IBU | International Burch University

CEN 308 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

“Hanjaries” - memories sharing application

Students:

Hanjalic Haris

Sacic Mela

Proposed to:

Durmic Nermina, Assist. Prof. Dr.

Kovacevic Aldin, Teaching Assistant

Sarajevo, 2021.

TABLE OF CONTENTS

Introduction

1.1 About the project

‘Hanjaries’ is a web application made for sharing different memories of your life. Every user needs to register in order to create a new hanjary (in future text, ‘hanjary’ means ‘memory’).
If the user wants simpler registration, we have enabled Google’s OAuth 2.0, which means if you have an existing Google account, you can register through their services.

Our GitHub repo can be found here: https://github.com/hanjalicharis/SE-2.0

Front-end for our web application is deployed to the following link:
Back-end for our web application is deployed to the following link:

1.2 Project functionalities and screenshots

In the next few paragraphs, I will list main features of our application:

- Register new user with email and password
- Register new user using Google’s services
- Creating a new hanjary
- Updating the existing hanjary
- Deleting (only your own) hanjaries
- Liking hanjaries

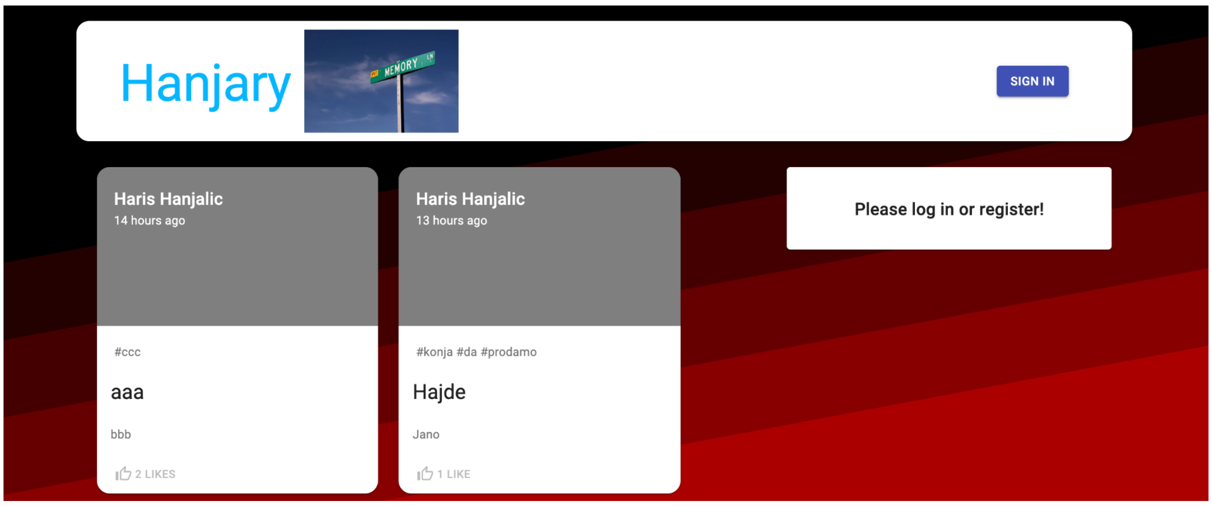

Figure 1: Main screen

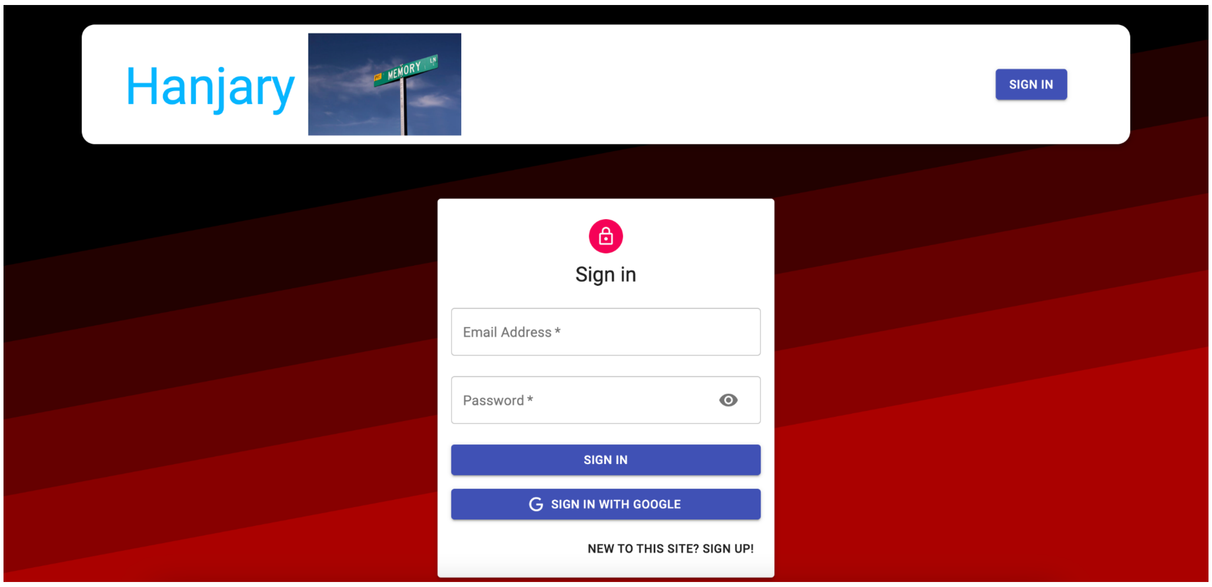

Figure 2: Login screen

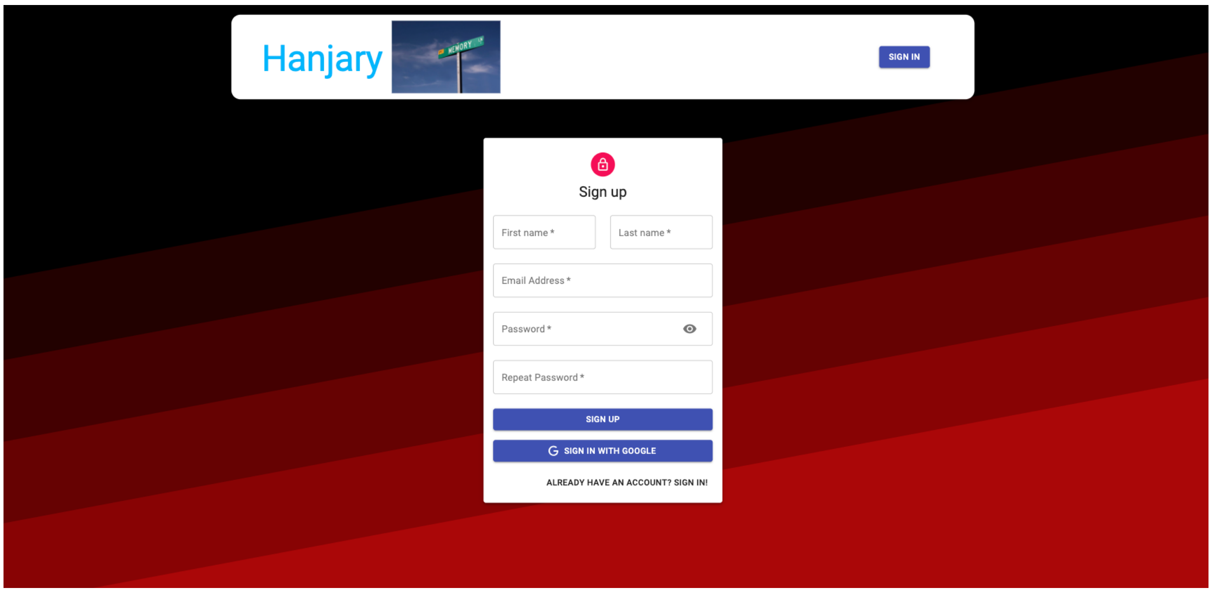

Figure 3: Sign-up screen

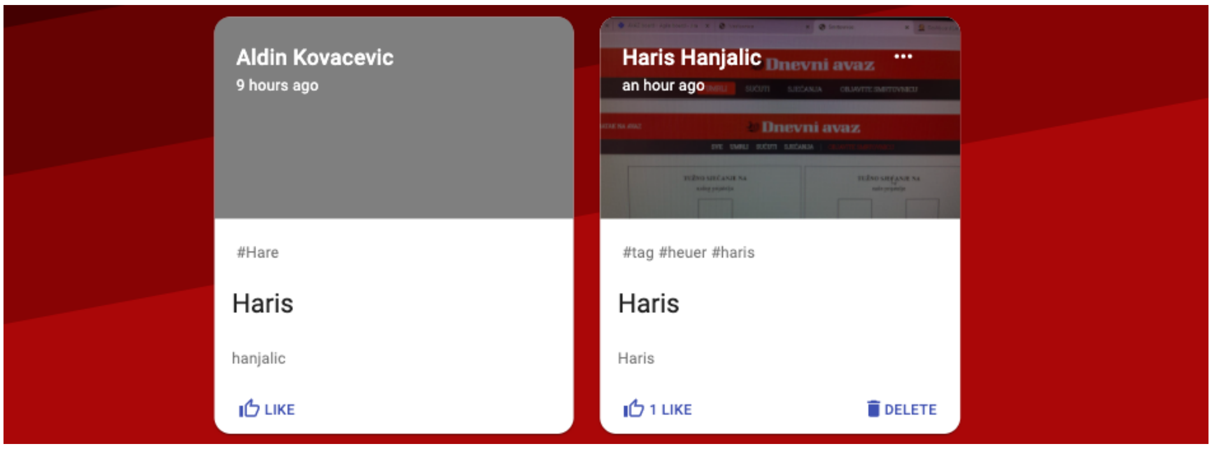

Figure 4: Editable and non-editable post

Project Structure

2.1. Technologies

Technologies we've used for this project were :

- ReactJS - for frontend
- Node.js - for backend
- MongoDB - as main database

We have used many third-party libraries since ReactJS is an open-source JS library. Also, we have followed ReactJS's own coding standards (which most of them are thoroughly explained at https://reactpatterns.com/), and structured folders in our project in order to maximize the efficiency and scalability of the code.

2.2 Database collections

Our database has 2 main collections where all the informations are stored and those are:

- postmessages
- users

Please take into consideration that if this was done in MySQL, there would have been 5-6 entities, including all the bridge tables, creations of foreign keys, etc.

2.3 Architectural Pattern

The architectural pattern we decided to use in our web application was 'layered architecture pattern'. Why? Well, because our system is divided into two folders, client side and server side. On the server side, we've made a hierarchy where we implement base functions to some core components which can be used in higher levels, along with constants that are used system-wide. This is the part we mainly learned from the 'Introduction to Web Programming' course we had earlier. We did this in order to maintain code scalability and add more features without modifying large chunks of code.

2.4 Design patterns

When we talk about design patterns, we’ve used almost all of the design patterns ReactJS offer, so let’s talk about them and where they were used:

2.4.1 Conditional Rendering

Conditional Rendering is a design pattern we use for our code when we want to render certain JSX code based on some state.

- **client/src/components/Authentication/Authentication.js** (line 73-79)
- **client/src/components/Authentication/Authentication.js** (line 85)
- **client/src/components/NavBar/NavBar.js** (line 48-57)
- **client/src/components/Posts/Post/Post.js** (line 63-68)
- **client/src/components/Posts/Post/Post.js** (line 43-50)

2.4.2 React Hooks

React Hooks are new additions to React which allow the developers to use ReactJS without classes. Hooks we’ve used are useEffect (Effect hook) and useState (State hook).

- **client/src/components/Authentication/Authentication.js** (line 19-21)
- **client/src/components/Form/Form.js** (line 13 and line 28-30)
- **client/src/components/Homepage/Homepage.js** (line 14 and line 17-19)
- **client/src/components/NavBar/NavBar.js** (line 24 and line 28-38

2.4.3 Style component

In almost every component, style.js was used as an external style-sheet to make our application look better and we have reduced the inline style as much as we could.

2.4.4 Expressions

As for the NavBar component, we’ve used the expressions, as can be seen in the code below, where we used the login information to be displayed in the NavBar when a user is signed in (if the user does not have the image, it uses the first letter of their name (<Avatar> div) as the profile photo):

```
<Avatar className={classes.purple} alt={user.result.name} src={user.result.imageUrl}>{user.result.name.charAt(0)}<
<Typography className={classes.userName} variant="h6">{user.result.name}</Typography>
```

Conclusion

As a conclusion, I want to say that we are satisfied with our application overall, and we are really happy that we’ve created a deployment-ready application which can be used to create and share memories with other people.

The most challenging part for us was the implementation of login and registration with JWT token, where we rethink our whole logic, but actually, we’ve just made a typo.

All in all, it was a real satisfaction creating this whole project, working on it, and it was a great experience for us to work as a team and implement everything we’ve learned from our lectures.

In the end, we would like to give special thanks to the assistant Aldin who helped us through the whole project whenever we encountered an error we couldn’t solve by ourselves.