Course: SDN302 Server-Side development with NodeJS, Express, and MongoDB	**Contribution: 20% of course**
This assignment should take an average student who is up-to-date with tutorial work approximately 5 weeks	
Learning Outcomes: CLO1, CLO2, CLO3, CLO4, CLO5, CLO6, CLO7, CLO8 - Develop a fully functional **Node.js backend application** using **Express.js**. - Implement **RESTful APIs** with proper routing, middleware, and database integration. - Utilize **MongoDB & Mongoose** for data persistence. - Implement **user authentication & security** - Run the backend application locally on your machine using Node.js and MongoDB. If deployment is required, use Firebase Functions (for serverless functions) OR deploy a full Express.js server on a cloud platform (e.g., Heroku, Render, Railway). - Follow best practices in **system architecture, API design, and database modeling**.	

Plagiarism is presenting somebody else's work as your own. It includes copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing or buying coursework from someone else and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with failure of the course.
All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

I. Assignment Submission Requirements

- Source code zipped in .zip file
- Document Report for NodeJS application, Database MongoDB, Slide to Presentation
- *Lack one of them, student will not be allowed to do assignment's demonstration.*

II. Assignment Topic

- Develop a Full-Stack E-Commerce API Using Node.js & MongoDB

In this assignment, students will build a RESTful API for an E-Commerce (or other field) platform that allows users to browse, purchase, and manage products (or other business logic).

III. Project Requirements

You must provide a complete design and develop an Backend and Frontend application. Your Document Report should include:

1. **Team Formation**
    a. A team introduction: A brief introduction about the project
    b. Each group consists of 3-5 members.
    c. Each member must be responsible for at least **one key feature** of the project.
2. **A case study:** describes the system that you will implement (case study is not certain to be too detailed. It's just a paragraph so that the reader can understand the system that will be presented). You should mention the important issue, other detailed information can be presented in the form of Business rules.
    a. A brief description of the system.
    b. Identify main challenges and solutions for developing an e-commerce backend.
3. **Business analyse / System design:** a detailed description of how you analyse business and design the system. You should provide your understanding of your architecture and how your application implemented.
   + All functions in your application should be described
   + The database design or schema structure should be clearly defined.
   + A detailed description of any new technologies you find out (not in school) to develop applications.

a. **Business Analysis:**

- Identify user roles (**Admin, User**).
- Describe key business rules (**e.g., Orders cannot be canceled after payment**).

b. **System Design:**

- Design the **API** (**RESTful API, list of endpoints**).
- Design the **database** (**ERD, MongoDB Schema**).
- Provide a **system architecture diagram**.

4. **Backend Development (50%)**
   Each member should implement at least **one key feature**. Best Practices Required:
   - Use **Express.js Router** for modular structure.
   - Implement **middleware** (e.g., error handling, Implement user authentication & security).
   - Use **MongoDB** for cloud database hosting.
   - Follow **REST API principles**.
   - **Code Comments:** All code must be **well-documented** with meaningful comments

5. **Frontend**

- Frontend implementation is required using React.js. The frontend must interact with the backend via API calls. It must include authentication, product listing, and order management

6. **Deployment & Security**

- Deploy the backend application using Firebase Functions OR deploy a full Express.js server on a cloud platform (e.g., Heroku, Render, Railway) => Use the API link after successful deployment. Or Run the backend application locally on your machine
- Implement **environment variables (.env) for sensitive data**.
- Configure **CORS for frontend access**.
- Ensure **input validation & error handling**.

7. **Demo of your application**: Thorough all functions and explanations.

- **Live demo of the API functionality.**
- **Answer technical questions from the instructor.**

8. **Conclusion and Discussion**: the pros and cons of the application. What you've learned anything through the development of this application. In the future, if having more time, what would you do to improve it?

IV. Evaluation Criteria:

ask	Score	Conditions
Case Study	5%	Coherent explanation of the business requirements.
Business Analysis & System Design	15%	Well-structured API & database design, clear architecture.
Backend Development	50%	Fully functional API, well-structured code, proper database integration.
Deployment & Security	15%	API deployed on Firebase Functions OR a cloud platform (Heroku, Render, Railway). Security measures implemented.
Documentation & Presentation	15%	Clear, professional report & well-explained demo.

Evaluate the contribution of each member during the project

Topic	**Team Effort**	**Member 1**	**Member 2**	**Member …**
Case Study Analysis	100%	Ex: 40%	Ex: 30%	Ex: 30%

Business analysis	100%			
System design	100%			
Implementation (code)	100%			
Documentation	100%			

Your implementation:

- All source code must be zipped and uploaded to Edunext system.
- Code's comments are required

Your demonstration (15 minutes):

- You will be required to briefly PPT demonstrate your system (slide should be prepared). Prepare to answer the lecturer's questions

Evaluation		
Task	**Score**	**Condition**
Case study	5%	A case study certainly coherent
Business analysis,	15%	All functions are designed as standard and structured in accordance with the business rules. Each function should be described in detail, accompanied by screenshots of the respective screens. A detailed description of any new technologies you find out (not in school) to develop applications
System design	5%	A design architecture, database design or schema structure is expressed
Conclusion and discussion	5%	The personal opinions should be clarified. The knowledge learned should be highlighted.
Demonstration	70%	Programs comply with the proposed design.

		Operation with good quality. Each member will be responsible for demonstrating the functionality they have implemented.

- **Assignment Sample:**
  - **Objective:** Develop an NodeJS application for Product Sale. The application is used by customers, helps customers to view/buy products of a store

  - **Main Functions:**

Feature	Description	Assigned To
User Authentication	JWT-based authentication, bcrypt password hashing	Member 1
Product Management	CRUD operations for products	Member 2
Cart & Order Processing	Add to cart, checkout, order placement	Member 3
Payment Integration	(Optional) Simulate payment processing	Member 4
User Profile Management	Update profile, order history	Member 5

  - **Sample API Endpoints**

    a. **User Authentication (Member 1)**
       - **POST** `/api/auth/signup` → Register a user.
       - **POST** `/api/auth/login` → User login & JWT token issuance.

    b. **Product Management (Member 2)**
       - **GET** `/api/products` → Get all products.
       - **POST** `/api/products` → Create a new product (Admin).
       - **PUT** `/api/products/:id` → Update product details (Admin).
       - **DELETE** `/api/products/:id` → Remove a product (Admin).

    c. **Cart & Order Processing (Member 3)**
       - **POST** `/api/cart` → Add item to cart.
       - **GET** `/api/cart` → View cart items.
       - **POST** `/api/order` → Place an order.

    d. **Payment Integration (Optional - Member 4)**

- 🔷 **POST** `/api/payment` → Process payment (simulate with Stripe API).

e. **User Profile (Member 5)**

- **GET** `/api/user/profile` → Get user details.
- **PUT** `/api/user/profile` → Update user info.