Introduction

This assignment is the second of six assignments. It has been designed to give you practical experience creating and working with JavaScript modules and creating Express Handlebars views.

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.

Note: Database connectivity is **not** required for this assignment.

Reminder about academic integrity

You must comply with Seneca College's Academic Integrity Policy. Although you may interact and collaborate with others, 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**.
- Your views **must** be created with **Express-Handlebars**.
- You **can use** a front-end CSS framework if you would like.

Objectives

Data Module

In this assignment, you are not reading data from a database. Instead, you must create **one** separate back-end node.js module file to encapsulate the static *("fake")* data for both the "Top Meals" section on the home page and the meals shown on the "On-the-Menu" page. This module/JavaScript file is going to represent the Model for your MVC application.

Your module, when complete, will:

- Contain a local variable to store an Array of all meal kit objects.
  Note: You may only use a single variable to store the meal kits. This variable is a local variable and therefore it should not be exported.
- Export a function that will return an array of all meal kits grouped by their category name. This function is used on the "on-the-menu" page.
- Export a function that will return an array of all meal kits where the meal kit has been flagged as a "top meal". This function is used on the home page to display top meals.

Each meal kit will require at minimum the following properties:

- Title – *Sautéed Ground Pork over Jasmine Rice*
- What is included? – *Toasted Peanuts & Quick-Pickled Cucumber Salad*
- Description – *Gingery pork, crunchy cucumbers and toasty peanuts make for a classic culinary …*
- Category – *Classic Meals*
- Price – *$19.99*
- Cooking time – *25 minutes*
- Servings – *2*
- Calories Per Serving – *890*
- Image URL – *For now, point to an image placed in your static files folder.*
- Is it a Top Meal? (Boolean) – *true*

Handlebars Implementation

You have already created a home page in the previous assignment. The home page includes several components. Your first task is to move each component into reusable handlebars views. At minimum, you must create the following partial views. You may create additional partial views if desired.

- Header – contains the html used to construct the header and navigation bar.
- Hero – contains the html used to construct the hero.
- Footer – contains the html used to construct the footer area.

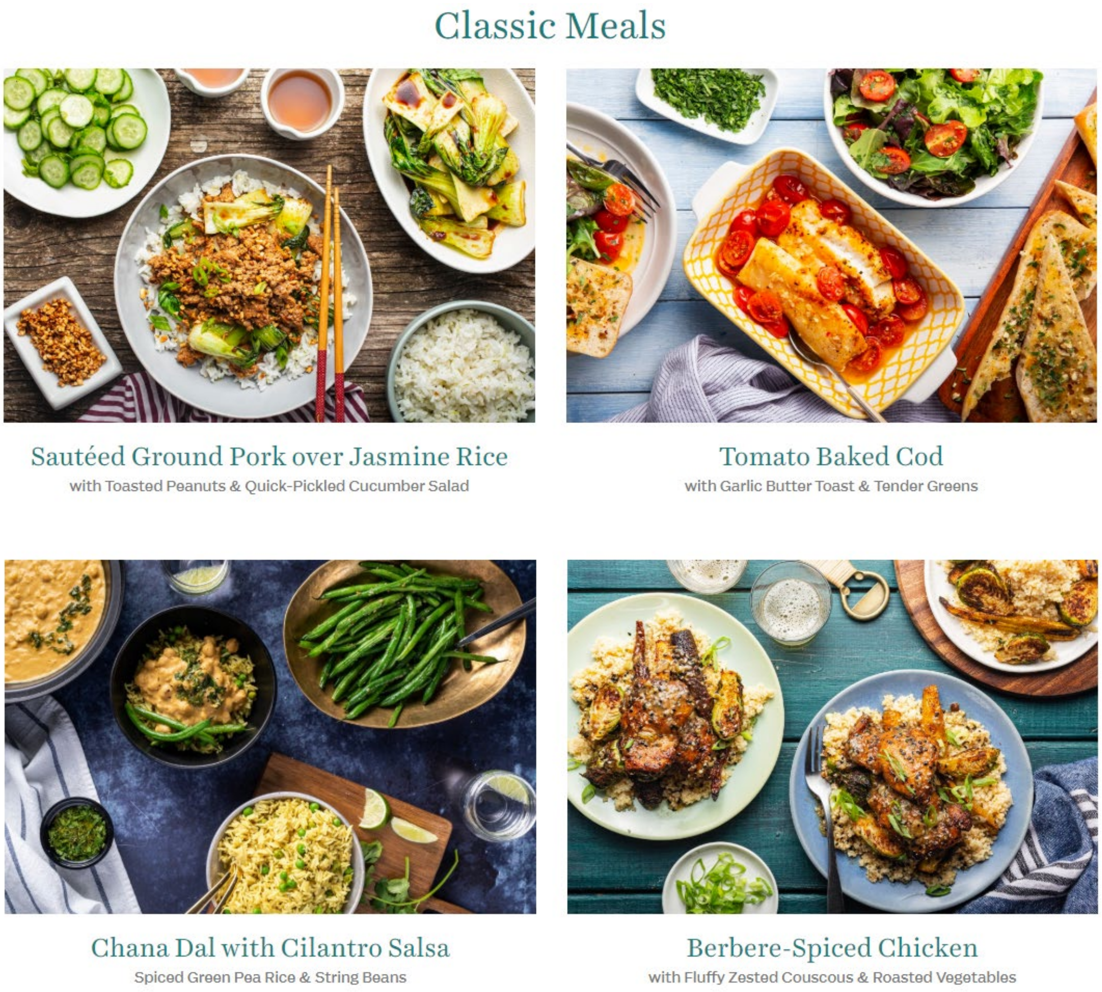
Create a handlebars layout (main) view. The layout will contain markup that is shared across the pages of your website.

Create handlebars views for the following pages. *Remember, you have already set up the routes in the previous assignment. Make sure these routes utilize the main layout view.*

- Home – contains the html used to construct the content section(s) and top meals section.
- On the Menu – you will create a new view to show the current meals.
- Registration – a customer registration page to allow new users to create an account.
- Sign-In – a login page to allow existing users to access their account.

"On the Menu" Page

You are required to build a well-designed meal listing page. This page **must** show at least four (4) sample meals within a 2x2 grid in at least one section, as shown in the image below:



Classic Meals

Sautéed Ground Pork over Jasmine Rice

with Toasted Peanuts & Quick-Pickled Cucumber Salad

Tomato Baked Cod

with Garlic Butter Toast & Tender Greens

Chana Dal with Cilantro Salsa

Spiced Green Pea Rice & String Beans

Berbere-Spiced Chicken

with Fluffy Zested Couscous & Roasted Vegetables

You will notice the meals in the above screen shot are categorized in a section titled “Classic Meals”. This title is read from the “Category” property of the meal kits. If you have created your data module correctly, the data you are working with is already grouped by category. To demonstrate this functionality, you must ensure that the on-the-menu view contains/displays **at least two sections**.

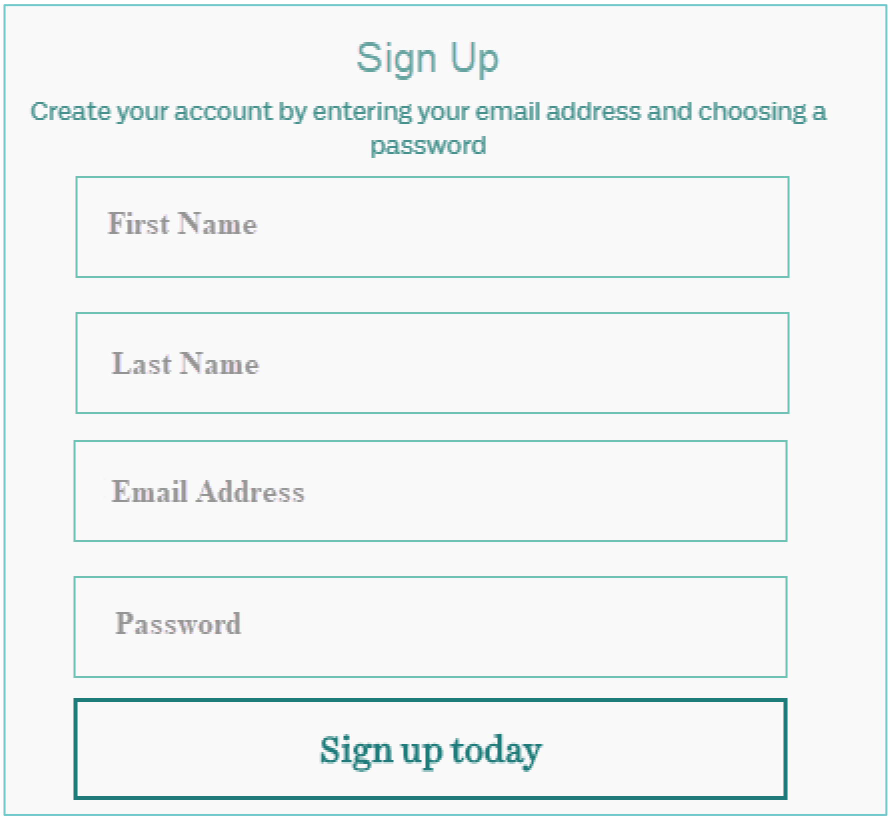
Every meal kit must show an image, title (name of the meal kit), and an explanation of “what is included”. Like the “Top Meals” section on the home page, the data for this section will not be pulled from a database, however, you are required to define the data in a separate back-end node.js module.

This view must include the header, navigation bar, hero section, and footer.

Registration Page

You are required to build a well-designed user registration form as shown below. You must ensure that the page maintains consistency. The view must include the header, navigation bar and footer. You may optionally include a hero section as well.

You do not need to handle/implement form submission, client-side validation, or server-side validation.



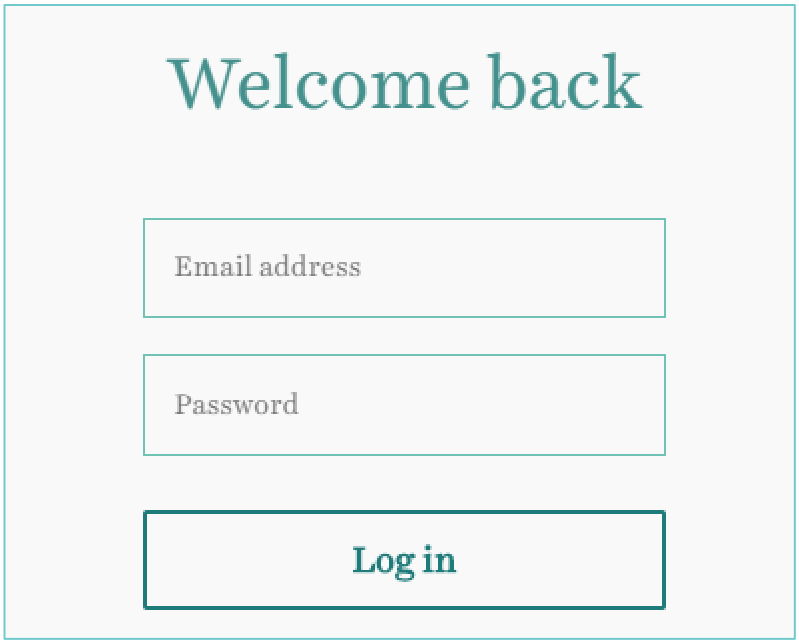
Sign Up

Create your account by entering your email address and choosing a password

Login Page

You are required to build a well-designed user login form as shown below. You must ensure that the page maintains consistency. The view must include the header, navigation bar and footer. You may optionally include a hero section as well.

You do not need to handle/implement form submission, client-side validation, or server-side validation.



Responsive Design

Ensure that your entire website renders well on a variety of devices, specifically on desktops, tablets, and smartphones. To accomplish this, you will need to use CSS Media Queries in combination with a modern CSS Layout Module (CSS Grids, or Flexbox, or both). You may optionally use Bootstrap in this assignment. Also ensure that all pages of your site follow the same branding outlined in your original home page. This means that all pages should use consistent colours, fonts, and styles.

Reminder

All back-end functionality **must** be done using **Node.js** and **Express**. Your views **must** be created with **Express-Handlebars**. You will have at least three partial views and four view pages. The page must operate responsively, in other words, must function well on all browser sizes.

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 Module • Contained in a separate Node.JS module. • Includes a local variable to store all meal kits as an array. • Contains a function to filter meal kits by “top meals”. • Contains a function to group meal kits by their “category”.			
Partial Views • Header and navigation bar. • Hero section • Footer section.			
Views • Shared (main) layout view. • Registration form. • Login form.			

Home Page • Converted to a view with necessary sections included. • “Top meals” data is dynamic and passed into the view.			
On the Menu • Meals broken by category. • Meals in a 2x2 grid. • Each meal shows an image, title and “what’s included”. • Meal data is dynamic and passed into the view.			
Responsive Design • Overall site looks polished on all devices, specifically on smartphones, tablets, and large screens.	**Poor (1)**	**Average (3)**	**Exceeds (6)**

Total: 38 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.

2. Compress the copied folder into a zip file. **Do not use 7z or other compression algorithms or your assignment will not be marked.**

3. Login to My.Seneca.

4. Open the **Web Programming Tools and Frameworks** course area and click the **Project** link on the left-side navigator. Follow the link for this assignment.

5. Submit/upload your zip file. The page will accept three 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.