|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2025 | |
|  | | | |
| Project author | | | |
|  | Name | | FN |
|  | Ilia Petrov | | 62520 |

|  |  |
| --- | --- |
| Project name | Fest Ticketing Platform |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The Fest project offers a ticketing platform, where users can buy tickets for different events and will receive invoice on their emails. Also there is functionality for admins to create/manage events and create tickets for the events. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse available events and see available tickets** | The *User* can browse for events and for the available tickets (Home and EventDetals views) | All users |
| * 1. **Login/Register** | *Users can create account for the system, otherwise they can’t view available events and tickets. Admin users are setup manually, so their account is not created from Register view. Both clients and users login via the Login view* | All users |
| * 1. **Manage events** | *Admin can create event via the admin REST API and also can upload the thumbnail for the given event* | *Administrator* |
| * 1. **Manage tickets** | *Administrator* can populate with tickets each event. The tickets can be in 2 categories: STADARD and VIP | *Administrator* |
| * 1. **Buy ticket** | *Each user can select what ticket he wants (STANDARD or VIP) for a given event and can fill up his credit/debit card information in the Checkout view* | *Administrator, Registered User* |
| * 1. **Receive ticket on email** | *After successful payment user receives on his email a pdf, which contains its ticket that contains id, place, price and type for the bought ticket* | *Administrator, Registered User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents the available events | /home |
| * 1. **Login** | Presents the login view, where the user should enter his email and password | */login* |
| * 1. **Register** | Presents the register view, where the user should enter his email and password in order to create an account | */register* |
| * 1. **EventDetails** | Presents *available tickets for a given event* | */events/:id* |
| * 1. **Checkout** | Presents a view allowing the *user to enter his credit/debit card information in order to buy the ticket* | */checkout* |
| * 1. **PaymentStatus** | Presents a payment status of the ticket | */payment/:id* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Get all available events | *GET /api/v0/events* |
| * 1. **Login** | Enter the login credentials in order to get the JWT token to auth the user | *POST /api/v0/login* |
| * 1. **Register** | Enter the register credentials in order to get the JWT token to auth the user | *POST /api/v0/register* |
| * 1. **EventDetails** | Load metadata for the given event + data about the available tickets | *GET /api/v0/events/:id*  *GET api/v0/events/:id/thumbnail*  *GET /api/v0/tickets/available* |
| * 1. **Checkout** | Gets the ticket details and setup Stripe form for payments. When data is sends a POST request to execute the payment | *GET /api/v0/payments/publishable-key*  *GET /api/v0/tickets/events/summary/:id*  *POST /*api/v0*/payments* |
| * 1. **PaymentStatus** | Monitors the status of the payment | *GET /api/v0/payments/:id* |
| * 1. **Admin** | Populate the system with events and after that attach to them a thumbnail. After that bulk insert each event with tickets | *POST /api/v0/events*  *PUT /api/v0/events/:id/thumbnail*  *POST /api/v0/tickets* |