|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2023 | |
|  | | | |
| Project author | | | |
|  | Name | | FN |
|  | Ivan Petrov | | 3MI0700035 |

|  |  |
| --- | --- |
| Project name | Travel Agency System |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| In this era of freedom for everyone to travel and go everywhere. The **Online Travel Agency System(OTAS)**provides ability for travelers and hotel owners to provide the best experience with elimination of going to multiple sites to make plan of your travel. In addition to that it allows users to register, and hotel owners to have full control over representation of their hotel, as well getting the chance of contacting the support team at any moment. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** and ***TailwindCSS/MUI*** as front-end , and ***Node.js + Express*** as back-end technologies. Each view will have a distinct URL, and the routing between pages will be done client side using ***React Router***. The back-end will be implemented as a ***REST*** ***API*** using **data modeling with PrismaORM**. There will be also a realtime chat system, multiple options to schedule your travels (Example showing you the possible flights). The main user roles (actors in UML) are:  • *Anonymous User* – can do nearly the same things as *Traveler* except saving all their travels on their account, having easy way of checking all support chat tickets  • *Traveler* (extends *Registered User*) – can choose a view all their reservations, as well they have the option to review hotels • *PropertyOwner*(extends *Registered User*) – can list property*,* answer and communicate with reviewers and monitor and edit all their properties trough dashboard  • Support(extends *Registered User*) – can block user from being PropertyOwner (due to going against TOS), see and accept support tickets, as well as seeing problem logs reported by *Travelers.* |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information** | The *User* can browse the information views (Home, Hotels, About) in *OTAS* . | 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. By default, all new registered users have *Traveler* role. The traveler can list a *Property and upon approval getting PropertyOwner role*. | *Anonymous User* |
| * 1. **Change User Data** | *Registered User* can view and edit their personal *User Data* as well as enter more *Token Keys* to participate in more *Student Groups*.  *Support* can view and remove roles of all *Users* and assign them *Roles*: Traveler, PropertyOwner. | *Registered User, Support* |
| * 1. **Manage Users** | *Support* can browse and filter users based on different criteria: first and last name, email, Role.  *Support* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC). | *Support* |
| * 1. **Manage Properties** | *PropertyOwner* can browse and filter *Properties* based on different criteria: name of *Property,* creation date, etc.  *PropertyOwner* can list new *Property.*  *Support and PropertyOwner* can choose a *Property* to manage, and delete it if necessary. | *PropertyOwner, Support* |
| * 1. **Manage Chat tickets** | *Anonymous users* get ticket link so they can save their chat *User* canbrowse their *Support tickets*, add new *Tickets* using *Request ticket.*  *Support* can browse tickets of all *Users*, accept and resolve them. | *Anonymous user,Registered user, Support* |
| * 1. **Add new reservation** | *Traveler can make new reservation for all hotels*  *PropertyOwner can contact reservation requester and decline reservations* | *Traveler, Property Owner* |
| * 1. **Add review** |  | *Traveler* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Front-end) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents how is the system different than others and showing the possibilities in the system. Prominently offers ability to register. | / |
| * 1. **Hotels** | Presents all hotels corresponding to chosen filters | */hotels* |
| * 1. **Support tickets** | Presents support chat available to 1 to 1 connection (Registered User and Support user) | */support/{id}* |
| * 1. **User Registration** | Presents register form for Anonymous users to register **OTAS** | */register* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data*, as well deactivation and deletion of an account. | */profile* |
| * 1. **Reservations** | Presents ability to browse, filter and manage reservations | */reservations* |
| * 1. **Dashboard** | Depending on Role it shows either dashboard for Support users or PropertyOwner users. For Support users shows dashboard with access to support tickets, the user management system based on Manage Users UCs | */dashboard* |
| * 1. **About** | Presents information about the OTAS project and his owner. | */about* |

|  |  |  |
| --- | --- | --- |
| 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 OTAS 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 OTAS*,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Reservations** | GET *Reservation*, and UPDATE *Reservation* (Id is auto-filled by OTAS and modified entity is returned as result from GET request), according to *User's Role* and identity security restrictions. | */api/reservations/{reservationId}* |
| * 1. **Support tickets** | GET all *Support Tickets* | */api/supticket* |
| * 1. **Support tickets** | GET, PUT, DELETE *Ticket data* (including *Chat messages*) for *Ticket* with specified *ticketId*. | */api/supticket/{supportTicketId}* |
| * 1. **Reservations** | GET *Reservation data* (according to *User's Role* and identity) | */api/reservations* |
| * 1. **Reservations** | GET, PUT, DELETE *Reservation data* (according to *User's Role* and identity) for *Reservation* with specified reservationId | */api/reservations/{reservationId}* |
| * 1. **Reviews** | GET *Review data* | */api/reviews/* |
| * 1. **Reviews** | GET, PUT, DELETE *Review data* (according to *User's Role* and identity) | */api/reviews/{reviewId}* |
| * 1. **Hotels** |  |  |
| * 1. **Rooms** |  |  |