| Assignment 1 | | Ticket system | |
| --- | --- | --- | --- |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2023 | |
|  | | | |
| Project author | | | |
|  | Name | | FN |
|  | Kamen Vakavchiev | | 6MI3400293 |

| Project name | Ticket system |
| --- | --- |

| 1. Short project description (Business needs and system features) |
| --- |
| In the modern age the way we buy tickets is very different from the past. All is done online which is faster and more convenient. The system will allow registration and account management, creation of events and venues, joining groups of events, purchasing of tickets which gives the user a receipt and a ticket qr code for entry to the event and giving review/feedback on the event. The system will be a dynamic web application using React.js as a front-end and Node.js with Express.js for backend. The backend will have a REST API.  The main user types are:  • *Guest* – can only view the available events  • *Client* (extends *Registered User*) – can choose a *Test* to complete – *Test Results* are saved automatically on test completion.  • *Organizer* (extends *Registered User*) – can create events and venues.  • *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, as well as all events and tickets. |

| 1. Main Use Cases / Scenarios | | |
| --- | --- | --- |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Check events** | The user can view events with their respective information - Date and time, venue, organizer, description, reviews. | All users |
| * 1. **Register** | *Gues User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. By default, all new registered users have *Client* role.  *Administrator* can change the role of an account. | *Anonymous User, Administrator* |
| * 1. **Manage User Data** | *Client Users* can change their own names and passwords.  *Administrator* can view and edit *User Data* of all *Users* and assign them *Roles.* | *Client User, Administrator* |
| **2.4. Create Venue** | *Organizer User* can create venues by providing them a name, coordinates/location, description and optinally photos | *Organizer* |
| **2.5. Create Event** | *Organizer User* can create events by providing them a venue/location, date and time, ticket price | *Organizer* |
| **2.6. Purchase Ticket(s)** | *Client User* can purchase tickets for events when on the page for a specific event. | *Client User* |
| **2.7. View Ticket(s)** | *Client User* can see and download the tickets he bought in his profile. | *Client User* |
| **2.8. Leave review/feedback on an event** | *Client User* can leave feedback on the event by giving 1 to 5 star rating, text review and optionally some photos. | *Client User* |
| **2.9. Creating groups** | Organizer can create groups for a type of events. | Organizer |
| **2.10. Joining groups** | *Clients can join groups in order to get notified about the events in the specific group.*  *Organizers can join groups to create events in them.* | Organizer, Client User |
|  |  |  |
|  |  |  |

| 1. Main Views (SPA Frontend) | | |
| --- | --- | --- |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | The main page which will display a view of all events from the groups user is in. | */* |
| * 1. **Login** | Login for Guests | */login* |
| * 1. **Register** | Registration for Guests | */register* |
| * 1. **Account management** | Allows registered users to manage their accounts(names, password) | */account* |
| * 1. **Opens the event view** | Opens the event view where you see event information if you are *Client User.*  Opens the event view where you can edit the information for the event if you are *Organizer User.* | */events/{eventId}* |
| * 1. **Purchase tickets** | Opens a view for puchasing a ticket for the specified event if you are *Client User.* | */events/{eventId}/purchase* |
| * 1. **Leave a review** | You can leave a review if you are *Client User.* | */events/{eventId}/review* |
| * 1. **See tickets** | Displays the tickets you have purchase if you are a *Client* *User*. | */tickets* |
| * 1. **Group subscriptions** | Clients and Organizers can see groups for joining | /groups |
| * 1. **Group management** | Admin manages and approves groups | /groups |

| 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 *OKTS* 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 decribed 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 *OKTS,* and invalidating the issued *Security Token*. | */api/logout* |
| **4.5. Register** | POST a registration for Guests. | */api/register* |
| * 1. **Check event info** | GET see information for an event. | */api/events/{eventId}* |
| * 1. **Purchased tickets** | GET the purchased tickets for the currently logged in user. | */api/tickets* |
| * 1. **Leave and check reviews** | GET - check reviews,  POST - leave review | /api/events/{eventId}/reviews |
| * 1. **Purchase tickets** | POST - purchase ticket | /api/events/{eventId}/purchase |
| * 1. **Create event** | POST - create an event.  Only for *Organizer* User. | /api/events/ |
| * 1. **Join groups** | POST - Clients and Organizers join the specified groupfid | /groups/{groupId}/join |
| * 1. **Admin approve groups** | GET - see list of all group not approved yet  POST - send groups to be approved | /groups/ |