**Important Instructions:**

1. **Please read the document thoroughly before you code.**
2. **Import the given skeleton code into your Eclipse.**
3. **Use Java 8 and Spring 5 for solving the ICT.**
4. **You have to test the code and ensure there are no compilation errors before submission**
5. **Business Scenario:**

‘Softech’ company is automating Indian railway tatkal ticket booking system. IRCTC wants passenger to provide details of ticket booking like source and destination city , travel class type and number of tickets .

The company needs a minimum functional service as to get booked ticket details as PNR number, date of travel and fare amount etc. as per passenger travel class and city details.

1. **Functional Requirement Specification:**

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Req. Name** | **Req. Description** |
| **1** | Provide Ticket Booking Details | Get the ticket booking details as customer name, from city,to city, mobile number, travel class, number of tickets etc. |
| **2** | Display Booked Ticket Details | Based on ticket booking details, booked ticket details should be displayed. If from and to city is same , appropriate error message should be shown |

1. **Software Requirements**

|  |  |  |
| --- | --- | --- |
| **#** | **Item** | **Specification/Version** |
| **1** | Eclipse IDE | Oxygen/2020-06 |
| **2** | Maven | 3.x |
| **3** | JDK | 8 |

1. **Skeleton File for Development**

Import the below skeleton code into your eclipse project and implement the required functionalities.



1. **Use case Diagram**

Application User

Provide Ticket Booking Details

Display booked ticket details

details

1. **Technical Requirements**

For all the functional requirements 1 and 2, there is a single component and method specification. Go through them in detail before you start the implementation.

**A. Component Specification:**

|  |  |
| --- | --- |
| ***Requirement Name*** | 1. **Provide Ticket Booking Details and View the booked ticket details** |
| ***Component Definition*** | Launches the ticket booking home page. Allows user to enter the details in the page. On submit, helps to validate the user details. Books ticket as per given details and render it back as results. |
| ***Files Included***  ***(refer Skeleton)*** | * Maven Dependencies are already added in skeleton, pom.xml * JSP files are given in the folder \src\main\webapp\WEB-INF\jsp\. You may need to build the form as per the UI mock-ups provided. Refer UI Design section for more details. * \src\main\resources will have the application.properties file. Use this file for adding view resolvers, server.port etc. |
| ***Responsibilities*** | * Responsible to launch the home page of IRCTC which is named as ticketBooking.jsp * Tie up the model object “TicketBooking.java” to this page and perform basic validations are per design rules * If the details are valid, allow page submit * Based on the values of TicketBooking object, book ticket as per details and render in the ticketBookingResult.jsp |
| ***Design Constraints*** | 1. IrctcApplication must be the starter class of Spring Boot 2. Do not add any additional Maven Dependencies, Do not modify POM.xml 3. View Resolvers must be added only in application.properties 4. All the beans must be defined only using annotations. Use only autowiring for injecting dependencies 5. Use @Component, @Service annotations appropriately for Spring Boot to scan the components 6. Follow the Request Mapping URLs and HTTP methods as specified in the subsequent section. 7. Service class must be autowired to Controller class. 8. No custom validations are required. Use Spring MVC Validations using javax validation for all members of TicketBooking object. The default error messages used for respective annotations on the member variables of TicketBooking class is sufficient. Refer the screen mock document on validation failure scenarios. Refer UI Design section for more details. |
| ***Resources*** | application.properties file must be used to handle view resolvers |
| ***Process Flow*** | 1. Once the app is up and running, launch the home page by using appropriate request mapping URL 2. The Ticket Booking Page will be rendered 3. All the mentioned validations must be performed before page submit 4. The mobile number, from city, to city and number of tickets are essential to book ticket ,to be rendered back as results. For example, if the selected from city is Pune and to city is Bangalore and travel class is Sleeper, number of tickets are 2, application has to book ticket and display PNR number, date of travel, and total fare amount along with train details as train number, train name, departure, duration, arrival. Refer UI Design section for more details. 5. Date of travel is tomorrows date . Total fare amount is multiplication of number of tickets and fare per passenger. E.g. No. Of tickets is 2 and fare per passenger between given cities is 300, then total fare amount will be 600. Passenger can book at a time maximum 4 tickets. Refer UI Design section for more details. 6. PNR number is created with combination of train number and passenger mobile number. It is first 3 digits of train number(available train as per ticket booking) , followed by “-”and first 7 digits of mobile number. Refer UI Design section for more details 7. If ticket booking result does not get any train, display message “ Sorry, currently no train available for given details!!!” Do this check inside controller method. And navigate to trainNotAvailable page. Refer UI Design section for more details |
| ***Exceptional Conditions*** | 1. If from and to city is same, throw an ApplicationException in the controller. 2. Controller Advice must be used to catch all exceptions thrown by the controller, and redirect the user to error.jsp with appropriate error details as shown in screen mockups. Refer UI Design section for more details. |

**B. Method Specification:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Class******Name*** | ***Method Name*** | ***Input Parameters*** | ***Output Parameters*** | | ***Request Mapping*** | |
| IrctcController | showTicketBookingForm | @ModelAttribute("ticketBooking") TicketBooking ticketBooking | String | | URL: /showTicketBookingForm  Method: GET | |
| IrctcController | getCarSearchResultForm | @Valid @ModelAttribute("ticketBooking") TicketBooking ticketBooking, BindingResult result, ModelMap map | String | | URL: /getTicketBookingResultPage  Method: POST | |
| IrctcController | populateFromCities | NA | List<String> | | NA | |
| IrctcController | populateToCities | NA | List<String> | | NA | |
| IrctcController | populateClassType | NA | List<String> | NA | |
| IrctcService | getTicketBookingResult | TicketBooking ticketBooking | TrainInfo | NA | |
| CarStoreService | getPNRNumber | TicketBooking ticketBooking | String | NA | |

**UI Design:**

**Please refer the attached screen mock ups and UI Component Design details while designing the application.**

****

**Note:** Follow Java Naming Conventions, test the code quality by running PMD rules in Eclipse or any other IDE that you use.