Events book Summary

1. Project Description

1.1. Idea - Eventsbook is a system for management, sharing and searching of events and the tickets for them. It utilizes groups where many users can participate. In every group, the organizers can create and manage events.

1.2. Technologies

- 1.2.1. Hibernate - Persistence layer
- 1.2.2. Spring Boot 6 - Web Framework
- 1.2.3. Thymeleaf - templates

1.3. Roles

- 1.3.1. Admin - can review the information for other users
- 1.3.2. User

1. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
1.1. Register	*Anonymous User* can register in the system by providing the necessary information.	*Anonymous User*
1.2. Login	*Anonymous User can login providing the correct password and email.*	*Anonymous User*
1.3. Manage Profile Data	Every *Registered User*can manage his personal data and also.	*Registered User*
1.4. Manage Card	Every *Registered User* can manage his card, adding tickets to his card, removing and browsing the tickets in his card.	*Registered User*
1.5. Browse Orders history	Every *Registered User* can see what orders he has done in the past, including information for the ticket, event, price and time.	*Registered User*
1.6. Manage Groups	*Group Administrator and Group Owner* can create events, accept requests and see the balance of the group.	*Group Administrator, Group Owner*
1.7. Buy Tickets for the Events	Every *Registered User* can check out his card and thus buys the items he has chosen.	*Registered User*
1.8. Activate Events	The *Group Administrator and Group Owner* can activate the ticket of the User, who wants to participate in a particular event, if necessary. This User provides the Ticket Number to the Group *Administrator or Group Owner*. Then, they should be on the page of the Event, fill the Ticket Number in a short form and submit. If the Ticket Number and the Ticket are valid for the Event, it appears a success message, otherwise the reason is shown.	*Group Administrator, Group Owner*
1.9. Browse Event Groups for joining	All Users can browse groups in order to join these groups.	*All Registered Users*
1.10. Create Events	*Every Group Administrator and Group Owner* can create Events, specifying the time, name, tags, photo and description.	*Group Administrator, Group Owner*
1.11. Create Tickets	*Every Group Administrator and Group Owner* can create tickets for a particular event, specifying the description, price and tickets count.	*Group Administrator, Group Owner*

2. Main Views (Frontend)

View name	Brief Descriptions	URI
2.1. Login	Presents a view allowing the users to login	/auth*/login*
2.2. Register	Presents a view allowing the users to register	*/auth/register*
2.3. User Profile	Presents a view allowing the users to see their profile information like names, email, balance and other metadata about their profile. Also they can add and withdraw funds from their account	*/profile*
2.4. User Card	Presents a view allowing the users to see the tickets in their card.	*/card*
2.5. Orders History	Presents a view allowing the users to the orders they've done	*/profile/orders*

2.6. User Tickets	Presents a view allowing the users to see the concrete tickets they've bought and when they were activated.	*/profile/tickets*
2.7. Join Groups	Presents a view allowing the users to see groups where they can join.	/groups/join
2.8. User Groups	Presents a view allowing the users to see the groups where they participate.	*/profile/groups*
2.9. Group	Presents a view allowing the users to see all the information about a group, including, its members and events.	/groups/{groupId}
2.10. Event	Presents a view allowing the users to see all the information about an event, including available tickets and reviews.	/events/{eventId}

3. API Resources (Java Backend)		
View name	**Brief Descriptions**	**URI**
3.1. Users	GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *the Backend* and modified entity is returned as result from POST request). Available only for *Administrators*.	*/api/users*
3.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions described in UCs.	*/api/users/{userId}*
3.3. Login	POST *User Credentials* according to the User role and receive a valid *Security Token* to use in subsequent API requests.	*/api/login*
3.4. Logout	POST a logout request for ending the active session with *Events Book,* and invalidating the issued *Security Token*.	*/api/logout*
3.5. Event Groups	GET *Event Groups*, and POST new *Event Group* (Id is auto-filled by *EventsBook* and modified entity is returned as result from POST request), according to *User's Role* and identity security restrictions.	*/api/groups*
3.6. Event Group	GET, PUT, DELETE *Event Group* for *Student Group* with specified *groupId*.	*/api/groups/{groupId}*
3.7. Events	POST a new event (Id is auto filled by *Events Book* and modified entity is returned as a result from POST request). GET Events with many filtering options.	*/api/events*
3.8. Event	GET, PUT, DELETE *Event Data* for *Event* with specified *eventId*.	*/api/event/{eventId}*
3.9. Event Reviews	GET *Reviews*, providing eventId, userId, organizerId, and POST new *Event Review* (Id is auto-filled by *Events Book System* and modified entity is returned as result from POST request).	*/api/reviews*
3.10. Event Review	GET, PUT, DELETE *Event Review* (according to *User's Role* and identity) for *Review* with specified *reviewId*.	*/api/reviews/{reviewId}*
3.11. Tickets	GET *Tickets*, providing eventId, userId, organizerId, and POST new *Ticket Result* (Id is auto-filled by *Events Book System* and modified entity is returned as result from POST request).	*/api/tickets*
3.12. Ticket	GET, PUT, DELETE *Event Ticket* (according to *User's Role* and identity) for with specified *ticketId*.	*/api/ticket/{ticketId}*