**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:**

Windsor car showroom has cars of various brands. Company wants to automate searching cars on the basis of brand, fuel type, budget etc.

The company needs a minimum functional service as to get car details as model name, mileage, seating capacity etc. as per customer choice/search criteria.

1. **Functional Requirement Specification:**

|  |  |  |
| --- | --- | --- |
| **Req. #** | **Req. Name** | **Req. Description** |
| **1** | Provide Car Searching Details | Get the car searching details as customer name, city,mobile number,brand, fuel/transmission, budget upto etc. |
| **2** | Search/Display cars | Based on car search criteria, car details should be displayed. If budget is low, 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 Car Search

Details

Display Cars 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 Car Searching Details and View the cars details as per search criteria** |
| ***Component Definition*** | Launches the car search home page. Allows user to enter the details in the page. On submit, helps to validate the user details. Searches cars as per search criteria 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 car-store which is names as carSearch.jsp * Tie up the model object “CarSearch.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 CarSearch object, search car details and render in the carSearchResult.jsp |
| ***Design Constraints*** | 1. CarStoreApplication 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 CarSearch object. The default error messages used for respective annotations on the member variables of CarSearch class is sufficient. Refer the screenshots document to check the screen mock ups 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 Car Searching Page will be rendered 3. All the mentioned validations must be performed before page submit 4. The brand, fuel/transmission, budgetUpto are essential to search cars ,to be rendered back as results. For example, if the selected brand name is Maruti Suzuki, fuel type is Petrol and budget is upto 5 Lakh, application has to search and display cars with these all criteria. 5. Search result car details along with model name, mileage, seating capacity should be display on carSearchResult.jsp Page. Refer UI Design section for more details. 6. If search result does not get any car, display message “Sorry, No car available matching your profile. Do this check using JSTL tags in JSP. And navigate to home page. Refer UI Design section for more details |
| ***Exceptional Conditions*** | 1. If budget is below 3 Lakh, 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*** | |
| CarStoreController | showCarSearchForm | @ModelAttribute("carSearch") CarSearch carSearch | String | | URL: /showCarSearchForm  Method: GET | |
| CarStoreController | getCarSearchResultForm | @ModelAttribute("carSearch") CarSearch carSearch, ModelMap map,  BindingResult result | String | | URL: /getCarSearchResultPage  Method: POST | |
| CarStoreController | populateCities | NA | List<String> | | NA | |
| CarStoreController | populateBrands | NA | List<String> | | NA | |
| CarStoreController | populateBudget | NA | List<String> | NA | |
| CarStoreController | populateFuelTypes | NA | List<String> | NA | |
| CarStoreService | getCarSearchResult | CarSearch carSearch | List<Car> | 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.