

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

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 implemented using only **Node.js**, **Express**, and **Express-Handlebars**. Other frameworks/packages are not allowed for this assignment.
- 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 Module

In this assignment, you are not reading data from a database. Instead, you will 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 is going to represent the Model for your MVC application. Place the file in a “models” folder and call it “mealkit-db.js”.

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. You will need to include at minimum six mealkits: four in one category and two in another.
- Export a function called **getTopMealkits()** 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.
- Export a function called **getMealkitsByCategory()**, used by the “on-the-menu” page, that will return a single array of the meal kits grouped by their category name.

Each meal kit will require at minimum the following properties. Use the property name and data type provided.

- **title** (String) – *Sautéed Ground Pork over Jasmine Rice*
- **includes** (String) – *Toasted Peanuts & Quick-Pickled Cucumber Salad*
- **description** (String) – *Gingery pork, crunchy cucumbers, and toasty peanuts.*
- **category** (String) – *Classic Meals*
- **price** (Number) – *\$19.99*
- **cookingTime** (Number, in minutes) – *25*
- **servings** (Number) – *2*
- **imageUrl** (String) – *For now, point to an image placed in your static files folder.*
- **topMeal** (Boolean) – *true*

Notes regarding the `getMealsByCategory()` function.

- You must not hardcode or limit the number of category names that will be returned.
- All logic must be written into a single function using vanilla JS. There is no need to have supporting local functions.
- You may use ES6 JavaScript features to help but this is not mandatory.
- The array returned should look similar to this:

```
[ {
  "categoryName": "Classic Meals",
  "mealKits": [
    { title: "Mealkit 1", "includes": "", ... "topMeal": true },
    { title: "Mealkit 2", "includes": "", ... "topMeal": false }
  ] },
{
  "categoryName": "Vegan Meals",
  "mealKits": [
    { title: "Mealkit 3", "includes": "", ... "topMeal": false },
    { title: "Mealkit 4", "includes": "", ... "topMeal": 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.hbs** – contains the html used to construct the header and navigation bar.
- **hero.hbs** – contains the html used to construct the hero and content section(s).
- **footer.hbs** – contains the html used to construct the footer area.
- **mealkit.hbs** – contains the html used to construct a single mealkit card.

Create a handlebars layout view (**main.hbs**). The layout will contain markup that is shared across the pages of your website. This page must include the partial views for the header and footer. Do not include the hero or mealkit partial views in the main layout file.

Create handlebars views for the following pages. *Remember, you have already set up the routes in the previous assignment. Make sure these pages utilize the main layout view. Anytime a mealkit is displayed, ensure you are using the "mealkit" partial view.*

- Home – contains the hero partial view and the html used to construct the top meals section.
- On the Menu – you will create a new view to show all the mealkits.
- 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 mealkit listing page. This page must show mealkits in at most two columns wide, 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



Berber-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**. At least one section should show four mealkits.

Every meal kit must show an image, title (name of the meal kit), an explanation of “what is included”, and a price. 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, and footer. Don’t forget, the html for the mealkit cards are contained in a partial view and included on this page.

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. Do not include the hero or content section(s) on this page.

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

First Name

Last Name

Email Address

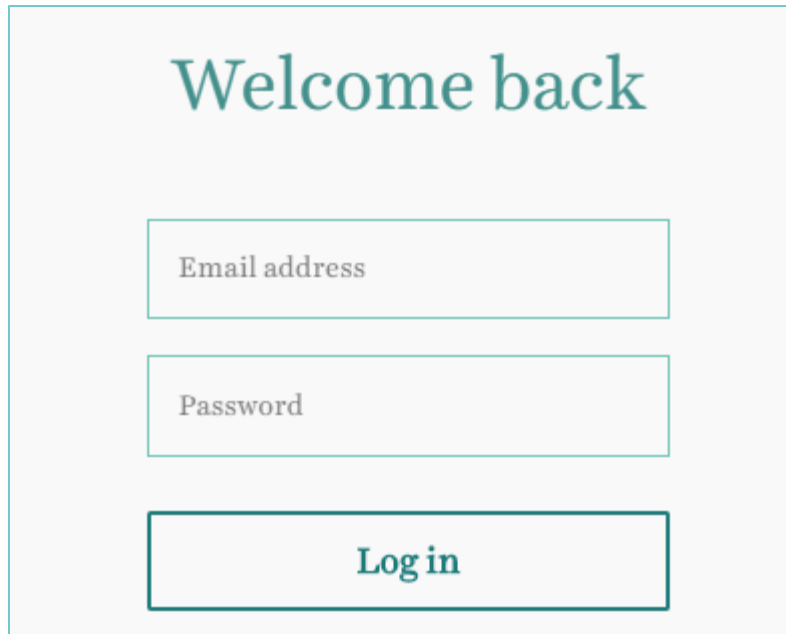
Password

Sign up today

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. Do not include the hero or content section(s) on this page.

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



Welcome back

Email address

Password

Log in

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**, **Express** and **Express-Handlebars**. Your views **must** be created with **Express-Handlebars**. You will have at least four partial views and four view pages. All pages must operate responsively, in other words, must function well on all browser sizes.

Rubric

Criteria	Not Implemented (0)	Partially Implemented (1)	Fully Implemented (2)
	Little or no work done. Unacceptable attempt.	Work is minimally acceptable but is incomplete or needs significant modification.	Work is complete and done perfectly.
<p>Data Module</p> <ul style="list-style-type: none"> Contained in a separate Node.JS module called "models/mealkits-db.js" and does not include additional functions or variables. Includes <u>one</u> local array variable to store at least six mealkits across two categories. Contains the getTopMealkits() function coded as specified. Contains the getMealkitsByCategory() function coded as specified. 			
<p>Partial Views</p> <ul style="list-style-type: none"> Header (with nav bar). Hero section (with content). Footer. Mealkit "card". 			

Views <ul style="list-style-type: none"> • Main layout view. All views use the layout file. • Registration form. • Login form. 			
Home Page <ul style="list-style-type: none"> • Converted to a view with necessary sections included. • “Top meals” data is dynamic and passed into the view. 			
On the Menu <ul style="list-style-type: none"> • Meals broken by category. • Meals in a two-column grid. • Each mealkit shows an image, title, what’s included, and price. • Mealkit data is dynamic and passed into the view as specified. 			
Responsive Design <ul style="list-style-type: none"> • Overall site looks polished on all devices, specifically on smartphones, tablets, and large screens. 	Poor (1)	Average (3)	Exceeds (6)

Total: 40 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.