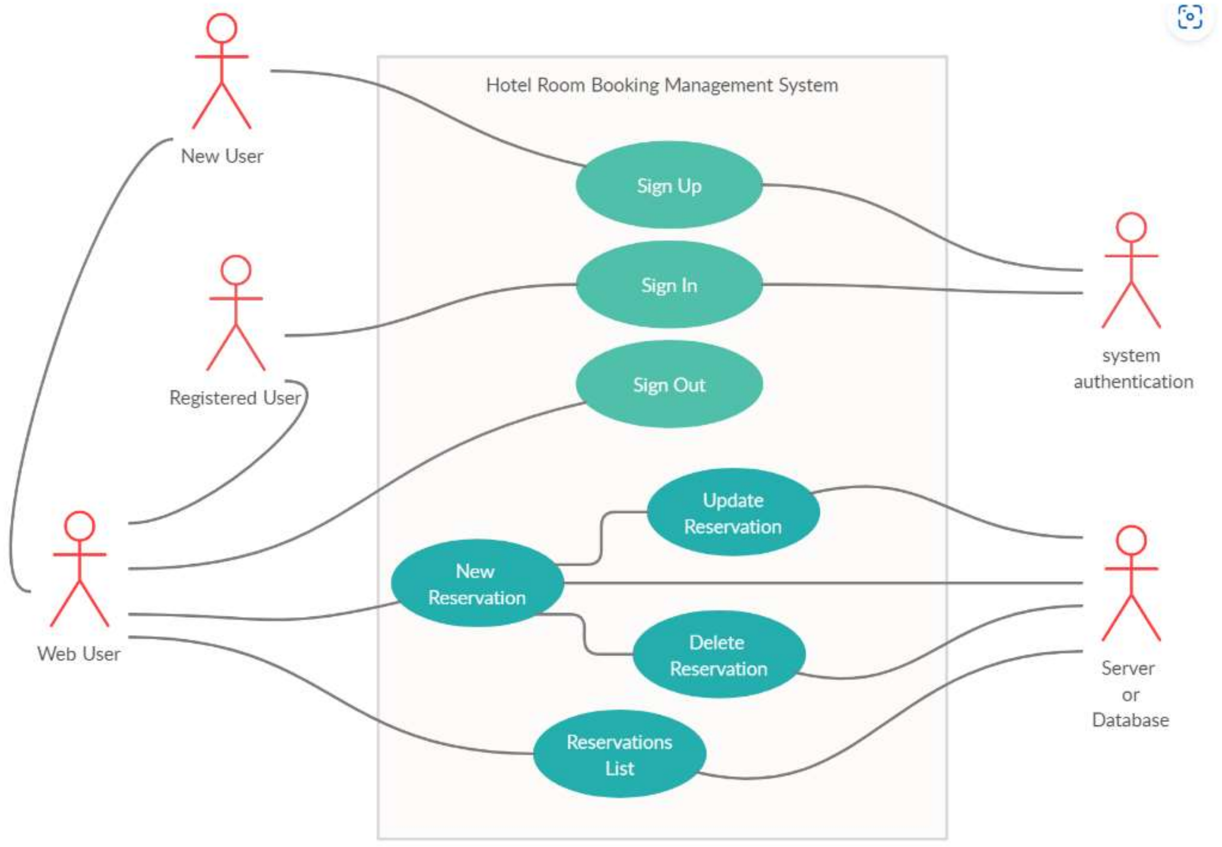
FSD C5 - Hotel Room Booking Application

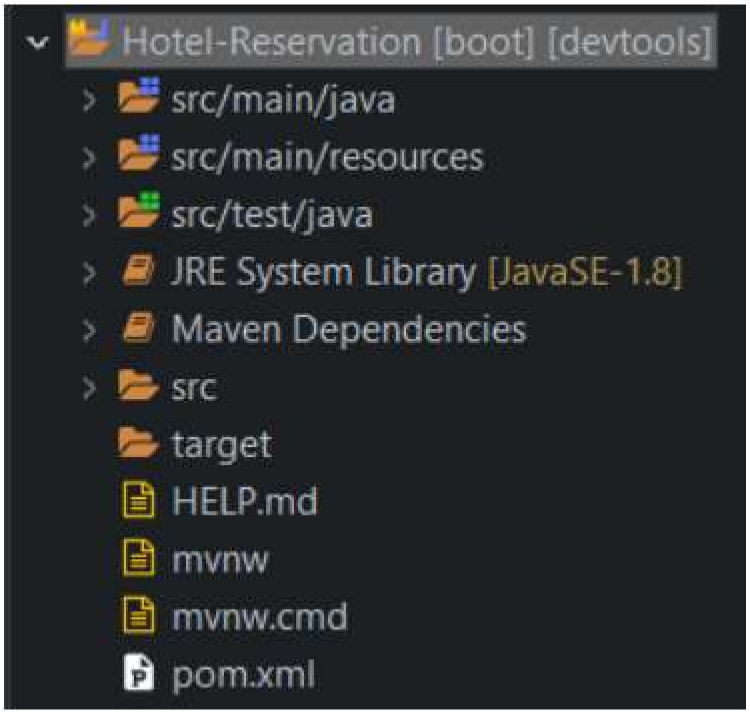
Hotel Room Booking Management System, It’s a Web project using Spring Framework in java, Spring Boot in back-end with html template engine for manage front-end, In addition to Spring Security to handle authentication and authorization, And Java Persistence API (JPA) with entity manager to handle MySql connector with database. Entity classes defined and two Structural Patterns Services and Dao, And using Maven build tool to manage project architecture for classes, resources and dependencies. In the website you can sign up and get a new account of your own and login, every account can book, update and delete reservation form (CRUD) features.

UML System Design

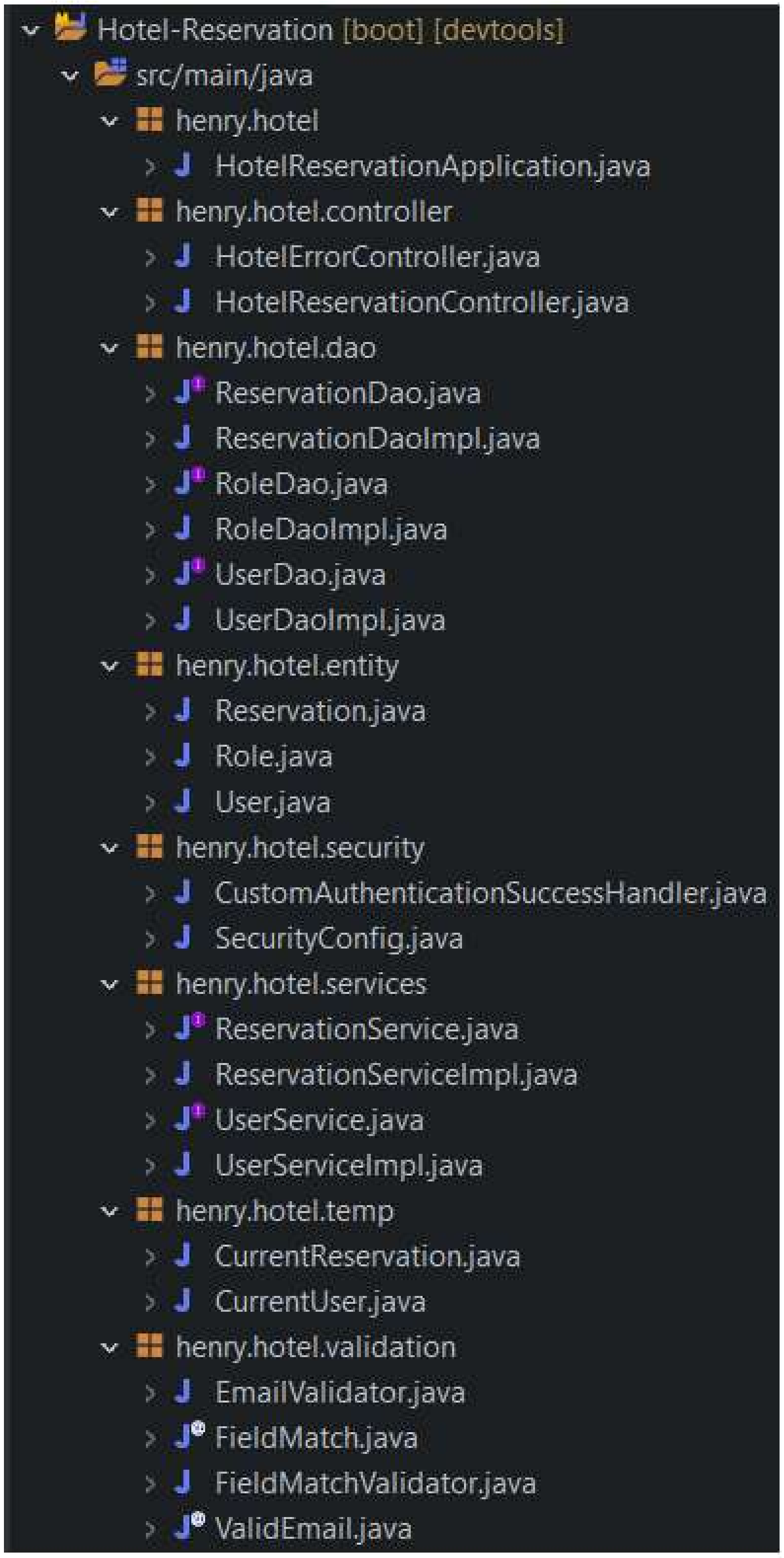


Hotel System Design Architecture

• Using Maven Build Tool.

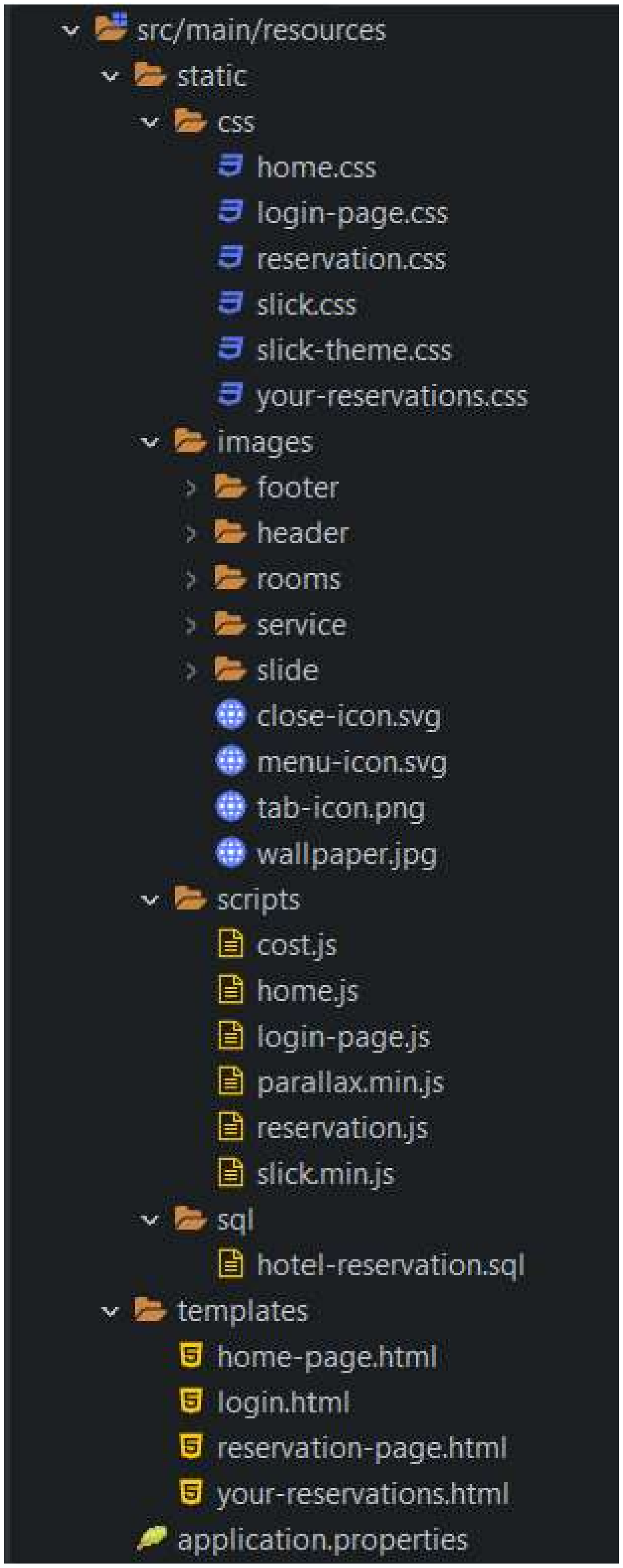

```
Hotel-Reservation [boot] [devtools]
    src/main/java
    src/main/resources
    src/test/java
    JRE System Library [JavaSE-1.8]
    Maven Dependencies
    src
    target
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
```

• Back-End Packages



```
Hotel-Reservation [boot] [devtools]
  src/main/java
    henry.hotel
      HotelReservationApplication.java
    henry.hotel.controller
      HotelErrorController.java
      HotelReservationController.java
    henry.hotel.dao
      ReservationDao.java
      ReservationDaoImpl.java
      RoleDao.java
      RoleDaoImpl.java
      UserDao.java
      UserDaoImpl.java
    henry.hotel.entity
      Reservation.java
      Role.java
      User.java
    henry.hotel.security
      CustomAuthenticationSuccessHandler.java
      SecurityConfig.java
    henry.hotel.services
      ReservationService.java
      ReservationServiceImpl.java
      UserService.java
      UserServiceImpl.java
    henry.hotel.temp
      CurrentReservation.java
      CurrentUser.java
    henry.hotel.validation
      EmailValidator.java
      FieldMatch.java
      FieldMatchValidator.java
      ValidEmail.java
```

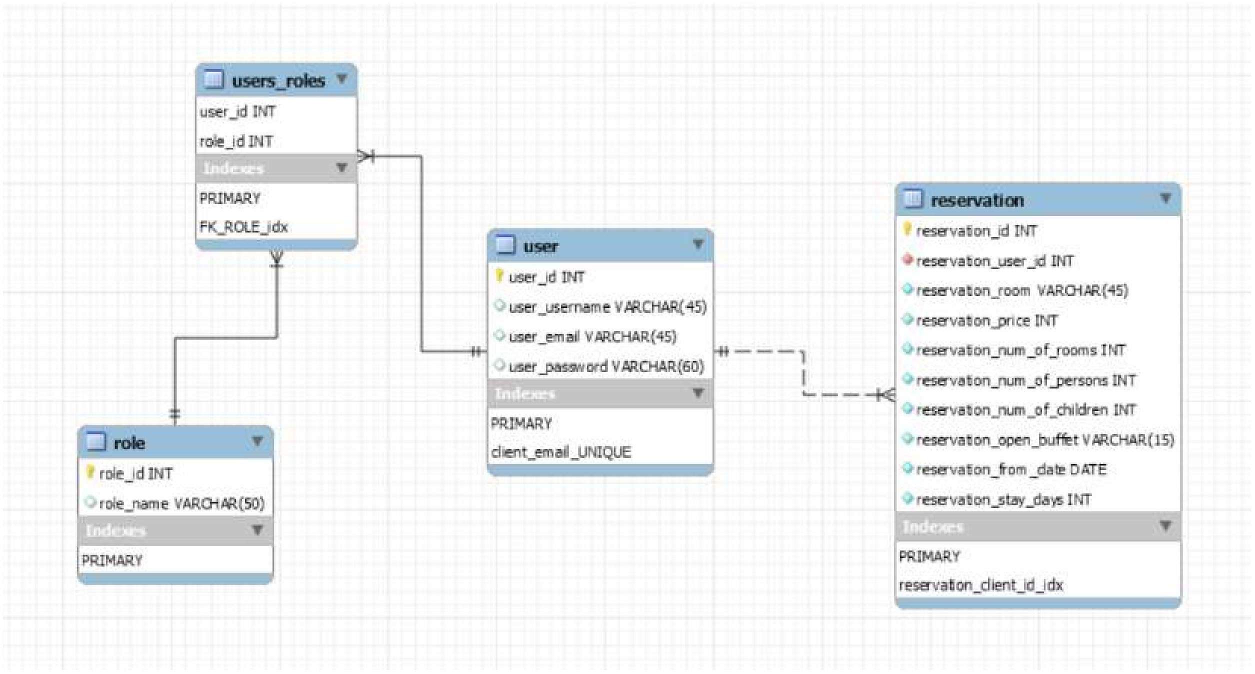
• Front-End Packages



```
src/main/resources
  static
    css
      home.css
      login-page.css
      reservation.css
      slick.css
      slick-theme.css
      your-reservations.css
    images
      footer
      header
      rooms
      service
      slide
      close-icon.svg
      menu-icon.svg
      tab-icon.png
      wallpaper.jpg
    scripts
      cost.js
      home.js
      login-page.js
      parallax.min.js
      reservation.js
      slick.min.js
    sql
      hotel-reservation.sql
  templates
    home-page.html
    login.html
    reservation-page.html
    your-reservations.html
  application.properties
```

Database Design Architecture

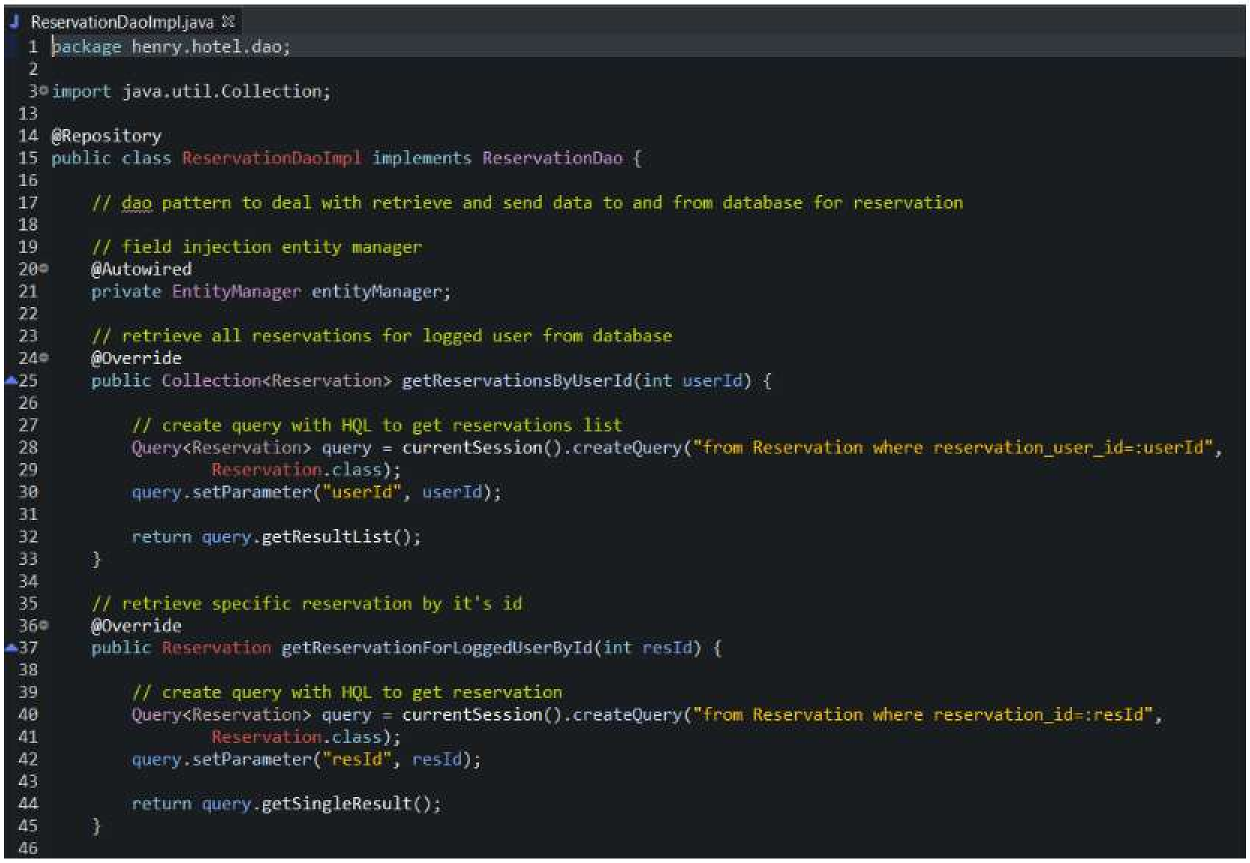
• Using MySql open source database.


users_roles
user_id INT
role_id INT
Indexes
PRIMARY
FK_ROLE_idx
user
user_id INT
user_username VARCHAR(45)
user_email VARCHAR(45)
user_password VARCHAR(60)
Indexes
PRIMARY
client_email_UNIQUE
reservation
reservation_id INT
reservation_user_id INT
reservation_room VARCHAR(45)
reservation_price INT
reservation_num_of_rooms INT
reservation_num_of_persons INT
reservation_num_of_children INT
reservation_open_buffet VARCHAR(15)
reservation_from_date DATE
reservation_stay_days INT
Indexes
PRIMARY
reservation_client_id_idx
role
role_id INT
role_name VARCHAR(50)
Indexes
PRIMARY

System Engineering Life Cycle

• Data Access Object Pattern (DAO):

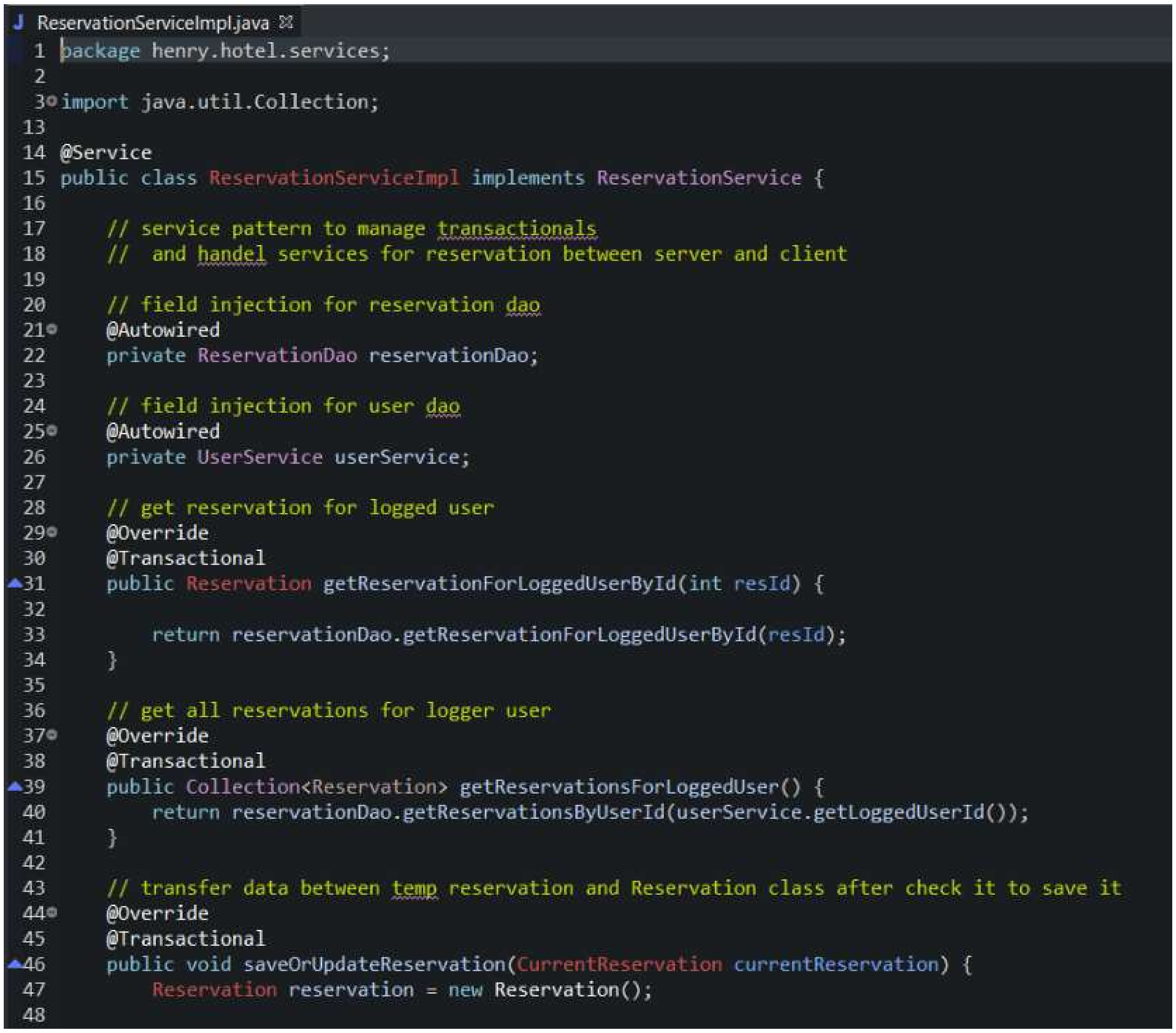
It’s a pattern that used deal with retrieve data from database and to save data too, using JPA with entity manager we can make a custom query with JPQL or can use modified methods. Example: Reservation Dao Implementation Class.



```java
package henry.hotel.dao;

import java.util.Collection;

@Repository
public class ReservationDaoImpl implements ReservationDao {

    // dao pattern to deal with retrieve and send data to and from database for reservation

    // field injection entity manager
    @Autowired
    private EntityManager entityManager;

    // retrieve all reservations for logged user from database
    @Override
    public Collection<Reservation> getReservationsByUserId(int userId) {

        // create query with HQL to get reservations list
        Query<Reservation> query = currentSession().createQuery("from Reservation where reservation_user_id=:userId",
                Reservation.class);
        query.setParameter("userId", userId);

        return query.getResultList();
    }

    // retrieve specific reservation by it's id
    @Override
    public Reservation getReservationForLoggedUserById(int resId) {

        // create query with HQL to get reservation
        Query<Reservation> query = currentSession().createQuery("from Reservation where reservation_id=:resId",
                Reservation.class);
        query.setParameter("resId", resId);

        return query.getSingleResult();
    }

```

Service Pattern:

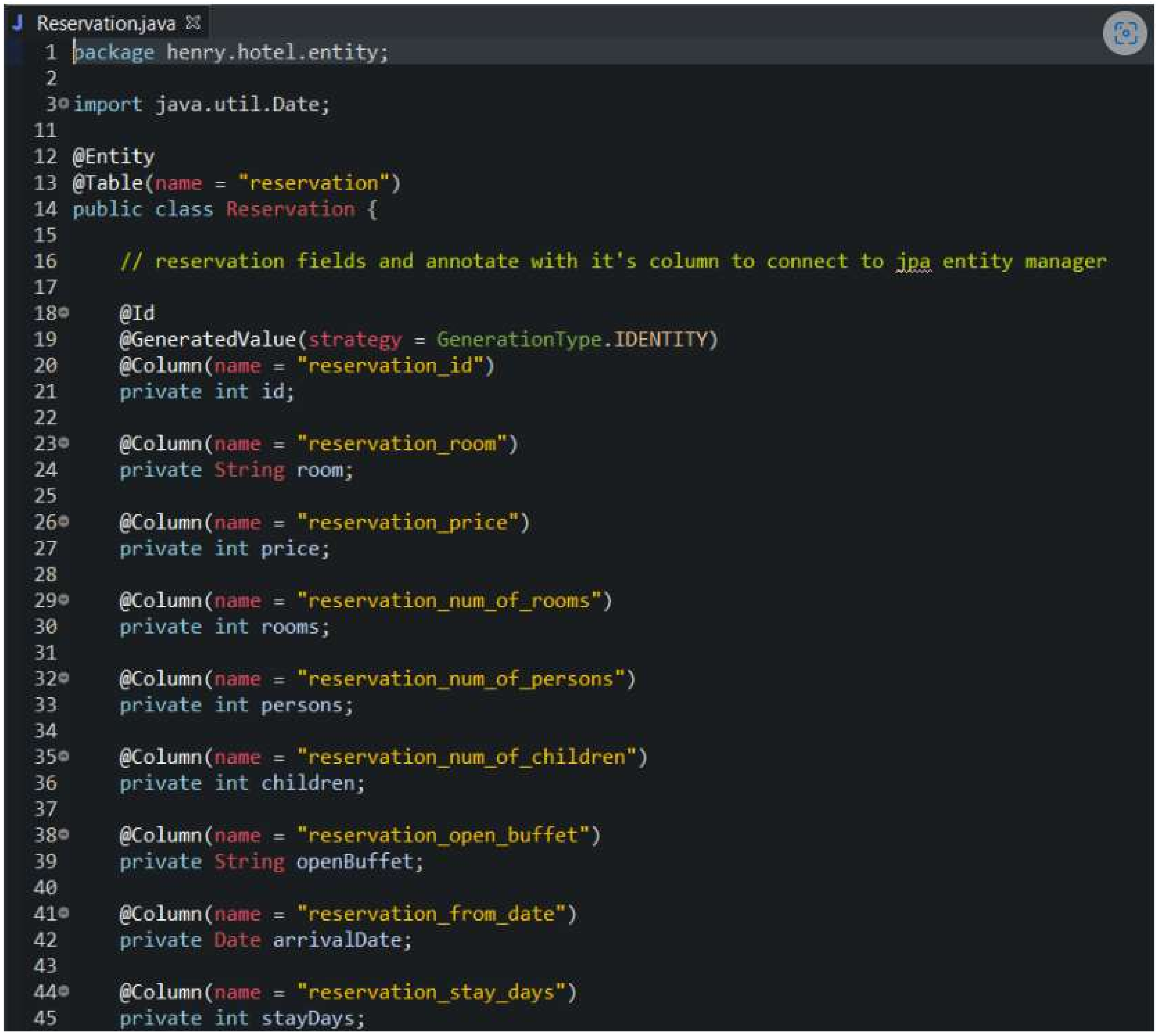
It’s a pattern used between DAO and Controller that’s modify data, manage the service inventory, manage transactional and control all database layers to handle requests between server and client. Example: Reservation Service Implementation Class.



```java
package henry.hotel.services;

import java.util.Collection;

@Service
public class ReservationServiceImpl implements ReservationService {

    // service pattern to manage transactionals
    //  and handel services for reservation between server and client

    // field injection for reservation dao
    @Autowired
    private ReservationDao reservationDao;

    // field injection for user dao
    @Autowired
    private UserService userService;

    // get reservation for logged user
    @Override
    @Transactional
    public Reservation getReservationForLoggedUserById(int resId) {

        return reservationDao.getReservationForLoggedUserById(resId);
    }

    // get all reservations for logger user
    @Override
    @Transactional
    public Collection<Reservation> getReservationsForLoggedUser() {
        return reservationDao.getReservationsByUserId(userService.getLoggedUserId());
    }

    // transfer data between temp reservation and Reservation class after check it to save it
    @Override
    @Transactional
    public void saveOrUpdateReservation(CurrentReservation currentReservation) {
        Reservation reservation = new Reservation();

```

• Entity Package:

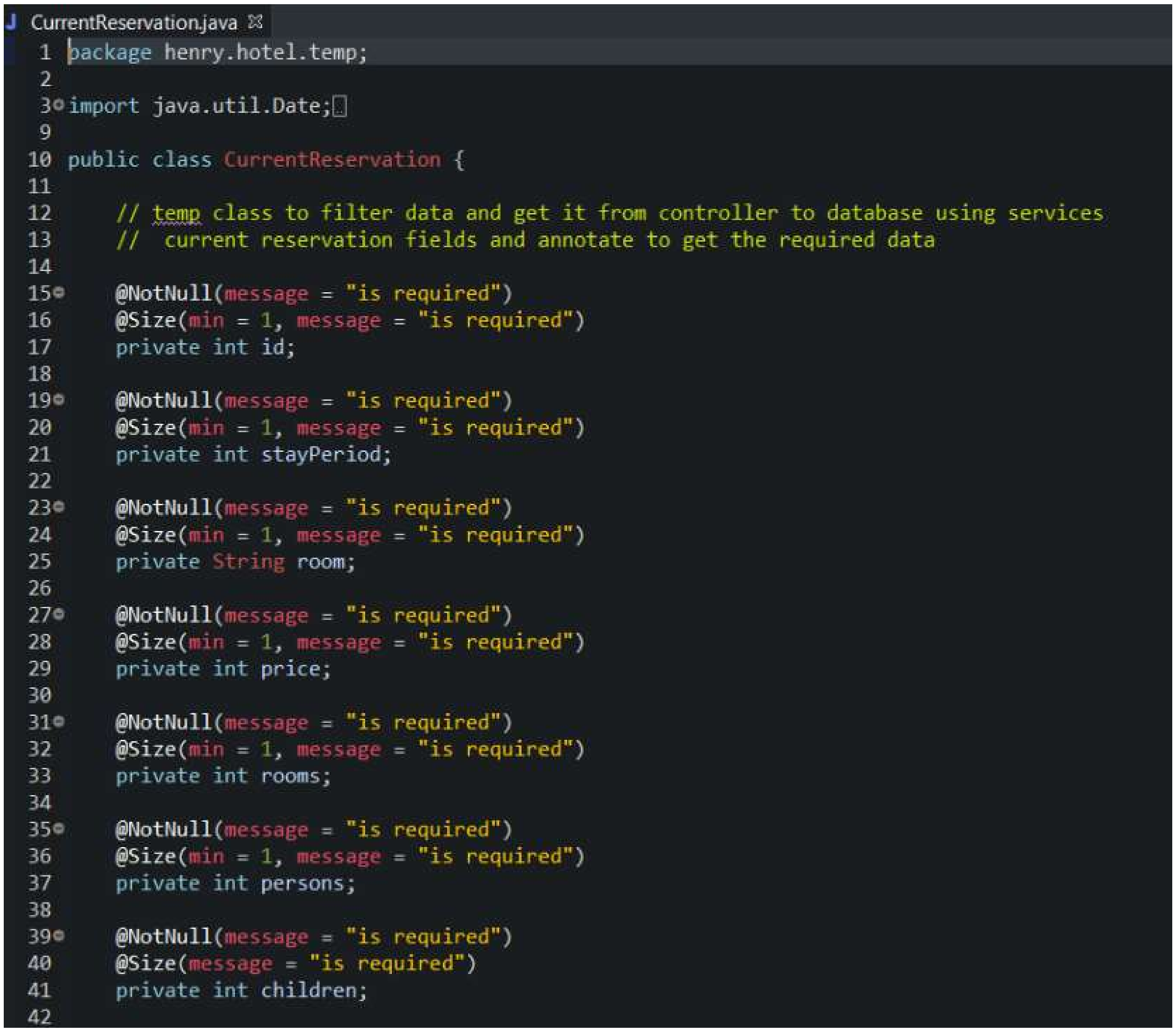
It’s the Java POJOs classes that represents a stored table in database, entity annotation scan it so can connect with entity manager (JPA) in DAO pattern. Example: Reservation (Entity Class).


Reservation.java

```java
package henry.hotel.entity;

import java.util.Date;

@Entity
@Table(name = "reservation")
public class Reservation {

    // reservation fields and annotate with it's column to connect to jpa entity manager

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "reservation_id")
    private int id;

    @Column(name = "reservation_room")
    private String room;

    @Column(name = "reservation_price")
    private int price;

    @Column(name = "reservation_num_of_rooms")
    private int rooms;

    @Column(name = "reservation_num_of_persons")
    private int persons;

    @Column(name = "reservation_num_of_children")
    private int children;

    @Column(name = "reservation_open_buffet")
    private String openBuffet;

    @Column(name = "reservation_from_date")
    private Date arrivalDate;

    @Column(name = "reservation_stay_days")
    private int stayDays;
```

Temporary Package:

This package for handle data input and match client inputs with required data, save data temporary in this class and after checking it using service pattern we pass it to entity classes. Example: Current Reservation Class.

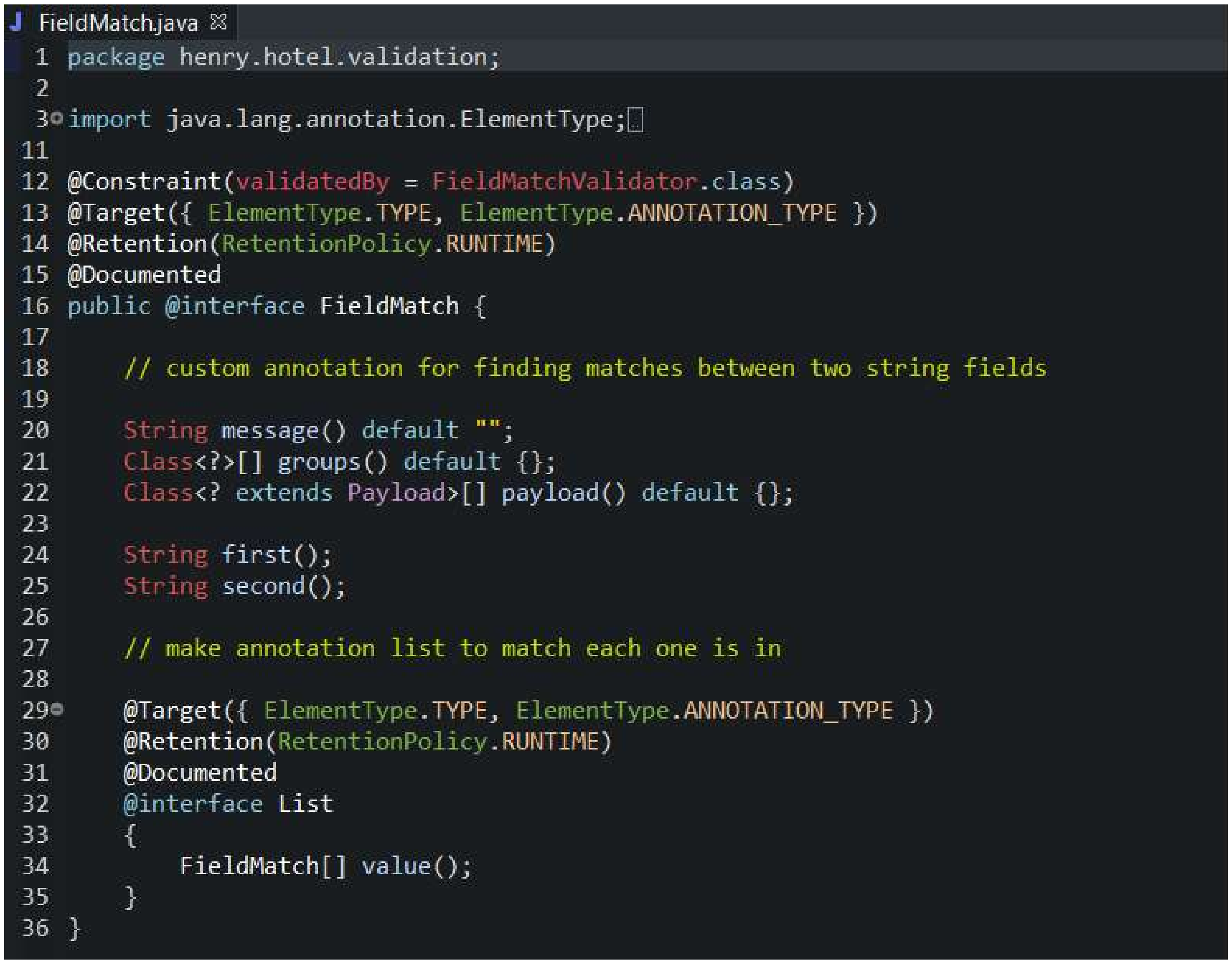


CurrentReservation.java

```java
package henry.hotel.temp;

import java.util.Date;

public class CurrentReservation {

    // temp class to filter data and get it from controller to database using services
    //  current reservation fields and annotate to get the required data

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private int id;

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private int stayPeriod;

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private String room;

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private int price;

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private int rooms;

    @NotNull(message = "is required")
    @Size(min = 1, message = "is required")
    private int persons;

    @NotNull(message = "is required")
    @Size(message = "is required")
    private int children;

```

Validation Package:

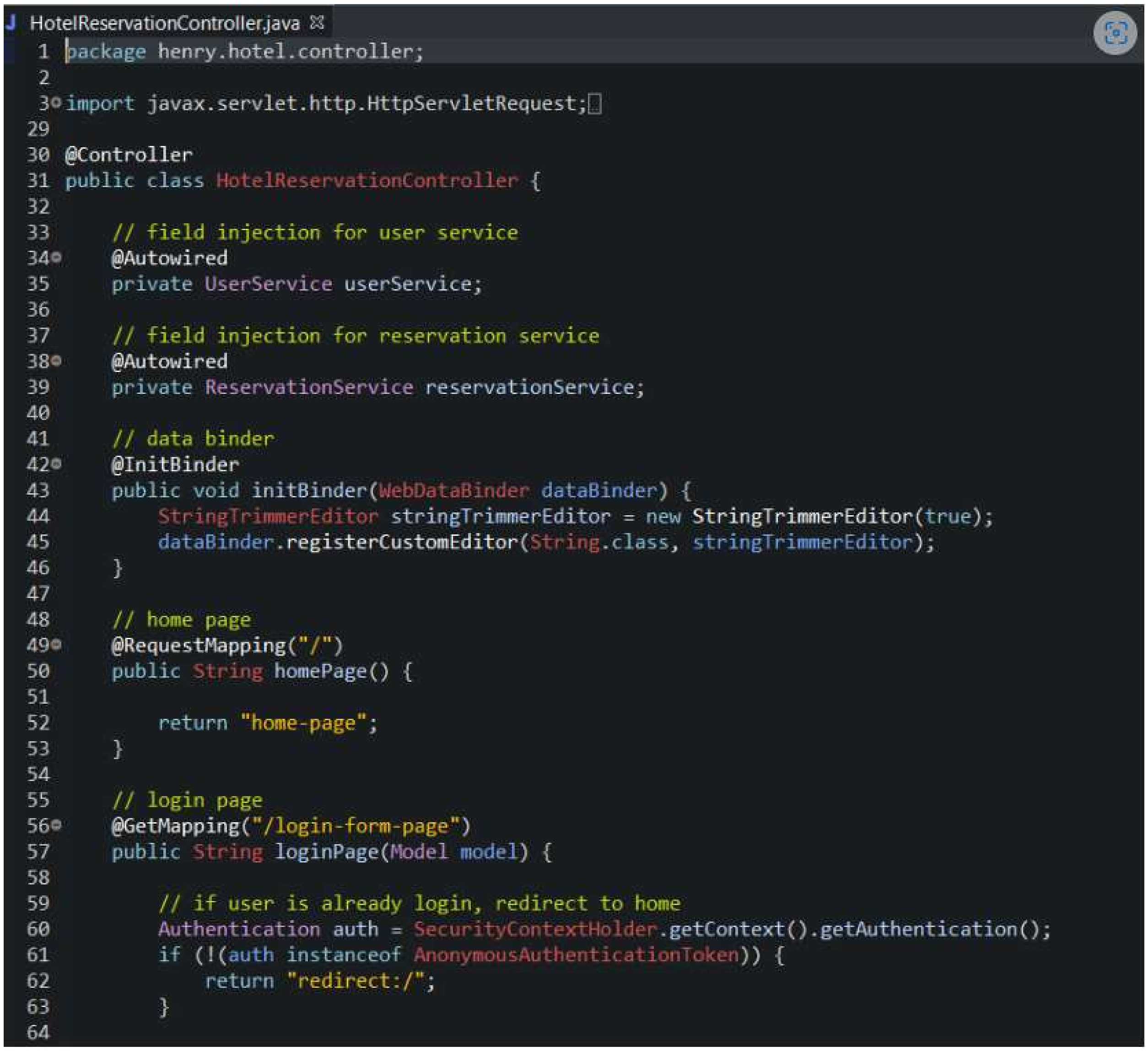
It’s some custom annotations with custom pattern regex or custom feature required to match data input with required data in Temporary Package. Example: Field Match Annotation.



```java
package henry.hotel.validation;

import java.lang.annotation.ElementType;

@Constraint(validatedBy = FieldMatchValidator.class)
@Target({ ElementType.TYPE, ElementType.ANNOTATION_TYPE })
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface FieldMatch {

    // custom annotation for finding matches between two string fields

    String message() default "";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};

    String first();
    String second();

    // make annotation list to match each one is in

    @Target({ ElementType.TYPE, ElementType.ANNOTATION_TYPE })
    @Retention(RetentionPolicy.RUNTIME)
    @Documented
    @interface List
    {
        FieldMatch[] value();
    }
}
```

Controller Package:

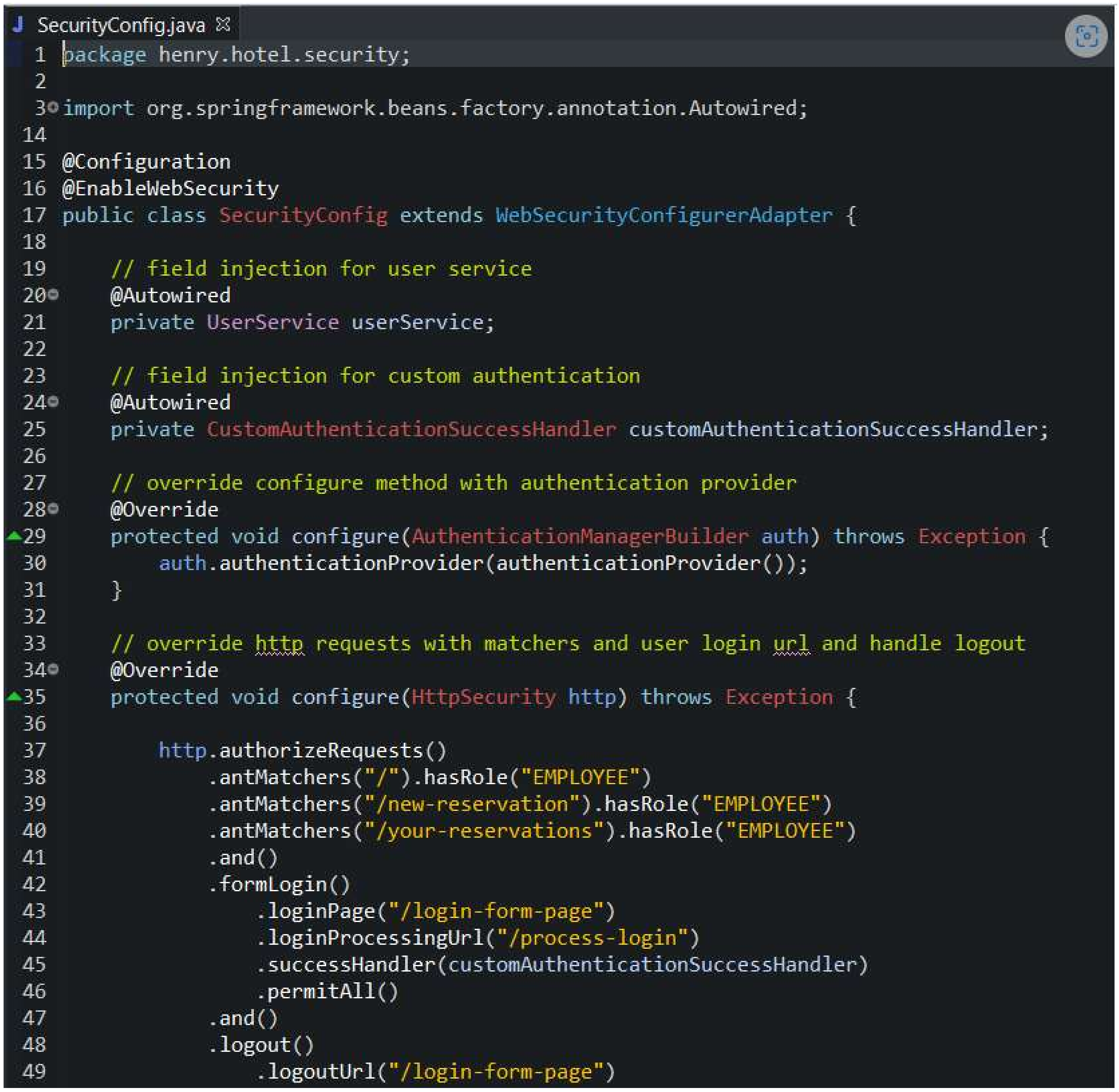
Controller is a Model View Controller (MVC) that's handle the developing of user interfaces and manage data by represent and accept data to and from client using Service pattern. Example: Hotel Reservation Controller Class.


HotelReservationController.java

```java
package henry.hotel.controller;

import javax.servlet.http.HttpServletRequest;

@Controller
public class HotelReservationController {

    // field injection for user service
    @Autowired
    private UserService userService;

    // field injection for reservation service
    @Autowired
    private ReservationService reservationService;

    // data binder
    @InitBinder
    public void initBinder(WebDataBinder dataBinder) {
        StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);
        dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
    }

    // home page
    @RequestMapping("/")
    public String homePage() {

        return "home-page";
    }

    // login page
    @GetMapping("/login-form-page")
    public String loginPage(Model model) {

        // if user is already login, redirect to home
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (!(auth instanceof AnonymousAuthenticationToken)) {
            return "redirect:/";
        }

```

Security Package:

It’s a Spring Security class that handle the authentications and authorizations of the project and manage client and his role. Manage login and logout operations and all security features. Example: Security Config Class.



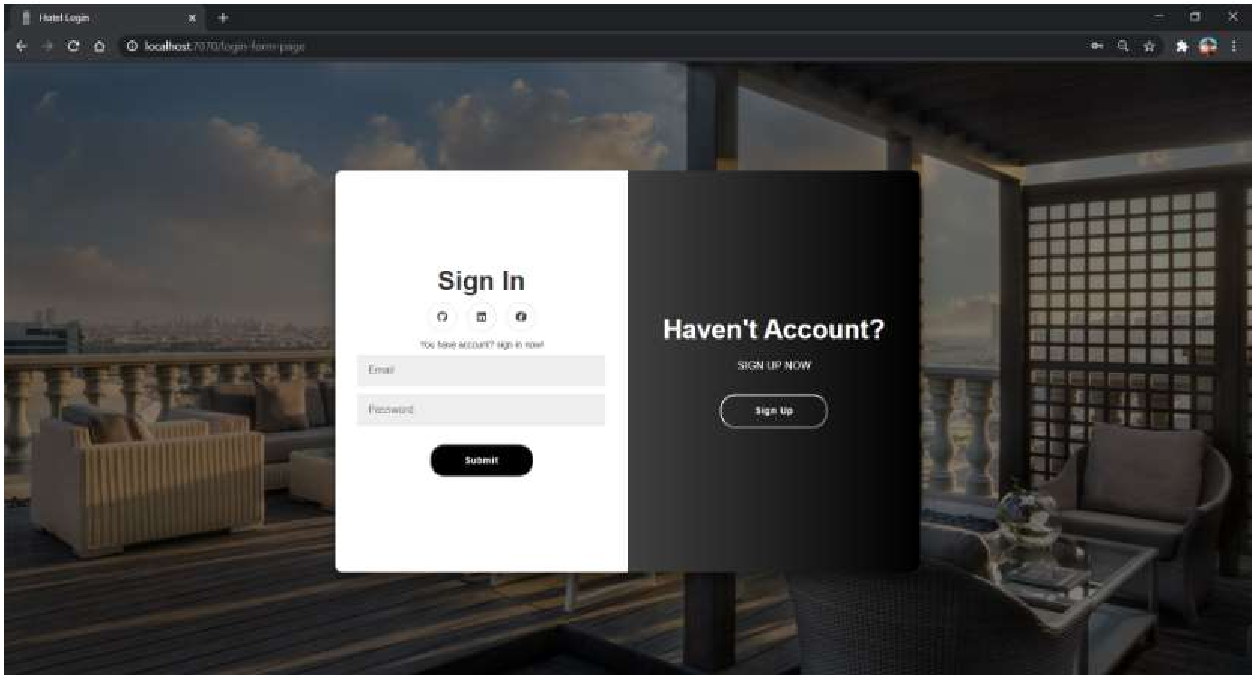
SecurityConfig.java

```java
package henry.hotel.security;

import org.springframework.beans.factory.annotation.Autowired;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    // field injection for user service
    @Autowired
    private UserService userService;

    // field injection for custom authentication
    @Autowired
    private CustomAuthenticationSuccessHandler customAuthenticationSuccessHandler;

    // override configure method with authentication provider
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(authenticationProvider());
    }

    // override http requests with matchers and user login url and handle logout
    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
            .antMatchers("/").hasRole("EMPLOYEE")
            .antMatchers("/new-reservation").hasRole("EMPLOYEE")
            .antMatchers("/your-reservations").hasRole("EMPLOYEE")
            .and()
            .formLogin()
                .loginPage("/login-form-page")
                .loginProcessingUrl("/process-login")
                .successHandler(customAuthenticationSuccessHandler)
                .permitAll()
            .and()
            .logout()
                .logoutUrl("/login-form-page")
```

Project Interface Test

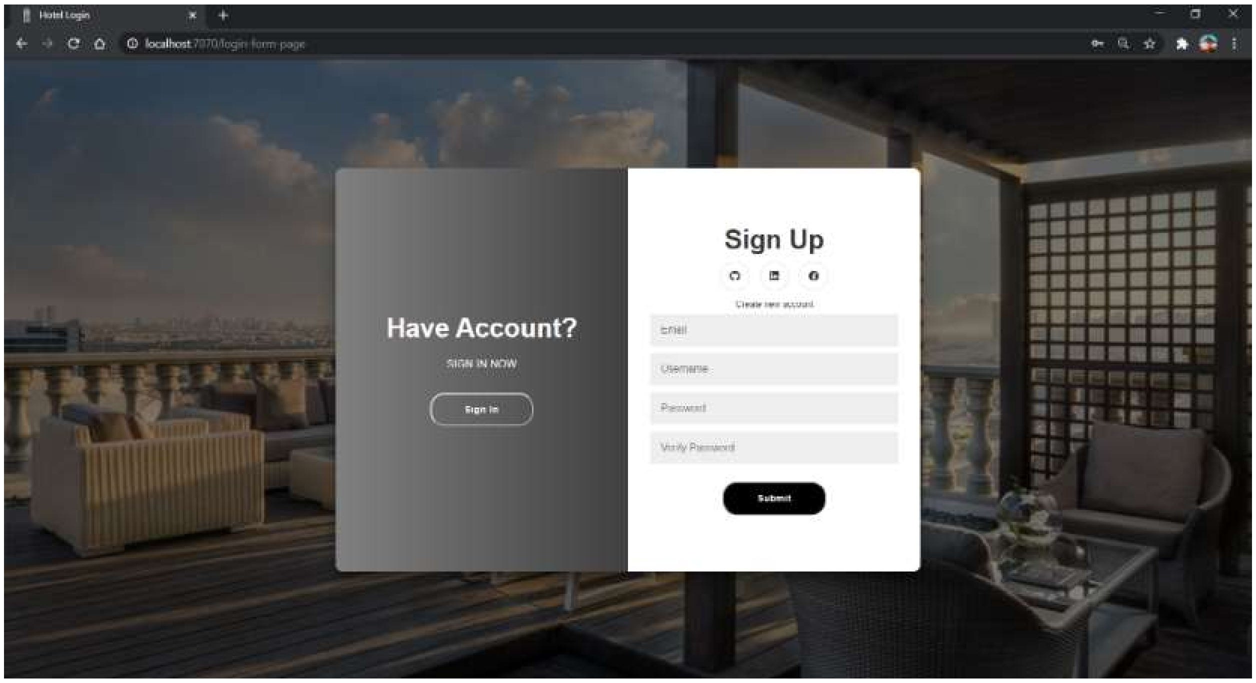
• Sign In:

You can’t open any another page without sign in, you will directly transfer automatic to this page.



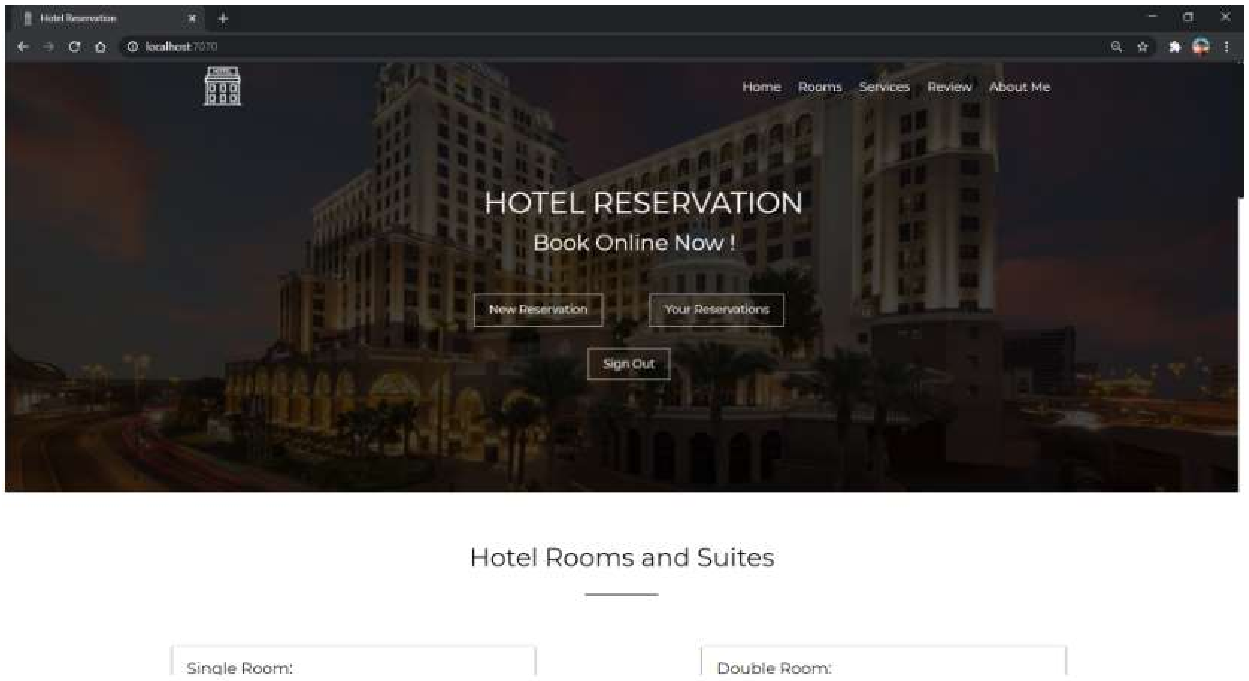
• Sign Up

Here you can Sign Up new account of your own and will be saved next time you can just sign in directly.



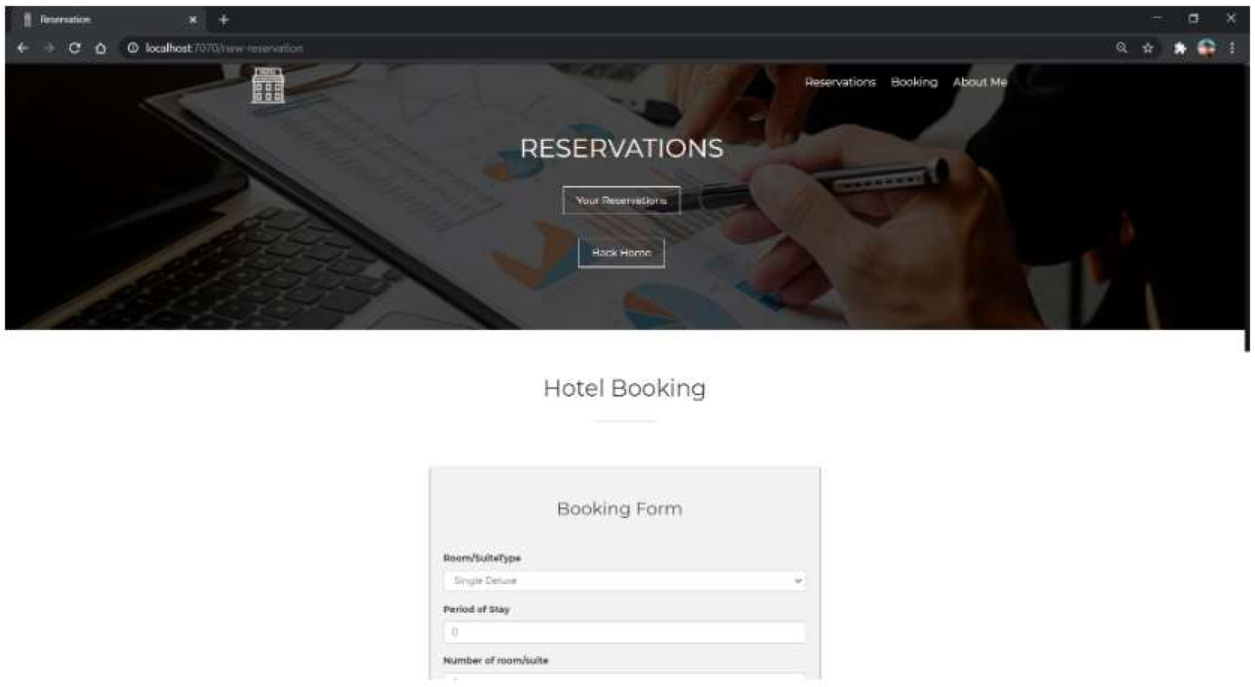
• Home:

There are many sections about the hotel to view it and information about it, Rooms, Services, Review. Etc It’s a full page with header and content and footer. You can use features like, Log Out, New Reservation or view logged in user reservations.



• New Reservation:

Reservations page, here you can book a room.



Hotel Booking

Booking Form

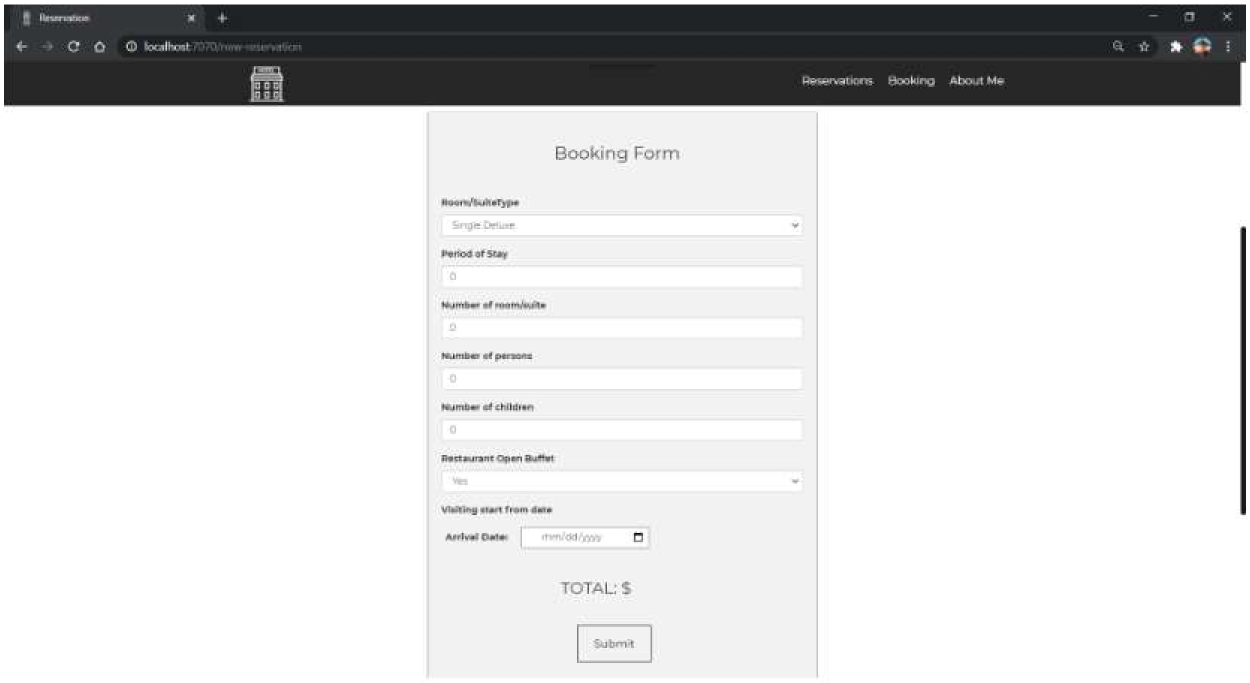
Room/SuiteType
Single Deluxe

Period of Stay
0

Number of room/suite

• Booking Form:

Here is a form that you can choose your required choices, with live price counter.



Booking Form

Room/SuiteType

Single Deluxe

Period of Stay

0

Number of room/suite

0

Number of persons

0

Number of children

0

Restaurant Open Buffet

Yes

Visiting start from date

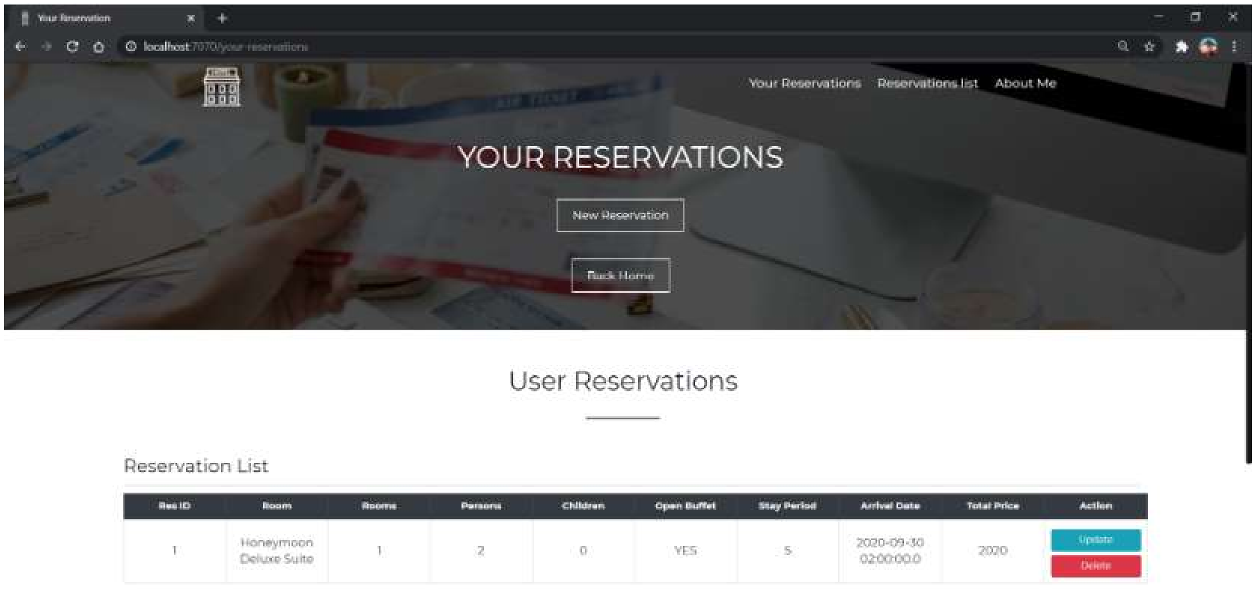
Arrival Date: mm/dd/yyyy

TOTAL: $

Submit

• Reservations List:

In this page you can modify your reservation and see list of user logged in reservations and can delete it or can update it with another choices.



User Reservations

Reservation List

Res ID	Room	Rooms	Persons	Children	Open Buffet	Stay Period	Arrival Date	Total Price	Action
1	Honeymoon Deluxe Suite	1	2	0	YES	5	2020-09-30 02:00:00.0	2020	Update Delete

Conclusion

This was the documentation of the Hotel Room Booking Application, What I'm learn in programming language two by resource from stack overflow was used and also some of the extra curriculum was used such as Spring Framework and Front-End code with as these were essential components to build full project. This project included many properties and capabilities where you can create account, sign in and the website keep you in cycle to sign out in the end. Then you can see information and pictures about hotel and rooms and see the reviews, services and packages. After that you can create new reservation by filling the book form. Then you can modify it in your reservations page that provide a list of your reservations that you can update or delete. This project will help the real life to book a far hotels without going to it so you can easily book from the internet so that make the operation faster and easier.