|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | **Web Application & Service Development with Spring Framework v5** **- 2019** | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Instructor | | face-to-face |
| 2 | i\_love\_chocolate | | face-to-face |
| 3 | nori | | face-to-face |

|  |  |
| --- | --- |
| Project name | Online Rent A Car System (ORACS) |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Over the years not only cars technology has expanded but also the ways you can book the desired car for your perfect travel. Our ***Online Rent A Car System*** can provide users the desired comfort, speed, charge and reliability of driving to your destination. It allows users to register, book the wanted car by browsing through a wide variety of models available at different destinations and choose the perfect one which is tested to be safe for driving.  The system will be developed using ***Spring 5 Application Development Framework***.It will implement web-based front-end client using ***JSP*** templates and ***jQuery*** or ***React.js/Vue.js/Angular*** JavaScript/TypeScript libraries for asynchronous page data updates. Each page will have a distinct URL, and the routing between pages will be done server side using ***SpringMVC/ WebFlux***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. There will be also a real-time event streaming from the server to the web client using ***WebFlux*** and ***Server Sent Events (SSE)/WebSocket*** in order to allow the Administrator to monitor real-time position of each rented car and also give information to the customer for speed tests, charges and comparison of different models(?). The main user roles (actors in UML) are:  • *Anonymous User* – can only view the information pages and some example car comparisons and speed tests and maybe one car’s information page.  • *Renting User* (extends *Registered User*) – can login, view cars, filter them, view real-time statistics of a vehicle, compare different models, rent a car, return a car, logout.  • *Support engineer?* (extends *Registered User*) – can add new cars, can put cars in and out of service ( for a repair and after ), can also remove cars*,* see the *Realtime positions* of cars*,* and check charges and states of travelers, also can report frauds and other misfortune events.  • *Administrator* (extends *Registered User*) – can manage (create, edit car and user data and delete) all *Registered Cars and Users*, as well as *Statistics*, *Test drives* and *Test Results*. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse models and see information** | The *User* can browse the information views and filter the view by position, model, price, year, type of gas. When clicking to a chosen model can see more information about it in an extended view. | All users except Anonymous user |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, mobile phone numbers, id(?) number, and choosing password. By default, all new registered users have *User* role.  *Administrator* can register new by entering *User Data* and choosing a Role (*Student, Support*, or *Administrator*). | *Anonymous User, Administrator* |
| * 1. **Validate user registration** | User data should be checked in a registry for identifying the user and associating him with a real person. | *Administrator, Support* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data (but not the provided id as it is used in case of damages.*  *Administrator* can view and edit *User Data* of all *Users* and assign them *Roles*: *Student, Support*, or *Administrator*. | *Registered User, Administrator* |
| * 1. **Rent a car** | *Users* can rent a car by choosing the car by it’s position, model and other characteristics. User should give the date of getting the car from the chosen car station and an around date ( of maximum 2days overdue) and place to return it. Also should provide a travel region description. | *User* |
| * 1. **Manage cars** | All cars are going to be added, put in (and out of ) service station and deleted. Cars will be checked when rented and monitored their real time positions. All kinds of eventual damages, frauds and policy breaking events will be displayed. | *Administrator, Support* |
| * 1. **Check charges and account** | User can check his charges (which will be added according used gas and % of the travelled distance) and account. |  |
| * 1. **Monitor comparisons of cars** | *Users* can seereal time comparisons between different cars. | *User, Administrator, Support* |
| * 1. **Monitor profits and losses** | According to real time profits and losses to change charges. | *Administrator, Support* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Index/Home** | Presents to the Anonymous user the advantages of using the system as showing a model and providing the ability to view a sample description of a car and some tests and statistics. Mentioning the ability to compare cars. | /home? |
| * 1. **Login** | A modern looking login modal. | */login* |
| * 1. **Registration** | A modern looking register modal with validation fields. | */registration* |
| * 1. **Browse a car** | Presents a screen with wide variety of models the user can choose from and a filtering tree in the right.(location, charges...etc.) | */index?* |
| * 1. **Car view** | Presents a view allowing the *User* to see the description of the car - brand, model, type of gas and other specifications. | */{carId}*  *as 123456 is an id of the car* |
| * 1. **Car tests, statistics and information** | By a button from the car view screen to see more information for test drives, statistics, charges, etc. | */{carId}/Information* |
| * 1. **Car comparison** | Provides ability to compare 2 models and make the choice easier. | */car1234\_vs\_car4567* |
| * 1. **User information** | Presents ability to browse, filter and manage User information*.* | */userInformation* |
| * 1. **Admin profit and losses view** | Presents in real time the Students’ progress on active Tests, subject to Role restrictions described in UCs. | */dashboard* |
| * 1. **Real time monitoring of cars position** | Presents ability to manage (CRUD) Users and their User Data (available for *Administrators* only, as described in UCs). | */carPosition* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *ORACS* and modified entity is returned as result from POST request). Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions described in UCs. | */api/users/{userId}* |
| * 1. **Login** | POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests. | */api/login* |
| * 1. **Logout** | POST a logout request for ending the active session with ORACS*,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Car information** | GET Car information, and POST new *cars or in/out of service* (Id is auto-filled by *ORACS* and modified entity is returned as result from POST request), according to *User's Role* and identity security restrictions. | */api/cars* |
| * 1. **Car statistics** | GET, PUT, DELETE Car Statistics (including test drives, charges, etc) | *{carId}/statisttics* |
| * 1. **Rent a car** | POST users request to rent a specific car with provided information | */api/rent/{caId}* |
| * 1. **Add, Edit, Remove a car** | GET, PUT, DELETE Car *Data* with specified car*Id*. | */api/cars/{operation}/{carId}* |
| * 1. **Monitor cars** | GET *Car Results* for *specific car by id,* and POST new Car *Information result as displaying diagrams and test drive results(how?)* (Id is auto-filled by *ORACS* and modified entity is returned as result from POST request). | */api/cars/information/{carId}* |
| * 1. **Monitor profits and losses** | GET *profit and losses results in diagram charts.* | */api/profitAndLosses* |
| * 1. **Active rents** | SSE event streaming of Cars’ renting status on currently active *Cars.* | */api/active-tests* |