Sofia University “St. Kliment Ohridski”

Faculty of Mathematics and Informatics

WhatToEat

Date: 03.07.2024г. **Author: Kirilka Doncheva, FN: 62289**

1. Short project description

In the fast-paced world we live in a lot of us can hardly make time to plan ahead what our next meal will be and what ingredients we need to add to our shopping lists in order to cook that meal. And ordering food for every meal is not very smart money-wise. And this is the issue the WhatToCook application aims to solve - it is a recipe sharing application allowing registered users to create meal recipes, create daily meal plans and shopping lists. The system will be developed as a Single Page Application (SPA) using Angular as front-end, and Node.js + Express as backend technologies.

2. User Roles

The main user roles in **WhatToEat** are **anonymous user**, **registered user** and **administrator**.

• **Anonymous User –** can only browse recipes.

• **Registered User –** can browse recipes, create recipes, create daily meal plans and shopping lists.

• **Administrator (extends Registered User) –** can manage (create, edit user data and delete) all Registered Users, as well as all Meal Recipes.

3. Functionality / Scenarios/

WhatToEat has the following main use cases:

- **Browse information**: The User can browse the information views (Home, Meal Recipes, About) in WhatToCook.

- **Registration**: Anonymous User can register in the system by providing a valid e-mail address, first and last name, and choosing a password. By default, all new registered users have Registered User role

- **Change User Data**: Registered User can view and edit their own personal User Data (first and last name, email, profile image and username). Administrator can view and edit User Data of all Users and change their role to Administrator.

- **Manage Users**:Administrator can manage all created Users. Administrator can also create, edit and delete recipes from the system.

- **Registered User or Administrator** can create, edit and delete collection from movies.

- **Administrator** user can create new recipes, edit and delete existing recipes. **Registered** user can create new recipes, edit and delete their own recipes.

- **Registered user** and **administrator user** can rate recipes by giving them points in the range of 1-5. They can only rate recipes which are not created by them.

- **All Users** can filter recipes based on rating, time to prepare, number of ingredients, level of difficulty

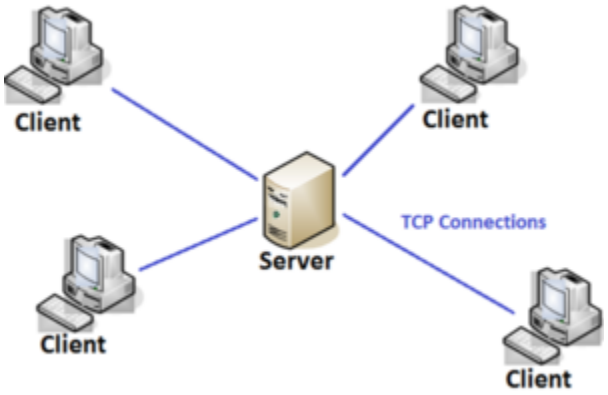
- **Registered user** and **administrator user** can create new meal plan for each day, edit and delete their own meal plans.

- **Registered user** and **administrator** user can generate new shopping lists, edit and delete their own shopping lists.

4. Architecture model and used technologies

WhatToEat uses **client-server architecture**. **Client-server architecture** is an architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer).

Client-server architecture



The system is developed as **a *Single Page Application (SPA)*** using ***Angular*** as front-end, **Angular Material** and **Ng Bootstrap** for styling, 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 ***Angular Router***. The backend is implemented as a ***REST/JSON API*** using JSON data serialization.

5. Frontend implementation

WhatToEat frontend is implemented from views, each of them has distinct URL.
The main views are:

- /home - Presents the introductory information for the purpose of the system as well as detailed instructions on how to start using it. Offers ability to register.
- /register - User Registration Presents a view allowing the Anonymous Users to register in WhatToCook.
- /login - Login Presents a view allowing the users to login.
- /myProfile - My Profile Presents ability to view and edit personal User Data.
- /recipes - Recipes Presents recipes available in the system. Offers abilities to browse, filter, choose, read recipes, as defined by the user's Role (for all users).

- /recipes/{recipeId} - Recipe view Presents a specific recipe view (for all users). Allows to perform CRUD operations on this recipe (for Administrators and the Registered User who is the owner of the recipe)
- /recipes/review - Rate recipe Presents ability to rate a recipe (for all Registered Users except for the recipe owner)
- /myMealPlans - My Meal Plans Presents ability to manage Meal Plans (for Administrators and Registered Users)
- /myShoppingLists - My Shopping Lists Presents ability to manage Shopping lists (for Administrators and Registered Users)
- /users - Users Presents ability to manage (CRUD) Users and their User Data (available for Administrators only).
- /about- Presents information about the WhatToCook project and his owner.

6. Backend implementation (backend REST API endpoints)

The **WhatToEat** backend is implemented from Rest API endpoints, each of them has distinct URL (endpoint).

The main **REST API endpoints** are:

- /api/users - GET User Data for all users, and POST new User Data (Id is auto-filled by WhatToCook and modified entity is returned as result from POST request)
- /api/users/{userId} - User GET, PUT, DELETE User Data for User with specified userId, according to restrictions described in UCs.
- /api/login - POST User Credentials (e-mail address and password) and receive a valid Security Token to use in subsequent API requests.
- /api/logout - POST a logout request for ending the active session with the system, and invalidating the issued Security Token.
- /api/recipes - GET Recipes, and POST new recipe (Id is auto-filled by WhatToCook and modified entity is returned as result from POST request), according to User's Role and identity security restrictions.
- /api/recipes/{recipeId} - GET, PUT, DELETE Recipe (according to User's Role and identity)
- /api/recipes/{recipeId}/review - POST Review Recipe, GET reviews (according to User's Role and identity)

- /api/shoppingLists - GET Shopping Lists, and POST new shopping list (Id is auto-filled by WhatToCook and modified entity is returned as result from POST request), according to User's Role and identity security restrictions.
- /api/shoppingLists/{listId} - List GET, PUT, DELETE Shopping List (according to User's Role and identity)
- /api/mealPlans - Meal Plans GET Meal Plans, and POST new Meal Plan (Id is auto-filled by WhatToCook and modified entity is returned as result from POST request), according to User's Role and identity security restrictions.
- /api/mealPlans/{planId} - GET, PUT, DELETE Meal Plan (according to User's Role and identity)

7. Set up WhatToEat

Execute following commands in the order they are given:

- **Go to project folder**
- **Open new terminal** there and install node modules with **`npm install`** or **`yarn`**
- Run **`ng serve`** in the terminal
- **Open second terminal**, go to 'server' folder in project (type **`cd ./backend`)**, run **`npm install`** and then run **`npm start`**

References

1. https://angular.io/docs Angular Docs
2. https://nodejs.org/dist/latest-v16.x/docs/api/ Node.js Docs
3. https://expressjs.com/ Express web framework
4. https://material.angular.io/ Angular Material

5. https://ng-bootstrap.github.io/#/home Ng bootstrap
6. https://cio-wiki.org/wiki/Client_Server_Architecture Client server architecture

GitHub repository

https://github.com/kirilkadoncheva/WhatToEat