Name : Shrey Shah

ID: N01549811

Week 7

Warmup Activity

**Step1:** Download and unzip the given Nodejs project. Use npm i to install all dependencies.

**Step2:** Run the app. You will see the home page and a registration form, don;t fill up the form and just click on submit button. What happen?

A pink and white rectangle

Description automatically generated

A screenshot of a computer

Description automatically generated

It will give us the validations related to username and email that username should be of atleast of characters nd email format is not valid.

**Step3:** Answer the following questions:

a) identify all ROUTES in this app.  
  
The root route (app.get('')) renders the 'index' view.

The '/register' route has two handlers:

The first (app.get('/register')) renders the 'register' view.

The second (app.post('/register')) handles form data validation.

b) How form-data-validation is implemented in this app.

The form-data validation is implemented using the express-validator library in the '/register' POST route. The validation rules are defined using the check function from the library.

c) Identify all middle wares in this app.

urlencodedParser is a middleware created using body-parser to handle URL-encoded data in the POST request to '/register'.

The second parameter of the '/register' POST route contains an array of middleware functions for form-data validation using express-validator.

d) Identify "template engine views" in this app.

The app uses the Handlebars template engine (express-handlebars) for rendering views. The 'index' and 'register' views are rendered using the Handlebars template engine.

e) What is the role of "body-parser" in this app? Can we remove it?

The body-parser middleware (urlencodedParser) is used to parse URL-encoded data from the POST request body. It is necessary for processing form data sent from HTML forms.

Removing it would result in the inability to parse form data, and the data in the req.body object would not be accessible.

f)What if the role of two <script> block inside the "register.hbs" and "index.hbs" ?

Without the actual content of the <script> blocks, it's challenging to provide specific details. However, in general, <script> blocks in HTML files are used to include JavaScript code.

In the context of these Handlebars views, the <script> blocks could contain client-side JavaScript code that is meant to be executed in the browser. This could include functionality for handling user interactions, making AJAX requests, or other client-side logic.

The role and impact of these <script> blocks would depend on the content within them. If they contain specific functionality or logic, removing them might affect the expected behavior of the web pages.

**Step4:** Run the app. Fill up all the fields in the registration form, and click on Submit. What happen? what is missing? Any idea how to fix it?

A screenshot of a computer

Description automatically generated

After submit it is not doing anything it is just refreshing.

Reason for above query is that there is a missing part in the code related to handling the case where the form data is valid. Currently, the code only handles the case where there are validation errors. If the form data is valid, there is no code to process the registration or provide a response to the user.  
  
To fix this issue, we need to add the code that handles the case where the form data is valid. Typically, in an application like this, you would have code to store the user information in a database or perform some other action related to user registration

A screen shot of a computer program

Description automatically generated

A white rectangular object with pink border

Description automatically generated

Activity 1

**Activity1:**This activity will add a new "views" to the warmup activity to display the submitted-form-data.

Complete the following steps:

**Step1:** Add a new views to your project file, named it "output.hbs" with the following contents:

<li>{{data}} </li>

A screenshot of a computer

Description automatically generated

**Step2:** Modify app.js, and add the following else block to app.post() route:

else res.render('output', {

     data: 'nothing',

     layout: false

  });

**Step3:** run the app, and fill up the form with valid data, click on Submit button, what heppen?  
A white background with pink lines

Description automatically generated

**Step4:** What if you remover the "else" and keep the rest. Run the app, what did you notice?  
  
  
Validation is taking place here, but the response element is taking time; the application will offer the 'output' view with the informational index set to 'nothing' whether or not there are approval errors. This means that regardless of whether there are approval errors, the 'yield' view will always return the 'nothing' information.

A screenshot of a computer

Description automatically generated

A pink and white rectangle

Description automatically generated

**Step5:** We want to change "data :'nothing'" and pass form-data to the "output.hbs". How to do this?

In this I have update the output.hbs file as it takes the username field value from register.hbs

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**Use of Custom Helper:**

I have added the calculation helper first into output.hbs and used custom way to call it

**A computer screen shot of a computer code

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Second way:**

I have used the “options” parameter, helper can be used as{{#helperName}} ... {{/helperName}} in .hbs

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Group Activity:**

Changed layout to main.hbs so it can use the default one.

**A screen shot of a computer program

Description automatically generated**

**A computer code on a black background

Description automatically generated**

**A screen shot of a computer screen

Description automatically generated**

**A white background with pink border

Description automatically generated**

**Now welcome.hbs part**

Removed the layout part to call the partial one

Welcome.hbs

A white background with pink border

Description automatically generated

Deployment

First Put a code into github

A screenshot of a computer

Description automatically generated

Then attached a repo to cyclic and the deployment is live

A screenshot of a computer

Description automatically generated

A white rectangular object with pink border

Description automatically generated

Link : <https://frantic-eel-twill.cyclic.app/>