Documentation for Project Work Dinder

Yuxin Du 000797135

Introduction:

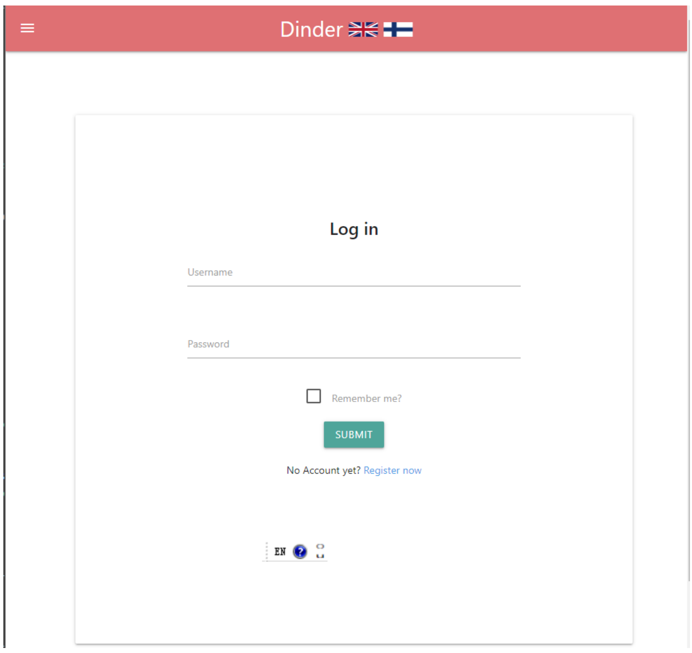
This project aims to be a mocked Tinder app, so I named it Dinder. It is a full-stack application that includes all the mandatory requirements as well as several stretch requirements. This document would first explain the tech stack used, the detailed functionalities, and instructions to run the application on your browser..

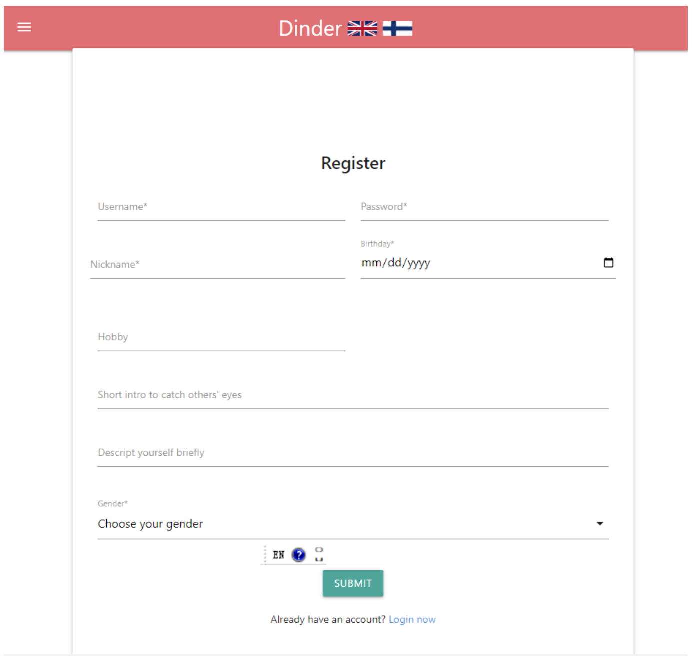
Tech stack:
The back end is created using Express.js with MongoDB. The front end is a React.js app. The authentication is done using JSON Web Token or (JWT) because JWT is more scalable compared to session-based counterparts. The JWT is stored using HttpOnly cookies to avoid risks such as Cross-site scripting since HttpOnly cookies cannot be accessed by JavaScript running on the front end. To make messaging synchronize in real-time, Web Socket is used. The app is available in both English and Finnish (Finnish text is translated from English using ChatGPT) and the translation is done with i18n.

Functionalities (With screenshots):

Mandatory functionalities:

Implementation of backend with Node.js	[done]
Utilization of database	[done]
Users must have an option to register and login	[done]



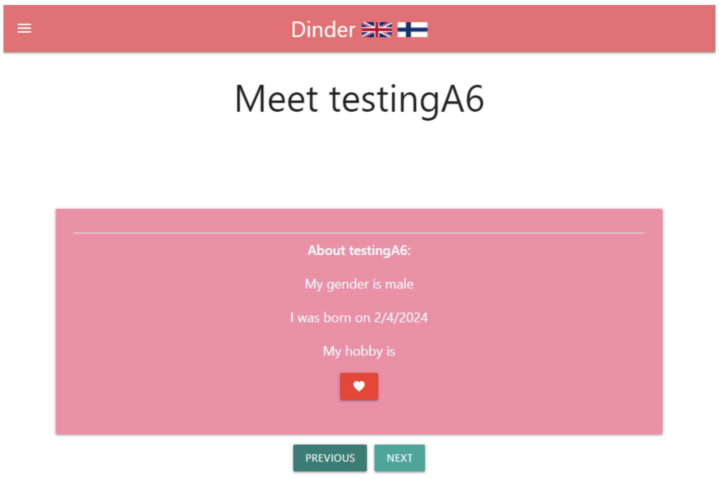


You can use JWT or session-based authorization [done]

Only authenticated users can post, comment (or vote) [done]

(Note, all routes except login and register can't be accessed without authentication, login and register cannot be accessed after logging in)

Authenticated users can see other users and like / dislike them [done]



Meet testingA6

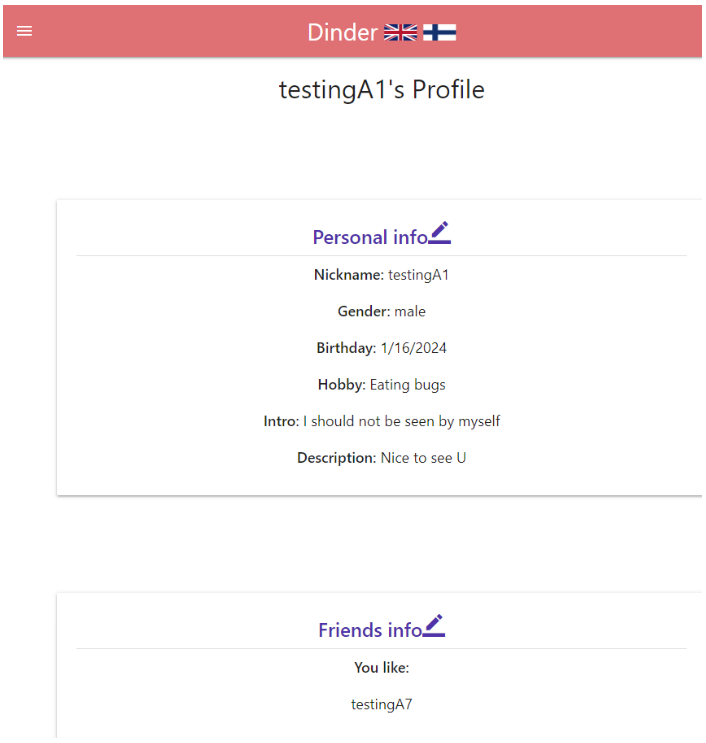
About testingA6:

My gender is male

I was born on 2/4/2024

My hobby is

Authenticated users can update their profile [done]



testingA1's Profile

Personal info

Nickname: testingA1

Gender: male

Birthday: 1/16/2024

Hobby: Eating bugs

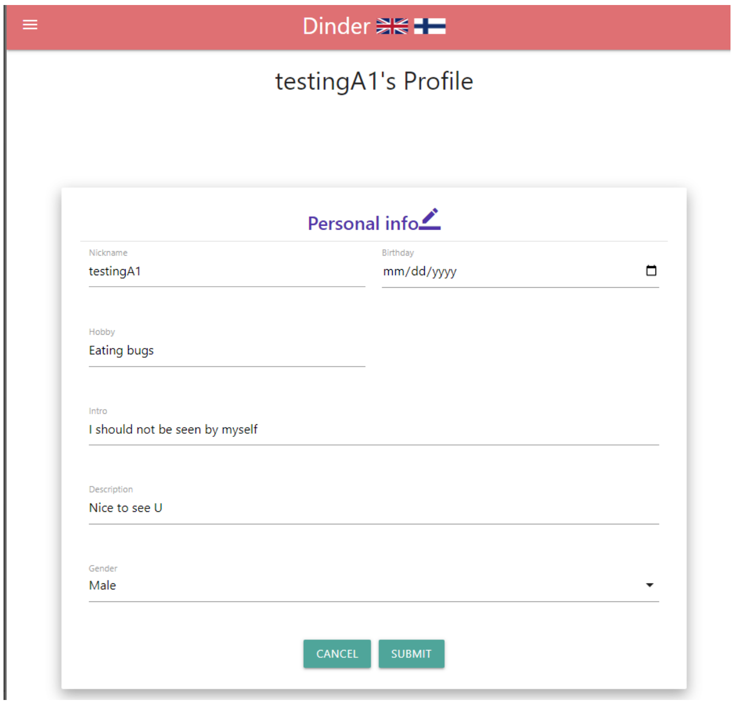
Intro: I should not be seen by myself

Description: Nice to see U

Friends info

You like:

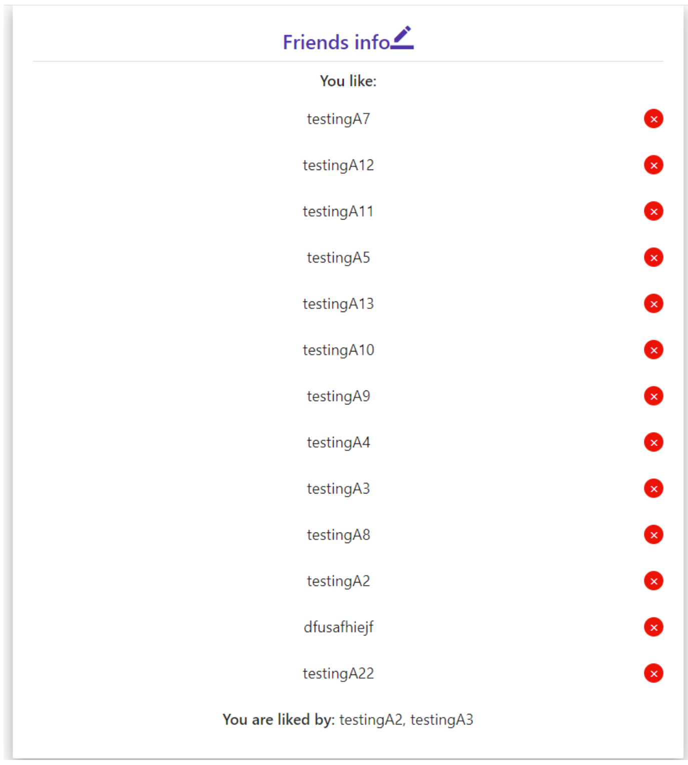
testingA7



testingA1's Profile

Personal info

Nickname: testingA1	Birthday: mm/dd/yyyy
Hobby: Eating bugs	
Intro: I should not be seen by myself	
Description: Nice to see U	
Gender: Male	



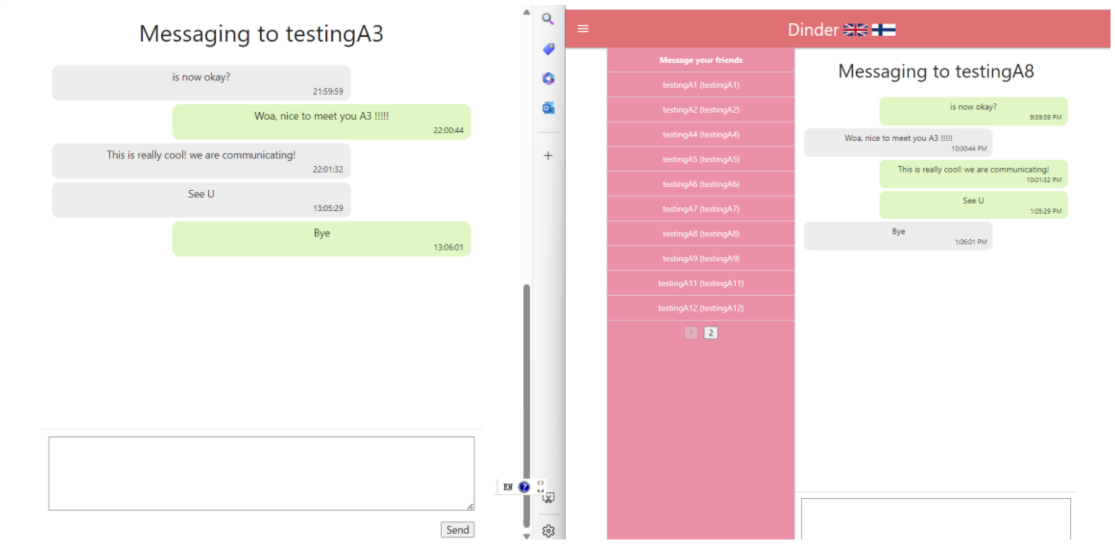
Friends info

You like:

testingA7
testingA12
testingA11
testingA5
testingA13
testingA10
testingA9
testingA4
testingA3
testingA8
testingA2
dfusafhiejf
testingA22

You are liked by: testingA2, testingA3

Authenticated users can chat with users that have liked both ways [done]


Messaging to testingA3

is now okay?
21:59:59

Woa, nice to meet you A3 !!!!!
22:00:44

This is really cool! we are communicating!
22:01:32

See U
13:05:29

Bye
13:06:01

Dinder

Messaging to testingA8

is now okay?
9:59:59 PM

Woa, nice to meet you A3 !!!!!
10:00:44 PM

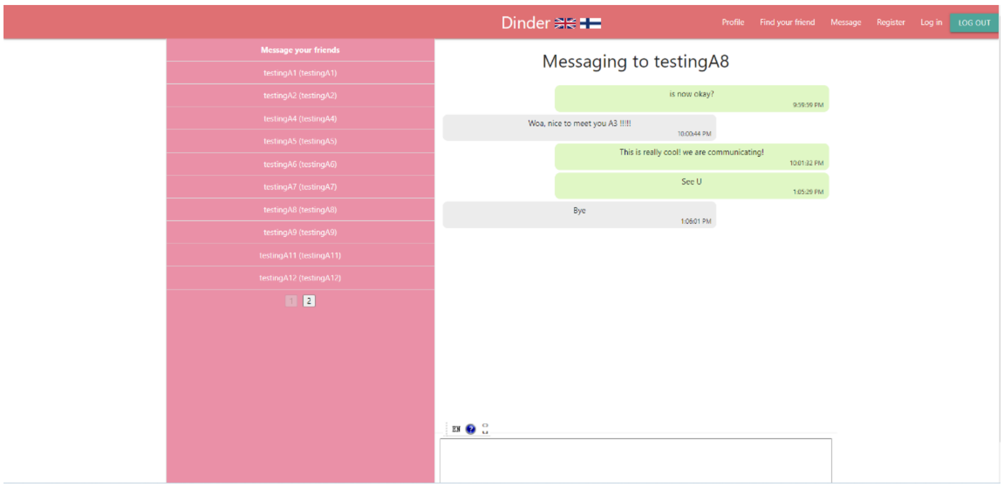
This is really cool! we are communicating!
10:01:32 PM

See U
1:05:29 PM

Bye
1:06:01 PM

The app needs to be usable with mobile devices and desktop browsers [done]
The UI follows Materialize theme and is fully responsive.

Full screen view:


Dinder

Messaging to testingA8

is now okay?
9:59:59 PM

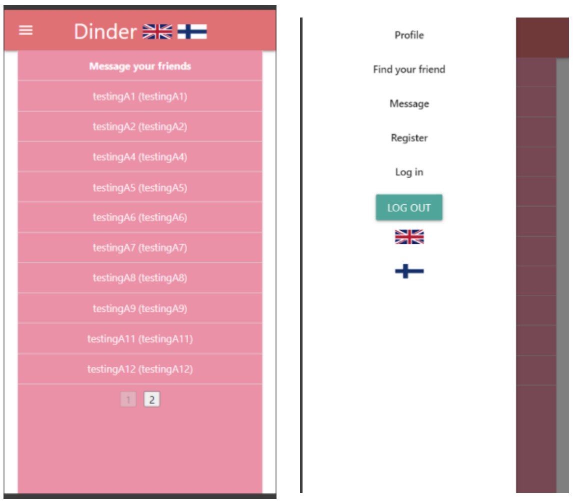
Woa, nice to meet you A3 !!!!!
10:00:44 PM

This is really cool! we are communicating!
10:01:32 PM

See U
1:05:29 PM

Bye
1:06:01 PM

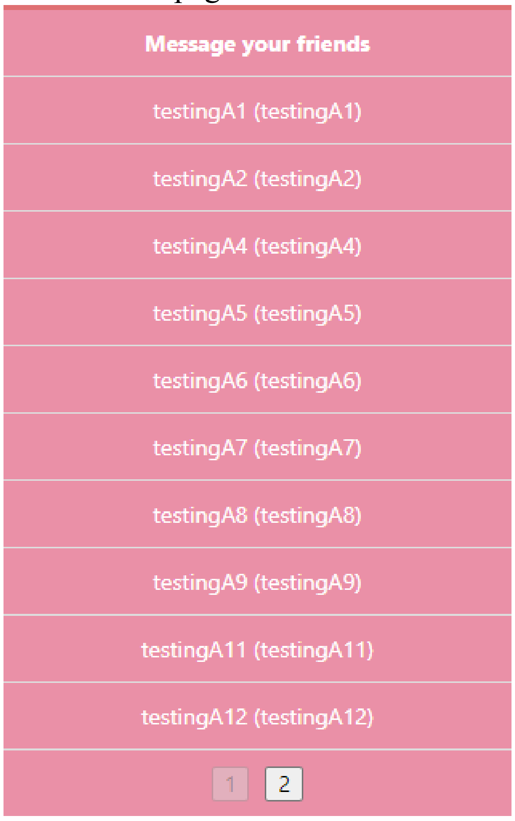
Mobile view:

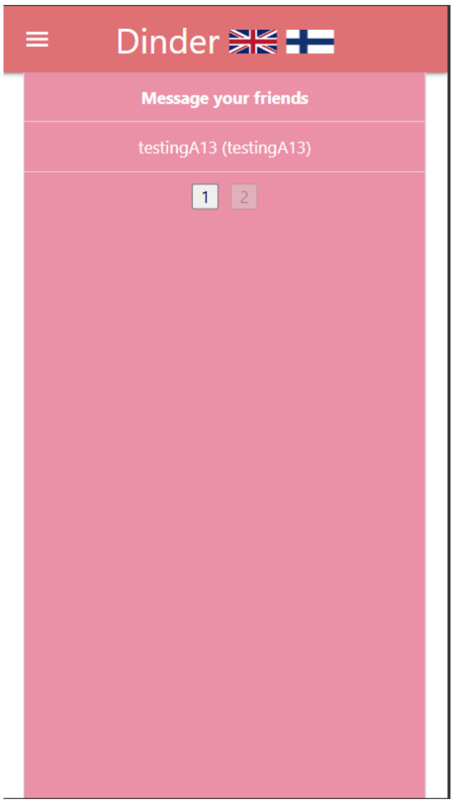


Additional features (listed):

Utilization of a frontside framework, such as React, but you can also use Angular, Vue or some other [done+5]

Use of a pager when there are more than 10 chats available [done+2]

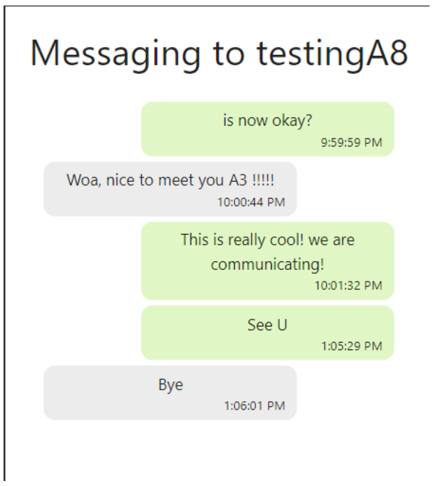

Message your friends
testingA1 (testingA1)
testingA2 (testingA2)
testingA4 (testingA4)
testingA5 (testingA5)
testingA6 (testingA6)
testingA7 (testingA7)
testingA8 (testingA8)
testingA9 (testingA9)
testingA11 (testingA11)
testingA12 (testingA12)



Message your friends
testingA13 (testingA13)

Pager is functioning well.

Last edited timestamp is stored and shown within chat [done +2]


Messaging to testingA8

is now okay? 9:59:59 PM

Woa, nice to meet you A3 !!!!! 10:00:44 PM

This is really cool! we are communicating! 10:01:32 PM

See U 1:05:29 PM

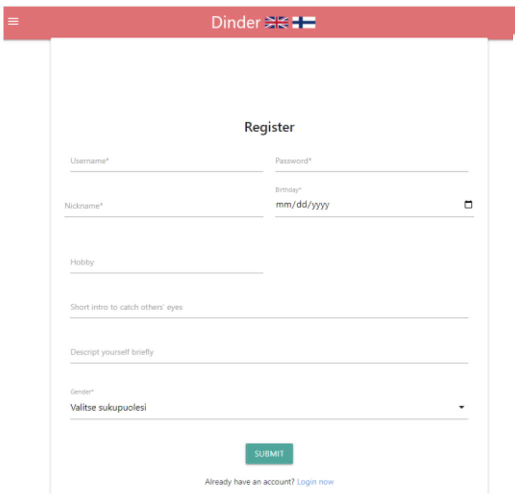
Bye 1:06:01 PM

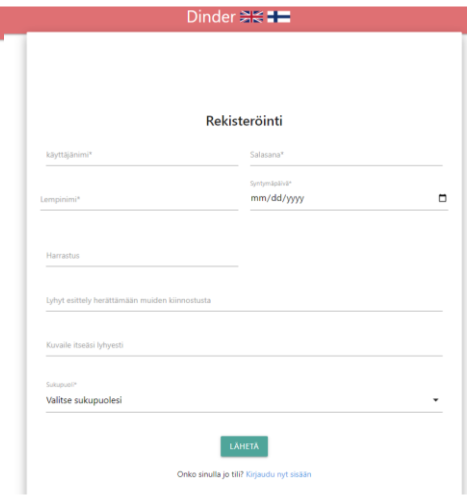
A timestamp is presented in each message.

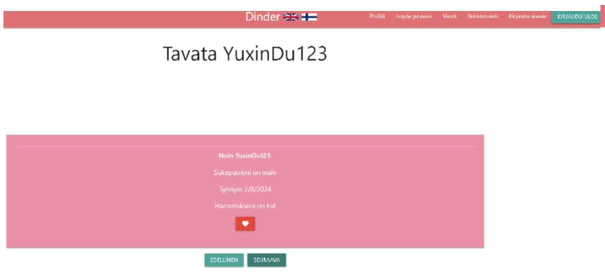
Translation of the whole UI in two or more languages [done+2]

The whole UI is translated into Finnish (translation is done by ChatGPT, I cannot guarantee the accuracy of the translation…) using i18n.

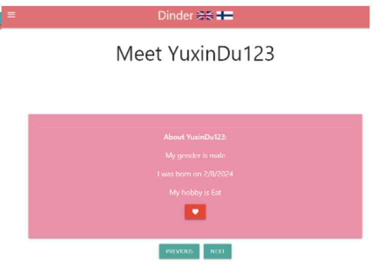
Project work: Dinder







Noin YuxinDu123:
Sukupuoleni on male
Synnyin 2/8/2024
Harrastukseni on Eat

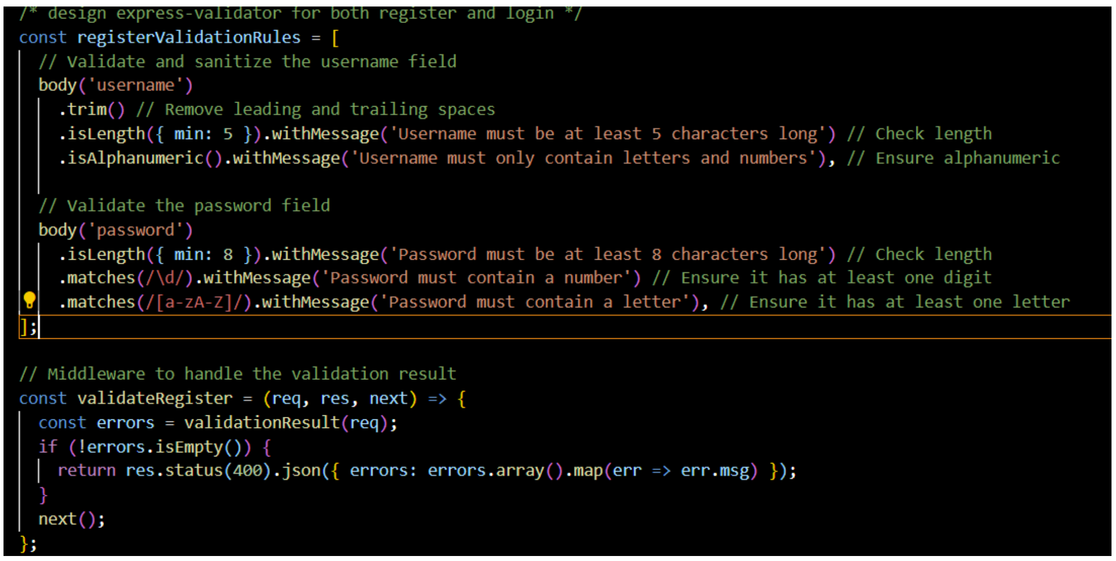


About YuxinDu123:
My gender is male
I was born on 2/8/2024
My hobby is Eat

User can click username and see user profile page where name, register date, (user picture) and user bio is listed [not sure +2?]
All the user info is directly displayed in profile page, user can see other user's info without the need of clicking the username. I am not sure whether this counts.

Additional features (self-added)

1. Nice looking Nav bar. The nav bar looks cool and is fully responsive. [done+2?]
2. Express validator for input validation. A quite comprehensive strategy is implemented to validate and sanitize username and password when doing registration and login to enhance security. [done+2?]



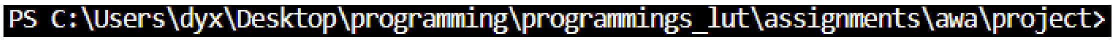
```
/* design express-validator for both register and login */
const registerValidationRules = [
  // Validate and sanitize the username field
  body('username')
    .trim() // Remove leading and trailing spaces
    .isLength({ min: 5 }).withMessage('Username must be at least 5 characters long') // Check length
    .isAlphanumeric().withMessage('Username must only contain letters and numbers'), // Ensure alphanumeric

  // Validate the password field
  body('password')
    .isLength({ min: 8 }).withMessage('Password must be at least 8 characters long') // Check length
    .matches(/\d/).withMessage('Password must contain a number') // Ensure it has at least one digit
    .matches(/[a-zA-Z]/).withMessage('Password must contain a letter'), // Ensure it has at least one letter
];

// Middleware to handle the validation result
const validateRegister = (req, res, next) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array().map(err => err.msg) });
  }
  next();
};
```

3. When a user browses for new friends, his/herself will not be displayed, nor will the people who are already liked by the user. [done+2?]
4. Users have the choice to select how long the login is valid. If the user ticks "Remember me" when logging in, the JWT token will last one hour. Otherwise, it only lasts for 15 minutes. [done+2?]
5. The authentication process is carefully handled. The expiration of JWT token would not cause the app to crash but redirect the user to login page. The user can also log out by clicking logout button. [done+2?]

How to run?

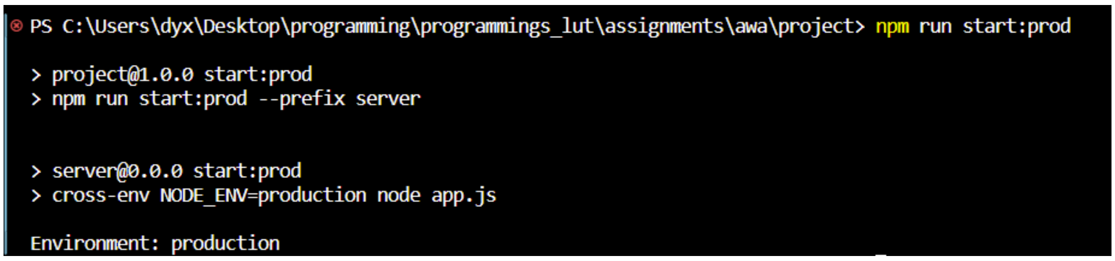
Step one, install all the dependencies.

Step two, be sure you are inside project folder, not front_end folder or server folder.



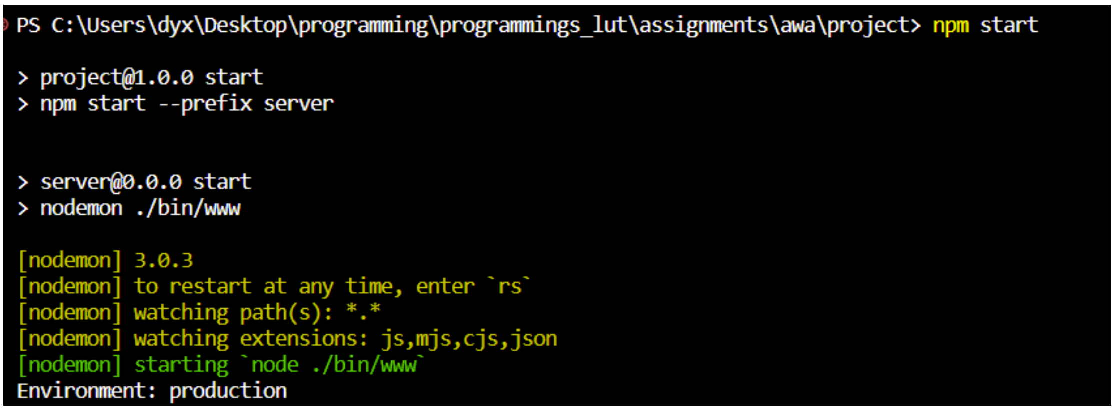
```
PS C:\Users\dyx\Desktop\programming\programmings_lut\assignments\awa\project>
```

Step three, start production env by using command: "npm run start:prod"



```
PS C:\Users\dyx\Desktop\programming\programmings_lut\assignments\awa\project> npm run start:prod

 > project@1.0.0 start:prod
 > npm run start:prod --prefix server


 > server@0.0.0 start:prod
 > cross-env NODE_ENV=production node app.js

 Environment: production
```

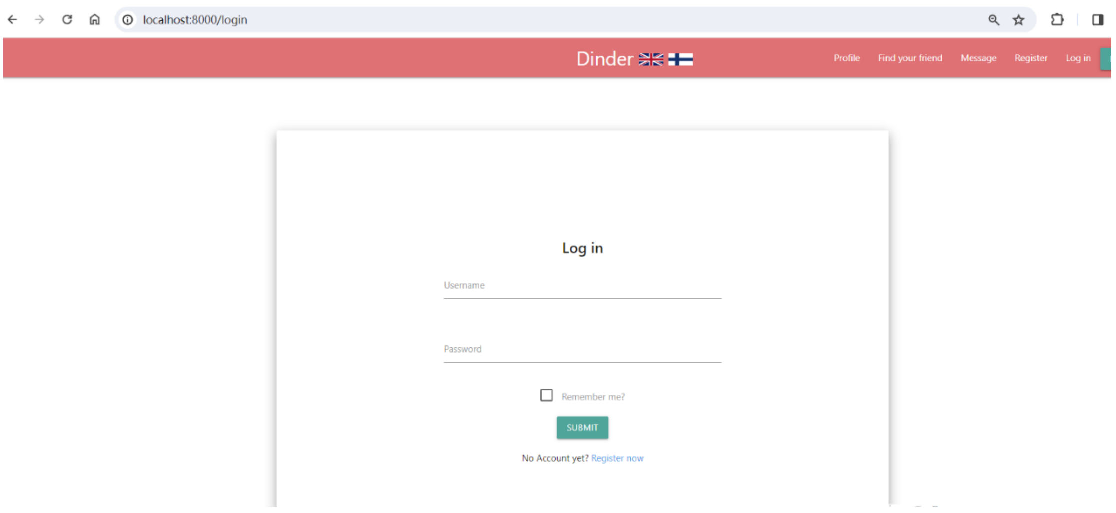
Step four, start the server "npm start"(use another terminal window)



```
PS C:\Users\dyx\Desktop\programming\programmings_lut\assignments\awa\project> npm start

> project@1.0.0 start
> npm start --prefix server


> server@0.0.0 start
> nodemon ./bin/www

[nodemon] 3.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./bin/www`
Environment: production
```

Step 5, open the browser and navigate to "localhost:8000". The site is on port 8000

Project work: Dinder


localhost:8000/login
Dinder
Profile
Find your friend
Message
Register
Log in
Log in
Username
Password
Remember me?
SUBMIT
No Account yet? Register now