

## Introduction

*Goodfood* (<https://www.makegoodfood.ca/>) is a popular Canadian meal delivery company that creates recipes and delivers the ingredients straight to your door. You have been contracted as a full-stack programmer to develop your version, a competing product, of the *Goodfood* website. Of course, your version will not be nearly as feature rich, but it will be a great showpiece for your CV.

You must not name your website *Goodfood* but must instead come up with your own name and brand. Also, your website cannot look the same as the Goodfood website however you can use the website (or similar meal kit websites) to inspire the look and feel for your implementation.

The implementation of the website will be spanned over three assignments. In this assignment, you will focus on building an Express server, create and work with JavaScript modules, and design and implement several views using EJS templating. There is no database connectivity required at this time.

This assignment is worth 10% 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 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 **must** use a front-end CSS framework such as Tailwind CSS with 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

### Express web server set-up

Create an Express web server that listens to incoming HTTP requests on port 8080. The logic for your web server must be placed in a file called “server.js”. The file must include the code snippet provided on Blackboard.

Remember to fill in your name, student ID, and the course code (including the section code) in the header. Failure to do so will result in a mark of zero.

### Implement Route Handlers

Create route handlers that will direct users to specific pages when they navigate to the following routes. **Each route must be located at the URL specified.**

- The “home” page – “/”
- An “on-the-menu” page used to display the current week’s meal kits – “/on-the-menu”
- A registration page used to sign up to the service – “/sign-up”
- A login page used to log in (after a user has signed up) – “/log-in”

**Remember:** You will use *res.render()* to render the appropriate views.

Specifications for creating the views are provided below. First, you will need to implement a data module, the specifications for the module are provided next.

## 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: at least 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 by 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.

The array returned by the function should look like the example provided. Notice you are returning a single array of many objects. Also notice 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

You will use EJS to implement your views.

1. Create a "static folder" named *"public"* at the root of your project. This folder can be used to store images, scripts, and CSS files. Keep this folder organized by creating appropriate subfolders. Remember to install the "static" middleware in your web app (by modifying the *"server.js"* file).
2. Configure your web app to enable EJS (by modifying the *"server.js"* file).
3. Configure EJS to use a default layout view (you will create the view next).
4. Create a default layout view called **main.ejs** and place it in the *"views/layouts"* folder.
  - The layout will contain markup that is shared across all pages of your website.
  - Add the base HTML required for your web app to display correctly.
  - Configure your `<head>` and include the necessary scripts and CSS files.
  - Within the `<body>` tag:
    - Render the header and footer partial views (you will create these in a moment).
    - Render the body.
5. Create the following EJS view files to support the routes defined in your web app. Place each file in the *"views"* folder. For now, simply leave these blank, you will code them soon.
  - **home.ejs** – This is the home page; it contains the hero(s), content sections, and featured meal kits.
  - **on-the-menu.ejs** – This is the meal kits page; it displays meal kits grouped by category.
  - **sign-up.ejs** – This is the customer registration page; it allows new users to create an account.
  - **log-in.ejs** – This is the login page; it allows existing users to log in to their account.
6. Create the following partial views and refrain from adding additional ones. Place each file in the *"views/partials"* folder. For now, simply leave these blank, you will code them soon.
  - **navbar.ejs** – This is the header; it contains the html used to display your logo and construct the navigation bar.

- **footer.ejs** – This is the footer; it contains the html used to construct the footer area.
- **mealkit.ejs** – This is a meal kit card; it contains the html used to construct a single meal kit. This file must not contain an iterator.

*Be careful and make sure you name all files as specified above. You are not implementing controllers in this assignment, so place all your views in the “views” folder and partial views in the “partials” folder.*

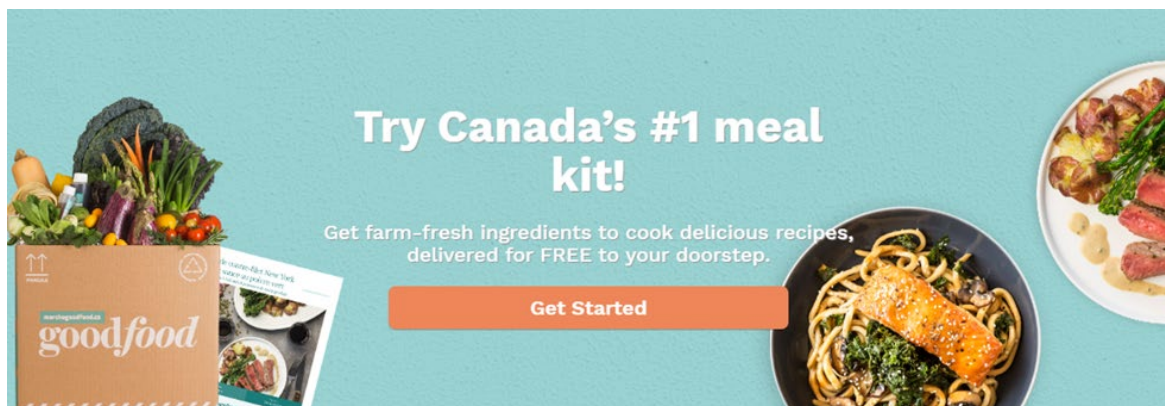
## The “Home” Page (home.ejs)

You are required to build a well-designed, responsive, home page that consists of the following sections:

- **Navigation Bar/Header (navbar.ejs):**
  - The navigation bar **must** contain a logo for your website. The logo must be an image file implemented using an `<img>` tag. Make sure that clicking the logo links the user to the home page (`/`). *Don’t forget, all `<img>` tags should include an “alt” attribute.*
  - The navigation bar **must** contain links that navigate visitors to the “on-the-menu” page, a sign-up page, and the login page. Users must be able to navigate to the home page by clicking the logo, but you can add an (optional) “home” link as well.





- **A Hero:** This section **must** have a prominent image or video element that is placed at the top of your page, below the navigation bar. Any text will be coded with HTML and not “hard-coded” into the image or video file. This will make it easier to build a responsive website. Here is an example:



- **One or More Content Section(s):** These sections **must** use a combination of grids, words, headings, and images with the sole purpose of highlighting some selling features of the website, how it will work, and/or the services that it will provide. On your phone, this will display as a single column but on a larger screen it will display all four (4) columns in the row. Each column should contain an image, heading, and description. Here is an example:


## HOW IT WORKS

Watch the video 




**We create original recipes.**

Our in-house culinary team uses premium ingredients to create unique menus and products for breakfast, lunch, and dinner.




**You choose your preferences.**

Choose from a variety of healthy, delicious meals each week that accommodate your dietary preferences.



**We deliver weekly — for free.**

Your ingredients are packaged in our temperature-conditioned boxes, so food stays fresh - even when you aren't at home.




**You eat incredible meals every day.**

Our recipes are created to cater to all cooking levels and styles. From a one-minute weekday breakfast to a delicious two-course dinner made with your own hands, our recipes cater to all cooking levels and styles.


- **Featured Meal Kits:** On your phone, this will display as a single column but on a larger screen it will display three (3) meal kits in a row. In this assignment, the data for this section will not be pulled from a database, instead it will be read from the data module you created earlier, then passed into the view. You will need to iterate through each meal kit object (using “forEach”) and render the meal kit partial view file “**mealkit.ejs**”. The HTML that constructs a meal kit card should not appear in this view.

Each meal kit must display an image, title, what is included, and a price. Here is an example:




**Chicken & Basil Meatballs**  
with Tomato Orzo & Side Salad

\$19.99



**BBQ Meatballs with Honey-Glazed Vegetables**  
Roasted Potatoes

\$16.99



**Pan-Seared Cheese Tortellini & Bruschetta**  
with String Beans Amandine & Parmesan Shavings

\$15.99



- **Footer (footer.ejs):** This section must include menu items, social media links, and any other information you deem necessary. For the menu items, you can simply create a link for each page of your site. The social media links must link to the appropriate website, but you don't need to create a profile/account on those services. Ensure there are no broken links.

The footer must contain the entire copyright statement exactly as shown here:

*By submitting this assignment I declare that this assignment is my own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) or distributed to other students.*

*@ StudentFirstName StudentLastName, StudentEmail@myseneca.ca, WEB322 Fall 2025*

Be sure to include your first and last name. For example, your professor would show the following copyright statement:

*@ Nick Romanidis, nick.romanidis@senecacollege.ca, WEB322 Fall 2025*

NAVIGATE			GET IN TOUCH
Home	Blog	Press	1(855) 515-5191
On The Menu	FAQ	Terms of Use	chef@makegoodfood.ca
Pricing	Our Ingredients	Purchase Terms &	
Goodfood WOW	Our Commitment	Conditions	
Join our team!	Gift Cards	Investors	
Our Story			FOLLOW US
			 

[On-the-Menu Page \(on-the-menu.ejs\)](#)

You are required to build a well-designed, responsive, 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



**Berbere-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 grid. 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, responsive, 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

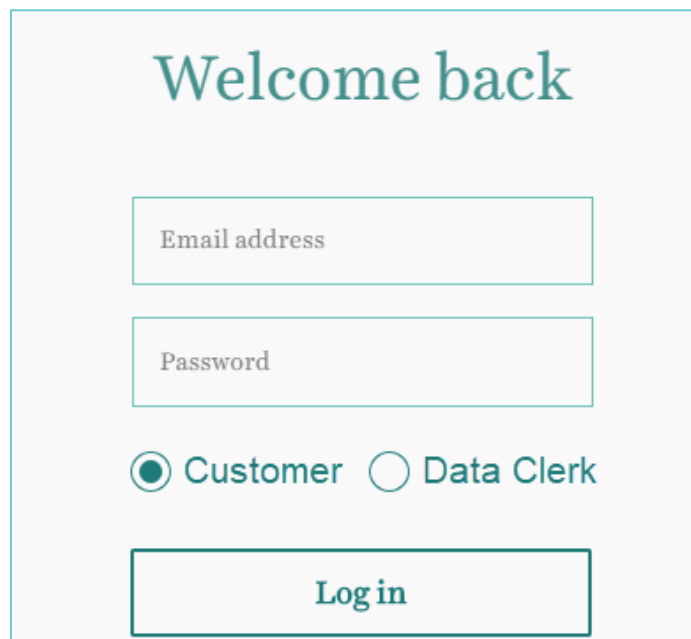
Sign up today

## Login Page

You are required to build a well-designed, responsive, 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 three fields. Do not add additional fields and do not remove any fields. You will include “Email”, “Password”, and a radio group containing two options, “Customer” and “Data Clerk”.**



Welcome back

Email address

Password

☒ Customer ☐ Data Clerk

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 must use Tailwind CSS with daisyUI or Bootstrap.

Ensure that all pages of your site follow the same branding guidelines. This means that all pages should use consistent colours, fonts, and styles.

## Reminder

All back-end functionality **must** be implemented 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.
<p>Project initialization and Express web server configuration.</p> <ul style="list-style-type: none"><li>• “npm init” was correctly run. All fields, including the author, description, and keywords, were provided, and entered correctly.</li><li>• The file “server.js” was built from the code snippet provided on Blackboard. The header at the top has been filled in correctly. The bottom portion of the file is unaltered.</li><li>• All routes have been set up with the correct URL.</li><li>• A static folder, called “public” has been correctly configured to deliver assets. It is nicely organized.</li></ul>			

<p>Data Module</p> <ul style="list-style-type: none"><li>• 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.</li><li>• 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.</li><li>• Contains the getAllMealKits() and getFeaturedMealKits() functions coded as specified.</li><li>• Contains the getMealKitsByCategory() function coded as specified.</li></ul>			
---	--	--	--

<p>Partial Views</p> <p><i>Navigation Bar</i></p> <ul style="list-style-type: none"><li>• Contains links to the “on-the-menu”, registration, and login pages. Named navbar.ejs.</li><li>• Contains a custom <u>logo image</u> that is clickable and navigates the user to the home page.</li></ul> <p><i>Footer</i></p> <ul style="list-style-type: none"><li>• Contains menu links, social media links, the academic policy disclaimer, and the copyright message. Named footer.ejs.</li></ul> <p><i>Meal Kit “Card”</i></p> <ul style="list-style-type: none"><li>• Includes all required fields.</li><li>• Does not include an iterator, used for a single card only. Named mealkit.ejs.</li></ul>			
<p>Views</p> <ul style="list-style-type: none"><li>• Main layout view, as specified. EJS properly configured, all views use the layout file. Named main.hbs</li><li>• Registration form, as specified. Named sign-up.ejs.</li><li>• Login form, as specified. Named log-in.ejs.</li></ul>			



<p>Home page</p> <ul style="list-style-type: none"><li>• Named home.ejs. Header and footer are partial views are included in the layout, not the page.</li><li>• Hero section meets minimum requirements. Text is not “hard-coded”.</li><li>• Contains one or more content sections that meet the minimum requirements.</li><li>• Contains a featured meal kits section that meets the minimum requirements.</li><li>• “Featured meal kits” data is dynamic and passed into the view.</li></ul>			
<p>On-the-Menu Page</p> <ul style="list-style-type: none"><li>• Meal kits broken by category and displayed according to the requirements. Named on-the-menu.ejs.</li><li>• Meal kit data is dynamic and passed into the view as specified.</li></ul>			
<p>Aesthetics</p> <ul style="list-style-type: none"><li>• There are no broken links and all images display correctly. You used pleasing typography, color palettes, and imagery. Branding is consistent across the app.</li></ul>			

Responsive Design	Poor (1)	Average (3)	Exceeds (6)
<ul style="list-style-type: none"><li>Overall site looks polished on all devices, specifically on smartphones, tablets, and large screens.</li></ul>			

**Total:** 54 Marks

**Note:** Half marks may be awarded.

## Important – Read Me!

Your assignment will not be marked if any of the following conditions are not met:

- Your entry point file is not named “server.js”.
- Your entry point file was not based on the provided template.
- Running “npm install” or “node server” fails.
- Your server listens on a port other than 8080.
- “node\_modules” folder is not deleted before submitting.
- An unapproved NPM package was installed.
- A framework, such as Vue, React, or AngularJS are used.
- Your name does not appear as the “Author” in package.json.
- You have not filled out the header in “server.js” with your name and other information.
- The footer does not contain the academic integrity disclaimer with your name.
- The specifications were not correctly followed.

## 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 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.

**Submissions after the due date will not be accepted. You must submit your assignment on Blackboard using the instructions above. Submissions via MS Teams, email, or any other service will be ignored.**