**PROJECT TITLE:E-COMMERCE APPLICATION ON IBM CLOUD FOUNDRY**

**Problem Definition:** The project is to build an artisanal e-commerce platform using IBM Cloud Foundry. The goal is to connect skilled artisans with a global audience, showcasing their handmade products and providing features like secure shopping carts, payment gateways, and an intuitive checkout process. This involves designing the e-commerce platform, implementing necessary features, and ensuring a seamless user experience.

**Features:**

1. Platform Design: Design the platform layout with sections for product categories, individual product pages, shopping cart, checkout, and payment.
2. Product Showcase: Create a database to store product information such as images, descriptions, prices, and categories.
3. User Authentication: Implement user registration and authentication features to enable artisans and customers to access the platform.
4. Shopping Cart and Checkout: Design and develop the shopping cart functionality and a smooth checkout process.
5. Payment Integration: Integrate secure payment gateways to facilitate transactions.
6. User Experience: Focus on providing an intuitive and visually appealing user experience for both artisans and customers.

Project Objective:Platform Design:An ecommerce platform is the content management system (CMS) and commerce engine websites use to manage catalogued products, register purchases and manage a users relationship with an online retailer. It doesn't matter if your business is large or small, B2B or B2C, selling tangible goods or providing remote services.

**Product Showcase:**

1. High Quality Photos. High-definition and well-staged photos make all the difference to a product listing page. ...
2. Great Product Descriptions. ...
3. Categorise Logically. ...
4. Product Reviews. ...
5. Use Search Engine Optimisation (SEO) ...
6. Use Videos.

User Authentication: There are three basic types of authentication. The first is knowledge-based — something like a password or PIN code that only the identified user would know. The second is property-based, meaning the user possesses an access card, key, key fob or authorized device unique to them. The third is biologically based.

Shopping Cart and Checkout:An ecommerce shopping cart is software that enables customers to make purchases on a website. It takes the customer's payment, verifies the payment method, processes the transaction, and completes the checkout process, all in a matter of seconds.

Payment and Intergration:As the phrase implies, integrated payments connect the critical payment processing function with other vital business systems and software. When all the systems are compatible, it's easier for a merchant to accept payments and consolidate important data.

User Experience:Ecommerce user experience (UX) is putting yourself in the shoes of the user and determining what will provide them with an enjoyable, informative, and easy to use experience. Ecommerce user experience is all about caring for the customer and ensuring a feel-good factor that in turn contributes to the sale of a product.

Conclusion: The conclusion paragraph of an eCommerce website would typically summarize the main points of the website, such as the products or services offered, any promotions or deals currently available, and any unique features or benefits that the website offers.

Innovation

Product Review

1. Put your reviews front and center.
2. Get on the same sites as your customers.
3. Follow up with buyers.
4. Ask the right questions.
5. Contact those who leave you glowing reviews.
6. Contact those who leave you negative reviews.
7. Run a contest.
8. Reward those who review.

Wishlist

E-commerce best practices are those that can bring more new customers and optimize revenue. You can add a Share button to the wishlist page allowing shoppers to introduce their desired items to others. This excellent free method helps you attract more visitors, make them aware of your brand, and promote your products.

### Personalized Recommendations

Ecommerce personalization lets you treat each customer like a VIP. Personalization platforms use AI and machine learning-driven algorithms to track onsite behavior and customer data points in real-time to deliver a unique experience to each site visitor.

* Strategic product recommendations.
* Streamlined shopping.
* User-generated content.
* Suggested searches.
* Abandoned cart follow up.
* Feedback.
* Additional product emails.
* Special deals.

E-Commerce personalization tools increase sales, customer engagement, and customer loyalty. 45% of consumers say they are more likely to shop on an E-Commerce site that offers personalization.

Personalized product recommendations are when a site shows a selection of product recommendations that's unique to the individual visitor, based on their behaviors and profile. This is almost always based on a machine learning algorithm.

* Provide personalized recommendations.
* Suggest additional relevant products.
* Tailor the digital shopping journey.
* Offer consultative customer service.
* Design a personalized email campaign.
* Offer continuous shopping for returning customers.

#### Conclusion

E-commerce still represents one of the business methods that take advantage if done the right way, even if the stock market and commodities fell, but E-Commerce still able to survive and receive high transaction.

Development Part

e-Commerce platform on IBM Cloud Foundry

How to begin building your artisanal e-commerce platform and setting up a database on IBM Cloud Foundry.

1. IBM Cloud Foundry Setup:

- First, ensure you have an IBM Cloud account and have logged in.

- Access IBM Cloud Foundry and create a new Cloud Foundry application to host your platform.

2. Platform Layout Design:

- Sketch out the layout of your e-commerce platform, including the homepage, product pages, and checkout process. Consider user experience and intuitive navigation.

3. Database Setup:

- Choose a suitable database service from IBM Cloud, such as IBM Db2 or IBM Cloudant, based on your project requirements.

- Create a new database instance and configure it to store product information. Define the necessary fields like product name, description, price, and images.

4. Connect Platform and Database:

- Integrate your platform with the database by configuring database connection parameters in your application code.

- Implement functions to read, write, update, and delete product information from the database.

5. Testing:

- Test your platform thoroughly to ensure that it can retrieve and display product information correctly from the database.

- Verify that the layout and design are user-friendly and responsive on different devices.

6. Deployment:

- Once testing is successful, deploy your artisanal e-commerce platform on IBM Cloud Foundry. Ensure it's accessible and functional in the live environment.

7. Submission:

- Document your design choices, database schema, and the steps you took to set up the platform and database.

- Prepare any necessary documentation or reports required for your assignment submission.

To create a database in IBM Cloud Foundry, you typically use a database service provided by IBM Cloud. Here's an example of creating a database using IBM Db2 on Cloud, which is one of IBM's database offerings. Please note that the exact steps and commands might vary based on the specific service you are using and the command-line interface you prefer (Cloud Foundry CLI or IBM Cloud CLI).

Using IBM Cloud Foundry CLI:

1. Log in to IBM Cloud:

cf login -a https://api.ng.bluemix.net

2. Create a Db2 Service Instance:

cf create-service dashDB Entry my-db-instance

Here, `dashDB` is the service name, `Entry` is the plan, and `my-db-instance` is the name you are giving to your service instance.

3. Bind the Service to Your Application:

cf bind-service YOUR-APP-NAME my-db-instance

Replace `YOUR-APP-NAME` with the name of your Cloud Foundry application.

4. Restage Your Application:

cf restage YOUR-APP-NAME

This step is necessary to ensure your application recognizes the newly bound database service.

5. Access Database Credentials:

cf service-key my-db-instance my-db-credentials

Replace `my-db-instance` with your service instance name and `my-db-credentials` with the name you want to give to your credentials.

This command will provide you with the necessary connection credentials (such as hostname, port, database name, username, and password) that your application needs to connect to the Db2 database.

Please adapt these commands based on the specific database service and plan you are using, and make sure to refer to the official IBM Cloud documentation for the most accurate and up-to-date instructions. The combination of the code set and territory values must be valid.

Not all collating sequences are valid with every code set and territory combination.

The table space definitions that are specified on CREATE DATABASE apply to all database partitions on which the database is being created. They cannot be specified separately for each database partition. If the table space definitions are to be created differently on particular database partitions, the CREATE TABLESPACE statement must be used.

The combination of the code set and territory values must be valid.

Not all collating sequences are valid with every code set and territory combination.

The table space definitions that are specified on CREATE DATABASE apply to all database partitions on which the database is being created. They cannot be specified separately for each database partition. If the table space definitions are to be created differently on particular database partitions, the CREATE TABLESPACE statement must be used.

When defining containers for table spaces, $N can be used. $N will be replaced by the database partition number when the container is actually created. This is required if the user wants to specify containers in a multiple logical partition database.

E-commerce Application on IBM Cloud Foundry

Building an e-commerce platform with user authentication, a shopping cart, and checkout functionality is a complex task. It typically involves multiple components and technologies. I'll provide a high-level overview of the steps and technologies involved in implementing these features.

**1. User Authentication:**

User authentication is crucial for securing user data and providing personalized experiences. Here's how you can implement it:

**a. User Registration:**

* Create a registration form where users can enter their information like name, email, password, and address.
* Use a backend server (e.g., Node.js, Python with Django, Ruby on Rails) to handle user registration.
* Hash and salt passwords to store them securely in your database.

**b. User Login:**

* Develop a login form for users to enter their credentials.
* Implement a session management system or use JSON Web Tokens (JWT) to handle user sessions.

**c. Password Recovery:**

* Add a feature for users to reset their passwords through email verification.

**2. Shopping Cart:**

A shopping cart is where users can add and manage items they want to purchase.

**a. Cart Data Structure:**

* Create a data structure to store cart items for each user.
* Use a database (e.g., MySQL, MongoDB) to store this information.

**b. Cart Operations:**

* Implement functionality to add, remove, and update items in the cart.
* Calculate the total price of items in the cart.

**c. User Interface:**

* Design a user-friendly cart page for users to view and manage their items.

**3. Checkout Functionality:**

Checkout functionality involves order summary, payment processing, and order confirmation.

**a. Order Summary:**

* Create an order summary page displaying the items in the cart, their prices, and the total order cost.

**b. Payment Processing:**

* Integrate a payment gateway (e.g., Stripe, PayPal) to handle payment processing.
* Implement secure payment handling and store transaction records.

**c. Order Confirmation:**

* Show users an order confirmation page and send a confirmation email.
* Store order details in a database.

**4. Security:**

Ensure the security of user data, payment information, and your platform in general. This includes using HTTPS, encryption, and input validation to prevent security vulnerabilities.

**5. User Profiles:**

Create user profiles where users can view their order history, shipping addresses, and payment methods.

**6. Testing:**

Thoroughly test the entire e-commerce platform, especially the checkout process, to ensure a smooth user experience and identify and fix any bugs.

**7. Responsive Design:**

Make sure the platform is responsive and works well on various devices and screen sizes.

**8. Legal and Compliance:**

Comply with data protection regulations, such as GDPR, and other e-commerce laws applicable in your region.

**9. Scalability:**

Design your platform to be scalable so it can handle increased traffic and data over time.

**10. Hosting and Deployment:**

Select a reliable hosting service, and deploy your platform. Use continuous integration and continuous deployment (CI/CD) for efficient updates.

Remember that building an e-commerce platform is a significant undertaking, and it's essential to keep security, scalability, and user experience in mind throughout the development process. It may be helpful to work with a team of developers, designers, and possibly legal experts to ensure the platform's success.

Top of Form

Implementing user registration and authentication features using a backend server (e.g., Node.js or Python) is a fundamental step in building your e-commerce platform. Here's a basic outline of how to achieve this:

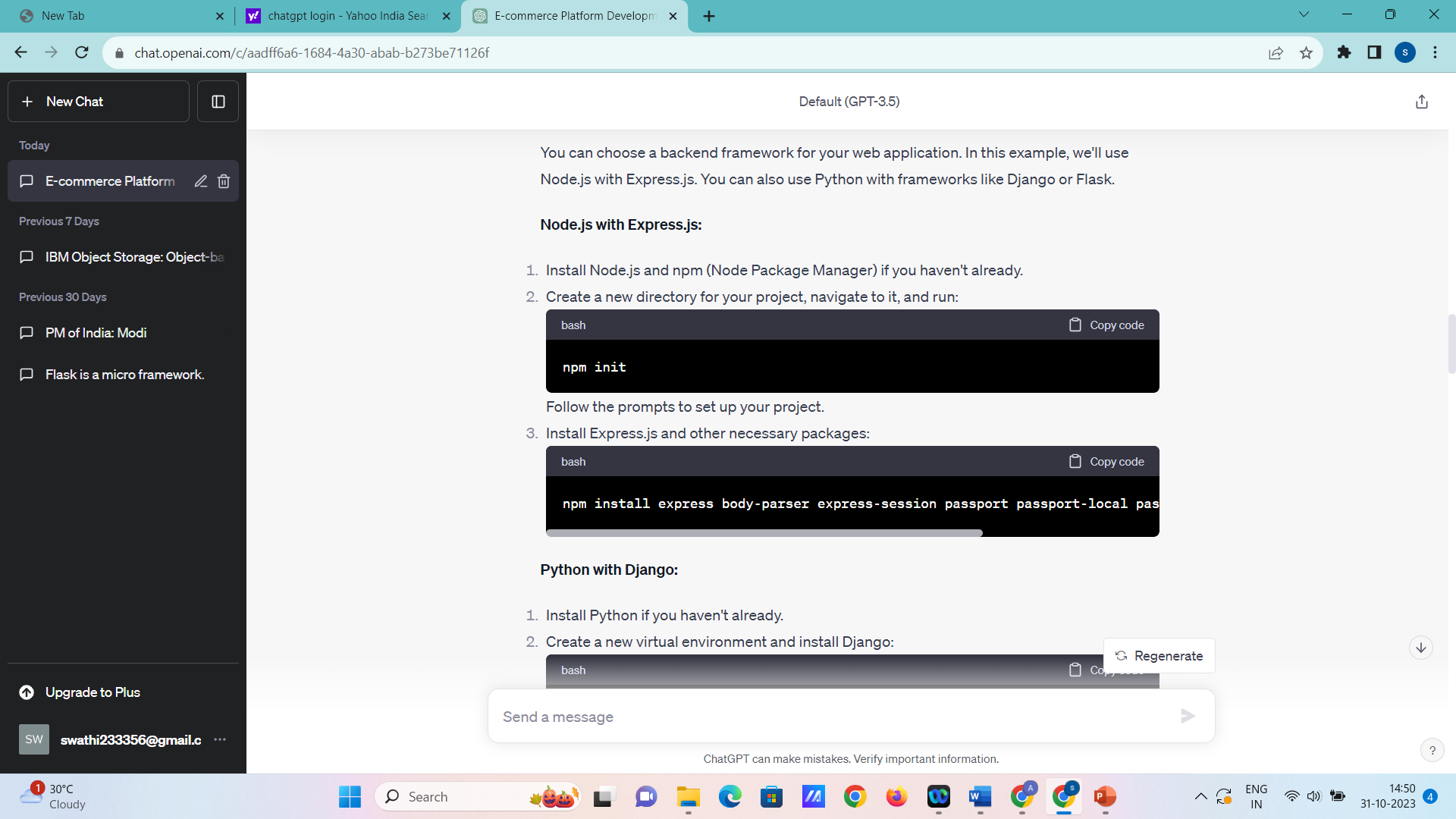
**1. Set Up Your Backend:**

You can choose a backend framework for your web application. In this example, we'll use Node.js with Express.js. You can also use Python with frameworks like Django or Flask.

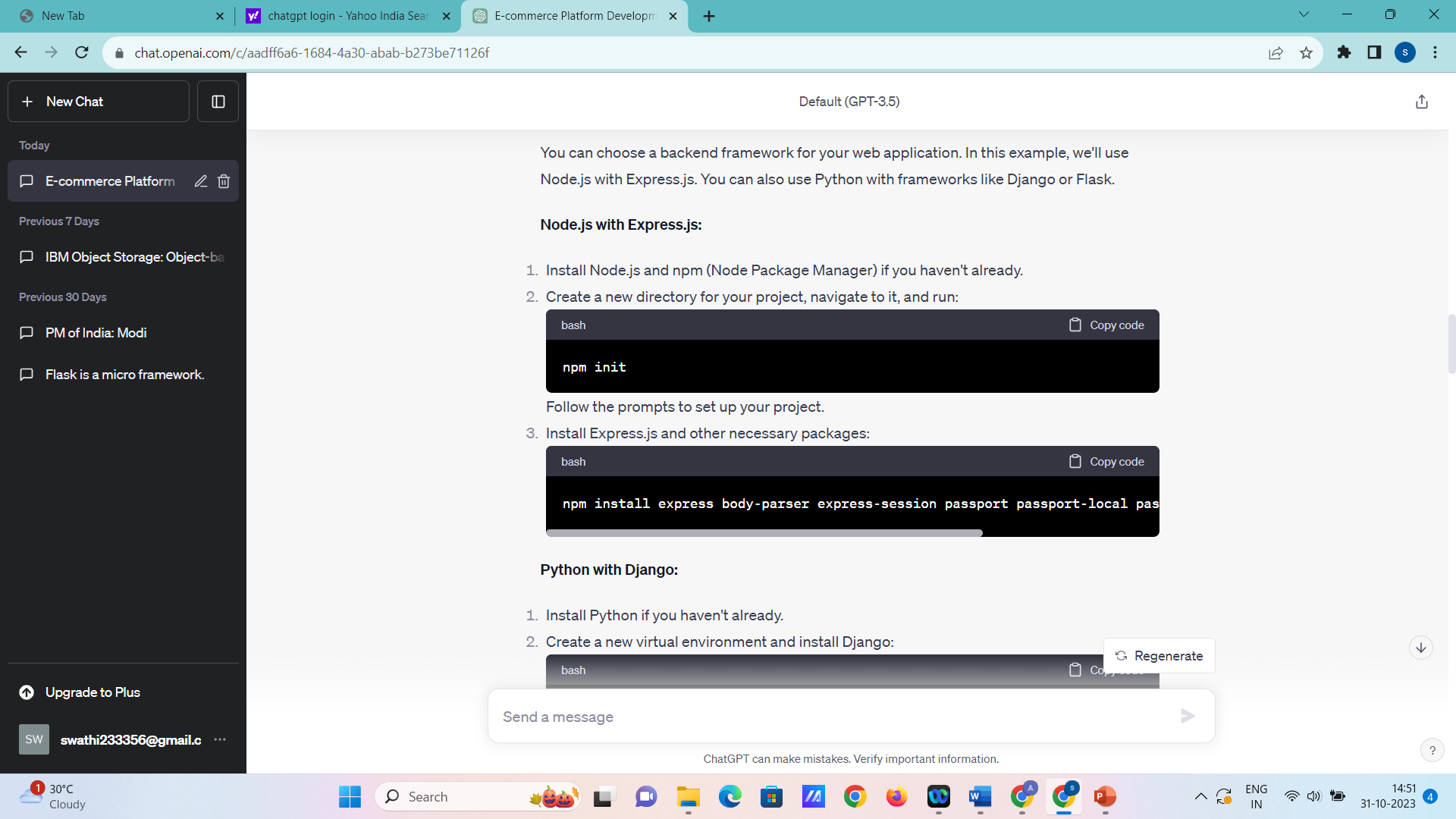
**Node.js with Express.js:**

1. Install Node.js and npm (Node Package Manager) if you haven't already.
2. Create a new directory for your project, navigate to it, and run:

Follow the prompts to set up your project.

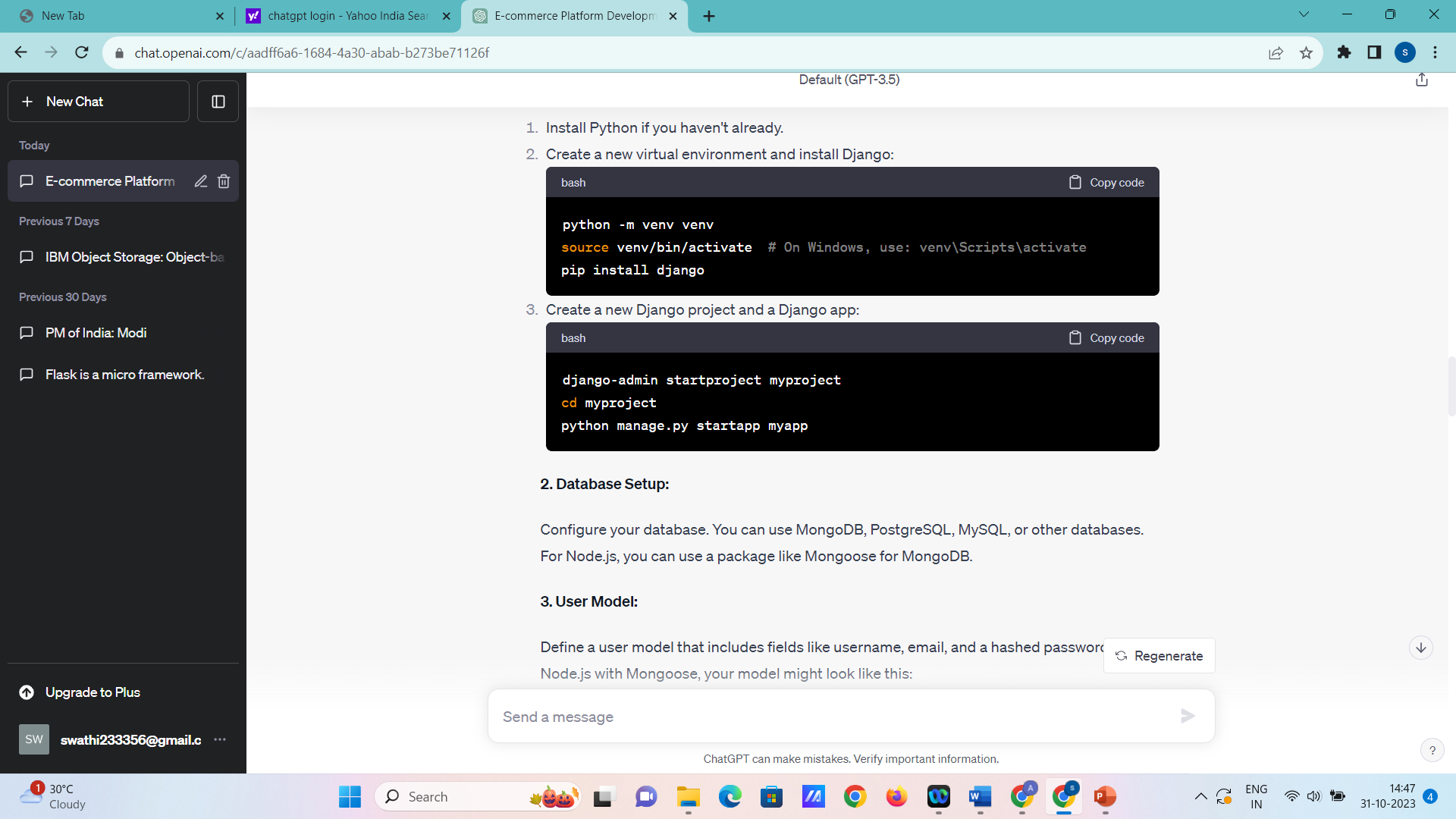


1. Install Express.js and other necessary packages:

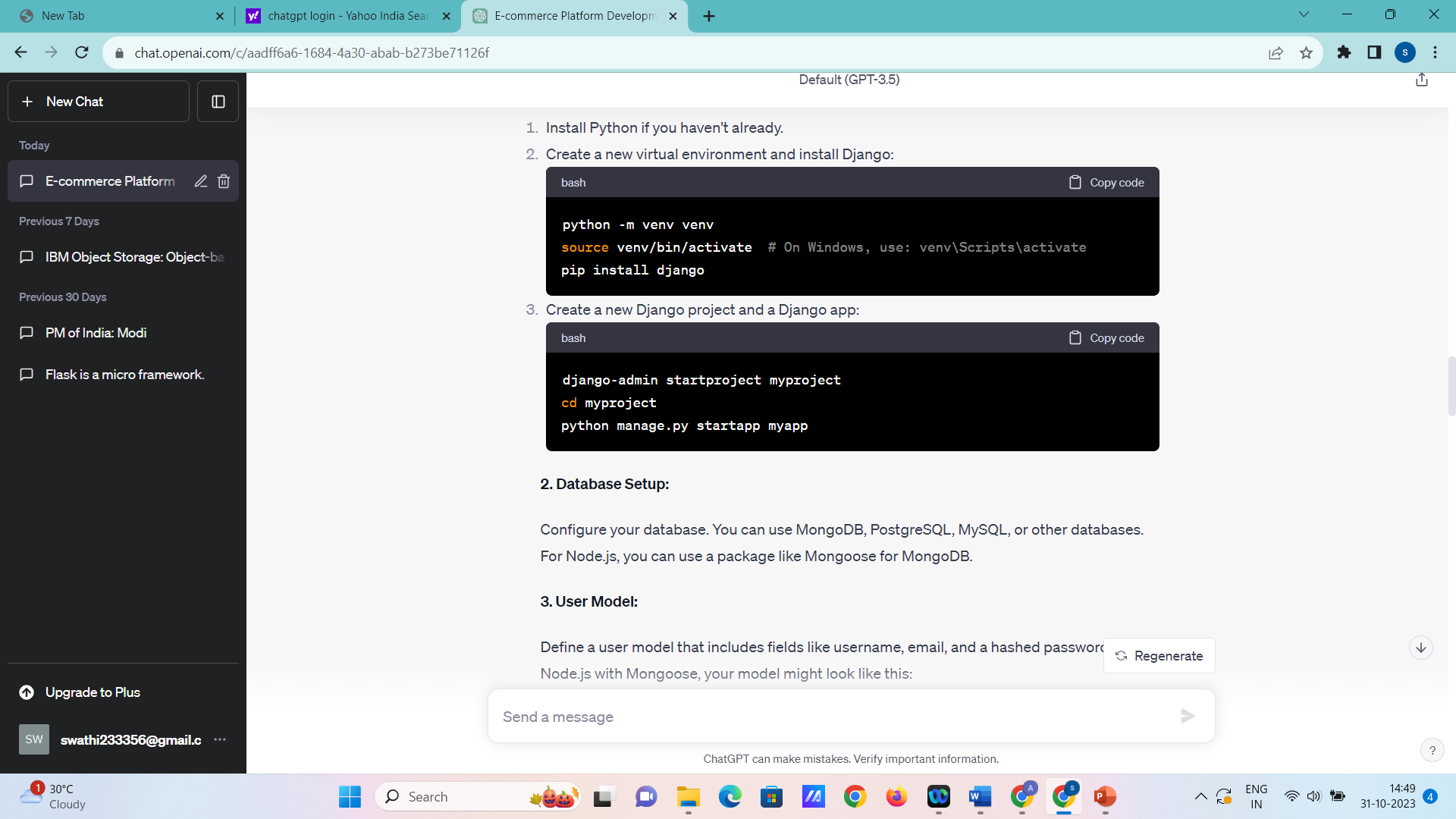
mongoose

**Python with Django:**

1. Install Python if you haven't already.
2. Create a new virtual environment and install Django:



1. Create a new Django project and a Django app:



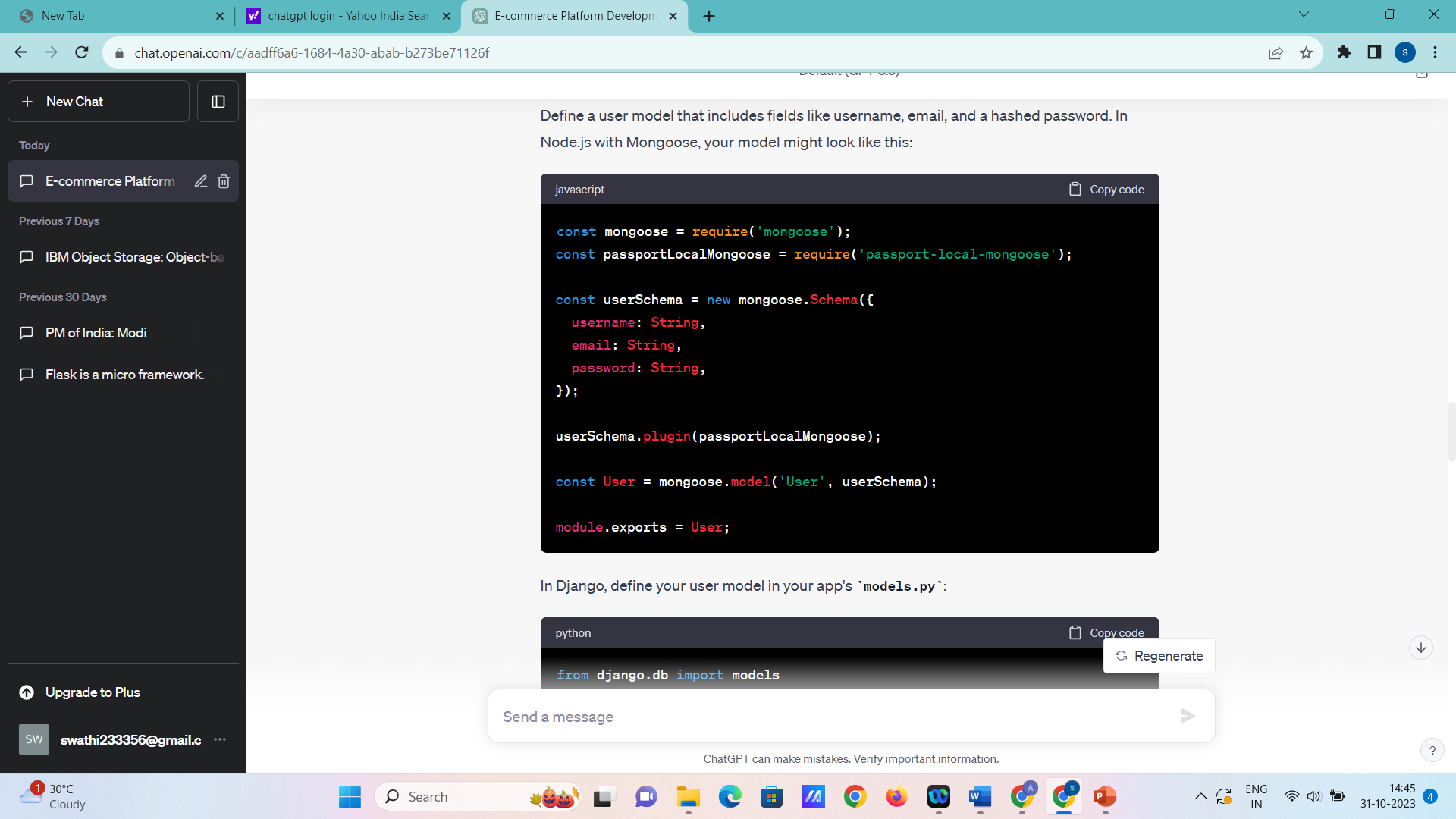
pp

**2. Database Setup:**

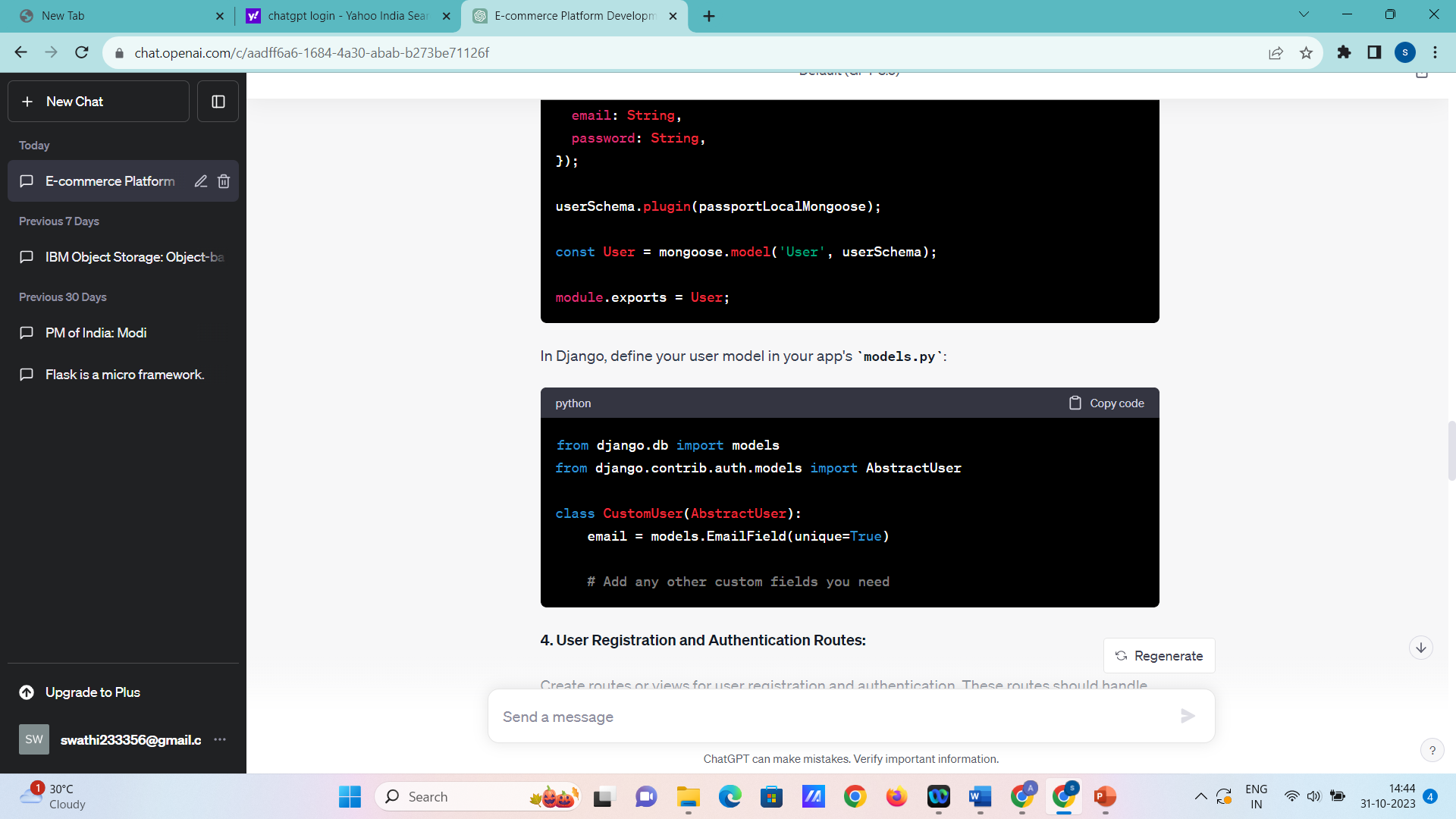
Configure your database. You can use MongoDB, PostgreSQL, MySQL, or other databases. For Node.js, you can use a package like Mongoose for MongoDB.

**3. User Model:**

Define a user model that includes fields like username, email, and a hashed password. In Node.js with Mongoose, your model might look like this:



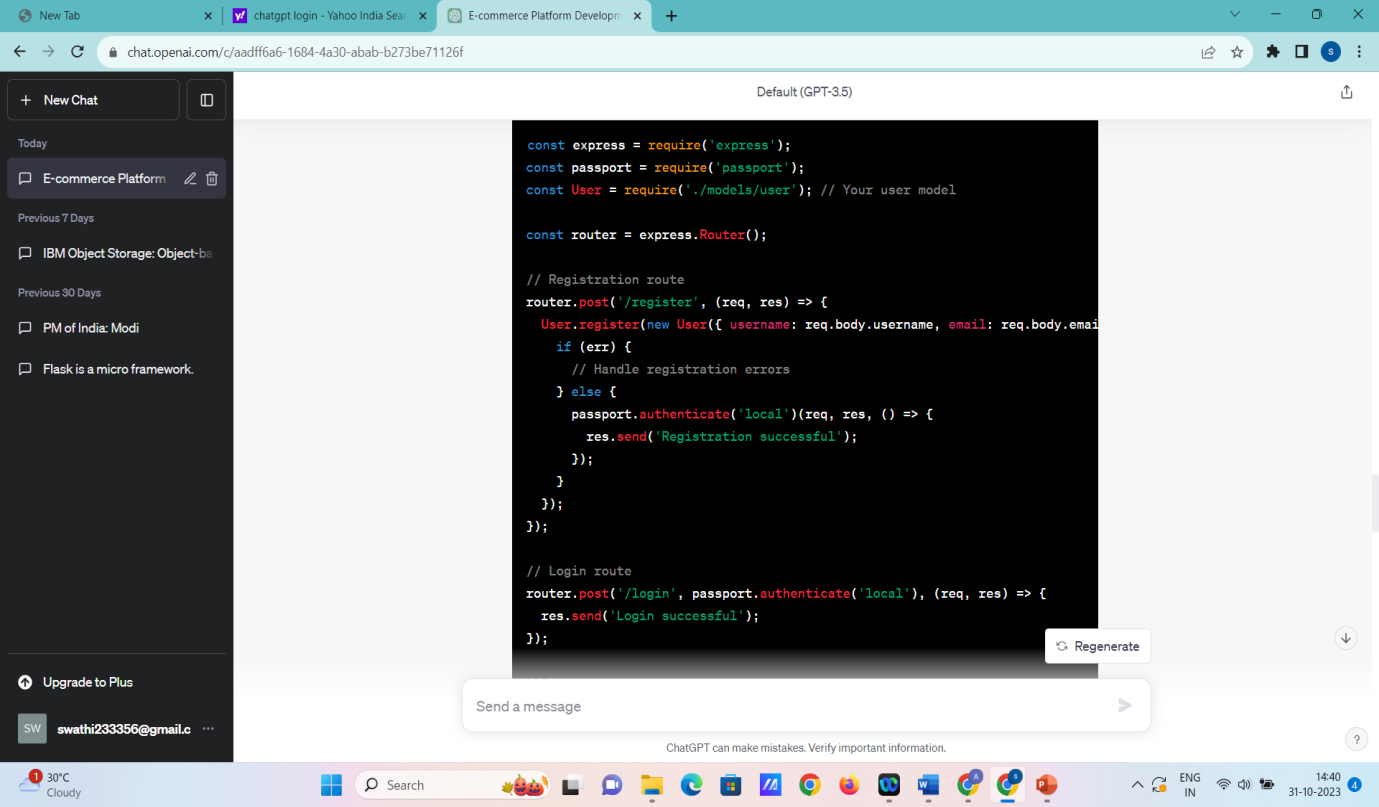
In Django, define your user model in your app's **models.py**:



**4. User Registration and Authentication Routes:**

Create routes or views for user registration and authentication. These routes should handle user registration, login, and logout.

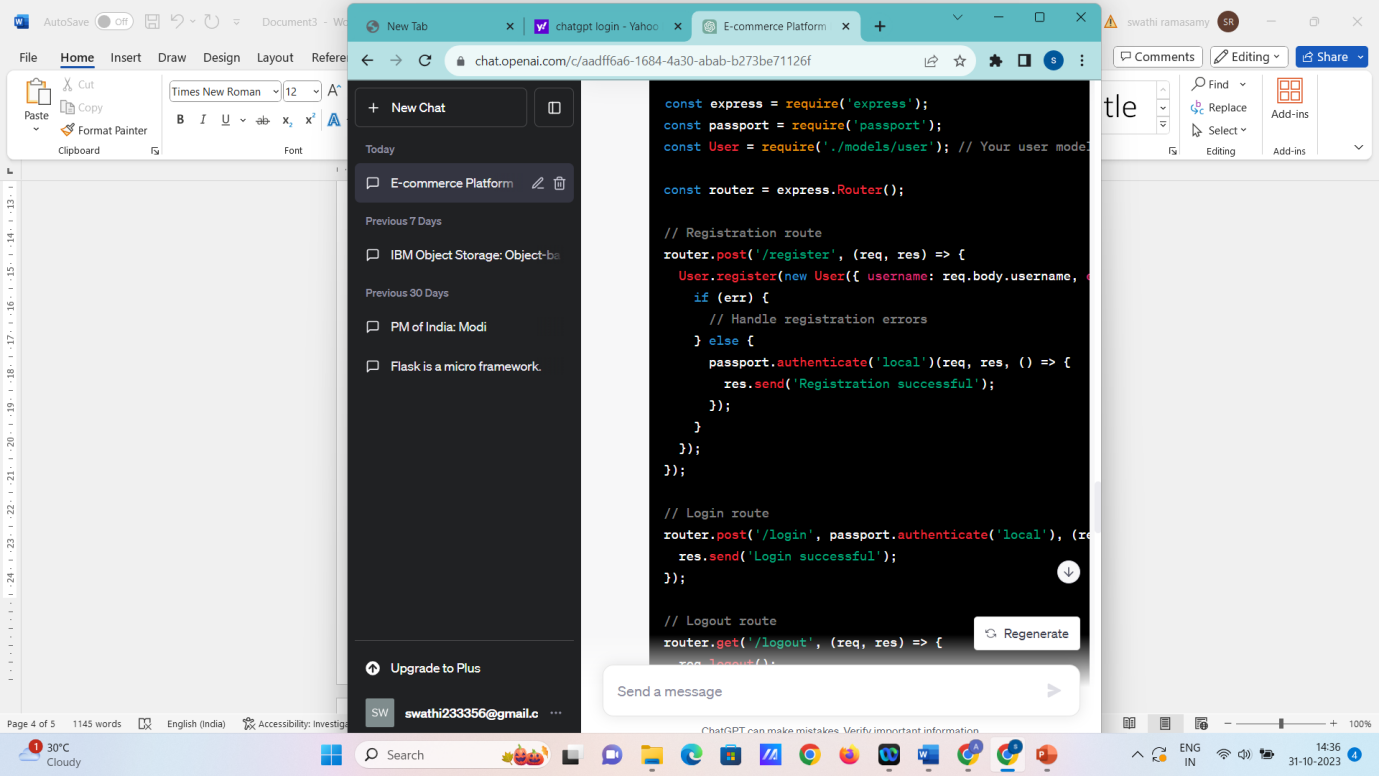
**Node.js with Express.js**



**Python with Django:**

In Django, you can use Django's built-in authentication views and forms for registration and login. You'll need to configure URLs and templates accordingly.

**5. Middleware and Passport (Node.js):**



**6. Frontend Integration:**

Create registration and login forms on your frontend and connect them to your backend routes. You'll typically use AJAX, Fetch, or a frontend library like React, Angular, or Vue.js for this.

**7. Test and Secure:**

Thoroughly test your registration and authentication features to ensure they work as expected. Implement security features like rate limiting, input validation, and password policies to enhance security.

This is a simplified outline of setting up user registration and authentication using Node.js with Express.js or Python with Django. Depending on your specific requirements, you may need to expand on these steps and implement additional features such as email verification, password reset, and user profile management.

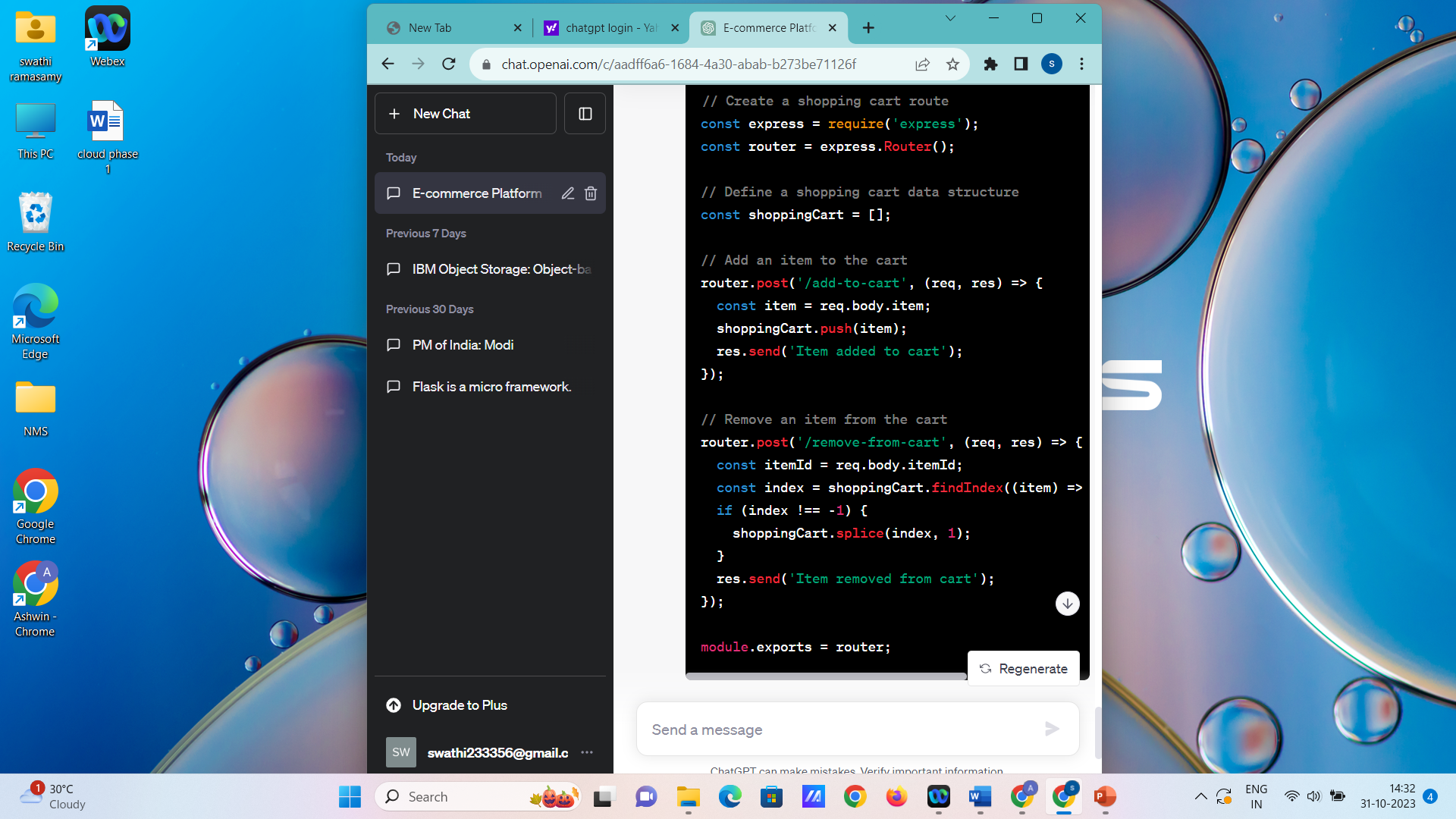
Top of Form

Implementing shopping cart functionality, calculating the total, and enabling a smooth checkout process is crucial for your e-commerce platform. Below are the steps to implement these features:

**1. Shopping Cart Functionality:**

Create the shopping cart functionality that allows users to add and manage items in their cart.

**Node.js with Express.js:**



In a Python Django application, you would follow a similar pattern, creating views and routes for adding and removing items

**Objective:**

The objective of the project is to develop and deploy a user-friendly E-commerce platform on IBM Cloud Foundry, enabling businesses to sell products online, manage inventory, and facilitate seamless transactions. The application aims to provide a secure, scalable, and responsive shopping experience for both desktop and mobile users.

**Design Thinking Process:**

1. **Empathize:** Understand the needs of potential users and businesses. Conduct market research and user surveys to identify pain points and requirements.
2. **Define:** Define clear goals based on user feedback. Create user personas, set feature priorities, and establish the project scope.
3. **Ideate:** Brainstorm and generate ideas for the application's layout, features, and user interactions. Prioritize innovative and user-friendly solutions.
4. **Prototype:** Develop wireframes and interactive prototypes to visualize the user interface and workflow. Gather feedback and iterate on the design based on user testing.
5. **Test:** Conduct usability testing to ensure the platform is intuitive and functional. Gather feedback and make necessary adjustments before development.

**Development Phases:**

1. **Backend Development:**
   * Set up a secure and scalable backend infrastructure using IBM Cloud Foundry services.
   * Implement server-side logic, including user authentication, product management, order processing, and payment gateway integration.
2. **Frontend Development:**
   * Design responsive user interfaces for both desktop and mobile platforms.
   * Implement interactive features, product catalog, search functionality, cart management, and user account pages.
3. **Integration and Testing:**
   * Integrate frontend and backend components to ensure seamless communication.
   * Conduct thorough testing, including unit tests, integration tests, and performance testing, to identify and resolve issues.
4. **Deployment and Monitoring:**
   * Deploy the application on IBM Cloud Foundry, ensuring high availability and scalability.
   * Set up monitoring tools to track application performance, user interactions, and error logs.

**Platform Layout, Features, and Technical Implementation:**

* **Platform Layout:**
  + Homepage displaying featured products, promotions, and categories.
  + Product catalog with detailed product pages, including images, descriptions, prices, and customer reviews.
  + User authentication and account management functionality.
  + Shopping cart and checkout process with multiple payment options.
  + Order history and tracking for users.
  + Admin dashboard for managing products, inventory, and orders.
* **Features:**
  + User-friendly interface with intuitive navigation.
  + Secure user authentication and data encryption.
  + Integration with payment gateways for secure transactions.
  + Real-time inventory management to prevent overselling.
  + Automated order confirmation emails and shipping notifications.
  + Responsive design for optimal viewing on various devices.
* **Technical Implementation:**
  + Backend: Node.js or Python with Express.js, interacting with databases (e.g., MongoDB or PostgreSQL).
  + Frontend: HTML, CSS, JavaScript (React.js or Angular) for responsive and interactive UI.
  + Database: Cloud-based database services such as IBM Db2 on Cloud or IBM Cloudant.
  + Security: SSL/TLS encryption, secure API endpoints, and input validation to prevent security vulnerabilities.
  + Deployment: IBM Cloud Foundry for hosting the application, with continuous integration and deployment pipelines.
  + Monitoring: IBM Cloud Monitoring tools for performance monitoring and error tracking.

