IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name:
Appukutti Munasinghe

Student Reference Number:

Module Code: **PUSL2024**	Module Name: **Software Engineering 2**

Coursework Title: **Hotel Management System Coursework**

Deadline Date: **13/01/2022**	Member of staff responsible for coursework: **Mr. Kaneeka Vidanage**

Programme:
Bsc.(Hons) Computer Security

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

Edirisinghe Edirisinghe	
Galpola Galpola	
Appukutti Munasinghe	
Warnakulasooriya Peiris	
Hettiarachchige Hettiarachchi	

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed : -

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

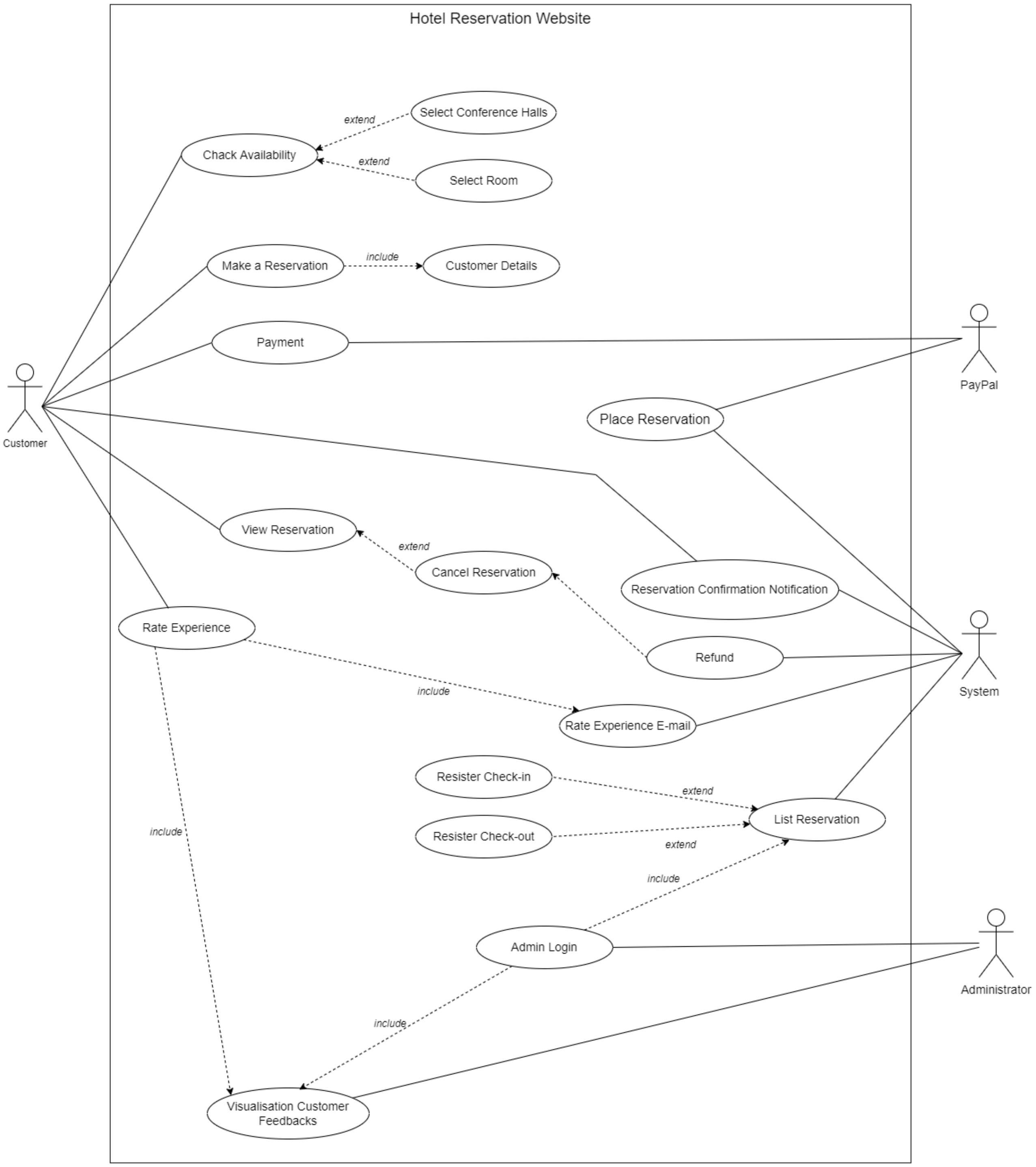
I *have used/not used translation software.

If used, please state name of software……………………………………………………………

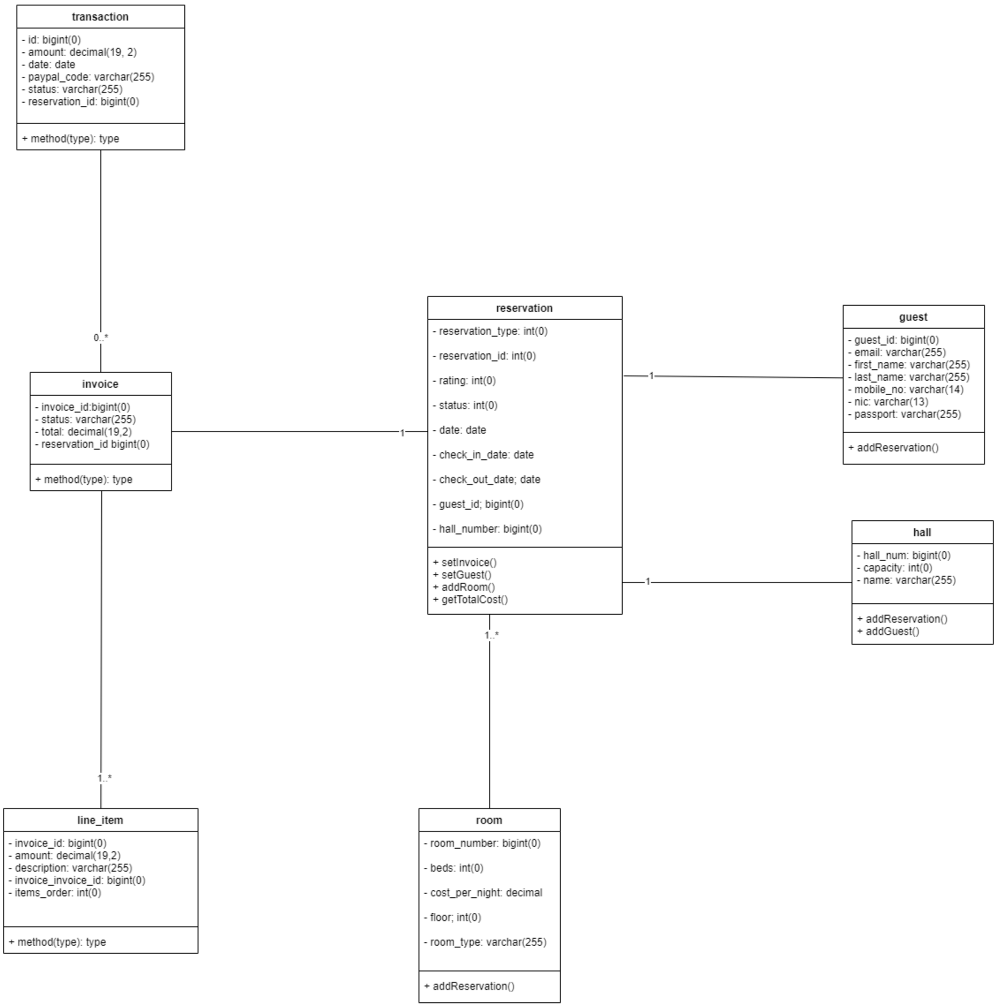
Overall mark _____% Assessors Initials ______ Date_________

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

1. The Use Case Diagram

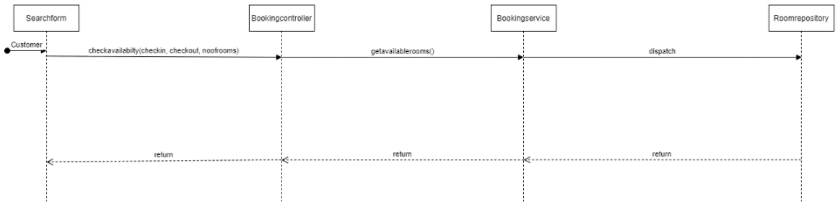


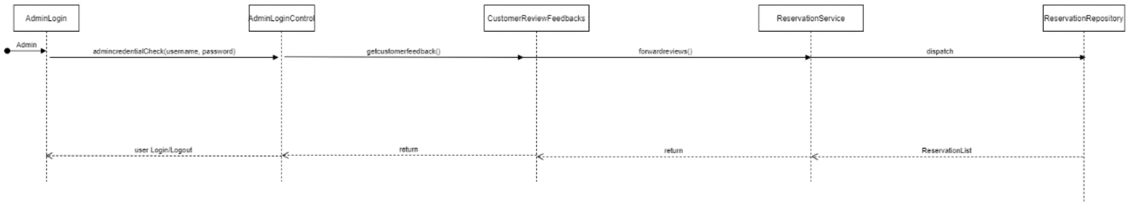
2. The Class Diagram

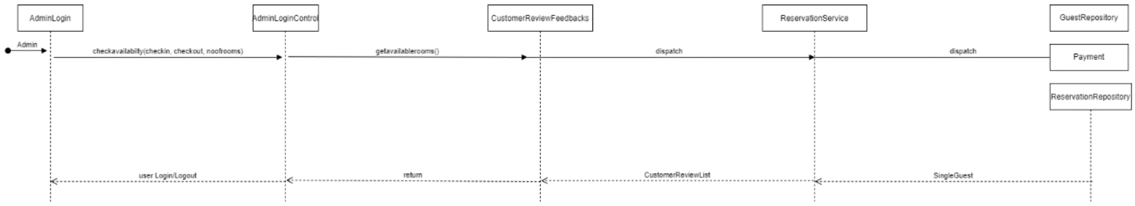


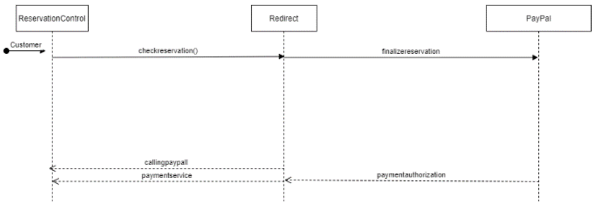
Some methods are on entities are by directional due to JPA standard requirements. Every private variable will have public get and set methods.

3. The Sequence Diagram

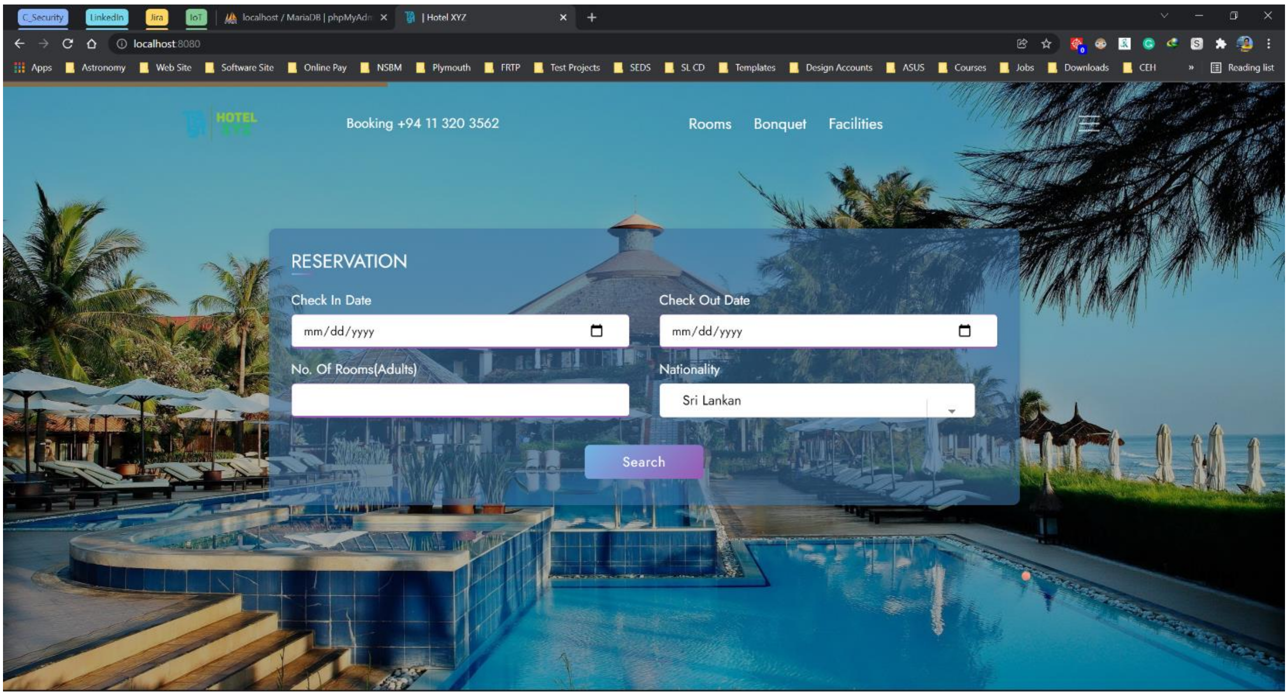








4. Code



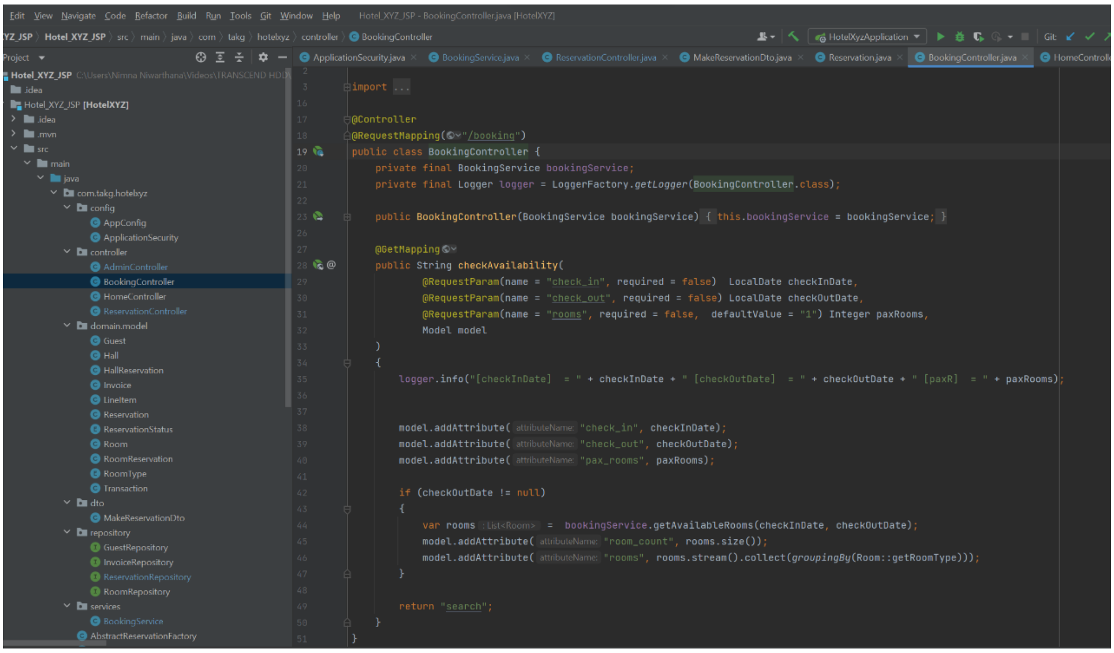
RESERVATION

Check In Date: mm/dd/yyyy

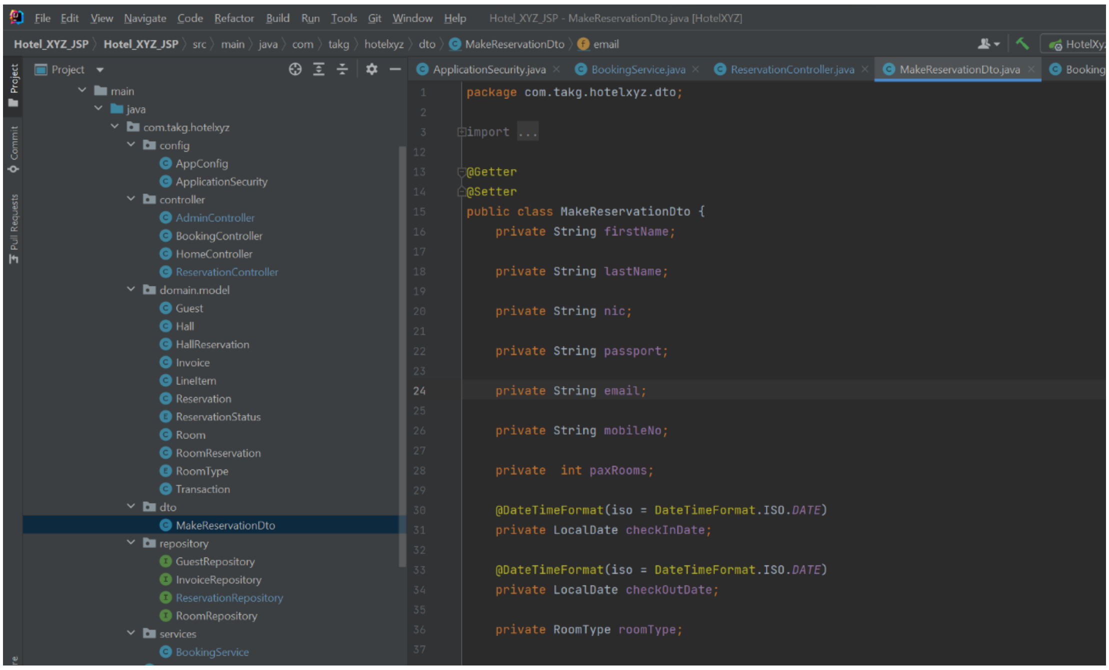
Check Out Date: mm/dd/yyyy

No. Of Rooms(Adults):

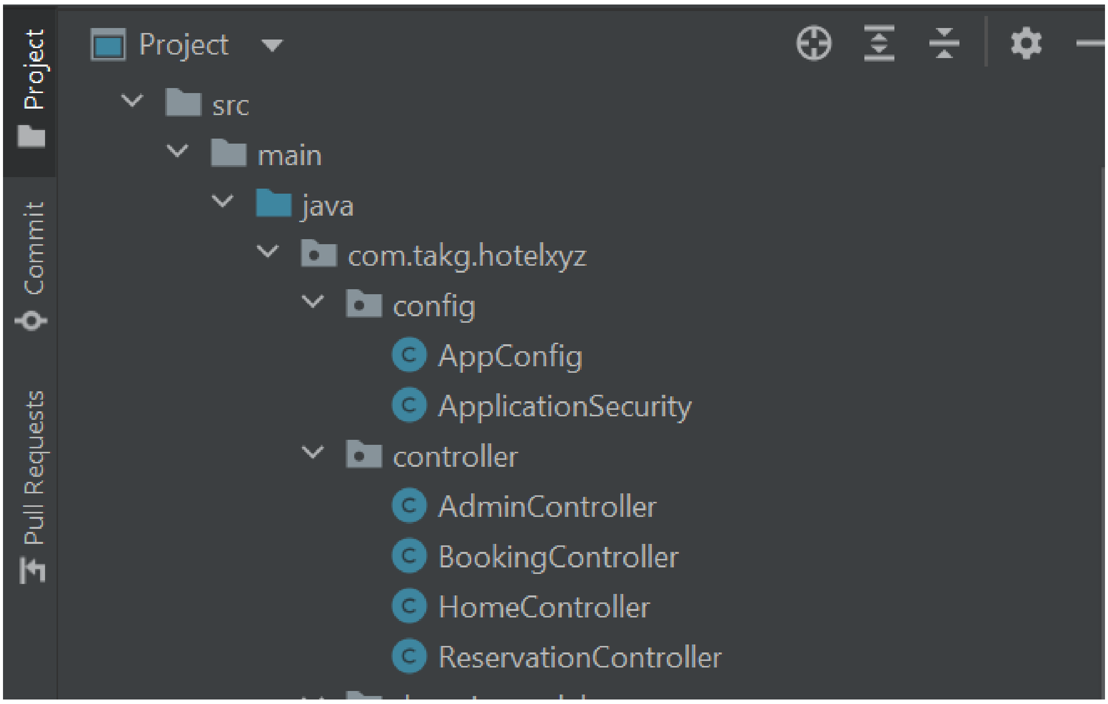
Nationality: Sri Lankan

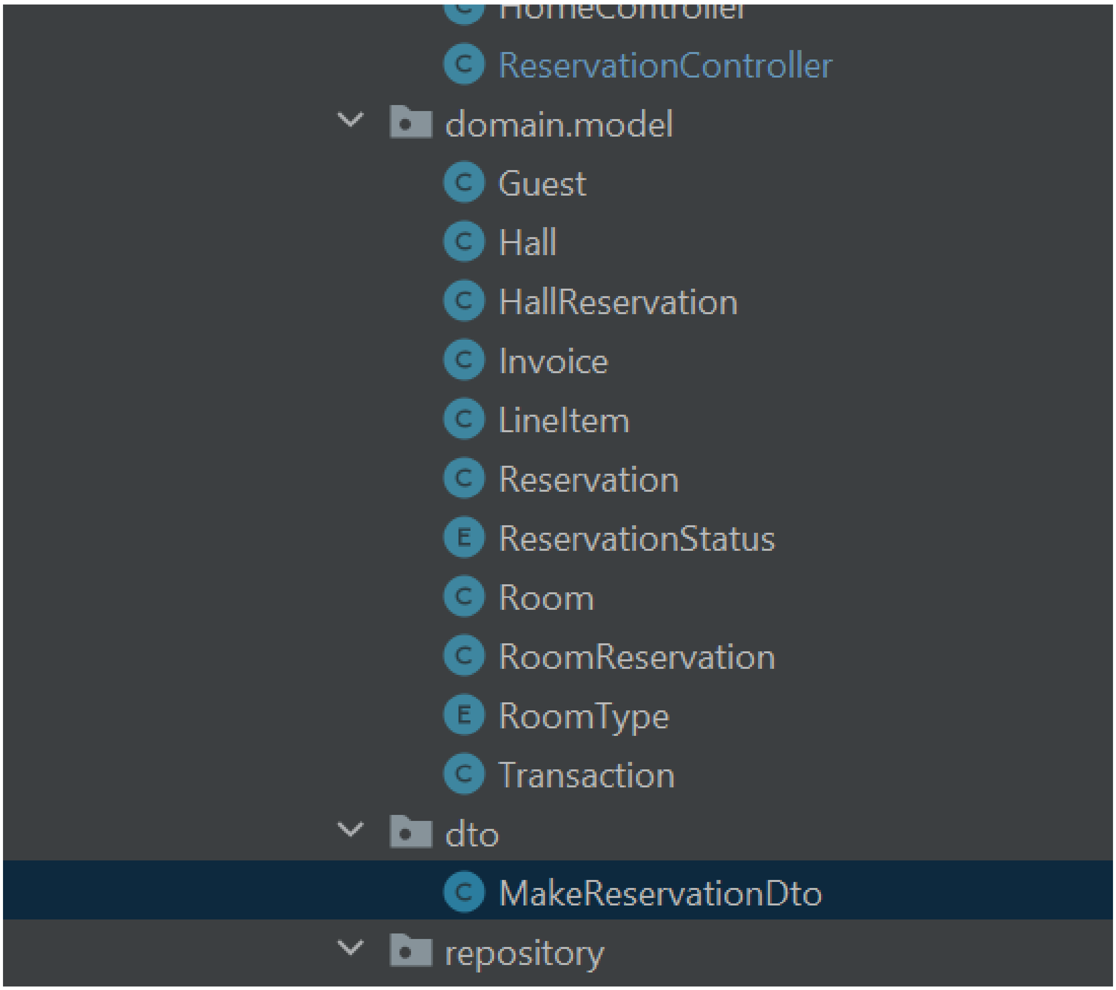


```java

import ...

@Controller
@RequestMapping("/booking")
public class BookingController {
    private final BookingService bookingService;
    private final Logger logger = LoggerFactory.getLogger(BookingController.class);

    public BookingController(BookingService bookingService) { this.bookingService = bookingService; }

    @GetMapping
    public String checkAvailability(
            @RequestParam(name = "check_in", required = false)  LocalDate checkInDate,
            @RequestParam(name = "check_out", required = false) LocalDate checkOutDate,
            @RequestParam(name = "rooms", required = false,  defaultValue = "1") Integer paxRooms,
            Model model
    )
    {
        logger.info("[checkInDate]  = " + checkInDate + " [checkOutDate]  = " + checkOutDate + " [paxR]  = " + paxRooms);


        model.addAttribute( attributeName: "check_in", checkInDate);
        model.addAttribute( attributeName: "check_out", checkOutDate);
        model.addAttribute( attributeName: "pax_rooms", paxRooms);

        if (checkOutDate != null)
        {
            var rooms : List<Room> =  bookingService.getAvailableRooms(checkInDate, checkOutDate);
            model.addAttribute( attributeName: "room_count", rooms.size());
            model.addAttribute( attributeName: "rooms", rooms.stream().collect(groupingBy(Room::getRoomType)));
        }

        return "search";
    }
}
```

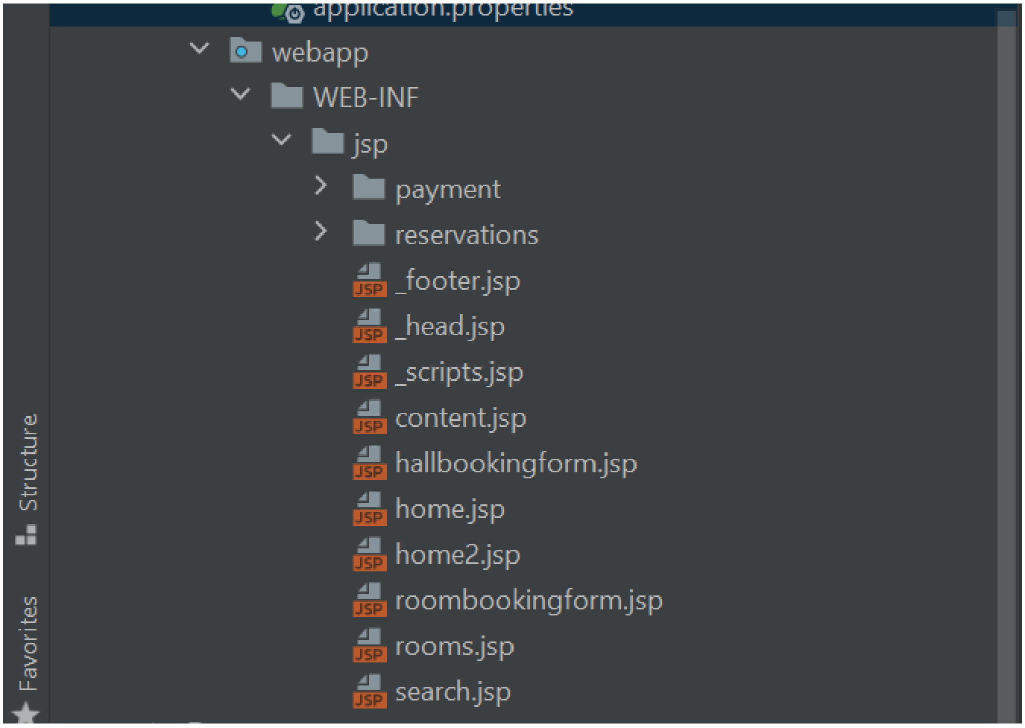


```java
package com.takg.hotelxyz.dto;

import ...

@Getter
@Setter
public class MakeReservationDto {
    private String firstName;

    private String lastName;

    private String nic;

    private String passport;

    private String email;

    private String mobileNo;

    private  int paxRooms;

    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private LocalDate checkInDate;

    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
    private LocalDate checkOutDate;

    private RoomType roomType;

```

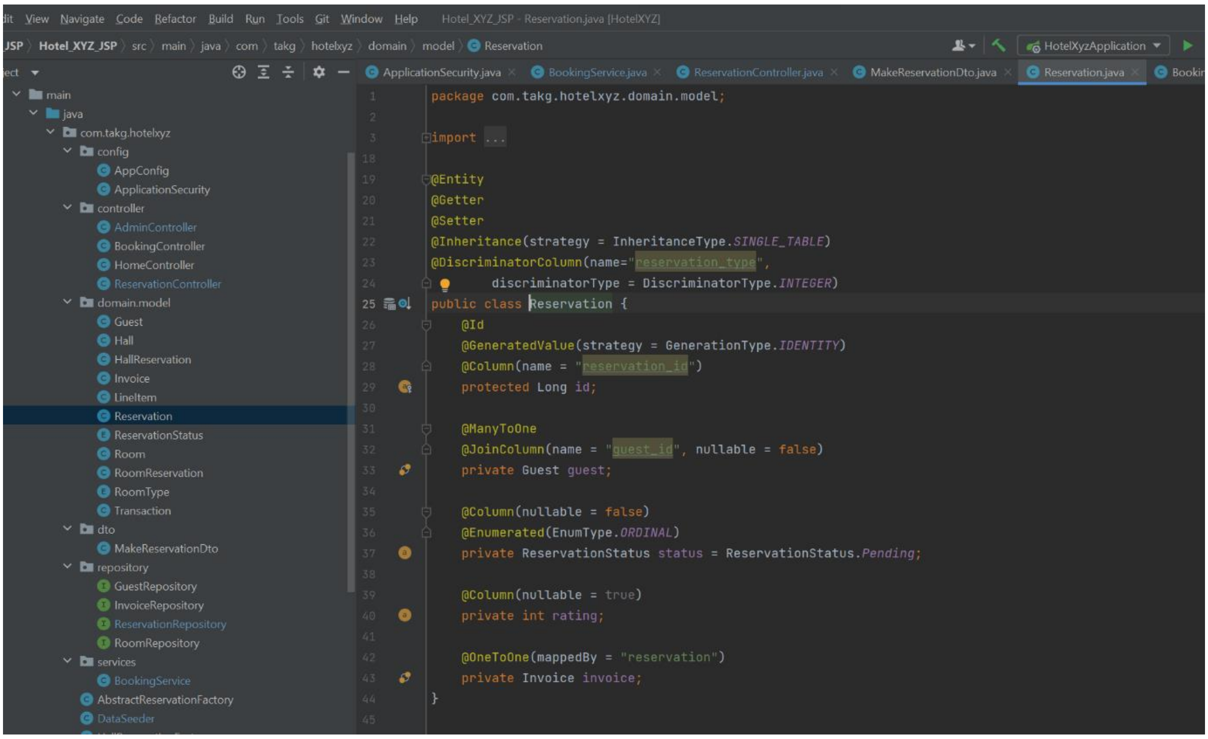




```
        HomeController
        ReservationController
    domain.model
        Guest
        Hall
        HallReservation
        Invoice
        LineItem
        Reservation
        ReservationStatus
        Room
        RoomReservation
        RoomType
        Transaction
    dto
        MakeReservationDto
    repository
```



```
        application.properties
  webapp
    WEB-INF
      jsp
        payment
        reservations
        _footer.jsp
        _head.jsp
        _scripts.jsp
        content.jsp
        hallbookingform.jsp
        home.jsp
        home2.jsp
        roombookingform.jsp
        rooms.jsp
        search.jsp
```



```java
package com.takg.hotelxyz.domain.model;

import ...

@Entity
@Getter
@Setter
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="reservation_type",
        discriminatorType = DiscriminatorType.INTEGER)
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "reservation_id")
    protected Long id;

    @ManyToOne
    @JoinColumn(name = "guest_id", nullable = false)
    private Guest guest;

    @Column(nullable = false)
    @Enumerated(EnumType.ORDINAL)
    private ReservationStatus status = ReservationStatus.Pending;

    @Column(nullable = true)
    private int rating;

    @OneToOne(mappedBy = "reservation")
    private Invoice invoice;
}
```

5. Design Pattern

We used three types of design patterns within this system, Singleton, Prototype and Factory.

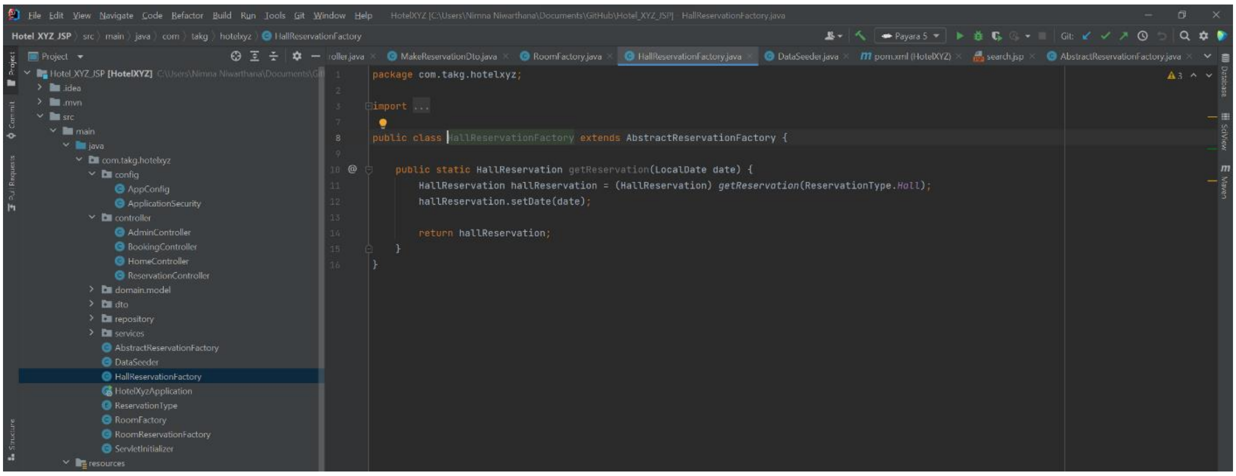
The singleton connection is created in one instance but implemented in several locations.

Prototypes – All the service and repository classes follow the prototype design patterns. These classes are created and initialized by spring framework CDI.

Singleton – A JDBC connection is very expensive to create repeatedly, therefore, we maintained a single instance that will be shared among the multiple connections.

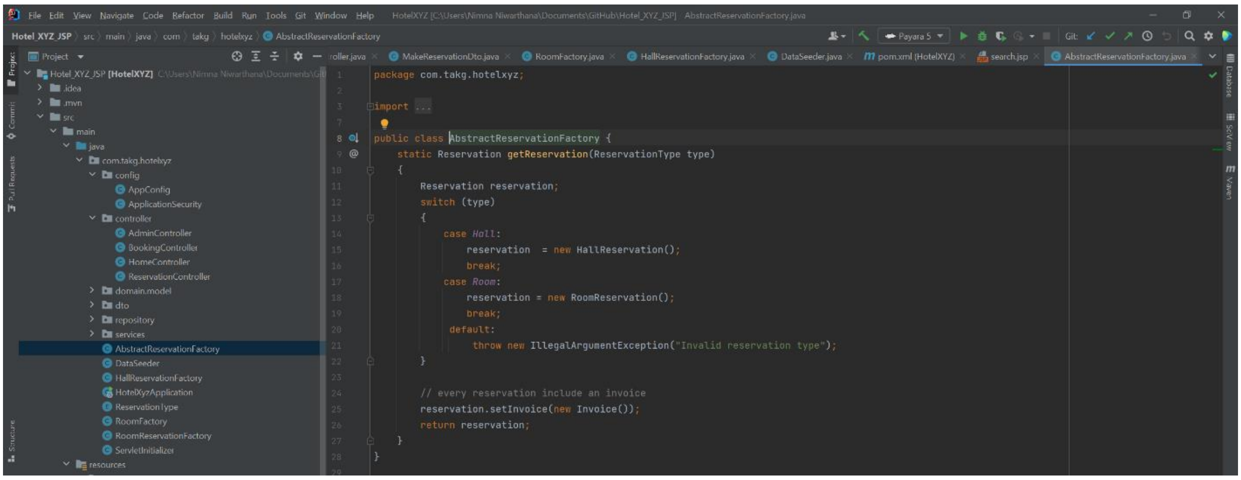
Factory method design pattern can be defined as a class used for creating and object. However, factory design pattern lets the subclass select the class to instantiate. Factory design pattern assigns the responsibility of initializing a class from the client to the virtual constructor.

HallReservationFactory Class



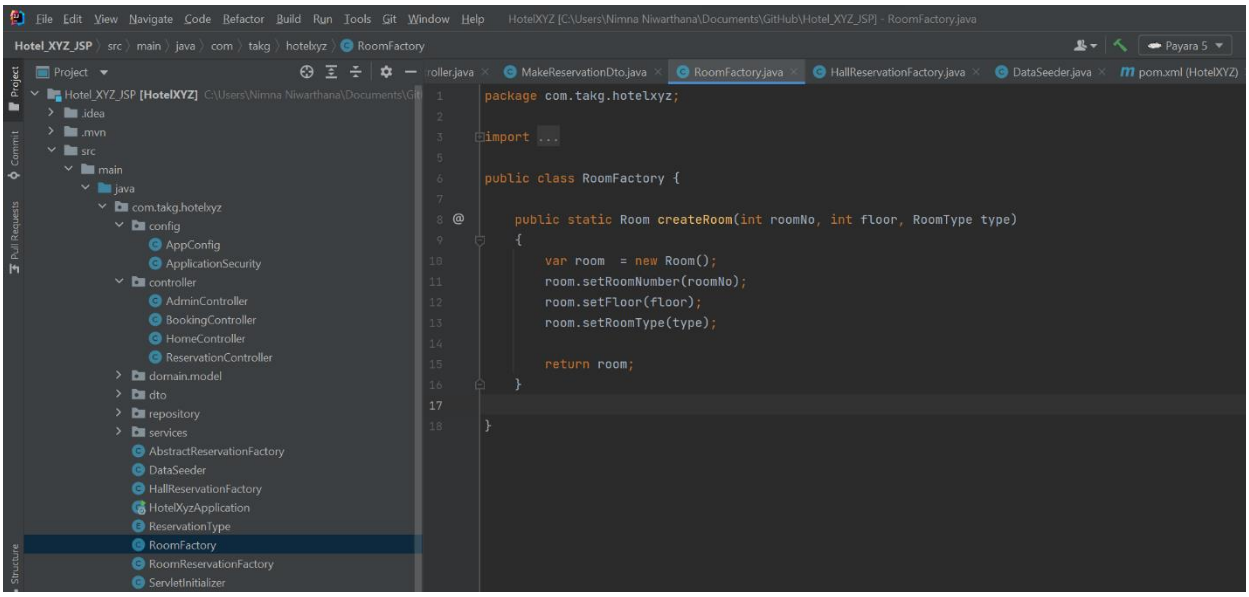
```java
package com.takg.hotelxyz;

import ...

public class HallReservationFactory extends AbstractReservationFactory {

    public static HallReservation getReservation(LocalDate date) {
        HallReservation hallReservation = (HallReservation) getReservation(ReservationType.Hall);
        hallReservation.setDate(date);

        return hallReservation;
    }
}
```

AbstractReservationFactory Class



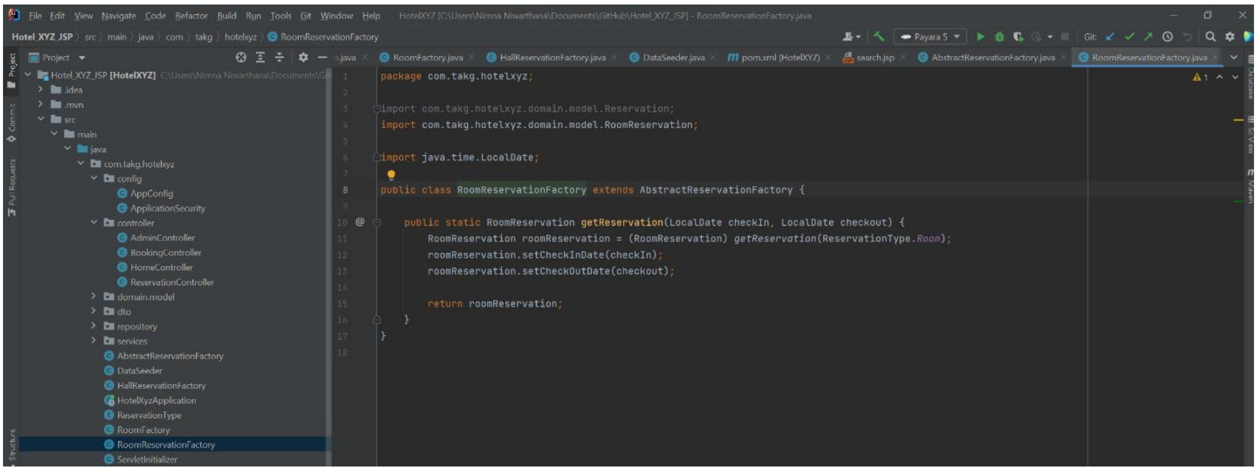
```java
package com.takg.hotelxyz;

import ...

public class AbstractReservationFactory {
    static Reservation getReservation(ReservationType type)
    {
        Reservation reservation;
        switch (type)
        {
            case Hall:
                reservation  = new HallReservation();
                break;
            case Room:
                reservation = new RoomReservation();
                break;
             default:
                 throw new IllegalArgumentException("Invalid reservation type");
        }

        // every reservation include an invoice
        reservation.setInvoice(new Invoice());
        return reservation;
    }
}
```

RoomFactory Class



```java
package com.takg.hotelxyz;

import ...

public class RoomFactory {

    public static Room createRoom(int roomNo, int floor, RoomType type)
    {
        var room  = new Room();
        room.setRoomNumber(roomNo);
        room.setFloor(floor);
        room.setRoomType(type);

        return room;
    }

}
```

RoomReservationFactory Class



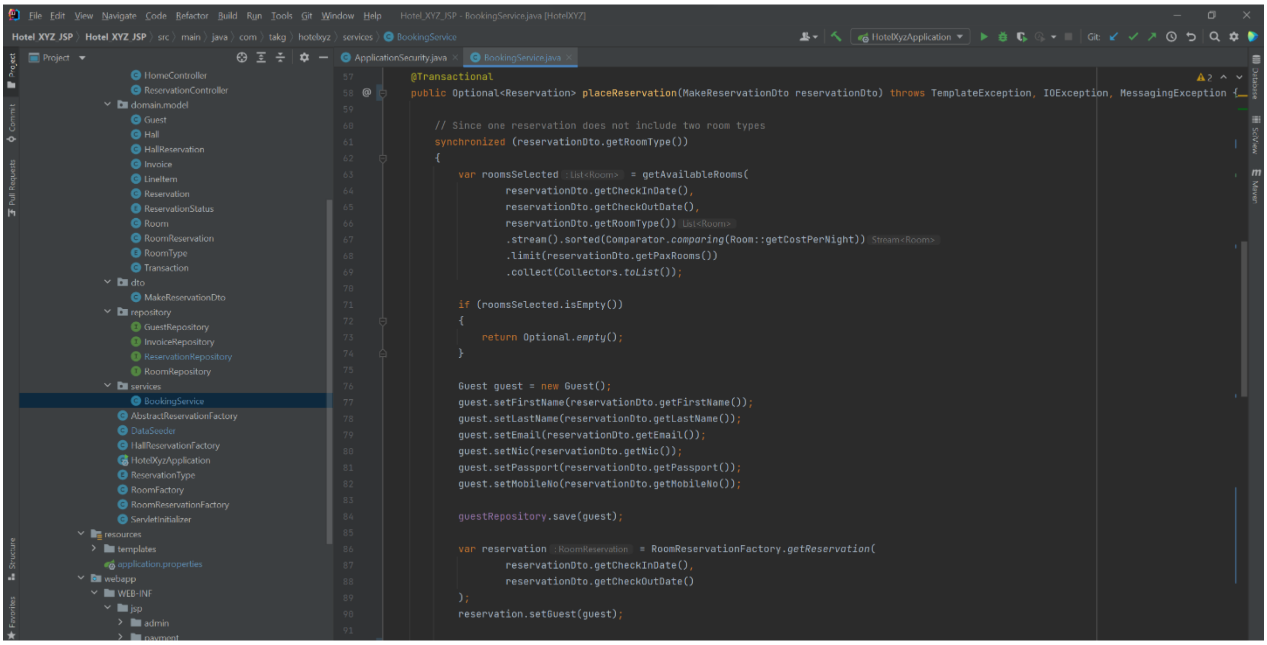
```java
package com.takg.hotelxyz;

import com.takg.hotelxyz.domain.model.Reservation;
import com.takg.hotelxyz.domain.model.RoomReservation;

import java.time.LocalDate;

public class RoomReservationFactory extends AbstractReservationFactory {

    public static RoomReservation getReservation(LocalDate checkIn, LocalDate checkout) {
        RoomReservation roomReservation = (RoomReservation) getReservation(ReservationType.Room);
        roomReservation.setCheckInDate(checkIn);
        roomReservation.setCheckOutDate(checkout);

        return roomReservation;
    }
}
```

6. Evidence and clarification of the used design patterns.

The singleton connection is created in one instance but implemented in several locations. This allows the code to be simple and efficient rather than having the singleton implemented at every location. Prototyping is implemented on each connection using the spring Framework CDI, to create objects, services, and repositories. The reason for this decision is that each user can get involved in a single transaction and can be removed afterwards to create better transactions and improved security. Evidence for using the design patterns can be seen in the figures posted for the above question.

7. Synchronized thread safe functionality within room reservation

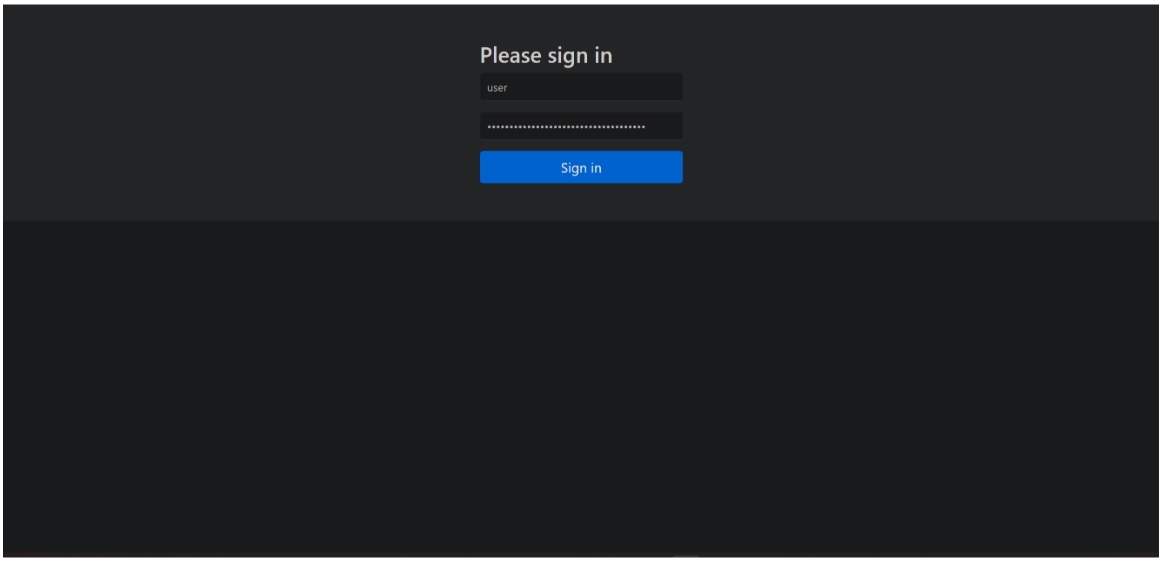
Package/services/BookingService



```java
57  @Transactional
58  public Optional<Reservation> placeReservation(MakeReservationDto reservationDto) throws TemplateException, IOException, MessagingException {
59
60      // Since one reservation does not include two room types
61      synchronized (reservationDto.getRoomType())
62      {
63          var roomsSelected = getAvailableRooms(
64                  reservationDto.getCheckInDate(),
65                  reservationDto.getCheckOutDate(),
66                  reservationDto.getRoomType())
67                  .stream().sorted(Comparator.comparing(Room::getCostPerNight))
68                  .limit(reservationDto.getPaxRooms())
69                  .collect(Collectors.toList());
70
71          if (roomsSelected.isEmpty())
72          {
73              return Optional.empty();
74          }
75
76          Guest guest = new Guest();
77          guest.setFirstName(reservationDto.getFirstName());
78          guest.setLastName(reservationDto.getLastName());
79          guest.setEmail(reservationDto.getEmail());
80          guest.setNic(reservationDto.getNic());
81          guest.setPassport(reservationDto.getPassport());
82          guest.setMobileNo(reservationDto.getMobileNo());
83
84          guestRepository.save(guest);
85
86          var reservation = RoomReservationFactory.getReservation(
87                  reservationDto.getCheckInDate(),
88                  reservationDto.getCheckOutDate()
89          );
90          reservation.setGuest(guest);
91
```

When a booking happens, we have assumed that there will not be any collision between rooms of separate types (Deluxe bookings will not affect the Suite and Premium bookings). But if there arises an instance where two rooms of the same type are being booked, the first user to call the function (*Line 50 of above figure*) will lock it from the latter. This means that the second user will have to wait until the former has finished the booking, to book their own room.

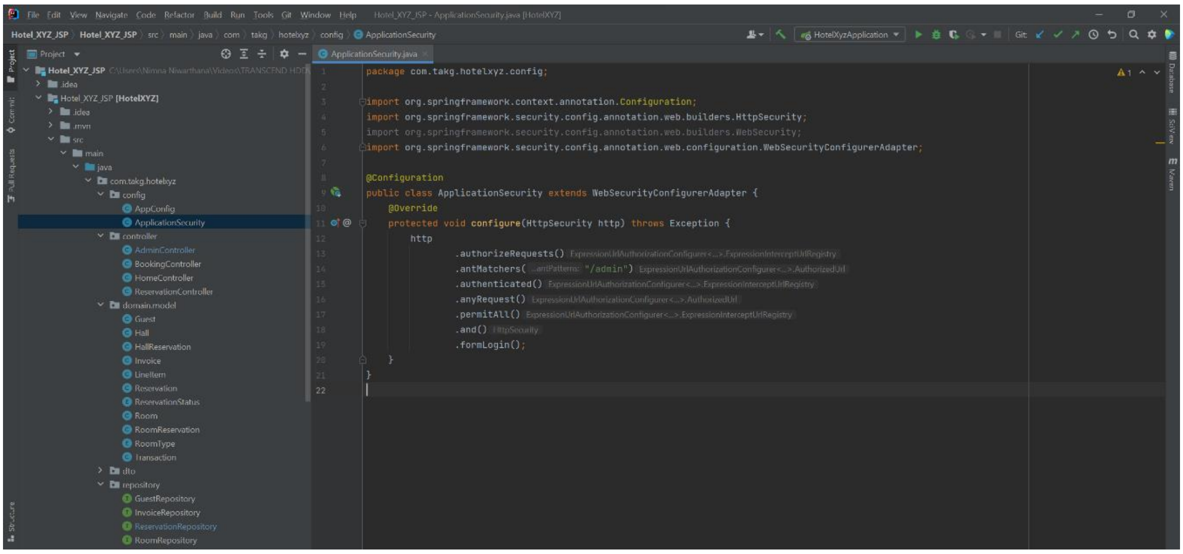
8. The Figures posted below display the view for the admin after they have logged in using proper credentials.



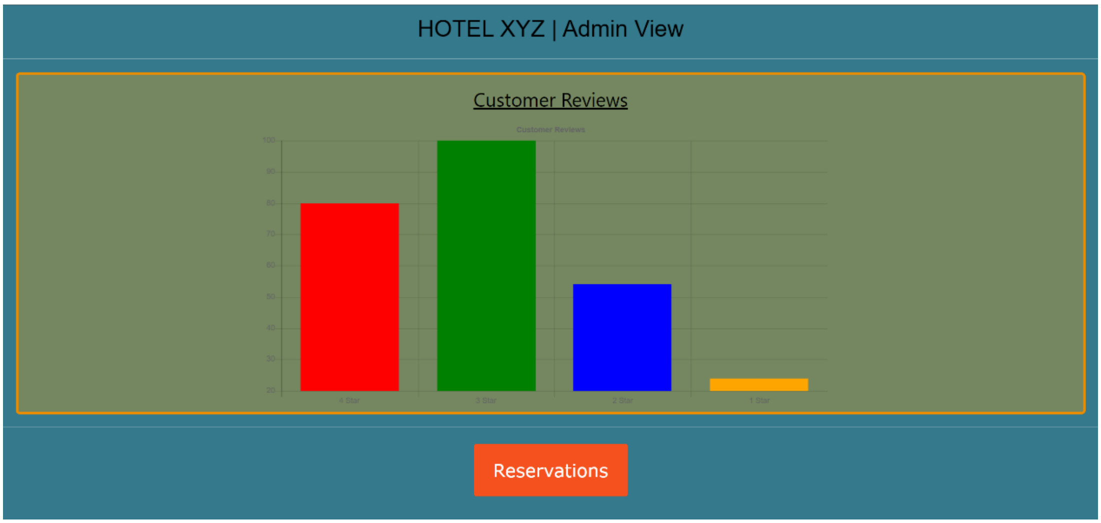
Please sign in

user

••

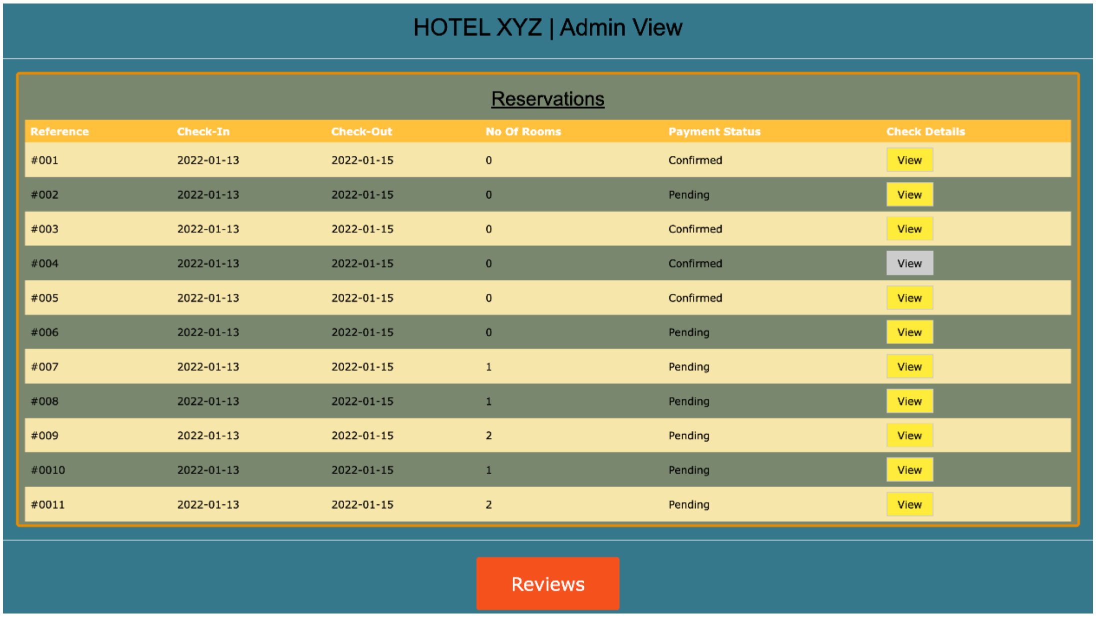


```java
package com.takg.hotelxyz.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
public class ApplicationSecurity extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .antMatchers( ...antPatterns: "/admin") ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl
                .authenticated() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .anyRequest() ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl
                .permitAll() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .and() HttpSecurity
                .formLogin();
    }
}
```



HOTEL XYZ | Admin View

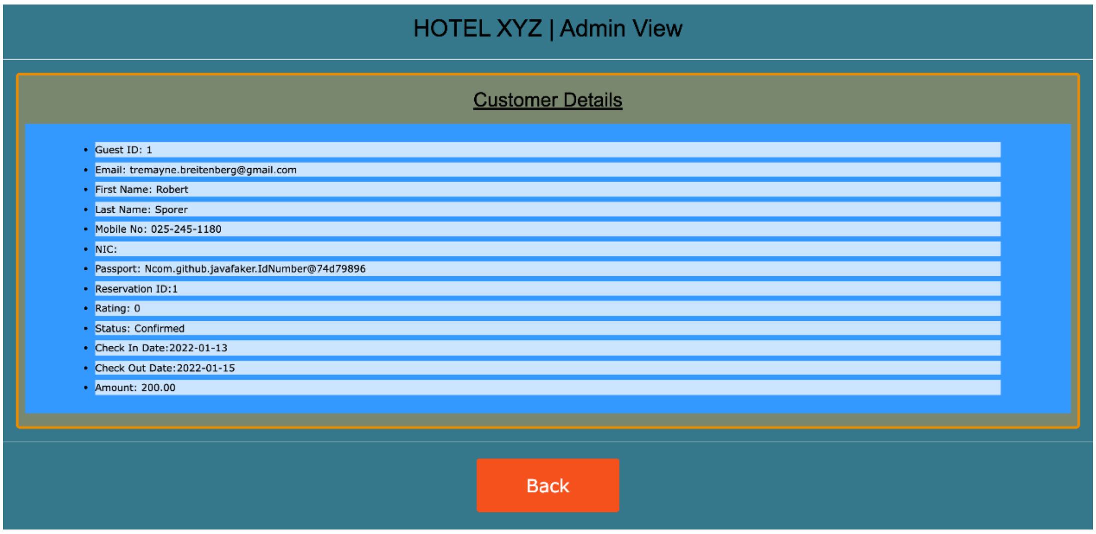
Customer Reviews



HOTEL XYZ | Admin View

Reservations

Reference	Check-In	Check-Out	No Of Rooms	Payment Status	Check Details
#001	2022-01-13	2022-01-15	0	Confirmed	View
#002	2022-01-13	2022-01-15	0	Pending	View
#003	2022-01-13	2022-01-15	0	Confirmed	View
#004	2022-01-13	2022-01-15	0	Confirmed	View
#005	2022-01-13	2022-01-15	0	Confirmed	View
#006	2022-01-13	2022-01-15	0	Pending	View
#007	2022-01-13	2022-01-15	1	Pending	View
#008	2022-01-13	2022-01-15	1	Pending	View
#009	2022-01-13	2022-01-15	2	Pending	View
#0010	2022-01-13	2022-01-15	1	Pending	View
#0011	2022-01-13	2022-01-15	2	Pending	View

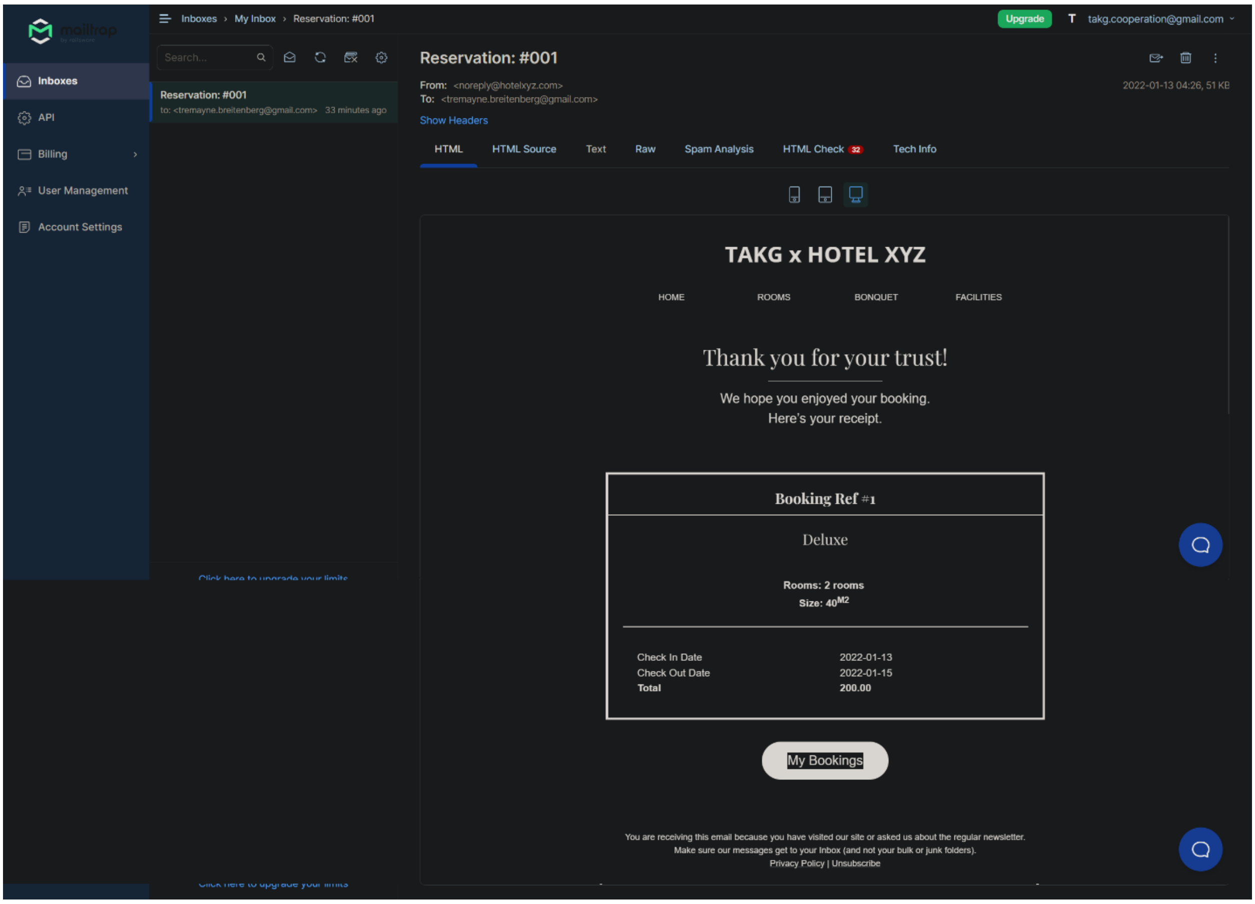


HOTEL XYZ | Admin View

Customer Details

- Guest ID: 1
- Email: tremayne.breitenberg@gmail.com
- First Name: Robert
- Last Name: Sporer
- Mobile No: 025-245-1180
- NIC:
- Passport: Ncom.github.javafaker.IdNumber@74d79896
- Reservation ID:1
- Rating: 0
- Status: Confirmed
- Check In Date:2022-01-13
- Check Out Date:2022-01-15
- Amount: 200.00

9.



Reservation: #001
to: <tremayne.breitenberg@gmail.com> 33 minutes ago

Reservation: #001

From: <noreply@hotelxyz.com>
To: <tremayne.breitenberg@gmail.com>

2022-01-13 04:26, 51 KB

TAKG x HOTEL XYZ

HOME ROOMS BONQUET FACILITIES

Thank you for your trust!

We hope you enjoyed your booking.
Here's your receipt.

Booking Ref #1

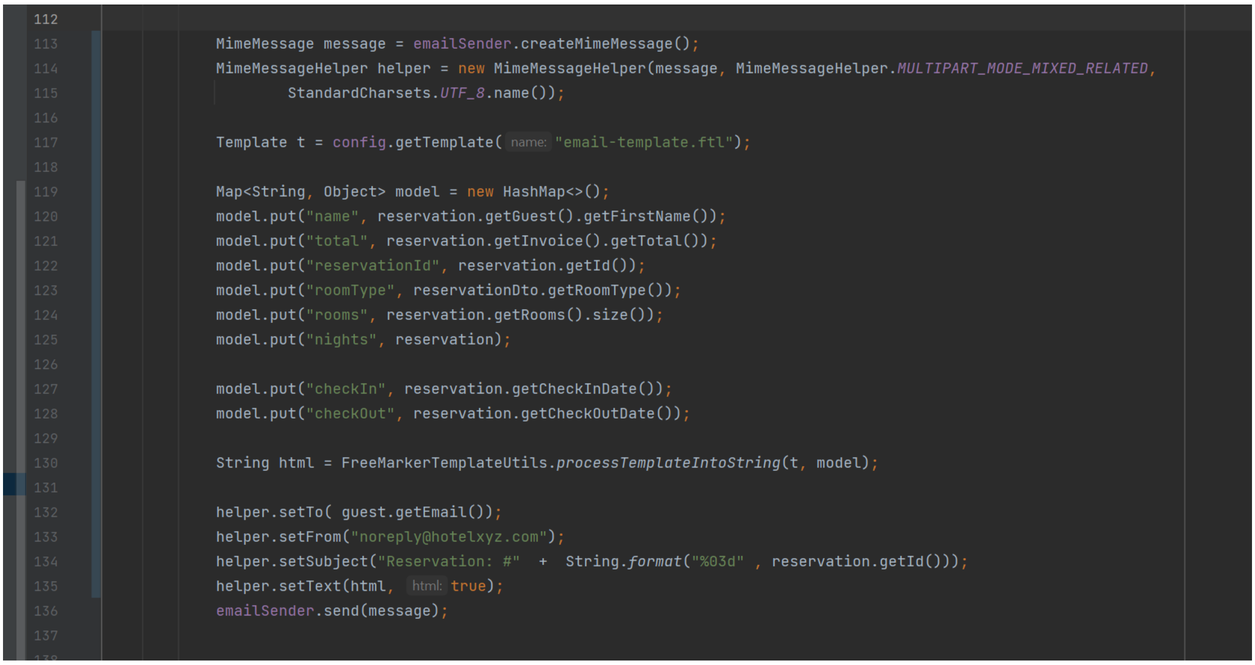
Deluxe

Rooms: 2 rooms
Size: 40M2

Check In Date	2022-01-13
Check Out Date	2022-01-15
Total	200.00

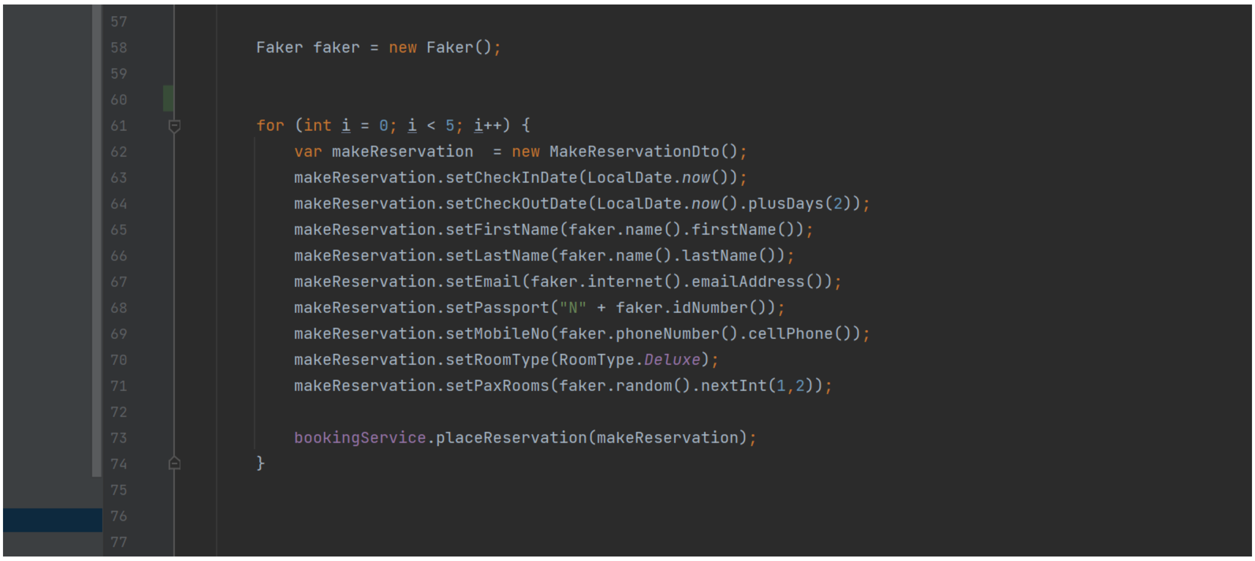
My Bookings

You are receiving this email because you have visited our site or asked us about the regular newsletter.
Make sure our messages get to your Inbox (and not your bulk or junk folders).
Privacy Policy | Unsubscribe



```java
MimeMessage message = emailSender.createMimeMessage();
MimeMessageHelper helper = new MimeMessageHelper(message, MimeMessageHelper.MULTIPART_MODE_MIXED_RELATED,
        StandardCharsets.UTF_8.name());

Template t = config.getTemplate( name: "email-template.ftl");

Map<String, Object> model = new HashMap<>();
model.put("name", reservation.getGuest().getFirstName());
model.put("total", reservation.getInvoice().getTotal());
model.put("reservationId", reservation.getId());
model.put("roomType", reservationDto.getRoomType());
model.put("rooms", reservation.getRooms().size());
model.put("nights", reservation);

model.put("checkIn", reservation.getCheckInDate());
model.put("checkOut", reservation.getCheckOutDate());

String html = FreeMarkerTemplateUtils.processTemplateIntoString(t, model);

helper.setTo( guest.getEmail());
helper.setFrom("noreply@hotelxyz.com");
helper.setSubject("Reservation: #" + String.format("%03d" , reservation.getId()));
helper.setText(html, html: true);
emailSender.send(message);
```

The figure shown below displays how the test data was seeded to the system.



```
        Faker faker = new Faker();


        for (int i = 0; i < 5; i++) {
            var makeReservation  = new MakeReservationDto();
            makeReservation.setCheckInDate(LocalDate.now());
            makeReservation.setCheckOutDate(LocalDate.now().plusDays(2));
            makeReservation.setFirstName(faker.name().firstName());
            makeReservation.setLastName(faker.name().lastName());
            makeReservation.setEmail(faker.internet().emailAddress());
            makeReservation.setPassport("N" + faker.idNumber());
            makeReservation.setMobileNo(faker.phoneNumber().cellPhone());
            makeReservation.setRoomType(RoomType.Deluxe);
            makeReservation.setPaxRooms(faker.random().nextInt(1,2));

            bookingService.placeReservation(makeReservation);
        }
```

- The link to the Google Drive that contains all written code displayed in this report is pasted down below.