CS-C3170 - Web Software Development

Course Project – WeCare Application

Documentation

Haibi Peng 875552

□ CREATE TABLE statements needed to create the database used by the application

```
//table used to store user info
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  email VARCHAR(320) NOT NULL,
  password CHAR(60) NOT NULL
);
```

```
//table used to store report info
CREATE TABLE reports (
  id SERIAL PRIMARY KEY,
  date DATE,
  morning BOOLEAN,
  sleepduration decimal(2,1),
  sleepquality INT,
  mgenericmood INT,
  night BOOLEAN,
  sporttime decimal(2,1),
  studytime decimal(2,1),
  eatregnqua INT,
  ngenericmood INT,
  user_id INTEGER REFERENCES users(id)
);
```

□ The address at which the application can currently be accessed

The application can currently be accessed by using the link below:

https://wsd-project-wecare.herokuapp.com/

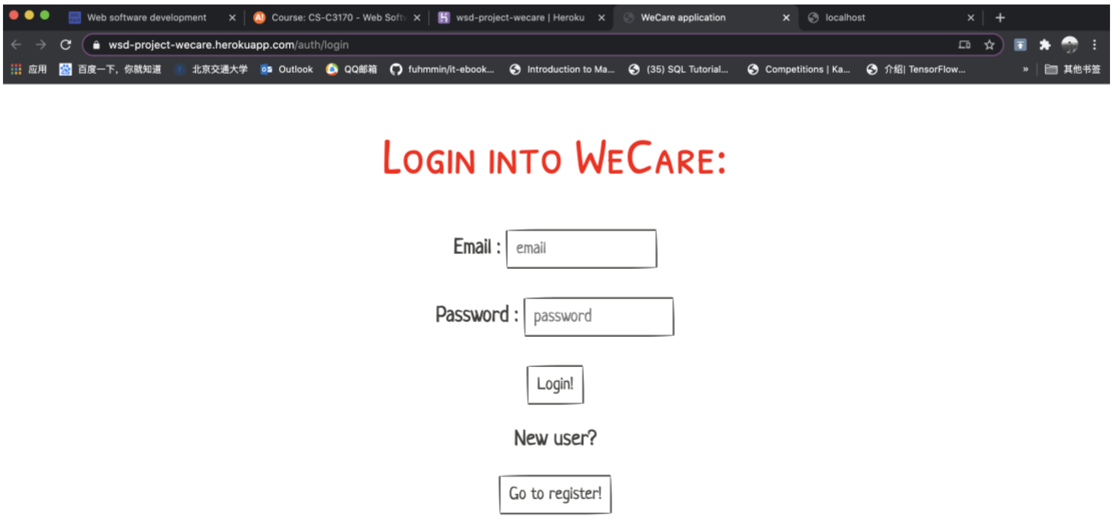
□ The Github repository of the project:

The project can be accessed on Github by using the link below:

https://github.com/HaibiPeng/CS-C3170-WSD-CourseProject-WeCare-Application

□ Guidelines for running the application

a. Running the application by link above: Using Chrome to open the link



wsd-project-wecare.herokuapp.com/auth/login

Login into WeCare:

Email : email

Password : password

Login!

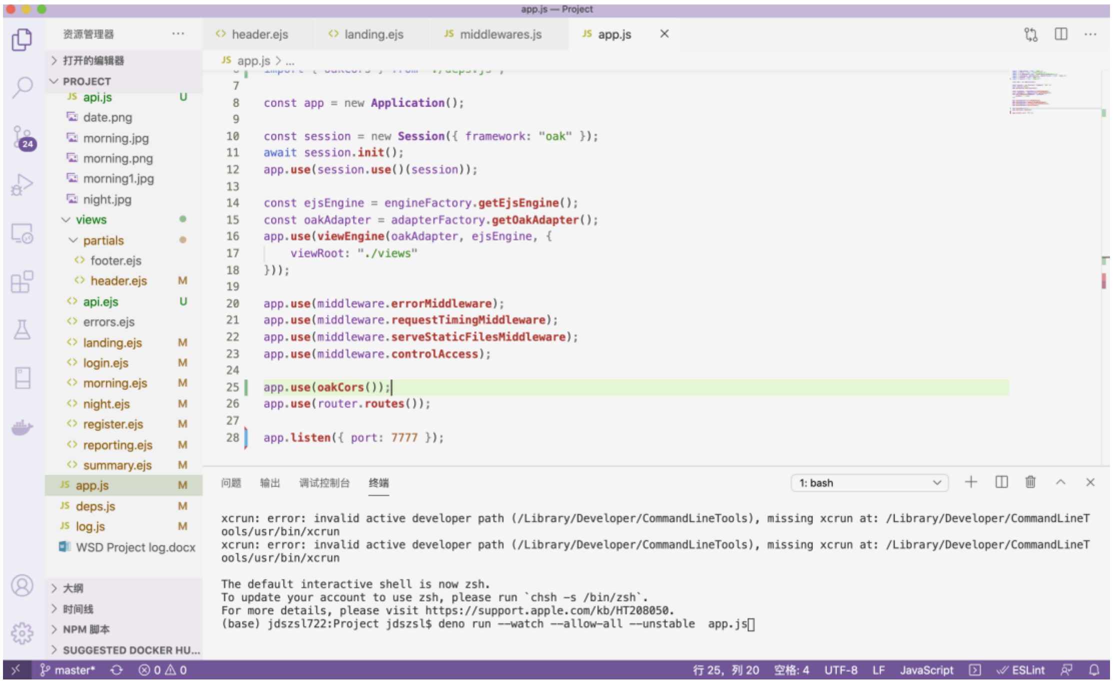
New user?

Go to register!

b. Running the application using local host:

1. Using Visual Studio Code to open the application file folder
2. Change the directory of Terminal/CMD/CLI in Visual Studio Code to the directory where the ***app.js*** file lies
3. Start the application by entering the following command and strike enter:

```
deno run --watch --allow-all --unstable   app.js
```

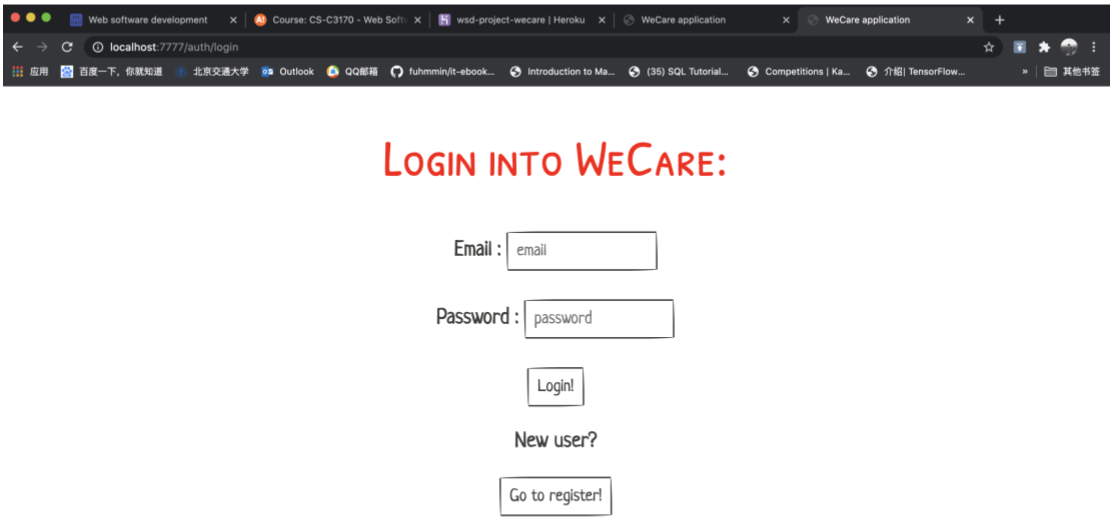


```
const app = new Application();

const session = new Session({ framework: "oak" });
await session.init();
app.use(session.use()(session));

const ejsEngine = engineFactory.getEjsEngine();
const oakAdapter = adapterFactory.getOakAdapter();
app.use(viewEngine(oakAdapter, ejsEngine, {
    viewRoot: "./views"
}));

app.use(middleware.errorMiddleware);
app.use(middleware.requestTimingMiddleware);
app.use(middleware.serveStaticFilesMiddleware);
app.use(middleware.controlAccess);

app.use(oakCors());
app.use(router.routes());

app.listen({ port: 7777 });
```

```
xcrun: error: invalid active developer path (/Library/Developer/CommandLineTools), missing xcrun at: /Library/Developer/CommandLineT
ools/usr/bin/xcrun
xcrun: error: invalid active developer path (/Library/Developer/CommandLineTools), missing xcrun at: /Library/Developer/CommandLineT
ools/usr/bin/xcrun

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) jdszsl722:Project jdszsl$ deno run --watch --allow-all --unstable  app.js
```

4. Open the application by entering the link below in Chrome:

http://localhost:7777/



□ Guidelines for running tests

1. Using Visual Studio Code to open the application file folder
2. Change the directory of Terminal/CMD/CLI in Visual Studio Code to the directory where you want to run tests
3. Run tests by entering the following command and strike enter:

```
deno test --allow-all --coverage --unstable xxx_test.js
```

where xxx_test.js is the test file in the folder

□ The functionality for handling duplicate reports

Since morning and night reporting data of one day are in the same line, so there are 3 cases regarding whether the data have been reported.

In addition to the needed information columns, this application adds two more columns, *morning/night (Boolean values, true/false)*, ***to record whether the morning and night data have been reported as the creating table statements above.***

When a new reporting is to be stored in the database, we have:

```
//To check whether the morning and night data have been reported by using columns date and morning/night
const morninginfo = await executeCachedQuery("SELECT COUNT(*) FROM reports WHERE date=$1 and morning=true and user_id=$2;", data.date, userID);
```

```
    const nightinfo = await executeCachedQuery("SELECT COUNT(*) FROM reports WHERE date=$1
and night=true and user_id=$2;", data.date, userID);
if (morning data reported) {
        update the morning data using ‘UPDATE’ in SQL, where morning value does not need change
since we already set the morning value as true
    } else if (night data reported && morning data not reported) {
        update the morning data using ‘UPDATE’ in SQL, where morning value need to be set as true
since we do not have the morning data before
    } else {
        insert the morning data using ‘INSERT’ in SQL, where all the morning reporting values are need
    }
```

And it is the same logic with the night data.