# Software Design Specification (SDS)

**Project Name**: Medify

**Prepared By**:
Ahmed Sameh 202202151
Salma Rami 202201759
Shahd Tarek 202202263
Mohamed Tarek 202201058
Seif Amr 202201510

---

**Table of Contents**

---

# 1. Introduction

## 1.1 purpose

Medify's mission revolves around enhancing the working of a given hospital through recording patient history, assigning doctors, tracking pharmacy-related details, and coordinating appointment schedules. In addition, it also makes it possible for the patients to communicate with the hospital through an online platform, all which improves the delivery of health services.

## 1.2 Scope

The system integrates patients, doctors, pharmacies and administrative staff of the hospitals into one website with the aim to promote organized efforts and communication.

- Patient Registration
- Appointment Scheduling
- Doctor Assignment and Profile Access
- Communication Portal

---

# 2. System Overview

The system consists of the following components:

- **Frontend**:HTML, CSS, Bootstrap, JS
- **Backend**: Flask (Python)
- **Database**: MySQL

---

# 3. System Architecture

## 3.1 Architectural Design

This project utilizes client-server system architecture, where front, back, and database ends combine to address requests fully operational within the system:

- **Interaction between Front-end and Back-end components**: The front end of the application employs RESTful API calls over HTTPS to communicate with the backend, this guarantees security of data while transmitting.
- **Interaction between the Backend and the Database**: The backend uses MySQL (or another relevant database language) to interact with the database for purpose of performing data related tasks.

## 3.2 Data Flow

1. **User Interaction:** Interaction with the system is done using the web or mobile interface (UI) whereby users perform actions, such as creating a new appointment, updating their profile, or reviewing a physician's record, within the system.
2. **Request Processing**: For every action, the front end makes a call to the backend server through a RESTful API call sending necessary data payloads and parameters.
3. **Data Handling:** The backend accepts and executes the request after validating the parameters, performing the requisite business logic, and communicating with the database to obtain, add, modify, or eliminate the records managed by the request.
4. **Response:** The backend sends a response, including the data that was elicited, if available, or a statement of such registration, which is relayed to the user interface on the frontend which updates itself in line with the users action.

---

# 4. Database Design

The system will store data in database type relational (MySQL) with the following entities and relationships:

### Table 1: Patients Table

**Description**: Stores information about patients.

**patient_id**: Unique identifier for each patient (Primary Key).

**name**: Full name of the patient.

**contact_info**: Phone number and/or email address of the patient.

**medical_history**: A brief summary of the patient's past medical records

### Table 2: Doctors Table

**Description**: Stores information about doctors.

**doctor_id**: Unique identifier for each doctor (Primary Key).

**name**: Full name of the doctor.

**specialization**: Area of medical expertise (e.g., cardiologist, dermatologist).

**contact_info**: Phone number and/or email address of the doctor.

**Table 3: Appointments Table**

**Description**: Stores information about patient appointments with doctors.

**appointment_id**: Unique identifier for each appointment (Primary Key).

**patient_id**: Foreign Key linking to the Patients Table.

**doctor_id**: Foreign Key linking to the Doctors Table.

**appointment_date**: Scheduled date and time of the appointment.

**status**: Status of the appointment (e.g., confirmed, canceled).

*Note: Describe the main tables or collections and their relationships (e.g., one-to-many, many-to-many).*

**Relationships**:
**One-to-Many:** A patient can have multiple appointments, but each appointment is linked to one patient.
**One-to-Many**: A doctor can have multiple appointments, but each appointment is linked to one doctor.

---

## 5. Technology Stack

- **Frontend**: JAVASCRIPT, HTML, CSS
- **Backend**: Flask(python)
- **Database**: MySQL
- **Hosting**: AWS

---

## 6. Testing Plan

### 6.1 Unit Testing

As for unit testing, every module and function provided by the system for users interacts with other components like UI elements, API endpoints, database queries and they can be verified as functional. Testing will cover isolated functions within the frontend (React components) and backend (Flask functions).

### 6.2 Integration Testing

Integration tests will ensure that different modules work in the intended way and together as promised. This includes:

-Frontend and Backend integration, confirming that the API calls made in the frontend give the correct output and can be displayed.

-Integration between the backend and the Database where the backend handles requests to and from the MySQL database for data management and retrieval.

### 6.3 User Acceptance Testing (UAT)

End users, for instance, hospital staff, doctors and patients will take part in User Acceptance Testing to validate that the system is capable of their needs and expectations. As for this activity, it is planned to focus on the tests of usability, the correctness of the data flow processes, and satisfaction of end users with the most important functions in the system including patients making appointments, registration of patients, viewing of doctor profiles

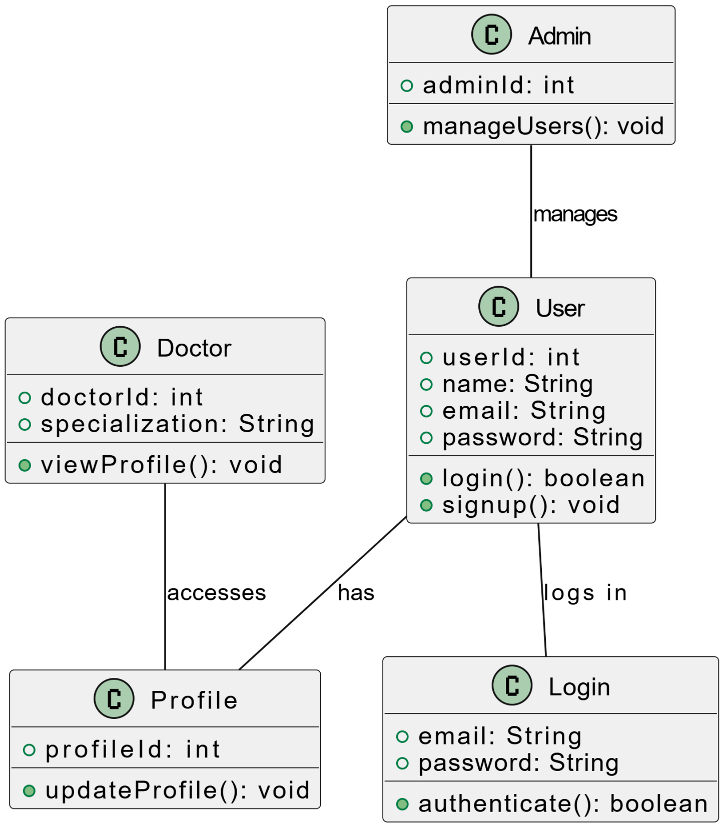
### 6.4 Performance Testing

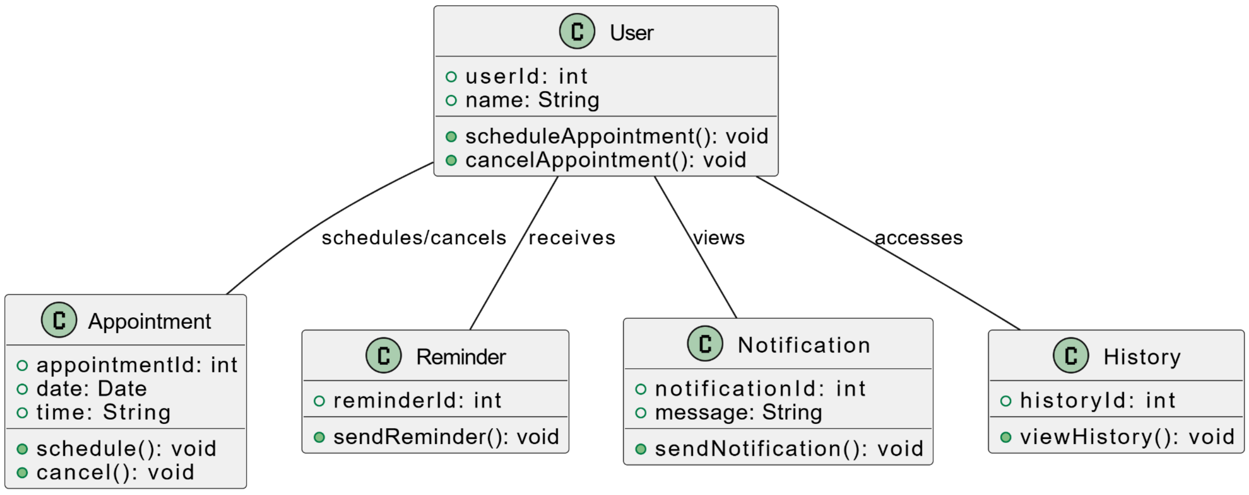
Stress testing and load testing will be done to determine the maximum expected number of users and transactions that can be supported by the system without a noticeable loss in performance. This seeks to include response times, page loading times, and resource consumption under normal and peak traffic conditions.

## 7. Conclusion

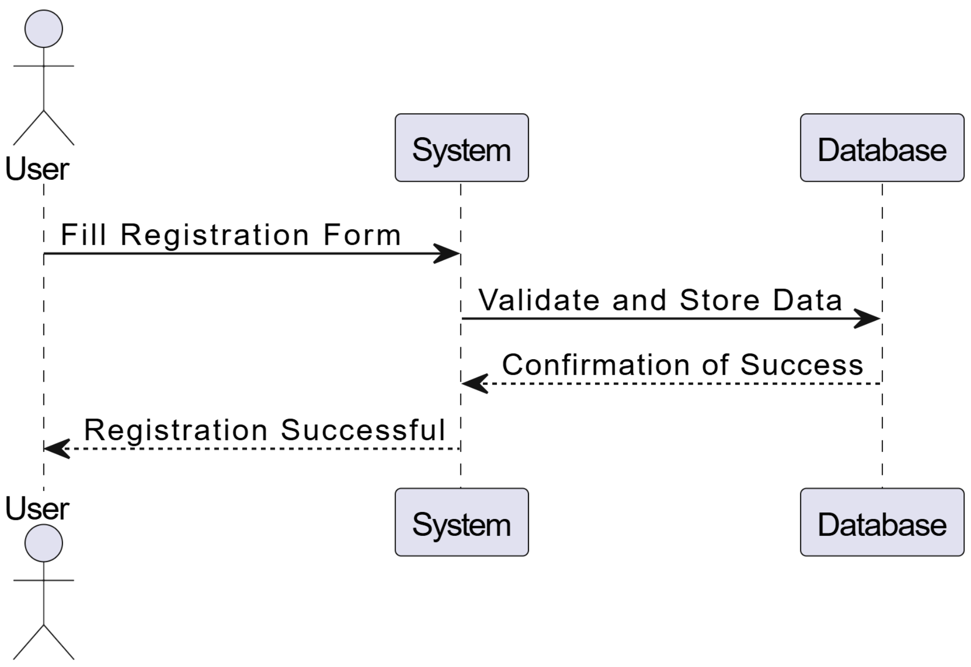
The Medify Hospital Management System has been developed in a such a way to satisfy the functional and non-functional requirements targeted in this system design specification document. Coupled with its modular design, the architecture of this system will be secure, ready for expansion, and easy to use. Notably, by embedding crucial features such as appointment management, patients management, and Protected Health Information (PHI) management, Medify will create value to its users, making healthcare processes more efficient and effective.

## Class Diagrams:

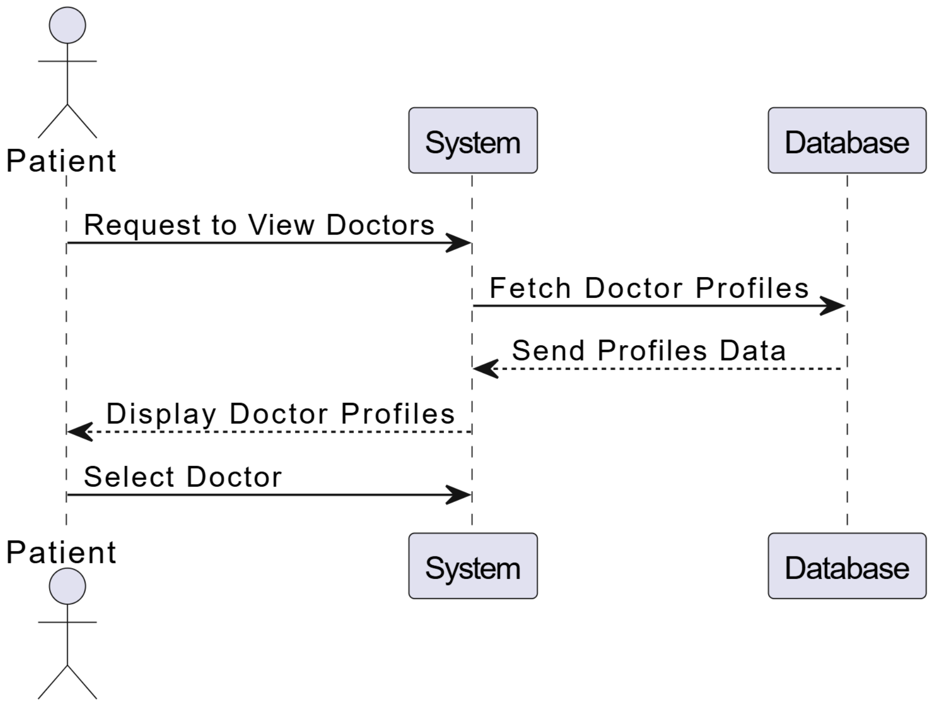




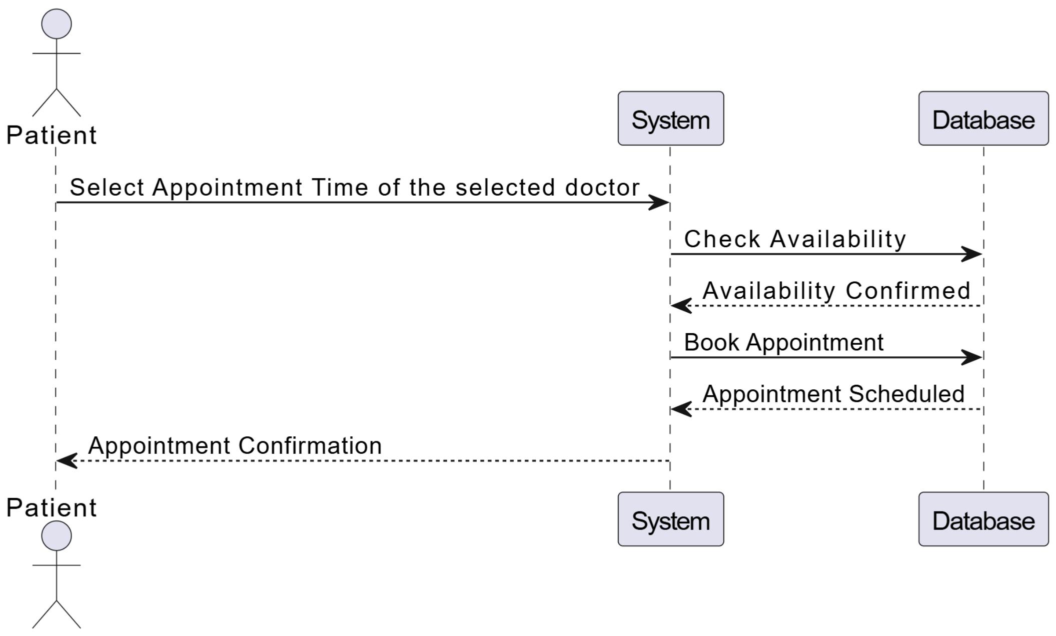
## Sequence Diagrams:
## 1. Registration



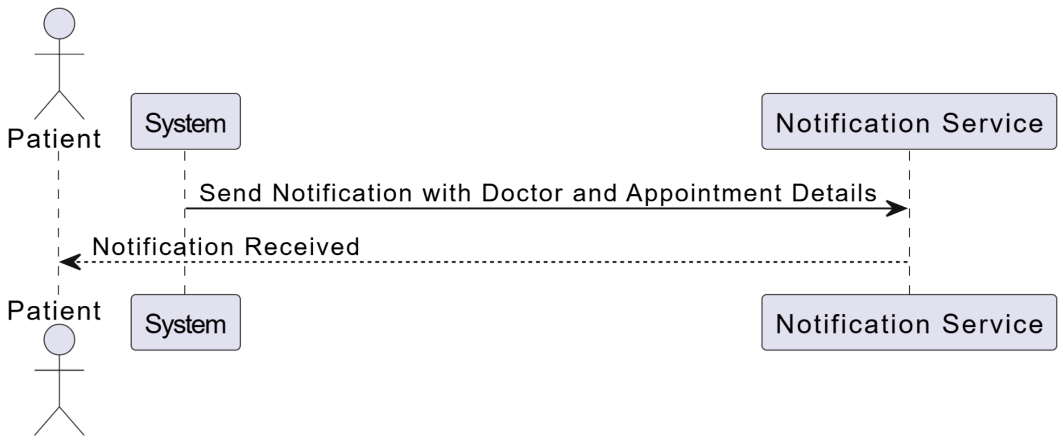
## 2. View Doctors (With Doctor Selection)



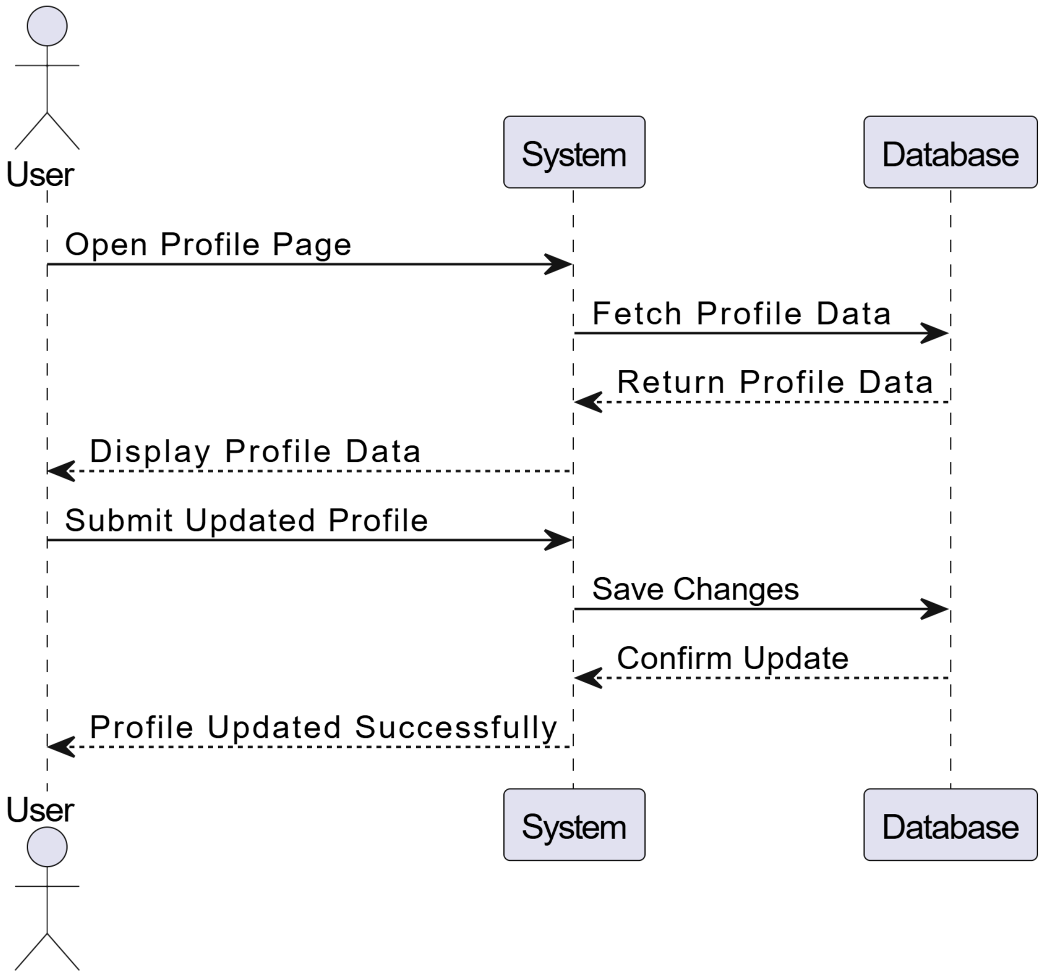
## 3. Book Appointment (With Time Selection)



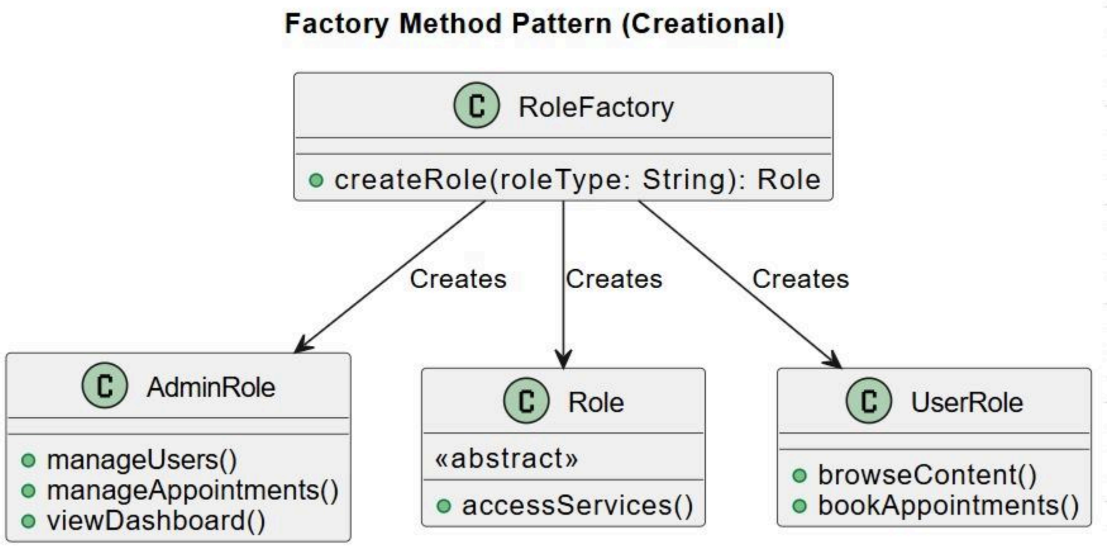
## 4. Notification (With Selected Doctor and Appointment Time)



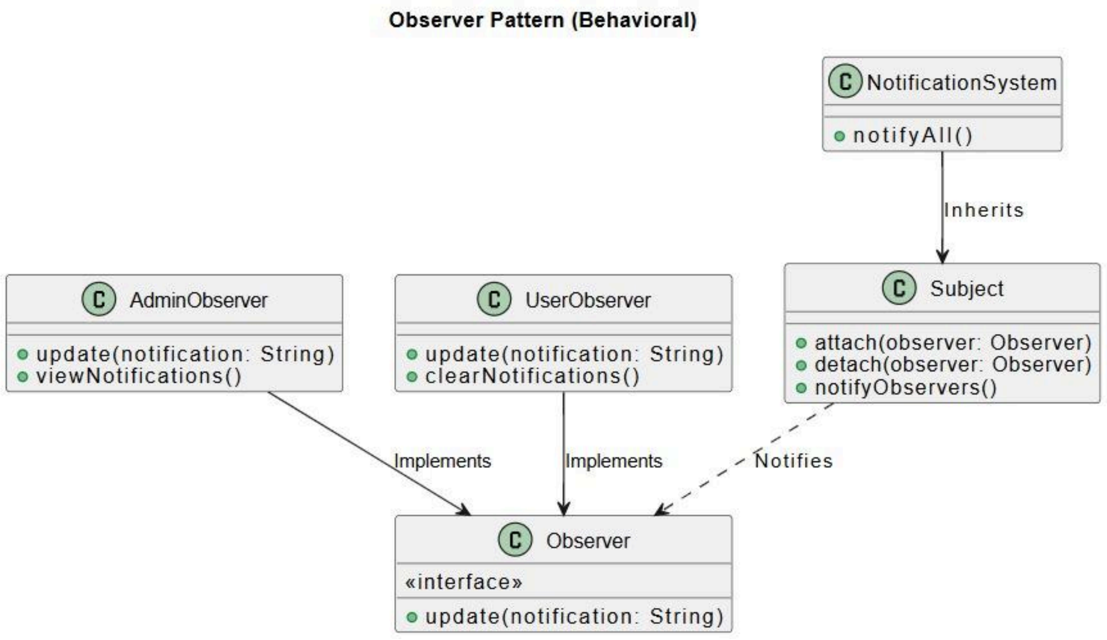
## 5. Edit Profile



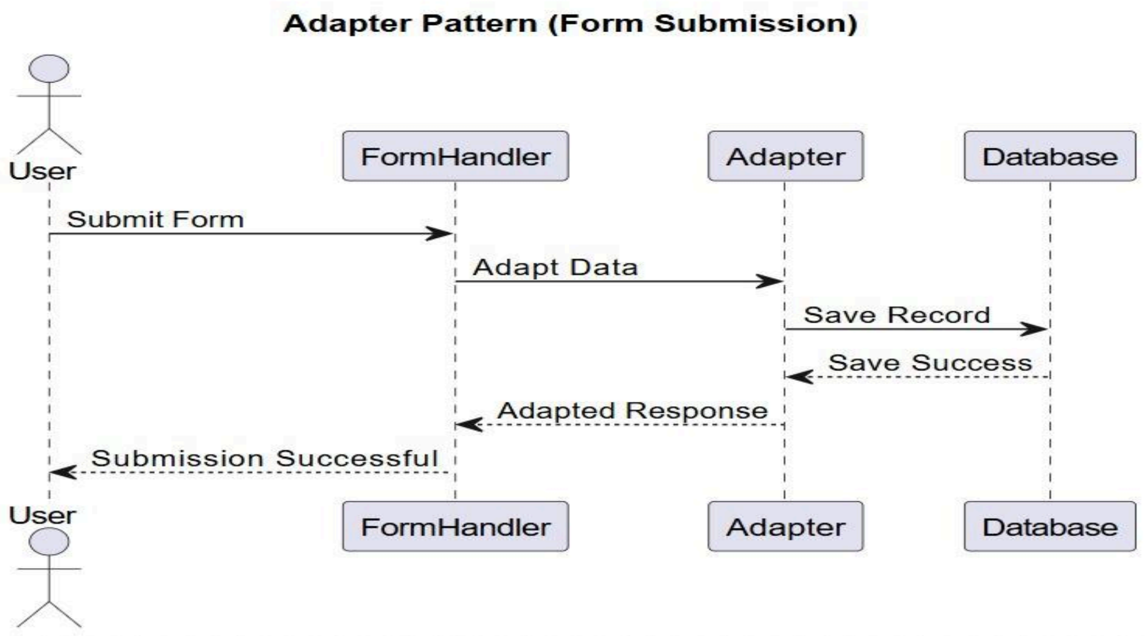
## UML for 4 design patterns:
## 1.(creational)



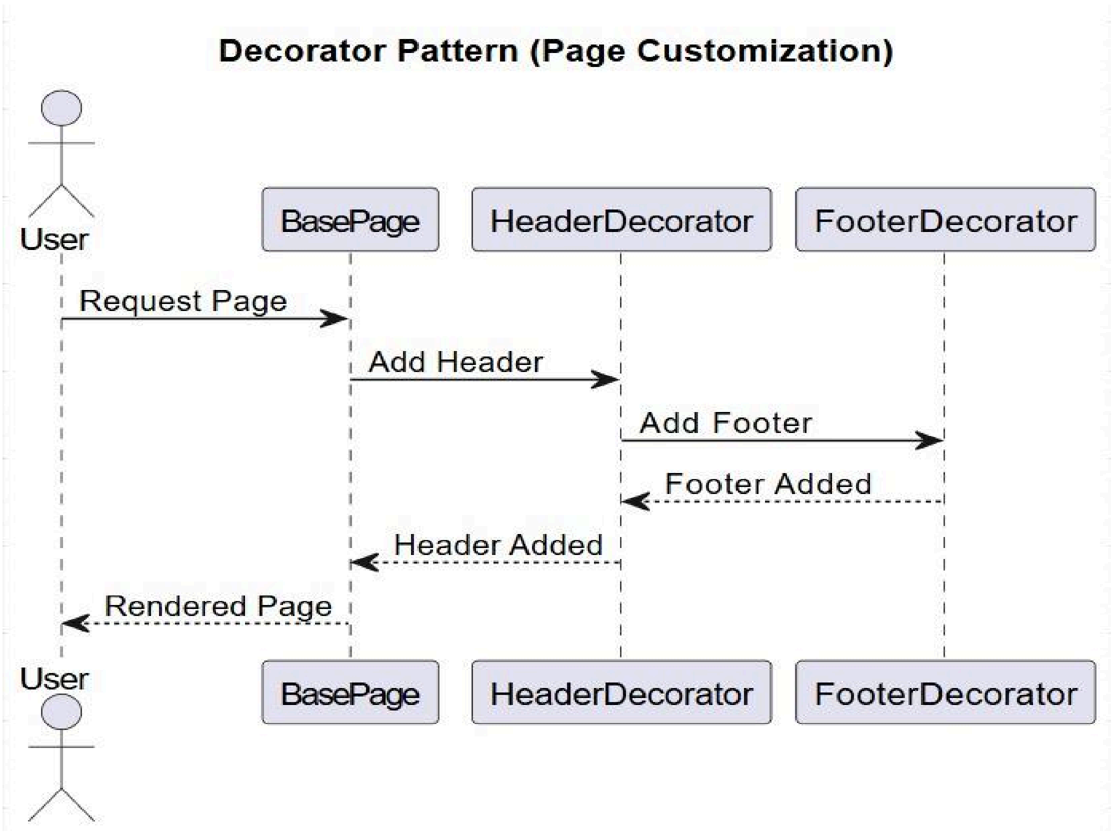
## 2.(behavioral)



## 3.(strctural)



## 4.(structural)


Decorator Pattern (Page Customization)
User
BasePage
HeaderDecorator
FooterDecorator
Request Page
Add Header
Add Footer
Footer Added
Header Added
Rendered Page
User
BasePage
HeaderDecorator
FooterDecorator