CMPS 350 Web Development Fundamentals
Lab 12 -View Template using Handlebars

Objective

The objective of this lab is to practice implementing a simple Web application that provides a Web interface that is dynamically generated using Handlebars View Template engine. You will be using handlebars, fetch API and other packages to achieve the project.

Project Setup

1. Download **"Lab 12 - HBS View Template"** from the GitHub Repo.
2. Open the project folder using **WebStorm**. Ensure that your **Webstorm** JavaScript language is set to **ECMASript 6** and **Node.js Core** Libraries are enabled.
3. The project is organized using the following folders:
    - *a.* **views** folder which contains HTML pages, templates, css and client-side JavaScript
    - *b.* **data** folder has JSON files to be used in this lab.
    - *c.* **repository :** contains all the repo classes
    - ***d.*** **service** folder contains the controller classes.
    - ***e.*** **models** folder contains the model classes.

PART A - Tutorial

1. Create a Web API to allow the user to get CENG programs and to get courses offered by a program:
    - Create a *CourseRepository* class and add the following two methods:

getPrograms	Fetches the content of https://cmps356s17.github.io/data/ceng-programs.json and return the programs
getCourses(programCode)	Fetches the content of https://cmps356s17.github.io/data/ceng-courses.json and returns the courses offered by a program.

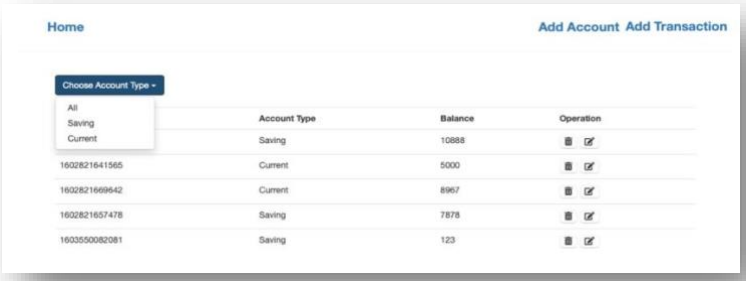
    Tip: use node-fetch npm package to fetch data from a url

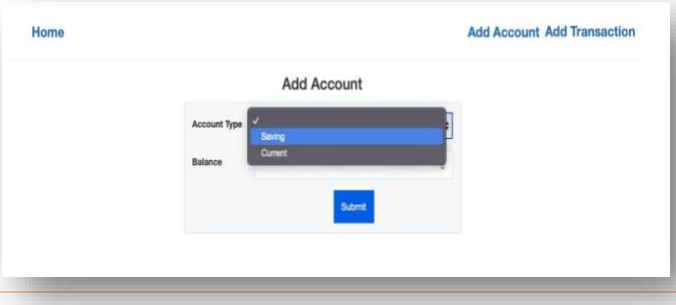
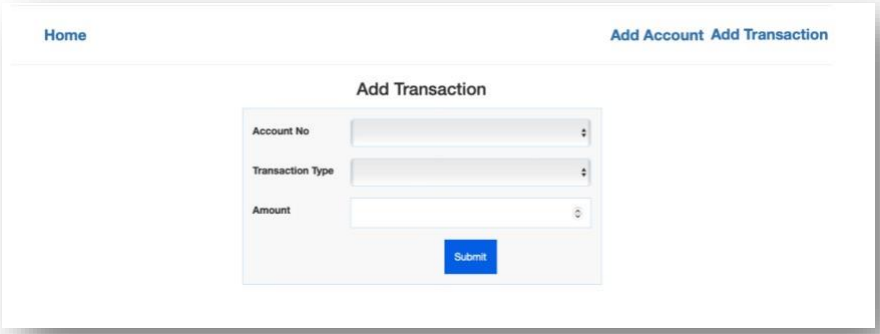
    - Create a CourseRepository-Test.js and test the *CourseRepository* methods:
    - Make *CourseRepository* methods accessible as Web API: create a *CourseController* class in the *controllers* folder and add two routes to the routes.js file.
    - Test the course Web API (e.g., api/programs and api/courses/cs)

2. **Create a view to allow the use to get courses offered by a program:**
    - In ***app.js*** configure express *app* to use handlebars template engine. The default layout should *layout.hbs*
    - Create *course.hbs* template to return CENG programs in a dropdown
    - Add a new '*/courses*' route to routes.js to render *course.hbs* view using the programs returned by CourseRepository.getPrograms
    - Add a client-side event handler to listen to the **onChange** event of *programs* dropdown. Whenever the program changes, then the event handler should fetch the courses for the selected program using an AJAX call. The json return should be used to update the page to display the list of courses offer by the select program. You should use a handlebar template and render it at the client side.

PART B – Banking Web APP

Your task is to develop a user interface for Banking app you developed in the previous labs. The base solution of **Banking application** is provided for you. The Web UI you need to design and implement are:

Web Page	Functionally
index.html	First, the index page should have a main navigation menu providing 3 links: • *Home* link is the home page that allows getting accounts. • *Add Account* link allows adding an account. • *Add Transaction* link allows adding a transaction. - All the app pages should be accessible from the index.html page. - This page allows getting accounts by type. It provides a dropdown to select the account type: *Saving*, Current or All. When the page load or when a different account type is selected then the page should fetch the accounts from /api/accounts?type=acctType - For each account with the balance=0, a *Delete* button is provided to enable deleting the account. Then this button is clicked the account should be deleted using /api/accounts/:id Web API. Also, the account row should be deleted from the html accounts table. 

acct-form.html	Allows creating a new account by posting the account data to */api/accounts* 
acct-trans.html	Allows selecting a particular account from the accounts dropdown then sub mitting a deposit or withdrawal transaction. The new transaction should be posted to /api/accounts/:id/trans. The page should display any error returned from the Web API (e.g., insufficient balance). 

You need to test your implementation as you progress and document your testing. After you complete the lab, fill in the ***Lab12-TestingDoc-Grading-Sheet.docx*** and save it inside **Lab 12 - HBS View Template** folder. Sync your repository to push your work to **GitHub**.