|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | **Web Application & Service Development with Spring Framework v5** **- 2024** | |
|  | | | |
| Project author | | | |
| № | Names of Participants | | FN |
| 1 | Tugay Kafa | | 4MI0600275 |

|  |  |
| --- | --- |
| Project name | Event Management Platform |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The ***Event Management Platform (EMP)*** allows users to create, manage, participate in events and upload content to the event. The system will be developed using ***Spring 5 Application Development Framework***.It will implement web-based front-end client using ***Thymeleaf*** and ***jQuery***. Each page will have a distinct URL, and the routing between pages will be done server side using ***SpringMVC***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization.  The main user roles (actors in UML) are:  • *Anonymous User* – can only view the events without signing up for them.  • *Organizer* (extends *Registered User*) – can create and manage their own events. They can add venue information and upload participant materials. Organizers only have control over the events they create themselves.  • *Attendee* (extends *Registered User*) – can view and register for events. After registration, they can receive materials related to the events in which they participate.  • Admin (extends Registered User) – can add, edit and remove places. Also have access to every event and resources and can organize events, but cannot participate in them. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse events** | The *User* can browse events . | 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. After registration every user can sign up for events and create his own. | *Anonymous User* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data.* | *Registered User* |
| * 1. **Create Event** | Any Registered User can create an event by giving it a name, choosing a place, specifies how many sessions will have and determining the number of participants. Also he can add materials.  Places can be added only for Admin | *Registered User, Admin* |
| * 1. **Edit Event** | *Organizer can edit information about event like place, number of participants and materials.* | *Organizer, Admin* |
| * 1. **Delete Event** | *Only Organizer can delete event.* | *Organizer, Admin* |
| * 1. **Register for an Event** | *Registered User can register for an event if there are available places and he is not Organizer of the event.* | *Registered User* |
| * 1. **Unsubscribe from an event** | Attendees can unsubscribe from an event if they want. In this way they free up space for other attendees | *Attendee* |
| * 1. **Get Event Resources** | *Organizers and Attendees can access event materials.* | *Organizer, Attendee, Admin* |
| * 1. **Edit Event Resources** | *Organizer can add more resources or edit/delete current.* | *Organizer, Admin* |
| * 1. **Browse Event** | *Every user can browse events by given name, place, organizer or duration*. | *All users* |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | POST new *User Data* (Id is auto-filled by *EMP* and modified entity is returned as result from POST request). Available only for *Anonymous User*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId* | */api/users/{username}* |
| * 1. **Login** | POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests. User can only sign in if his account is approved! | */api/login* |
| * 1. **Logout** | POST a logout request for ending the active session with *EMP,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Events** | GET *Events* | */api/events* |
| * 1. **Event** | POST new *Event* (Id is auto-filled by *EMP* and modified entity is returned as result from POST request) | */api/events/create* |
| * 1. **Event** | GET, PUT, DELETE *Event* (including resources to event) for *event* with specified *eventId*, according to *User's Role* and identity security restrictions. | */api/events/{eventId}* |
| * 1. **Resources** | GET event resources (according to *User's Role* and identity) and POST new *resource* (Id is auto-filled by *EMP* and modified entity is returned as result from POST request). | */api/events/{eventId}* |
| * 1. **resource** | GET, PUT, DELETE *event resource* for Even*t* with specified *resourceId*. | */api/events/{eventId}/resources/{resourceId}* |
| * 1. **Places** | GET *available places in database* for *Events*, and POST new Place(Id is auto-filled by *EMP* and modified entity is returned as result from POST request). | */api/places* |
| * 1. **Test Result** | GET, PUT, DELETE *Place*(according to *User's Role* and identity) for with specified {*placeId}*  *.* | */api/places/{placeId}* |
| * 1. **Approve User** | Patch User Data(changes account type to approved) | */api/users/{userId}* |