|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | **Web Application & Service Development with Spring Framework v5** **- 2019** | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Anjana | | face-to-face |
| 2 | doivanov | | Face-to-face |

|  |  |
| --- | --- |
| Project name | Boaty McBoatface |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The Boaty McBoatface project intends to provide a universal and highly configurable platform for online purchasing of tickets for public transports in any country you go. It lets different countries provide and configure their set of rules that match with their ticketing system (prices, time of validity for ticket, cards etc.). For the end users the system will provide a unified way of buying public transport tickets for all countries that have decided to use our system. The system will be developed using ***Spring 5 Application Development Framework***. For the front-end part we’ll use React.js or Angular. For the backend we’ll use SpringMVC. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information about offered services** | The *User* can browse the home page which provides general info about the system’s services. If available user can also read general info about current country ticketing system | All users |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. | *All users* |
| * 1. **View users** | View all registered users in the system | *Admin* |
| * 1. **Assign admin role** | *Assign admin role to already registered user* | *Admin* |
| * 1. **Add types of public transport** | Create types of public transport for which end users can buy tickets through the system | *Admin* |
| * 1. **Add lines** | Add concrete lines for given type of public transport | *Admin* |
| * 1. **Add parking information** | Add parking information – working hours and days, price model, time limit etc | *Admin* |
| * 1. **Add tickets** | Add ticket that can be bought by end user. User can specify time validity of ticket, transports/zones for which ticket is valid etc. | *Admin* |
| * 1. **Add general info** | Create article about how public transport is organized (how ticketing system works) in current country/city | *Admin* |
| * 1. **Pair card** | Pair credit/debit card to system | *Registered User* |
| * 1. **Top-up account** | Transfer money from paired card to account | *Registered User* |
| * 1. **Buy ticket** | Buy ticket for public transport | *Registered User* |
| * 1. **Pay parking** | Pay parking in city | *Registered User* |
| * 1. **View active tickets** | View tickets that are currently valid | *Registered User* |
| * 1. **View paid parking** | View currently paid parkings | *Registered User* |
| * 1. **Check ticket validity** | Check ticket validity | *Inspector* |
| * 1. **Check if parking paid** | Check if parking paid for particular car in particular place | *Inspector* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents the introductory information for the purpose of the system as well as detailed instructions on how to start using it. Prominently offers ability to register. | / |
| * 1. **Ticketing system info in country** | Presents summarized information about how the ticketing system works in given country | */general\_info* |
| * 1. **Pair card** | Page to register card details | */pair\_card* |
| * 1. **Buy ticket** | Page to choose from available tickets and buy one | */tickets/buy* |
| * 1. **View active tickets** | Page that shows active tickets details | */tickets/active* |
| * 1. **Login or register** | Presents a view allowing the users to login or register. | */login* |
| * 1. **Pay parking** | View showing available parking offers and allowing user to choose and pay for one | */parking/pay* |
| * 1. **View active parkings** | View active parkings | */parking/active* |
| * 1. **View registered users** | View showing list of all registered users. Admin | */users/list* |
| * 1. **Assign admin role** | .Choose user from list whom to assign admin role | */users/list* |
| * 1. **Registered transport types** | View currently registered transport types | */transport\_types* |
| * 1. **Register transport type** | Add new transport type | */transport\_types/add* |
| * 1. **Registered tickets** | View currently registered tickets | */tickets/registered* |
| * 1. **Register ticket** | Add new ticket type | */tickets/add* |
| * 1. **General info** | View/add/edit general info article | */general\_info/edit* |
| * 1. **Check parking** | View to enter vehicle details for which to be checked if parking has been paid | */parking/check* |
| * 1. **Check ticket** | Ticket scanning | */tickets/check* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* | */api/users* |
| * 1. **User** | GET, PATCH, DELETE *User Data* for *User* with specified *userId* | */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 *OKTS,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Users** | List payed parking | /api/users/{user\_id}/parking |
| * 1. **Users** | List bought tickets | /api/users/{user\_id}/tickets |
| * 1. **Users** | List user’s cards | /api/users/{user\_id}/cards |
| * 1. **Users** | List user’s fines | /api/users/{user\_id}/fines |
| * 1. **Users** | GET, POST payment details | /api/cards |
| * 1. **Payments** | GET, PATCH, DELETE a single card | /api/cards/{card\_id} |
| * 1. **Transport** | GET, POST transport data. Create or read a types of public transport | /api/transports |
| * 1. **Transport** | GET, POST, DELETE transport data. Delete, edit or view single type  Of public transport | /api/transports/{transport\_id} |
| * 1. **Lines** | GET, POST view or create a type of transport line | /api/lines |
| * 1. **Lines** | GET, PATCH, DELETE one transport line. Edit, delete or read line details. | /api/lines/{line\_id} |
| * 1. **Parking** | GET, POST list or create a parking instance. | /api/parkings |
| * 1. **Parking** | GET, PATCH, DELETE view, update or delete a single parking instance | /api/parkings/{parking\_id} |
| * 1. **Parking** | Pay for parking | /api/parkings/{parking\_id}/pay |
| * 1. **Tickets** | GET, POST list, create a new type of ticket | /api/tickets |
| * 1. **Tickets** | GET, PATCH, DELETE view, edit or delete a single type of ticket | /api/tickets/{ticket\_id} |
| * 1. **Tickets** | POST buy a ticket | /api/tickets/{ticket\_id}/buy |
| * 1. **Fines** | GET, POST view or create a new fine | /api/fines |
| * 1. **Fines** | GET, DELETE view or delete a fine | /api/fines/{fine\_id} |