Fpt University

TRƯỜNG ĐẠI HỌC FPT

CAPSTONE PROJECT REPORT

EasyDoctor - An e-health platform that connects patients and doctors

Report 4 – Software Design Document

– Danang, August 2022 –

Table of Contents

I. Record of Changes

Date	A* M, D	In charge	Change Description
		Vẽ	https://app.diagrams.net/#G1eC63BqGXqGT44s4HO3KrjALs96-CFH6X
		Github code	https://github1s.com/huyhue/EasyDoctor
		Tham khảo	https://drive.google.com/drive/folders/1Z4mBotHtAHgDyyIlkQFQa8y9A1LsdOZ7
		Học youtube	https://www.youtube.com/watch?v=soadc5aXU1c&fbclid=IwAR1pv7H86PRfVNE4EghUv_5BPZDNO66XUIzxz14HdeDTbq2UoSquxJTn3XI
		WEB	https://easydoctors.herokuapp.com/
		VIDEO	https://www.youtube.com/playlist?list=PLCA6YmLMBvjjTTzTtruiUcNRzqlG_wEud

*A - Added M - Modified D - Deleted

II. Software Design Document

1. System Design

1.1 Overall Description

We build the EasyDoctor platform using some technologies such as SpringBoot, Thymeleaf, WebSocket and MySQL database.

On the client side, we use SpringBoot framework which provides short page load time and fast scraping of data retrieved from users. Users can find Doctor information, disease, medical knowledge quickly and with less performance risk. In addition, Bootstrap and Thymeleaf used to build the front-end.

As for the server side, the EasyDoctor system has been developed based on the Java web framework Spring Boot, the database management system MySQL is a database management system that allows you to manage relational databases. They are popularly used due to its outstanding high-performance features.

1.2 System Architecture

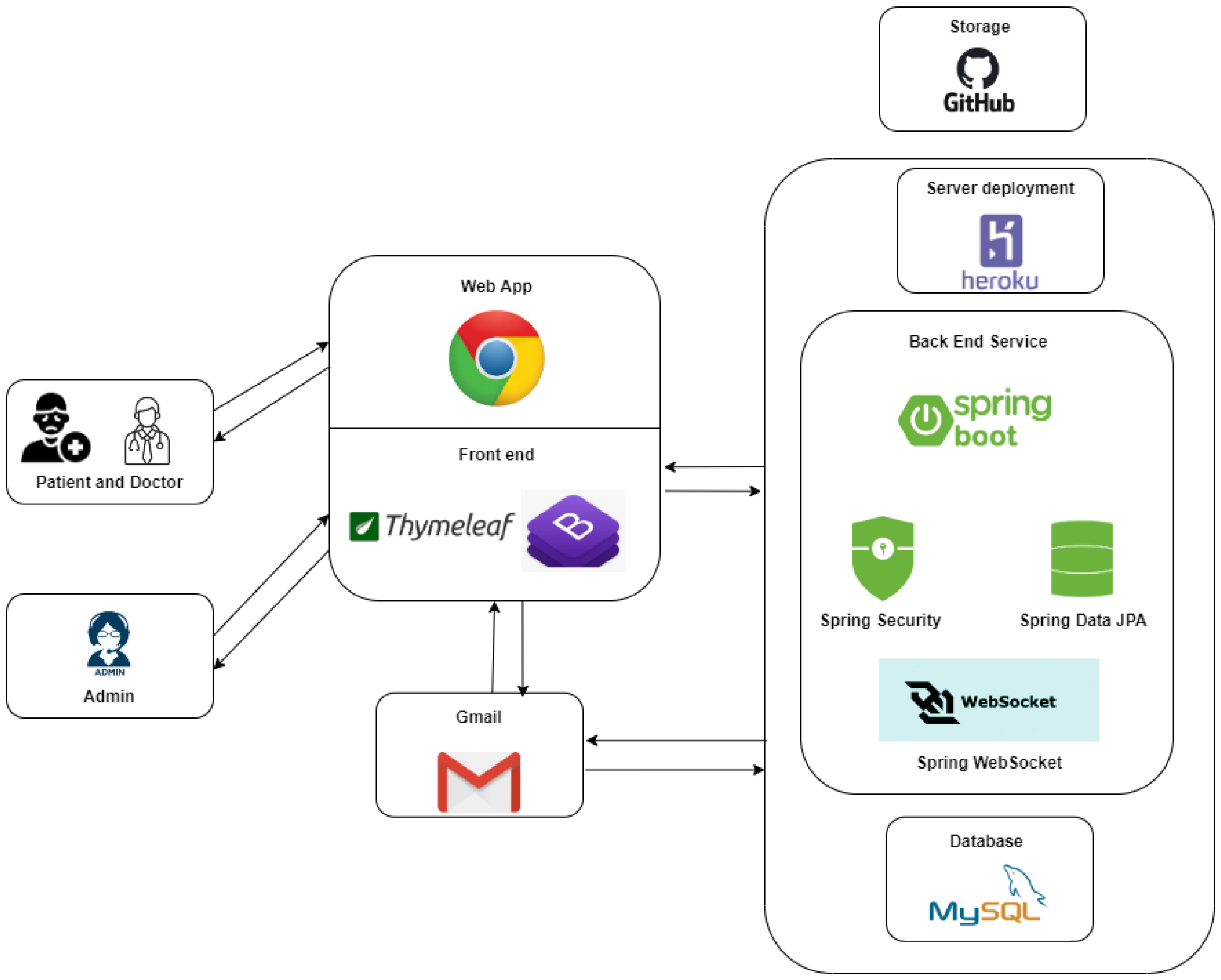

Figure 1: System Architecture

1.2.1 Back-end

EasyDoctor system back-end is built with Spring Boot version 2.4.4 with some advantages as below

- Fast and easy development of Spring-based applications.
- No need for the deployment of war files.
- The ability to create standalone applications.
- Helping to directly embed Tomcat into an application.
- Reduced amounts of source code.

1.2.1.1 Spring Boot Application Architecture

Spring Boot Application Architecture is an open-source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase.

We are going to use a three-layer or three-tier architecture to develop our spring boot application. We have a controller layer, service layer, and DAO or repository layer

● Controller layer is used to develop arrest APIs and we can call the controller layer an API layer because we basically create a spring MVC controller and we keep all the rest APIs in the controller class.

● In the service layer, we keep all our business logic.

● In the DAO layer, we keep our database-related logic or persistence logic, and the DAO

layer is responsible for communicating with the database.

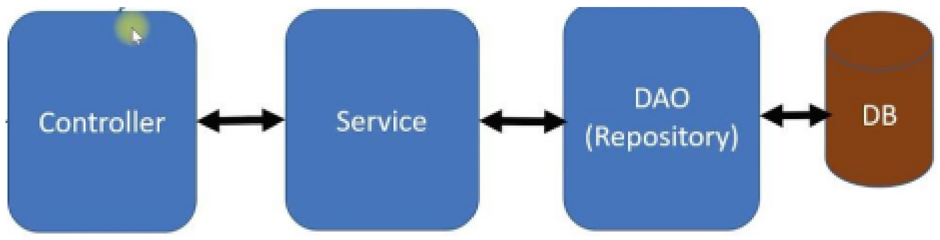

Figure 2: Spring Boot App

1.2.1.2 System Architecture Explanation

1.2.1.2.1 Spring Boot

Spring Boot is an open source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase. Spring Boot makes developing web applications and microservices with Spring Framework faster and easier through three core capabilities:

- Autoconfiguration: this means that applications are initialized with pre-set dependencies that you don't have to configure manually.
- An opinionated approach to the configuration: it uses an opinionated approach to adding and configuring starter dependencies, based on the needs of your project.
- The ability to create standalone applications: you can launch your application on any platform. It lets you create standalone applications that run on their own, without relying on an external web server, by embedding a web server such as Tomcat into your app during the initialization process

1.2.1.2.2 Spring JPA

Spring Boot JPA is a Java specification for managing relational data in Java applications. It allows us to access and persist data between Java objects/ classes and the relational database. JPA follows Object-Relational Mapping. It is a set of interfaces. It also provides a runtime EntityManager API for processing queries and transactions on the objects against the database. It uses a platform-independent object-oriented query language JPQL (Java Persistence Query Language).

JPA is widely used in because:

- It is simpler, cleaner, and less labor-intensive than JDBC, SQL, and hand-written mapping.
- It allows mapping in XML or using Java annotations.
- When we need to perform queries using JPQL, it allows us to express the queries in terms of Java entities rather than the (native) SQL table and columns

1.2.1.2.3 Spring Security

Spring Security is a framework that provides authentication, authorization, and protection against common attacks. With first-class support for securing both imperative and reactive applications, it is the de-facto standard for securing Spring-based applications.

The benefits of Spring Security are not limited to helping us with Authentication and Authorization. It can also help us to apply best practices in saving users, and building a sign-up feature.

1.2.1.2.4 My SQL

In our project, we use the MySQL database. MySQL is a database management system that allows you to manage relational databases. It is open-source software backed by Oracle. A is popularly used for its outstanding features:

- High Performance: MySQL can meet the performance of a high-volume website that services a billion queries a day.

- Web and Data Warehouse Strengths: MySQL is the de-facto standard for high-traffic websites because of its high-performance query engine, tremendously fast data insert capability, and strong support for specialized web functions like fast full-text searches.
- Strong Data Protection: MySQL offers exceptional security features that ensure absolute data protection.
- Comprehensive Application Development: MySQL provides comprehensive support for every application development need.
- Management Ease: MySQL offers the exceptional quick-start capability and a complete suite of graphical management and migration tools

1.2.2 Client

1.2.2.1. Web Application Architecture

1.2.3 Front End

In our product, Web Application is designed by Thymeleaf. Below are its advantages:

- Provides an elegant and well-formed way of creating templates.
- High integration with Spring Framework.
- Bring elegant natural templates to development workflow — HTML that can be correctly displayed in browsers and also work as static prototypes.

1.2.3.1 HTML and CSS

HTML, HyperText Markup Language, provides content structure and meaning by defining it, for example, a title, paragraph, or image.

CSS, or Cascading Style Sheets, is a presentation language used to style the appearance of content using, for example, fonts or colors.

1.2.3.2 Bootstrap

Bootstrap allows the website design process to be faster and easier based on the basic elements available such as typography, forms, buttons, tables, grids, navigation, image carousels...

Bootstrap is a free collection of open source tools and tools for creating a complete website template. With predefined interface properties such as size, color, height, and width..., designers can create many new products but still save time when working with this framework in the process of web interface design.

1.2.3.3 Thymeleaf

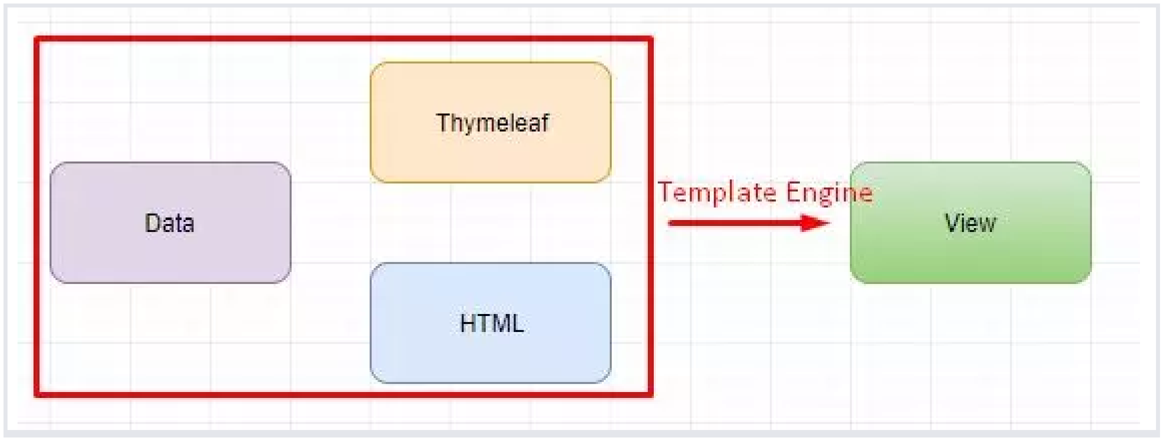

Figure 3: Thymeleaf

Thymeleaf is a Java template engine used to process and generate HTML, XML, Javascript, CSS, and text. The main goal of thyme leaf is to bring natural, uniform, and simple templates to development work.

Benefits of Thymeleaf With thyme leaf, we can display everything just by using HTML files (no need for JSP...). Thymealeaf will participate in rendering HTML files as attributes in HTML tags --> so we don't need to add any non-HTML tags. Since it is HTML, we can view files without starting the server. Thymeleaf supports a caching mechanism, so you can cache data or custom to display the view when there is a change without restarting the server.

1.2 Package Diagram

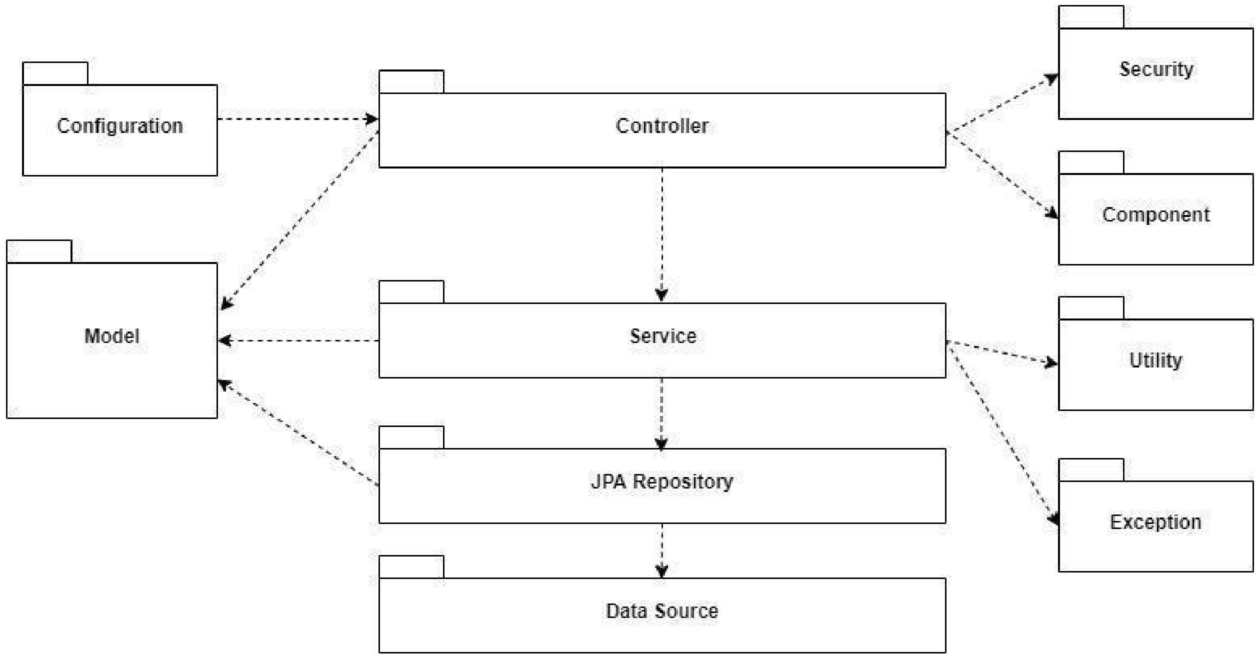

Figure 4: Package diagram

Details of each package are described in the table below:

No	Package	Description	Consist of
1	Model	Contains entity classes which are the persistence objects stored as a record in the database	Appointment, AppointmentStatus, BaseEntity, Clinic, Comment, Declaration, Doctor, FileModel, Gender, History, Invoice, Message, Notification, Packages, Post, Patient, Review, Role, Speciality, User, WorkingPlan
2	Controller	Contains controller classes which are responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response.	AdminController, AjaxController, AppointmentController, ChatController, DoctorController, HomeController, InvoiceController, NotificationController, PatientController, BlogController
3	Service	Contains services classes that are used to write business logic in a different layer, separated from controller	AppointmentService, ClinicService, DotorService, EmailService,

			InvoiceService, JwTokenService, NotificationService, OTP Service, PostService, PackagesService, ScheduleTasksService, User Service, WorkingPlanService
4	JPA Repository	Contains JPA repository interface that encapsulates the logic required to access data sources	AppointmentRespository, CommonUserRespository, DoctorRespository, ClinicRespository, FileModelRespository, HistoryRespository, InvoiceRespository, MessageRespository, NotificationRespository, PackagesRespository, PatientRespository, PostRespository, ReviewRespository, RoleRespository, SpecialtyRespository, UserRespository, WorkingPlanRespository
5	Data Source	A factory for connections to the MySql database	

6	Utility	Contains utility classes which perform common, often reused functions	PdfGeneratorUtil, Utils
7	Security	Implements authentication, authorization, and protection against common attacks	CustomAuthenticationSuccessHandler, CustomUserDetails, CustomUserDetailsService, PasswordEncoderConfig, WebSecurityConfig
8	Exception	Provides a mechanism to treat exceptions that are thrown during execution of handlers	AppointmentNotFoundException, PackagesNotFoundException
9	Configuration	Contains configuration classes which consist principally of @Bean - annotated methods that define instantiation, configuration, and initialization logic for objects that are managed by the Spring IoC container	VersionInterceptor WebMvcConfig WebSocketConfig WebSocketEventListener
10	Component	It is responsible for getting the necessary fields from the model to display on the screens.	AppoinmentDto, AppoinmentRegisterForm, AppoinmentSerializer, ChangePasswordForm, ChatMessage, CommentDTO, CommonMsg, DayPlan,

			DoctorDto, PatientDtio, PostDTO, ReviewDto, ReviewForm, TimePeriod, UserForm

Table 1: Details of each package

2. Database Design

Below is Easy Doctor system database design:

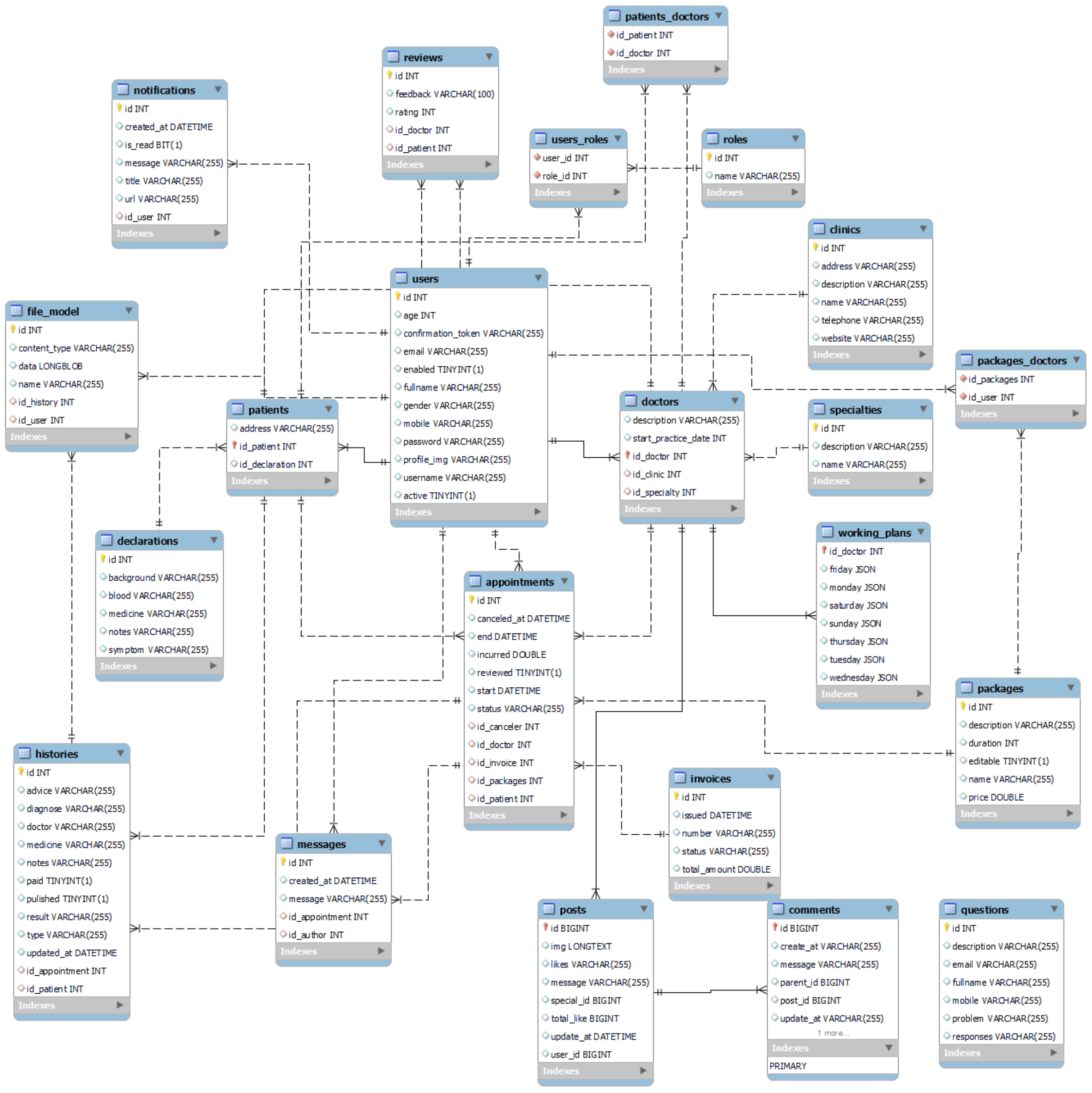

Figure 5: Database Design

The following is a detailed description of each table in the diagram:

No	Table	Description
01	**appointments**	- **Primary keys:** id INT - **Foreign keys:** +) id_canceler INT +) id_doctor INT +) id_invoice INT +) id_packages INT

		+) id_patient INT +) id_review INT - **Field name:** +) canceled_at DATETIME +) end DATETIME +) start DATETIME +) status VARCHAR(255) +) reviewed TINYINT(1) +) incurred DOUBLE
02	**users**	- **Primary keys**: id INT - **Foreign keys:** None - **Field name:** +) age INT +) confirmation_token VARCHAR(255) +) email VARCHAR(255) +) enabled TINYINT(1) +) full name VARCHAR(255) +) gender VARCHAR(255) +) mobile VARCHAR(255) +) password VARCHAR(255) +) profile_img VARCHAR(255) +) username VARCHAR(255)
03	**reviews**	- **Primary keys:** id INT - **Foreign keys:** +) id_doctor INT +) id_patient INT - **Field name:** +) rating INT +) review VARCHAR(255) +) feedback VARCHAR(100)
04	**packages_doctors**	- **Foreign keys:** +) id_packages INT +) id_user INT
05	**exchanges**	- **Primary keys:** id INT - **Foreign keys:** +) id_appointment_requested INT +) id_appointment_requestor INT - **Field name:** +) exchange_status VARCHAR(255)

06	**messages**	- **Primary keys**: id INT - **Foreign keys:** +) id_appointment INT +) id_author INT - **Field name:** +) created_at DATETIME +) message VARCHAR(255)
07	**specialties**	- **Primary keys:** id INT - **Foreign keys:** None - **Field name:** +) description VARCHAR(255) +) name VARCHAR(255)
08	**packages**	- **Primary keys:** id INT - **Foreign keys:** None - **Field name:** +) description VARCHAR(255) +) duration INT +) editable TINYINT(1) +) name VARCHAR(255) +) price DOUBLE +) target VARCHAR(255)
09	**patients**	- **Primary keys:** id_patient INT - **Foreign keys:** +) id_pathological INT +) id_declaration INT +) id_patient INT - **Field name:** address VARCHAR(255)
10	**roles**	- **Primary keys:** id INT - **Foreign keys:** None - **Field name:** name VARCHAR(255)
11	**users_roles**	- **Primary keys:** None - **Foreign keys:** +) user_id INT +) role_id INT
12	**notifications**	- **Primary keys:** id INT - **Foreign keys:** id_user INT - **Field name:** +) created_at DATETIME

		+) is_real BIT(1) +) message VARCHAR(255) +) title VARCHAR(255) +) url VARCHAR(255)
13	**doctor**	- **Primary keys:** id_doctor INT - **Foreign keys:** +) id_clinic BIGINT +) id_specialty INT +) id_doctor INT - **Field name:** +) certification VARCHAR(255) +) description VARCHAR(255) +) start_practice_date DATETIME
14	**histories**	- **Primary keys:** id INT - **Foreign keys:** +) id_patient INT +) id_appointment INT - **Field name:** +) doctor VARCHAR(255) +) result VARCHAR(255) +) start DATETIME +) type VARCHAR(255) +) advice VARCHAR(255) +) notes VARCHAR(255) +) diagnose VARCHAR(255) +) medicine VARCHAR(255) +) paid TINYINT(1) +) published TINYINT(1) +) updated_at DATETIME
15	**invoices**	- **Primary keys:** id INT - **Foreign keys:** None - **Field name:** +) number VARCHAR(255) +) status VARCHAR(255) +) issued DATETIME +) total_amount DOUBLE
16	**working_plans**	- **Primary keys:** id_doctor INT - **Foreign keys:** id_doctor INT - **Field name:** +) monday JSON

		+) tuesday JSON +) wednesday JSON +) thursday JSON +) friday JSON +) saturday JSON +) sunday JSON
17	**pathologicals**	- **Primary keys:** id INT - **Foreign keys:** None
18	**file_model**	- **Primary keys:** id INT - **Foreign keys:** +) id_history INT +) id_user INT - **Field name:** +) content_type VARCHAR(255) +) data LONGBLOB +) name VARCHAR(255)
19	**patients_doctors**	- **Primary keys:** None - **Foreign keys:** +) id_patient INT +) id_doctor INT
20	**clinics**	- **Primary keys:** id BIGINT - **Foreign keys:** None - **Field name:** +) address VARCHAR(255) +) description VARCHAR(255) +) name VARCHAR(255) +) telephone VARCHAR(255) +) website VARCHAR(255)
21	**comments**	- **Primary keys:** id BIGINT - **Foreign keys:** None - **Field name:** +) create_at VARCHAR(255) +) message VARCHAR(255) +) parent_id BIGINT +) post_id BIGINT +) update_at VARCHAR(255) +) user_id BIGINT
22	**declarations**	- **Primary keys:** id INT

		- **Foreign keys:** None - **Field name:** +) background VARCHAR(255) +) blood VARCHAR(255) +) medicine VARCHAR(255) +) notes VARCHAR(255) +) symptom VARCHAR(255)
23	**posts**	- **Primary keys:** id BIGINT - **Foreign keys:** None - **Field name:** +) create_at VARCHAR(255) +) img VARCHAR(255) +) likes VARCHAR(255) +) message VARCHAR(255) +) special_id BIGINT +) total_like BIGINT +) update_at VARCHAR(255) +) user_id BIGINT
24	**questions**	- **Primary keys:** id INT - **Foreign keys:** None - **Field name:** +) description VARCHAR(255) +) email VARCHAR(255) +) full name VARCHAR(255) +) mobile VARCHAR(255)

3. Detailed Design

3.1 Class Diagram

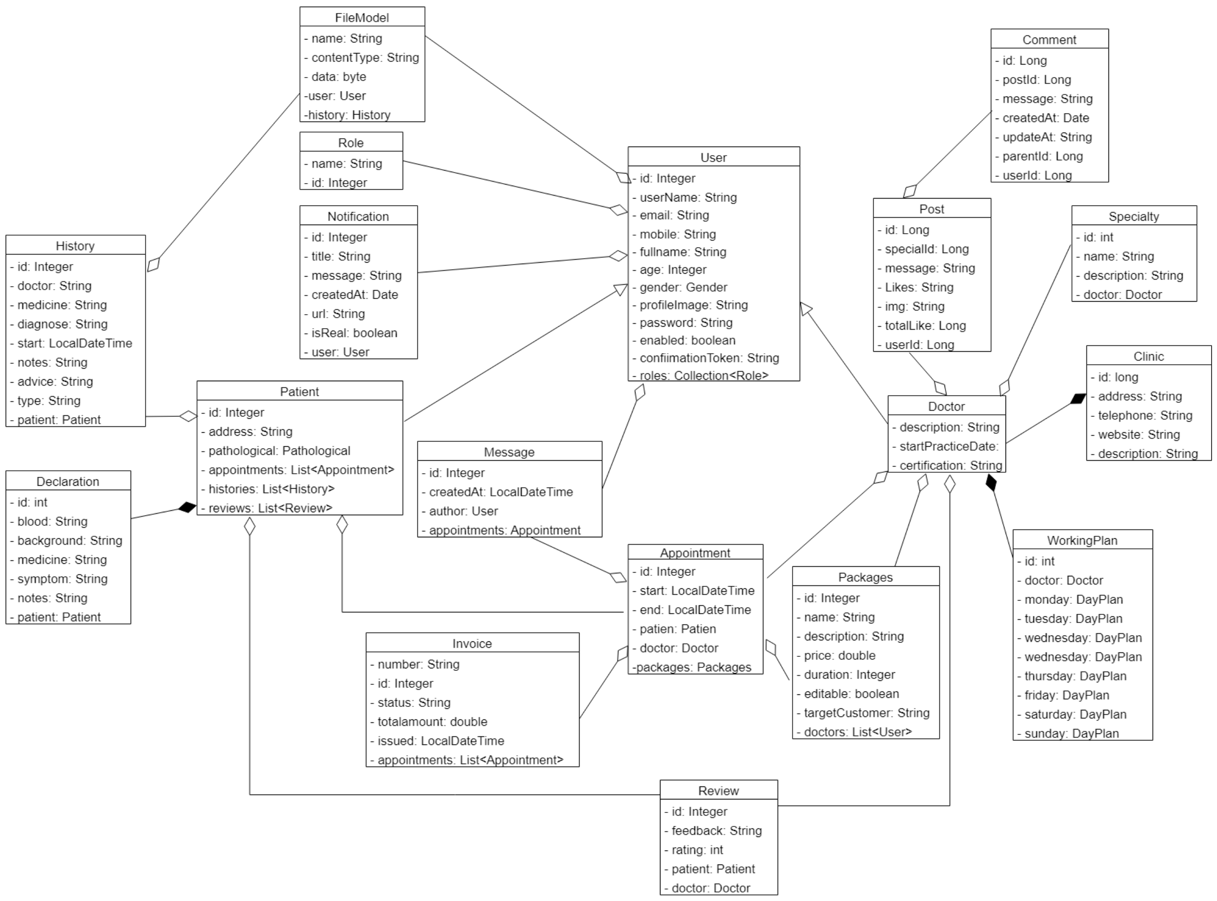

Figure 6: Class Diagram

3.2 Class Specifications

Detailed specifications for each class are shown below:

3.2.1. User Class

No	Attributes	Type	Description
1	userName	String	Username of the patient, admin, or doctor, using for login
2	id	int	Unique identifier, auto-increment
3	password	String	Password of the patient, admin, or doctor, using for login
4	email	String	Email of the patient, admin, or doctor, using for login and forgot password

5	mobile	String	Phone number of the patient, doctor
6	full name	String	The first and the last name of the patient, admin, or doctor
7	age	Integer	Age of the patient, doctor
8	profile image	String	The patient, doctor avatar
9	gender	Gender	Gender of the patient, doctor
10	enabled	boolean	Online, the offline status of user
11	confirmation token	String	Token of the patient, admin, or doctor when the password is forgotten

3.2.2. Role Class

No	Attributes	Type	Description
1	id	Integer	Unique identifier, auto-increment
2	name	String	Name of role

3.2.3. Appointment Class

No	Attributes	Type	Description
1	start	LocalDateTime	Time to start medical examination
2	end	LocalDateTime	End time of medical examination
3	canceledAt	LocalDateTime	Time when the patient, admin, or doctor cancels appointment
4	status	Object	Status of appointment
5	patient	Patient	Patient book an appointment
6	doctor	Doctor	Doctor receives appointment
7	canceler	Patient, admin, or doctor	Cancel appointment
8	packages	Packages	Examination package is booked

9	chatMessages	List<Message>	Patient chat with doctors
10	reviewed	boolean	Doctor reviews
11	invoice	Invoice	Bill payment to patient

3.2.3. Clinic Class

No	Attributes	Type	Description
1	id	Long	Unique identifier, auto-increment
2	name	String	Name of clinic
3	address	String	Address of clinic
4	telephone	String	Telephone of clinic
5	website	String	Website of clinic
6	description	String	Description about clinic

3.2.4. Declaration

No	Attributes	Type	Description
1	id	int	Unique identifier, auto-increment
2	blood	String	Blood of patient
3	background	String	Patient's underlying disease
4	medicine	String	Medicine of patient
5	symptom	String	Symptom of patient
6	notes	String	Note of patient
7	patient	Patient	Patient who creates declaration

3.2.5. Doctor Class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	Specialty	Specialty	Specialty of doctor
3	clinic	Clinic	Address of clinic
4	appointments	List<Appointment>	Appointment of patient and doctor
5	reviews	List<Review>	Review of Doctor
6	packages	List<Packages>	Packages given by the doctor
7	workingPlan	working plan	Doctor creates examination time

3.2.6. FileModel Class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	name	String	Name of file
3	contentType	String	ContentType of file
4	data	byte	Data of file
5	user	User	User create file
6	history	HIstory	Patient's previous medical history

3.2.7. History Class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	doctor	String	Doctor information
3	medicine	String	Medicine of doctor for patient
4	diagnose	String	Diagnose of doctor
5	notes	String	Note of doctor for patient

6	advice	String	advice of doctor
7	start	LocalDateTime	time to start medical examination
8	type	String	type of examination
9	patient	Patient	examiner information

3.2.8. Invoice class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	number	String	Number of invoice
3	status	String	Status of invoice
4	total amount	double	Total amount of invoice
5	issued	LocalDateTime	Issue date of invoice
6	appointments	List<Appointment>	Appointment of patient and doctor

3.2.9. Message class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	createdAt	LocalDateTime	Create message
3	author	User	Patient book an appointment
4	appointments	Appointment	Appointment of patient and doctor

3.2.10. Notification Class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	title	String	Title of notifications
3	message	String	Message notifications

4	createAt	double	Create At notifications
5	url	LocalDateTime	URL of notifications
6	isRead	List<Appointment>	Did you read the notifications?
7	user	User	User of notifications

3.2.11. Packages Class

No	**Attributes**	**Type**	**Description**
1	id	integer	Unique identifier, auto-increment
2	name	String	Name of packages
3	description	String	Description of packages
4	price	double	Price of packages
5	duration	Integer	Duration of packages
6	editable	boolean	Edit packages
7	targetCustomer	String	Patients who tend to use medical examination packages
8	doctor	List<User>	Creator of medical packages

3.2.12. Patient Class

No	**Attributes**	**Type**	**Description**
1	id	integer	Unique identifier, auto-increment
2	address	String	Address of patient
3	declaration	Declaration	Declaration of patient
4	appointment	List<Appointment>	Appointment of patient and doctor
5	Histories	list<History>	History of patient
6	review	List<Review>	Review of the patient about doctor

3.2.13 Review Class

No	Attributes	Type	Description
1	id	integer	Unique identifier, auto-increment
2	feedback	String	Comment of the patient about doctor
3	rating	int	Vote of the patient about doctor
4	patient	Patient	Who has been examined by a doctor
5	doctor	doctor	The person being evaluated

3.2.14. Specialty Class

No	Attributes	Type	Description
1	id	int	Unique identifier, auto increment
2	name	String	Name of specialty
3	description	int	Description about specialty
4	doctor	Doctor	The person examining that department

3.2.15. WorkingPlan Class

No	Attributes	Type	Description
1	id	int	Unique identifier, auto-increment
2	doctor	Doctor	Doctor creates examination time
3	monday	DayPlan	Day of the week
4	tuesday	DayPlan	Day of the week
5	wednesday	DayPlan	Day of the week
6	thursday	DayPlan	Day of the week
7	friday	DayPlan	Day of the week
8	saturday	DayPlan	Day of the week

9	sunday	DayPlan	Day of the week

3.2.16. Post Class

No	Attributes	Type	Description
1	id	Long	Unique identifier, auto-increment
2	userId	Long	Unique identifier, auto-increment
3	specialId	Long	Unique identifier special, auto-increment
4	message	String	Message post
5	img	String	Img of patient
6	likes	String	Like of user
7	totalLike	Long	Total likes of the post

3.2.17. Comment Class

No	Attributes	Type	Description
1	id	Long	Unique identifier, auto-increment
2	postId	Long	Unique identifier post, auto-increment
3	createAt	String	Create comment
4	message	String	Message comment
5	updateAt	String	Update comment
6	parentId	Long	Unique identifier parent, auto increment
7	userId	Long	Unique identifier, auto increment

3.3 Sequence Diagrams

For each function we build the sequence diagram as below:

3.3.1. General Patient and Doctor

1. Login

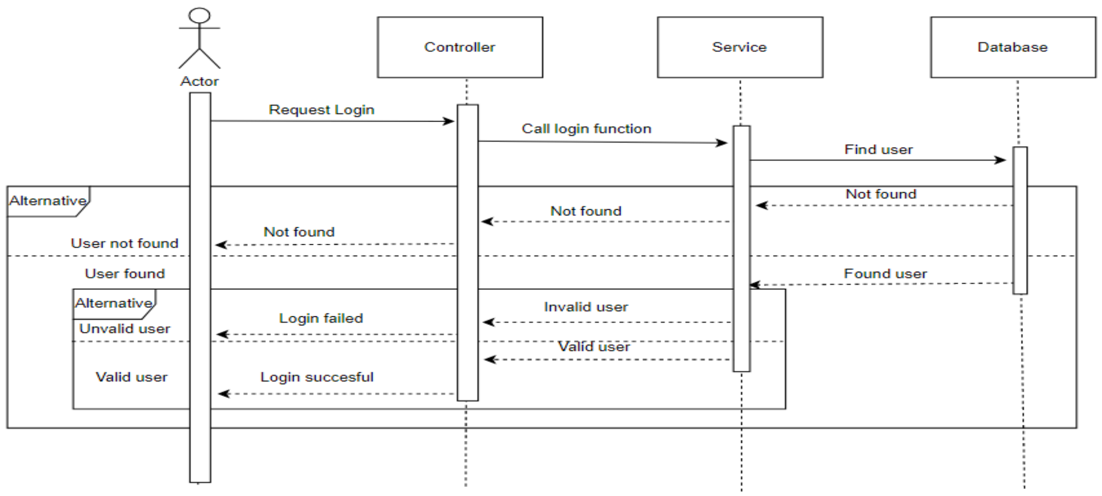

Figure 7: Login Sequence Diagram

2. Reset password

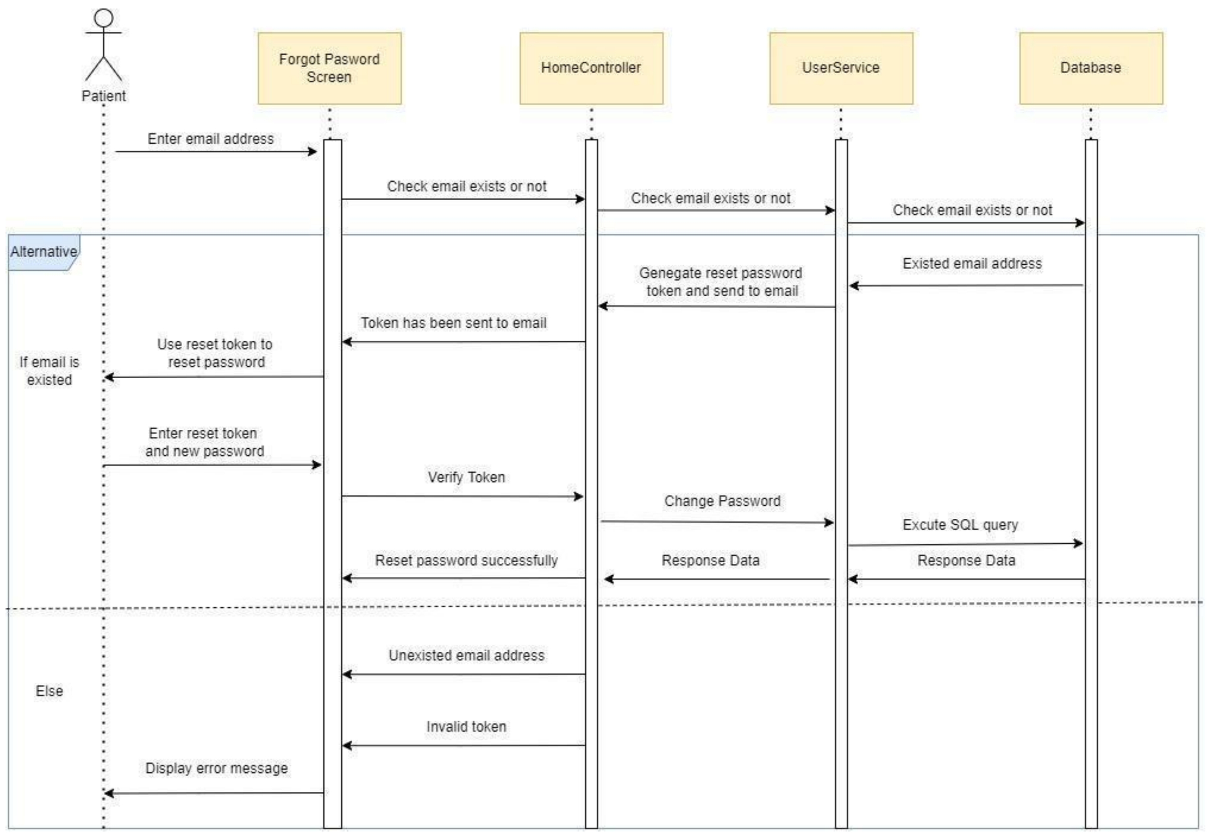

Figure 8: Reset Password Diagram

3. Edit profile

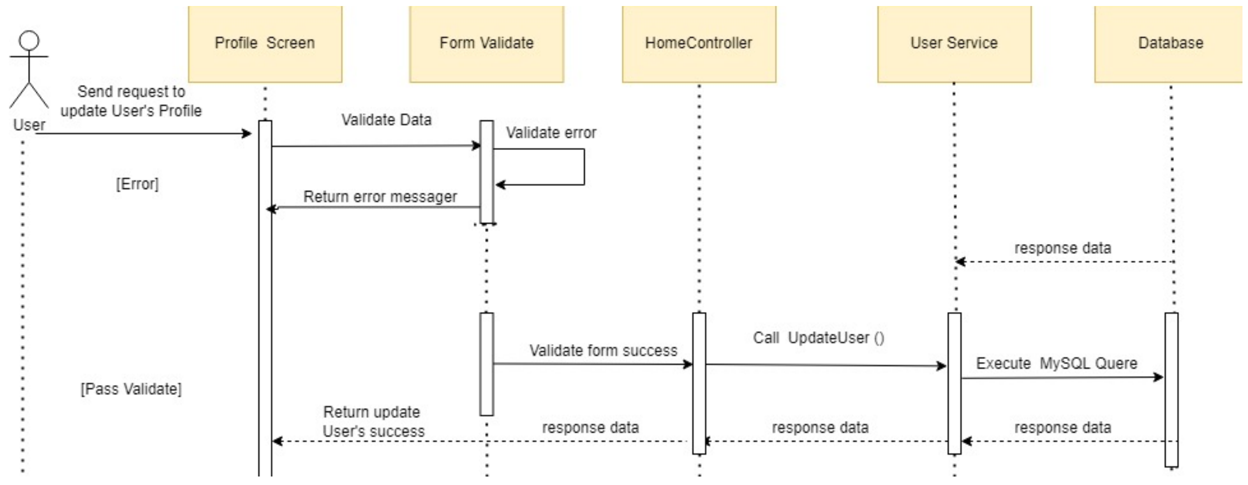

Figure 9: Edit Profile Diagram

4. Change password

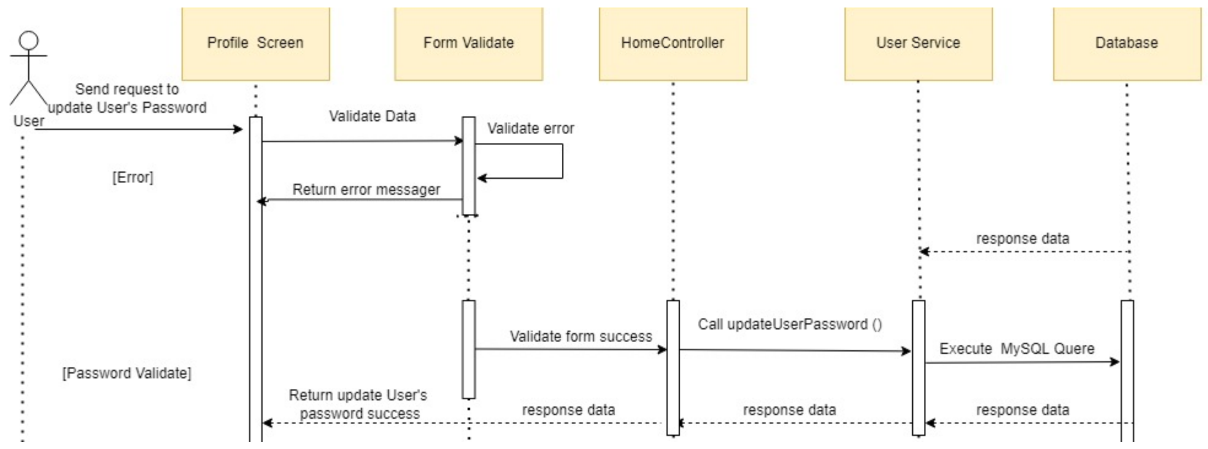

Figure 10: Change Password Diagram

5. Detail appointment

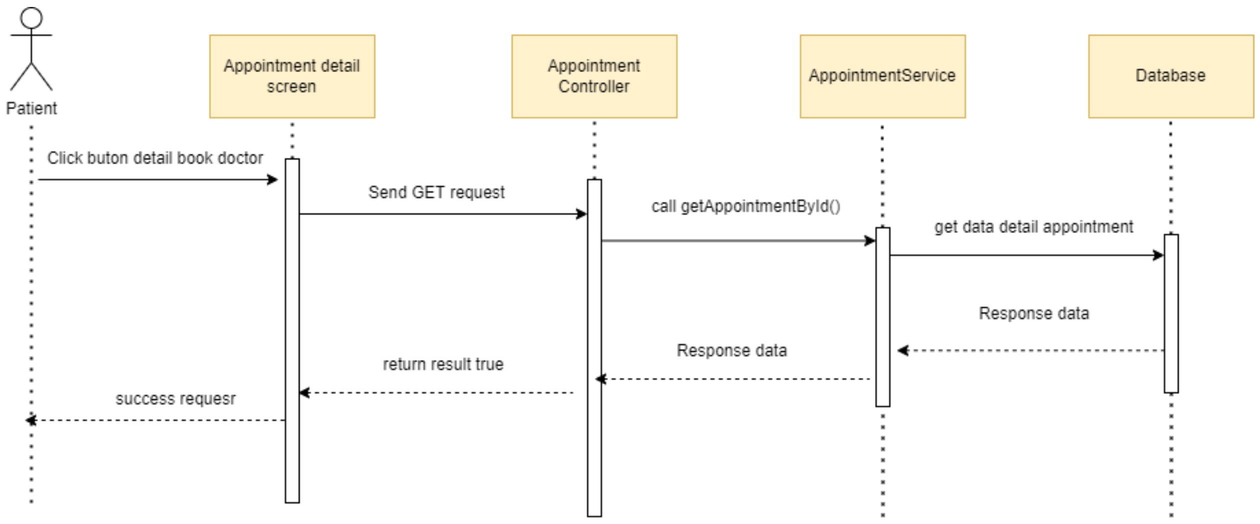

Figure 11: Detail Appointment Diagram

6. *Cancel appointment*

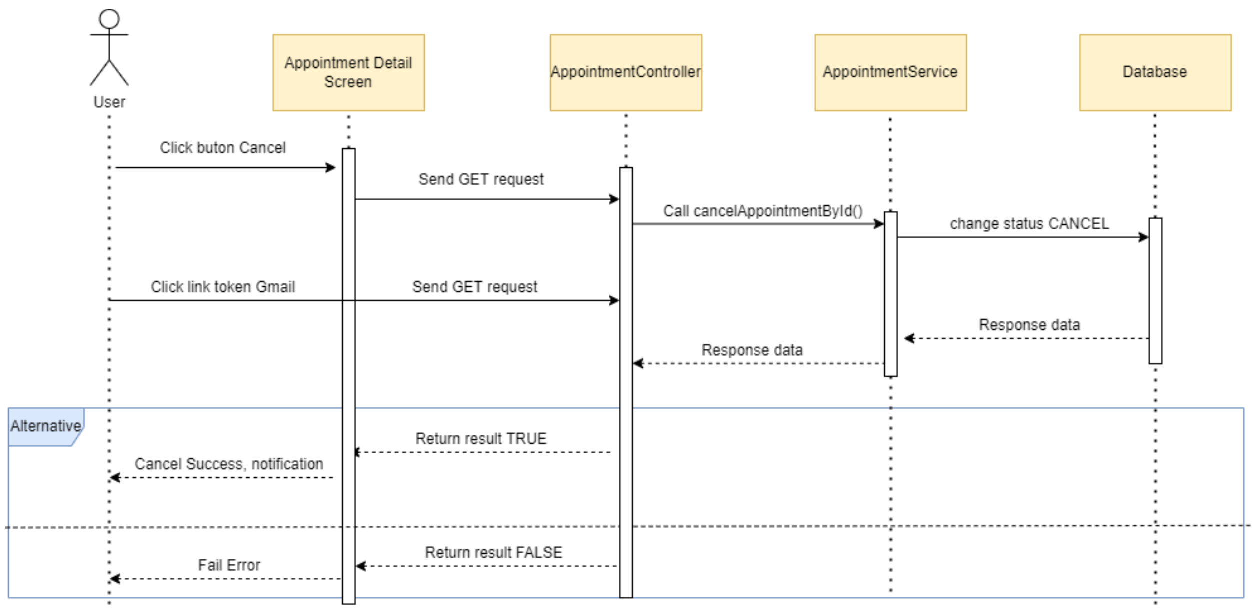

Figure 12: Cancel Appointment Diagram

7. *Reject appointment*

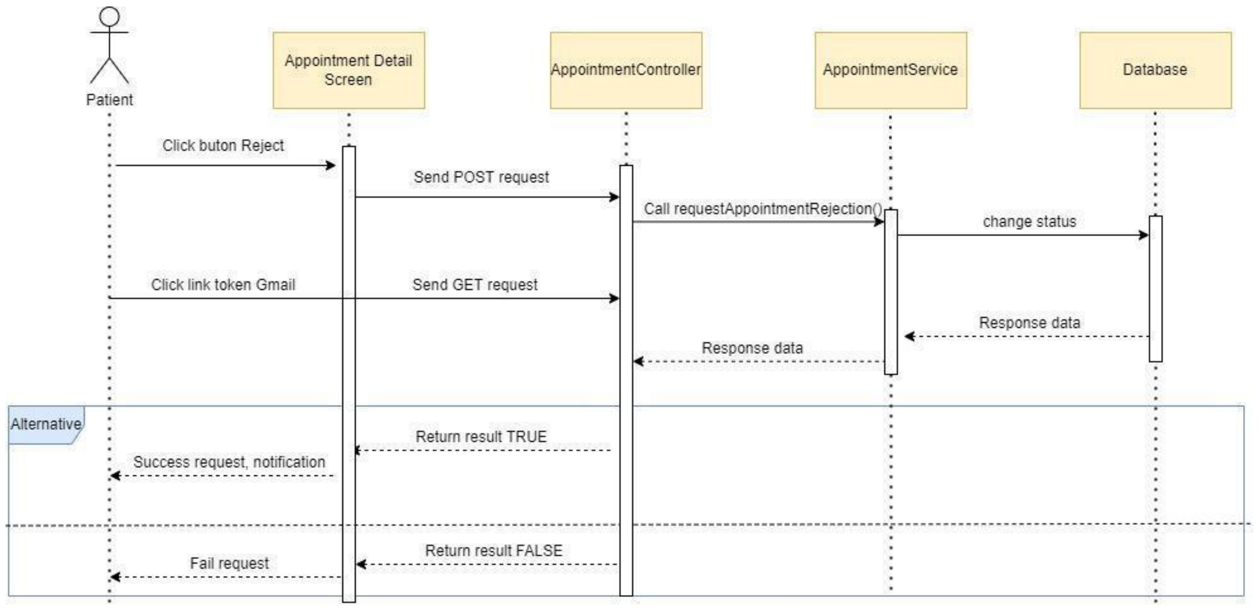

Figure 13: Reject Appointment Diagram

8. *Download invoice*

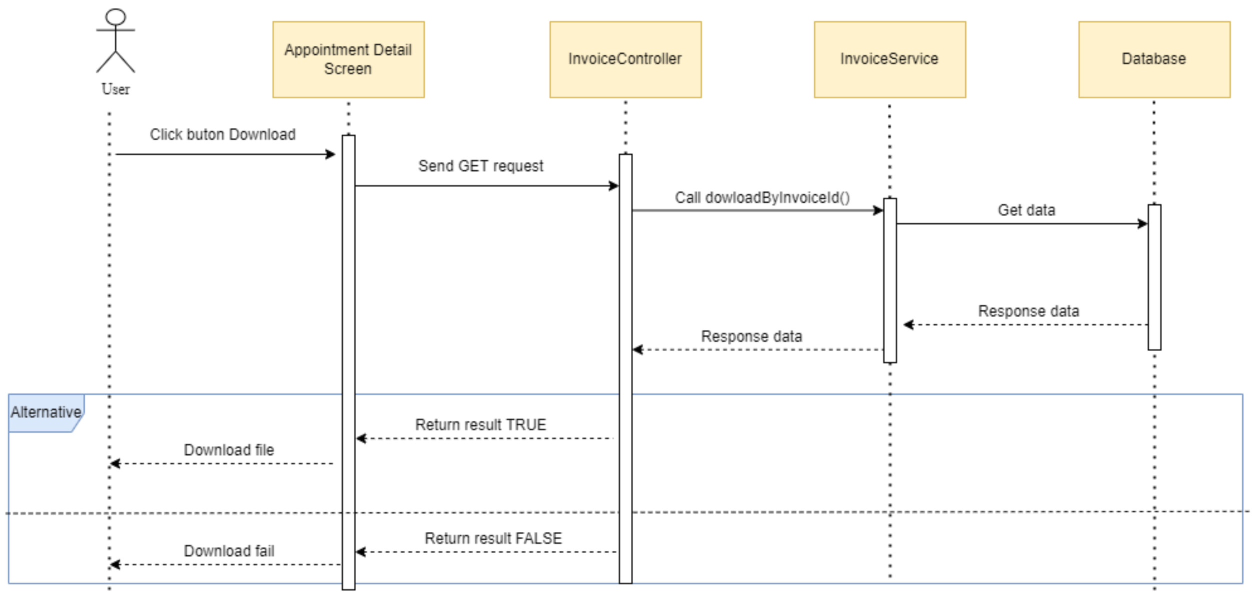

Figure 14: Download Invoice Diagram

9. *Send message real-time*

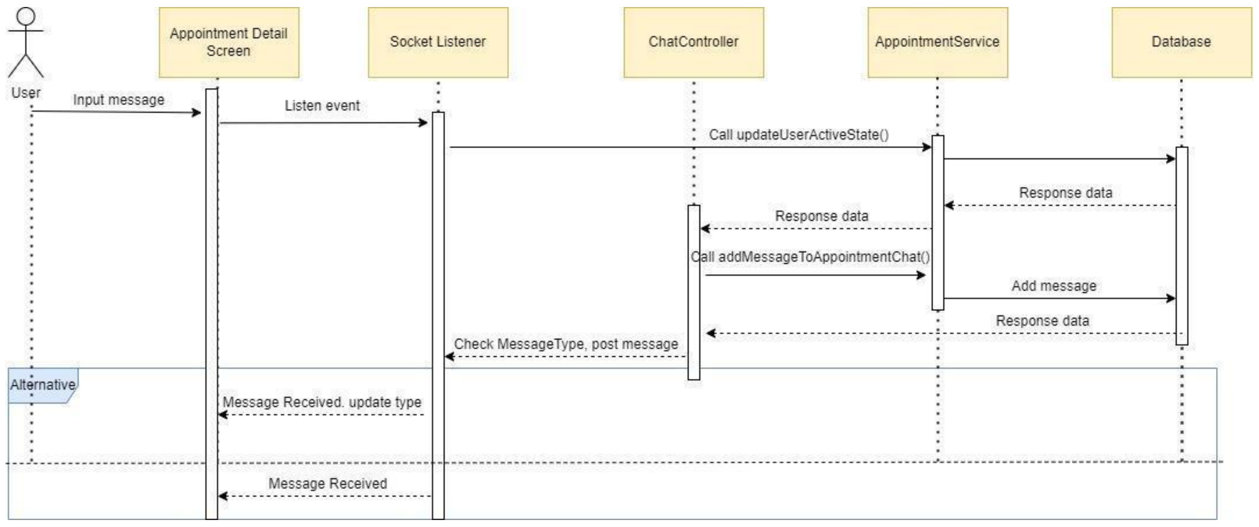

Figure 15: Send Message Real Time Diagram

10. List notification

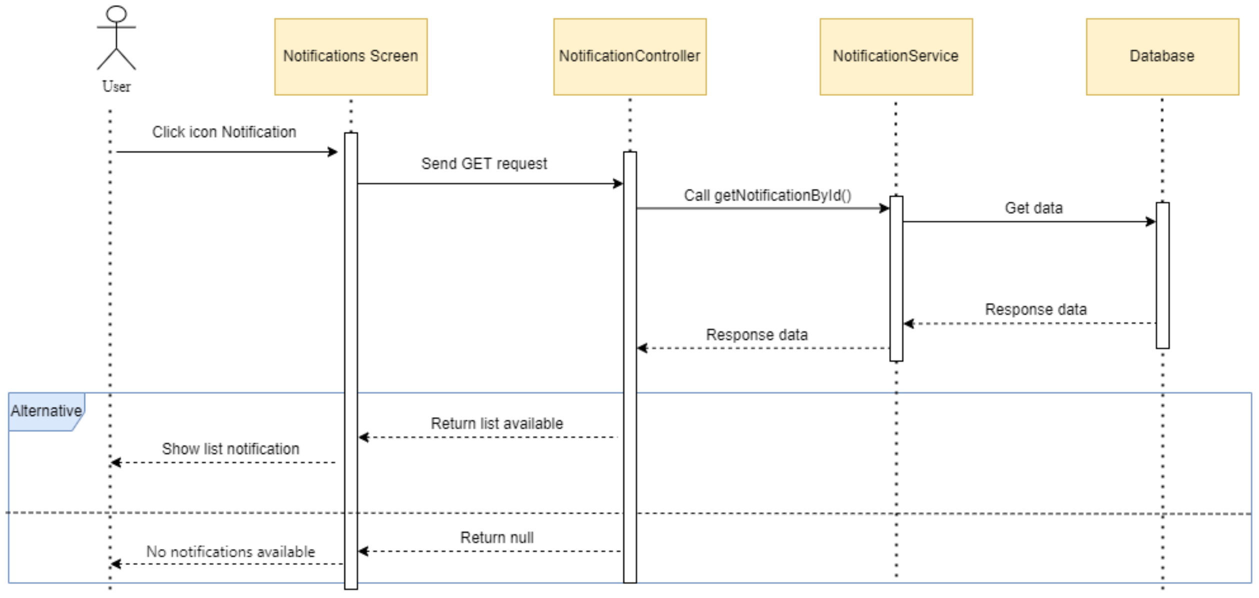

Figure 16: List Notification Diagram

3.3.2. Patient

1. Register

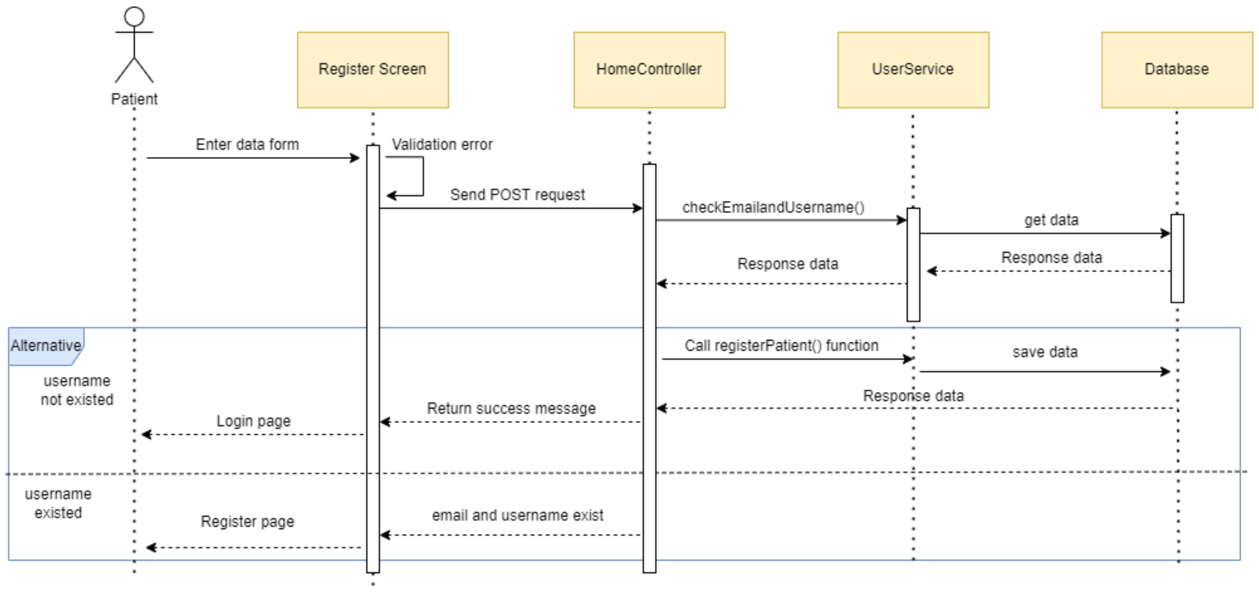

Figure 17: Register Patient Diagram

2. *Detail doctor*

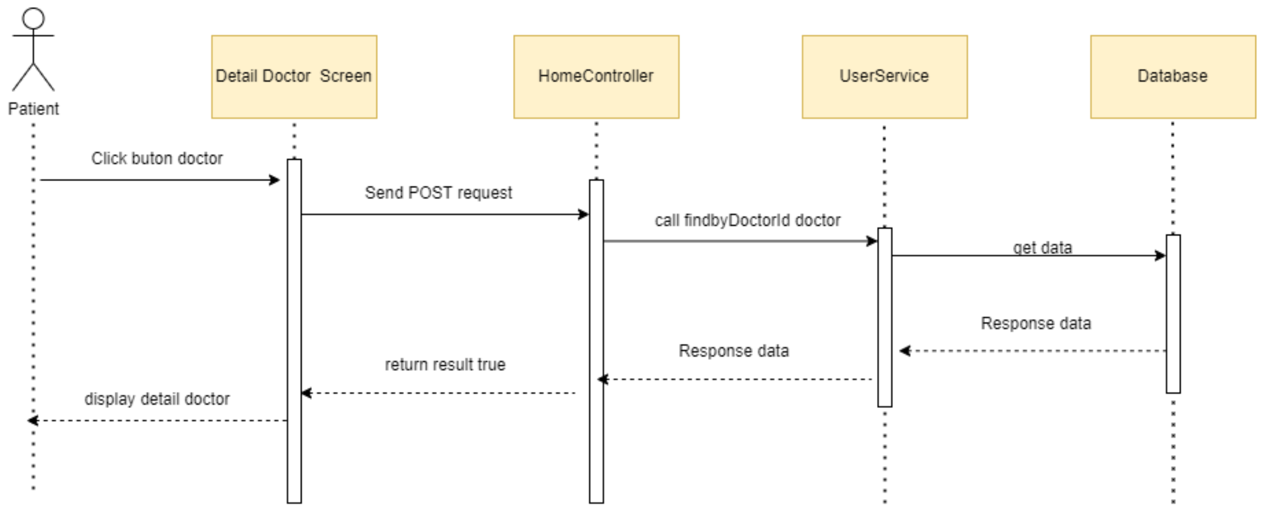

Figure 19: Detail Doctor Diagram

3. *Choose available time*

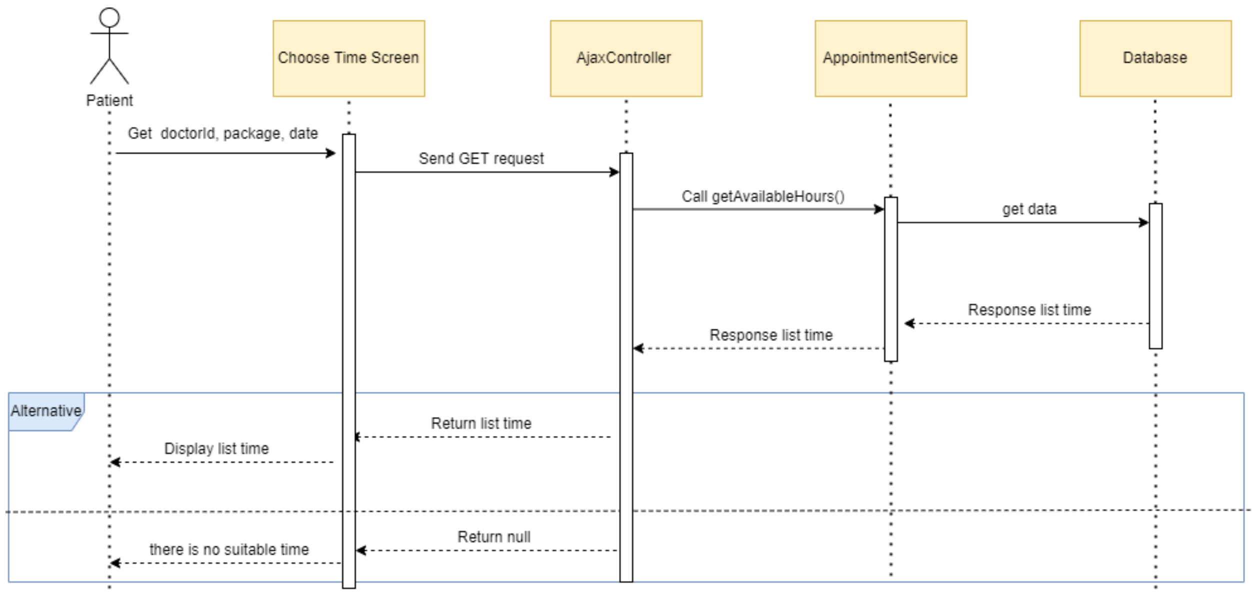

Figure 20: Choose Available Time Diagram

4. *Enter OTP*

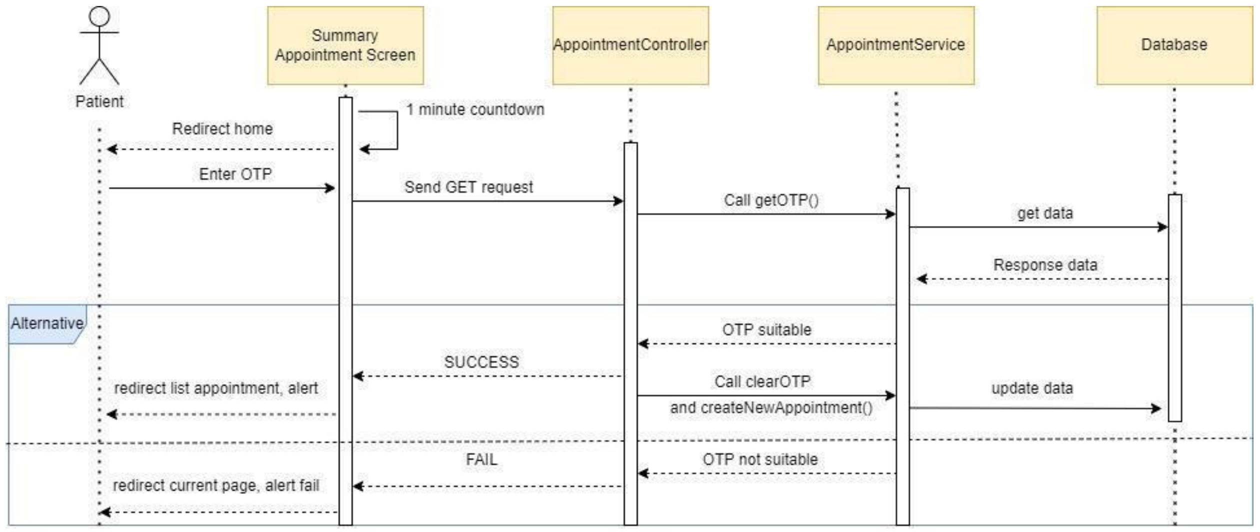

Figure 21: Enter OTP Diagram

5. *Review doctor*

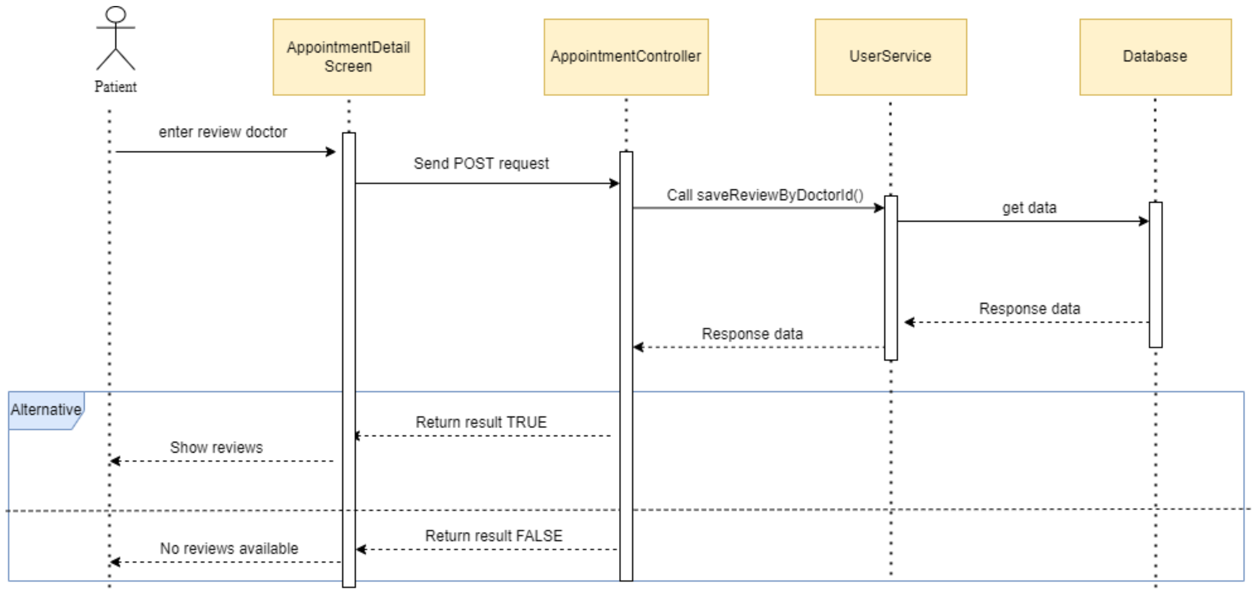

Figure 22: Review Doctor Diagram

6. *Follow Doctor*

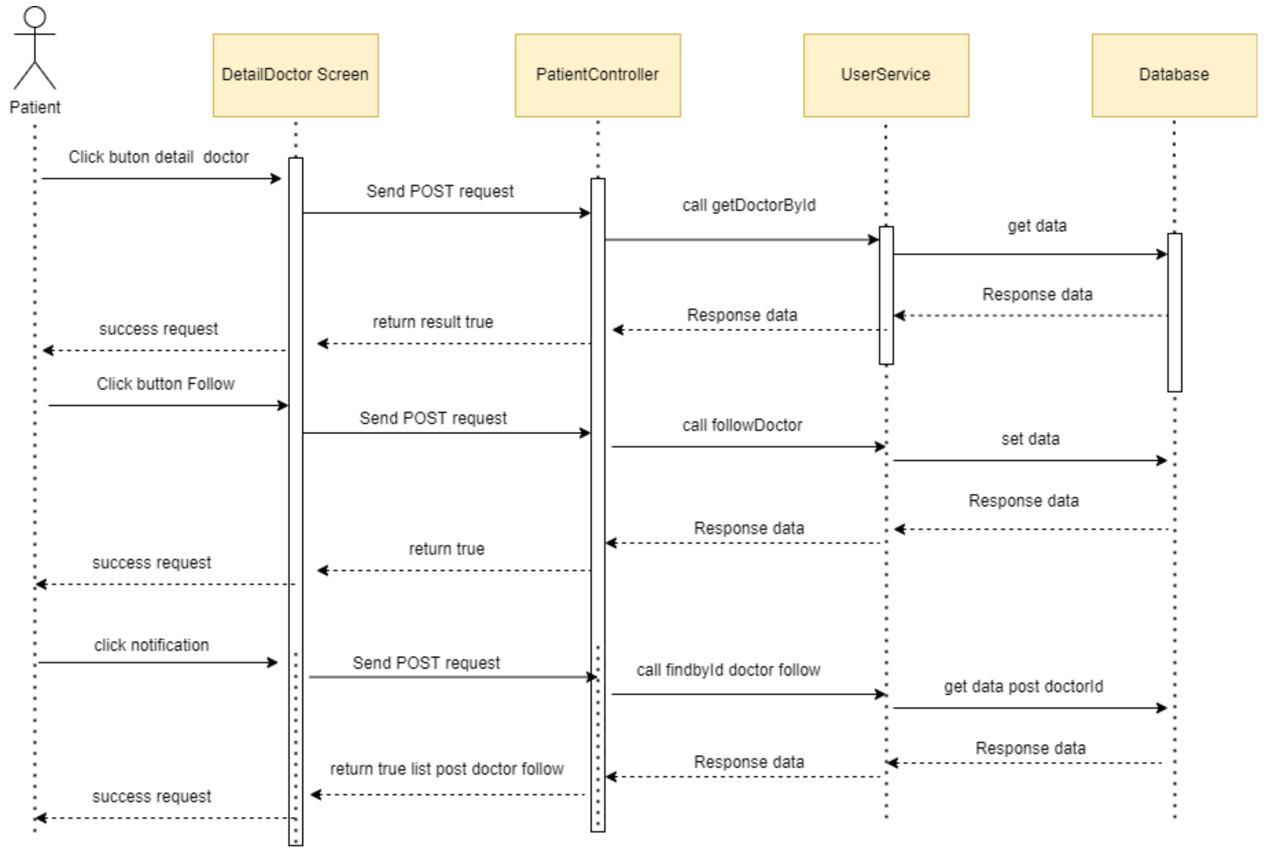

Figure 23: Follow Doctor Diagram

7. *Unfollow Doctor*

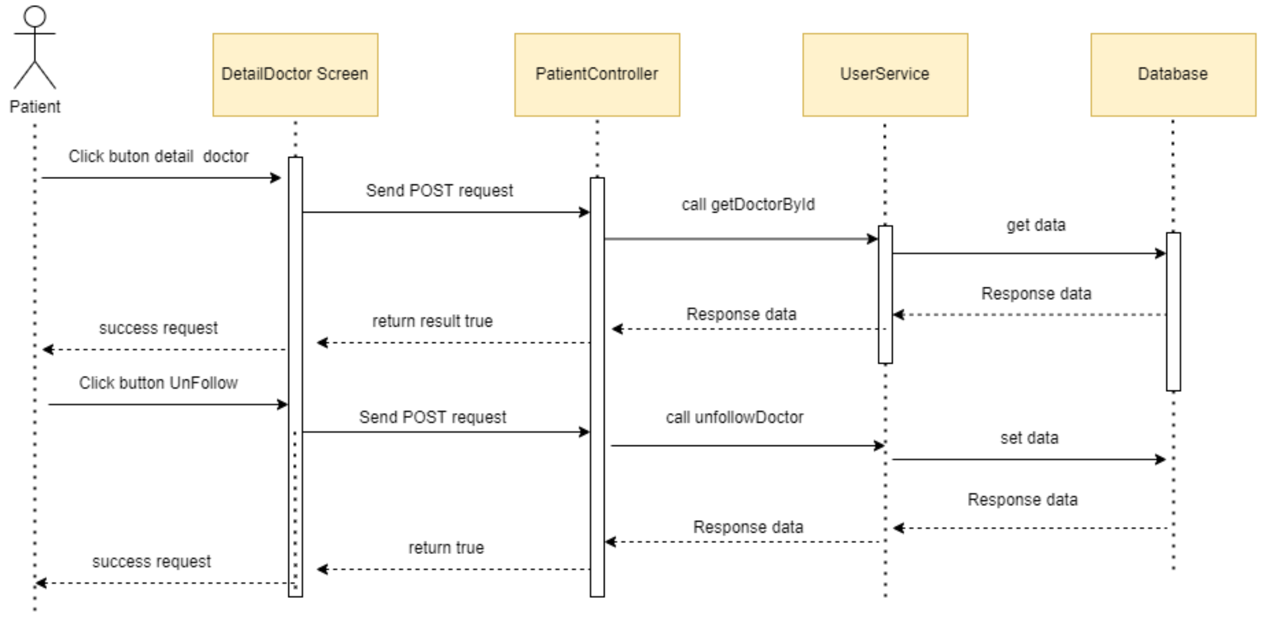

Figure 24: Unfollow Doctor Diagram

3.3.3. Doctor

1. Accepted reject appointment

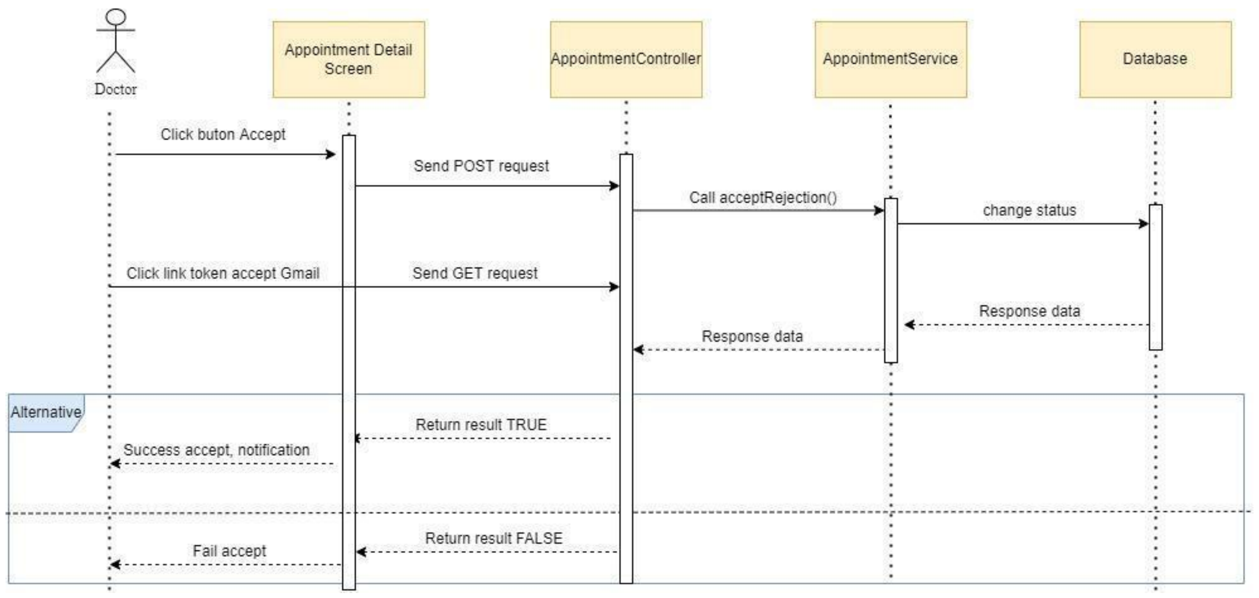

Figure 25: Accepted Reject Appointment Diagram

2. Change schedule

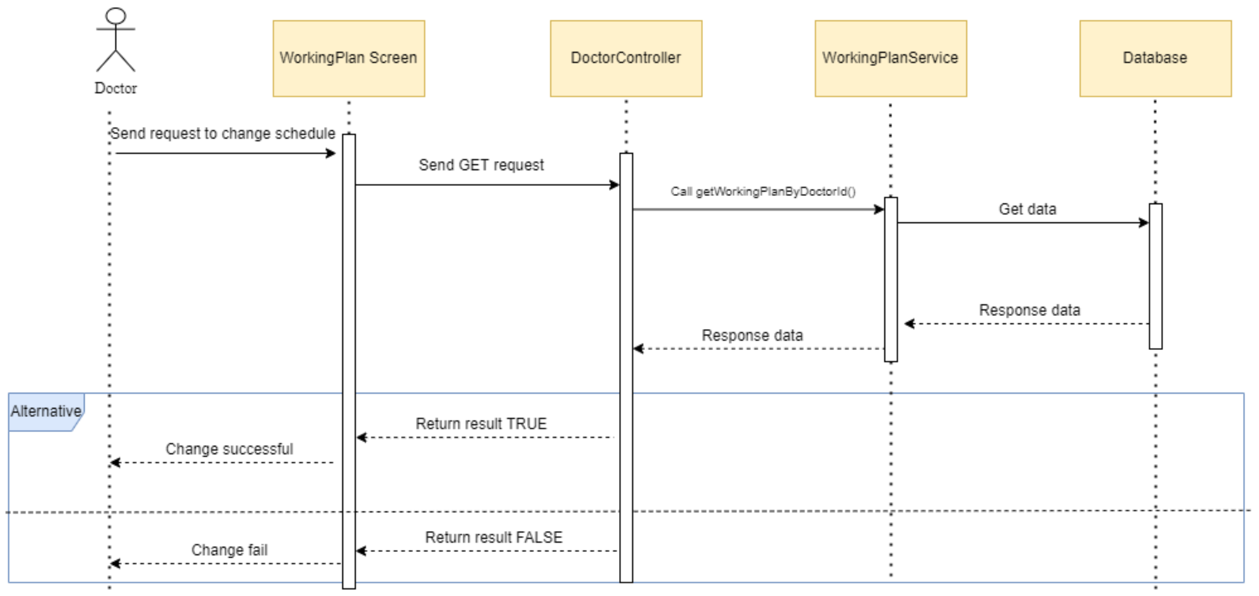

Figure 26: Change Schedule Diagram

3. *Add result for patient*

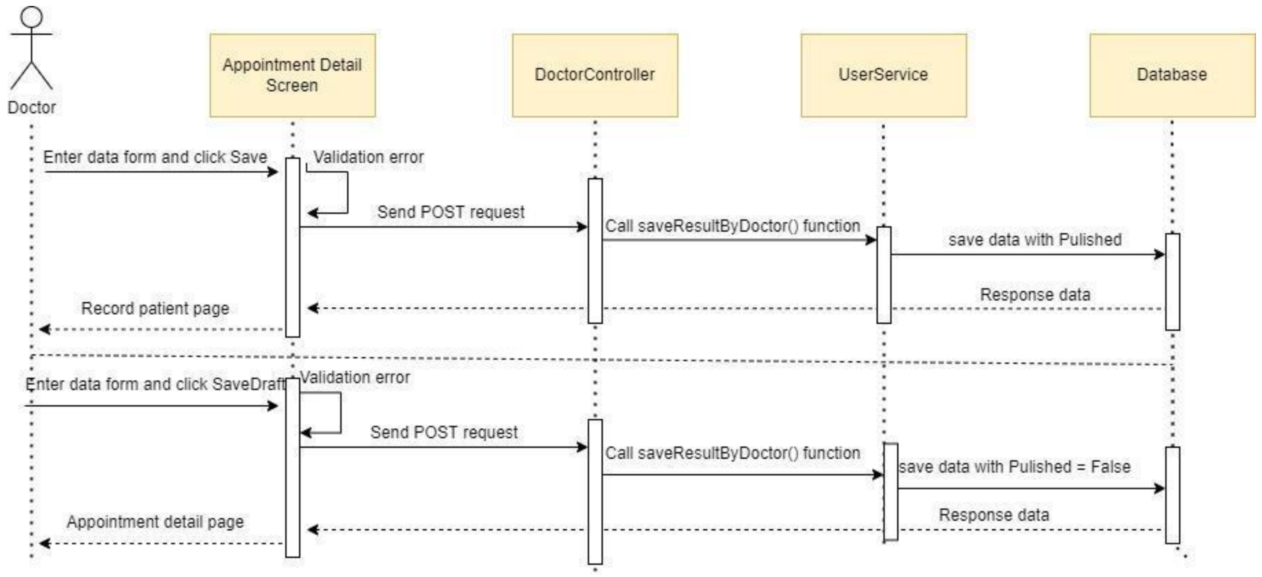

Figure 27: Add Result for Patient Diagram

4. *Add post*

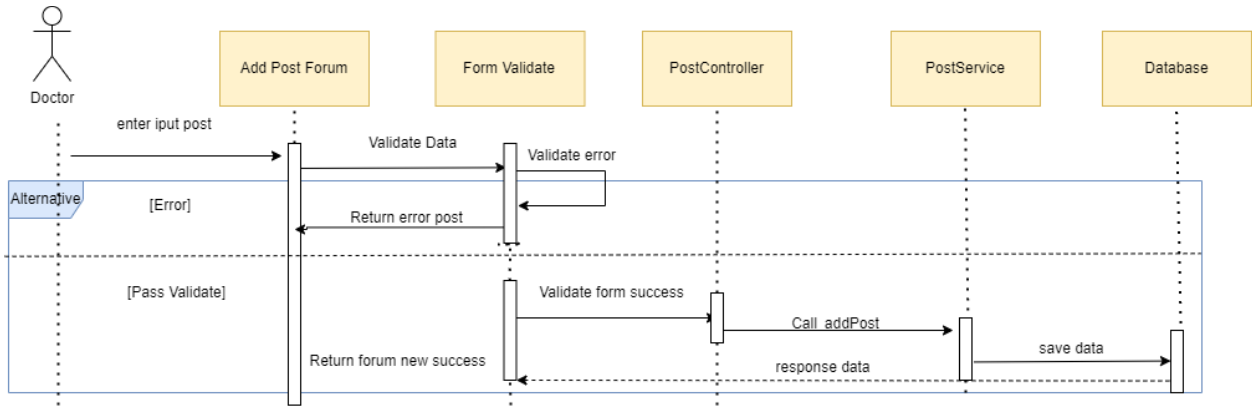

Figure 28: Add Post Diagram

3.3.4. Admin

1. Add Doctor

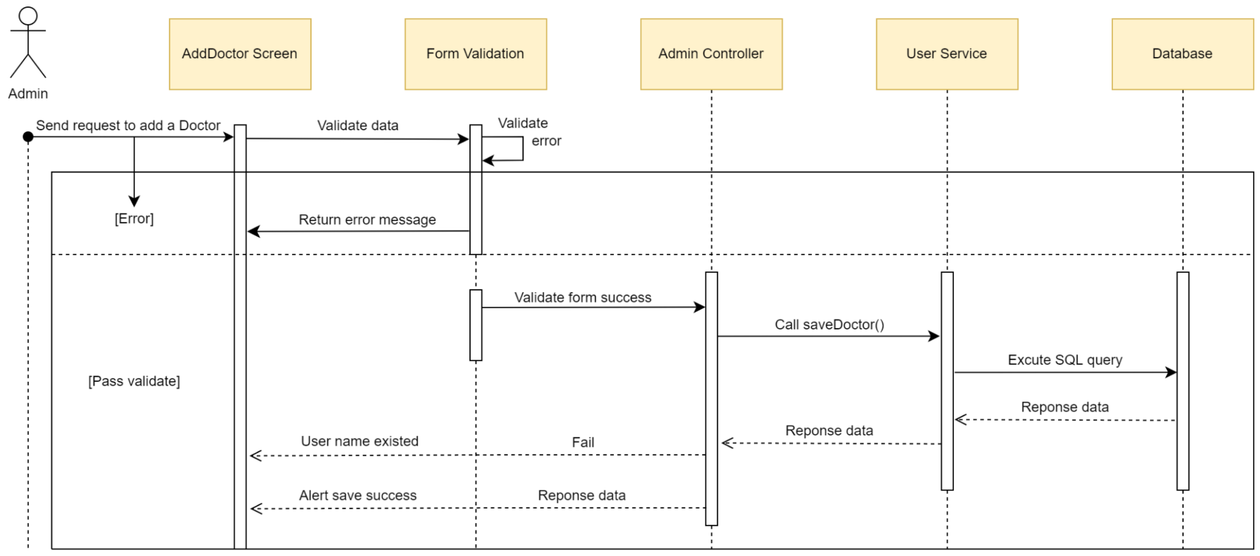

Figure 29: Add Doctor Diagram

2. Update Doctor

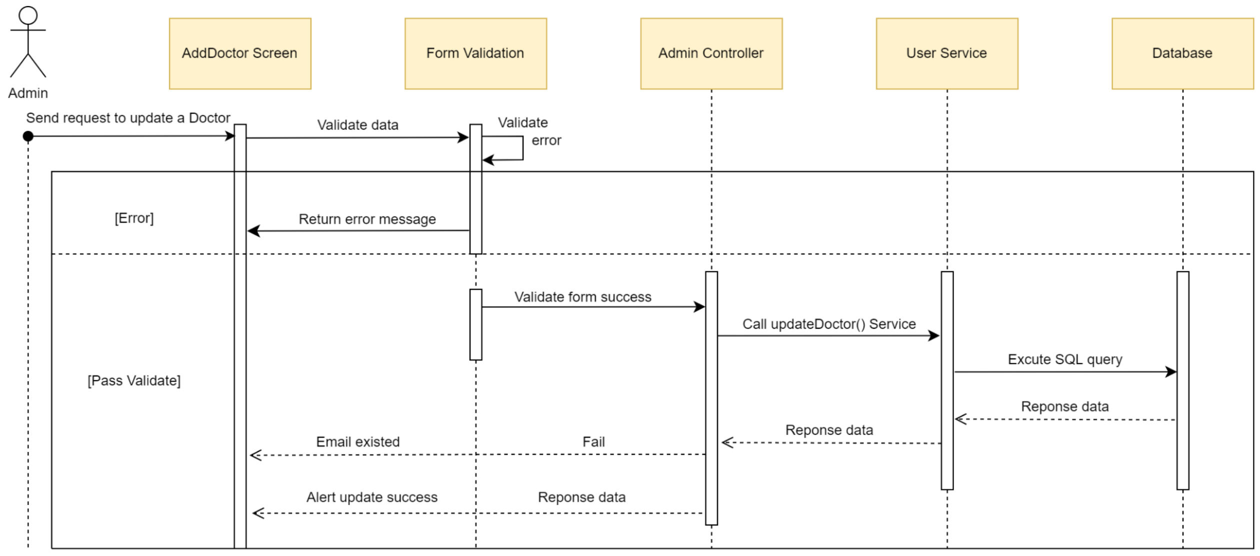

Figure 30: Update Doctor Diagram

3. *Create Clinic*

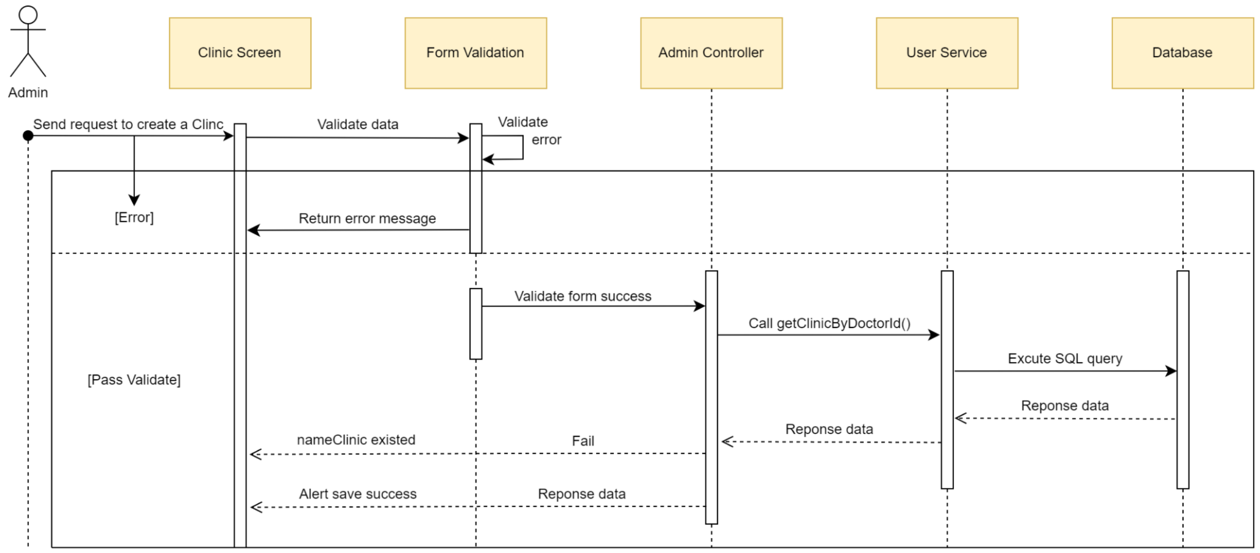

Figure 31: Create Clinic Diagram

4. *Update Clinic*

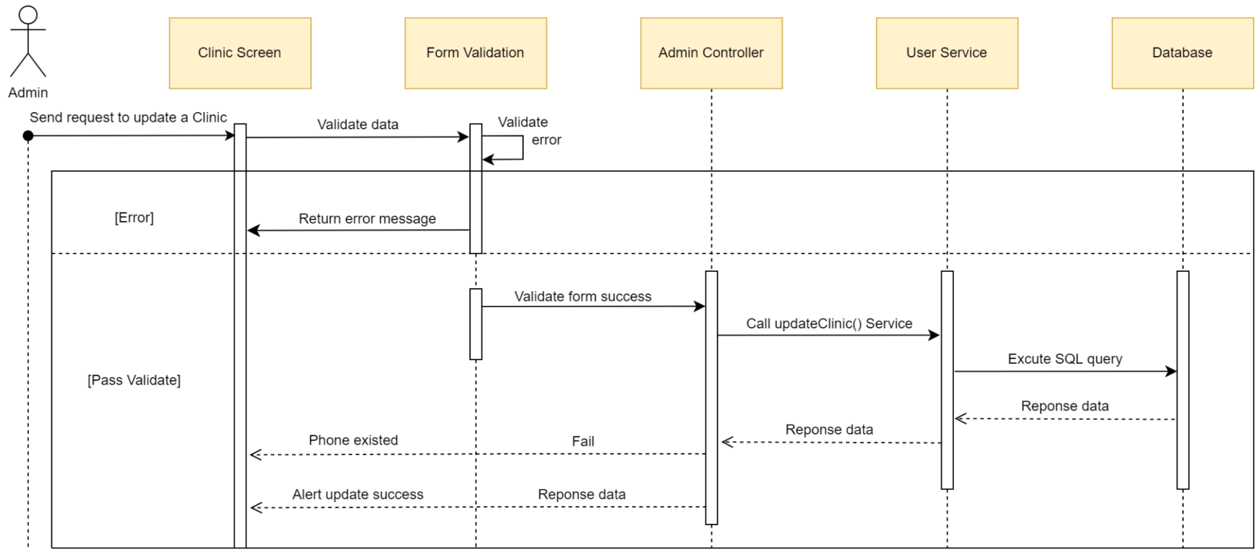

Figure 32: Update Clinic Diagram

5. *Delete Clinic*

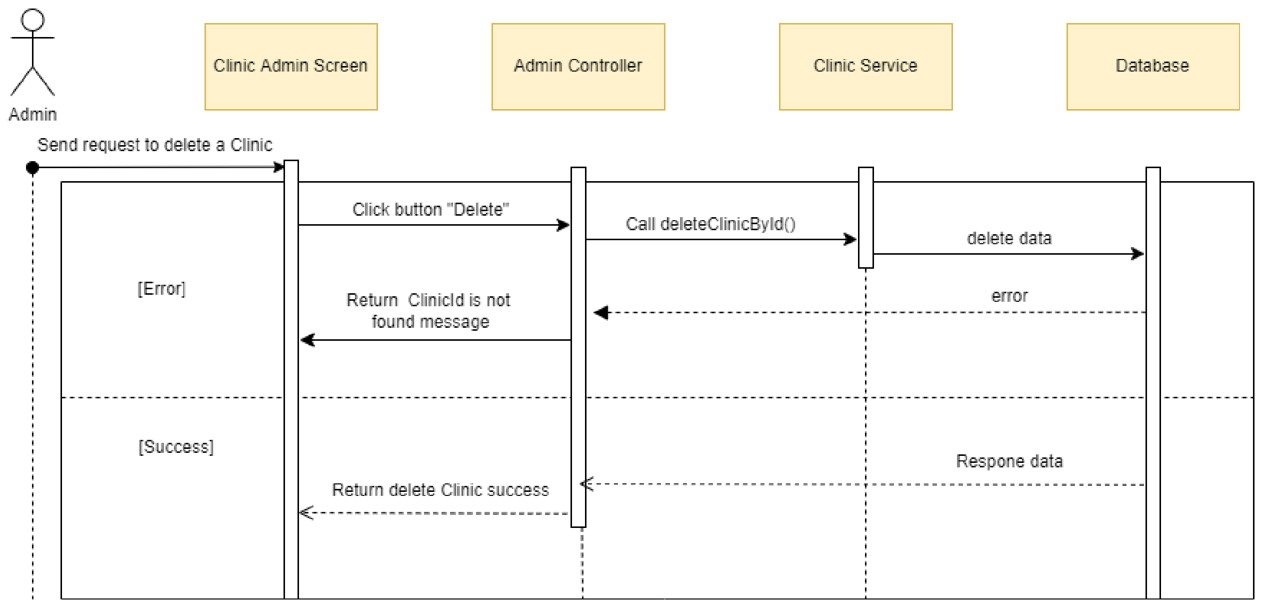

Figure 33: Delete Clinic Diagram

6. *Create Package*

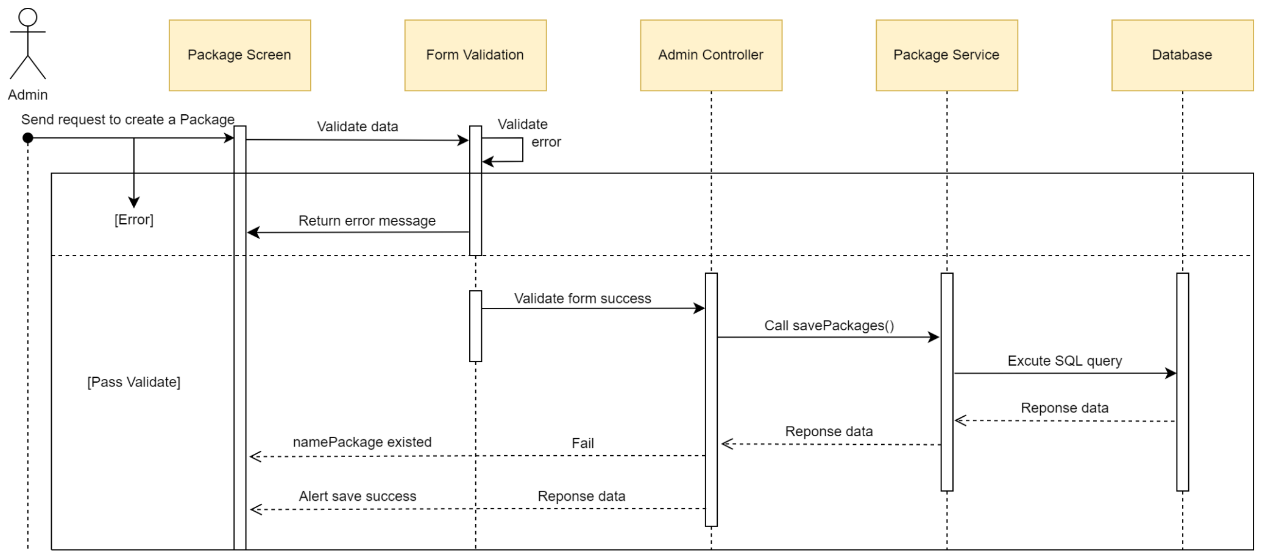

Figure 34: Create Package Diagram

7. *Update Package*

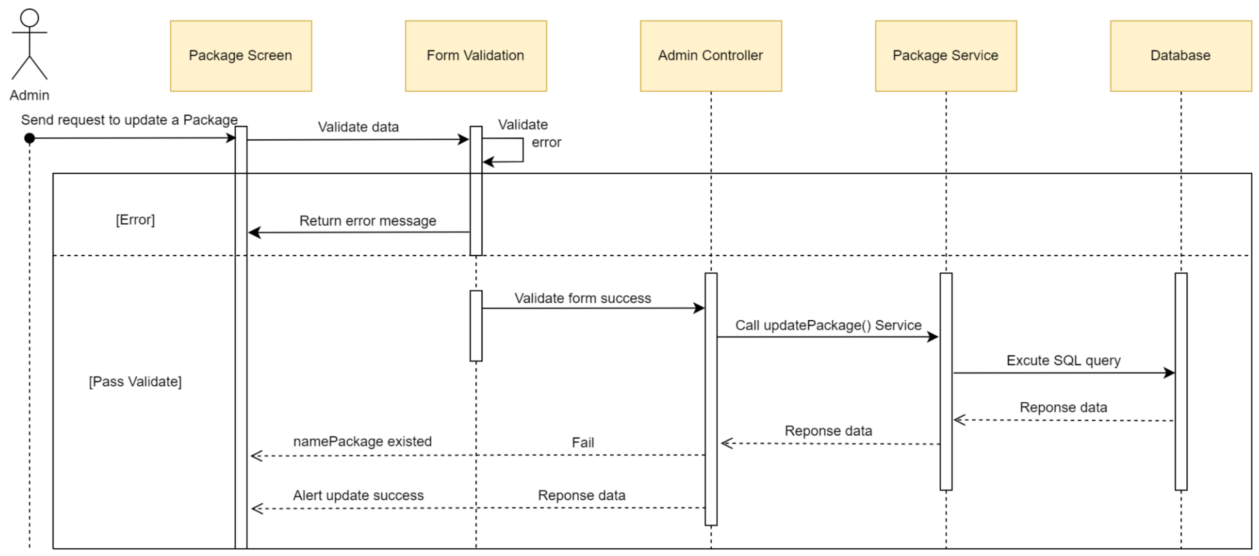

Figure 35: Update Package Diagram

8. *Delete Package*

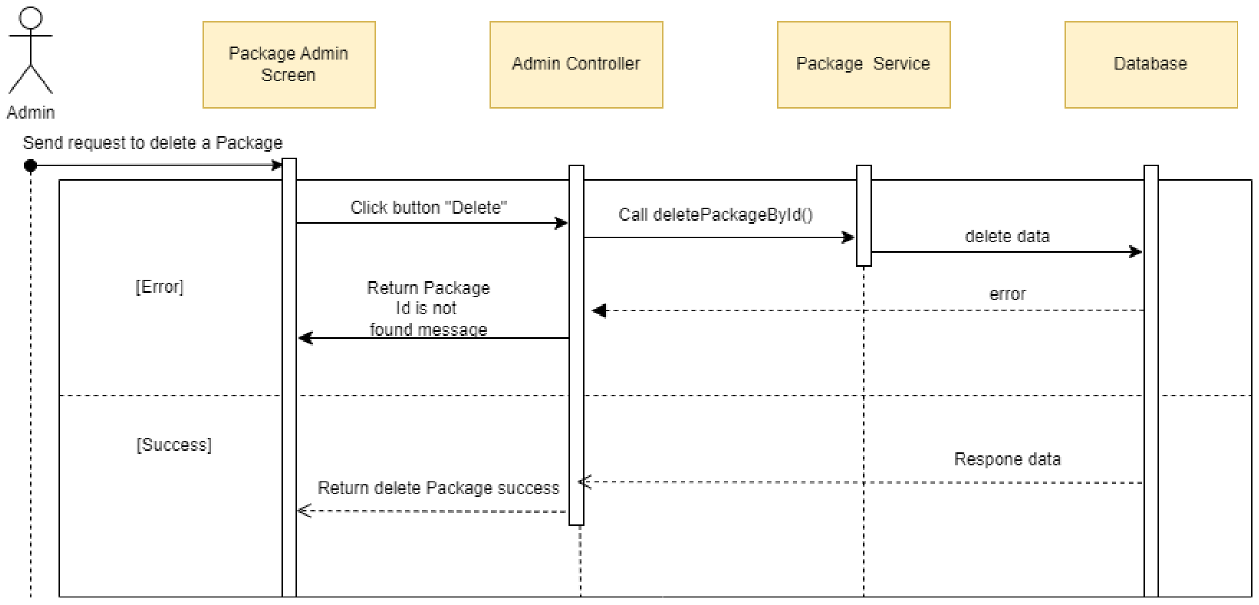

Figure 36: Delete Package Diagram

9. *Update Patient*

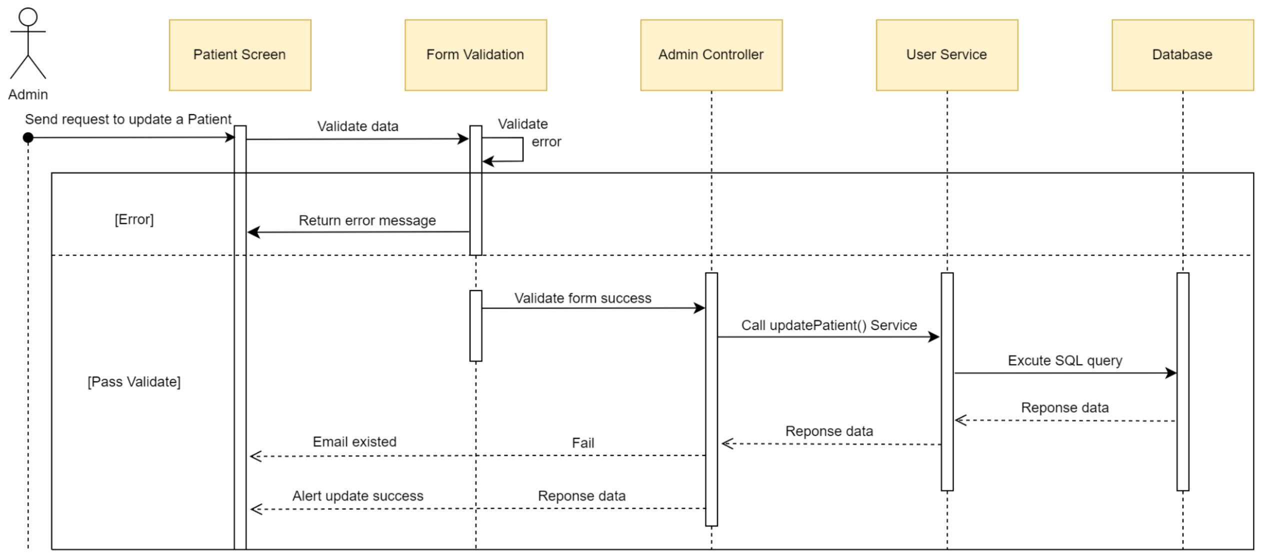

Figure 37: Update Patient Diagram

10. *Pay Invoice*

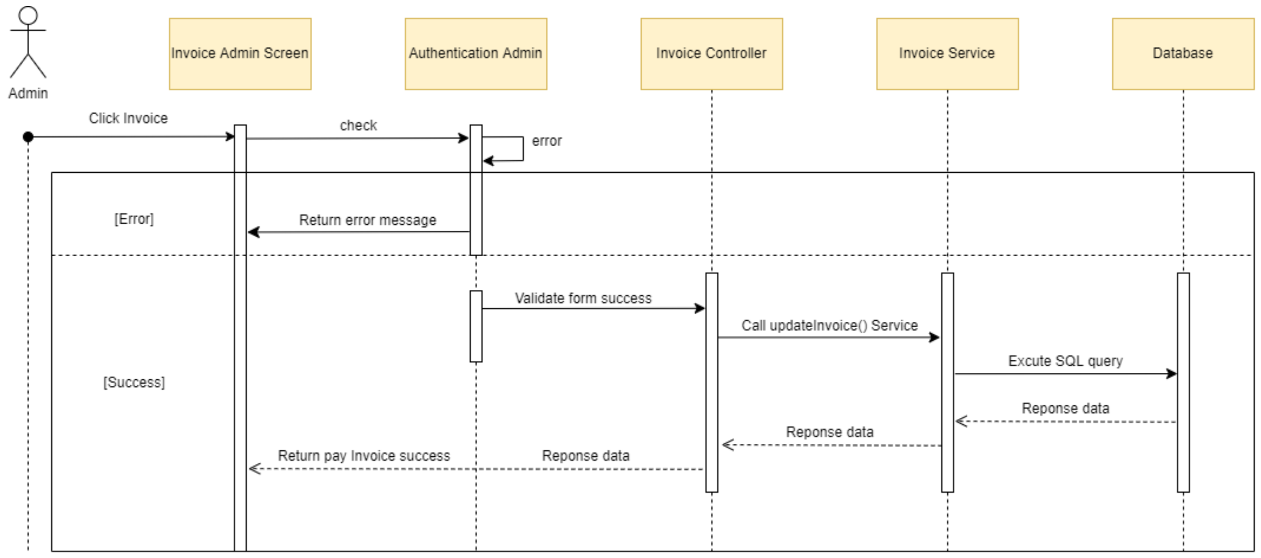

Figure 38: Pay Invoice Diagram

11. Invoice appointment

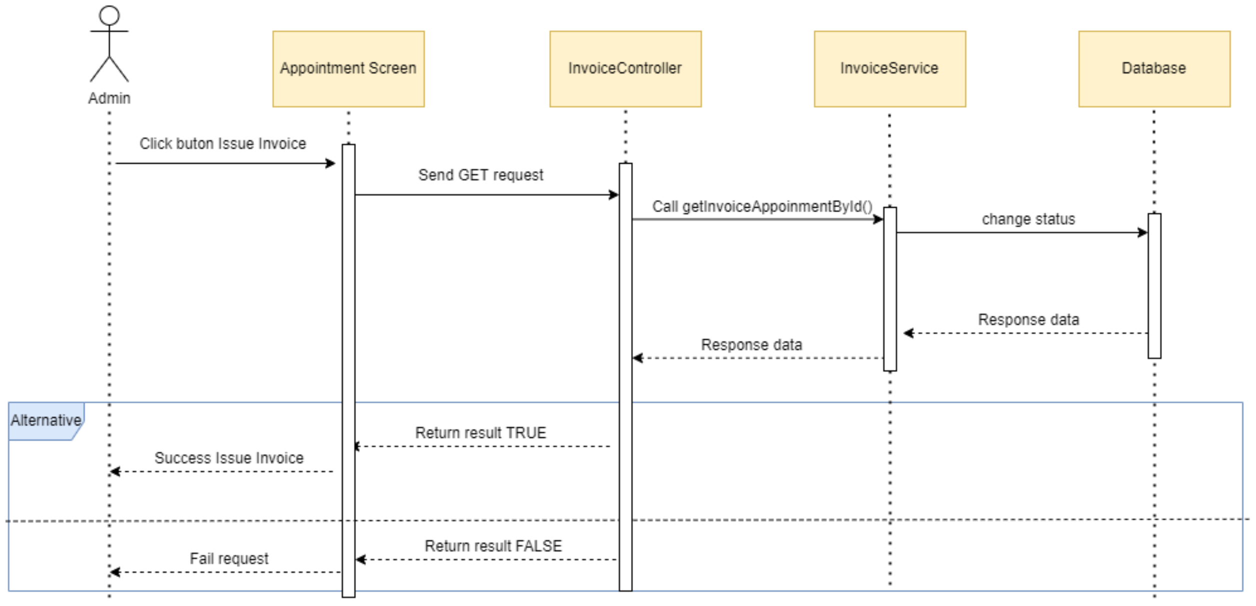

Figure 39: Invoice Appointment Diagram

12. Automatically issue invoices

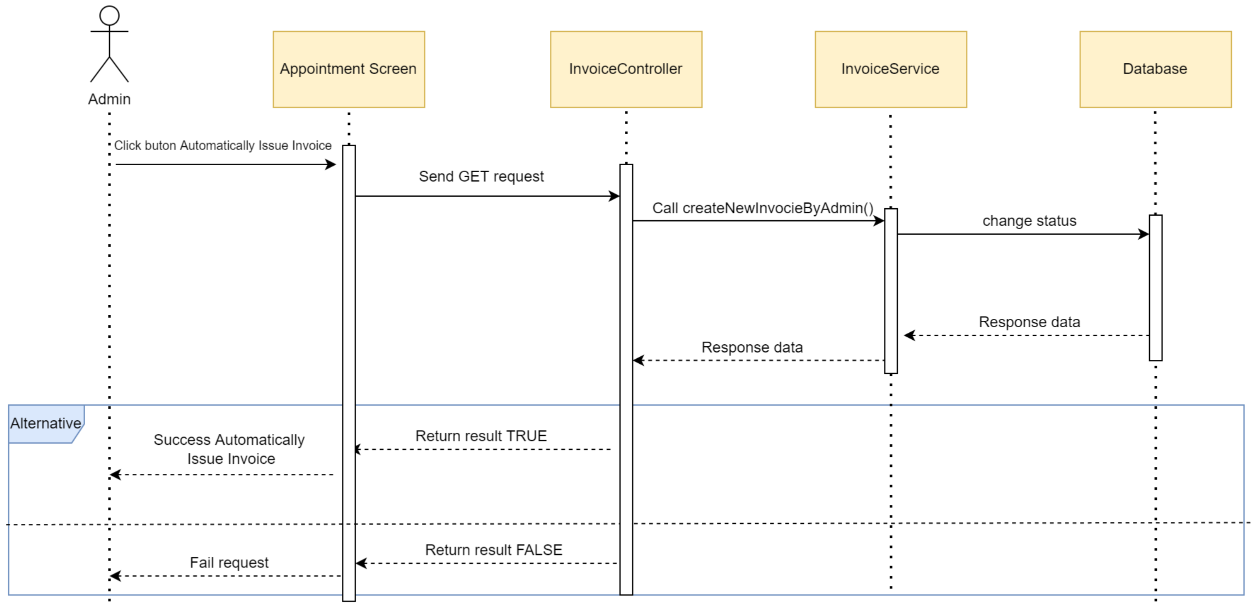

Figure 40: Automatically Issue Invoices Diagram

13. *Delete review*

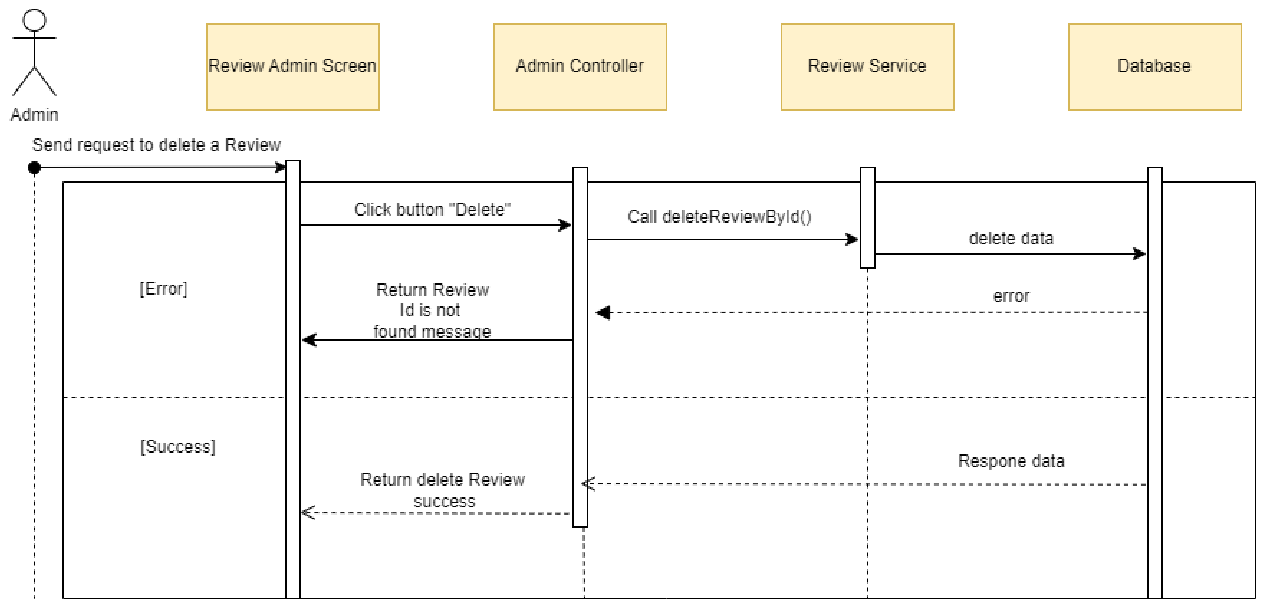

Figure 41: Delete Review Diagram

14. *Respond to questions*

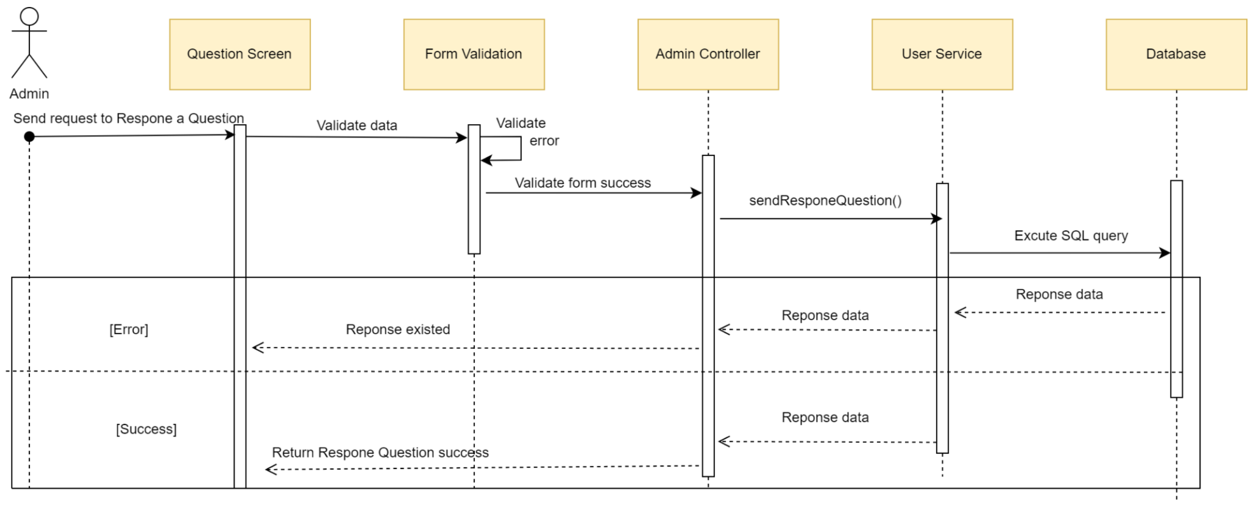

Figure 42: Respond to Questions Diagram