Introduction

This assignment is the fifth of six assignments. In this assignment you will upload image files as part of a multi-part form submission.

Before you begin this assignment, you must finish your previous assignment. All objectives listed for this assignment are to be made "on top" of your previous assignment.

This assignment is worth 9% of your final grade.

Reminder about academic integrity

Most of the materials posted in this course are protected by copyright. It is a violation of Canada's Copyright Act and Seneca's Copyright Policy to share, post, and/or upload course material in part or in whole without the permission of the copyright owner. This includes posting materials to third-party file-sharing sites such as assignment-sharing or homework help sites. Course material includes teaching material, assignment questions, tests, and presentations created by faculty, other members of the Seneca community, or other copyright owners.

It is also prohibited to reproduce or post to a third-party commercial website work that is either your own work or the work of someone else, including (but not limited to) assignments, tests, exams, group work projects, etc. This explicit or implied intent to help others may constitute a violation of Seneca's Academic Integrity Policy and potentially involve such violations as cheating, plagiarism, contract cheating, etc.

These prohibitions remain in effect both during a student's enrollment at the college as well as withdrawal or graduation from Seneca.

This assignment must be worked on individually and you must submit your own work. You are responsible to ensure that your solution, or any part of it, is not duplicated by another student. If you choose to push your source code to a source control repository, such as GIT, ensure that you have made that repository private.

A suspected violation will be filed with the Academic Integrity Committee and may result in a grade of zero on this assignment or a failing grade in this course.

Technical Requirements

- All back-end functionality **must** be done using **Node.js** and **Express**.
- You will use the **body-parser** module to handle form submissions.
- You will use the **express-session** module to handle user session state information.
- You will use **bcrypt.js** to encrypt user passwords.
- You **must** use MongoDB as your database engine.
- Your views **must** be created with **Express-Handlebars**.
- You **can use** a front-end CSS framework such as Bootstrap, Bulma or Materialize CSS to make your website responsive and aesthetically pleasing.
- You are **not allowed** to use any Front-End JavaScript Frameworks. For example, you may not use React, Vue, or Angular.

Objectives

Data Loading

Create a throw-away "load data" controller but do not delete the controller; the controller must be submitted so it can be marked by your professor.

The "load data" controller must contain a route that will populate meal kit data. The following are the requirements for the route:

1. A data clerk can load meal kit data to the database by accessing the URL "/load-data/meal-kits".
2. This functionality must be placed in its own controller and must be coded using the MVC design patterns learned in class.
3. You must check to ensure there are no meal kits in the database before adding new meal kits; this is it to prevent adding of duplicate entries.
4. You must populate a minimum of three "top meals" and six "on the menu" meals.
    a. The "on the menu" page should contain at least two different categories.
    b. It is okay for "top meals" to appear on both the home page and the "on the menu" page.
5. After accessing a load data route, display a Handlebars view with a message describing the outcome:
    a. If data was loaded to the database, display a message like "Added meal kits to the database".
    b. If data has already been loaded to the database, display a message like "Meal kits have already been added to the database".
    c. Only data clerks can load data. If the user is not a data clerk, display a message like "You are not authorized to add meal kits".

   d. The message is displayed on a page showing the proper header and navigation bar.
6. For simplicity, add a link from the data clerk dashboard to the load-data page.

Data Entry Clerk Module

You are required to implement a Data Entry Clerk Module that allows a "logged-in" data entry clerk to perform the following tasks.

1. /cleark/list-mealkits: View a list of all the meal kits. This functionality is different than viewing meal kits on the home page and "on-the-menu" page. This page will allow the data clerk to perform CRUD operations on the meal kits database. You already started this page in Assignment 4.

2. /clerk/add-mealkit: Create meal kits. The data entry clerk must be able to add new meal kits to the database. All of the fields from assignment 2 must be included. For example:
   a. Title – *Sautéed Ground Pork over Jasmine Rice*
   b. What is included? – *Toasted Peanuts & Quick-Pickled Cucumber Salad*
   c. Description (This is a text area, not a text input) – *Gingery pork, crunchy cucumbers and toasty peanuts make for a classic culinary …*
   d. Category – *Classic Meals*
   e. Price – *$19.99*
   f. Cooking Time – *25*
   g. Servings – *2*
   h. Is it a Top Meal? – *true (a checkbox control is recommended)*
   i. Upload a photo of the meal kit *(to keep the project simple, only upload one image per meal kit).*

3. /clerk/edit-mealkit/:id: Edit meal kit details for a selected meal kit. All fields can be changed, including the photo.

4. /clear/remove-mealkit/:id: Remove meal kits from the database. You do not need to delete the photo file from your web server, just the document from the database.

5. Ensure that all new meal kits that were entered into the database are populated on the front-end of the web application.
   a. All meal kits must appear on the "on-the-menu" page.
   b. Only the meal kits that were set as "Top Meals" should be populated in the "Top Meals" section of the home page.
   c. A visitor does not need to be logged in to view the meal kits on the home page and the "on-the-menu" page.

6. Ensure that a user can only upload images stored as one of the following media types (extensions): jpg, jpeg, gif, or png.

GitHub

You will continue to push your web application to a remote GitHub repository in your own account. Continue to use the same repository you set up in assignment 3.

If you haven't already done so, **set your remote repository to private.** Remember to add your professor as a collaborator so he/she can view your web application.

A realistic view of your progress must be demonstrated by looking at your commits.

Cyclic

This assignment will be marked locally (on your professor's machine). You do not need to deploy this assignment to Cyclic.

Rubric

Criteria	**Not Implemented (0)** Little or no work done. Unacceptable attempt.	**Partially Implemented (1)** Work is minimally acceptable but is incomplete or needs significant modification.	**Fully Implemented (2)** Work is complete and done perfectly.
Data Clerk Module • The data entry clerk can view a list of all meal kits (excludes home page and on-the-menu page). • A data entry clerk can create meal kits supplying necessary data. • The data entry clerk can edit all meal kit details. • A data entry clerk can upload a photo for a new meal kit or an existing meal kit. • Only images files can be uploaded as a photo. • A data entry clerk can remove/delete meal kits. • New meal kits appear on the “on-the-menu” page with the correct category. • Meal kits that are set as top meals are shown on the home page.			

Load Data Module • Only data clerks can load data by navigating to "/load-data/meal-kits". • Duplicate meal kits are not added when run multiple times. • A correct, formatted, message is displayed after attempts to load data. The header and navigation bar present. Controllers • Functionality for loading data and the data clerk module are modularized into dedicated controllers.			

Total: 24 Marks

Note: Half marks may be awarded.

Submitting your work

Make sure you submit your assignment before the due date and time. It will take a few minutes to package up your project so make sure you give yourself a bit of time to submit the assignment.

1. Locate the folder that holds your solution files. You may choose to delete the node_modules folder but do not delete any other files or folders.

2. Compress the copied folder into a zip file. **You must use ZIP compression, do not use 7z, RAR, or other compression algorithms or your assignment will not be marked.**

3. Login to My.Seneca, open the **Web Programming Tools and Frameworks** course area, then click the **Project** link on the left-side navigator. Follow the link for this assignment.

4. Submit/upload your zip file. The page will accept unlimited submissions so you may re-upload the project if you need to make changes. Make sure you make all your changes before the due date. Only the latest submission will be marked.