**ArtifyAI**

**Description:**

The objective behind the idea, is to produce an E-Commerce website where curated collection of AI- generated artwork is available to purchase.

**APPLICATION USAGE PROCESS:**

* Complete project is deployed into web servers. There is no need of running those manually. All services are completely up and running in cloud.
* As an end user, just i can browse with Frontend Application URL which is mentioned below:

<https://ecommerce-frontend-ui.azurewebsites.net/>

**Application URL: (Brower)**

<https://ecommerce-frontend-ui.azurewebsites.net/>

GitHub URLS: (Source Code)

Frontend Project:

<https://github.com/PraveenSuggula/EcommerceUI>

Backend Project:

<https://github.com/PraveenSuggula/EcommerceAPI>

Authentication Project:

<https://github.com/PraveenSuggula/EcommerceAuth>

**Technologies Used:**

Frontend: React, JavaScript, Node, HTML, CSS, Bootstrap, Axios, Jest

Backend: Dotnet Core 6.0, C#, Web API Swagger and Stripe for Payment Gateway

Authentication: Express.js, Node js.

**Tools and IDEs Used:**

Visual Studio Code (VS Code)

Visual Studio (Optional)

Postman

Microsoft Azure Cloud

**To Understand Development and Deployment Process:**

- We have Total 3 Projects,

1. Frontend (Client or UI)

2. Backend (Server)

3. Multi factor Authentication

**Note:** We have separated Authentication from backend server because of improving security to User Logins and Validation process by separately deploying it. Also, we can improve coding standards and reusability principles by using Service-Oriented Architecture (SOA). Which means, Separate project for every unique service that we needed.

**APPLICATION TESTING PROCESS:**

**1. Frontend URL:**

* Browse below URL to access deployment version of complete working application. <https://ecommerce-frontend-ui.azurewebsites.net/>

**2. Backend URL:**

* Backend will not be browsable form UI. Because there are no get API. If we want to check from browser, we can use weatherforcast just for depolyment check helps like ping.

<https://ecommerce-backend-api.azurewebsites.net/WeatherForecast>

* To Access Backend, Use Postman curl URL (Download and install postman and click on import and paste below URL)

*curl --location 'https://ecommerce-backend-api.azurewebsites.net/checkout' \*

*--header 'accept: \*/\*' \*

*--header 'Content-Type: application/json' \*

*--data '[*

*{*

*"Id": "price\_1Nw9YoEQLsa2XJpCPrsjuhLp",*

*"Quantity": 2*

*},*

*{*

*"Id": "price\_1Nw9afEQLsa2XJpCiLFFm9M3",*

*"Quantity": 1*

*'*

*}*

*]*

Modify data in above URL to test our API

**3. Multi factor Authentication (MFA):**

For test purpose, we can browse all users enrolled in our application with below URL

<https://ecommerce-registeration-service.azurewebsites.net/getUsersData>

* To Login from Postman (without using UI), use below curl

*curl --location 'https://ecommerce-registeration-service.azurewebsites.net/login' \*

*--header 'Content-Type: application/json' \*

*--data-raw '{*

*"email":"pragnesh@mail.com",*

*"password": "password"*

*}'*

* To Register from Postman (without using UI), use below curl

*curl --location 'https://ecommerce-registeration-service.azurewebsites.net/register' \*

*--header 'Content-Type: application/json' \*

*--data-raw '{*

*"email":"pragmesh02@gmail.com",*

*"password": "123"*

*}'*

**TO RUN AND DEBUG APPLICATION: (Developer mode)**

**Pre-Requests:**

- Download and install Visual Studio Code (VS Code)

- Install "C#" extension for language support

- Install ".NET Install Tool" for run backend dotnet commands.

**Project Setup:**

1. Authentication Project:

- Clone and Open EcommerceAuth and navigate to ‘EcommerceAuth/registration-server’ path in cli.

- Install dependencies with “npm install”.

- To Run authentication project, use "npm start".

2. Frontend Project:

- Clone and Open EcommerceUI and navigate to store path in cli.

- Install dependencies with “npm install”.

- To Run Frontend UI, use "npm start".

3. Backend Project:

- Clone and Open EcommerceAPI and navigate to EcommerceAPI/EcommerceAPI path in VS Code Terminal.

- To Run Backend, use "dotnet run".

- To check APIs, use <https://localhost:7255/swagger/index.html>

**Working with application:**

- For UI, Browe with "http://localhost:3000"

- Register with email id and password

- Login with same email id and password

- Use '0000' for OTP

- Add items to cart and checkout

- Click Purchase items

- Redirects to payment gateway.

- Enter card details. For random card usage use stripe test cards <https://stripe.com/docs/testing#cards>

- If details are validated successfully, then its auto redirects to <http://localhost:3000/success>

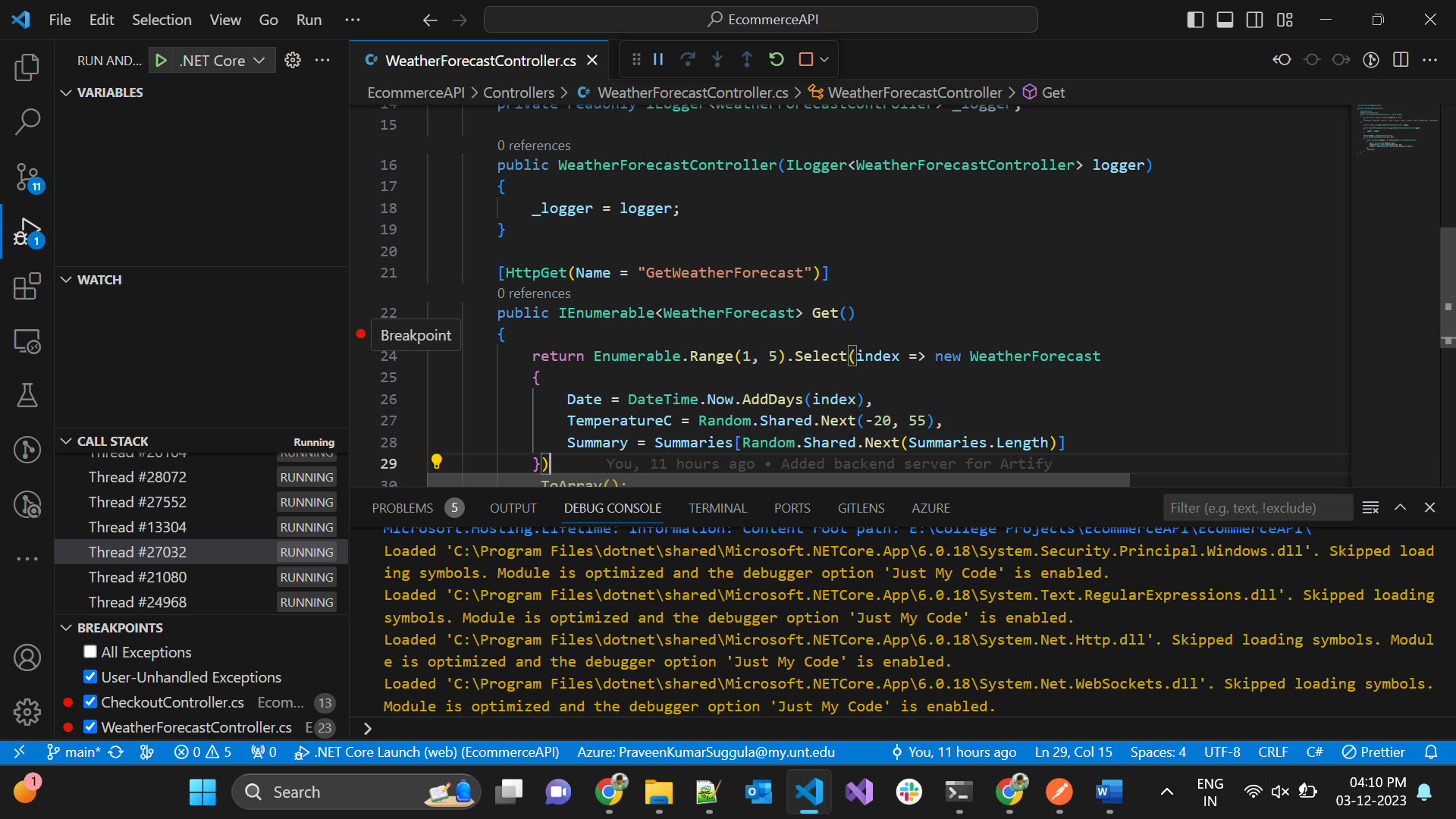
- Purchase Completed.

**API Debugging Process:**

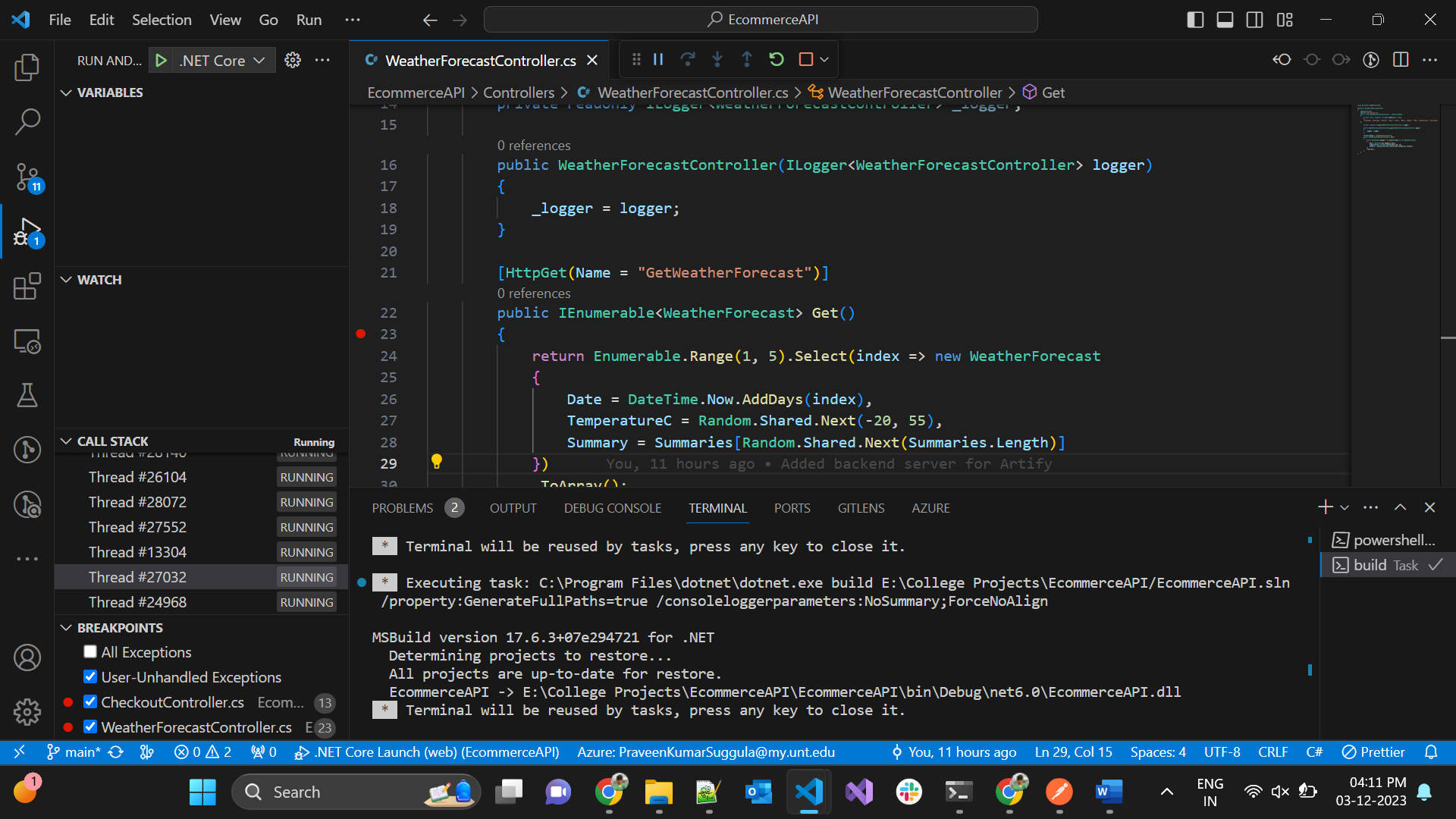
**-** Open Backend project in VS Code.

- Open "Run and Debug" in sidebar and click on "Run and Debug" to configure debugging process.

- Set break point in code that you want to inspect.

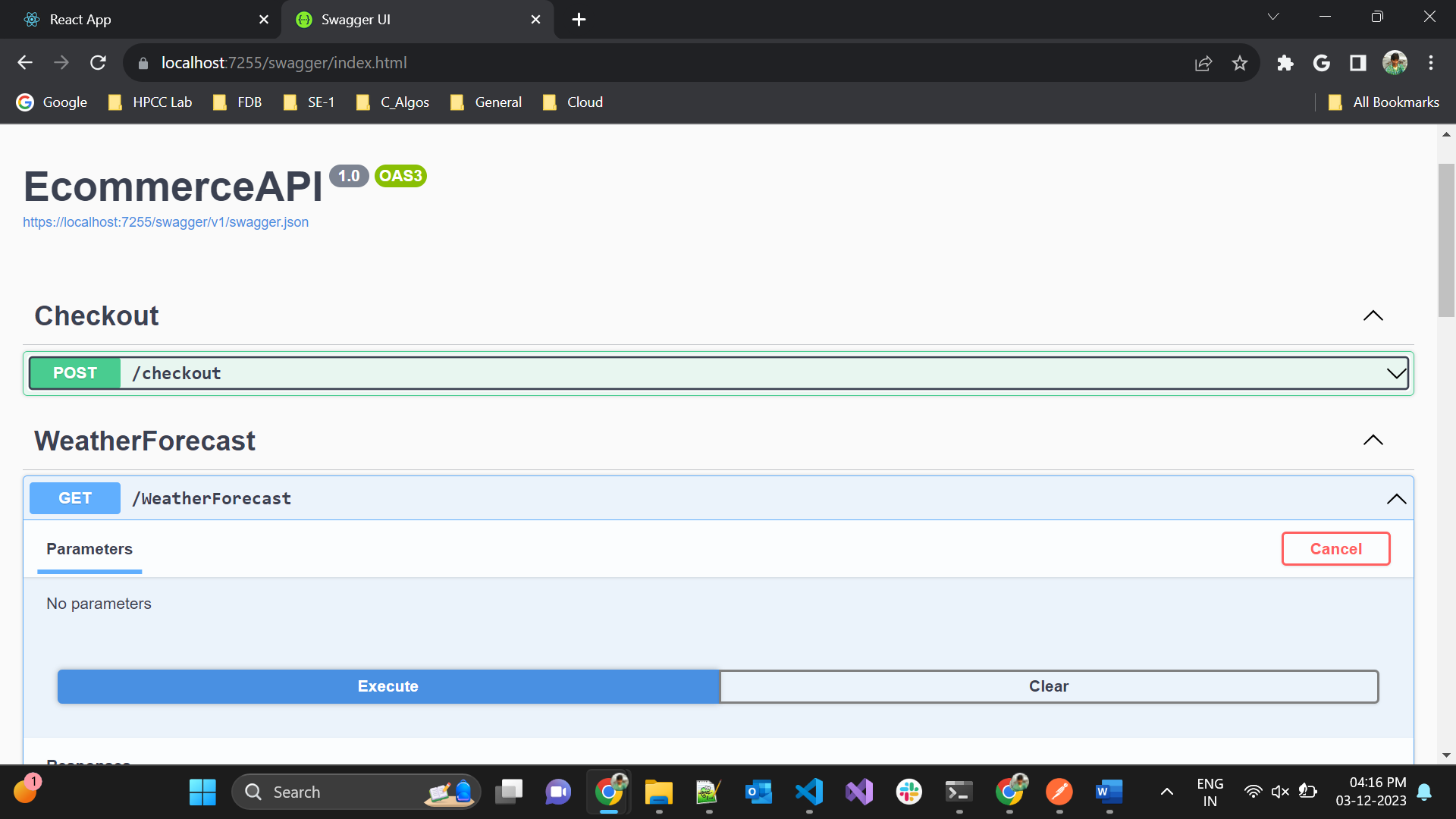


- Run the Project and browser endpoint

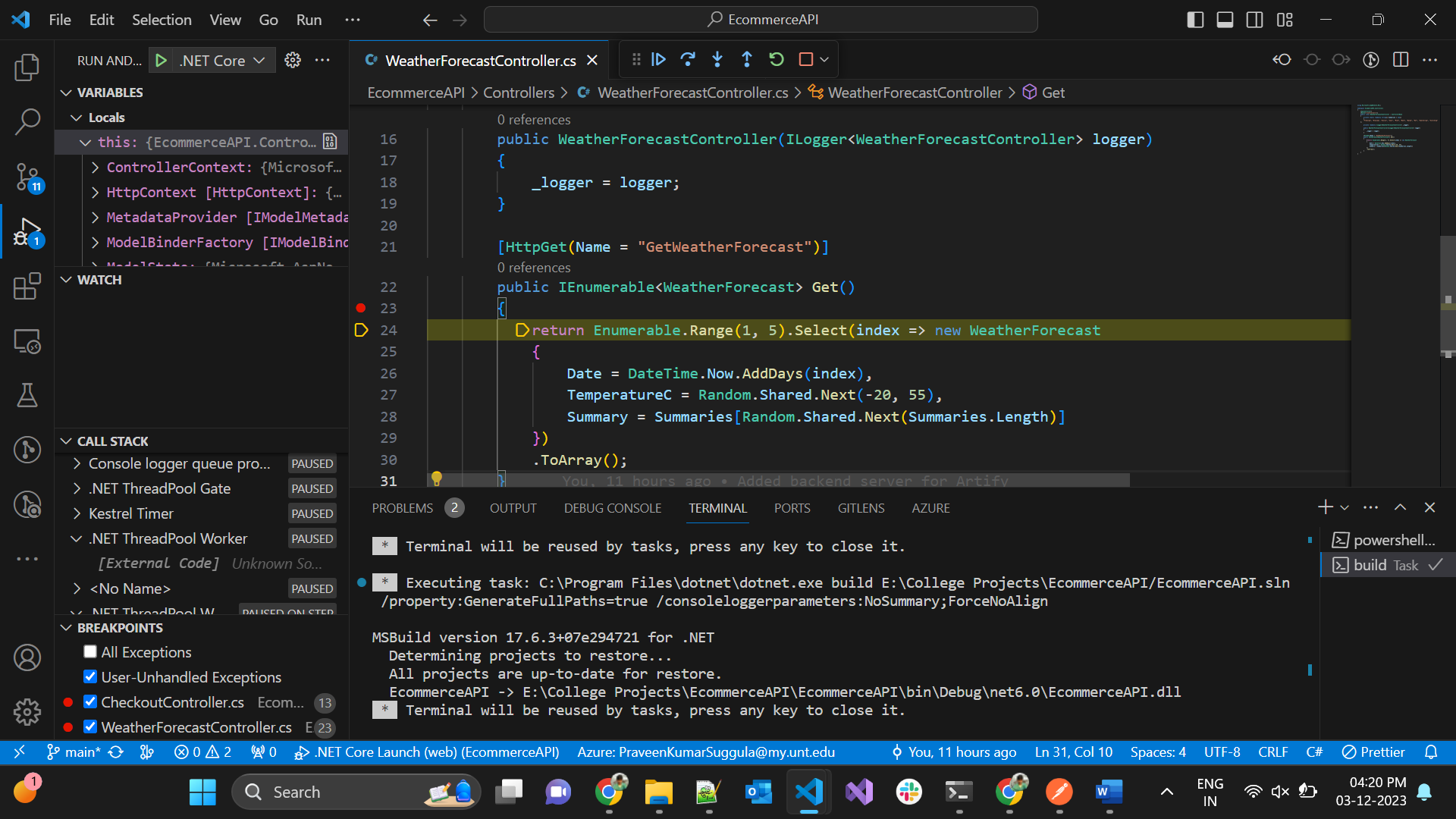


-When running dotnet core 6 project, Swagger is automatically configured. Browser that endpoint or running project. <https://localhost:7255/swagger/index.html>

-Hit weather forecast in this case.



-Press F10 or Next to debug API code.



**Project Demo:**

