SOFTWARE DEVELOPMENT PRACTICE

TEAM C. HOTEL MANAGEMENT SYSTEM

TEAM MEMBERS

Vindhya Hegde
Sushant Huilgol
Kamal Kumar Sardiwal
Cristian Kohn
Kalpesh Patil
Ambuj Solanki

21 DECEMBER 2023
SRH HEIDELBERG

Contents

1. Project Introduction

Welcome to the collaborative efforts to embark on the ambitious journey of developing an advanced Hotel Management System. This collective effort is driven by the shared vision of addressing the intricate challenges inherent in the hospitality industry and optimizing hotel

operations.

Project Overview

Our Hotel Management System is like a tool that is set to change how hotels work. It's made to be super easy for both the hotel workers and the guests. We used the latest technologies and the best ways of doing things in the hotel industry to make it. The way it's set up helps all the different parts of the system talk to each other smoothly, making sure everything runs well in the busy hotel. The system also has a strong booking feature that lets users easily make or cancel bookings.

2. Key Features

User Roles

1. Guest
2. Hotel Staff

1. Booking Management

- a) Guests are allowed to create new bookings after registration.
- b) Staff can book rooms for guests.
- c) The staff can also update the status of the booking when the guest checks in, checks out or wants to cancel the room.
- d) The system makes sure your booking goes through smoothly and updates you accordingly. If everything works as planned, it notes the successful booking and keeps a record of the event. However, if there's an issue, any amount debited will be refunded.

2. Room Management

- At the heart of our system lies an efficient room management module.
- Both guests (including unregistered guests) and hotel staff can check room availability based on dates and check the prices, services included.
- Currently these are the room types our system accommodates.
  1. Superior Room Sea View
  2. Superior Room Garden View
  3. Junior Suite Sea View
  4. Junior Suite Garden View
  5. Executive Room Sea View
  6. Executive Room Garden View
  7. Luxury Suite Sea View
  8. Luxury Suite Garden View

3. Additional Services

- Additionally, guests can avail themselves of various services during their stay, thereby enhancing overall comfort and convenience. From room service to special requests, the system facilitates a personalized and memorable guest experience.
- Any additional service availed during stay, will be added to booking by the hotel staff and billed during checkout.
- The Additional Services feature allows guests to conveniently add extra services to their online bookings, enhancing their overall experience.

4. Employee features

- To support hotel staff in their roles, our system incorporates functionalities like add services to existing bookings for guests during the stay, create holidays and view bookings.
- This includes features for creating and managing holidays ensuring the accurate room pricing based on demand.

5. Logging and Monitoring

- A comprehensive logging and monitoring system is integral to our design. This system tracks all relevant activities, aiding in issue resolution and providing valuable insights into system performance and user interactions.
- The Logging and Monitoring feature ensures comprehensive tracking of system activities. The logger, implemented using the spring framework's default logger, records events, and errors throughout the system, providing valuable insights for issue resolution and continuous improvement.

Future Scope

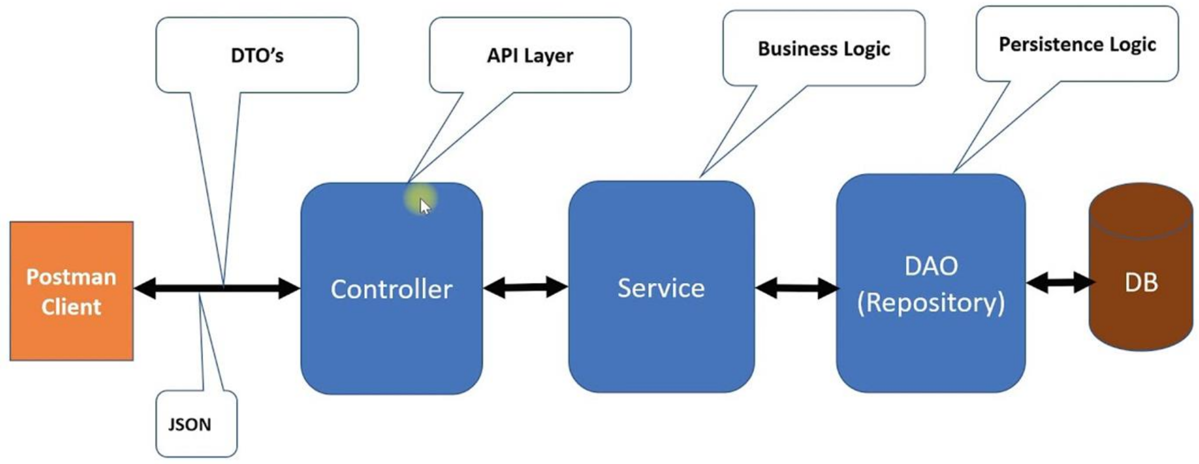
- Real-time updates that enable hotel staff to stay well-informed, fostering a well-organized and responsive service.
- A staff management system.
- In the future, when a feature allowing staff to update and create room types is developed, they can use it to customize the room types based on the hotel management's needs, through the user interface (UI).
- The calculation of surge fees will be addressed at a later time; currently, we have introduced a Surge Fee column in both the Holidays table and Room Types table.
- JWT encryption for storage in local storage is still pending, and due to time constraints, we have not yet implemented this process.

3. Technology Stack

- Frontend: The front end is developed using Angular, providing an intuitive user interface.
- Database: MySQL is employed as the database management system, offering effective data storage and retrieval.

- Backend: Spring Boot, along with Java™ Database Connectivity (JDBC), Spring Data JPA facilitates seamless communication between the application and the database. The project's build and dependency management are handled efficiently using the Maven tool.

4. Spring-Boot RESTful API Architecture



i. Presentation Layer:

Role: This layer deals with user interaction and presentation of data. It is responsible for receiving user requests, processing them, and returning appropriate responses.

In Spring Boot: In a Spring Boot API, the presentation layer is typically implemented using controllers. Controllers receive HTTP requests, handle them, and communicate with the service layer to process the business logic.

Example: In a hotel management web application, a presentation layer includes controllers that handle requests for user registration, login, retrieving data, making bookings and every other api.

ii. Service/Business Layer:

Role: This layer contains the business logic of the application. It performs operations and implements business rules. It acts as an intermediary between the presentation layer and the persistence layer.

In Spring Boot: Services in Spring Boot encapsulate the business logic. They are responsible for processing data, applying business rules, and coordinating with the persistence layer for data storage and retrieval.

Example: In the hotel management system, the service layer contains logic for making booking, payment, retrieving status based booking data, retrieving room data based on check-in and check-out dates etc.

iii. Persistence Layer:

Role: This layer is responsible for storing and retrieving data from a database. It deals with the actual data storage and retrieval operations.

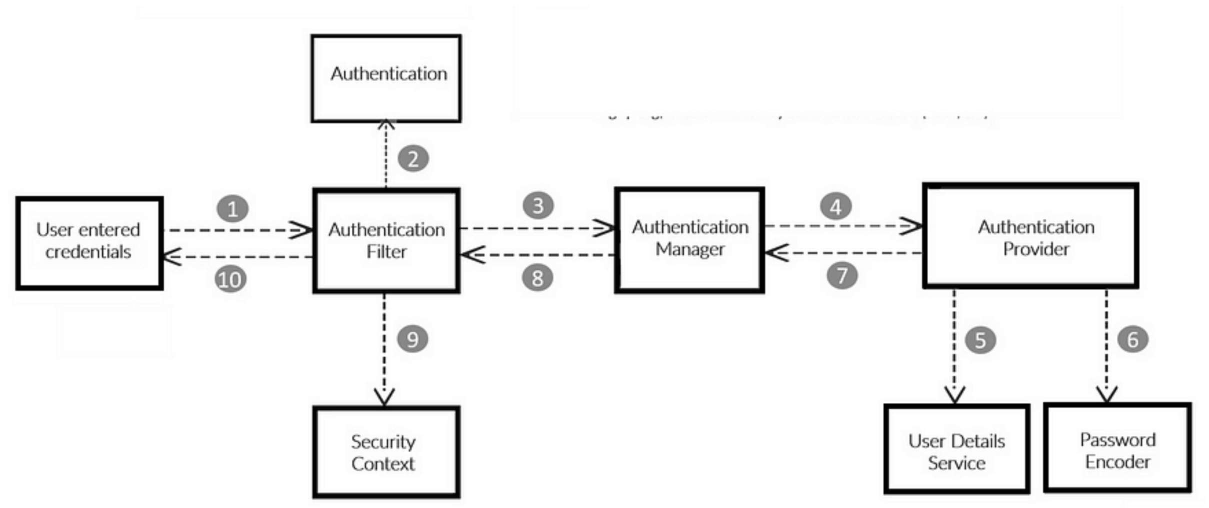
In Spring Boot: The persistence layer is often implemented using repositories, which are interfaces extending JpaRepository. These repositories provide methods for common database operations.

Example: In the implemented hotel management application, the persistence layer handles storing and retrieving bookings, rooms, users from a database.

List of APIs:

Functionality	Route	Method
Login	/api/auth/login	POST
Guest Registration	/api/auth/register	POST
Book Room	/api/room/book	POST
Check In Room	/api/room/checkin/{bookingId}	POST
Check Out Room	/api/room/checkout/{bookingId}	POST
Cancel Room	/api/room/cancel/{bookingId}	POST
Refund	/api/room/refund/{bookingId}	POST
Fetch Rooms Available	/api/rooms	GET
Fetch Services	/api/rooms/services	GET
Create Holidays	/api/employees/holidays	POST
Fetch Holidays	/api/employees/holidays	GET

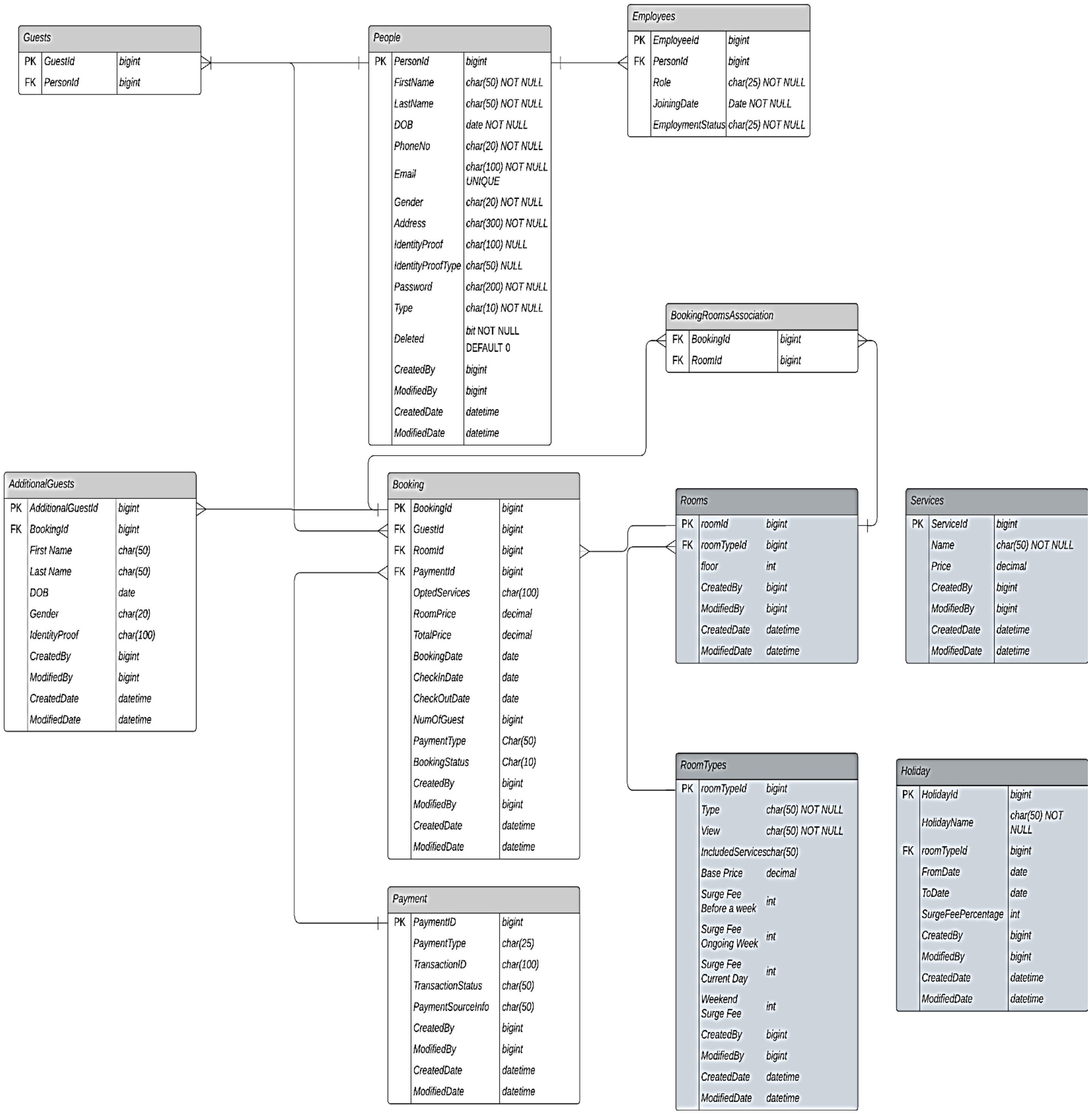
5. Spring Security Flow



i. When the user enters the credentials, an authentication filter present in spring security framework intercepts the request.

ii. After that it will try to convert the authentication details received from the user into an authentication object. This object is the base, where all the validation of user credentials will be validated in further steps.
iii. Authentication Manager will identify what is the authentication provider that the request has to go. The HMS uses database to validate credentials.
iv. Authentication Provider uses two other interfaces CustomUserDetailsService which extends UserDetailsService and Password Encoder.
v. A User Details Service holds the user schema, like how the user details should look like. Password Encoder will tell how the password has to be encoded/ decrypted.
vi. Once Authentication Provider validates the input using User Details Service and Password Encoder, it will transfer the request to Authentication Manager, followed by Authentication Filter.
vii. Now the authentication object, which we had initially sent from the Authentication Filter will hold the information whether the user is valid or not, along with other details like authorities, roles etc.
viii. The authentication filter will pass the authentication object to Security Context, where the details will be stored in the (Spring) container. This authentication object is given back to the browser, when the browser wants to send the request next time, Spring security will validate if the authentication object has valid token or not.

6. ER Diagram



Guests		
PK	GuestId	bigint
FK	PersonId	bigint

People		
PK	PersonId	bigint
	FirstName	char(50) NOT NULL
	LastName	char(50) NOT NULL
	DOB	date NOT NULL
	PhoneNo	char(20) NOT NULL
	Email	char(100) NOT NULL UNIQUE
	Gender	char(20) NOT NULL
	Address	char(300) NOT NULL
	IdentityProof	char(100) NULL
	IdentityProofType	char(50) NULL
	Password	char(200) NOT NULL
	Type	char(10) NOT NULL
	Deleted	bit NOT NULL DEFAULT 0
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

Employees		
PK	EmployeeId	bigint
FK	PersonId	bigint
	Role	char(25) NOT NULL
	JoiningDate	Date NOT NULL
	EmploymentStatus	char(25) NOT NULL

BookingRoomsAssociation		
FK	BookingId	bigint
FK	RoomId	bigint

AdditionalGuests		
PK	AdditionalGuestId	bigint
FK	BookingId	bigint
	First Name	char(50)
	Last Name	char(50)
	DOB	date
	Gender	char(20)
	IdentityProof	char(100)
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

Booking		
PK	BookingId	bigint
FK	GuestId	bigint
FK	RoomId	bigint
FK	PaymentId	bigint
	OptedServices	char(100)
	RoomPrice	decimal
	TotalPrice	decimal
	BookingDate	date
	CheckInDate	date
	CheckOutDate	date
	NumOfGuest	bigint
	PaymentType	Char(50)
	BookingStatus	Char(10)
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

Rooms		
PK	roomId	bigint
FK	roomTypeId	bigint
	floor	int
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

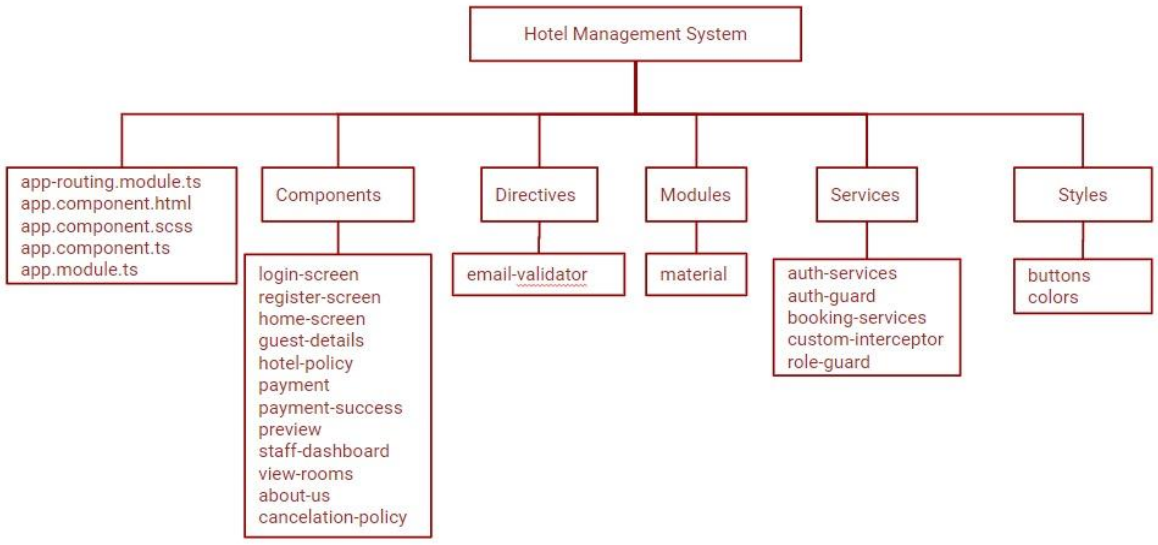
Services		
PK	ServiceId	bigint
	Name	char(50) NOT NULL
	Price	decimal
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

Payment		
PK	PaymentID	bigint
	PaymentType	char(25)
	TransactionID	char(100)
	TransactionStatus	char(50)
	PaymentSourceInfo	char(50)
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

RoomTypes		
PK	roomTypeId	bigint
	Type	char(50) NOT NULL
	View	char(50) NOT NULL
	IncludedServices	char(50)
	Base Price	decimal
	Surge Fee Before a week	int
	Surge Fee Ongoing Week	int
	Surge Fee Current Day	int
	Weekend Surge Fee	int
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

Holiday		
PK	HolidayId	bigint
	HolidayName	char(50) NOT NULL
FK	roomTypeId	bigint
	FromDate	date
	ToDate	date
	SurgeFeePercentage	int
	CreatedBy	bigint
	ModifiedBy	bigint
	CreatedDate	datetime
	ModifiedDate	datetime

7. Angular Structure



Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

The structure of our angular project is customized in such a way, that it supports reusability and such that it is easy to maintain. Following are the in-detail description of the project folders and files:

i. App Component

In an Angular application, these files play crucial roles in defining the structure, functionality, and styling of the application:

app-routing.module.ts:

- This file is used for setting up the routing configuration in an Angular application.
- It typically imports Angular's RouterModule and defines the routes for different components.
- It contains an array of route configurations that map specific URLs to corresponding components.
- The RouterModule.forRoot() method is used to configure the application-wide routes.

app.component.html:

- This file represents the HTML template associated with the root component of the Angular application.
- It contains the HTML markup that defines the initial view structure of the application.
- Angular components use this file to display the content and layout that users see in the browser.

app.module.ts:

- This file is the main module file of the Angular application.
- It defines the root module where other modules, components, services, and features are imported and configured.
- It uses the NgModule decorator to specify metadata about the application, including the components, directives, services, and the root component (AppComponent) that will bootstrap the application.
- It also imports other modules like BrowserModule, HttpClientModule, and additional custom modules required for the application.

app.component.ts:

- This file contains the TypeScript code for the root component (AppComponent) of the Angular application.
- It defines the component class that interacts with the HTML template (app.component.html) and encapsulates the component's logic and behavior.
- The @Component decorator is used to provide metadata such as selector, template, styles, etc., linking this TypeScript file with its associated HTML and SCSS/CSS files.

app.component.scss:

- This file contains the styles (CSS or SCSS) specific to the AppComponent.
- It is where you define the component-specific styles using CSS rules or preprocessors like Sass/SCSS.
- Styles defined in this file will be scoped to the AppComponent, following Angular's encapsulation mechanism unless specified otherwise.

These files together form the foundational structure of an Angular application. They establish the routing, define the root component, connect the logic with the UI via templates, and manage styling for the application's initial view. Understanding and appropriately utilizing these files is crucial for building and organizing an Angular application effectively.

ii. Common

The Common folder includes components that remain constant throughout the application such as header and footer. The header includes important links to Login, Register and Logout.

iii. Components

This folder contains all the components required that we see within the header and the footer bars in our UI. These components contain aesthetic UI and logic written in typescript that makes the UI dynamic.

iv. Directives

Directives in Angular extend HTML with new attributes, allowing developers to create reusable components, alter the appearance, behavior, or structure of elements, and build dynamic views. The custom directive in our project is used across the application to validate the email format.

v. Modules

In Angular, module.ts files are TypeScript files that define and configure modules using the @NgModule decorator. These files play a pivotal role in organizing and structuring an Angular application.

vi. Services

This folder includes all the service files that are used to perform dependency injection. The service files basically include the request to the API endpoints using HTTP methods.

The folder also includes guard files to manage routing based on specific role and also based on authentication credentials. It restricts the user to navigate within his defined routes.

The folder also includes an interceptor that handles any errors apart from 200 status code from the APIs and does the appropriate actions.

vii. Styles

The styles folder includes the scss files which contain the common CSS styles to be followed across the application like the color pallet, the styling of the buttons across the application. Centralizing styles in common stylesheets ensures consistency in the appearance and layout across different components and modules within the Angular application. This helps maintain a uniform look and feel, enhancing the user experience.

8. Conclusion

In summary, our Hotel Management System is a demonstration of the collaborative efforts and dedication of our team. With a dependable and effective solution for the dynamic hospitality industry, this project hopes to make a major contribution to the improvement of hotel operations.