ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

WEB PROGRAMMING WITH NODE.JS - 502070

SEMESTER 1 – ACADEMIC YEAR 2024 – 2025

Lecturer: Mai Van Manh

Version 1.0, Last updated on Sept 29, 2024. Future updates will be shown here and highlighted in green

FINAL PROJECT

E-commerce Website

I. OVERVIEW

The final project for this course is to develop a fully functional e-commerce website. This project will test your knowledge of web development using Node.js on the backend, along with any front-end framework of your choice. The project will be completed in groups of 2-3 members. Each group will be required to design, implement, and deploy an e-commerce application that includes essential features such as user authentication, product management, and an order system. You are free to choose supporting libraries and tools to enhance your application's functionality, but the core backend must be developed using a Node.js framework like Express.js.

You will also need to deploy your application to a public hosting service (e.g., Heroku, Vercel, AWS, etc.) or, alternatively, containerize your project using Docker Compose and provide the necessary Docker configurations for it to run locally on any machine.

II. PROJECT REQUIREMENTS

1. Landing Page (home page)

- This is the first page that users will see when they visit the website. This page displays products in different categories (e.g. phones, laptops, most viewed, best selling, new products), each category only displays a number of representative products. Users are not required to log in to view and purchase products.
- The website should allow users to make purchases without logging in. They will then only need to enter their name, phone number, and shipping address. If they make a purchase without logging in, an account will be automatically created for the user (if they do not have one already), the order information will be saved, and in the future if the customer logs in, they will be able to review their previous orders.
- If the customer purchases after logging in, the default address information will be pre-filled in the checkout section. In addition, the user can also change to a different shipping address if desired.
- To keep things simple, the system only needs to support two types of users: customers and a single administrator.

2. User Management

- User Registration and Login: Allows users to create accounts, log in, and manage their profiles.
- Social Media Authentication: Users can log in using social media accounts like Google or Facebook for convenience.
- Profile Management: Users can update their personal information, change/recover passwords, and manage multiple delivery addresses and contact information for each delivery address.
- User Roles and Permissions: Defines roles like customer and admin, with varying levels of access and permissions.

3. Product Management

- Product Catalog: A dedicated page that displays a list of products with basic information such as name, price, image and short description (in listview or gridview), with a pagination mechanism applied to limit the number of products displayed at the same time.
- Product Variants: Allows products to have multiple variations (e.g., size, color) with separate stock tracking.
- Inventory Management: Tracks stock levels of each product and variant, ensuring accurate availability.
- Product Search and Filtering: Users can search for products and filter results by attributes such as price, category, or rating.
- Product Ordering: Sort the list of results (products) by some criteria like: price (ascending/descending), relevance, etc.
- Product Categories and Tags: Organizes products into categories and tags to improve navigation and SEO.

4. Cart and Checkout

- Add to Cart: Allows users to add products to a shopping cart, with the ability to update quantities or remove items.
- Cart Summary: Displays a summary of the items in the cart, including total price, taxes, and shipping fees.
- Checkout Process: Guides users through a multi-step process to enter payment and shipping details, and confirm orders.
- Guest Checkout: Lets users complete purchases without creating an account.
- Coupon Codes and Discounts: Supports applying promo codes or discounts at checkout for reduced prices.

5. Order Management

- Order Creation: Generates an order record after successful checkout, linking it to the user's account.
- Order Tracking: Allows users to track the status of their orders (e.g., pending, confirmed, shipping, delivered).
- Order History: Displays a user's previous orders, along with details like order number, date, total, and status.
- Shipping and Delivery Options: Provides users with shipping options (standard, express), with real-time shipping cost calculations.
- Payment Confirmation: Sends confirmation of successful payment, along with an order receipt.

6. Admin Dashboard

- Simple Dashboard: a high-level overview of the store's performance, key metrics, and actionable insights. This includes things like: Total number of users, number of new users, number of orders, revenue, best-selling products represented through charts.
- View order list: The administrator can view the system-wide order list, sorted with the most recent orders first. Pagination should be applied, displaying around 20 items per page. This interface should also allow the administrator to filter orders by various time ranges, such as: today, yesterday, this week, this month, or a specific date range (start-end).
- View order details: The administrator can select an order to view its detailed information (such as buyer's name, purchase time, total amount, whether a discount was applied, etc.), as well as the list of products in the order. Additionally, the admin can also change the order status (e.g., from pending to confirmed).
- Advanced Dashboard: Allows the administrator to view information in the form of charts, such as a bar chart comparing revenu/profit/the number of orders across the past 12 months/weeks.
- User Management: Admins can view and manage all users, including banning or updating user information.
- Product Management: Admins can add, update, or delete products, manage categories, and handle inventory from a central dashboard.
- Order Management: Admins can view, update, and process orders (e.g., changing status from pending to confirmed).

7. Marketing and Promotions

- Coupon management: Admin can view the list of coupons along with related details (creation time, discount value, whether it has been used or not, expiration date), and also has the ability to create additional coupons.
- Product Reviews and Ratings: Anyone can write a review, but only customers who have purchased the product can rate it.
- Loyalty Programs: Earn 5% of the total order value in points. For an order worth 1,000,000, customers will earn 50 points. These points can be redeemed for 50,000 on the next purchase and never expire.

8. Deployment

You are required to choose one of the two methods bellow to deploy your team's project. If this deployment is not completed, the project will not be graded:

1. Public Hosting:
   - The website should be deployed on any public cloud hosting platform (e.g., Heroku, Vercel, AWS, Netlify).
   - Provide the public URL to access your project.
   - Ensure that the website functions properly, just as it does in the local environment, as it will serve as the basis for evaluating your grade. Don't forget to provider username/password for the admin account.
2. Docker Compose:
   - If you choose not to deploy to a online hosting service, you must containerize the application using Docker Compose.
   - Each component (frontend, backend, database, etc.) should be in separate containers.
   - Provide the docker-compose.yml file, with clear instructions for running the project locally.
   - Make sure you have tested this Docker Compose setup and that it is functioning properly before submitting your project.

9. Other requirements

- UI/UX: The web app must have a clear, user-friendly design with intuitive navigation. Focus on user experience, quick load times, and easy interaction with elements.
- Team Collaboration: Team members must work together using version control (e.g., Git), dividing tasks and ensuring smooth integration of contributions. Regular communication is essential.
- Responsive Design: The web app should be responsive, adapting seamlessly to different devices and screen sizes. Use frameworks like Bootstrap or CSS Grid to ensure compatibility.
- Horizontal Scaling: Design the app for horizontal scaling, allowing it to handle more traffic by adding servers. Implement stateless architecture, load balancing, and consider microservices.

10. Bonus features

Successful implementation will earn an additional 0.5 points, with a maximum of 2 points for all bonus features

- Separate back end and front end, develop back end as rest api and use a front end framework such as: React, Vue, Angular.
- Using CICD pipeline during development: Jenkins, Github Actions, Circle CI, GitLab CI/CD, or one of the top 3 Cloud provider solution.
- Deploy the system following a Microservices Architecture. In addition to the front end and database, there must be at least three other services. You must also demonstrate asynchronous communication and decoupling between services using an intermediary channel, such as RabbitMQ or Redis.
- Integrate AI-related features, such as a smart chatbot that can suggest products directly related to this system, product search by image upload, and Sentiment Analysis for reviews and feedback.
- Integrate ElasticSearch for Product Search: Implement ElasticSearch to enhance product search speed and efficiency. This integration should allow for fast, relevant search results, improving the overall user experience.
- Integrate CDN for Static Asset Caching: Utilize a Content Delivery Network (CDN) to cache static assets such as images, stylesheets, and scripts. This will improve load times and reduce server load by delivering content from geographically distributed servers.

If your team successfully implements any of these bonus features, you must clearly document it in the self-assessment form, the README file, and the product introduction video. This ensures that your contributions are recognized and properly communicated to reviewers.

III. RUBRIK

ID	FEATURES		1	2	3
		POINTS	0 PT	1/2 PTS	FULL POINTS
CUSTOMER FEATURES – 6.0 points					
1	Social Media Authentication	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
2	View profile page	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
3	Change password	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
4	Password recovery	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
5	Manage multiple delivery address and contact information	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
6	View purchase history (login required)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
7	View purchase details (login required)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.

8	Landing Page	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
9	Product Catalog	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
10	Pagingation in Product Catalog	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
11	View product details	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
12	View product variants (in the same detail page)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
13	View products by category (with pagination as well)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
14	Product search by keyword	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
15	Product filtering	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
16	Product ordering (e.g. by price, time)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.

17	Display shopping cart	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
18	Update shopping cart	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
19	Checkout process	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
20	Using coupon when making purchase	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
21	Email notification (after placing an order)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
22	Product review (comment)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
23	Product rating	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
24	Loyalty Programs	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
ADMIN FEATURES – 2.0 points					

25	Product Management	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
26	Simple Dashboard	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
27	Advanced Dashboard	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
28	View order list	**0.5**	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
29	View order details (and modify order status)	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
30	Advanced Dashboard	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
31	Coupon management	0.25	The implementation is missing, non-functional, or contains major issues preventing basic use	The functionality is present but incomplete, with issues such as poor input validation, security flaws, or lack of adherence to standards.	The functionality is fully implemented, secure, and meets standards, with no major issues or flaws.
OTHER REQUIREMENTS – 2 points					
32	UI/UX	**0.5**	Basic UI and UX are present but inconsistent, with some usability issues and limited visual appeal.	The website has some user-friendly elements but lacks consistency in design and navigation. While basic usability is present, there are still significant areas for improvement that could enhance user experience.	UI and UX are well-designed, intuitive, and visually appealing, providing a seamless and engaging user experience.

33	Teamworking (Github insights)	**0.5**	No evidence of teamwork on GitHub; contributions are sporadic or missing, with no early or regular commits.	Some teamwork is visible with occasional commits, but work distribution is uneven, and there are delays or a lack of early contributions.	Evidence of effective teamwork with balanced work distribution, early and regular commits, and proactive collaboration throughout the project.
34	Responsive	**0.5**	The website is not responsive; it does not adjust to different screen sizes or devices. Users experience layout issues, making navigation difficult.	The website has some responsive elements but is not fully optimized. Certain pages or components may not display correctly on all devices, leading to inconsistent user experiences.	The website is fully responsive, providing an optimal user experience across all devices. All elements, layouts, and functionalities adjust seamlessly to different screen sizes, ensuring easy navigation.
35	Horizontal scaling	**0.5**	The application is not designed for horizontal scaling; it relies on a single server. This limits performance and may cause downtime under increased load.	The application shows some potential for horizontal scaling but is not fully implemented. Some components can be distributed across multiple servers, but there may be issues with load balancing or state management	The application is fully designed for horizontal scaling. It effectively utilizes multiple servers, employs load balancing, and maintains a stateless architecture, allowing for seamless handling of increased traffic.

The above description is only relative in nature and cannot list detailed instructions for each feature on correct or incorrect implementation. However, during grading, features must be relatively well-developed to be eligible for full points. Groups must be active in referring to related applications and applying their daily app usage experience in their assignments. For examples:

- Displaying product price:
  - Poor approach: Display data directly from the database as "75299000", which is hard to read.
  - Better approach: Format as "75,299,000^{đ}" with commas and currency, improving clarity and readability.
- When showing order list:
  - Poor approach: Only uses "select all" from the database, displaying data in ascending order. New orders appear at the bottom, with no pagination, making it difficult to search. Important information like total amount, buyer name, status, and date formats are not clearly displayed.
  - Better approach: Sort data in descending order by date, so new orders are on top. Implement pagination for easier viewing, ensuring all important information is displayed clearly with proper formatting and appropriate color coding.

IV. OUTPUT REQUIREMENTS

- Required submission components include:
  - The "**source**" folder:
    - **if you do not use docker compose**: the source folder should contain the entire source code of the web app (e.g. frontend, backend), along with relevant database files. Ensure that this source code can be run on the teacher's computer. The project needs to be "cleaned" to remove unnecessary content before submission and to reduce the size of the compressed file.
    - **if you use docker compose**: This folder must contain all source code for the necessary modules, the docker-compose file, and instructions for running the application on the instructor’s machine (e.g., where to run npm install and docker-compose up). Ensure thorough testing is completed before submission.
  - Introduction video "**demo.mp4**": A team representative should record a screen presentation showcasing the group's application, highlighting ALL features based on self-assessment. The video, with a minimum resolution of 1080p, should have clear and audible sound without theoretical explanations.
  - The “**git**” folder: Contains screenshots that demonstrate collaboration between team members on the github or gitlab repo. Multiple screenshots may be submitted, as long as there is evidence of effective teamwork with balanced work distribution, early and regular commits, and proactive collaboration throughout the project. Evidence of teamwork must clearly indicate that the project duration from the start to the present is at least one month, during which each member must have at least two commits per week.
  - **Readme.txt** file: Provide all necessary information for the evaluation process, such as project building and running instructions, URL + server login information (if applicable), and usernames/passwords for accounts with pre-loaded data for assessment. Include any relevant notes on building, running, and using the application for teachers to reproduce the project. If your team implements some optional features (which get extra points), it should be clearly stated in the readme file.
  - **The “Bonus”** folder**:** The bonus folder should include a description of the extra features your team implemented for additional points, along with evidence. Organize the information clearly, concisely, and convincingly.
  - **Rubrik.docx:** This file lists the required features for the project. Teams should self-assess their completion level in this file. The instructor will provide this file at the submission time. The file will also include the public URL to the web application and any required username/password for login.

- Organize all the above contents into a folder named **id1_fullname1_id2_fullname2**, then compress this folder in ZIP format with the same name, e.g., id1_fullname1_id2_fullname2.zip. A team representative should submit this file on the online learning system as instructed by the course instructor.
- Teachers do not accept sumissions via email, only elearning is accepted.

IV. IMPORTANT NOTES

- The Final Project must be implemented using a **Nodejs** project using the **Javascript** programming language. Groups are allowed to use any libraries within the framework of a Flutter project.
- Teams may use online services like Firebase or equivalent services, provided they function correctly during project evaluation. Ensure to supply all necessary access information.
- If your team submits an unrelated project, it will not be graded, and the entire team will receive a score of 0. For example, if your team uses the source code from a different e-commerce site and only implements a few features related to this assignment while most features are unrelated, the project will receive 0 points.
- The Essay is entirely independent of the Final Project. Therefore, all team members must participate in both the Essay and the Final Project. The Essay will be assessed by the lab instructor, while the Final Project will be assessed by the theoretical instructor.
- Groups are prohibited from sharing code with each other, obtaining source code from the internet, and must take responsibility for protecting their group's source code. Groups with similar source code (verified by specialized software) or code found online, even if only in part, will receive a score of 0 for all members, regardless of which group shared or received the code.
- Failure to submit source code will result in the entire team receiving a score of 0.
- If the team does not fill out and submit the **rubrik.docx** file, the submission will not be graded.
- If your team fails to deploy the project (e.g., public hosting or Docker Compose) or does not provide an introductory video, the submission will not be graded.
- Deductions will also apply in the following situations:
  - Late submission: 1-day late deducts 1 point. Submissions late by 1 second to less than 1 day are considered 1 day late.
  - Complex project configuration without specific instructions for instructors to compile and run the program: Deduction of 2 points.
  - Submit the entire project without performing a clean to remove unnecessary files: 0.5 point.

- Failure to provide necessary grading information, such as missing usernames/passwords, incorrect file naming, or not submitting required content: 1.0 point.