Advanced Web Applications
Project work

Name: Aleksi Haapalainen
Email: aleksi.haapalainen@student.lut.fi

1. Installation

To install and run this application on your own machine, you need to follow these instructions.

1.1 Prerequisite

- MongoDB server v7.0 or newer running on port 127.0.0.1:27017
- Node v18.18.0 or newer

1.2 Installation

- Clone the repository from GitHub.
- Open a terminal (preferably git bash).
- In project root run “npm run install” to install all dependencies.
- Run “npm run dev-server” to start the backend.
- Open another terminal at the project root and run “npm run dev-client” to start the client.
- Go to http://localhost:5173 to access the application.

1.3 Usage and testing

- Application can now be used in a local environment.
- It is strongly recommended to register at least 5 different users to test the application properly.

1.4 Disclaimer

This project has been developed and tested using the versions and tools mentioned above. Older versions might work but there is no guarantee. Also, Mozilla Firefox is recommended as the browser choice, but other browsers might work also. Git Bash is highly recommended when installing the application, but other CLI tools might work.

2. Introduction

The project goal was to develop a tinder-like application, which would contain basic features such as registration, authentication, users to be able to see other users and match with them, and eventually chat with them.
For this project I decided to create an application which would be specifically for videogame players to find new gaming companies in their own preferred games. The application should have features to set nickname, gaming platform, favorite genres and set a description to give more details. Other than that, the project should follow tinder in features and functionalities.

3. Technical solutions

3.1 Backend

As Instructed, the backend is implemented using Node.js. For personal reasons I decided to code the whole project with TypeScript instead of JavaScript. Database solution of the project is implemented with MongoDB, I considered using PostgreSQL for this project, but given the size of the project I figured that MongoDB is enough for this project. In addition to MongoDB, I used mongoose node module to connect database to the server. To set up the server, I used Express to implement simple REST API.
In registration I used Bcrypt module to hash the password. Authentication is implemented using JSON web token, and passport to validate user's authentication. To validate user's input from client, I used Express-validator to validate email and password.

3.2 Frontend

I decided to implement frontend using React. React is one of the most popular front-end frameworks in the industry at the moment, so I figured it would be best choice for this implementation. The course teaches to use create-react-app to initialize react project, but since this has been deprecated a year ago, I decided to use Vite, which is much lighter weight than create-react-app.
For the UI I decided to use Material UI, since it has very good documentation, and a lot of functional features to use. Material UI components are also easy to modify and style.
For routing I used react-router-dom library to create easy and simple routing. For the swiping effect, I used react-tinder-card module to easily create swiping feature. For the chat implementation I used react-chat-elements library, because it offers simple components to create chat elements. For the pop up of a new match, I used reactjs-popup module. Lastly for the internationalization I used i18next module. It offers a simple way to detect language and change the language of the whole UI.
Overall React is a powerful tool to create frontend to the project, it can be a little more complex with states and hooks, but they are very effective. Also React is very extensible with many external modules and components.

4. Improvements

As this application is made with one man team and in a tight schedule, there certainly is a lot to improve and enhance. Testing has been very simple and quick. This application should be tested much more widely and therefore there might be some bugs in the current version. If there was more time, I'd create at least simple unit tests to test basic features. For testing purposes there should also be a function to fill the database with test users, to properly test the application.
Other than testing, I would have deployed this application to cloud environments such as Microsoft Azure, but again with the given time frame this can't be done. The installation guidelines are very inconvenient at the moment.

5. Features and grading

Feature	Description	Grading
Basic features	Node.js, Express, MongoDB, Registration, JWT Authentication, Like/dislike users, Update profile, Chat with matched users, Responsive design, Documentation.	25
TypeScript	Since the project is fully implemented with TypeScript and a lot of self-learning had to be made, it should be rewarded with a few points.	3
React	Frontend is completely made with React.	5
Swiping	User can swipe other users to like/dislike them.	2
Popup when new match is found*	When users match, the UI gives popup which has link directly to chat.	2
Translation of the UI	User interface can be translated either into English or Finnish.	2
Chat features	Messages have timestamps to tell when the message was sent. Chat listings show the latest message of the conversation without opening the chat. Chat is scrolled to the latest message when sent/received.	4
Password and email validation	Password and email are validated with express-validator in the backend.	2
Appearance and styling	A lot of effort has been used to make the application to look nice and to have a more unique idea than basic tinder.	3
Total		**48**

*Popup feature has minor bug. It does not trigger if the user matched is the last user in the card pile.