| | |
|---|---|
| Assignment 1 | Project Summary |
| Course | Fullstack Application Development with Node.js + Express.js + React.js - 2020 |

| Project author | | |
|---|---|---|
| № | Pseudonym | Face-to-face/ online |
| 1 | dragon4o | face-to-face |
| 2 | Evlogi | face-to-face |
| 3 | ese | face-to-face |

| | |
|---|---|
| Project name | Fintooth financial management website |

## 1. Short project description (Business needs and system features)

Keep track of your finances on personal, family or group level. Fintooth provides statistical and graphical data to help you analyse your incomes, expenses and more.

The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies. Each view will have a distinct URL, and the routing between pages will be done client side using ***React Router***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. The data will be stored in a NoSQL database (MongoDB). The main user roles are:

- *User* – can access the basic functionality of the application: store income/expense/ transfer history, participate in a family group with another user sharing a common budget, view graphical representations of his financial status and history.
- *Premium User* (extends *User*) – will have additional charts, ability to create and participate in larger groups where expenses can be proposed and consequently voted on.
- *Administrator* (extends *User*) – can manage (create, edit user data and delete) all *Registered Users* as well as user groups.

| 2. Main Use Cases / Scenarios | | |
|---|---|---|
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| **1. Register** | *Visitor* 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 *User* role. There is an additional option for becoming a premium user.<br><br>*Administrator* can register new by entering *User Data*. | *User* |
| **2. Add/Edit/Delete activity** | Basic activities include: Adding, editing and deleting expenses, incomes and transfers. | *User* |
| **3. Show activities over calendar** | The User can get detailed information about his activities over a certain period of time. | *User* |
| **4. Manage Users** | *Administrator* can browse and filter users based on different criteria: first and last name, email, Role.<br><br>*Administrator* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC) or delete.<br><br>*Administrator* can create a new user using *Register UC*. | *Administrator* |
| **5. Manage Groups** | *Administrator* can browse and filter *Groups* based on different criteria: name of *Group,* creation date, etc.<br><br>*Administrator* can choose a *Group* to manage, and can add or remove *Users* to the *Group,* or delete it if necessary. | *Administrator* |
| **6. Create/Modify Group** | Users can create a group, add other users to the group and remove current members. | *User* |
| **7. Visualise data** | Provides charts and diagrams for analysis of financial data of a user or a group. | *User* |
| **8. Propose expense voting for groups** | Premium users can vote on custom expenses proposed by other Premium users from the group. | *Premium User* |

| | | |
|---|---|---|
| **9. Exquisite visualisation for Premium Users** | Premium users will receive additional and customisable information on their financial habits. | *Premium User* |

| 3. Main Views (SPA Frontend) | | |
|---|---|---|
| **View name** | **Brief Descriptions** | **URI** |
| **1. Home** | Provides information about the activities, with the options to search, filter, sort, edit, add and delete. If there is no current user you'll be presented to the Register view. | / |
| **2. Group View** | Serves the purpose of a homepage to a selected group. Propose voting on expense. | */groups* |
| **3. Manage Group** | Allows to add and remove users from a group | */groups/ manage* |
| **4. Family Pack** | Serves the purpose of a homepage to your family with limited functionality of a group. | */family* |
| **5. User Registration** | Presents a view allowing the visitor to register in Fintooth, as well as the option to become premium. If you have an account you can reach the login view. | */register* |
| **6. Login** | Presents a view allowing the users to login. | */login* |
| **7. Edit User Info** | Presents ability to view and edit personal *User Data* and the option to become a premium user. | */settings* |
| **8. Calendar** | The User gets detailed information about his activities over a certain period of time. This applies to User, Family and Group. | */calendar* |
| **9. Charts and Stats** | The user is presented with graphical, analytical stats on his/hers financial habits. This applies to User, Family and Group. | */charts* |
| **10. Users** | Presents ability to manage (CRUD) Users( available for *Administrators* only, as described in UCs). | */user-manager* |
| **11. About** | Presents information about the *Fintooth* project and his owners. | */about* |

| 4. API Resources (Node.js Backend) | | |
|---|---|---|
| **View name** | **Brief Descriptions** | **URI** |
| **1. Users** | GET *User Data* for all users, available only for *Administrators*. POST new *User Data* available for *Administrators* and unregistered users. | */api/users* |
| **2. User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions described in Use Cases. | */api/users/{userId}* |
| **3. Login** | POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests. | */api/login* |
| **4. Logout** | POST a logout request for ending the active session and invalidating the issued *Security Token.* | */api/logout* |
| **5. Groups** | GET *Groups*, and POST new *Group* (Id is auto-filled). Normal *User* can only participate in a Group with 2 people (Family). | */api/groups* |
| **6. Group** | GET, PUT, DELETE Group with specified *groupId.* | */api/groups/{groupId}* |
| **7. Activity Logs** | GET Activity logs, and POST new Activity log (activityId is auto-filled). | */api/activity_logs* |
| **8. Activity log** | GET, PUT, DELETE Activity log for *User/ Group* with specified *activityId*. | */api/activity_logs/ {activityId}* |
| **9. Votes** | GET Votes, and POST new Vote (*voteId* is auto-filled). | */api/votes* |
| **10. Vote** | GET, PUT, DELETE Vote for specified voteId | */api/votes/{voteId}* |

| | | |
|---|---|---|
| **11. Results** | GET Results, and POST new Result (references voteId) | */api/votes/{voteId}/results* |
| **12. Result** | GET, PUT, DELETE Result (boolean) for specified voteId and *userId* | */api/votes/{voteId}/results/{resultId}* |