CS 319 - Object Oriented Software Engineering

Instructor: Eray Tüzün, TA: Muhammad Umair Ahmed

BilHealth

Analysis Report

Iteration 1

Mehmet Alper Çetin
21902324

Vedat Eren Arıcan
22002643

Uygar Onat Erol
21901908

Recep Uysal
21803637

Efe Erkan
21902248

March 20, 2022

Contents

1 Introduction

Our project is a health center management software, built in the form a web application. The main goal is to provide online, remote attention to patients to increase productivity for all parties involved. The primary method of interaction is through *case*s, which contain all relevant information for a given medical situation. Patients use the system to open cases and request appointments, and the health center staff acts on these requests to provide medical services.

Note that all diagrams on this document are in vector format, meaning that you can zoom in without any decrease in quality.

2 Current System

Our team has visited the Bilkent University health center to inquire about the application domain, along with the existing system they have in place. Our initial assumptions were then altered to respect their practical requirements.

The people interviewed were a staff member of the secretary, the head nurse, and the director of the health center. Consequently, below is a concise list of our findings.

- New patients are first examined by a nurse, after which the patient may be triaged to a medical specialist.
- During triaging, the nurse may measure the patient's heart rate, body temperature, blood pressure, etc. These details are naturally provided to the specialist who provides medical care to the patient.
- Triaging is not necessary for patients whose concerns fall into the psychological and dental domains.
- Appointments by phone are not available to students, as many do not show up to their appointments.
- Anyone who enters the health center for an examination has their records created in the system.
- Their system is in no way connected to the state's health system, named *E-Nabız*.
- Medical test results are stored in the system. The system is directly integrated with the devices carrying out the tests.
- The planning of the materials, budget, and such, takes into account the data stored in their patient record system.
- Personal medical records are kept in detail, including vaccination cards.
- Personal medical records are visible solely to the medical staff member who is responsible for the patient.

- The primary patient base is comprised of students. Those who stay in the university's housing, or are a part of the university personnel, are secondary. Needless to say, health care is given to anyone in a moment of emergency.
- As opposed to a patient's personal medical records, the medical staff is able to share notes about a particular visit, viewable by other medical staff.
- Patients reserve the right to choose their own caregivers. In any other case, the nurse is responsible for the triaging.

3 Proposed System

Our design was created with the context of Bilkent University in mind, as made clear by the project name. Some of our assumptions include the system maintained by the Bilkent Computer Center (BCC). For instance, we have assumed that majority of the patient base could be created through the general student records kept by the institute, and as such have forgone a public registration endpoint.

3.1 Actors

Health system management is a complex domain with many actors involved. Based on the current systems used, we have created five actor types to interact with the system in various capacities.

- **Nurse:** Part of the medical staff that can record patient data and forward patients to doctors.
- **Doctor:** Part of the medical staff that can undertake cases opened by patients.
- **Staff:** Part of the health center staff, but not for medical tasks, and as such cannot accept cases. They are mostly able to perform a subset of doctor and patient features on their behalf, so as to manage the process.
- **Patient:** Able to open cases and receive medical attention as a result of their interaction with the system.
- **Admin:** Manages the system but is not necessarily a part of the health center staff. This is a role for developers/maintainers of the system to do housekeeping, such as mass registering new users.

3.2 Functional Requirements

We have split up our functional requirements into actor types, to ease interpretation.

Nurse:

- A nurse, upon the initial visit to an open case, should be able to request a forward/triage to a specific doctor
- A nurse should be able to record patient details during the visit, such as heart rate, blood pressure, and body temperature
- A nurse should be able to update patient profile details

Doctor:

- A doctor should be able to receive open cases and provide medical attention online then on premise through appointments
- A doctor should be able to close cases when there is no longer a medical situation needing attention
- A doctor should be able to record notes about a patient's visit, viewable by other doctors
- A doctor should be able to follow up online on patients through the cases they have opened
- A doctor should have a profile with relevant information such as area of medical expertise
- A doctor should be able to add prescriptions to cases and generate printable documents for pharmacies
- A doctor should be able to approve appointment requests in cases, or cancel approved appointments
- A doctor should be able to make site-wide announcements
- A doctor should be able to request an inclusive anonymous report of campus-wide patient data

Staff:

- A staff member should be able to view cases, but without the ability to provide any medical assistance
- A staff member should be able to open cases on behalf of patients, and close them when necessary
- A staff member should be able to approve triaging requests made by nurses
- A staff member should be able to update patient profile details

- A staff member should be able to upload test result PDFs onto the system
- A staff member should be able to make site-wide announcements
- A staff member should be able to request an inclusive anonymous report of campus-wide patient data

Patient:

- A patient should be able to open a case to seek medical attention by specifying concerns
- A patient should be able to request appointments through the case they've opened
- A patient should be blacklisted from requesting an online appointment, if they repeatedly do not show up to their appointments
- Each appointment should contain a visit that holds information such as heart rate, body temperature, blood pressure, etc.
- A patient should be able to follow up with cases by adding messages onto the open case
- A patient should have a medical profile with medical history and relevant information such as past cases, physical measurements, vaccination history, etc.
- A patient should automatically have their medical profile made visible to the medical staff member responsible for their case
- A patient should be able to view their medical test results online
- A patient should be able to view announcements made by the health center
- A patient should receive notifications about updates to their cases, through the website or email
- A patient should be able to request a machine-generated report of personal medical history
- A patient should be able to use the service to do basic medical calculations such as BMI measurements

Admin:

- An admin should be able to register new users, including mass registering them
- An admin should be able to monitor the system by having access to most parts of the system views

3.3 Non-functional Requirements

Our non-functional requirements are as follows:

- **Security:** Since a person's health details are highly privacy sensitive, the system should protect a patient's records from any unauthorized activity
- **Usability:** The system and its user interface should be easily usable by all actors in order to achieve its goal of increasing productivity
- **Reliability:** The users' expectation of the system to be predictable in its reactions to user interaction should be satisfied
- **Scalability:** The system should be able to comfortably support a user base of up to 20000 users, with leeway for more
- **Maintainability:** The system should be implemented in a way that is practical to maintain
- **Deployment:** The system should not be tightly coupled with its hosting environment, to be able to deploy on most servers with ease
- **Extensibility:** The system should be implemented in a way that allows developers to extend functionality without refactoring majorly
- **Fault tolerance:** The system should be able to run despite database connection errors, and notify maintainers and users when a problem occurs
- **Integrability:** The system should be able to integrate services from another software system, such as Bilkent University's BCC infrastructure
- **Testability:** The system should be implemented in a way that allows automated testing tools to be used

3.4 Pseudo Requirements

- The system on the server side should be implemented in C#, using the ASP.NET Core library
- The user interface should be implemented in TypeScript, using React.js
- The system should use PostgreSQL as its data persistence solution

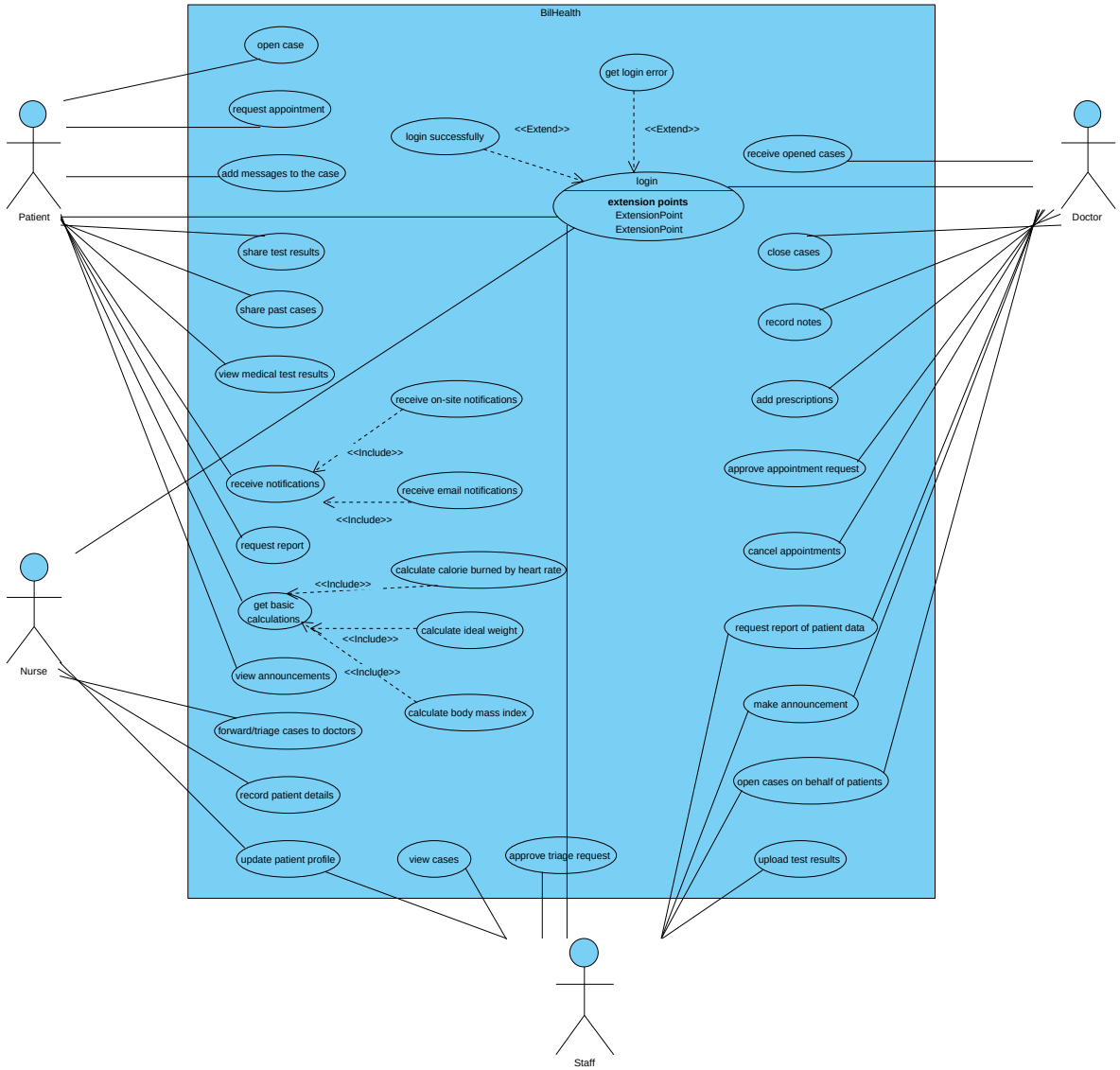
3.5 Scenarios

The average user experience on the project should be similar to the following scenario.

1. The patient logs into the system with their institute-provided credentials
2. The patient opens a case and specifies their concerns
3. The patient schedules their first appointment
4. The patient visits the health center at the time of the appointment
    - A nurse performs an initial examination, and if necessary, requests to forward the case to a specialist
    - The triage request may be approved by a staff member
5. A doctor is assigned to the case as a result of triaging
6. The patient schedules another appointment *(Repeat from step 4)*

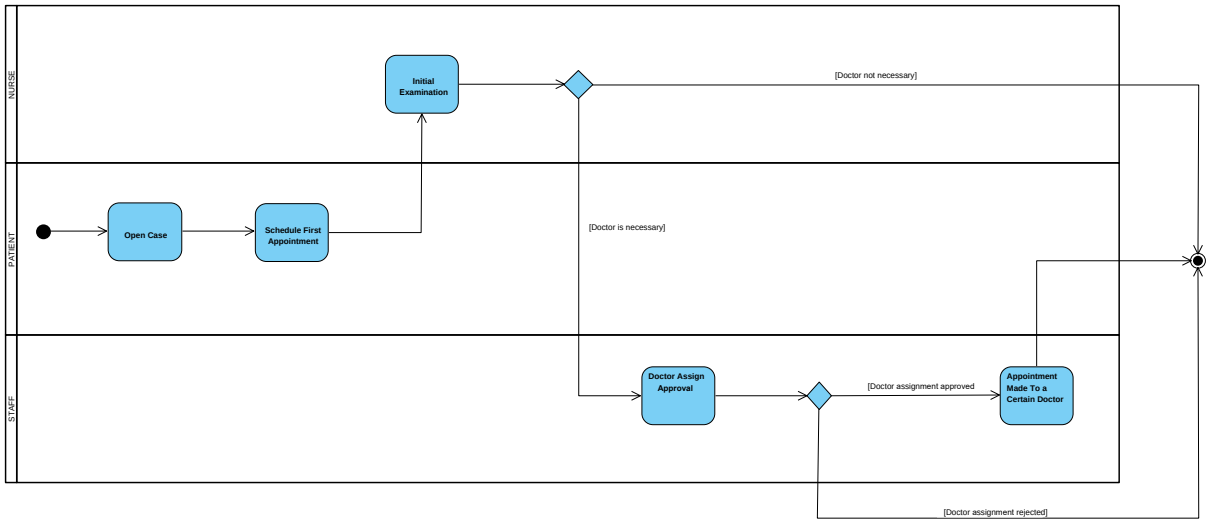
3.6 System Models

3.6.1 Use Case Diagram



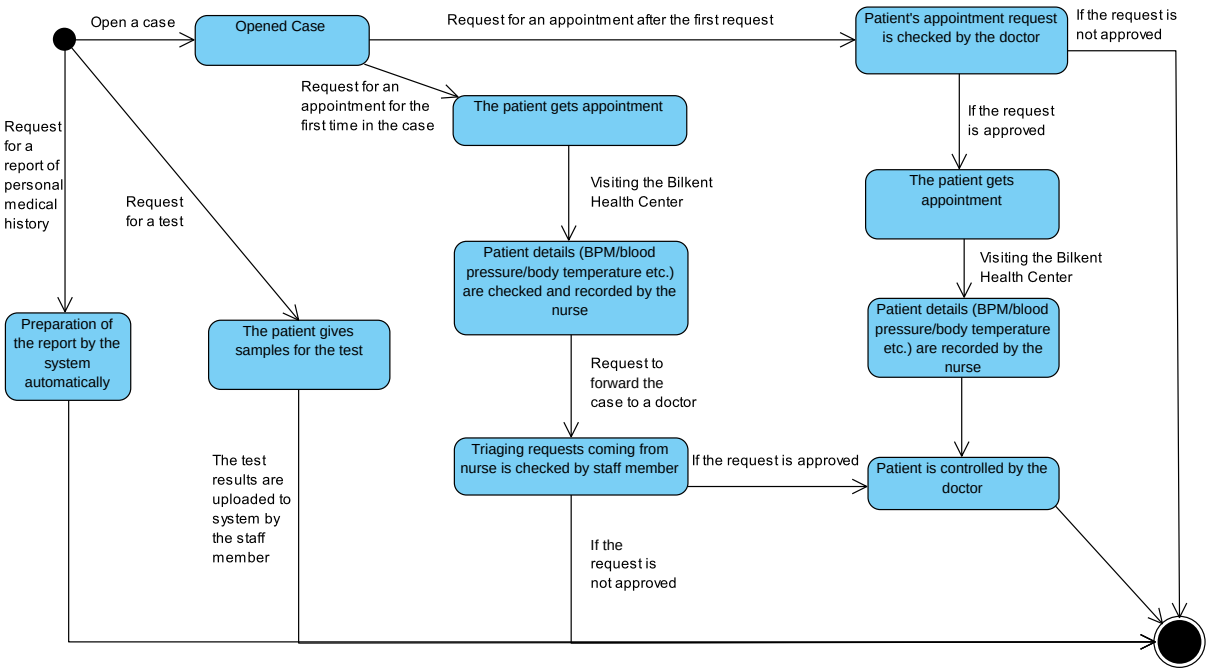
3.6.2 Activity Diagram

The following diagram shows the general flow of activities as a patient attempts to get an appointment.



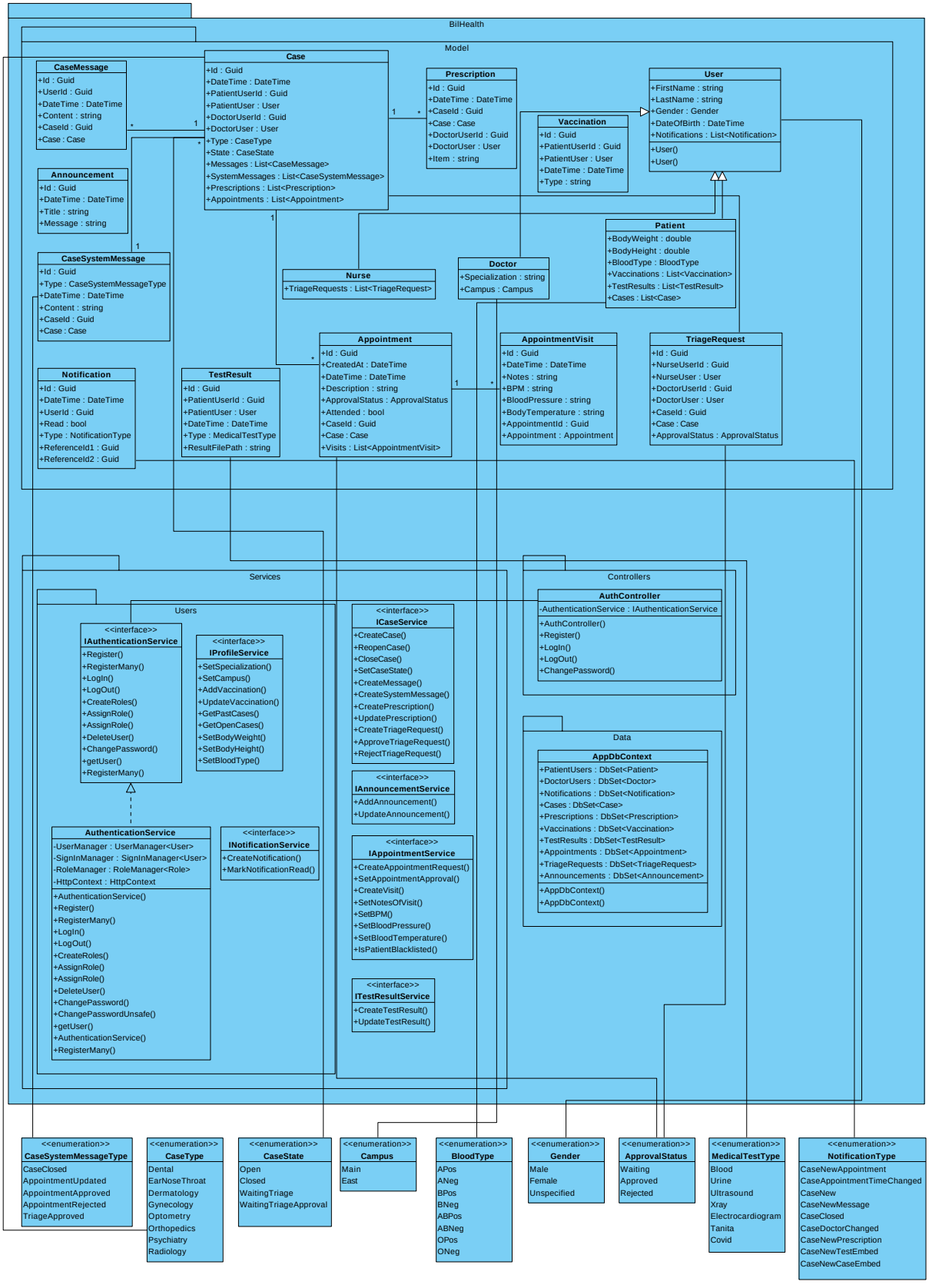
3.6.3 State Diagram

The following diagrams demonstrates the various states a case can be in at a given time.



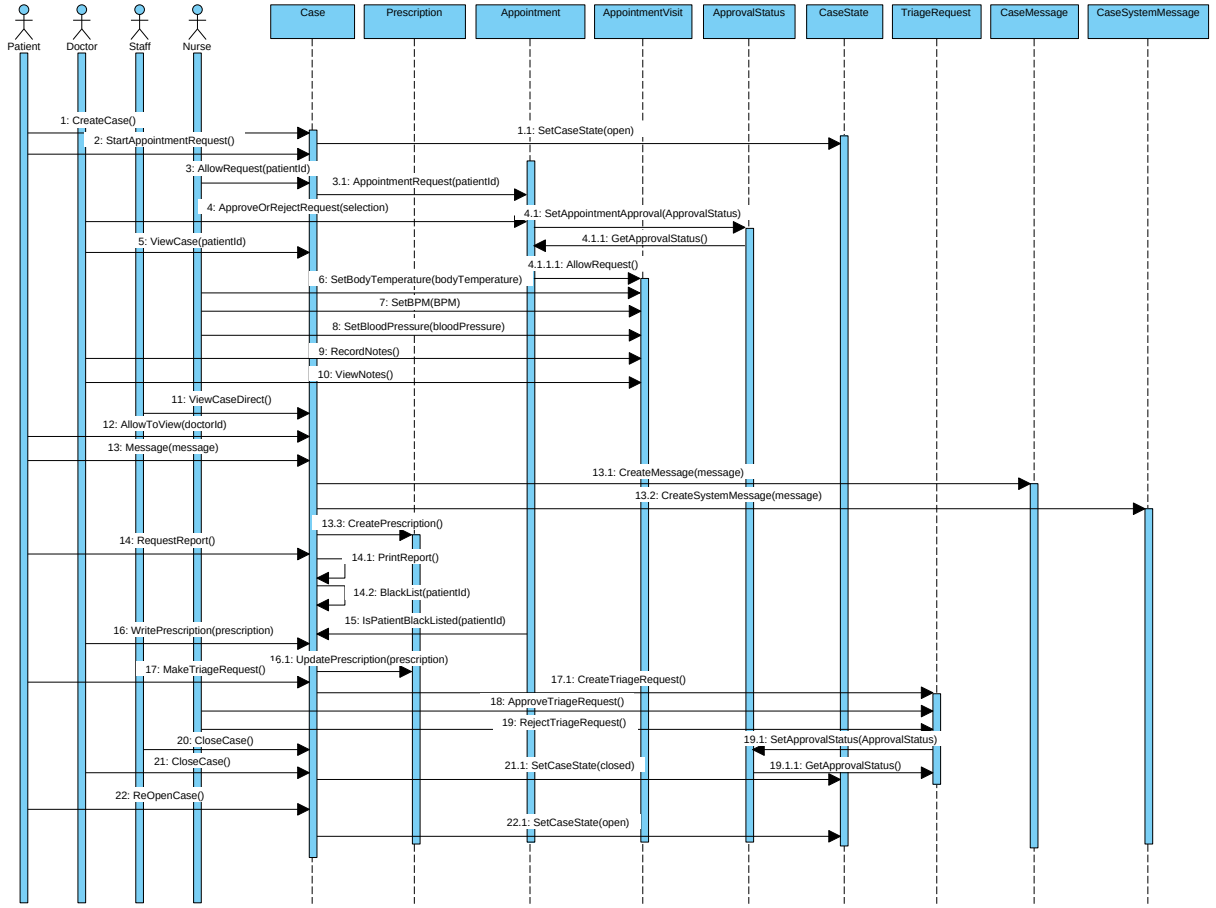
3.6.4 Class Diagram

As our project is a web application, its structure follows the model/service/controller layer paradigm. The most consequential classes are services.



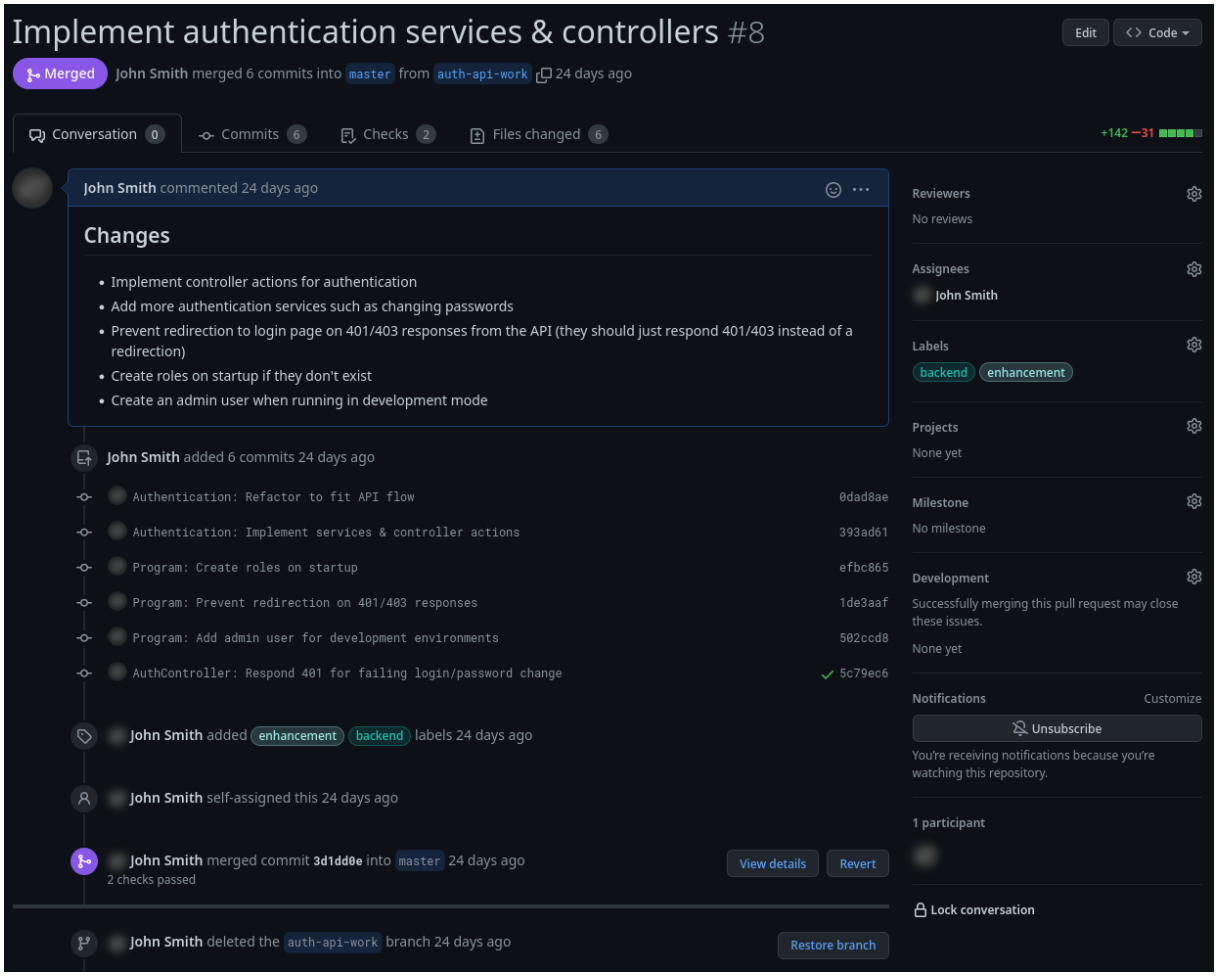
3.6.5 Sequence Diagram

The following diagram shows the rudimentary flow of method calls made during interactions with a case.



3.6.6 User Interface

The core *case* functionality of our project bears a strong resemblence to that of GitHub's issue tracker. Of course, the format GitHub uses is not unique, but we think it is a good example nonetheless. Commits can be seen as appointment requests, visits, prescriptions, and such. Messages act as communication between the patient and the medical staff. System messages are used to alert users of any changes to the events. A sidebar can be used to display the case information. As such, our final user interface is likely to look similar to the following design.



4 Glossary

- **Case:** The main point of interaction between a patient and health center staff. A case contains all the relevant information on a particular medical situation, such as appointments and prescriptions.