

Course : Server Side Web Development II (FSD202)	Amazon Project (Dynamic Website)	
Instructor: Kadeem Best	Date Given: Saturday, July 18th 2020	Date Due: Sunday, September 7th 2020

Notes for the Student: This Project is designed to give you practical experience in building Dynamic websites and Server-Side Web applications using Node.js and Express, Handlebars.

Background: You will need to have access to a code editor. You will also need a thorough understanding of HTML 5, CSS3, Javascript(Browser JS), “Node.JS”, Express, Handlebars and MySQL.

Assignment Submission Requirements

- A link to your GitHub repository of the web application.
- A link to your hosted application on Heroku.
- A link to a recorded video of you detailing the demonstration of your website. In the video, you must state your name, give an overview of the application, state the technologies used to develop the application and lastly, demo the functionality of the working web application.

Assignment Regulations

- This assignment must be done individually.
- Must be submitted to Google Classroom at midnight on the deadline date.
- You must demo the working application to the class on Sunday, August 16th 2020.

Technical Requirements

- All back-end functionality **MUST** be done using **Node JS** and **Express**.
- Your views **MUST** be created with **Express-Handlebars**
- You **MUST** use **MySQL** as your DBMS
- Your website **MUST** be responsive. Thus, it must look good on a number of different devices, specifically, smartphones, tablets, and desktops.
- You can use a CSS Framework such as Bootstrap, Bulma or Materialize CSS, CSS Tailwind to make your website responsive and aesthetically pleasing or you can also use Vanilla CSS to build your website. The option is yours.
- You can use any animation library (JavaScript or CSS) that you want. This is not mandatory, it is up to you.
- You are **NOT** allowed to use any Front-End JavaScript Frameworks, specifically, **No React, No Vue, No Angular**.
- Use SendGrid's API to send email
- Use Twilio to send SMS messages or to make phone calls.
- Use Stripe to handle payments

Detailed App Specification

Amazon is an ecommerce website that allows people to buy products from anywhere in the world. You have been contracted as a Full Stack developer to develop an Amazon clone with the below feature.

Application Architecture

1. Your application **MUST** be structured **FULLY** according to the MVC Design pattern. Thus, your views, models and controllers must be separated as per the design pattern requirements.
2. **All** sensitive credential information **must** be stored in **environment variables**. Examples include: your sendgrid access token, MySQL connection string, etc. Implement environment variables locally using the [dotenv](#) 3rd party package.

Express Web Server set up

Create an Express web server that listens to incoming HTTP requests.

Route Handlers Implementation

Ensure that you create route handlers that will direct users to specific views when navigated to.

Home Page

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

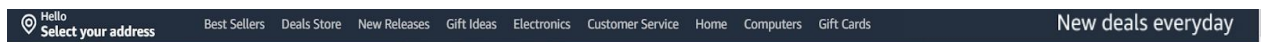
- **Header** - The header **MUST** contain the **logo** of your Amazon website, a **search bar**, an **Hello, Sign In area** (This should navigate visitors to a Sign Up and Login page) and a **shopping cart image**.

Figure 1.1



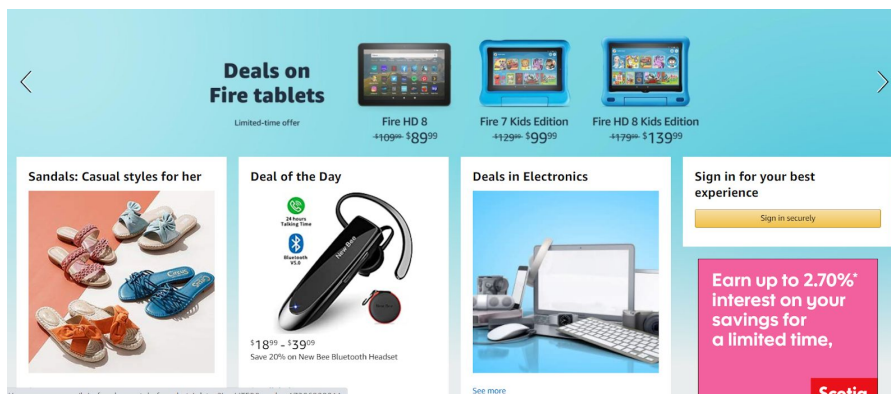
- **A Navigation bar** - The navigation bar **MUST have the following** :
 - A **Bestseller link** that navigates visitors to a Bestseller page. This page must list all the products that were set as a bestseller product .
 - **Four product categories links**. Each link **MUST** navigate visitors to a product category page that lists all the products associated with the clicked product category. **The product category links must be dynamically generated**.
 -

Figure 1.2



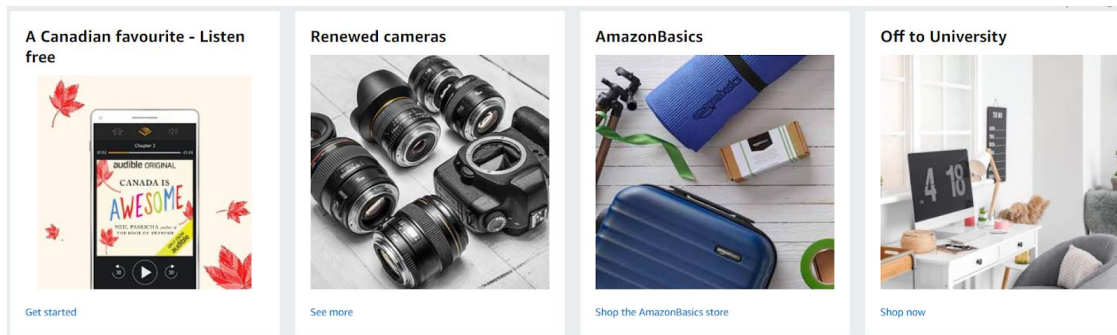
- **Hero Section/Promotional Section** - This section should present an automatic rotating carousel, as shown in **Figure 1.3**. Check the actual website to get a further understanding of how it should work..

Figure 1.3



- **Product Category Section** - This section must display at least four (4) product categories, as shown in **Figure 1.4**. **This section must be dynamic**, thus, all categories **MUST** be pulled from your MySQL database.

Figure 1.4



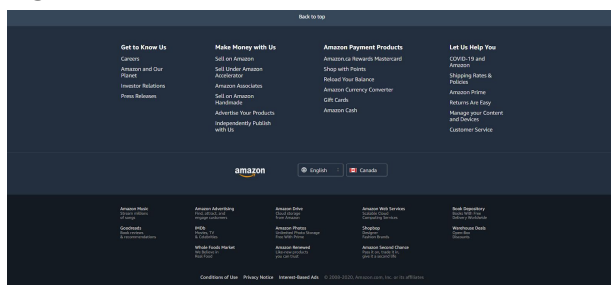
- **Best Seller Section**- This section must display a list of “**Bestseller**” products. **This section must be dynamic**, thus, all products that were set as Bestsellers **MUST** be pulled from your MySQL database and rendered to this section.

Figure 1.5



- **Footer** - This section **MUST** include footer menu items, social media links, and any other information you deem necessary.

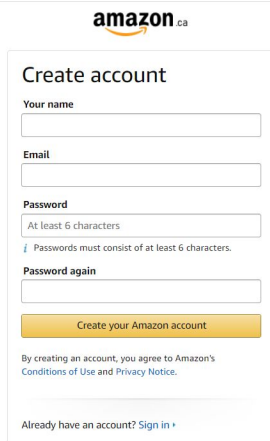
Figure 1.6



Registration Page

You are required to build a user registration page with a well designed form as shown in Figure1.7.

Figure 1.7

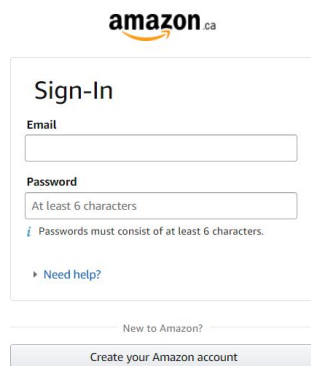


The image shows the Amazon.ca 'Create account' form. It features the Amazon.ca logo at the top. The form has a title 'Create account' and several input fields: 'Your name', 'Email', 'Password' (with a hint 'At least 6 characters'), and 'Password again'. A small blue icon with an 'i' and the text 'Passwords must consist of at least 6 characters.' is located below the password fields. A yellow button labeled 'Create your Amazon account' is positioned below the form. At the bottom, there is a link 'Already have an account? Sign in'.

Login Page

You are required to build a **well designed** login form as shown in Figure1.8 .

Figure 1.8



The image shows the Amazon.ca 'Sign-In' form. It features the Amazon.ca logo at the top. The form has a title 'Sign-In' and two input fields: 'Email' and 'Password' (with a hint 'At least 6 characters'). A small blue icon with an 'i' and the text 'Passwords must consist of at least 6 characters.' is located below the password field. A link 'Need help?' is positioned below the password field. A yellow button labeled 'Create your Amazon account' is positioned below the form. At the bottom, there is a link 'New to Amazon?'.

Server Side Validation

- You are required to implement Server-Side validation for both the login and registration form. **NO HTML5 VALIDATION OR CLIENT SIDE JS VALIDATION IS ALLOWED**
- For the login form, you are required to ONLY check for nulls (i.e, check to see if the user entered a value in the respective text fields).

- For the registration form, you have to check for nulls AND implement at least 2 complex validation criteria using regular expressions on two separate fields(For example, enforcing that the user must enter a password that is 6 to 12 characters and the password must have letters and numbers only) .
- Your form should not clear the data entered in the form if there are validation errors.
- All error messages must be rendered on their respective pages or areas and must be styled properly, as shown in **Figure 1.9**

Figure 1.9

Create account

Your name

 ! Enter your name

Email

 ! Enter your e-mail

Password

 At least 6 characters
 ! Enter your password

Password again

Create your Amazon account

By creating an account, you agree to Amazon's Conditions of Use and Privacy Notice.

Already have an account? [Sign in](#)

Registration Module

Provided that the user meets all the validation criteria, your website must create a user account in your MySQL database when they fill out the registration form and then hit the “**Create your Amazon account**” button from **Figure 1.7**.

Once the user account is created, your web application must do the following :

- Display a successful message to the user.
- Send an email to the user's email address welcoming them to the site.
- Redirect the user to a dashboard/profile page.

NOTE :

Passwords **must not** be stored in plain text in the database, thus your application must store passwords in an encrypted format. You can use a 3rd party package called [bcryptjs](#) to do the aforementioned.

Authentication Module

You are required to implement a fully functional authentication module with the following features:

- Your application must allow an Inventory Clerk and regular users (customers who want to purchase products), to log-in via the login form in **Figure 1.8**
- Upon a successful authentication(entering an email and password pair that exists in the database) **a session must be created to maintain the user state until they have logged out of the application.**
- **To implement sessions** in an Express app you can use <https://github.com/expressjs/session>
- Upon an unsuccessful authentication, the application **must** display an appropriate message (Example: **Sorry, you entered the wrong email and/or password**)
- Two users cannot have the same email address in this application.
- After successfully authenticating,the application must determine if the person logging in is an inventory clerk or a regular user and will be redirected to their respective dashboard.
- A regular user will be directed to a user dashboard and an inventory clerk will be directed to an inventory clerk dashboard.
- Both dashboards/profile page, must show the user's name(first name and lastname) and a logout link
- The logout link must destroy the session created when the user initially authenticated.
- Specific routes can only be accessed when users are logged-in, thus those routes must be protected.

Inventory Clerk Module

You are required to implement an Inventory Clerk module that allow an inventory clerk to do the following :

1. **Create Product Categories:** The Inventory Clerk must be able to add Product categories to the database. See the following data that must be added when a product category is created:
 - a. Category Title
 - b. Description
 - c. Product Category photo
2. View a list of all created product categories.
3. Edit and change product category details.
4. **Create Products:** The Inventory Clerk must be able to add products to the database. See the following data that must be added when a product is created:
 - a. Product name,
 - b. Product price,
 - c. Product description or details,
 - d. Product category (dynamically generated)
 - e. Product quantity
 - f. Have the ability to set a product as a Bestseller (or not).
 - g. Upload a photo of the product (**to keep the project simple, only upload one image per product**).
5. Ensure that all created products that were entered into the database are populated on the Front-End of the web application, specifically on the product listing page and on the Bestsellers section of the home page. **Please note, a visitor to the web application does not need to be logged in to view the products that were created by the inventory clerk.**
6. Ensure that a user can only upload an image, i.e jpgs,gifs,pngs, for the product photo.
7. View a list of all created products.
8. Edit and change product details for a selected product that was previously created. Example, product quantity, price, etc.

Please note, an Inventory clerk must be logged-in to the web application to be able to do the aforementioned (create products, view all created products, edit and change product details)

To facilitate the logging in of the Inventory clerk and to create the project simple, manually create one Inventory clerk record in the database - **I will further expound on this in class.**

Search Module

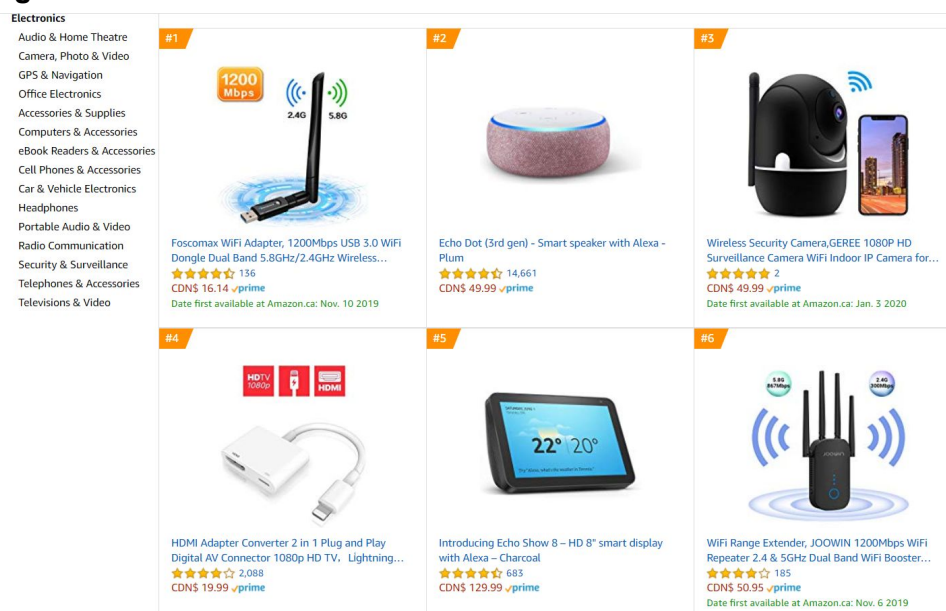
Visitors to the web application should be able to search for products via the search bar created in Figure 1.1. Visitors should be able to search by product category.

When a visitor selects a specific product category from the drop down list in the search bar and clicks the search button, the application must then present a list of all the products associated with the selected product category. Please note, a user does not have to be logged in to search for products.

Products Listing page

You are required to build a **well designed** product listing page that lists all the products in your MySQL database. The products should be presented in a grid with 3 columns, as shown in the below Figure 2.0. **You are required to implement pagination for this page. Each page in the pagination should present 12 products per page.**

Figure 2.0



Note, every product must have an image, title, price, category and a boolean attribute indicating if it is a bestseller or not.

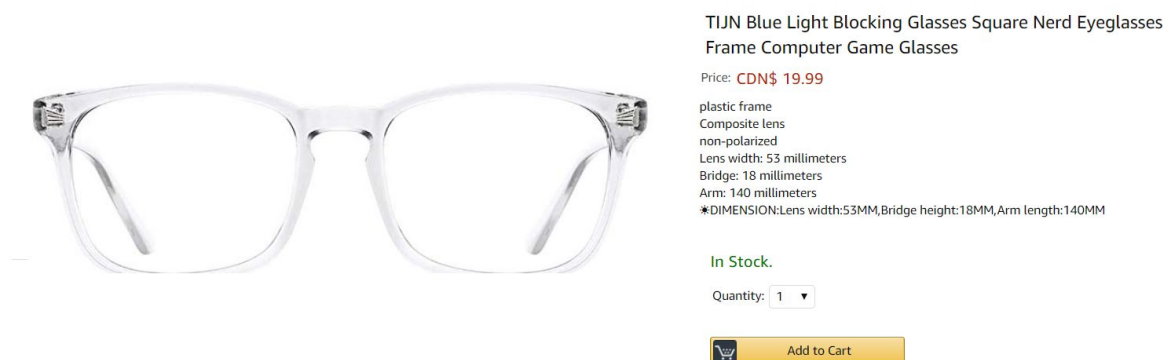
Both your Product category page and Bestseller page should implement all the above-mentioned functionality. However, unlike the product listing page that shows all products, they will both ONLY show their respective product data.

Product Description Page

When a user clicks on a particular product from the product listing page, product category page, bestseller page or bestseller section on the home page, they should be navigated to the **Product Description page** of the clicked product. The user can then add the product to their shopping cart.

Note, only logged in users should be able to “**purchase**” products by adding selected products to their shopping cart.

Sample of a Product Description Page Example :



The Product Description page of any product should list the following:

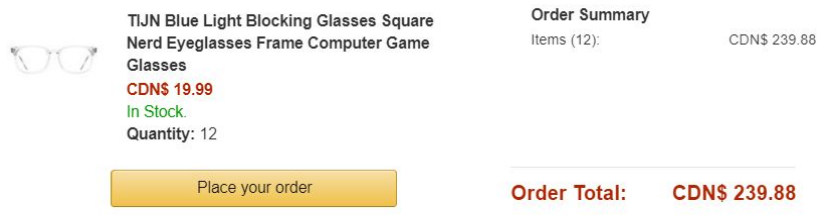
- A. Product Image
- B. Product title
- C. Product description
- D. Product price
- E. Quantity
- F. Add to Cart Button

When the user clicks the “**Add to Cart**” button, the given product will be added to the user's shopping Cart.

Shopping Cart Module

The user's dashboard should have a link to the logged-in user shopping cart . This page must display all the products that were added to the logged in user's shopping cart and how much of each product was added for the given user. Also, the page should have a "Place your order" button.

See Example

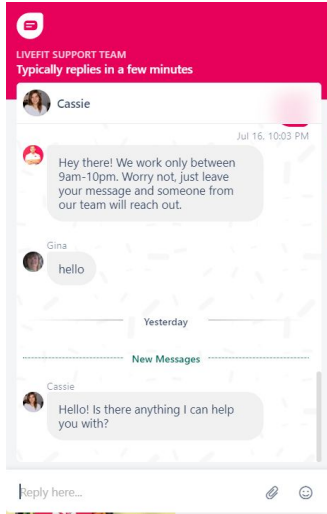


When the **"Place your order"** button is pressed, your web application should do the following :

- Capture the user's credit card information. This can be implemented as a separate page, or as a modal.
- **Receive and process the user payments using Stripe (You are to research and implement this on your own)**
- Once the user's payments is received and processed, you do the following :
 - "Clean" the user's shopping cart.
 - Send an email to the logged in user's email, indicating all the products purchased and the quantity amount for each product purchased as well as the user's order total.
 - Subtract the quantity purchased from inventory (for each product)
 - **Note, a user should not be able to add a product to their shopping cart if the quantity for the given product is 0**

Live Chat Module

You **MUST** implement a working live chat module. Have the live chat located at the bottom of the page and becomes functional when an agent is available.



You can use the below API to implement this module, however, you can research an API of your own to implement the functionality :

<https://purechat.com/support/javascript-api>

Innovative Section

I want you to find an intuitive way of implementing one of Twilio's API (SMS or Voice), Google Maps and another RESTful API in this assignment.

THE END