|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2019 | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Simeon Aleksandrov | | face-to-face |

|  |  |
| --- | --- |
| Project name | Stay Tuna |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| *Stay Tuna* enables its users to experience day-to-day micro-learning. With the power of web scraping, *Stay Tuna* delivers relatable pieces of information such as articles, video lessons etc. based on user’s preferences. It allows users to register, pick topics of interest and receive on demand or on daily basis useful information. The system will be developed as a React Native application in order to leverage the capabilities of native push notifications. It will be ported for web view and the backend will be built using Node.js + Expess.  The main user roles are:   * Unregistered user - can only view landing page and register * Registered user - can pick topics, receive information on daily basis or on demad, view history of recommendations, give feedback for relevance, receive push notifications on mobile or email. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| **2.1. Pick prefered topics** | The user can Add/Update/Delete topics of intererest which will be used for delivery of information | User |
| * 1. **Request learning recommendation** | The user can request a new piece of information on demand, or rather set time interval at which they will be prompted with a new piece of information | *User* |
| * 1. **Browse learninng recommendations** | *The user will be able to browse previous recommendations.* | *User* |
| * 1. **Give feedback** | After reviewing new recommendation, the user can give feedback in order to evaluate relevance of recommendation and improve web scraping service performance. | *User* |
| * 1. **User can enable/disable push notifications** | In the mobile version of the application, user can enable/disable push notifications on their device. | *User* |
| * 1. **User can enable/disable email notifications** | In the web version, user can enable/disable email notifications. | *User* |
| * 1. **Share recommendation with a friend** | *Low priority.* User can share a recommendation with a friend | *User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Landing page. Prompts users to reigster. Registered users can view previous recommendations or request new. | / |
| * 1. **Feedback** | User can submit a creative feedback form for each reviewed recommendation. | */feedback* |
| * 1. **Settings** | User interface for managing prefernces, notifications and personal details. | */personal* |
| * 1. **Login** | Simple login form | */login* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js 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). | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*. | */api/users/{userId}* |
| * 1. **Recommendations** | GET recommendations history, POST new recommendation for web scraping service. PUT for feedback for each recommendation | */api/recommendations* |
| * 1. **Preferences** | GET, PUT, DELETE *Preferences Data and POST for initial request.* | */api/preferences* |

|  |  |  |
| --- | --- | --- |
| 1. High Level Design | | |
|  | | |
|
|
|
|

|  |  |  |
| --- | --- | --- |
| 1. Used techonologies | | |
| * React-Native-Web * React-Native-Elements * Expo * Axios * Redux * React Navigation * Webpack * Node.js + Express * MongoDB * Mongoose * JWT * Puppeteer | | |
|
|
|
|