|  |  |
| --- | --- |
| Project name | BiblioExchange |

|  |  |  |
| --- | --- | --- |
| Project author | | |
| № | Names of Participants | FN |
| 1 | Deyan Delchev | 4MI0800210 |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The books existed long before Internet. It can be extremely tedious to an information about a specific book or purchase one online second-hand. **BiblioExchange**allows the user to find information about a book or author , or to add or find a listing for specific book. In addition to that it allows users to register, moderators to manage the library information (like Wikipedia) and administrators to manage the whole system. The system will be developed using ***Golang, it is going to use sql builder like jet or sqlc, Keycloak for Identity and Access Management ,Postgres and possibly object storage like Minio***.It will implement a web-based front-end client using Golang’s ***html/template*** package combined with HTMX OR ***React.js SPA.*** The backend most likely will be HyperText driven (if HTMX is used) or will be implemented as a ***REST/JSON API*** using JSON data serialization. There will also be a real-time chat using websockets. The main user roles are:  • *Anonymous User* – can only view the information the library(see information for books and authors) and browse listing.  • *Regular user*– can do all the things that *Anonymous User can*  + *add information in the library, add listings and chat with another users*.  • Moderators – can do all the thing that Regular user can + edit miss-information in the library, restrict specific users from adding or editing information in the library, restrict specific users from adding a new listings, remove listings, chat ban someone, permaban someone.  • *Administrator* – can do almost everything possible. It will have access to Keycloak admin panel. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse library** | The *User* can browse the information for different book and authors in the library | All |
| * 1. **Register** | New user can register. This use case is handled by Keycloak. | *Anonymous User* |
| * 1. **Change User Data** | This use case is handled by Keycloak. | *Regular, Moderator ,Administrator* |
| * 1. **Manage Users** | *Administrator* can browse and filter users based on different criteria: first and last name, email, Role.  *Administrator* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC) or delete.  *Administrator* can create a new user using *Register UC*.  Administrator can impersonate users.  This use case is handled by Keycloak. | *Administrator* |
| * 1. **Add or edit information in the library** | Users can add information for missing book or author OR can edit a existing one (Some mechanism except the report button will be needed to stop trolls from ruin the whole thing. Wikipedia or StackOverflow will be good place to check). Moderators can also can add and edit the information and they will have bigger priority. Also moderators can delete information. | *Regular, Moderator ,Administrator* |
| * 1. **Create listing** | *Users can create listing for book or group of books. They need to add photos, information about its/their condition, price and add links to the book/s page/s in the library.* | *Regular, Moderator ,Administrator* |
| * 1. **Browse and search listing** | *Every user can browse and search listing in the system. Also every user can view details for specific listing like what books is listing for, condition, price, ect* | *All* |
| * 1. **Chat listing owner** | *Every user with account can message the owner of specific listing for more information or purchase method.* | *Regular, Moderator ,Administrator* |
| * 1. **Personal message** | *Moderator can message everybody.* | *Moderator ,Administrator* |

|  |
| --- |
| 1. Frontend Missing(It has postman collections for testing of the apis) |

1. Backend api point: backend will return json.