Implemented Features

Business Requirements	
ID	**Requirement**
BR-001	A username must be alphanumeric for registration
BR-002	A password must contain at least 8 characters.
BR-003	All doctors must have a location.
BR-005	All actors must have a name.
BR-006	An appointment with a doctor is of 30 minutes duration.
BR-007	A Doctor must have office hours.(week granularity)
BR-010	A Doctor must inform the admin to approve his registration outside the system.

User Requirements	
ID	**Requirement**
UR-001	As an admin,I should be able to approve a new Doctor request
UR-002	As an admin, I should able to delete a Doctor.
UR-004	As a Patient, I should be able to search for doctors by their name, specialization and location so that I can choose the a doctor according to my needs
UR-005	As a Patient, I should be able to schedule an appointment with the doctor so that I can consult the doctor.
UR-006	As a Patient, I should be able to view my appointment schedule so that I can keep track of my appointments
UR-007	As a Doctor, I should be able to view the pending appointment requests from all the patients so that I

	can choose to approve/ reject the appointments.
UR-008	As a Doctor, I should be able to view the approved appointments so that I can take a look at upcoming appointments
UR-011	As a Patient, I should be able to view available slots before booking an appointment.
UR-012	As a new Patient, I should be able to register on the system so that I can access the system
UR-013	As an Admin, I should be able to view all the registered doctors to the system so that i have a collective view of all the doctors.
UR-014	As an admin, I should be able to delete a doctor so that i can remove them from the system.

Functional Requirements	
ID	**Requirement**
FR-001	As a system, a time slot for a doctor is blocked for further appointments if another patient has already taken that slot
FR-002	The system should mark the initial status of all the appointments as “pending”
FR-003	As a system, access should be granted only for actors who login.

Non-Functional Requirements	
ID	**Requirement**
NFR-001	**Reliability** -All the data entries must be stored in a persistent and reliable manner
NFR-002	**Security** - User account information and password must be stored in a secure manner

NFR-003	**Performance** - The search results should be displayed within 10 seconds upon providing the search criteria.
NFR-004	**Platform constraints** - Functionality of the system should be same on all the platforms(Windows/Mac OS)

Features Not Implemented

Business Requirements	
ID	**Requirement**
BR-004	A Doctor can have none or multiple specializations
BR-008	All Patient can schedule only one appointment with a particular Doctor on a particular day.
BR-009	The permission should not be allowed to edit his health record.

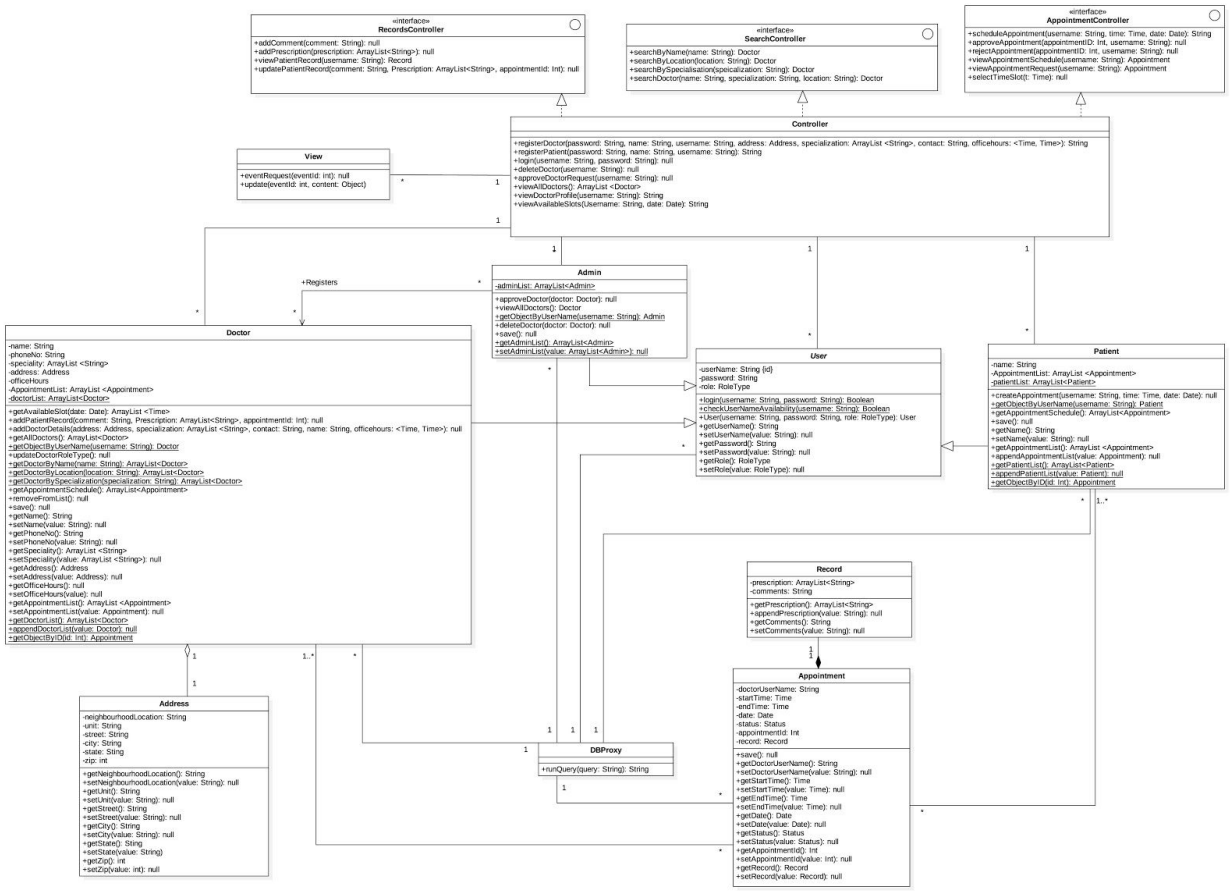
User Requirements	
ID	**Requirement**
UR-009	As a Doctor, I should be able to update the Patient's health record for an appointment so that patient can get the diagnosis and prescription.
UR-010	As a Patient, I should be able to view my records so that I can keep a track of my health

Functional Requirements	
ID	**Requirement**
FR-004	The system marks the final status of the appointment as “Completed” when the records are updated.

3. Part 2 Class diagram

Link:

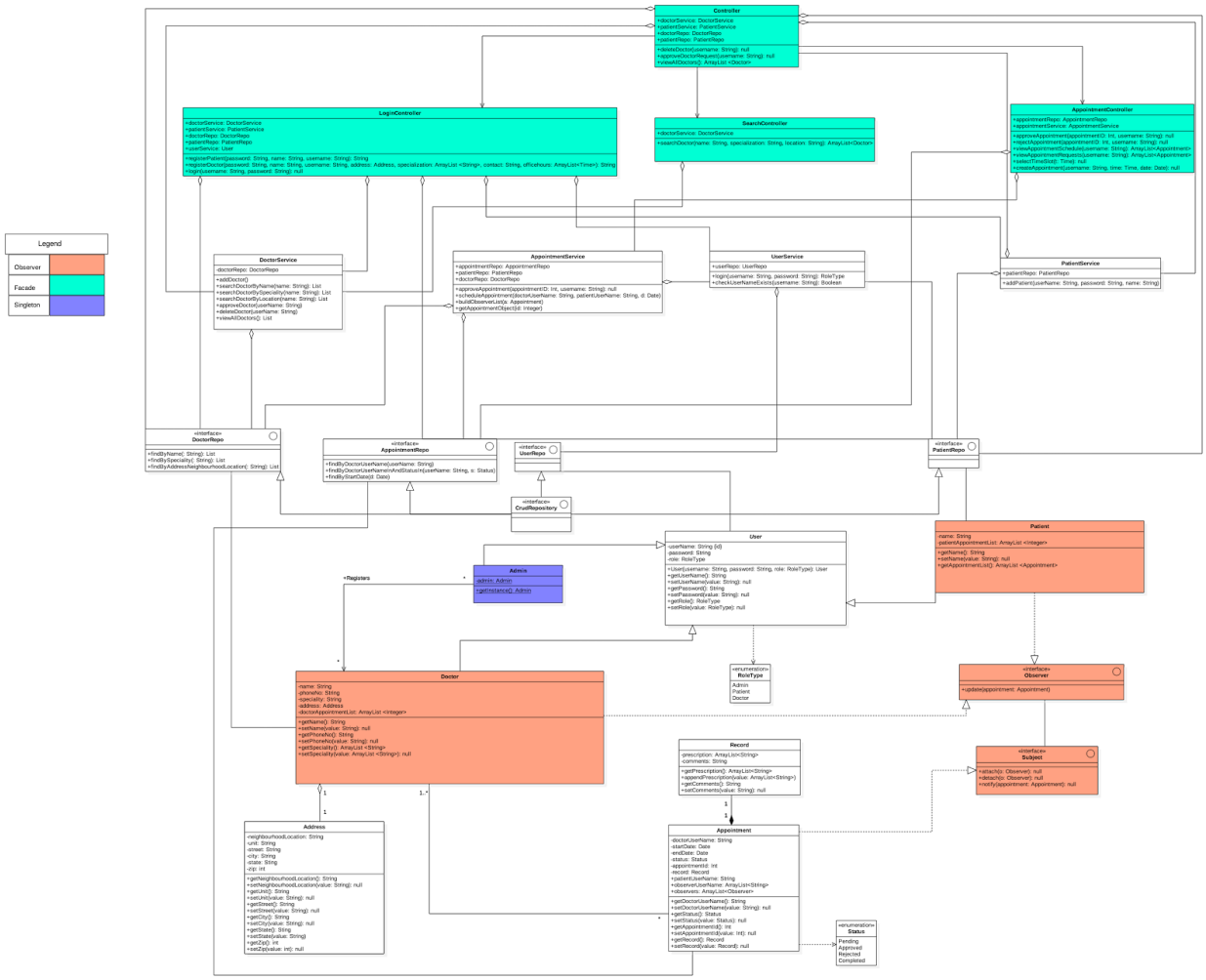
https://raw.githubusercontent.com/prashilbhimani/SpotTheDoc/master/initialClassDiagram.jpg



Final Class Diagram

Link:

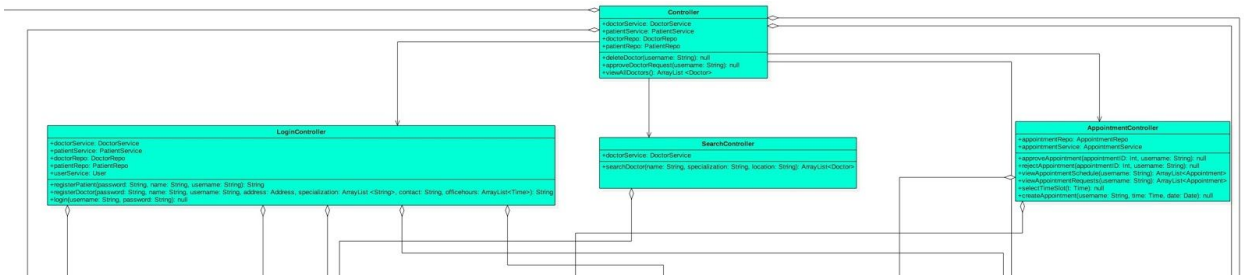
https://raw.githubusercontent.com/prashilbhimani/SpotTheDoc/master/FinalClassDiagram.jpg



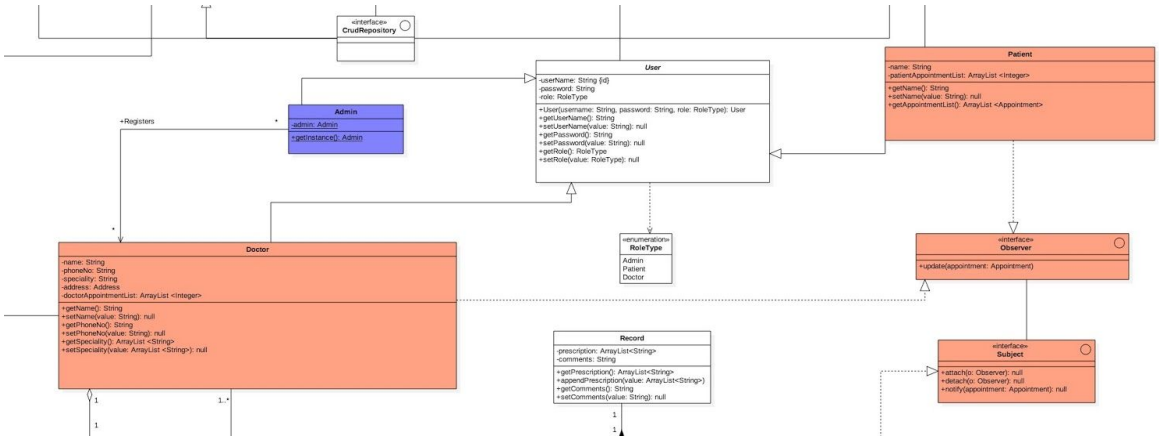
We decided to implement MVC using spring boot framework which led to a few changes in the class diagram.With the use of Spring MVC,we had to split the existing main controller into sub controllers.In order to support MVC framework,service and repository classes were included for every model class.Implementing MVC helped in increasing the cohesion within classes.Spring boot helped in achieving loose coupling through dependency injection.The Spring framework has ORM module which gives a high level abstraction for object-relational mapping API such as JPA which is used in our project.

4. With the use of MVC ,many GOF design patterns like Template,factory were naturally incorporated in the system.To optimise the functionality, we implemented Observer and Singleton pattern.

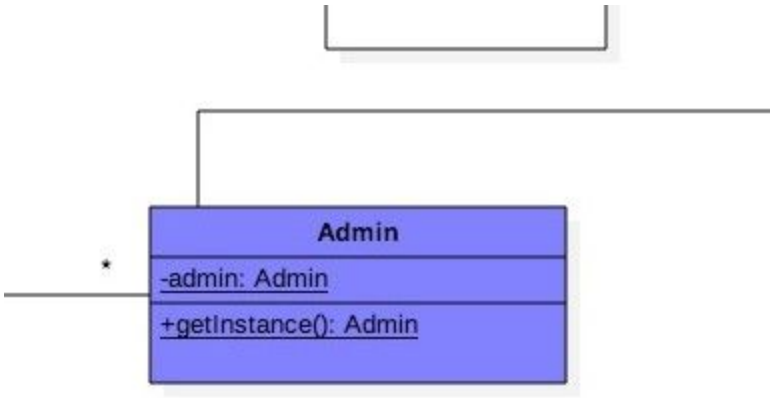
Facade Pattern



Observer Pattern



Singleton Pattern



5.With every phase in the process of analysis and design,we learned the following:

Requirements gathering

- The reason for building the system.
- What the system should do?
- What resources are required in building the system

System Design - Every diagram helped us in understanding different aspect of development

- Use case diagram helped to better understand the different actors involved and their interactions with the system.
- Activity diagram helped to understand the business flow and gave a clear picture of how the requirements would be implemented.
- Class diagram helped in establishing a base for the implementation of the system.
- Sequence diagram helped in refining the class diagram to better match the business flow.

Refactoring

- The class diagram was improvised to use the design patterns which helped in optimising by removing duplicate code and the various code smells.