|  |  |
| --- | --- |
| Project name | Go notes |

|  |  |  |
| --- | --- | --- |
| Project author | | |
| № | Names of Participants | FN |
| 1 | Kaloyan Mehandzhiyski | 5MI0700025 |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Every busy student from FMI needs to keep some notes because he can forget them – date for exams, themes for projects, conspectus and etc.  Let’s introduce Go notes!  The basic idea for the user is to write down and keep notes on a computer so he doesn’t forget something or just his PC to remember for him. Why using a computer but not a paper? Because it’s eco-friendly way, reducing a lot of paper and because there will be a challenge for the developers to do it!  The implementation of this application is using Golang, JSON, html and Chrome/Firefox browser for using its interface.  It will implement a web-based front-end client using Golang’s html/template package, HTTP queries and a local JSON files for a database.  There will be 1 main JSON DB file, containing all registered users with their passwords. Every user has a unique username and a password. The username will be a unique key for the DB and for the password there are no restrictions for the purpose of the app. And only registering a new user will be supported as a functionality (so user can not change his username or password or delete account). The username and the password are strings.  After a user is being logged, he can create its own notes. Each note has its unique name and its content and they will be strings. A user is allowed to see all his notes and their content, to add new notes, to delete old notes and to change their content.  • Anonymous *User* – can only view the information page, register or log-in into the system  • *Logged-in User* – can only view, create, edit, or delete it’s own notes. |
|  |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Register** | *Anonymous User* can register in the system by providing unique username and choosing password. If the username doesn’t exist in the DB, the new user will be registered successfully. | *Anonymous User* |
| * 1. **Login** | *User* can try to login with a username and a password. If they are correct the user is going to be logged in the system, so he can see/add/edit/delete his own notes.  Otherwise, if a username and password doesn’t match from the registered user, the Anonymous user can quit the application or try to register its new profile. | *Anonymous User, Registered User* |
| * 1. **Change content** | *User who is successfully logged-in can view/add/delete/change its notes.* | *Logged-in User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents the introductory information for the purpose of the system as well as detailed instructions on how to start using it. Prominently offers ability to register or login. | / |
| * 1. **User Registration** | Presents a view allowing the *Anonymous Users* to register. | */register* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **User Data** | Provides ability to view and edit personal *notes.* | */username* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *OKTS* 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 described 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 *OKTS,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Student Groups** | GET *Student Groups*, and POST new *Student Group* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request), according to *User's Role* and identity security restrictions. | */api/groups* |