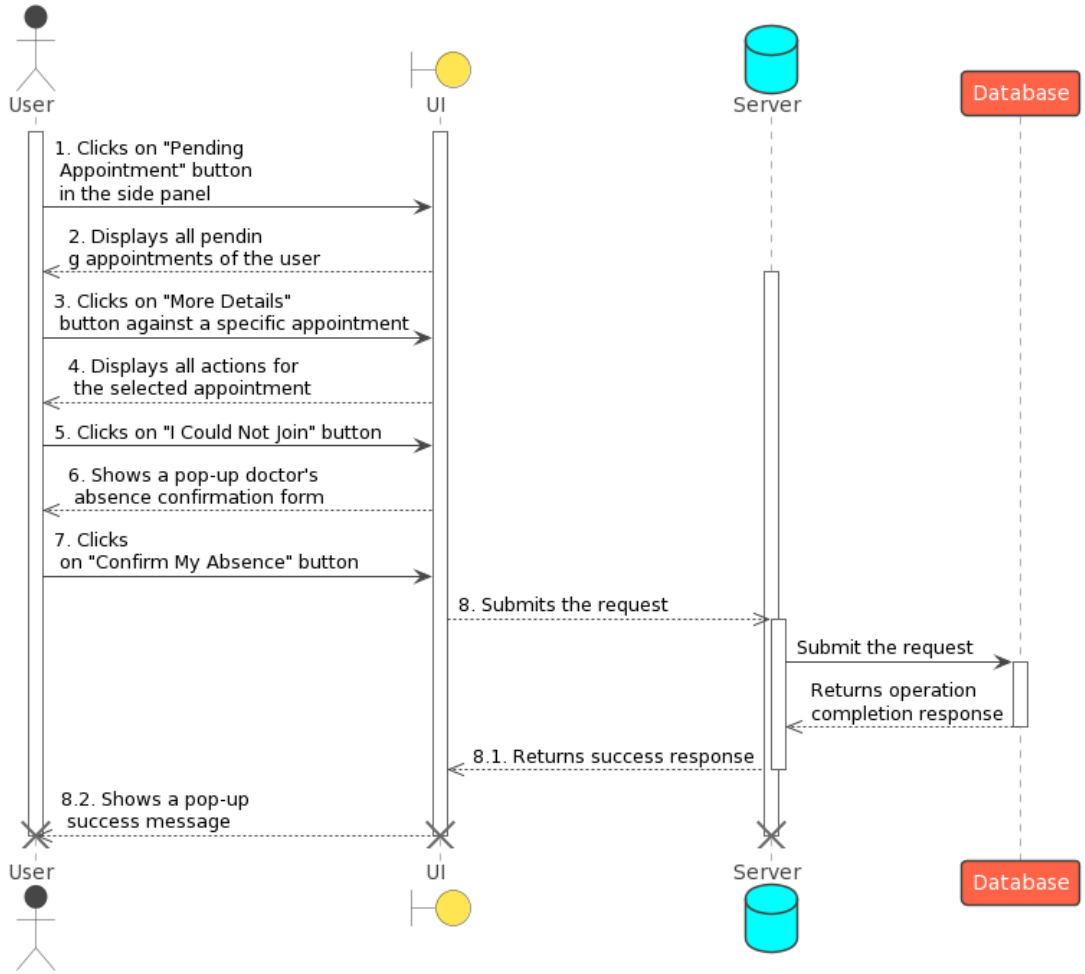

Figure 3.23: Sequence Diagram for Mark Doctor's Absence

3.4.20 Mark Patient's Absence

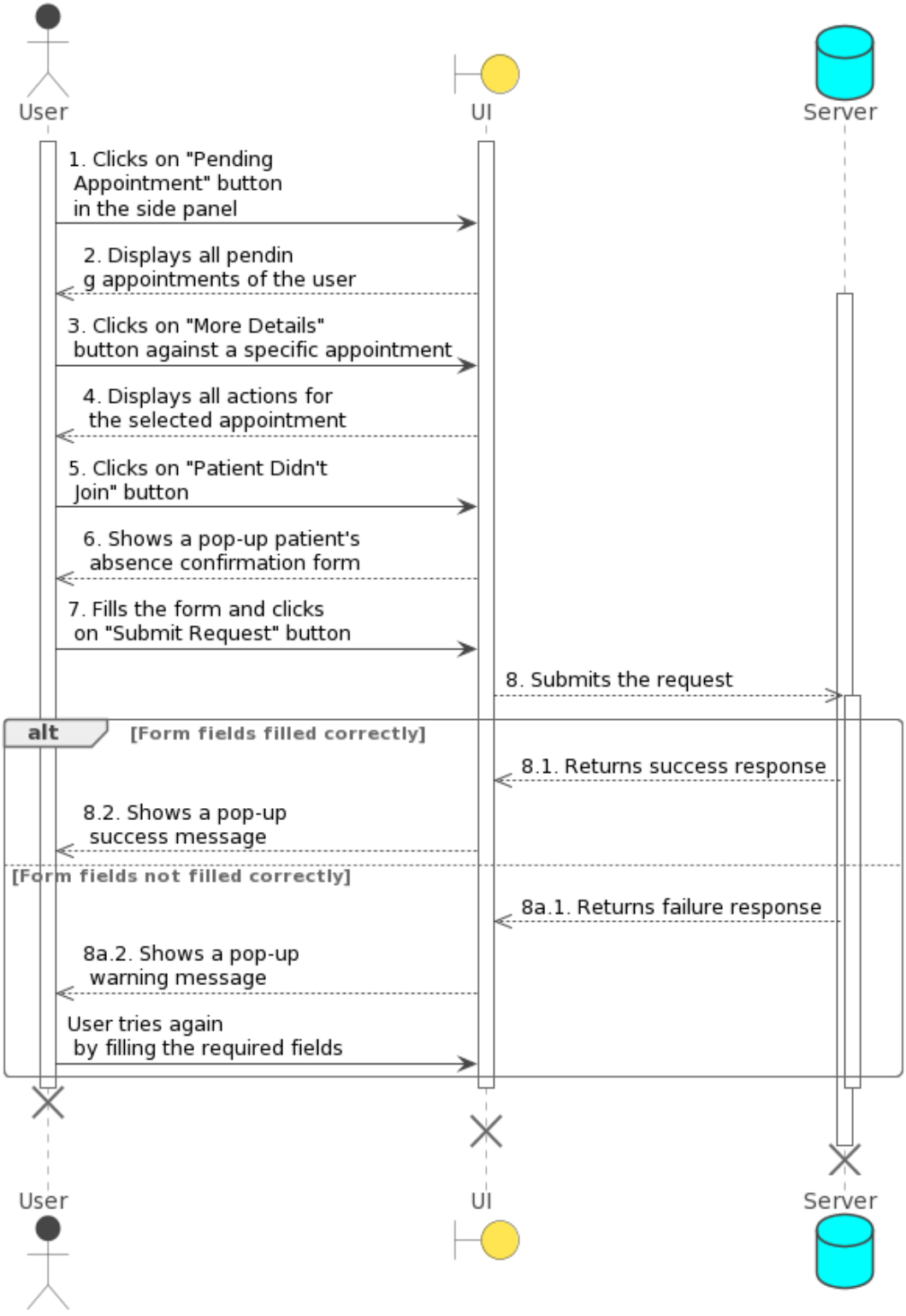

Figure 3.24: Sequence Diagram for Mark Patient's Absence

3.4.21 Cancel Appointment

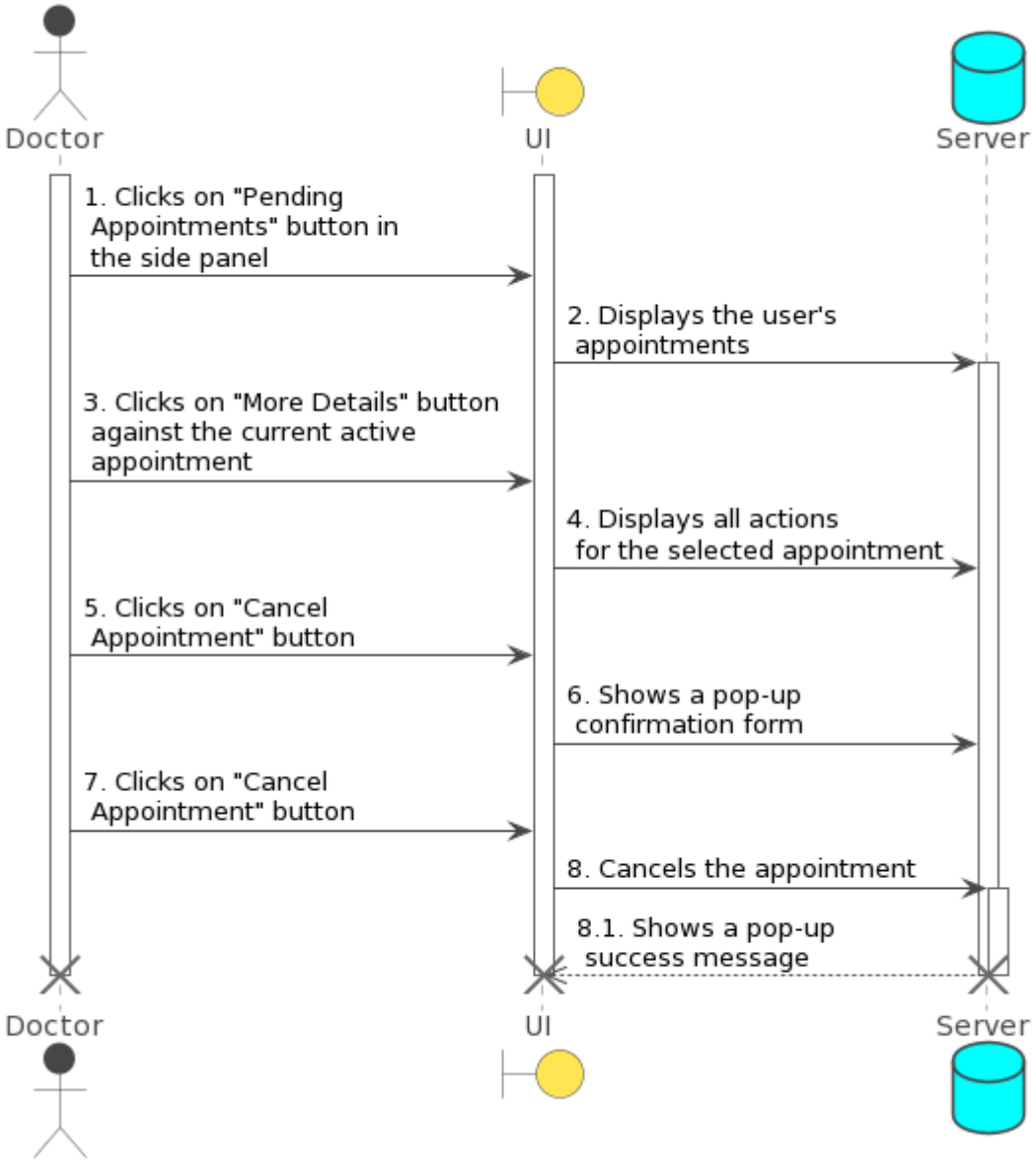

Figure 3.25: Sequence Diagram for Cancel Appointment

3.4.22 Withdraw Amount

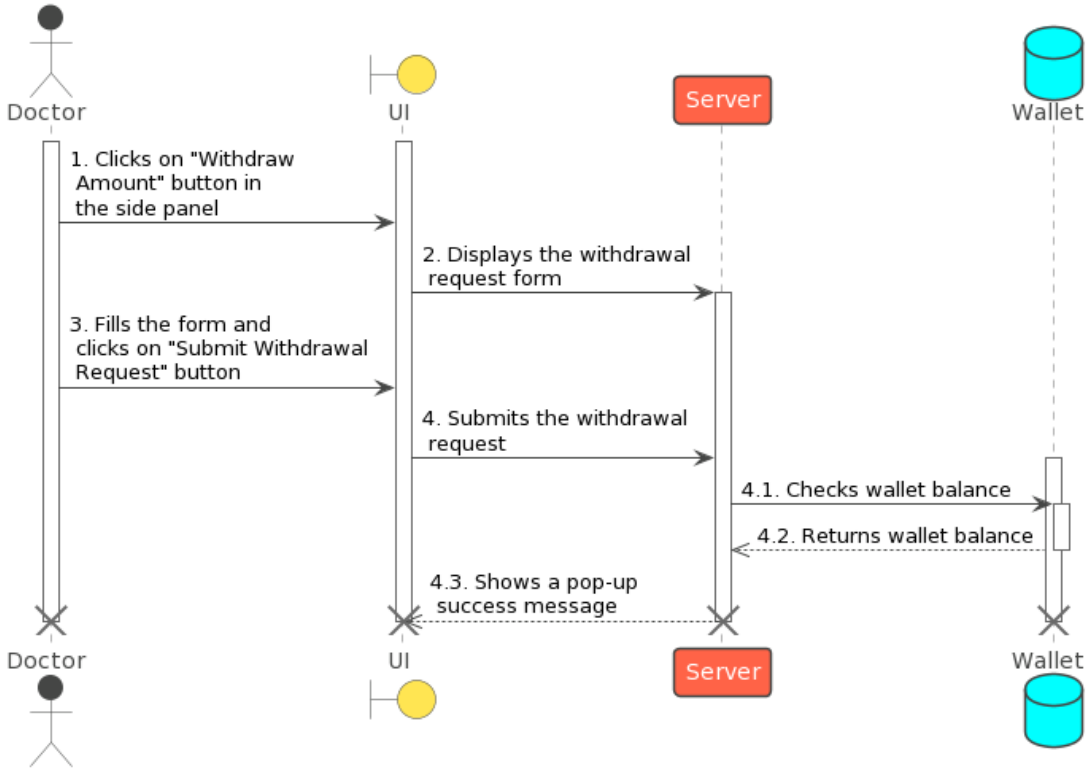

Figure 3.26: Sequence Diagram for Withdraw Amount

3.4.23 Update Patient's Details

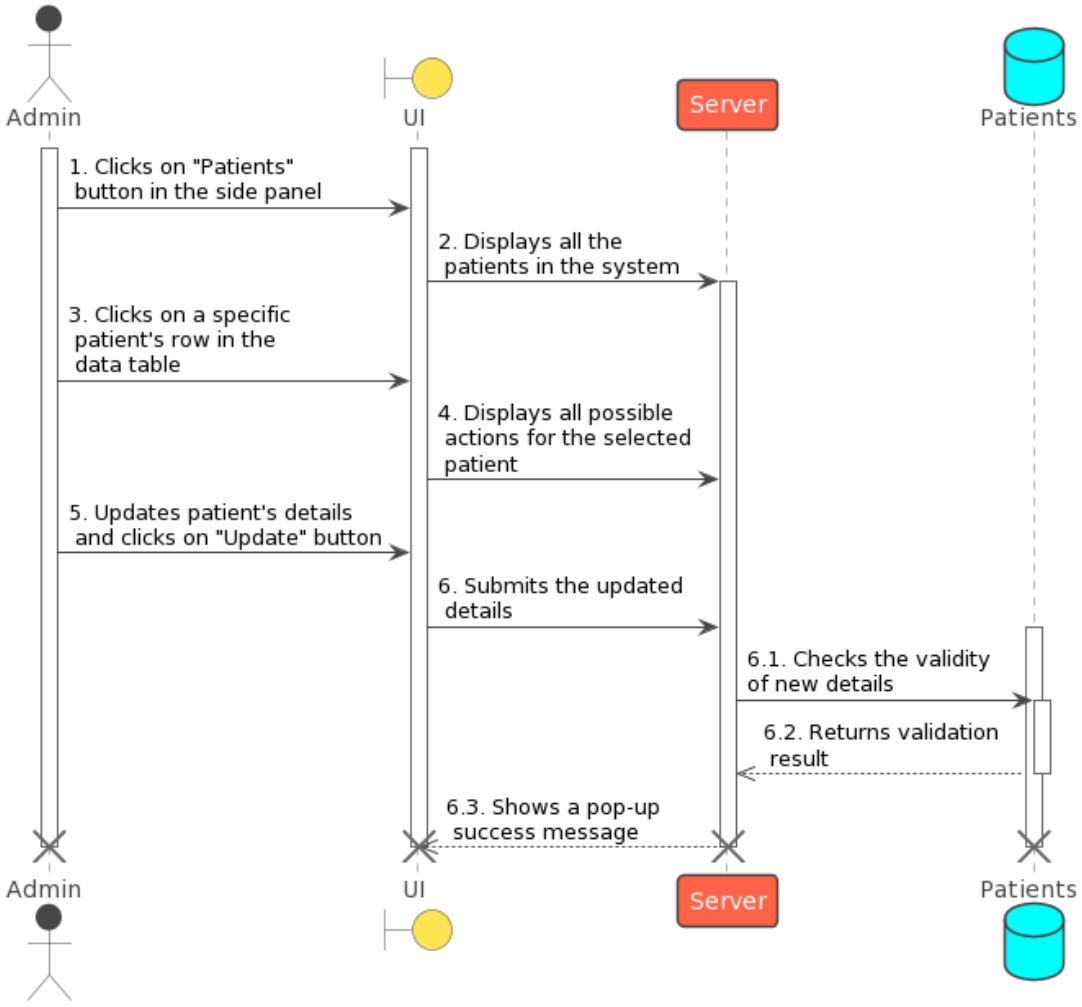

Figure 3.27: Sequence Diagram for Update Patient's Details

3.4.24 Update Doctor's Details

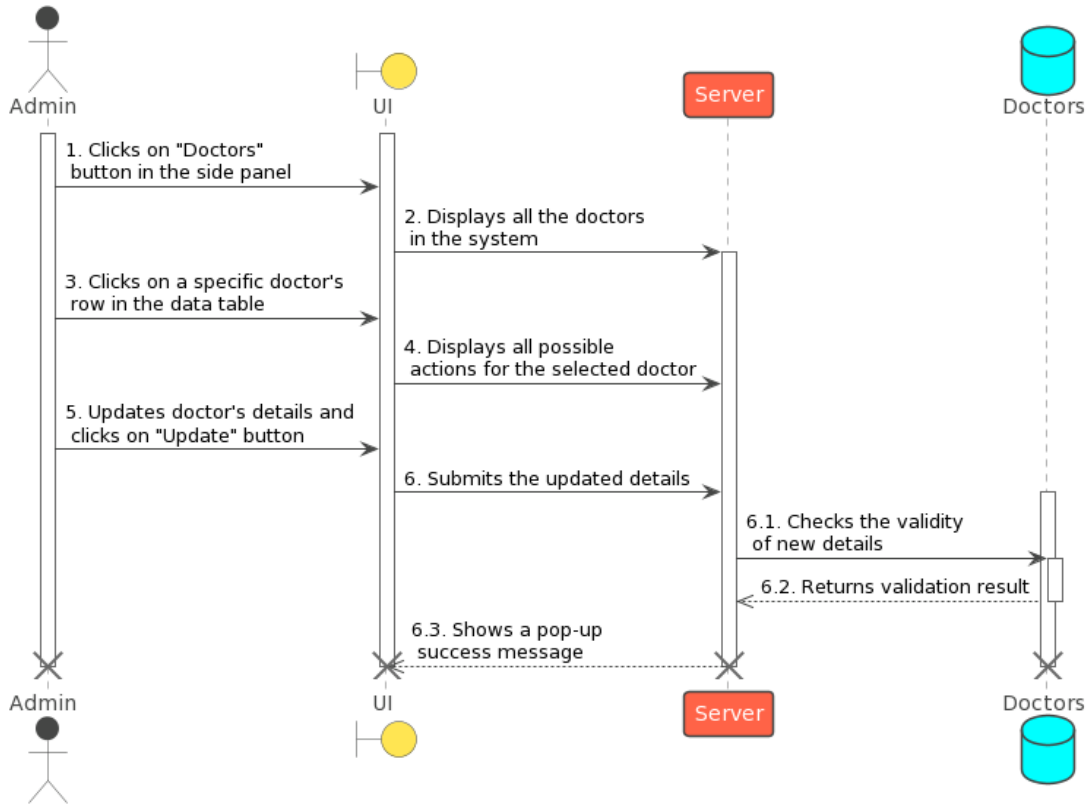

Figure 3.28: Sequence Diagram for Update Doctor's Details

3.4.25 Explore Associated Records

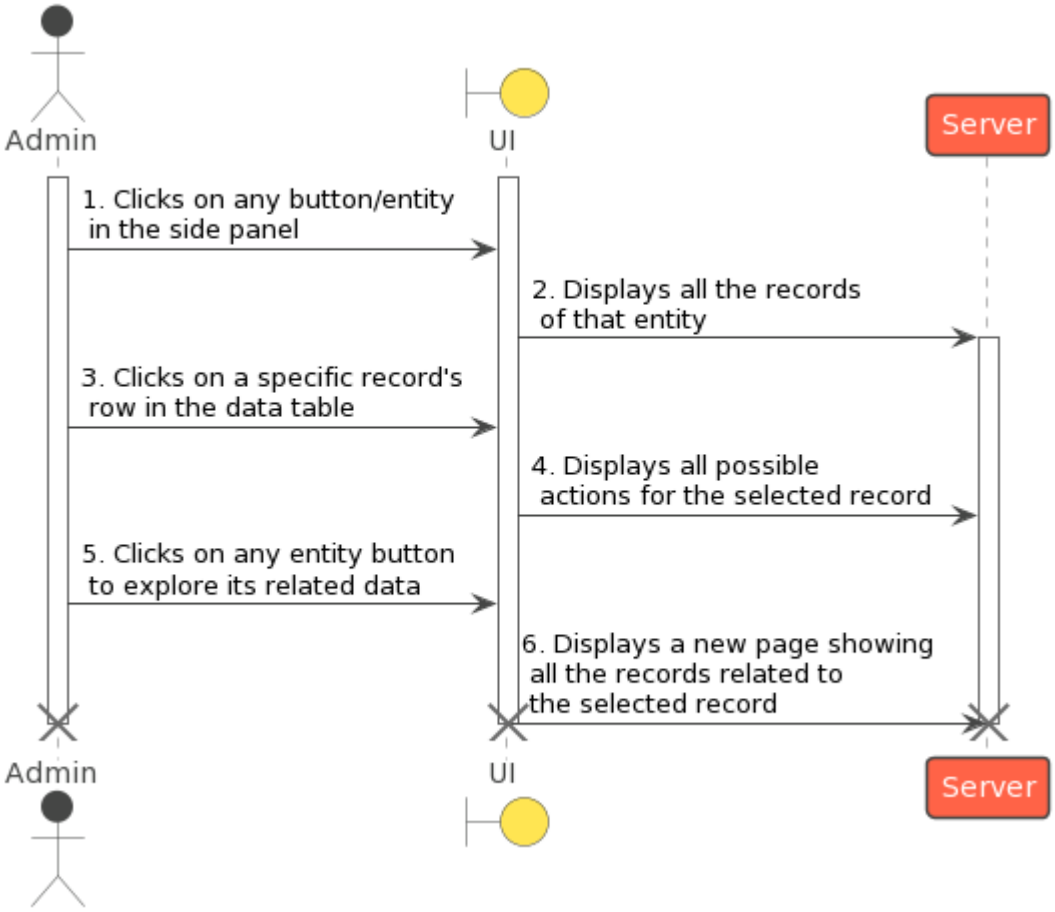

Figure 3.29: Sequence Diagram for Explore Associated Records

3.4.26 Accept/Reject Doctor's Approval

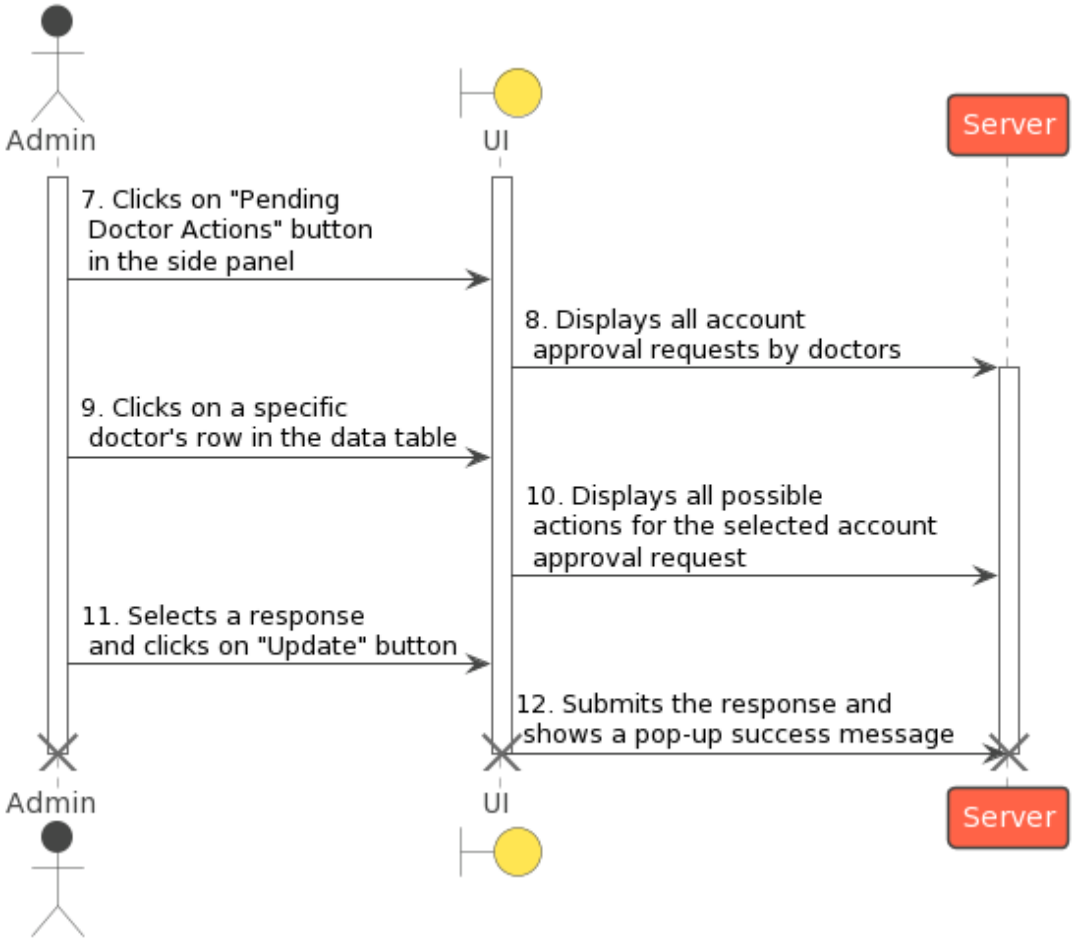

Figure 3.30: Sequence Diagram for Accept/Reject Doctor's Approval

3.4.27 Approve/Reject Patient's Absence

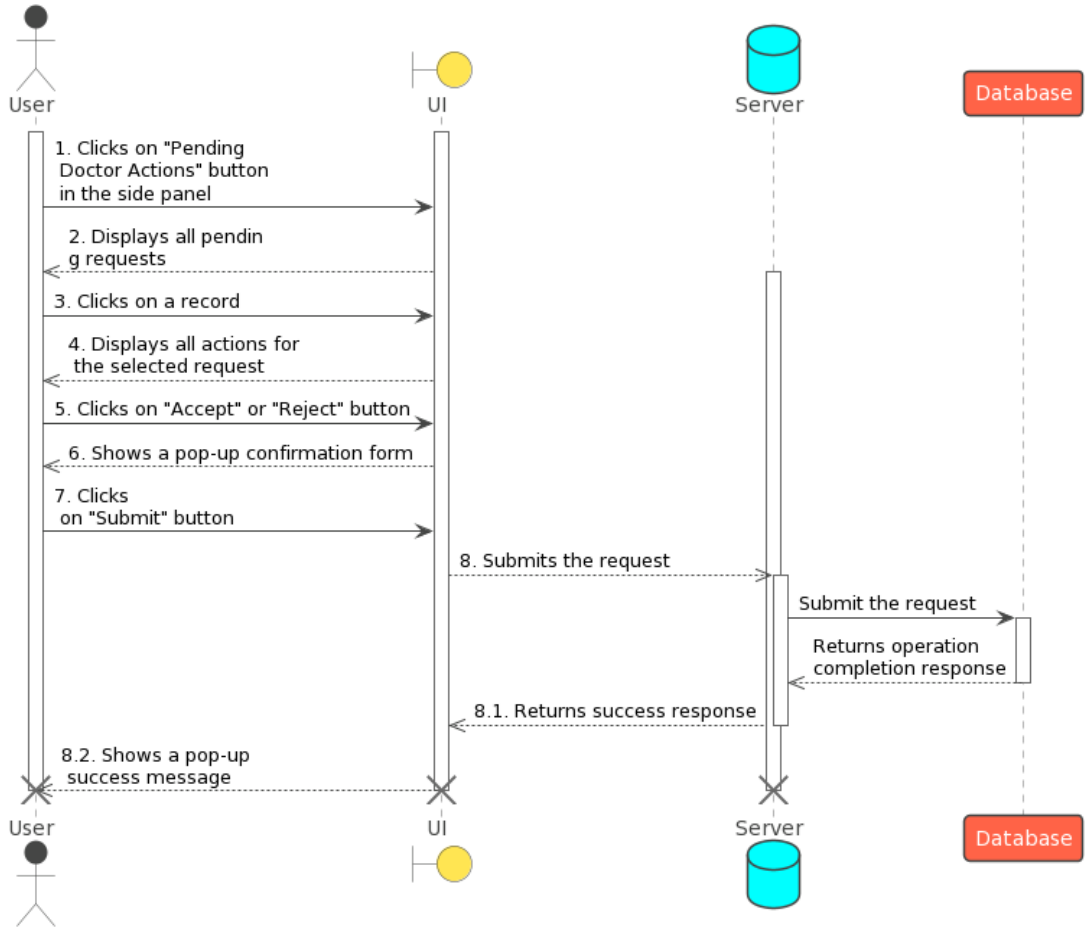

Figure 3.31: Sequence Diagram for Approve/Reject Patient's Absence

3.4.28 Accept Doctor's Withdrawal Request

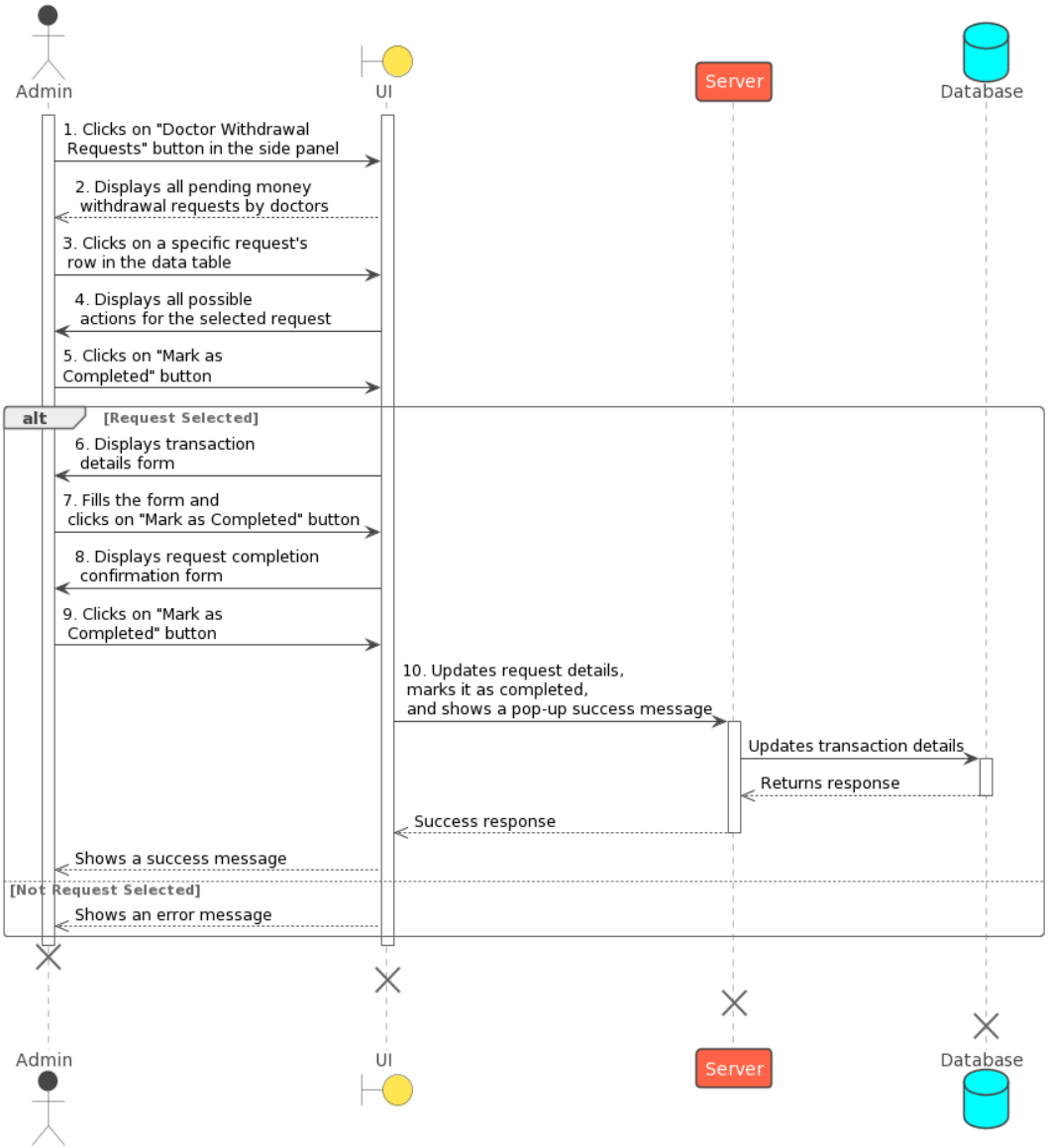

Figure 3.32: Sequence Diagram for Accept Doctor's Withdrawal Request

3.4.29 Reject Doctor's Withdrawal Request

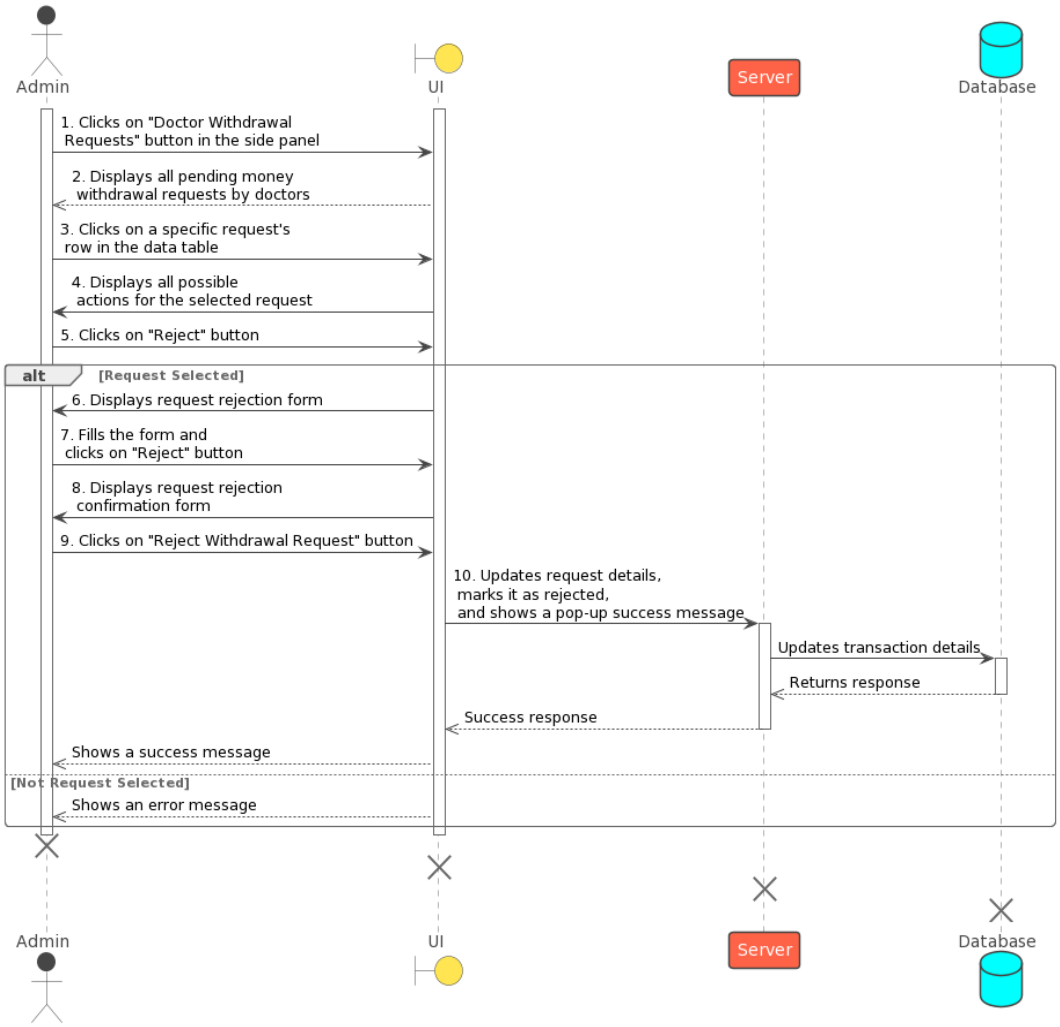

Figure 3.33: Sequence Diagram for Reject Doctor's Withdrawal Request

3.4.30 Accept Patient's Withdrawal Request

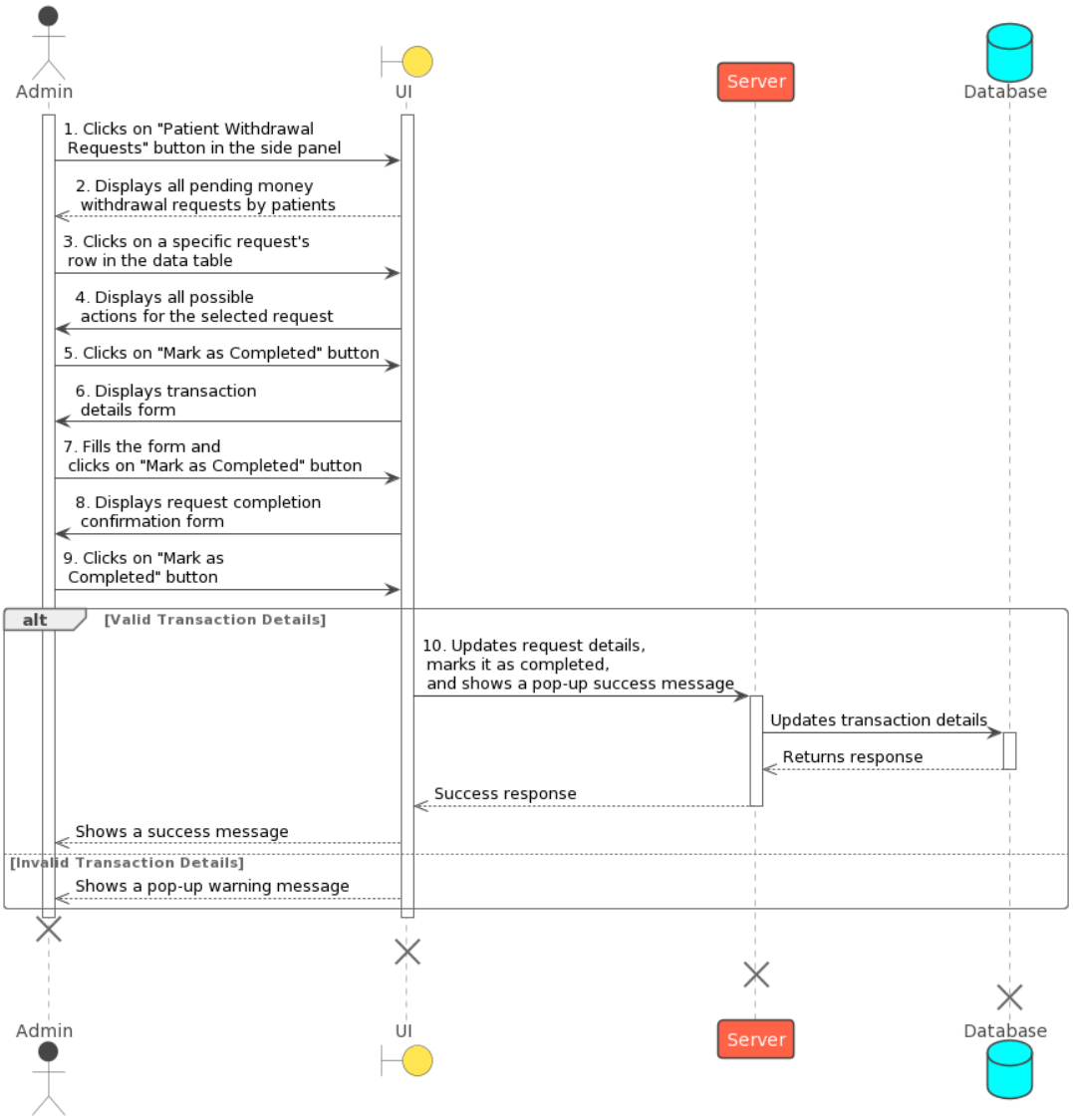

Figure 3.34: Sequence Diagram for Accept Patient's Withdrawal Request

3.4.31 Reject Patient's Withdrawal Request

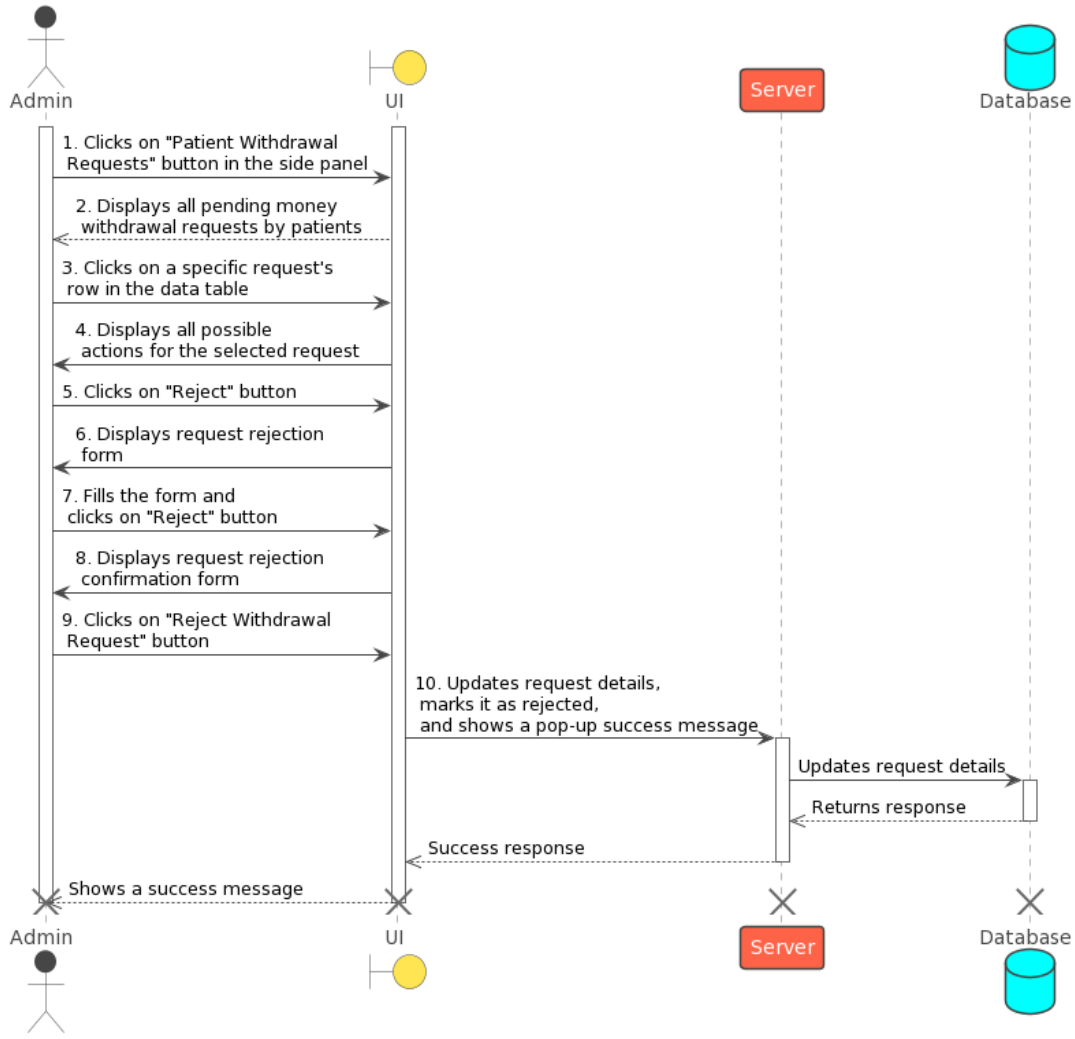

Figure 3.35: Sequence Diagram for Reject Patient's Withdrawal Request

3.4.32 Create Specialization Category

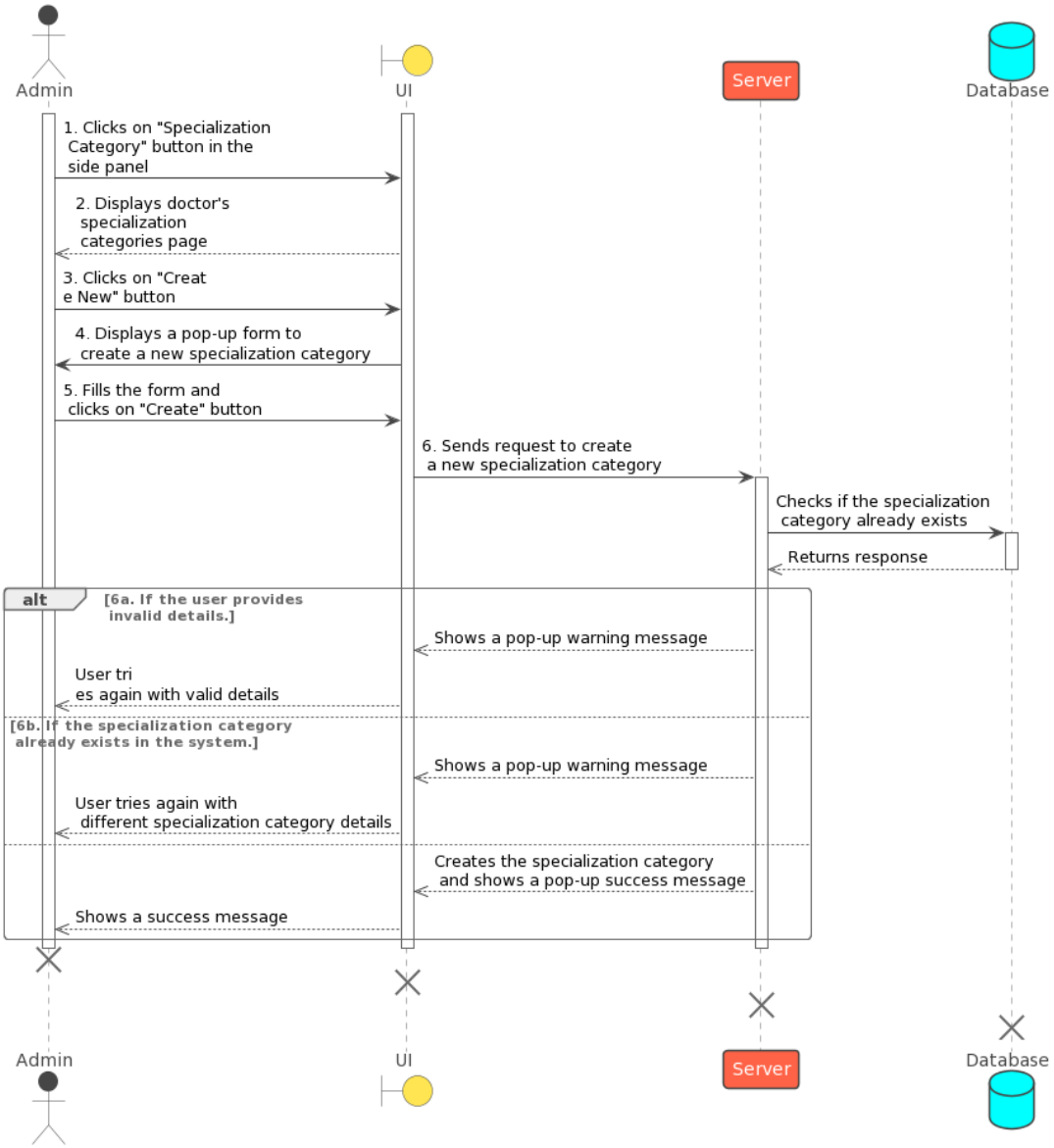

Figure 3.36: Sequence Diagram for Create Specialization Category

3.4.33 Update Specialization Category

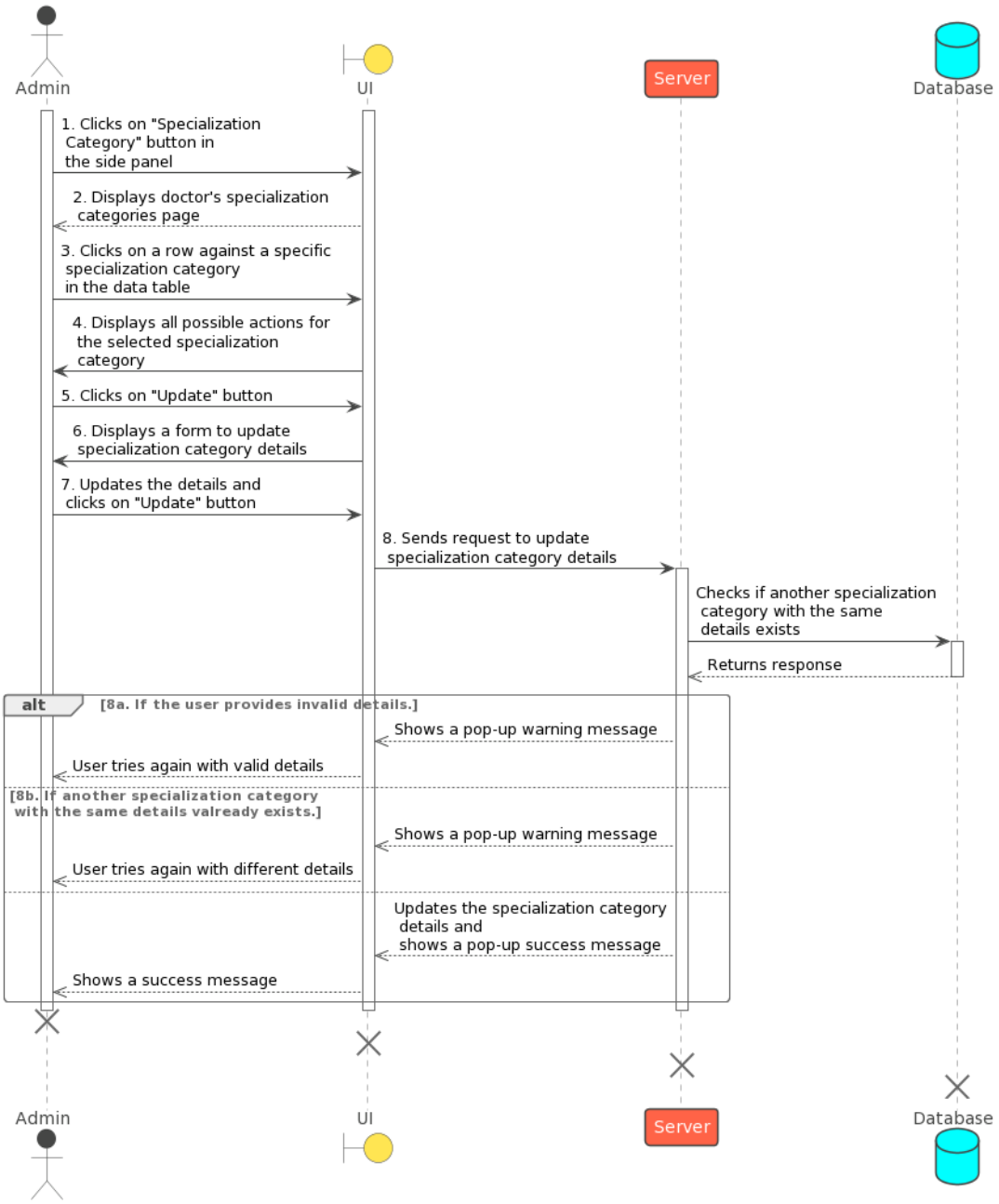

Figure 3.37: Sequence Diagram for Update Specialization Category

3.4.34 Delete Specialization Category

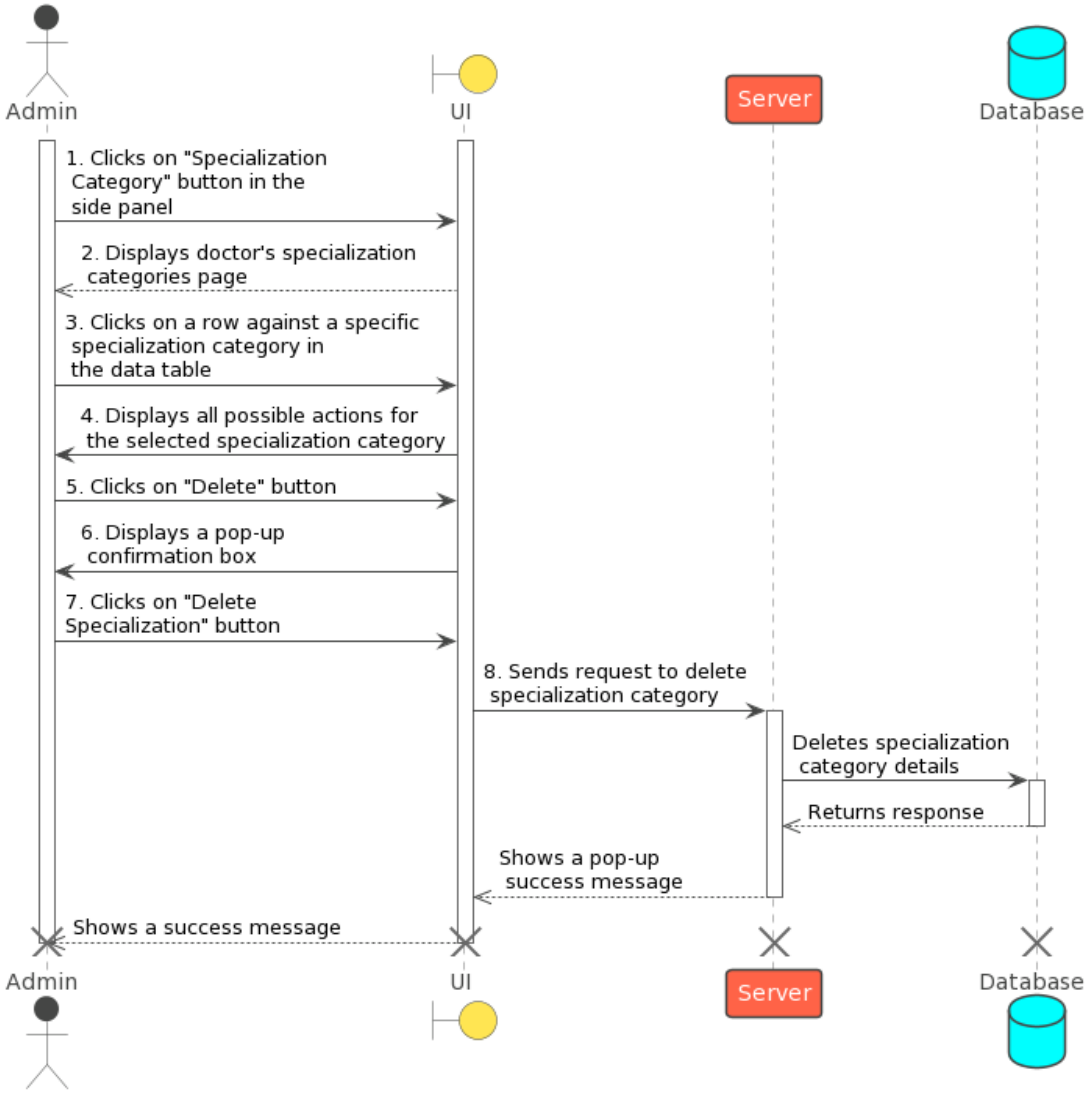

Figure 3.38: Sequence Diagram for Delete Specialization Category

3.4.35 Create Language

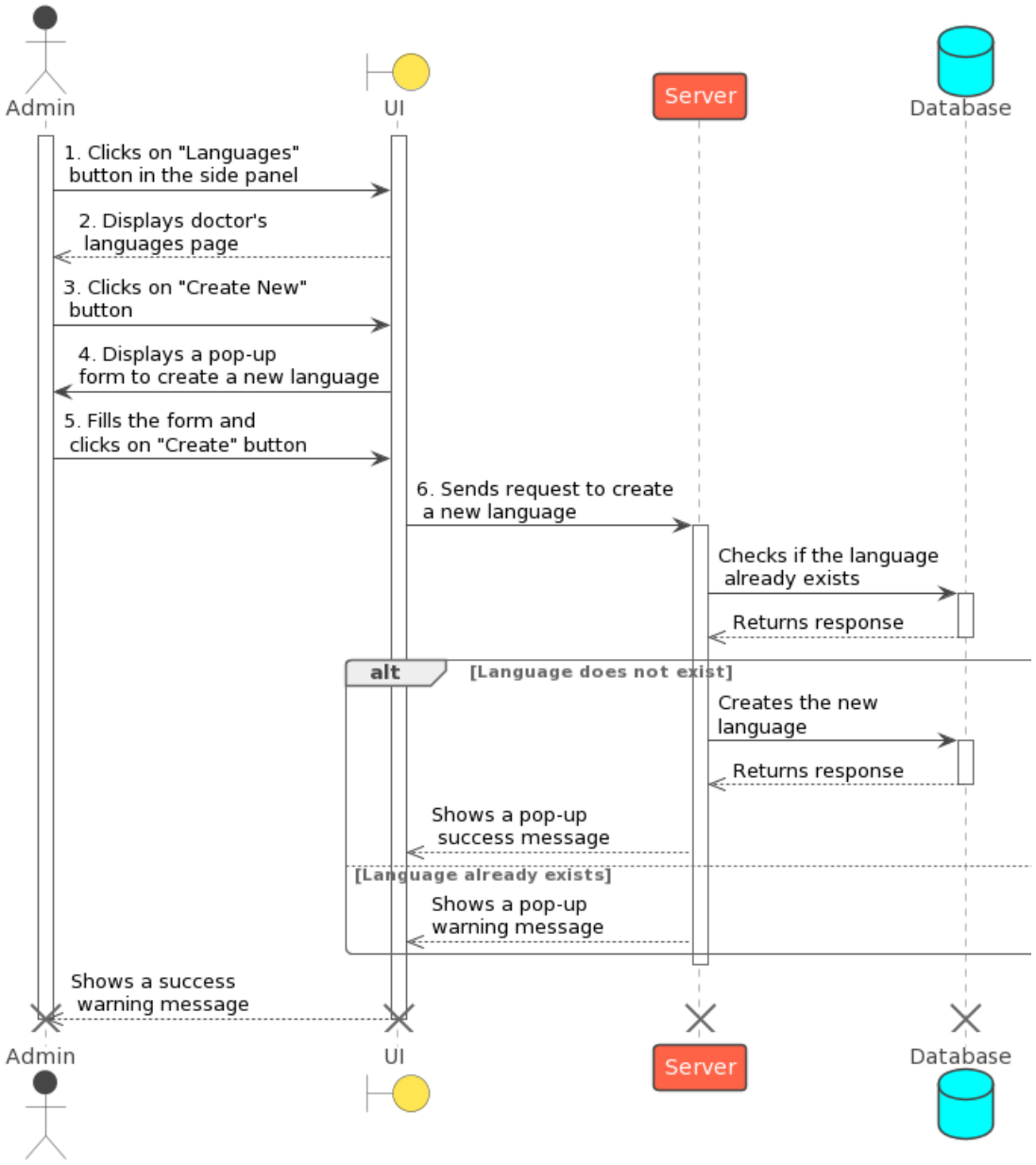

Figure 3.39: Sequence Diagram for Create Language

3.4.36 Update Language

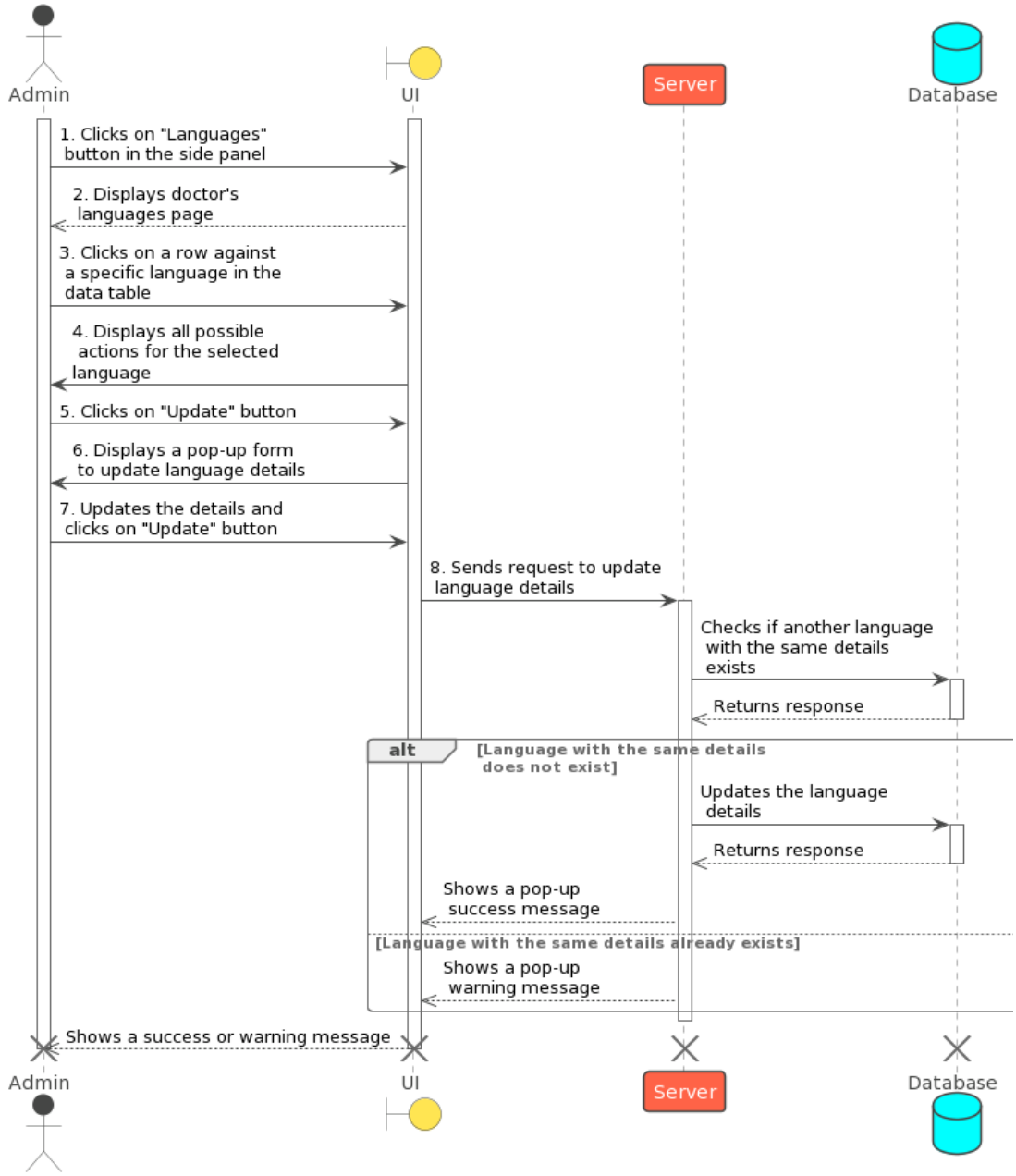

Figure 3.40: Sequence Diagram for Update Language

3.4.37 Delete Language

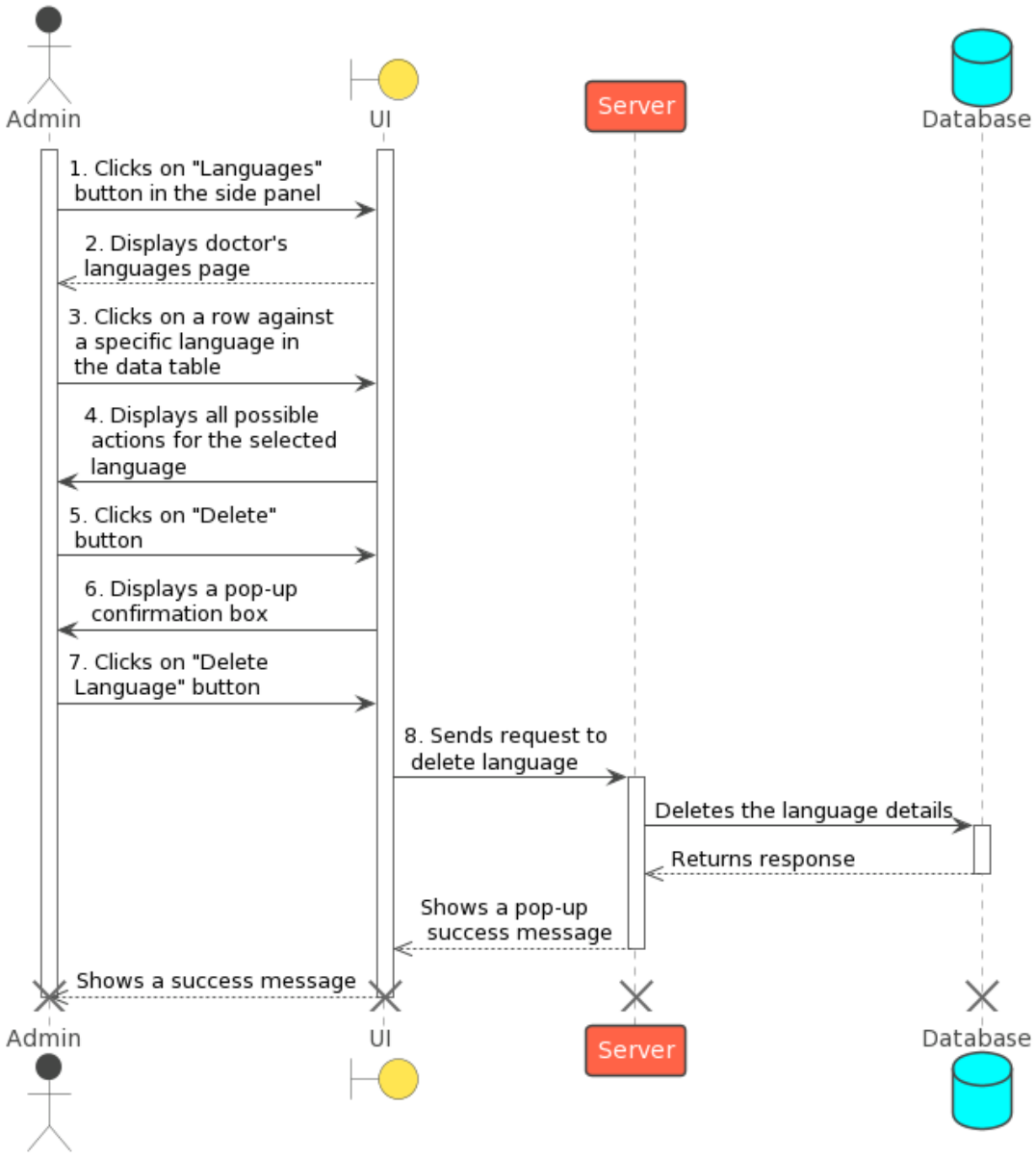

Figure 3.41: Sequence Diagram for Delete Language

3.4.38 Create Disease and Specialization Mapping

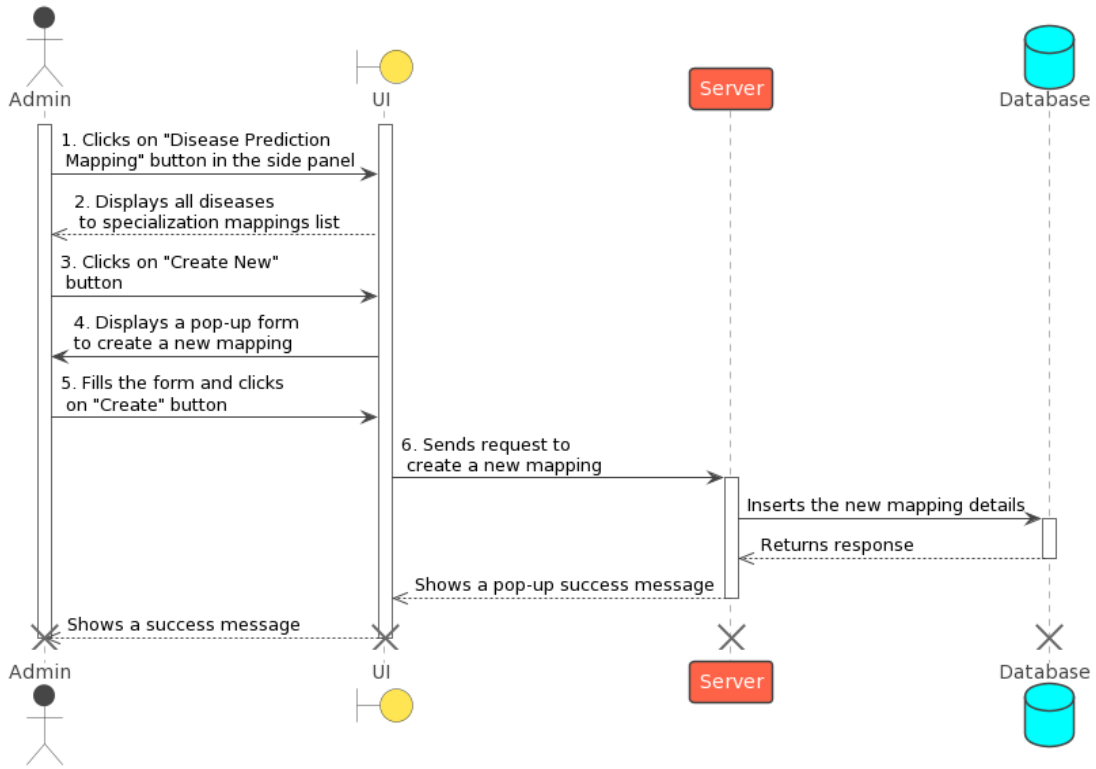

Figure 3.42: Sequence Diagram for Create Disease and Specialization Mapping

3.4.39 Delete Disease and Specialization Mapping

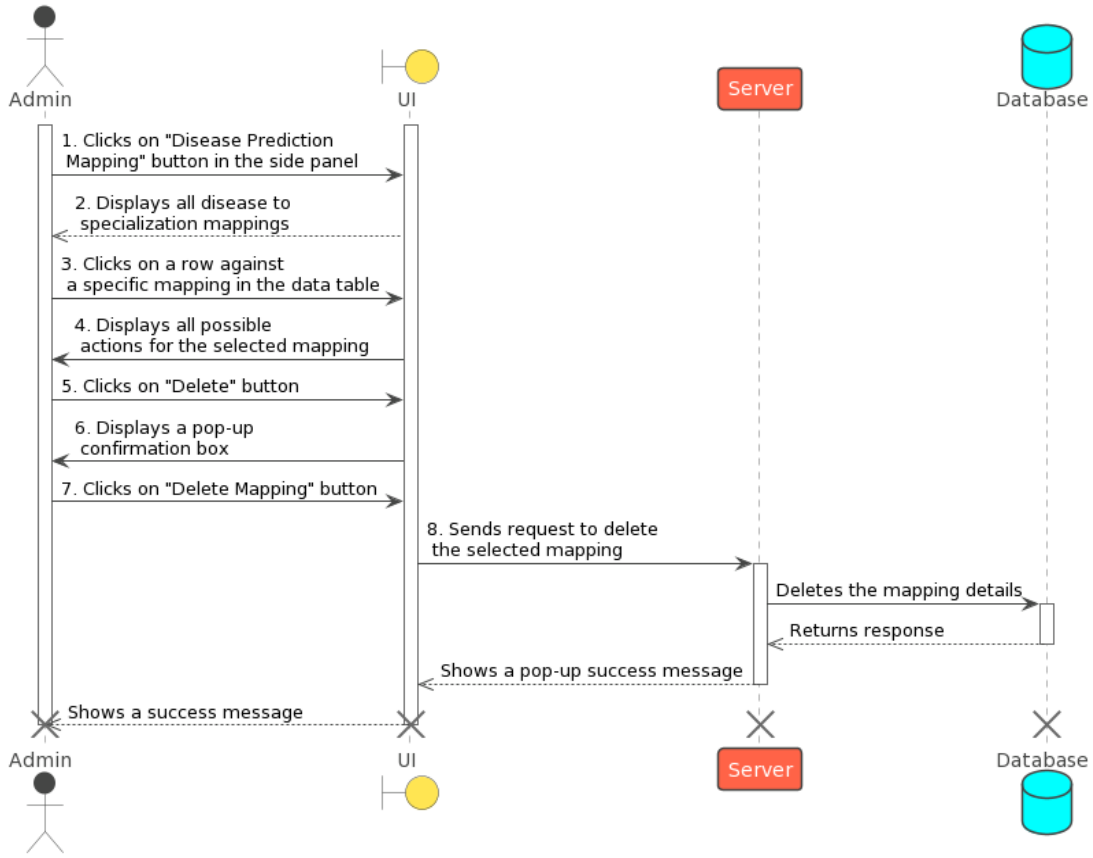

Figure 3.43: Sequence Diagram for Delete Disease and Specialization Mapping

3.4.40 Sign-out

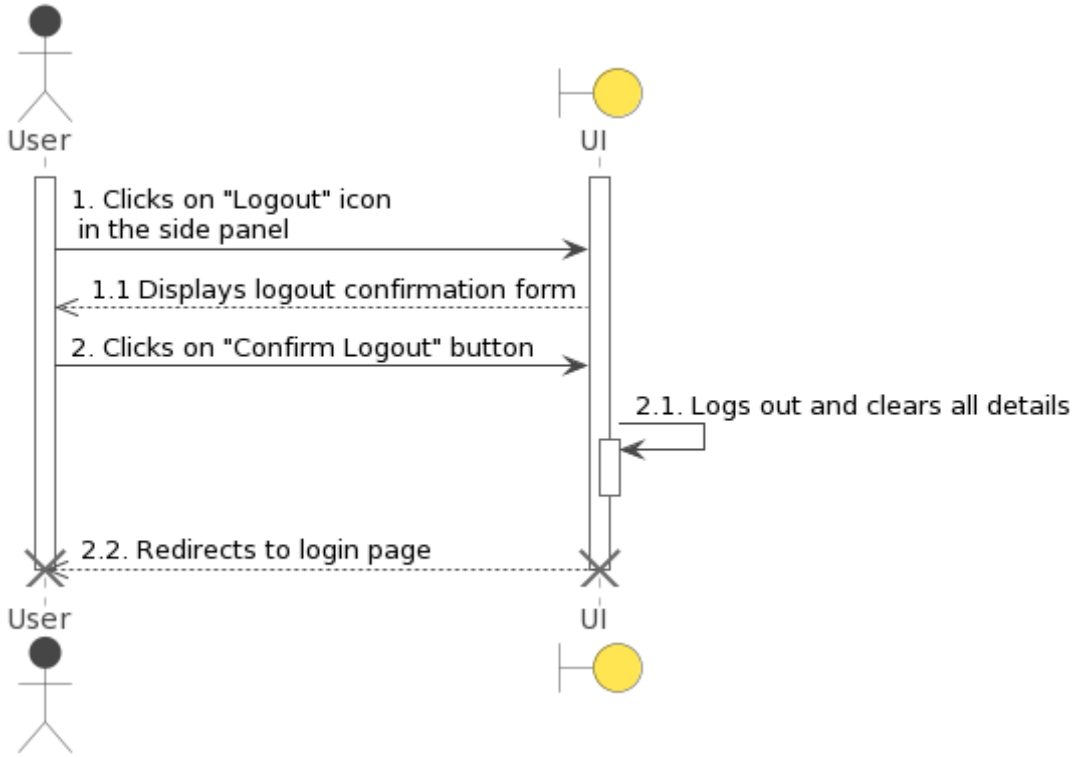

Figure 3.44: Sequence Diagram for Sign-out

3.5 ERD

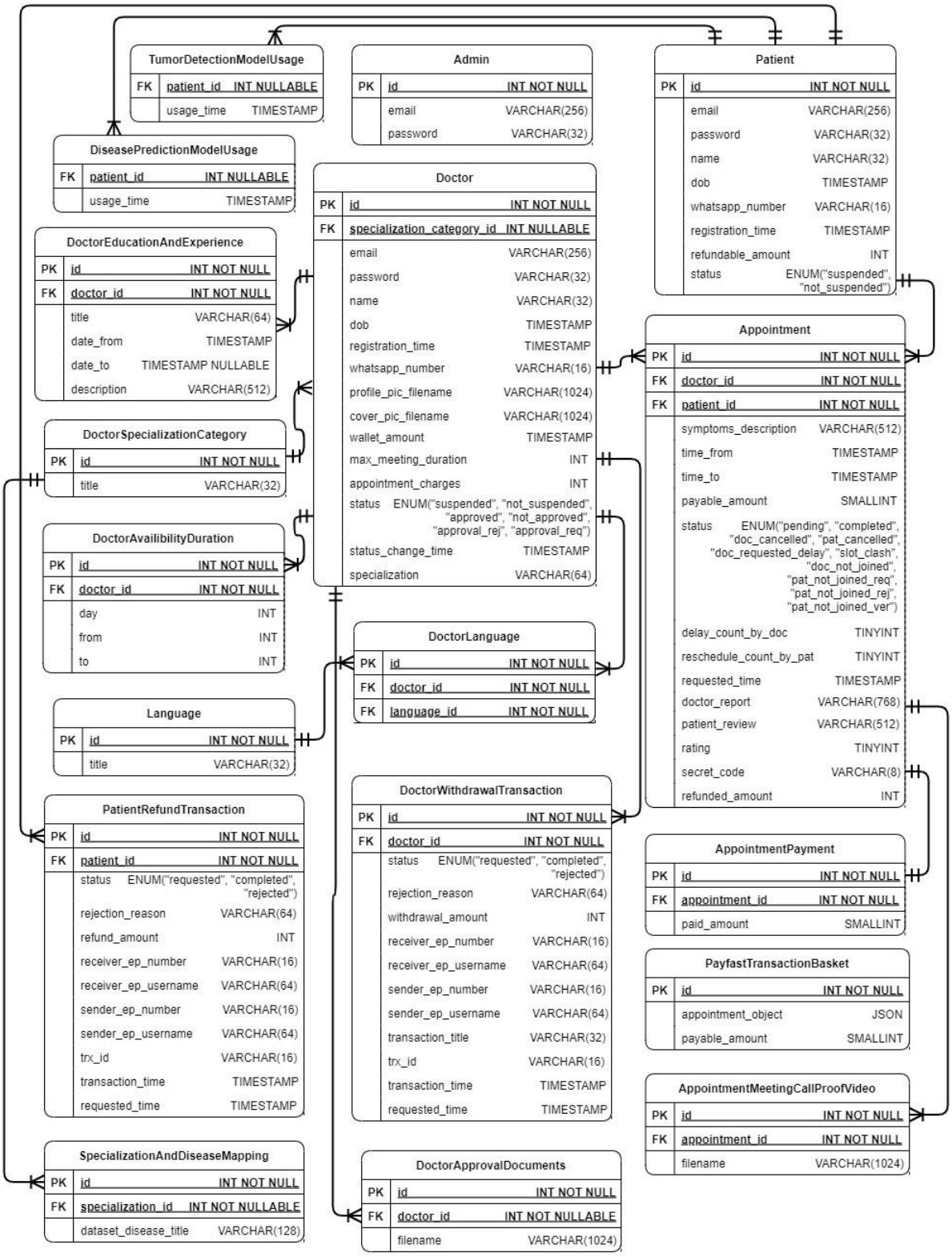

Figure 3.45: ERD Of AI Disease Predictor

References:

[1] "Disease Prediction Using Machine Learning Dataset" (Last Retrieved: October 2023). Retrieved from https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning/?select=Training.csv

[2] "Disease Symptom Prediction" (Last Retrieved: October 2023). Retrieved from https://www.kaggle.com/code/saraosmanbaza/disease-symptom-prediction-ml-99

[3] "Brain Tumor Classification (MRI)" (Last Retrieved: September 2023). Retrieved from https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri?select=Training

[4] "JEVELIN Website design inspiration" (Last Retrieved: October 2023). Retrieved from https://jevelin.shufflehound.com/medical/#contacts

[5] Sneha Grampurohit, Chetan Sagarnal, "**Disease Prediction using Machine Learning Algorithms**", International Conference for Emerging Technology (INCET), 10.1109/INCET49848.2020.9154130 IEEE (January 2020), pp-19887595.

[6] G. Hemanth, M. Janardian, L. Sujihelen, "**Design and Implementing Brain Tumor Detection Using Machine Learning Approach**", 2019 International Conference on Trends in Electronics and Informatics (ICOEI), 10.1109/ICOEI.2019.8862553 IEEE (April 2019), pp-19127324.