**Online Customized Gift Portal**

**Overview:**

An online customized gift portal is a platform that allows users to create and purchase personalized gifts for various occasions. These portals offer a wide range of customizable products, allowing customers to add their personal touch to gifts.

**Users of the System:**

**1 Admin**

**Role**:

·      Create account and manage orders for customer.

·      Manage the theme and gifts.

·      Review order.

·      Payment history

2.**Customer**

**Role**:

·      Create and manage their accounts

·      Create and manage orders.

·      Select themes and gifts

·      Make payments.

**Functional Requirements:**

·      Build an application that customers can buy customized gifts online.

·      The customers can choose themes, gifts, select gifts, order online.

·      The user will have option to edit and cancel the order.

·      The order value should be generated automatically.

·      The admin can add new themes, delete the themes, edit the theme details and view themes

·      The admin can also add, edit, view and delete gifts to be customized.

·      Customer can provide reviews.

·      Customer can make the payments.

**Non-Functional Requirements:**

1. **Security:** The system must implement robust security measures to protect user data, including user authentication, secure data storage, and encrypted data transmission.
2. **Scalability**: The system should be designed to handle an increasing number of gifts
3. **Usability**: The user interface should be intuitive and user-friendly, with responsive design for mobile and desktop users.
4. **Availability**: The system should be available 24/7 with minimal downtime for maintenance.
5. **Logging and Auditing**: Support logging and auditing of system activities for monitoring and troubleshooting.

**Modules of the Application:**

**ADMIN:**

Ø Register

Ø Login

Ø Dashboard

§ Gifts available

§ Themes available

§ Orders

§ Payment History

**CUSTOMER:**

Ø Register

Ø Login

Ø Dashboard

§ Theme menu

§ Gift menu

Ø Order page

Ø Payment page

**Technology Stack**

**Front End**

React , HTML, CSS

**Back End**

Java, Spring Boot, MySQL Database

**Authentication**

JWT for User Authentication

Application assumptions:

1.    The login page should be the first page rendered when the application loads.

2.    Manual routing should be restricted by implementing Auth Guard, utilizing the Can Activate interface. For example, if the user enters as  http://localhost:8080/signup or http://localhost:8080/home the page should not navigate to the corresponding page instead it should redirect to the login page.

3.    Unless logged into the system, the user cannot navigate to any other pages.

4.    Logging out must again redirect to the login page.

5.    To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.

6.    Design forgot password and forgot email buttons in login page

**Validations**:

1. Basic email validation should be performed.

2. Basic mobile number validation should be performed.

3. Basic password validation should be performed.

**Client-Side Validation:**

Implement client-side validation using HTML5 attributes and JavaScript to validate user input before making API requests.

Provide immediate feedback to users for invalid input, such as displaying error messages near the input fields.

**Server-Side Validation:**

Implement server-side validation in the controllers to ensure data integrity.

Validate user input and API responses to prevent unexpected or malicious data from affecting the application.

Return appropriate validation error messages to the user interface for any validation failures.

**Exception Handling**

Implement exception handling mechanisms in the controllers to gracefully handle errors and exceptions.

Define custom exception classes for different error scenarios, such as API communication errors or database errors.

Log exceptions for debugging purposes while presenting user-friendly error messages to users. Record all the exceptions and errors handled store in separate table “ErrorLogs”.

**Error Pages:**

**Create custom error pages for different HTTP status codes (e.g., 404** Not Found, **500** Internal Server Error) to provide a consistent and user-friendly error experience.

Ensure that error pages contain helpful information and guidance for users.

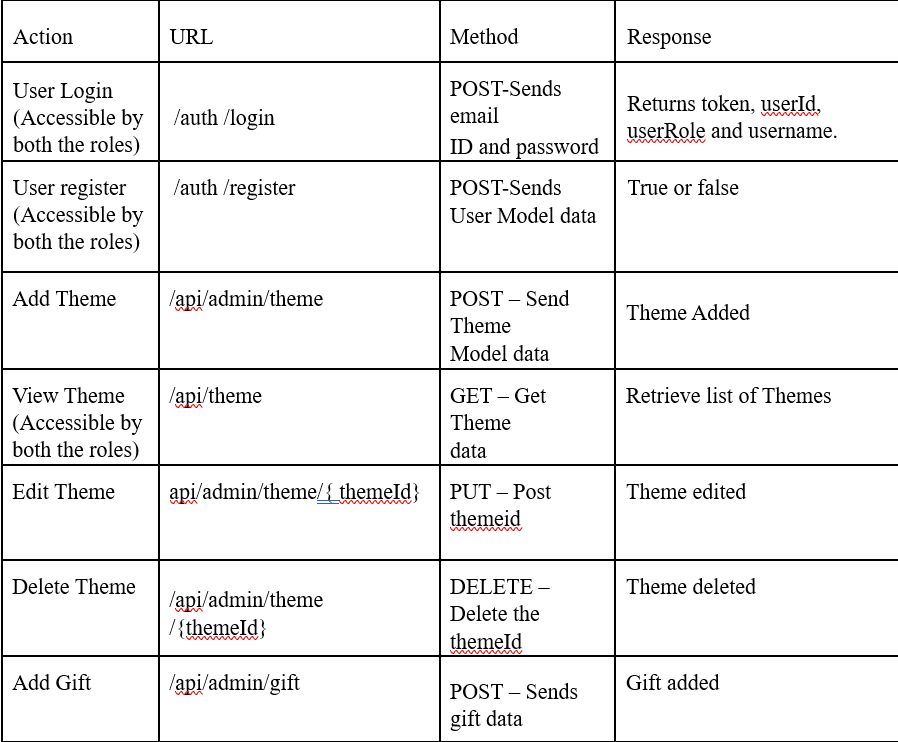
Thus, create a reliable and user-friendly web application that not only meets user expectations but also provides a robust and secure experience, even when faced with unexpected situations.

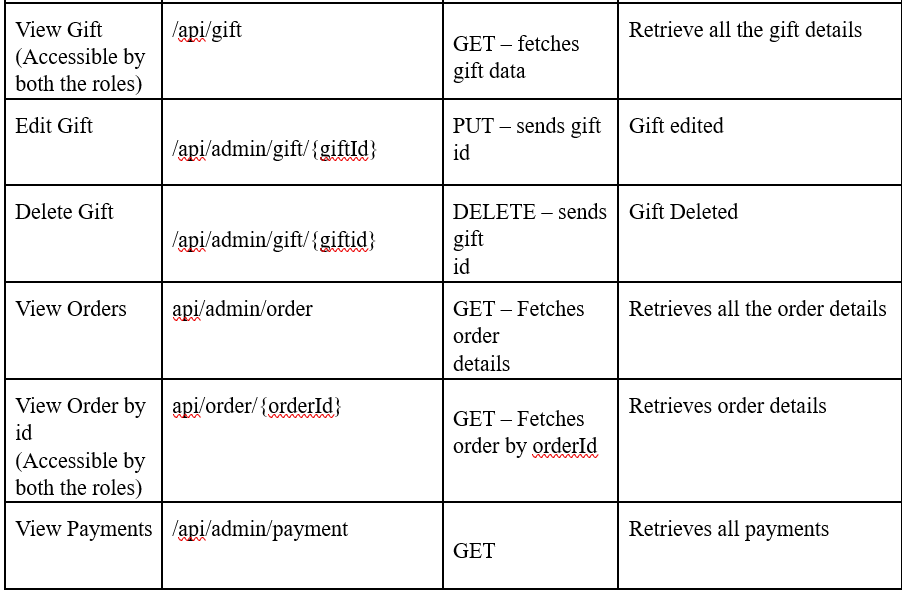
API Endpoints:

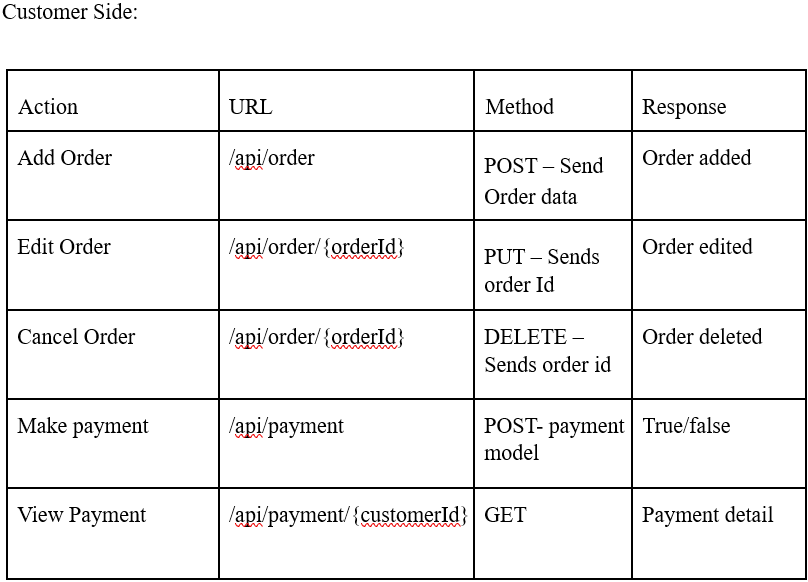
**Admin Role:**

* **Description:** This API endpoint is reserved for administrators and provides access to functionalities such as the admin dashboard, application review. Only users with ADMIN privileges should be able to access this endpoint.

**Admin Side**

****

****

**  
  
Backend :**

**Model Classes**

Ø **User**:

This class stores the user type (admin or the customer) and all user information.

§ email: String

§ userId: Long

§  password: String

§  username: String

§  mobileNumber: String

§  userRole: String

Ø **Customer**:

§ customerId; long

§ customerName: String

§ address: String

§ User user (OneToOne)

Ø **Theme**:

§ themeId; long

§ themeName: String

§ themeDetails: String

§ themePrice: Double

Ø **Gift**:

§ giftId: Long

§ giftName: String

§ GiftImageUrl: String

§ giftDetails: String

§ giftPrice: Double

§ List<Theme> themes (OneToMany)

Ø **Order**:

§ orderId: Long

§ orderDate: Date

§ orderPrice: Double (Combines both gift and theme price)

§ quantity: int

§ Gift gift (ManyToOne)

§ Customer: customer (ManyToOne)

§ Payment payment (OneToOne)

Ø **Payment**:

This class stores the details of the admission.

§ paymentId: long

§ status: String

§ amountpaid: Double

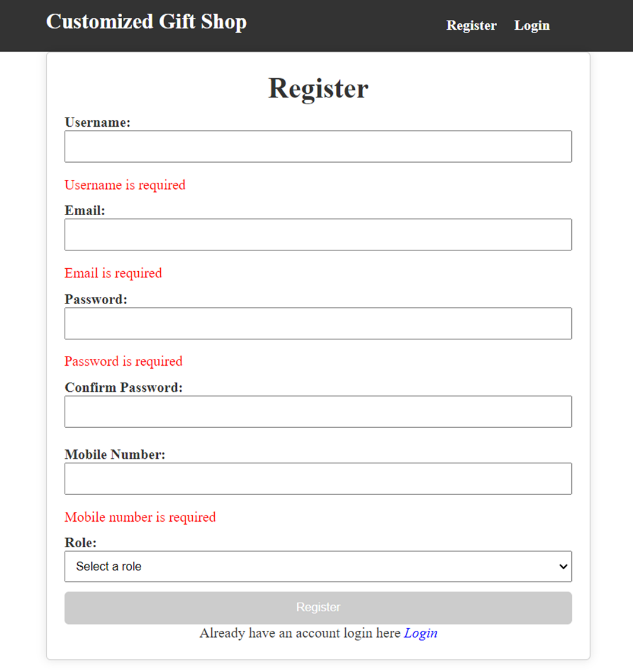
§ paymentDate: Date

§ modeOfPayment: String

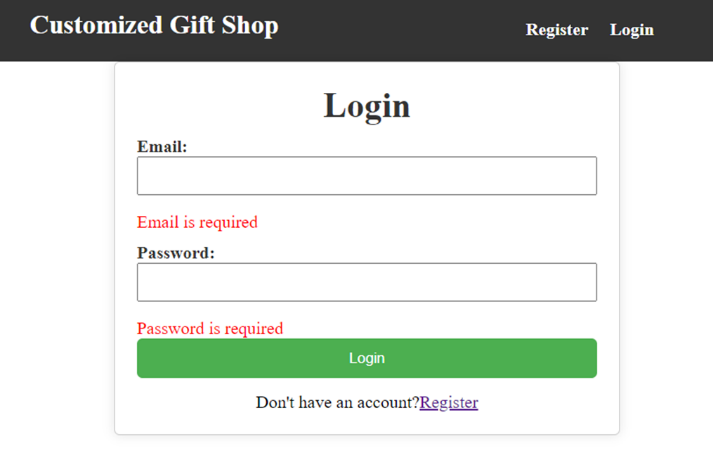
§ Customer customer (ManyToOne)

**Frontend Sample Screenshots:**

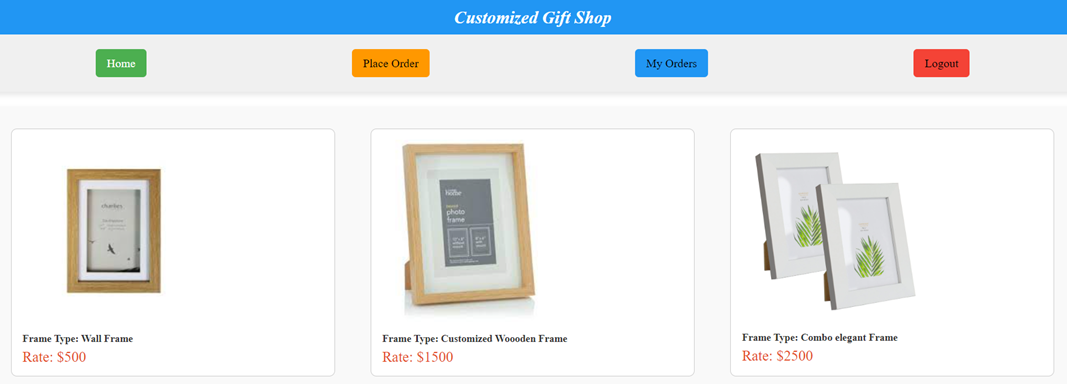
**Register Page: The Users can register the roles and details**



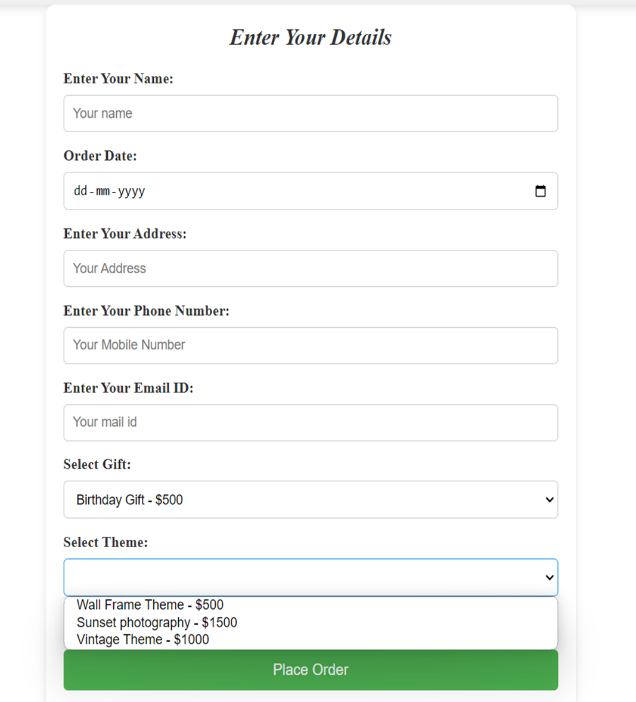
**Login Page**



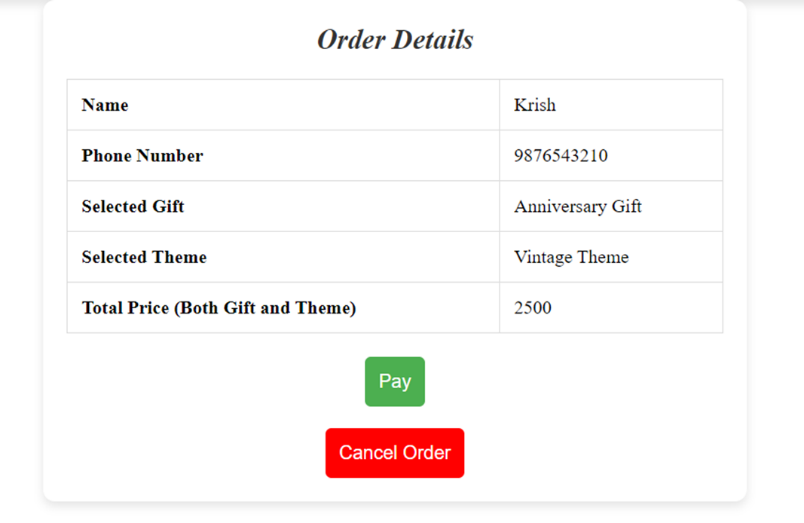
**Customer Side**



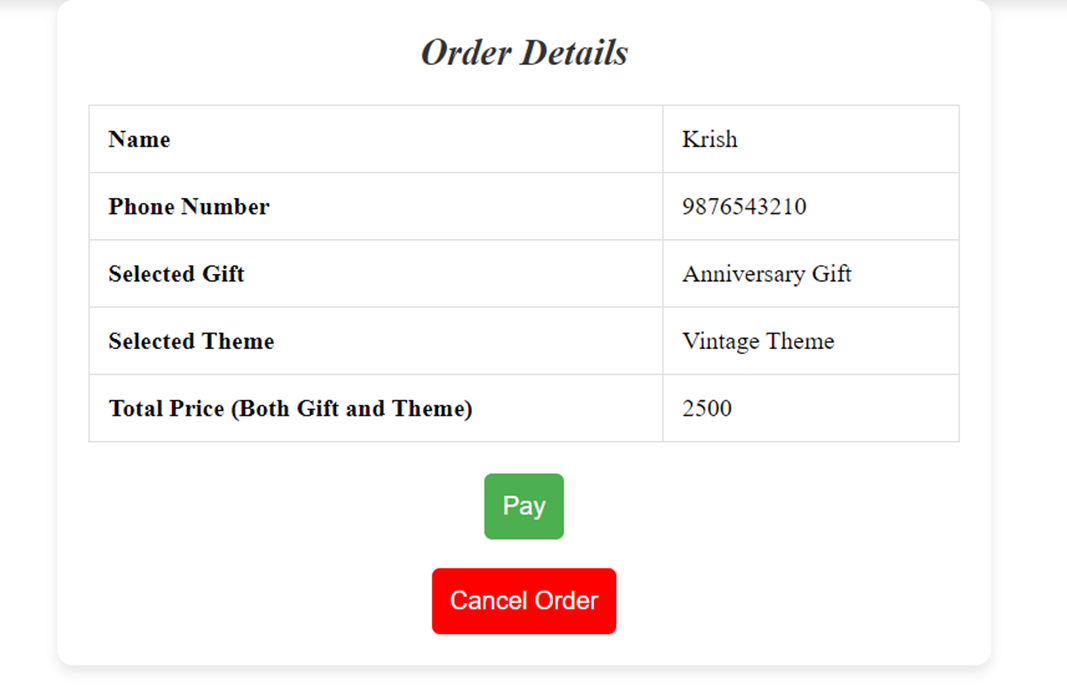
**Add Gift**



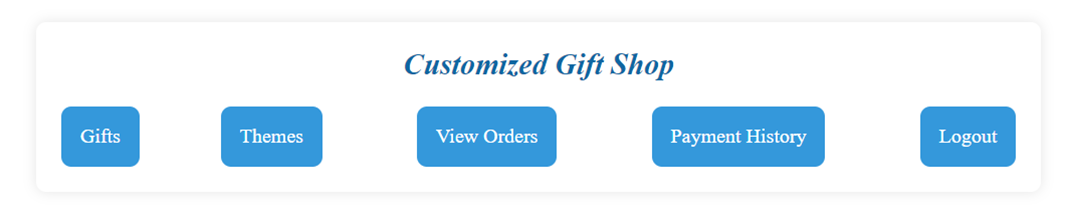
**My Orders Page:**



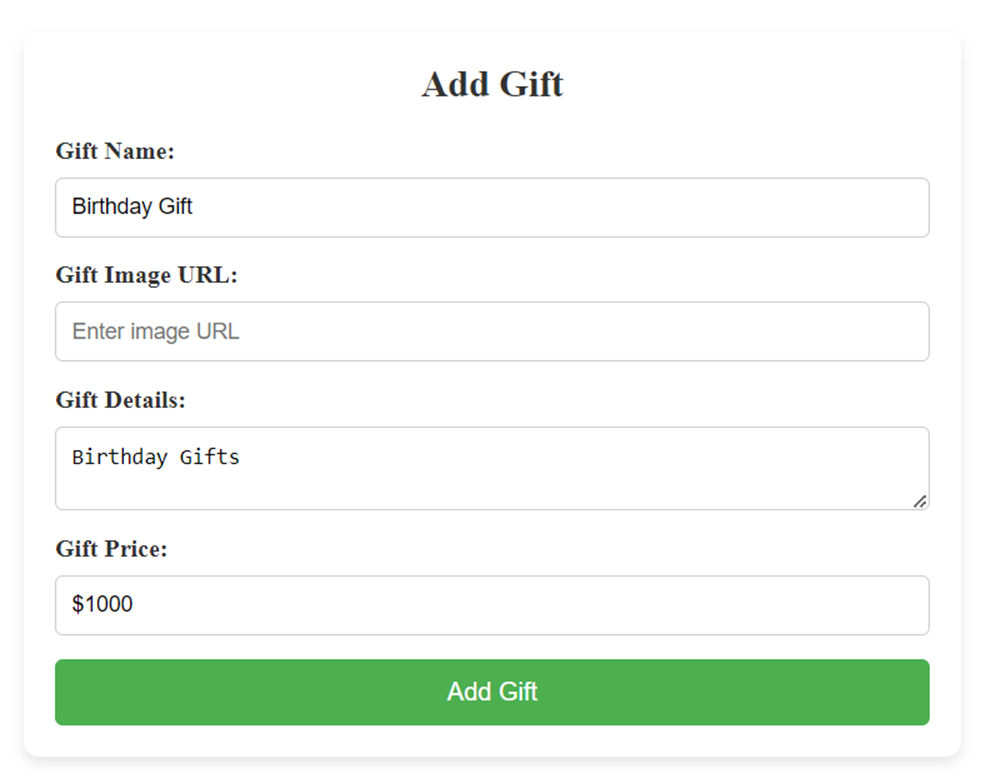
**Payment Page**

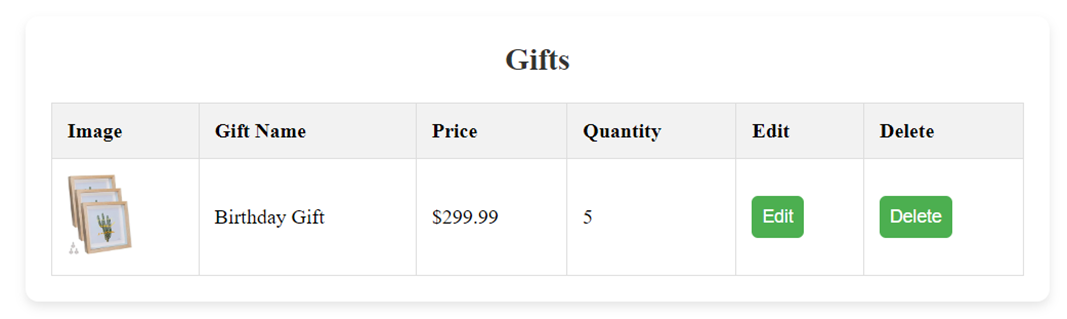


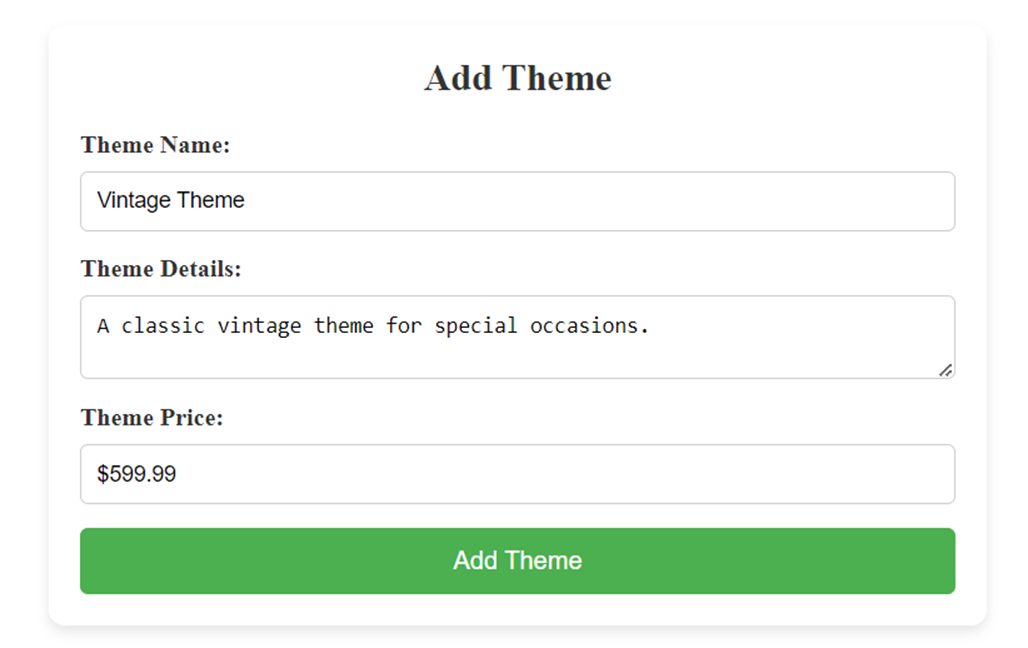
**ADMIN SIDE**

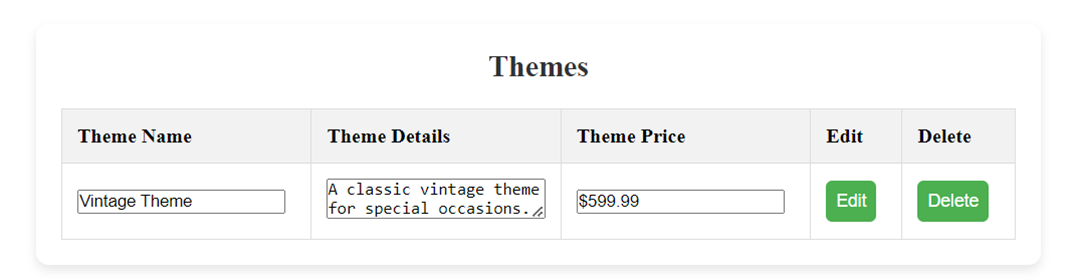


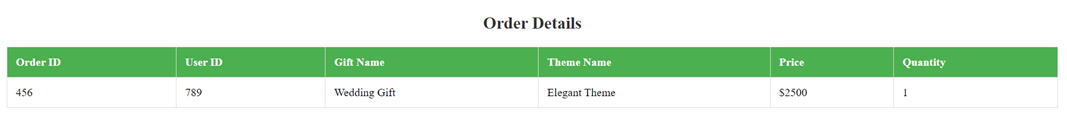
**Add Gift Page:**



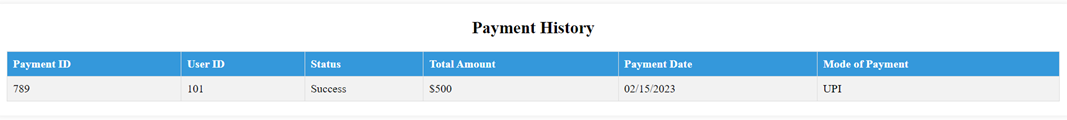
**After adding gift you can edit the Gift:  
  
**

**Add Theme Page:**  
  
  
  
**Also you can edit the theme.**

  
  
**Order Details:**



**Payment History:**

  
  
**Platform Prerequisites (Do’s and Don’ts):**

Ø The react app should run in port 8081.

Ø The spring boot app should run in port 8080.

**Other Important Key factors in the application:**

•       Should use Custom Exceptions mandatory.

•       Tables should have proper relationship and keys

•       Frontend Application should be menu driven.

•       Proper Menu / Navigation for corresponding role

•       Client side validations and server side validations are mandatory

•       Error should be handled.

•       Follow best programmer practice while developing

•       Provide proper Naming Conventions

•       Don't delete any files in a project environment.

•       You should use NotFound(), NoContent(), BadRequest(), CreatedAtAction() to handle the HTTP status code as return values for the Controller methods as mentioned.

•       Use MySql database

**How To Run the Project**

**Back End**

**API endpoint:**

8080

**Platform Guidelines:**

To run the command use **Terminal**in the platform.

**Spring Boot:**

Navigate to the springapp directory => **cd springapp**

To start/run the application '**mvn spring-boot:run**'

Click on the Run Test Case button to pass all the test cases

To connect the database

Cmd: **mysql -u root –protocol=tcp -p**

Password:examly

**HOW TO RUN THE PROJECT :**

**FRONTEND:**

**Step 1:**

Open the terminal

Use “nvm use 14” command to change node version to 14

**Step 1:**

Use "cd reactapp" command to go inside the reactapp folder

Install Node Modules **- "**npm install**"**

**Step 2:**

Write the code inside src folder

Create the necessary components

**Step 3:**

Click the run test case button to run the test cases

**Note :**

* Click PORT 8081 to view the result / output
* If any error persists while running the app , delete the node modules and reinstall them