

Project Report - Group 4

1. Introduction

Food has always been one of the basic necessities to bring people together in a social setting. Sharing meals with someone else is a common way of building connections and creating memories together. However choosing what restaurant to eat at can be a challenging task. That is where our project, FoodMatcher, comes in. It is a web-based application designed to help a group of people to choose what restaurant to eat at based on their food preferences.

Users are able to register an account and create or join groups. Users will then be given pictures from restaurants which they can like or dislike. A user can then go into a “match” page where they can enter a group name and be shown the most liked restaurant in that particular group.

FoodMatcher aims to provide a fun and interactive way for people to choose restaurants with friends who share the same passion for food. Whether the user is looking for a quick breakfast or a luxury dinner, FoodMatcher can aid them in this search.

In this report, the application FoodMatcher will be discussed. The design and development process of the application and the use of technology will be analyzed. The purpose of the report is to give the reader a clear view of how the application's development process has looked and also how it works in a more technical manner.

Link to the github repo of the project: <https://github.com/abbasfaizi/Web-Project-NodeJs>

2. A list of use cases

- A user can register an account
- A user can login with their account using their credentials.
- A user can like a particular restaurant.
- A user can dislike a particular restaurant.
- A user can create a group.
- A user can join a group.
- A user can see the most liked restaurant in a group.
- A user can switch between the different pages in the application.

3. User Manual

Firstly install Node.js on your machine. You can download it from the official website: <https://nodejs.org/en/download/>. Then you will have to clone or download the repository in your local machine. The easiest way to do this is to clone it using Visual Studio Code. Then navigate to the root folder of the project in the terminal or command prompt. You will then need to run the command “npm install” to install all required dependencies for the application. Then navigate to the client folder in the command prompt window and run the command “npm start”. This will launch the application in your default browser. Then navigate to the “server” folder in the command prompt window and run the command “npm run dev”. This will start the server side of the application in development mode.

Head to <http://localhost:3000> to view it in the browser.

Below is a step by step guide of how the application works with pictures as illustration.



Please Login

Don't have an account? [Create account](#)

© 2023 FoodMatcher.se

This is the login page. Here you can either log in with an existing account, or if you do not have an account you can press the “create account” link to go to the “create account” page. If you already have an account, you can input your credentials and press the login button. If the credentials match an account in the database, you should be logged in and redirected to the “main page”.



Register now and get
started

Register

Already have an account? [Login](#)

Main page

© 2023 FoodMatcher.se

Here you can register an account. You need to enter a username, password and then repeat the password. If all input is valid, after pressing “register” it should register your account to our database and redirect you to the login page. You can now login to your account.



Swipe your favorite food



Like

Dislike


Create Group

Join Group


Matches

© 2023 FoodMatcher.se

This is the main page. In this page you can like and dislike restaurants. You also have buttons to go to the “create group” page, “join group” page and the “matches” page.




The picture above is an illustration of the page where you can create a group within the application. To do this you will need to specify a group name, password and a location. Which are then saved in a database. Based on the specified location the application will get nearby restaurants and add them to the database so if a user is a member of the group they can like or dislike them via the main page.



The picture above is an illustration of a page where you can join an existing group. To achieve this you will need the group name and the password of that group to be able to join. You can then like or dislike restaurants that belong to that group via the main page. We remove duplicate restaurants from a user so they do not need to like/dislike restaurants

multiple times. This feature leads to if you already have liked/disliked restaurants in another group, if you join a new group with the same restaurants, the restaurants will not reappear.

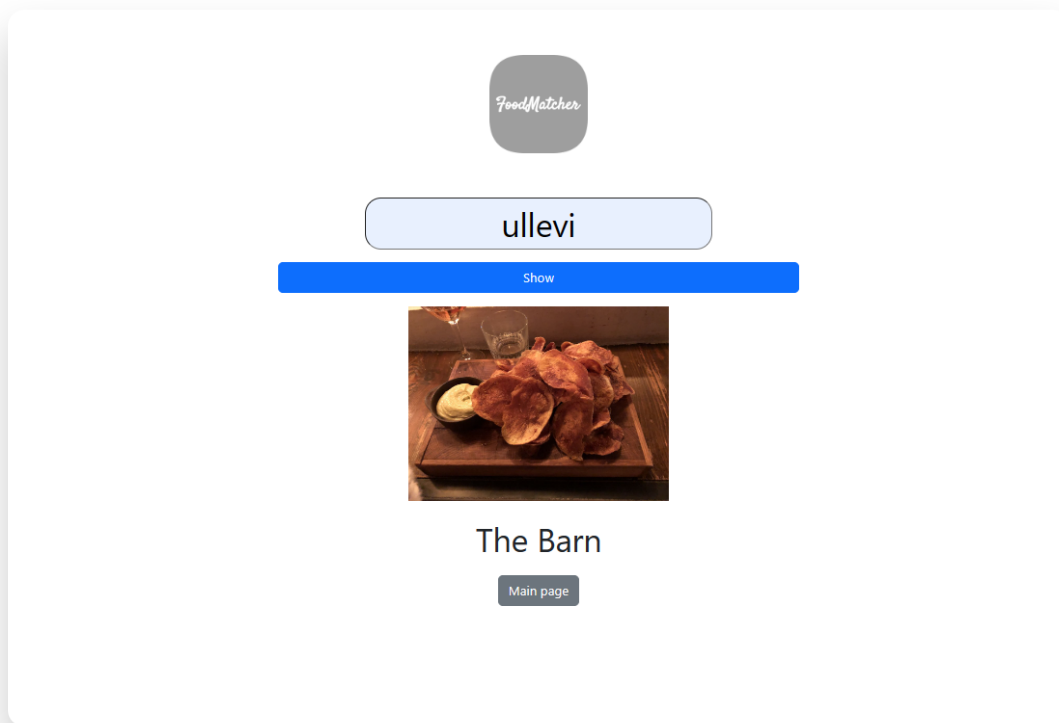


Group name

Show

Main page

Here is the matches page, where you can see the most liked dish in the given group. Firstly you will need to input the group name in which you want to see the dish that is most liked for that given group. After pressing the “show” button, a image and the name of the most liked restaurant in that group will be displayed on the page.



The figure above shows a dish that has been liked the most in a given group that is called “ullevi”.

4. Design

4.1 Technical Construction

In the project there have been a lot of libraries and technologies used. ReactJS is a popular javascript library used to build user interfaces. In FoodMatcher, ReactJS was used to build the front-end of the application, enabling users to interact with the site and any given functionality.

NodeJS was also used in the project. NodeJS is a popular open-source server-side runtime environment that allows the creation of scalable and high-performance applications. In FoodMatcher, NodeJS was used to create the back-end of the web application. It managed most of the server-side logic and the connection with the database.

ExpressJS is a minimal and flexible NodeJS web application framework. The framework simplifies the process of building the application and its APIs used. In the project ExpressJS was used to build the server-side logic.

Typescript is an open-source programming language that is a superset of Javascript. Typescript provides features such as writing interfaces, classes and creating maintainable and scalable code. In the project typescript was used to write both the server-side and front-end code.

CSS is a stylesheet language. In the project we have used it to style the pages and describe how different elements should be displayed on the screen.

Bootstrap is a front-end framework used to create responsive applications. It helps with making the website display the content right on different devices or screen sizes.

Axios is a popular HTTP client library that allows the application to make HTTP requests to the browser and to NodeJS. Axios was used in the project to make calls between the front-end and the back-end of the application. It provided the communication in the form of HTTP requests.

CORS (Cross-origin Resource Sharing) allows web applications to access resources from different domains.

Mongoose is a library for MongoDB and NodeJS. It provides access to be able to connect to an external database and provide features and operations that the application can perform with the database. It is used in the application to handle all database related operations like inserting, updating, creating and testing.

Jest is a popular javascript testing framework used to test code written for React applications. Jest provides a flexible and powerful testing environment which allows developers to test their code in a seamless way. In the project Jest was used to write and run tests for React components and the back-end.

4.2 API Specification

group.router.ts contains three endpoints:

GET /group/:group: Gets the restaurant object with the highest total "likes" from the specified group's members, if the user is a member of the group and authenticated. Returns 404 if no match is found, 401 if the user is not a member of the group or not logged in.

POST /group/create: Creates a new group with provided information, and connects with Yelp API to retrieve restaurant information based on provided location. The authenticated user creating the group becomes the group owner. Returns 400 if location is invalid, and 409 if group ID already exists.

POST /group/join: Adds the authenticated user to the specified group, if the group ID exists and password is correct. Returns 401 if the group ID does not exist or the password is incorrect, and 409 if the user is already a member of the group or not logged in.

restaurant.router.ts contains two endpoints:

GET /api/restaurants: Gets all restaurants for the logged-in user. Returns an array of restaurant objects, each containing id, name, and imageUrl fields. Returns 401 if the user is not logged in.

GET /api/restaurants/:rid: Gets details of a specific restaurant with the provided restaurant ID. Returns restaurantDetails object containing name, imageUrl, location, phone, rating, and review_count fields. Returns 400 if request parameters are invalid, and 404 if requested restaurant is not found.

user.router.ts contains three endpoints:

GET /likes: Gets an array of restaurants that the logged-in user has liked. Returns 401 if user is not logged in, 400 if user ID is null, and 404 if user with given ID is not registered.

GET /dislikes: Gets an array of restaurants that the logged-in user has disliked. Returns 401 if user is not logged in, 400 if user ID is null, and 404 if user with given ID is not registered.

POST /register: Registers a new user with provided username and password. Returns 400 if the request body is invalid, and 409 if the user with provided username already exists.

5. Responsibilities

In the course of the project all group members have worked in various areas. There have been a lot of peer programming involved which is why some of the responsibilities and tasks will overlap between group members.

Abbas Faizi: Mostly front-end development of the application. Was responsible for many of the interface changes to the application. Also did some testing for the application as a whole.

Zakariya Omar: Mostly back-end development of the application. Was core-responsible for the interaction with the database. Also worked on interactions with API and what calls should be implemented to what endpoints.

Love Rymo: Both front-end and back-end development. Had responsibility for many key functionalities between the front-end and the back-end. Brought great value to the development of the application in the form of connecting front-end and back-end calls.

Mustafa Bawi: Mostly front-end development. Worked mostly with the interface of the application and any changes made there. Also created documentation for the application and design. Also was core-responsible for the project report and the structure of the report. Brought great value to the development in the form of front-end development and design choices.

Fadi Abunaj: Both front-end and back-end development. Did some design choices and modifications to the front-end of the application. Also worked on connecting front-end to back-end calls. Brought great value in assisting the development of the application and taking it to a greater extent.