

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 EJS 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, and you **must not** implement the MVC design pattern.

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-ejs-layouts**. Other frameworks/packages are not allowed for this assignment.
- Your views **must** be created with **EJS**.
- You **can use** a front-end CSS framework such as Tailwind CSS, daisyUI, or Bootstrap 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** back-end **node.js module** file to encapsulate the static (*“fake”*) data for both the “Featured Meal Kits” section on the home page and the meal kits shown on the “on-the-menu” page. Place the file in a “modules” folder and call it “mealkit-util.js”. Your module, when complete, will:

- Contain a local variable called **mealkits** to store an Array of 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 meal kits: four in one category and two in another.
- Export a function called **getAllMealKits()** that will return an array of meal kits (stored in the **mealkits** variable).
- Export a function called **getFeaturedMealKits(mealkits)** that will accept an argument that is an array of meal kits and will return a single array of meal kits where each meal kit has been flagged as “featured”. This function is used on the home page to display the featured meal kits.
- Export a function called **getMealKitsByCategory(mealkits)** that will accept an argument that is an array of meal kits and will return a single array of meal kits grouped by category. This function is used on the “on-the-menu” page to display meal kits grouped into categories. More information about this function is available on the next page.

Each meal kit object will require at minimum the following JS properties. You must 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 assets folder.*
- **featuredMealKit** (Boolean) – *true*

Notes regarding the **getMealKitsByCategory(mealkits)** function.

- Do not hardcode or limit the number of categories that will be returned.
- All logic must be written into a single function using vanilla JS. There is no need to have any supporting local functions.
- You may use ES6 JavaScript features to help but this is not mandatory. If you use the reduce() or filter() functions, you must speak with your professor ahead of time.
- The array returned should look similar to the following. Notice it is an array of objects and the “mealKits” property is another array of objects.

```
[ {
  "categoryName": "Classic Meals",
  "mealKits": [
    { title: "Mealkit 1", "includes": "", ... "featuredMealKit": true },
    { title: "Mealkit 2", "includes": "", ... "featuredMealKit": false }
  ] },
{
  "categoryName": "Vegan Meals",
  "mealKits": [
    { title: "Mealkit 3", "includes": "", ... "featuredMealKit": false },
    { title: "Mealkit 4", "includes": "", ... "featuredMealKit": true }
  ] }
]
```

EJS Implementation

In the previous assignment, you created a home page containing several components, for example, a header, navbar, hero, content sections, and a footer. You will move some of these components into reusable EJS partial view files.

Create only the following partial views and refrain from adding additional ones.

- **navbar.ejs** – contains the html used to construct the header and navigation bar.
- **footer.ejs** – contains the html used to construct the footer area.
- **mealkit.ejs** – contains the html used to construct a single meal kit card. This file must not contain an iterator.

The remaining components, such as the hero and content sections will not be moved into partial views.

Create a default layout view called **main.ejs**. The layout will contain markup that is shared across all pages of your website. This view will render the partial views for the header and footer. The layout should not include the hero image, content sections, or the “mealkit” partial view file.

Create the following EJS view files for the routes defined in the previous assignment. *Make sure these pages use the default layout view. Anytime a meal kit is displayed, ensure you are using the “mealkit” partial view from above. Do not change the URL for any of the routes, simply use `res.render()` to display the correct view file. Remember that you are not implementing controllers, so place all your views in the “views” folder and partial views in the “partials” folder. Delete the `index.html` file once you have created the views.*

- **home.ejs** – Home page, contains the hero, content sections, and featured meal kits.
- **on-the-menu.ejs** – Meal kits page, displays meal kits grouped by category.
- **sign-up.ejs** – A customer registration page to allow new users to create an account.
- **log-in.ejs** – A login page to allow existing users to access their account.

On-the-Menu Page

You are required to build a well-designed meal kit listing page.

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

The meal kits are organized in a responsive grid. On a phone, only a single column is shown, however on a desktop, two or more columns are visible.

You will notice the meal kits in the above screenshot are categorized in a section titled “Classic Meals”. This title is read from the “categoryName” property. If you have created your data module correctly, the data you are working with is already grouped for you. Use two “forEach” iterators (one embedded in the other) to render the page. To demonstrate this functionality works correctly, you must ensure that the meal kits view displays **at least two sections**. At least one section should show enough meal kits to demonstrate the wrapping works as well.

Every meal kit card will display in a similar fashion as the home page (featured meal kits). Furthermore, like the “Featured Meal Kits” 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 meal kit cards is 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.

Do not implement form submission, client-side validation, or server-side validation.

Your form must contain exactly four fields. Do not add additional fields and do not remove any fields. You will include “First Name”, “Last Name”, “Email” and “Password”.

Sign Up

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

First Name

Last Name

Email Address

Password

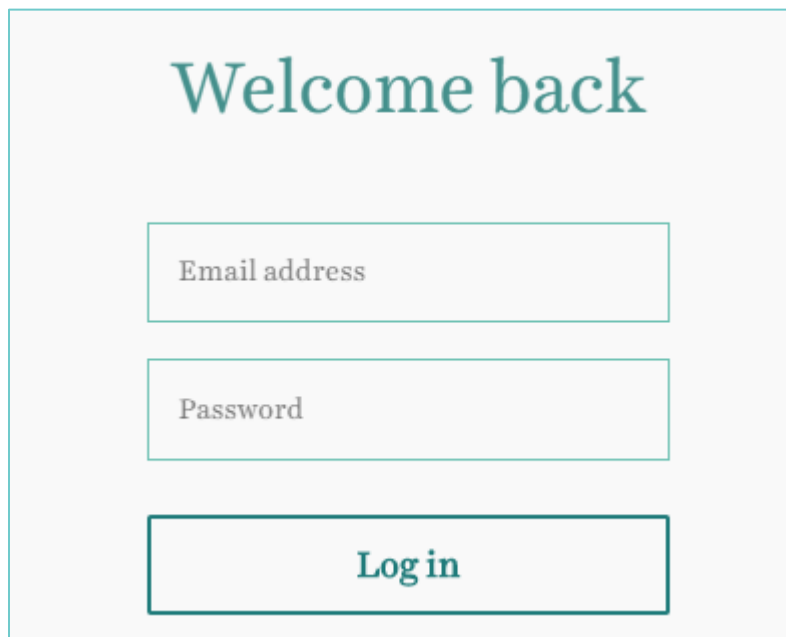
Sign up today

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.

Do not implement form submission, client-side validation, or server-side validation.

Your form must contain exactly two fields. Do not add additional fields and do not remove any fields. You will include “Email” and “Password”.



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 Tailwind CSS, daisyUI, or Bootstrap in this assignment.

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-ejs-layouts**. Your views **must** be created with **EJS**. You will have three 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.
Data Module <ul style="list-style-type: none">Contained in a separate Node.JS module named "modules/mealkit-util.js". Includes only one private/local variable, no private/local functions, and three public/exported functions.Contains a local variable that stores an array of six or more meal kits across two categories. Array only contains meal kit objects as described in the specifications.Contains the getAllMealKits() and getFeaturedMealKits() functions coded as specified.			

<ul style="list-style-type: none">Contains the <code>getMealKitsByCategory()</code> function coded as specified.			
<p>Partial Views</p> <ul style="list-style-type: none">Header with nav bar as specified. Named <code>navbar.ejs</code>.Footer as specified. Named <code>footer.ejs</code>.Meal kit “card” as specified, includes all required fields. Named <code>mealkit.ejs</code>.			
<p>Views</p> <ul style="list-style-type: none">Main layout view as specified. All views use the layout file. Named <code>main.hbs</code>Registration form as specified. Named <code>sign-up.ejs</code>.Login form as specified. Named <code>log-in.ejs</code>.			
<p>Home Page</p> <ul style="list-style-type: none">Converted to a view with the necessary partial views included, as per specification. Named <code>home.ejs</code>.“Featured meal kits” data is dynamic and passed into the view.			

On-the-Menu Page <ul style="list-style-type: none">Meal kits broken by category. Named on-the-menu.ejs.Meal kits are displayed in a responsive grid.Meal kit 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: 36 Marks

Note: Half marks may be awarded.

Deductions	No Deduction (0)	Partial Deduction (1)	Full Deduction (2)
A front-end CSS or JavaScript framework was used. Only Tailwind CSS, daisyUI, or Bootstrap are allowed.	No	Exactly one	More than one
An unapproved NPM package was installed.	No	Exactly one	More than one
Something other than EJS was used to build view files.	Assignment will not be graded.		
Additional deductions may occur if the requirements outlined in this document are not followed precisely.			

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 must 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 <https://learn.senecapolytechnic.ca>, 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.