Q1

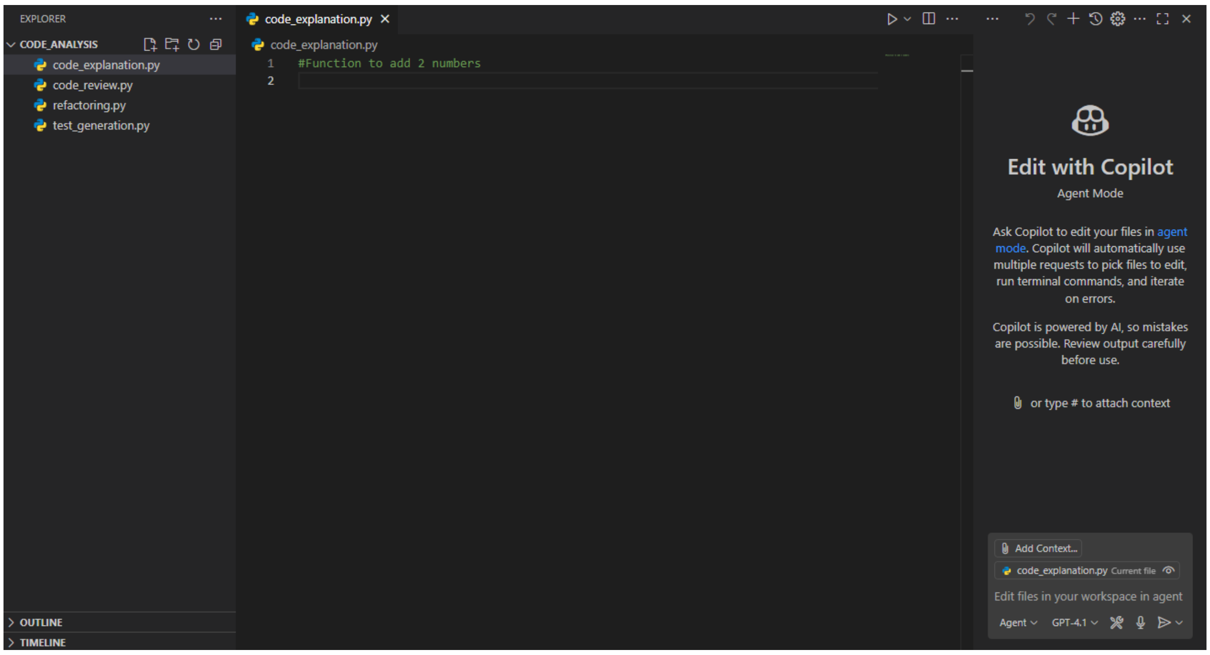
I created a folder named CODE_ANALYSIS to organize four use cases demonstrating how GitHub Copilot assisted in writing, understanding, refactoring, and testing code.

1. code_explanation.py – Code Understanding with Copilot

Objective:

Use GitHub Copilot to explain an existing piece of code.

I wrote a comment like: #Function to add 2 numbers

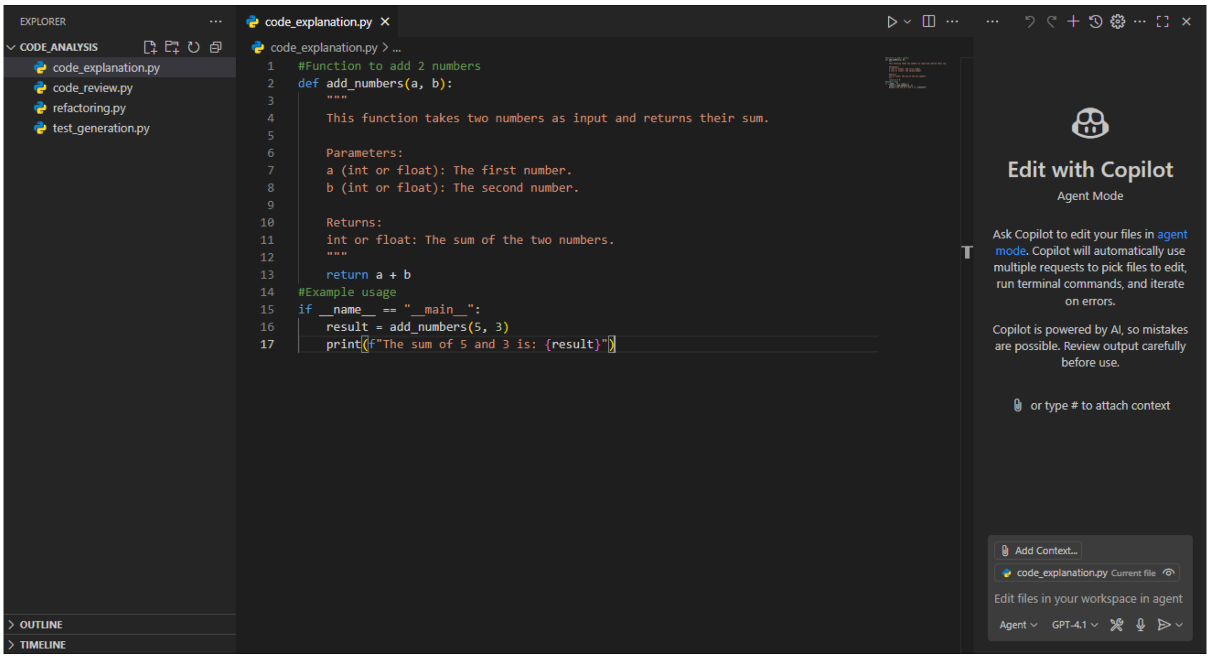


```
#Function to add 2 numbers
```

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

Copilot generated a function along with natural language explanation for each step in the function.



```python
#Function to add 2 numbers
def add_numbers(a, b):
    """
    This function takes two numbers as input and returns their sum.

    Parameters:
    a (int or float): The first number.
    b (int or float): The second number.

    Returns:
    int or float: The sum of the two numbers.
    """
    return a + b
#Example usage
if __name__ == "__main__":
    result = add_numbers(5, 3)
    print(f"The sum of 5 and 3 is: {result}")
```

Edit with Copilot

Agent Mode

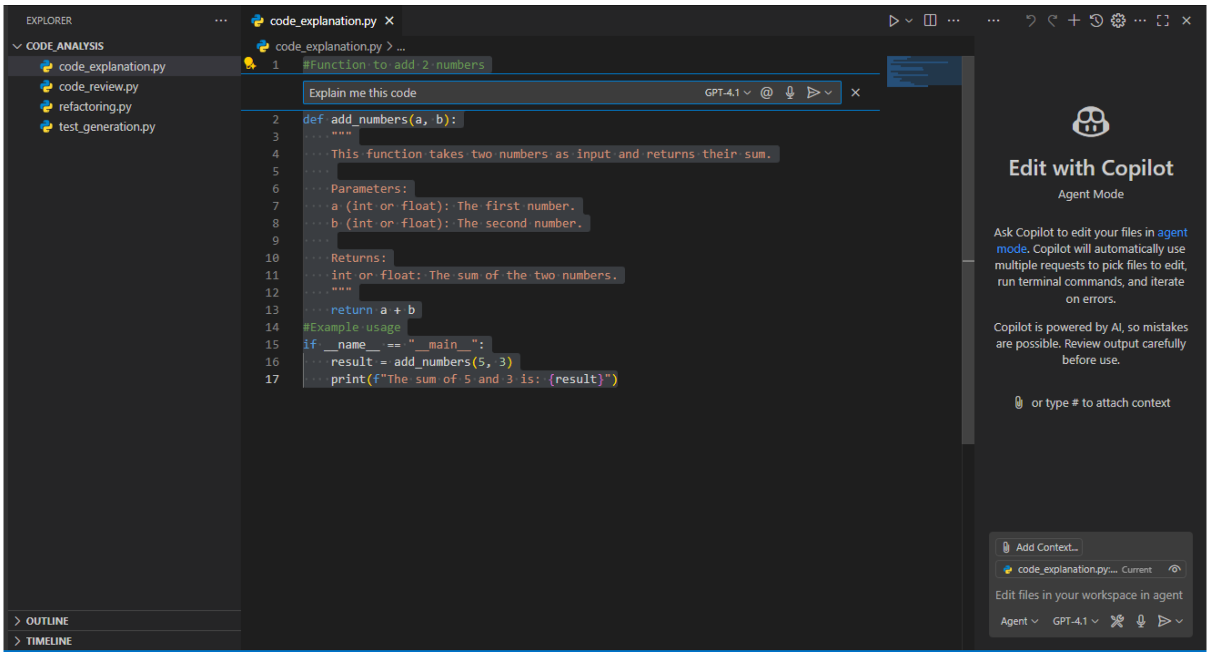
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...
code_explanation.py Current file
Edit files in your workspace in agent
Agent
GPT-4.1

- Pressed `Ctrl + A` to select the whole code.
- Pressed `Ctrl + I` (short cut to open copilot chat).
- Asked **"Explain me this code".**



```python
#Function to add 2 numbers
def add_numbers(a, b):
    """
    This function takes two numbers as input and returns their sum.

    Parameters:
    a (int or float): The first number.
    b (int or float): The second number.

    Returns:
    int or float: The sum of the two numbers.
    """
    return a + b
#Example usage
if __name__ == "__main__":
    result = add_numbers(5, 3)
    print(f"The sum of 5 and 3 is: {result}")
```

Edit with Copilot

Agent Mode

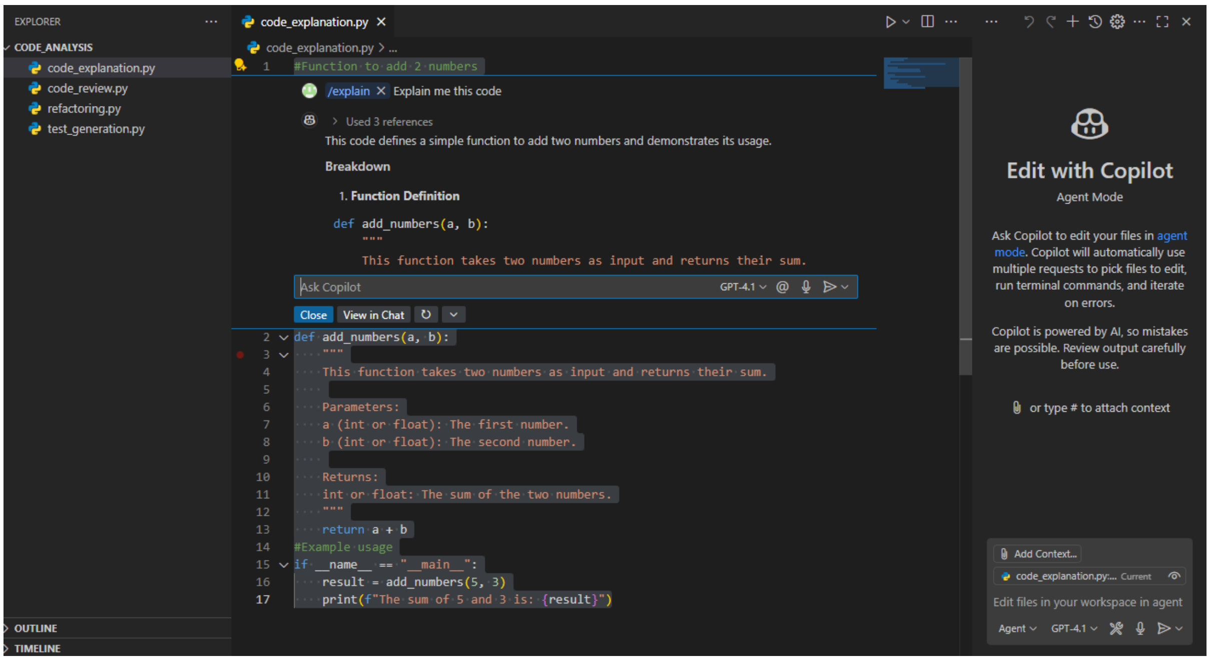
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...
code_explanation.py... Current
Edit files in your workspace in agent
Agent
GPT-4.1

Copilot explained me the whole code.



EXPLORER

CODE_ANALYSIS
- code_explanation.py
- code_review.py
- refactoring.py
- test_generation.py

code_explanation.py > ...

```python
#Function to add 2 numbers
```

/explain Explain me this code

Used 3 references

This code defines a simple function to add two numbers and demonstrates its usage.

Breakdown

1. Function Definition

```python
def add_numbers(a, b):
    """
    This function takes two numbers as input and returns their sum.
```

Ask Copilot

Close | View in Chat

```python
def add_numbers(a, b):
    """
    This function takes two numbers as input and returns their sum.

    Parameters:
    a (int or float): The first number.
    b (int or float): The second number.

    Returns:
    int or float: The sum of the two numbers.
    """
    return a + b
#Example usage
if __name__ == "__main__":
    result = add_numbers(5, 3)
    print(f"The sum of 5 and 3 is: {result}")
```

Edit with Copilot

Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...

code_explanation.py:... Current

Edit files in your workspace in agent

Agent GPT-4.1

OUTLINE

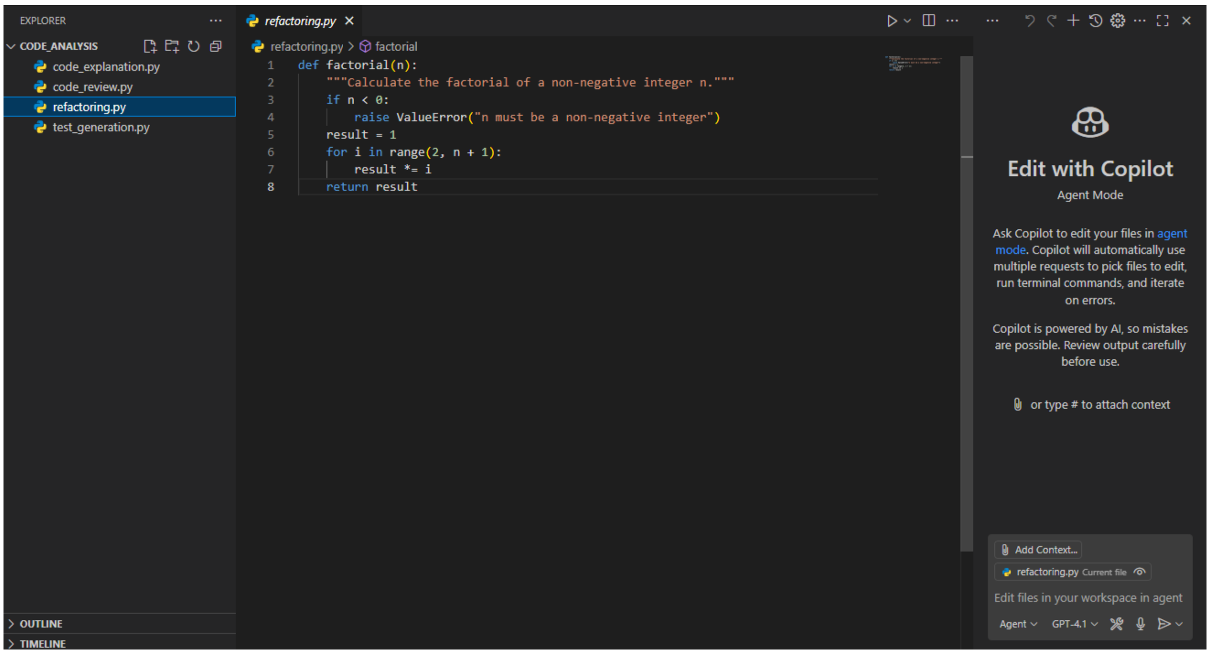
TIMELINE

2. refactoring.py – Code Refactoring

Objective:

Refactor inefficient or verbose code to make it cleaner and more efficient.

I used a raw function of finding factorial of a number



EXPLORER

CODE_ANALYSIS
- code_explanation.py
- code_review.py
- refactoring.py
- test_generation.py

refactoring.py > factorial

```python
def factorial(n):
    """Calculate the factorial of a non-negative integer n."""
    if n < 0:
        raise ValueError("n must be a non-negative integer")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

Edit with Copilot

Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...

refactoring.py Current file

Edit files in your workspace in agent

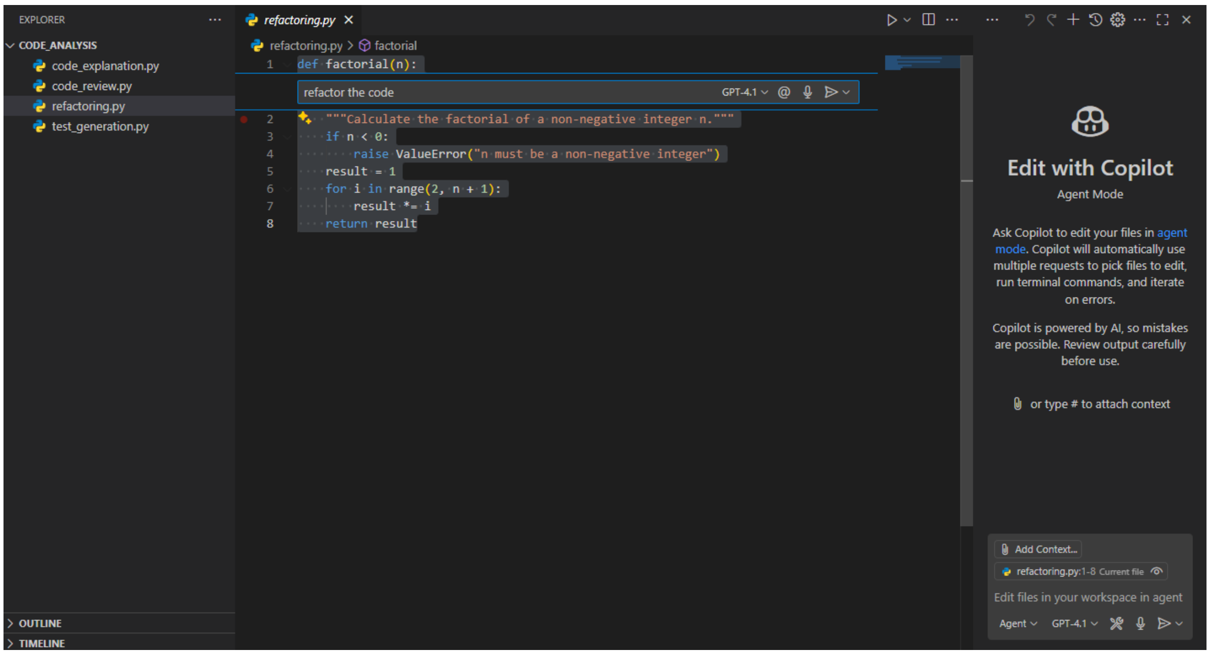
Agent GPT-4.1

OUTLINE

TIMELINE

- Pressed `Ctrl + A` to select the whole code.
- Pressed `Ctrl + I` (short cut to open copilot chat).

- Asked **"Refactor the code".**



EXPLORER
CODE_ANALYSIS
code_explanation.py
code_review.py
refactoring.py
test_generation.py

refactoring.py > factorial

refactor the code

```python
def factorial(n):
    """Calculate the factorial of a non-negative integer n."""
    if n < 0:
        raise ValueError("n must be a non-negative integer")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

Edit with Copilot
Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

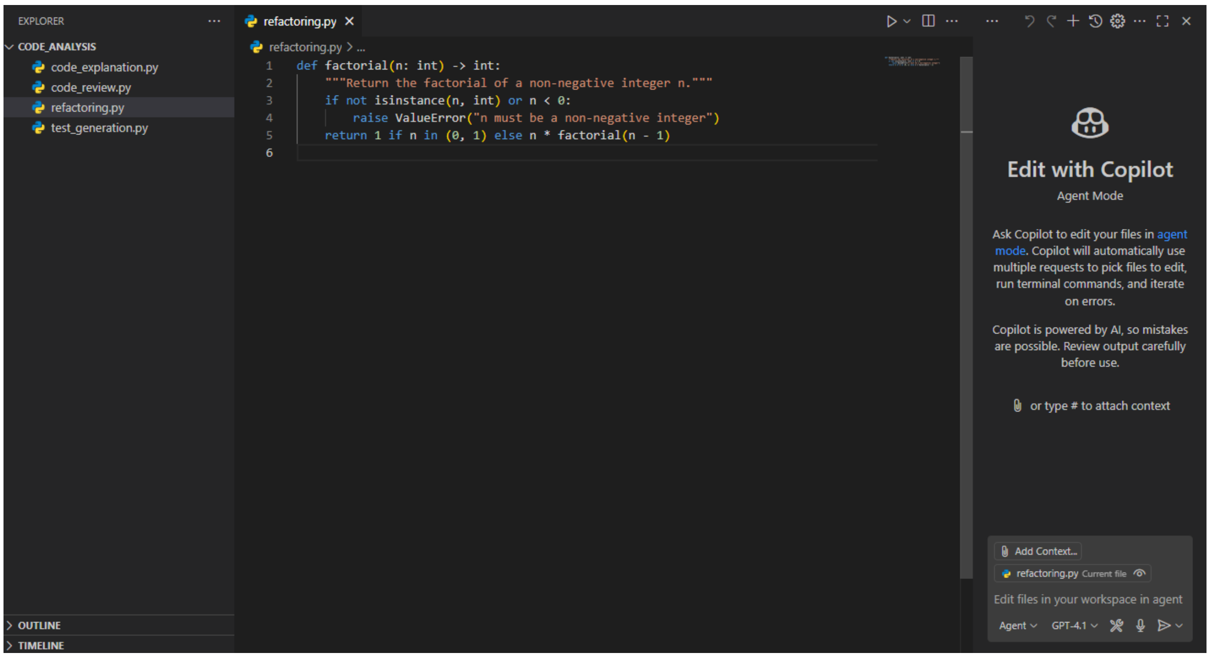
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...
refactoring.py:1-8 Current file
Edit files in your workspace in agent
Agent GPT-4.1

Copilot game me more efficient and optimised code than the present one.

I clicked accept to get the efficient code in the file.



EXPLORER
CODE_ANALYSIS
code_explanation.py
code_review.py
refactoring.py
test_generation.py

refactoring.py > ...

```python
def factorial(n: int) -> int:
    """Return the factorial of a non-negative integer n."""
    if not isinstance(n, int) or n < 0:
        raise ValueError("n must be a non-negative integer")
    return 1 if n in (0, 1) else n * factorial(n - 1)
```

Edit with Copilot
Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

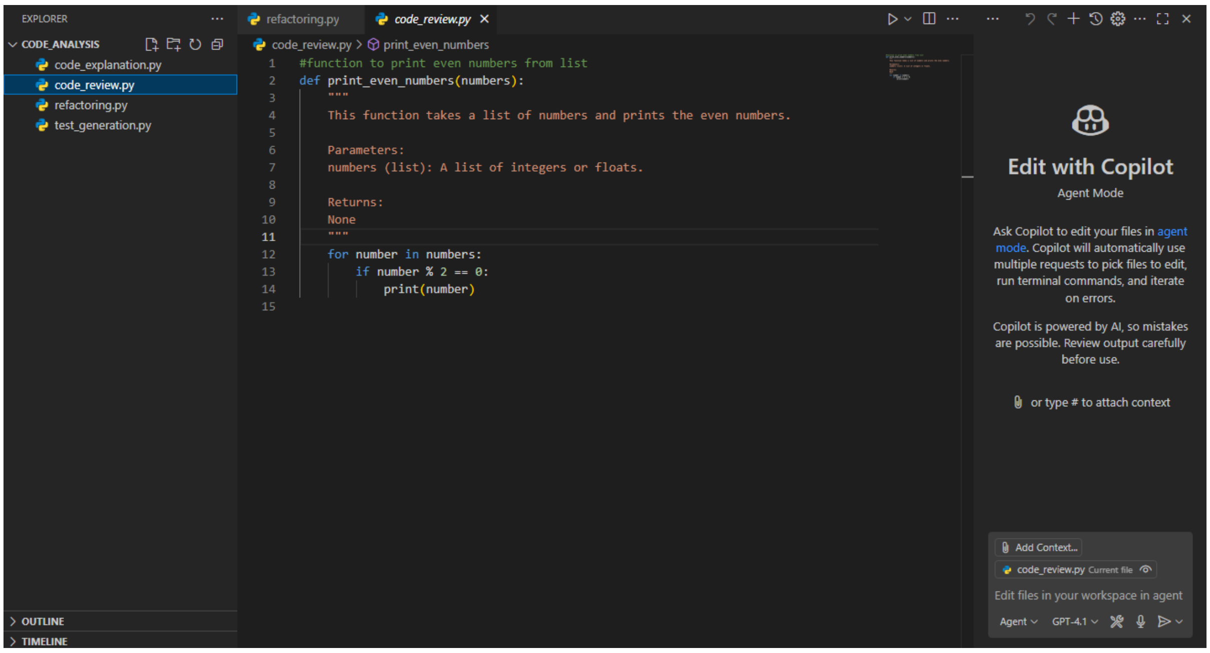
Add Context...
refactoring.py Current file
Edit files in your workspace in agent
Agent GPT-4.1

3. code_review.py – Code Review & Suggestions

Objective:

Simulate a mini code review session where Copilot provides improvement suggestions.

I generated a function to print even functions from the list.



```python
#function to print even numbers from list
def print_even_numbers(numbers):
    """
    This function takes a list of numbers and prints the even numbers.

    Parameters:
    numbers (list): A list of integers or floats.

    Returns:
    None
    """
    for number in numbers:
        if number % 2 == 0:
            print(number)
```

Edit with Copilot

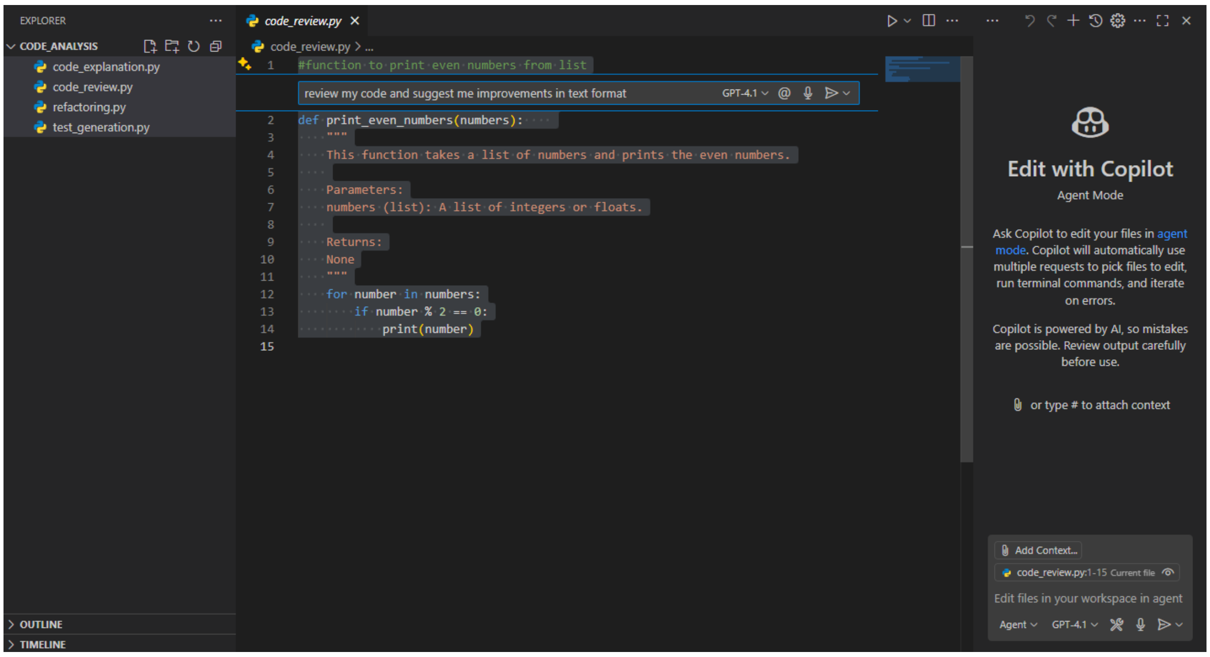
Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

- Pressed `Ctrl + A` to select the whole code.
- Pressed `Ctrl + I` (short cut to open copilot chat).
- Asked **"review my code and suggest me improvements in text format".**



```
#function to print even numbers from list
```

```
review my code and suggest me improvements in text format
```

```
def print_even_numbers(numbers):
    """
    This function takes a list of numbers and prints the even numbers.

    Parameters:
    numbers (list): A list of integers or floats.

    Returns:
    None
    """
    for number in numbers:
        if number % 2 == 0:
            print(number)
```

Edit with Copilot

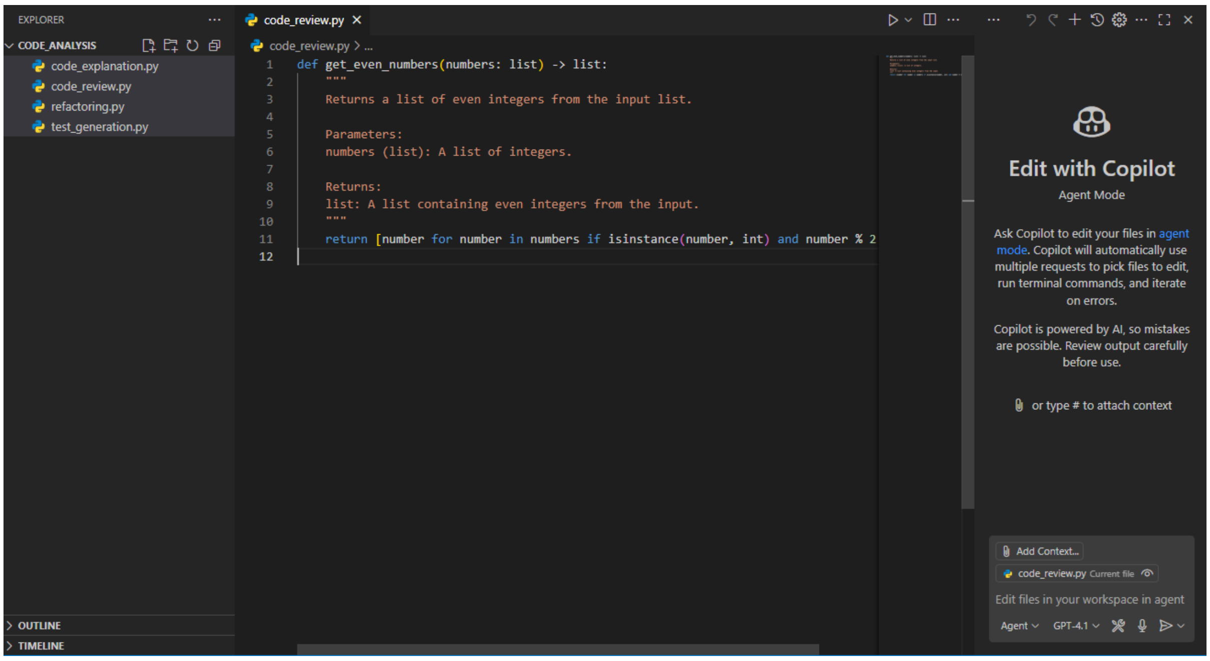
Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

I accepted the improvements suggested by copilot and applied to my code.



```
def get_even_numbers(numbers: list) -> list:
    """
    Returns a list of even integers from the input list.

    Parameters:
    numbers (list): A list of integers.

    Returns:
    list: A list containing even integers from the input.
    """
    return [number for number in numbers if isinstance(number, int) and number % 2
```

Edit with Copilot

Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

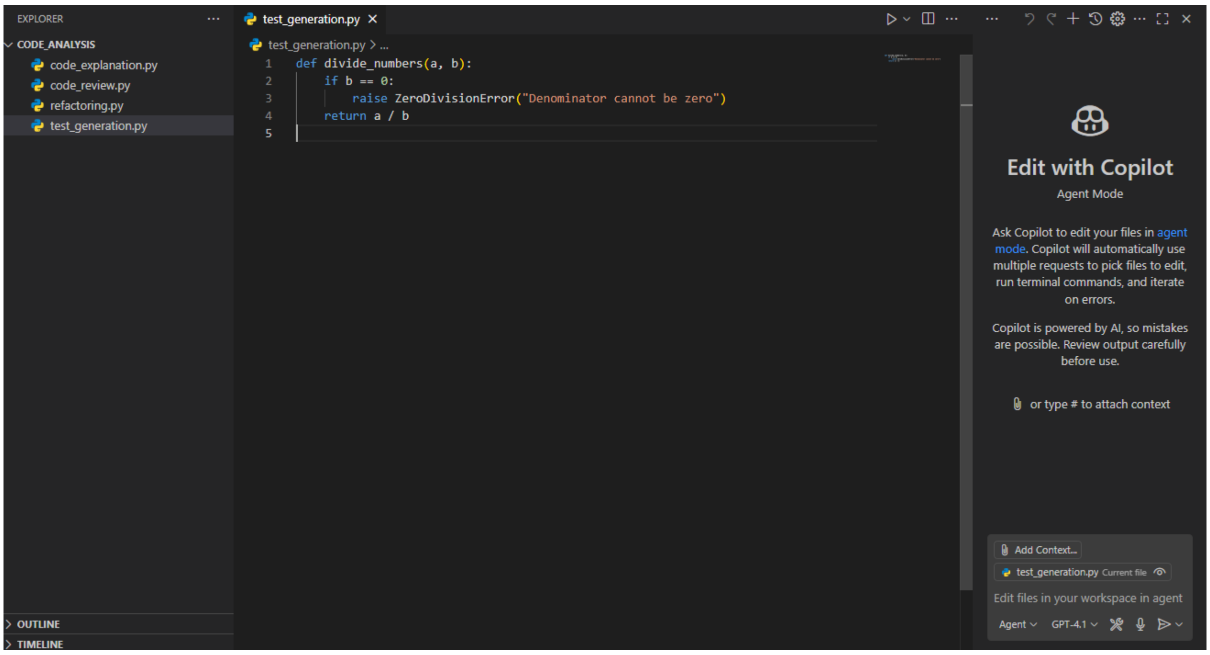
or type # to attach context

4. test_generation.py – Auto-Generating Test Cases

Objective:

Use Copilot to generate basic unit tests for a function.

I wrote a function divide_numbers to divide two numbers.

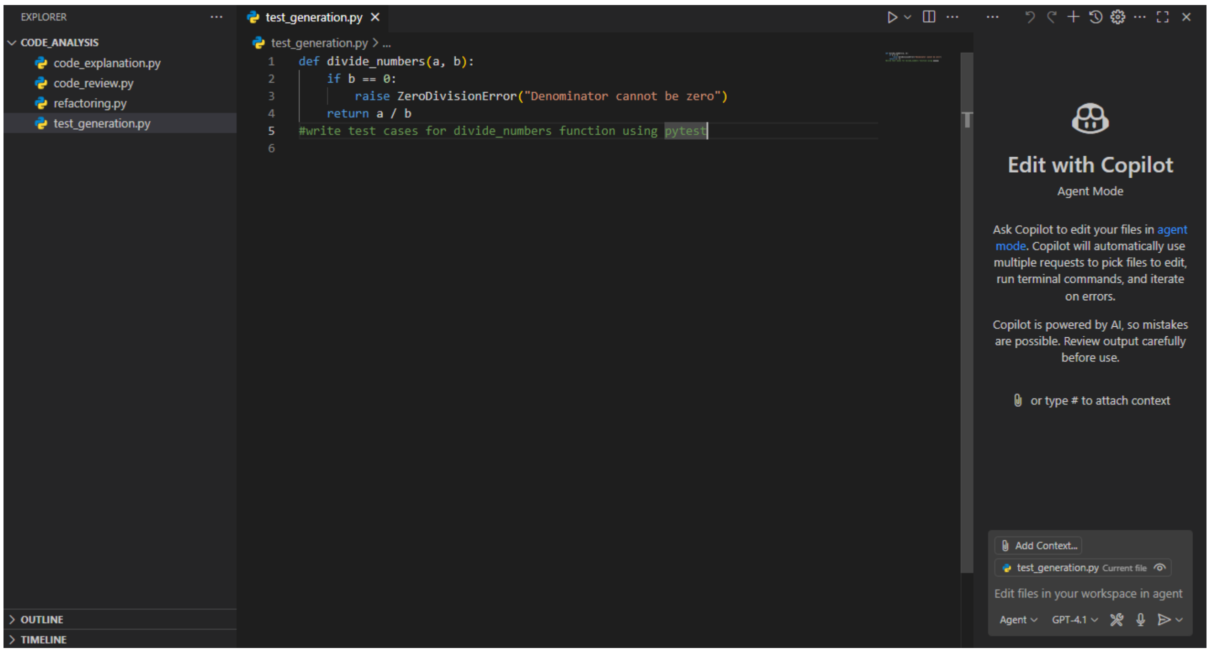


```python
def divide_numbers(a, b):
    if b == 0:
        raise ZeroDivisionError("Denominator cannot be zero")
    return a / b

```

EXPLORER
CODE_ANALYSIS
code_explanation.py
code_review.py
refactoring.py
test_generation.py
OUTLINE
TIMELINE
test_generation.py
Edit with Copilot
Agent Mode
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.
or type # to attach context
Add Context...
test_generation.py Current file
Edit files in your workspace in agent
Agent
GPT-4.1

I wrote a comment “#write test cases for divide_numbers function using pytest”

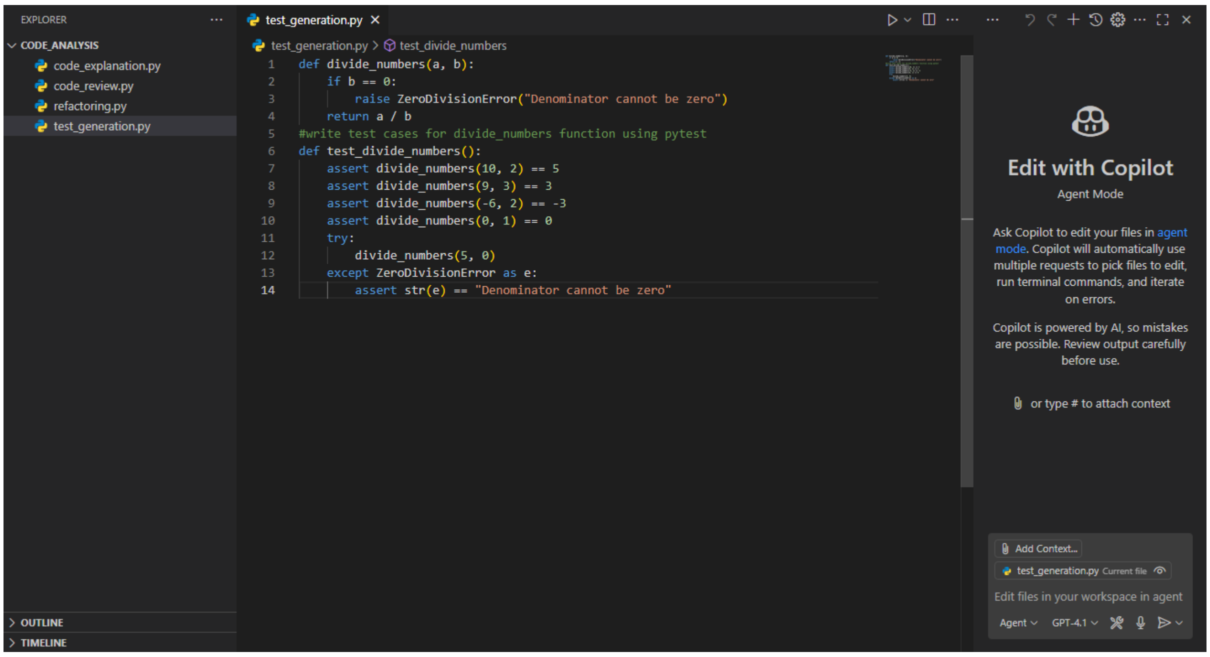


```python
def divide_numbers(a, b):
    if b == 0:
        raise ZeroDivisionError("Denominator cannot be zero")
    return a / b
#write test cases for divide_numbers function using pytest

```

EXPLORER
CODE_ANALYSIS
code_explanation.py
code_review.py
refactoring.py
test_generation.py
OUTLINE
TIMELINE
test_generation.py
Edit with Copilot
Agent Mode
Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.
or type # to attach context
Add Context...
test_generation.py Current file
Edit files in your workspace in agent
Agent
GPT-4.1

Copilot wrote function test_divide_numbers() which holds the test cases.



```python
def divide_numbers(a, b):
    if b == 0:
        raise ZeroDivisionError("Denominator cannot be zero")
    return a / b
#write test cases for divide_numbers function using pytest
def test_divide_numbers():
    assert divide_numbers(10, 2) == 5
    assert divide_numbers(9, 3) == 3
    assert divide_numbers(-6, 2) == -3
    assert divide_numbers(0, 1) == 0
    try:
        divide_numbers(5, 0)
    except ZeroDivisionError as e:
        assert str(e) == "Denominator cannot be zero"
```

Edit with Copilot

Agent Mode

Ask Copilot to edit your files in agent mode. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

Add Context...
test_generation.py Current file
Edit files in your workspace in agent
Agent
GPT-4.1

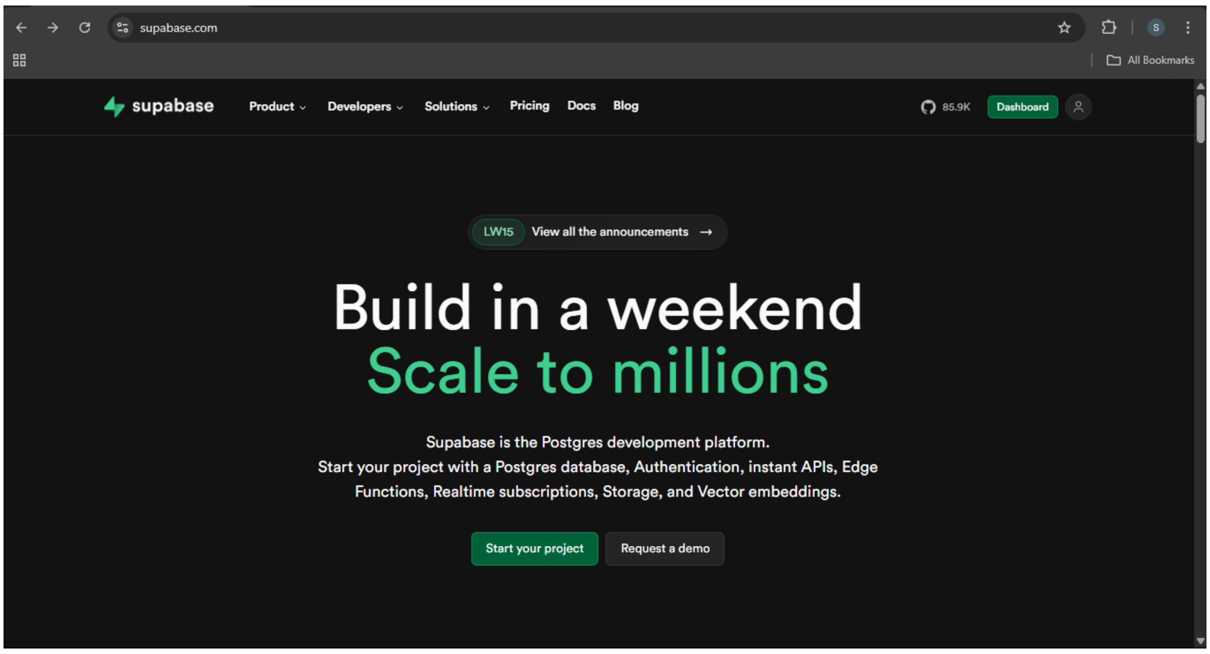
Using GitHub Copilot, I explored 4 powerful use cases in real development workflows:

File Name	Use Case
code_explanation.py	Code understanding
refactoring.py	Code cleanup & optimization
code_review.py	Code review & suggestions
test_generation.py	Auto test generation

Q2

Step 1: Open Supabase and Log In

I opened supabase.com, created an account, and logged into the dashboard to start a new project.

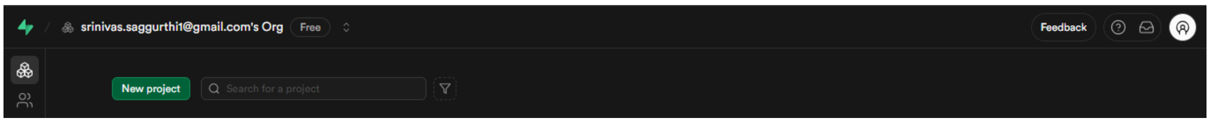


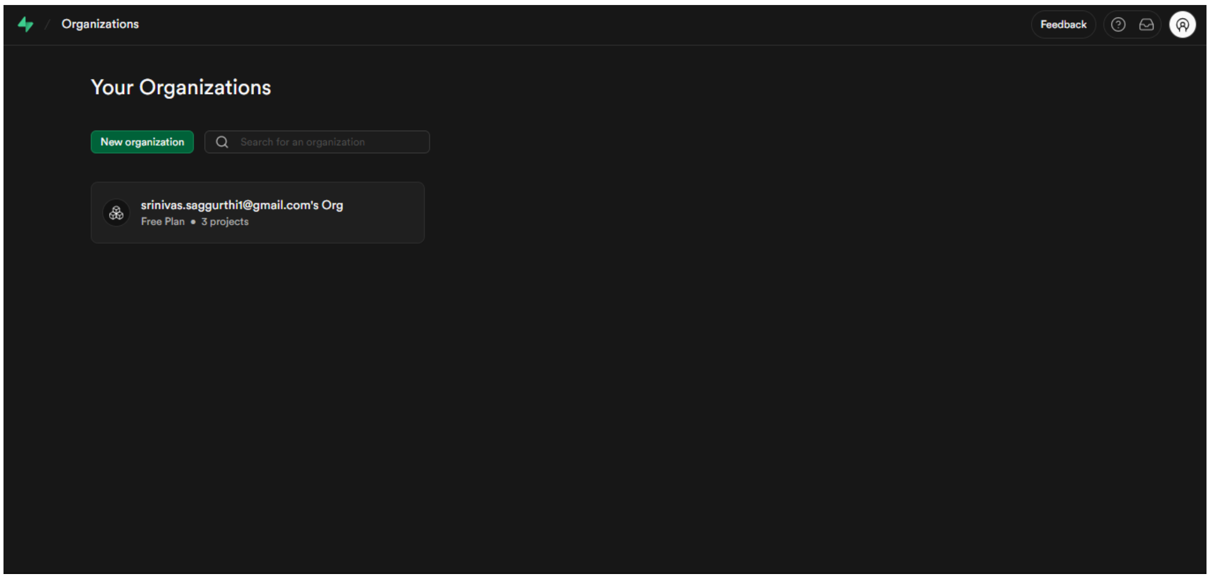
Build in a weekend
Scale to millions

Supabase is the Postgres development platform.
Start your project with a Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings.

Step 2: Create a New Supabase Project

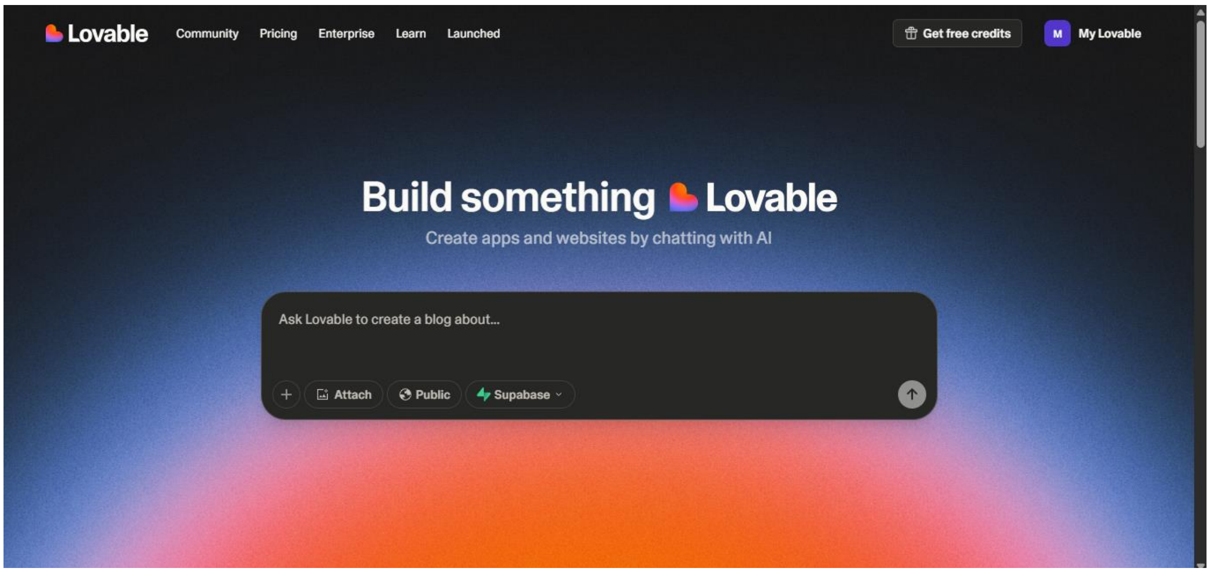
I clicked on “New Project”, gave it a name, selected the region, and set a strong password. Once the project was ready, I copied the Project URL and anon/public API key—which are needed to connect with Lovable.





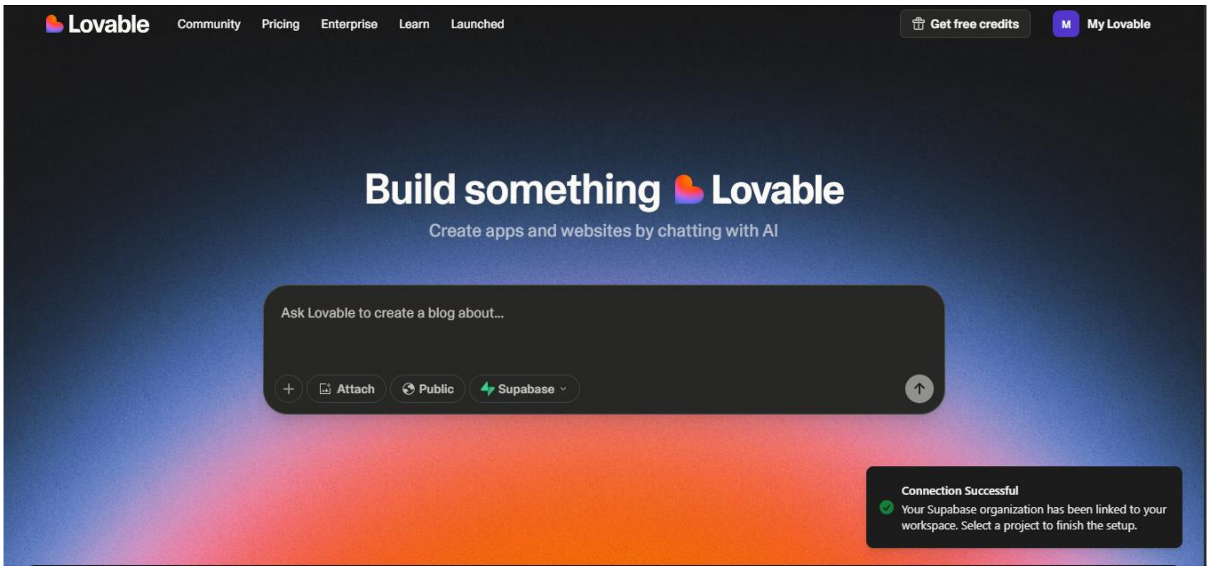
Step 3: Open Lovable.dev and Log In

I went to lovable.dev, signed in using my account, and entered the dashboard where prompts can be written to generate full-stack apps.



Step 4: Connect Lovable with Supabase

Under the Lovable prompt bar, I clicked “Connect to Supabase” , and pasted my Supabase Project URL and anon/public key. This allowed Lovable to directly access and use my Supabase project.



Connection Successful
Your Supabase organization has been linked to your workspace. Select a project to finish the setup.

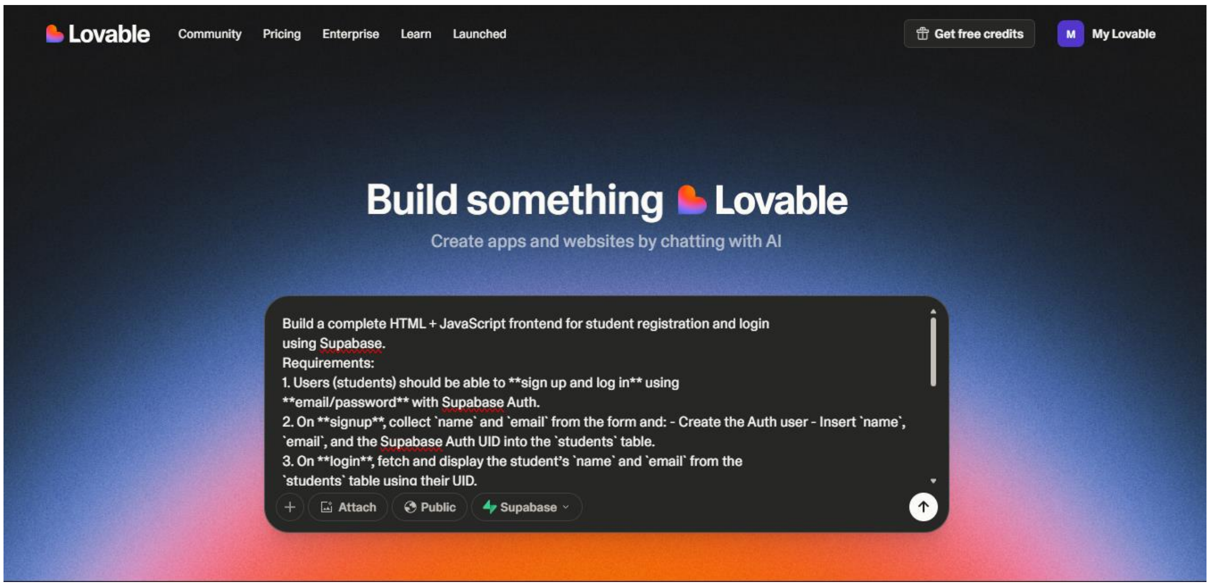
Step 5: Paste the Prompt in Lovable

I entered the prompt in Lovable:

Build a complete HTML + JavaScript frontend for student registration and login using Supabase..................

Lovable then generated the full working code including:

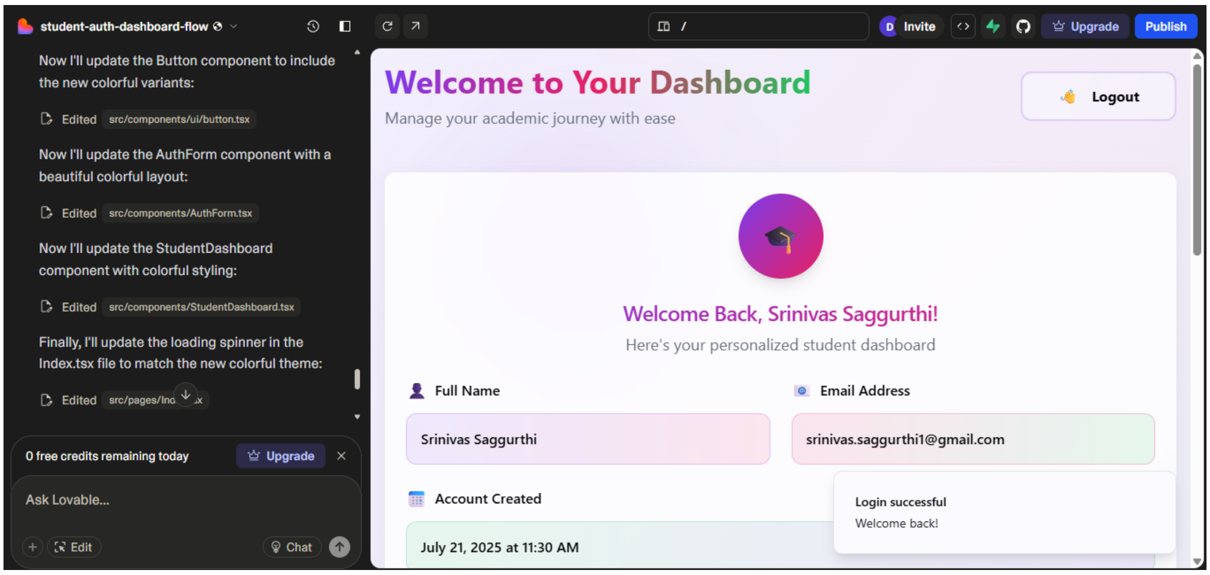
- Signup form with name, email, password
- Login form
- Auth integration with Supabase
- Data insertion into the `student's` table



Build a complete HTML + JavaScript frontend for student registration and login using Supabase.
Requirements:
1. Users (students) should be able to **sign up and log in** using **email/password** with Supabase Auth.
2. On **signup**, collect `name` and `email` from the form and: - Create the Auth user - Insert `name`, `email`, and the Supabase Auth UID into the `students` table.
3. On **login**, fetch and display the student's `name` and `email` from the `students` table using their UID.

Step 6: Modify the App (If Needed)

After the app was generated, I previewed it and made minor changes (like button labels, colors, or field validations) using Lovable's built-in editor.



Now I'll update the Button component to include the new colorful variants:

Edited src/components/ui/button.tsx

Now I'll update the AuthForm component with a beautiful colorful layout:

Edited src/components/AuthForm.tsx

Now I'll update the StudentDashboard component with colorful styling:

Edited src/components/StudentDashboard.tsx

Finally, I'll update the loading spinner in the Index.tsx file to match the new colorful theme:

Edited src/pages/Ind x

0 free credits remaining today

Welcome to Your Dashboard

Manage your academic journey with ease

Welcome Back, Srinivas Saggurthi!

Here's your personalized student dashboard

Full Name: Srinivas Saggurthi

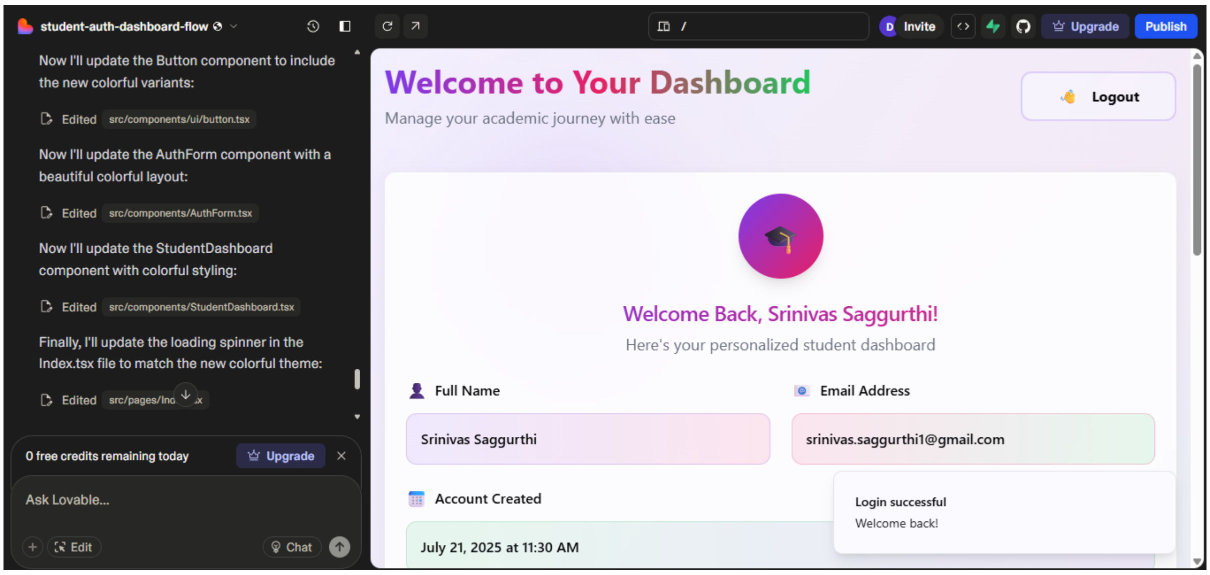
Email Address: srinivas.saggurthi1@gmail.com

Account Created: July 21, 2025 at 11:30 AM

Login successful
Welcome back!

Step 7: Log In and Get Redirected to Home Page

I signed up/logged in using the frontend. Upon successful login, I was redirected to the **Home page**, where a welcome message and user details were shown.


student-auth-dashboard-flow

Now I'll update the Button component to include the new colorful variants:

Edited src/components/ui/button.tsx

Now I'll update the AuthForm component with a beautiful colorful layout:

Edited src/components/AuthForm.tsx

Now I'll update the StudentDashboard component with colorful styling:

Edited src/components/StudentDashboard.tsx

Finally, I'll update the loading spinner in the Index.tsx file to match the new colorful theme:

Edited src/pages/Ind x

0 free credits remaining today

Ask Lovable...

Welcome to Your Dashboard

Manage your academic journey with ease

Welcome Back, Srinivas Saggurthi!

Here's your personalized student dashboard

Full Name	Email Address
Srinivas Saggurthi	srinivas.saggurthi1@gmail.com

Account Created

July 21, 2025 at 11:30 AM

Login successful

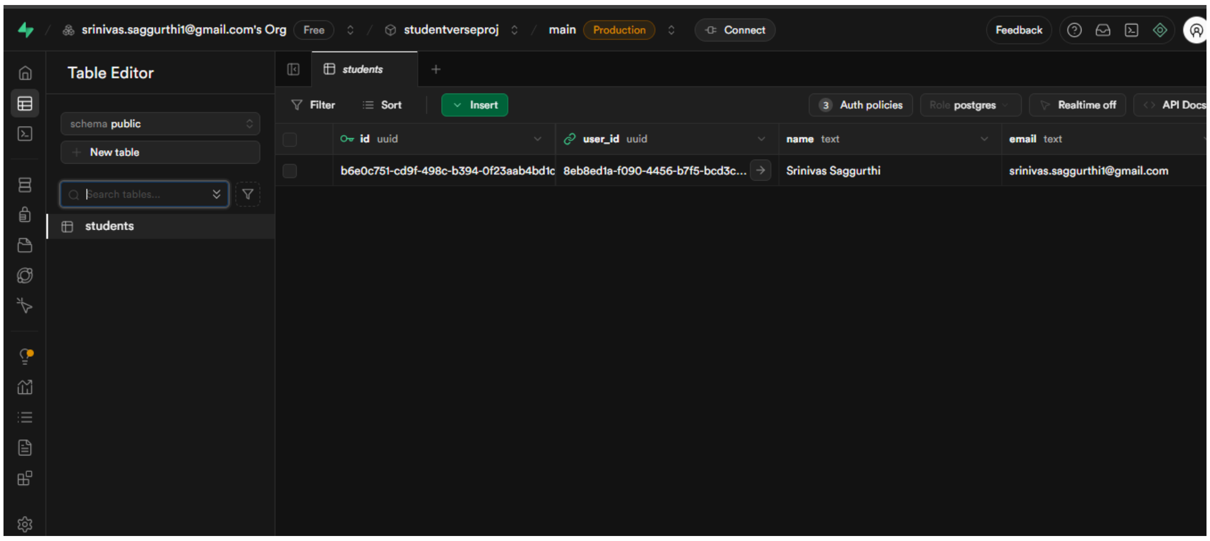
Welcome back!

Step 8: View User Data in Supabase Table

In Supabase, I opened the `students` table to verify the data. It contained:

- Auth UID
- Student name
- Email
- Timestamp

Data was also shown in JSON format under the "Row Details" view.



id uuid	user_id uuid	name text	email text
b6e0c751-cd9f-498c-b394-0f23aab4bd1c	8eb8ed1a-f090-4456-b7f5-bcd3c...	Srinivas Saggurthi	srinivas.saggurthi1@gmail.com

Final Result

This project successfully implements:

- Student Signup & Login using Supabase Auth
- Realtime data insertion and fetching from `student's` table
- Responsive frontend using HTML + JS
- Live hosting using Lovable.dev