East Coast Oracle Users Conference

# AUTONOMOUS CODE GENERATION WITH AI

Ahmed Aboulnaga
November 4, 2025 @ 10:10am EST
Room 10

# DISCLAIMER

*The views I express today are my own and do not necessarily reflect the views of my employer or its affiliates.*

# TABLE OF CONTENTS

## 01 OVERVIEW

Concepts and terminology

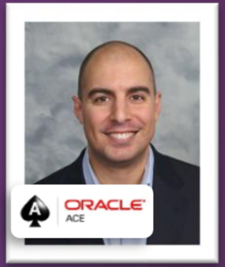
## 02 DEMO

Hands-on walkthrough

## 03 FINAL THOUGHTS

Summary, conclusion, and takeaways

# ABOUT ME

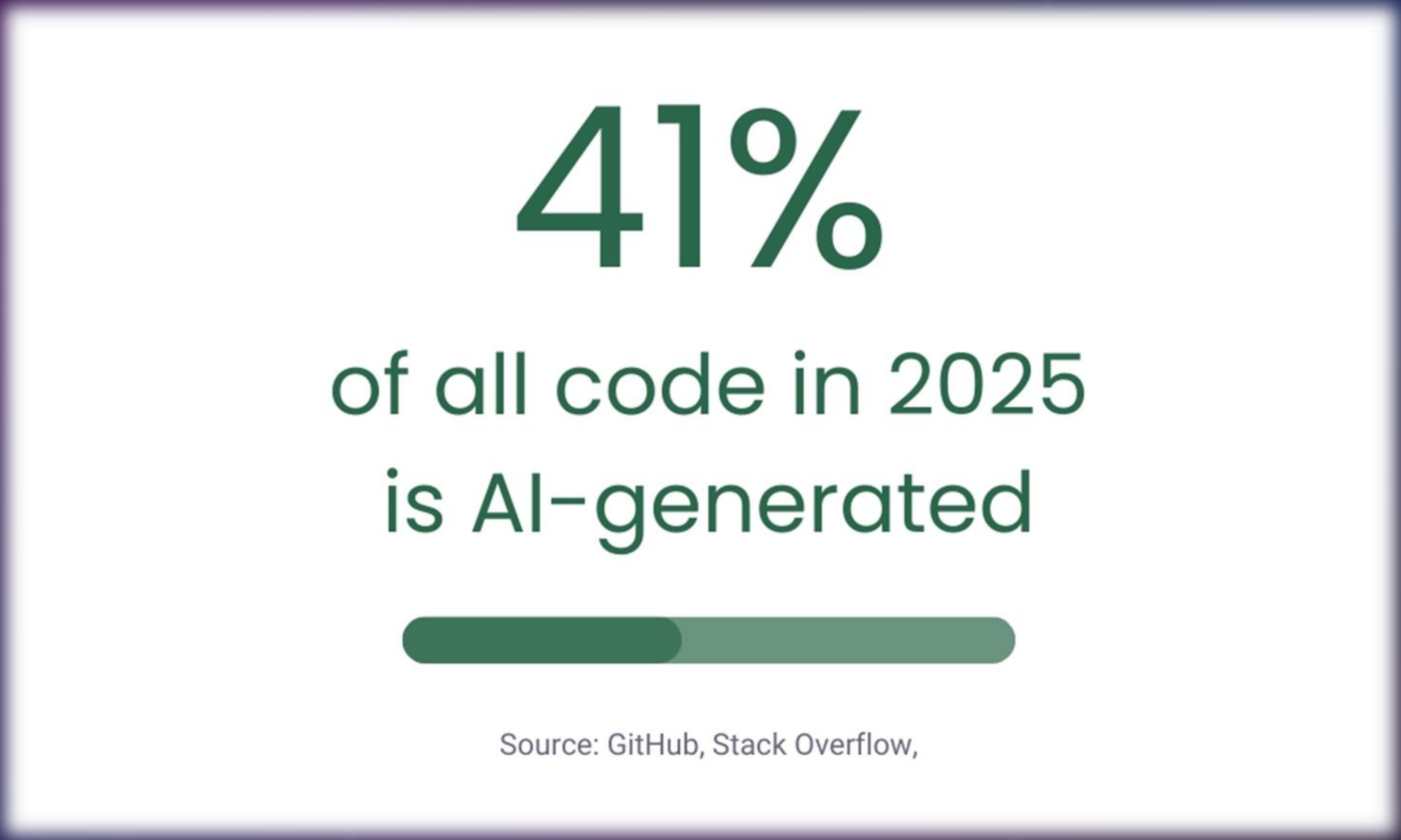


## AHMED ABOULNAGA

- Master's degree in Computer Science from GMU
- 25+ years in Information Technology consulting
- Recent emphasis on cloud, DevOps, middleware, cybersecurity
- Author, presenter, blogger CHRONICLER.TECH
- Author of "DevSecOps in Oracle Cloud" and others
- Oracle ACE
- Passionate about technology, movie buff, and comic book fan

ORACLE AI World
GLOC
MODULAR OPEN SYSTEMS SUMMIT
UKOUG
DATA SUMMIT
Georgia Oracle Users Group
WMOUG
SEMOP
eco East Coast Oracle Users Conference
COUG
ORACLE OPEN WORLD
COLLABORATE
Quest Oracle Community
RMOUG
BGOUG
DOAG
NATCAP-OUG
MOUS
POUG
ODTUG
utoug

# 1. OVERVIEW



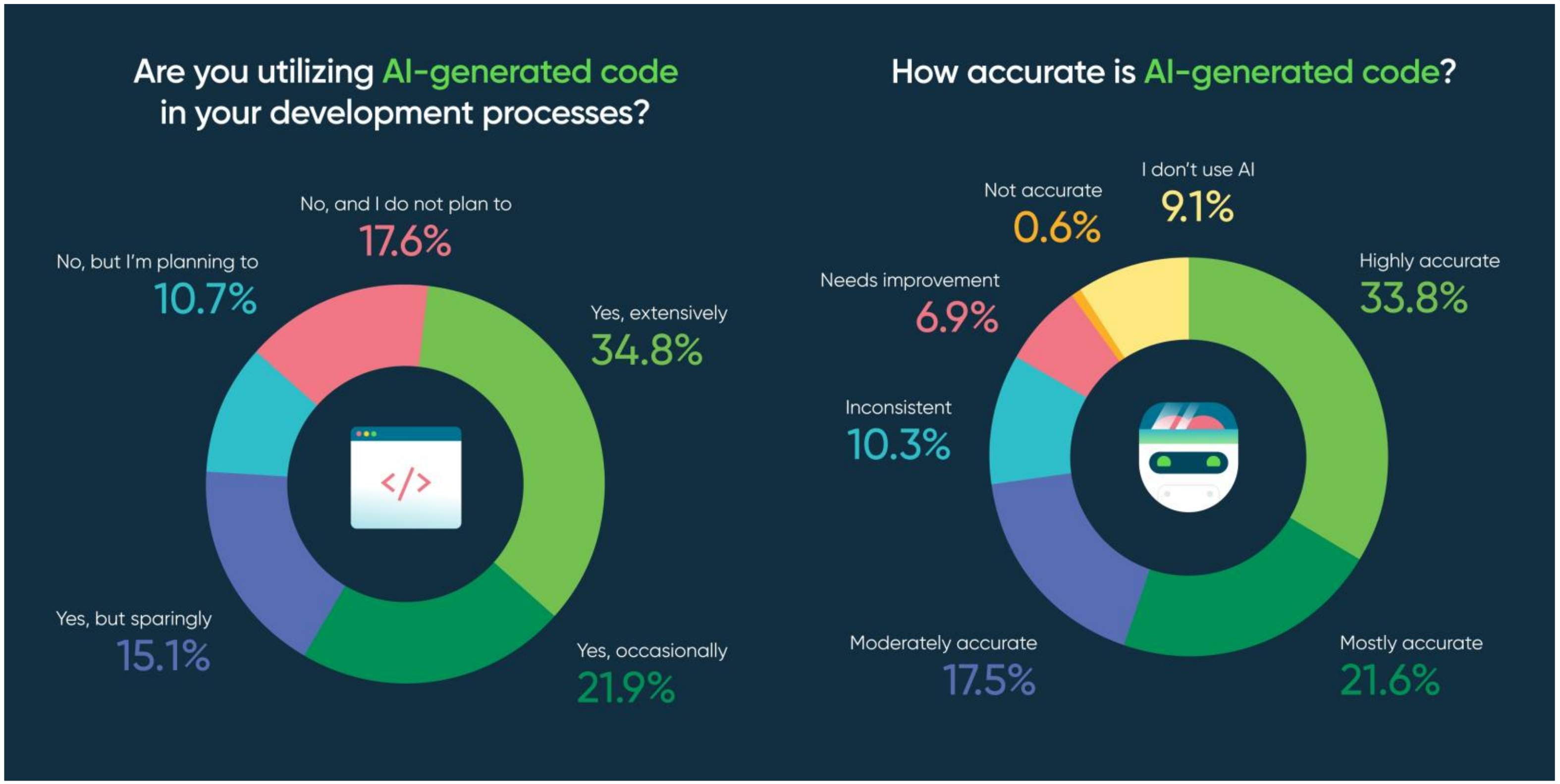
# 41%

of all code in 2025
is AI-generated

Source: GitHub, Stack Overflow,

https://www.netcorpsoftwaredevelopment.com/blog/ai-generated-code-statistics

# STATISTICS AT A GLANCE

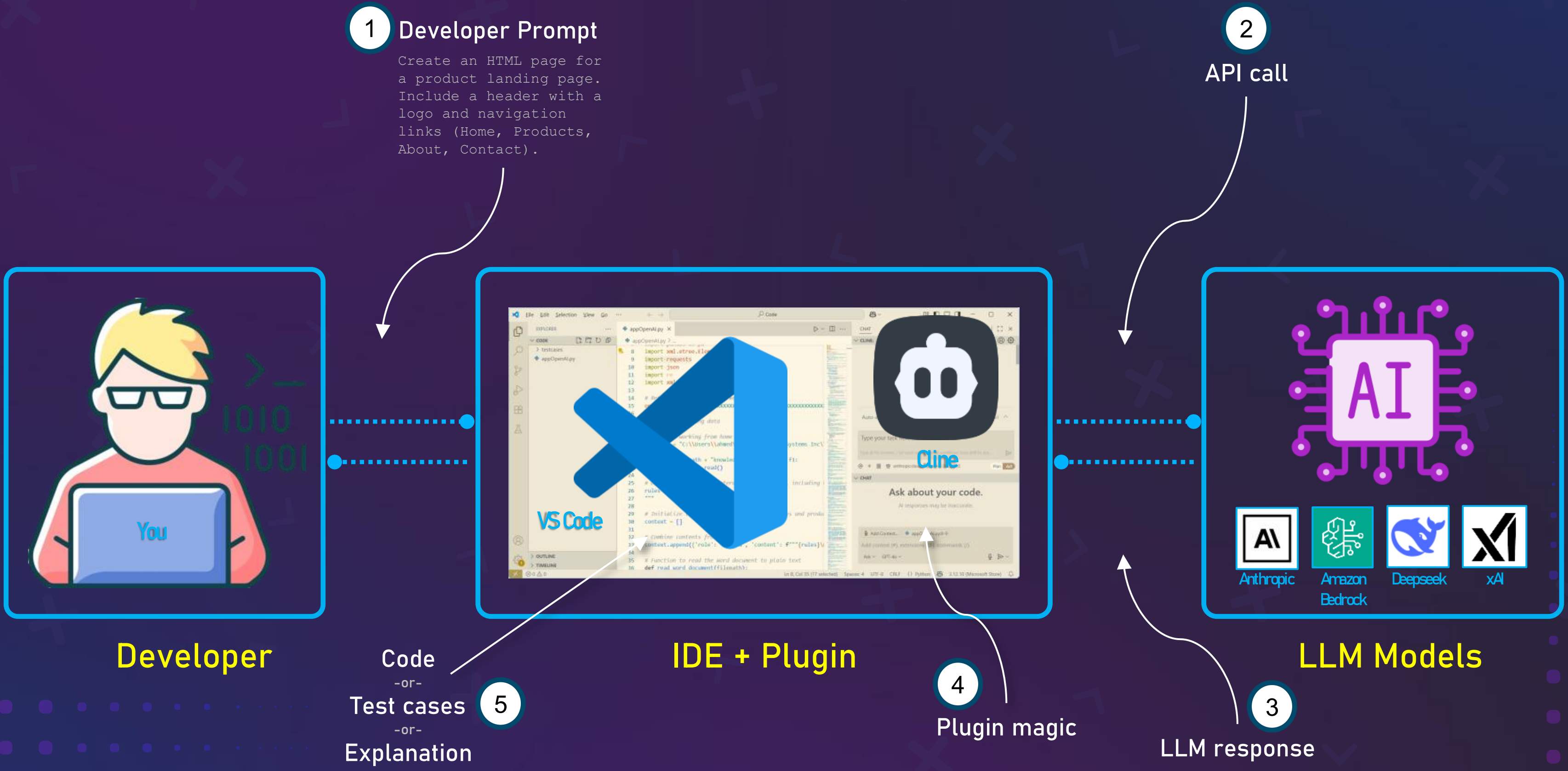


https://www.servicenow.com/platform/generative-ai/generative-ai-usage-in-coding-survey.html

# WHY AI CODE GENERATION?

- Enables developers to generate code faster
- Accelerates debugging and troubleshooting
- Boosts developer productivity
- Rapid experimentation
- Quickly and efficiently test and debug code
- Reduces the work of manually writing lines of code
- Alleviates developer’s mental load and reduces burnout
- Frees developers to focus on higher-value work
- Makes code development accessible to non-developers

# HOW AI CODE GENERATION WORKS



1 Developer Prompt

Create an HTML page for a product landing page. Include a header with a logo and navigation links (Home, Products, About, Contact).

2 API call

3 LLM response

4 Plugin magic

5 Code -or- Test cases -or- Explanation

Developer

IDE + Plugin

LLM Models

# TERMINOLOGY

## LLM

- Large Language Model
- A deep learning model trained on a massive dataset

## Natural Language Processing (NLP)

- A branch of AI
- Understands human language
- Extracts insights from unstructured text

## Generative AI

- AI systems that can generate text, images, video, or other output such as code
- Utilizes LLMs
- Deep-learning model that takes raw data and "learn" to generate statistically probable outputs when prompted

## AI Code Completion

- Automated suggestions for completing the current line of code as a developer types

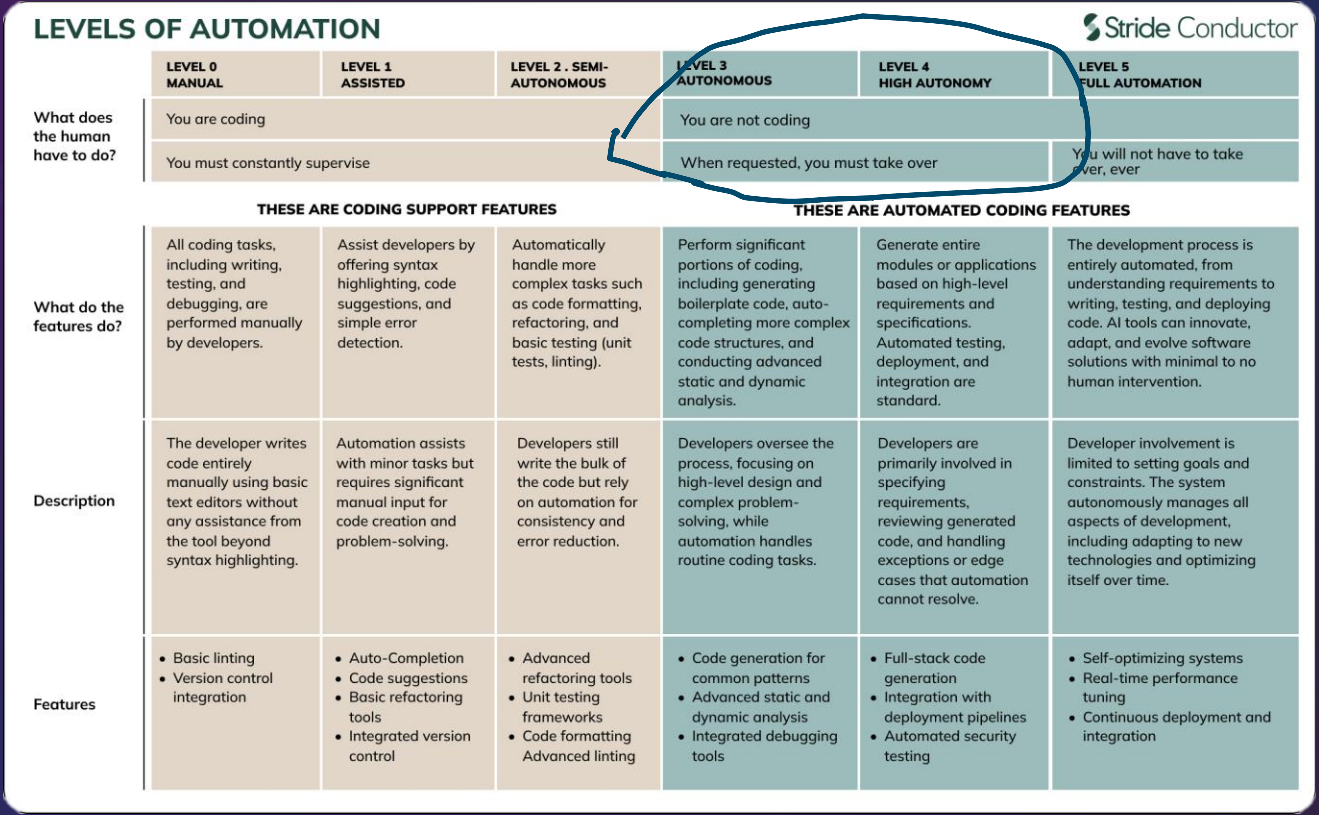
## AI Code Generation

- Relies on machine learning and natural language processing to automatically generate code

## Vibe Coding

- Aka *AI Assisted Coding*
- Aka *AI pair programming*
- Includes code suggestions dynamically and interactively
- Delivers a fluid and intuitive coding experience
- AI is an active collaborator instead of a passive assistant

# LEVELS OF AUTOMATION



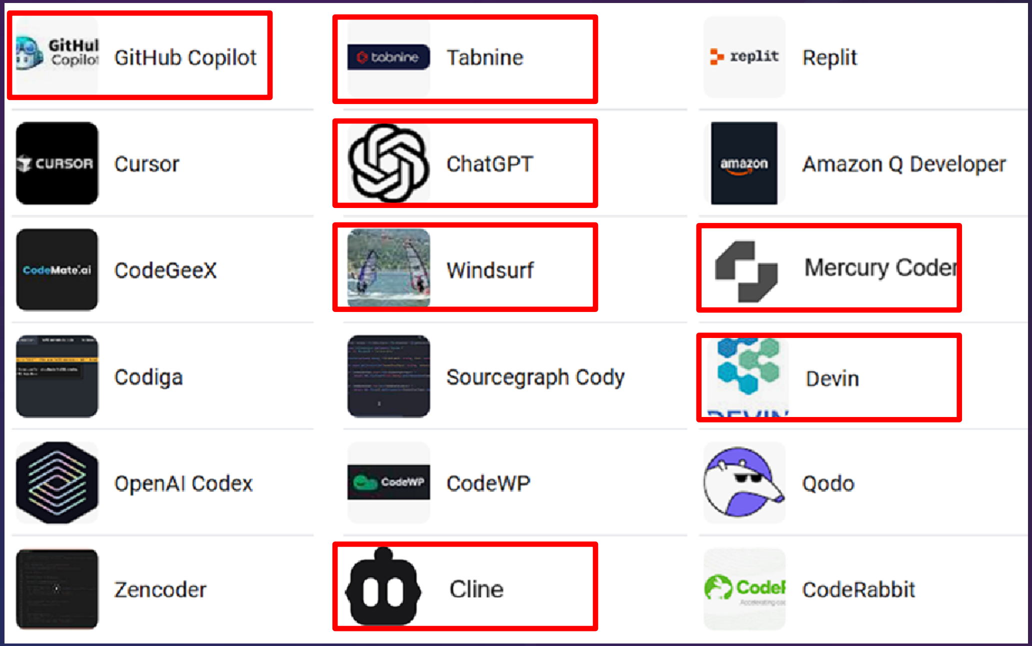
## LEVELS OF AUTOMATION

Stride Conductor

| | LEVEL 0<br>MANUAL | LEVEL 1<br>ASSISTED | LEVEL 2 . SEMI-AUTONOMOUS | LEVEL 3<br>AUTONOMOUS | LEVEL 4<br>HIGH AUTONOMY | LEVEL 5<br>FULL AUTOMATION |
|---|---|---|---|---|---|---|
| What does the human have to do? | You are coding | | | You are not coding | | |
| | You must constantly supervise | | | When requested, you must take over | | You will not have to take over, ever |
| | THESE ARE CODING SUPPORT FEATURES | | | THESE ARE AUTOMATED CODING FEATURES | | |
| What do the features do? | All coding tasks, including writing, testing, and debugging, are performed manually by developers. | Assist developers by offering syntax highlighting, code suggestions, and simple error detection. | Automatically handle more complex tasks such as code formatting, refactoring, and basic testing (unit tests, linting). | Perform significant portions of coding, including generating boilerplate code, auto-completing more complex code structures, and conducting advanced static and dynamic analysis. | Generate entire modules or applications based on high-level requirements and specifications. Automated testing, deployment, and integration are standard. | The development process is entirely automated, from understanding requirements to writing, testing, and deploying code. AI tools can innovate, adapt, and evolve software solutions with minimal to no human intervention. |
| Description | The developer writes code entirely manually using basic text editors without any assistance from the tool beyond syntax highlighting. | Automation assists with minor tasks but requires significant manual input for code creation and problem-solving. | Developers still write the bulk of the code but rely on automation for consistency and error reduction. | Developers oversee the process, focusing on high-level design and complex problem-solving, while automation handles routine coding tasks. | Developers are primarily involved in specifying requirements, reviewing generated code, and handling exceptions or edge cases that automation cannot resolve. | Developer involvement is limited to setting goals and constraints. The system autonomously manages all aspects of development, including adapting to new technologies and optimizing itself over time. |
| Features | • Basic linting<br>• Version control integration | • Auto-Completion<br>• Code suggestions<br>• Basic refactoring tools<br>• Integrated version control | • Advanced refactoring tools<br>• Unit testing frameworks<br>• Code formatting Advanced linting | • Code generation for common patterns<br>• Advanced static and dynamic analysis<br>• Integrated debugging tools | • Full-stack code generation<br>• Integration with deployment pipelines<br>• Automated security testing | • Self-optimizing systems<br>• Real-time performance tuning<br>• Continuous deployment and integration |

https://www.stride.build/thought-leadership/the-6-levels-of-codegen-automation

# AI CODING ASSISTANTS



GitHub Copilot
Cursor
CodeGeeX
Codiga
OpenAI Codex
Zencoder

Tabnine
ChatGPT
Windsurf
Sourcegraph Cody
CodeWP
Cline

Replit
Amazon Q Developer
Mercury Coder
Devin
Qodo
CodeRabbit

⚠ *The overwhelming majority of AI coding assistants are in beta*

## GITHUB COPILOT

- Pair programming integrated into IDEs
- Popular due to its integration with GitHub

## TABNINE

- Mostly good for autocomplete

## CHATGPT

- General purpose
- Not integrated into an IDE

## WINDSURF

- Formerly Codeium
- AI-native IDE

## CLINE

- My personal favorite
- Only works with Visual Studio Code

## MERCURY CODER

- Based on Diffusion LLM = super fast
- Still an experimental concept

## DEVIN

- Aims for autonomous codegen; plan, write, test, debug, submit PRs

# RISKS AND CHALLENGES

- **Low quality code** ––– hallucinations, lack of consistency and coding standards
    - Do this → Always review AI generated code
- **Security vulnerabilities** ––– data poisoning is a real problem
    - Do this → Continue scanning your code with SAST tools
- **Data privacy** ––– your code and data could be used for model training
    - Do this → Consider self-hosted AI coding assistants and/or verify vendor's policies
- **Licensing & copyright risks** ––– could include open-source code without attribution
    - Do this → Run generated code through license scanners and/or verify vendor's policies
- **Cost** ––– the more calls made, the more $$$ required
- **Speed** ––– complex tasks can take considerable time (~5-50 minutes)
- **Risk of technical debt** ––– can create long-term maintenance challenges
- **Skills atrophy & loss of developer control** ––– too much developer dependency on AI can contribute to reduced understanding of codebase

# COMPARE AND CONTRAST

| | Brief Description | Strengths | Weaknesses | Score |
|---|---|---|---|---|
| **GitHub Copilot** | AI pair programmer powered by OpenAI Codex, now GPT-4-based. Integrated directly into VS Code, JetBrains, and more. | - Deep IDE integration<br>- Fast inline suggestions<br>- Great for repetitive code<br>- Learns from your code context | - Occasionally produces insecure or outdated code<br>- Limited reasoning/explanation<br>- Struggles with large multi-file logic | 8.5 |
| **Tabnine** | AI assistant trained on permissively licensed code; focuses on privacy/local models. | - Privacy-friendly (can run locally)<br>- Lightweight and fast<br>- Supports many languages and IDEs | - Limited reasoning ability<br>- Weaker with complex tasks | 7.5 |
| **Cline** | A local, privacy-focused AI code assistant with deep context awareness (used via VS Code). | - Runs with your chosen model<br>- Excellent context retention | - Depends on external LLM quality<br>- Fewer prebuilt integrations | 8.7 |
| **ChatGPT** | Conversational AI by OpenAI that can write, debug, and explain code with strong reasoning. | - Exceptional explanations<br>- Handles complex, multi-file logic<br>- Great for learning and debugging | - No project context, broad quality<br>- No live IDE integration<br>- May over-explain or hallucinate | 7.5 |
| **Windsurf** | An AI IDE that combines real-time coding, refactoring, and chat-based assistance. | - Combines chat + live IDE context<br>- Understands full projects<br>- Good UI/UX and workflow | - Smaller ecosystem<br>- Some latency in large projects | 8.8 |
| **Devin** | Marketed as the first "autonomous AI software engineer" capable of executing projects end-to-end. | - Can plan, code, test, and debug autonomously<br>- High-level task management<br>- Strong reasoning and multi-step execution | - Extremely resource-intensive<br>- Reliability varies | 9.0 |

*IDE Integration*
**GitHub Copilot**

*Privacy*
**Tabnine** or **Cline**

*Explanation*
**ChatGPT**

*All-in-One*
**Windsurf**

*Most Autonomous*
**Devin**

⚠ *Scores here are subjective*

# 2. DEMO

# REQUIREMENTS

**Functional Requirements**

1. Create a registration form the ECO conference
2. Include a list of questions which I will provide
3. The web application should use a modern, responsive UI
4. The web application should use a CSS template or style similar to what is found on https://github.com/projects

**Non-Functional Requirements**

1. The application should be developed in Python
2. The application will be deployed to Oracle Cloud Infrastructure

# REQUIREMENTS

ROLE: <<< I am a web developer. >>>

CONTEXT: <<< I need help creating a prompt to use in my AI code generator. >>>

EXPECTATION: <<< I want you to create me a prompt that I will use in my AI code generator to create a web application for me. The details of this web application are as follows:
* The web application will be called "ECO Registration Form".
* The web application should use a modern, responsive UI.
* The web application should use a CSS template or style similar to what is found on https://github.com/projects.
* The web application is comprised of 2 pages total.
* None of the pages requires authentication, so they are publicly accessible.
* The first page is a form which is filled out by a user.
* Upon submitting the form, the user is taken to the second page, which is a confirmation page.
* On the second page, behind the scenes an email is sent out to ahmed@aboulnaga.com with all the information submitted in the form presented in table format in an HTML email using the Oracle Cloud Infrastructure (OCI) Email Delivery service.
* On the second page a message will be presented
been successfully submitted. The "homeicon.png" i
hyperlinked back to the first page.
* Each page will have the “ecologo.jpg" in the midd
* On the bottom of each page will be a footer that s
* The web application will be deployed to Oracle Cl
* Any combination of services is acceptable, but the
* The web application should be developed in Pyth
* Test cases must be provided to test the applicatio
* The code should include comments describing so
* Detailed instructions to create the infrastructure
* Detailed instructions to deploy the web application to OCI provided.
- Include anything relevant you t be adde

*meta prompting*

ChatGPT

*Let’s use ChatGPT to refine my prompt and tailor it to Cline*

Build "ECO Registration Form" Web Application

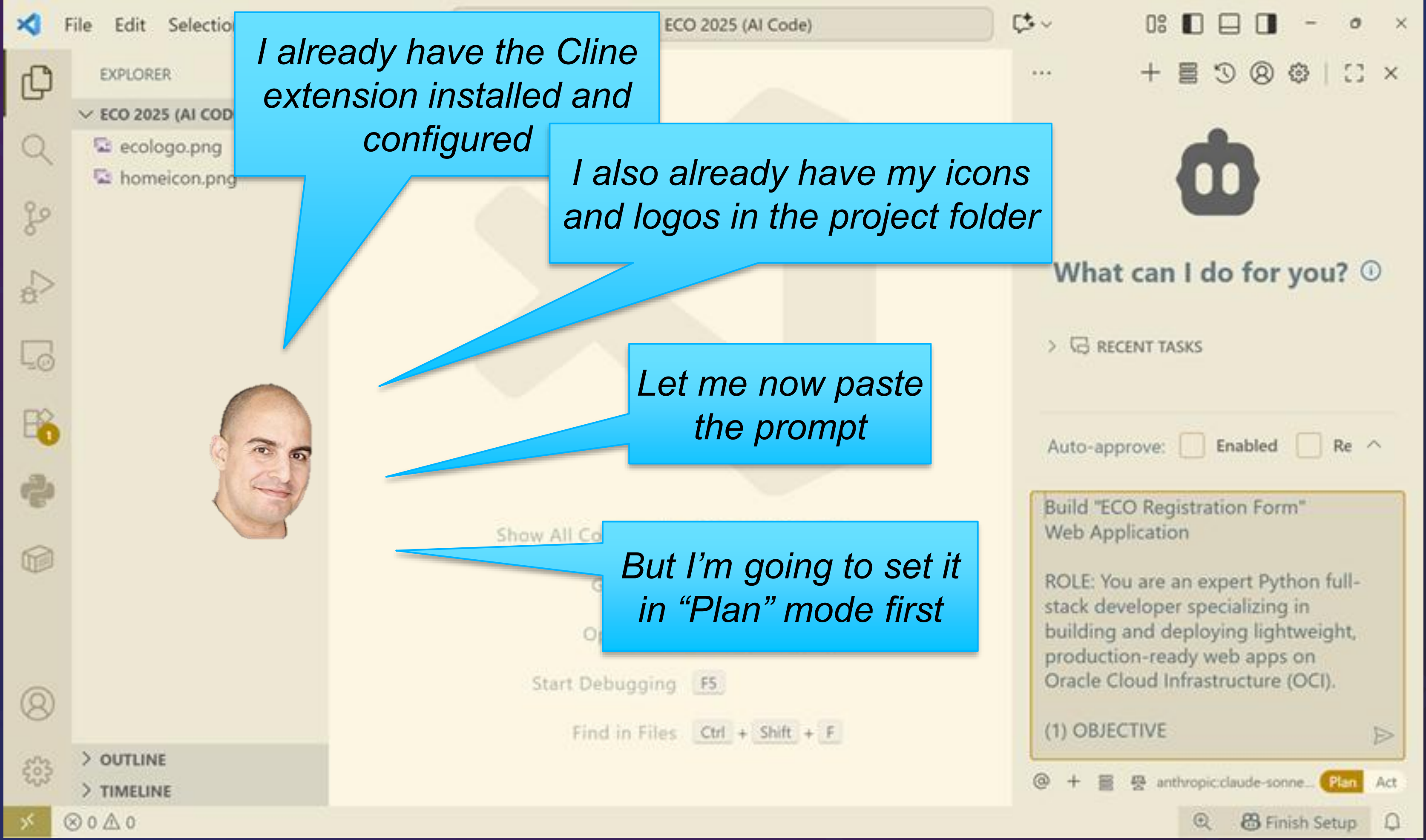
OLE: You are an expert Python full-stack developer specializing in building and deploying
htweight, production-ready web apps on Oracle Cloud Infrastructure (OCI).

) OBJECTIVE

ild a complete, deployable web application called “ECO Registration Form” that meets the
llowing requirements and outputs fully working code, test cases, and deployment instructions.

) OVERVIEW
Name: ECO Registration Form
anguage: Python
* Framework: Flask (or FastAPI if simpler)
* UI Style: Modern and responsive, styled similarly to https://github.com/projects
* Total Pages: 2
* Authentication: None (both pages are public)
* Deployment Target: Oracle Cloud Infrastructure (OCI)

(3) PAGE 1: Registration Form
Purpose: Collect user input.
Layout & Design:
* At the top center: display “ecologo.jpg"
* Form fields to include (all of them are optional):
- Full Name
- Email Address
- Company
- Phone
- Role / Title
- Technologies: (this should be a checkbox with the following options: EBS, Fusion Apps, OCI



Build "ECO Registration Form" Web Application

ROLE: You are an expert Python full-stack developer specializing in building and deploying lightweight, production-ready web apps on Oracle Cloud Infrastructure (OCI).

(1) OBJECTIVE

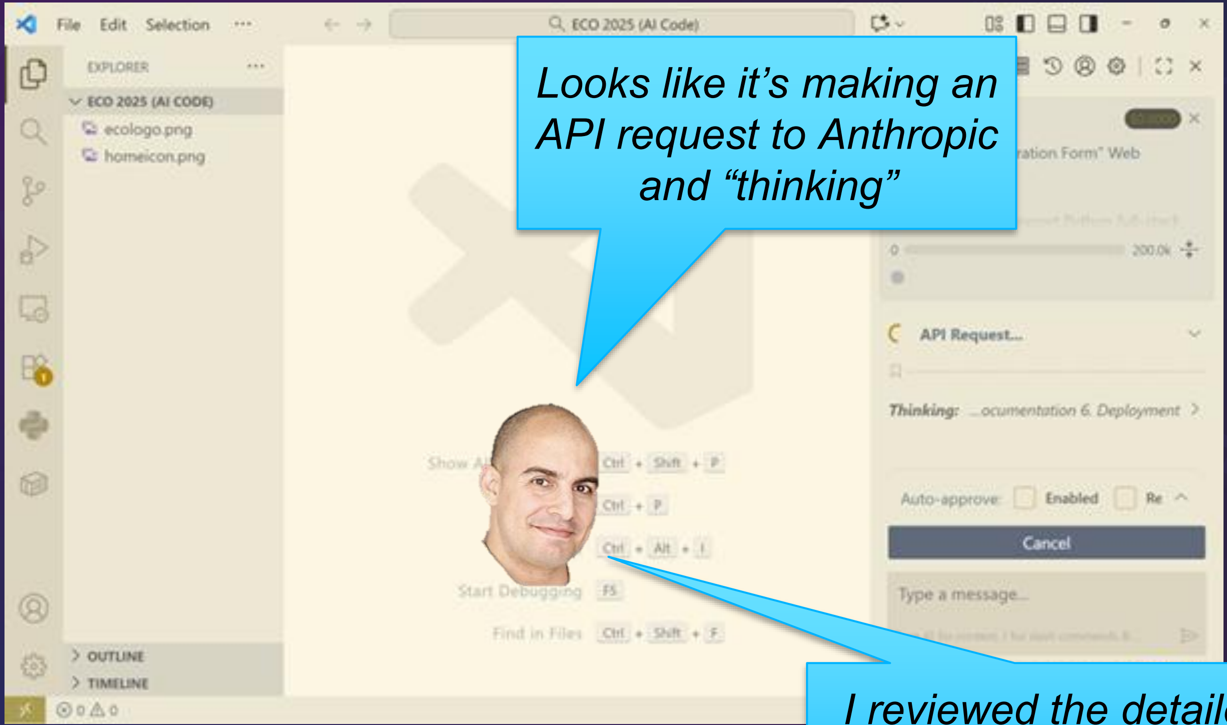
*I already have the Cline extension installed and configured*

*I also already have my icons and logos in the project folder*

*Let me now paste the prompt*

*But I'm going to set it in "Plan" mode first*

*Looks like it’s making an API request to Anthropic and “thinking”*

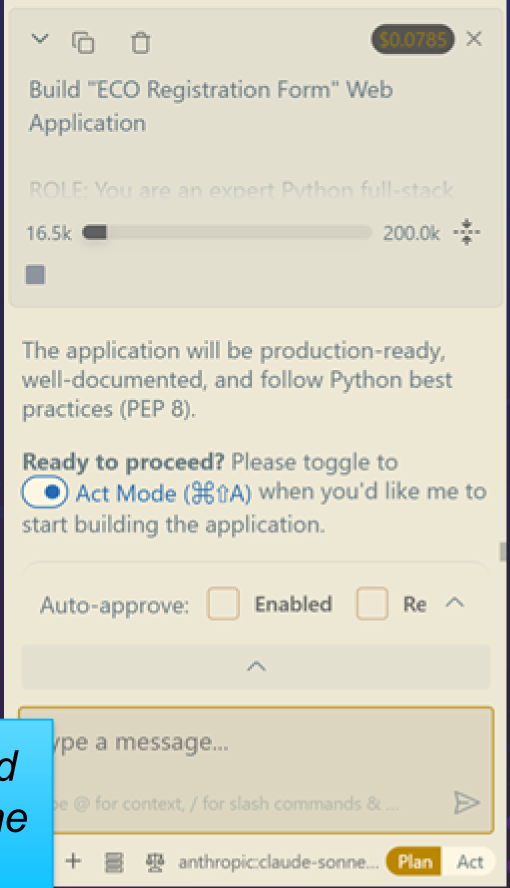


...ration Form" Web

API Request...

**Thinking:** ...ocumentation 6. Deployment

*I reviewed the detailed plan and design it came up with*

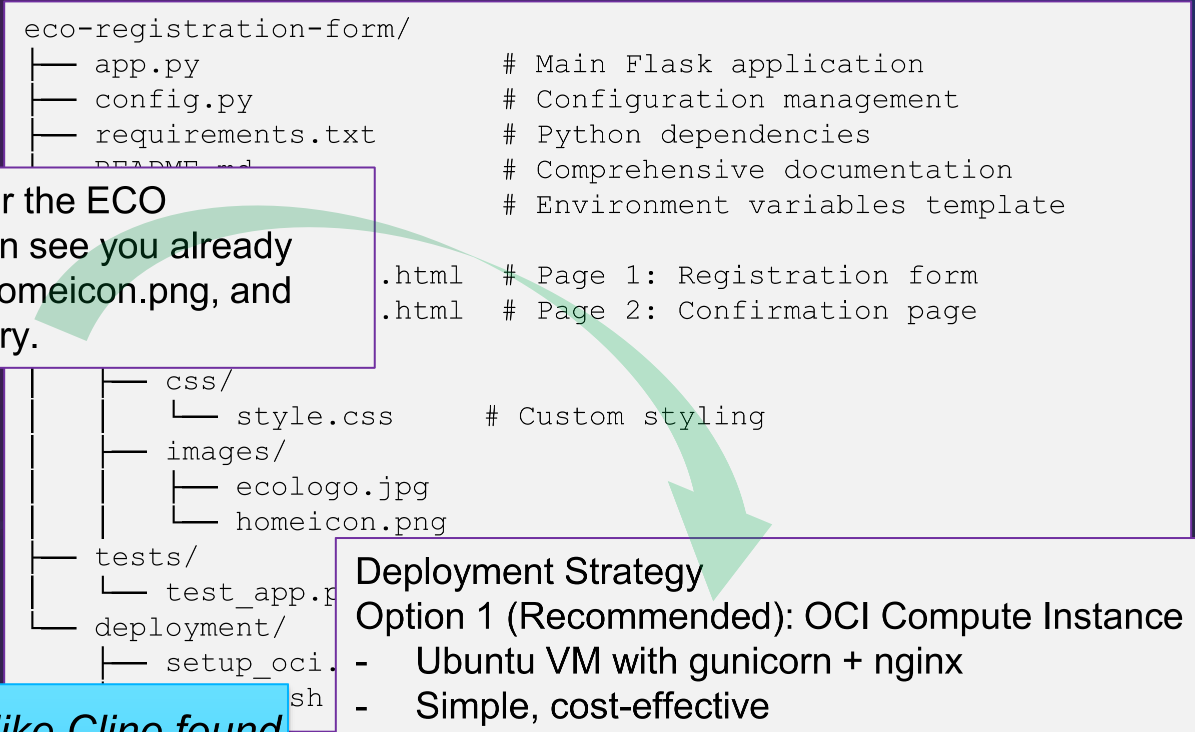

$0.0785
16.5k
200.0k
Auto-approve:
Enabled
Re
Type a message...
Type @ for context, / for slash commands & ...
anthropic:claude-sonne...
Plan
Act

Build "ECO Registration Form" Web Application

ROLE: You are an expert Python full-stack

The application will be production-ready, well-documented, and follow Python best practices (PEP 8).

**Ready to proceed?** Please toggle to Act Mode (⌘⇧A) when you'd like me to start building the application.



```
eco-registration-form/
├── app.py                    # Main Flask application
├── config.py                 # Configuration management
├── requirements.txt          # Python dependencies
│   README.md                 # Comprehensive documentation
                              # Environment variables template

                   .html      # Page 1: Registration form
                   .html      # Page 2: Confirmation page

│   ├── css/
│   │   └── style.css      # Custom styling
│   ├── images/
│   │   ├── ecologo.jpg
│   │   └── homeicon.png
├── tests/
│   └── test_app.p
└── deployment/
    ├── setup_oci.
                sh
```

I've reviewed your requirements for the ECO Registration Form application. I can see you already have the required image assets (homeicon.png, and ecologo.png) in the current directory.

Deployment Strategy

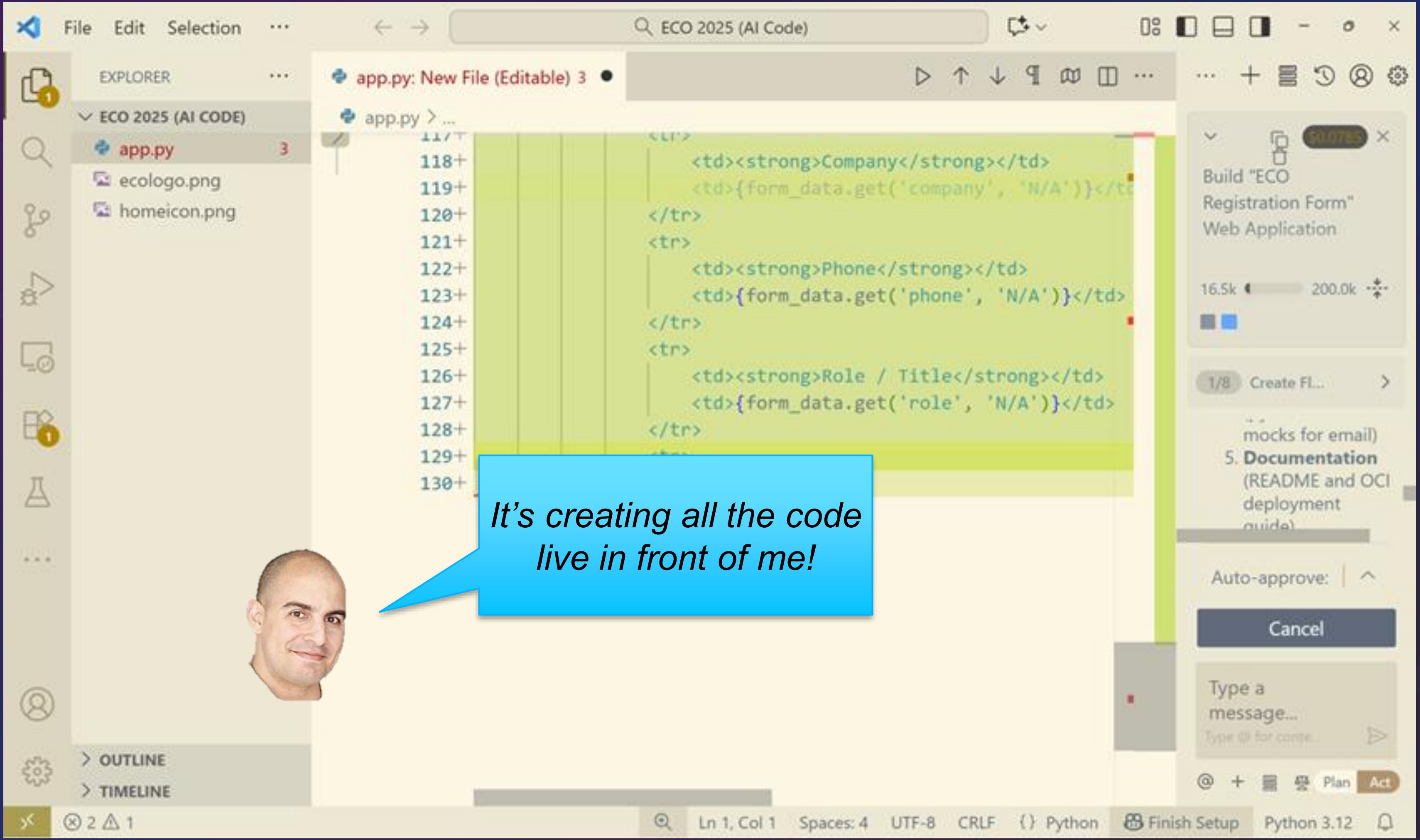
Option 1 (Recommended): OCI Compute Instance

- Ubuntu VM with gunicorn + nginx
- Simple, cost-effective
- Full control over environment

Option 2: OCI Container Instances

- Dockerized application
- More complex but scalable

*Nice! Looks like Cline found my images and came up with a nifty structure*



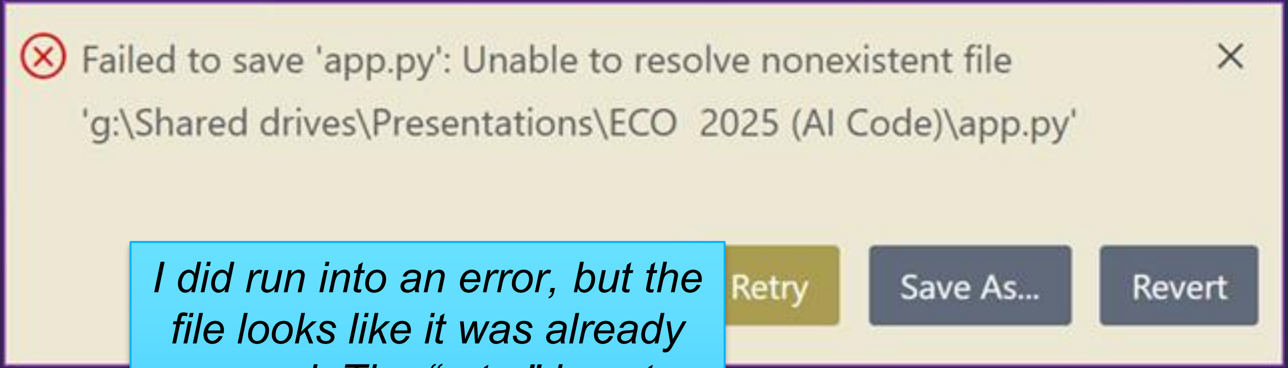
```
                <td><strong>Company</strong></td>
                <td>{form_data.get('company', 'N/A')}</td>
            </tr>
            <tr>
                <td><strong>Phone</strong></td>
                <td>{form_data.get('phone', 'N/A')}</td>
            </tr>
            <tr>
                <td><strong>Role / Title</strong></td>
                <td>{form_data.get('role', 'N/A')}</td>
            </tr>
```

Build "ECO Registration Form" Web Application

mocks for email)

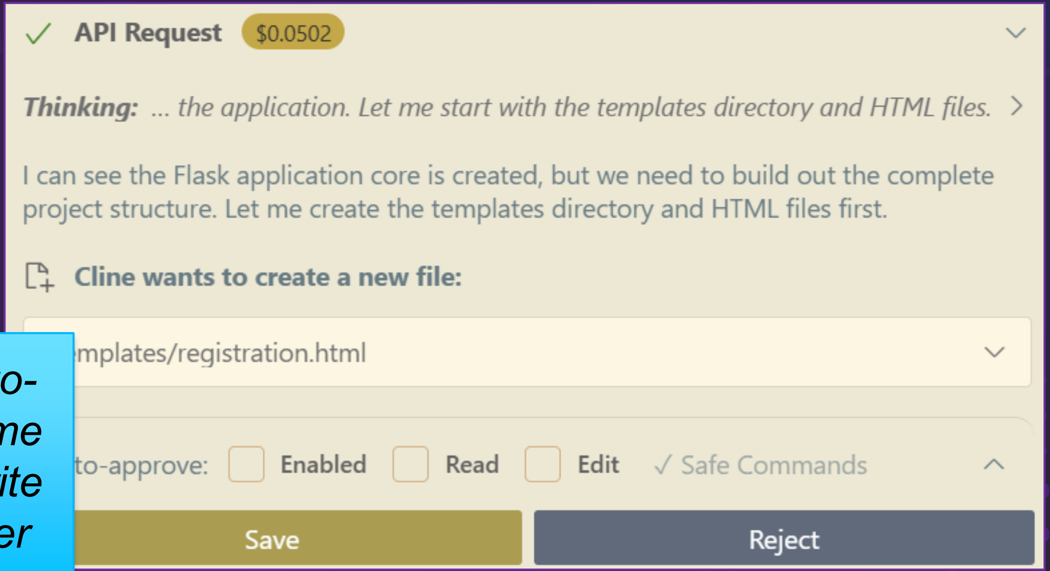
5. **Documentation** (README and OCI deployment guide)

*It's creating all the code live in front of me!*



Failed to save 'app.py': Unable to resolve nonexistent file 'g:\Shared drives\Presentations\ECO  2025 (AI Code)\app.py'

*I did run into an error, but the file looks like it was already saved. The "retry" is not working so I'll just move on*



✓ **API Request** $0.0502

***Thinking:*** *... the application. Let me start with the templates directory and HTML files.*

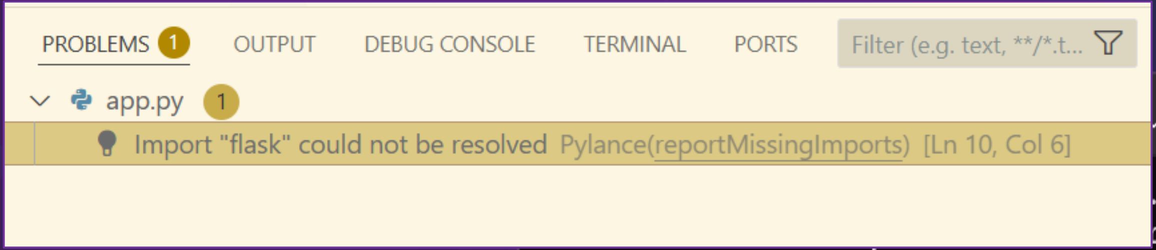
I can see the Flask application core is created, but we need to build out the complete project structure. Let me create the templates directory and HTML files first.

**Cline wants to create a new file:**

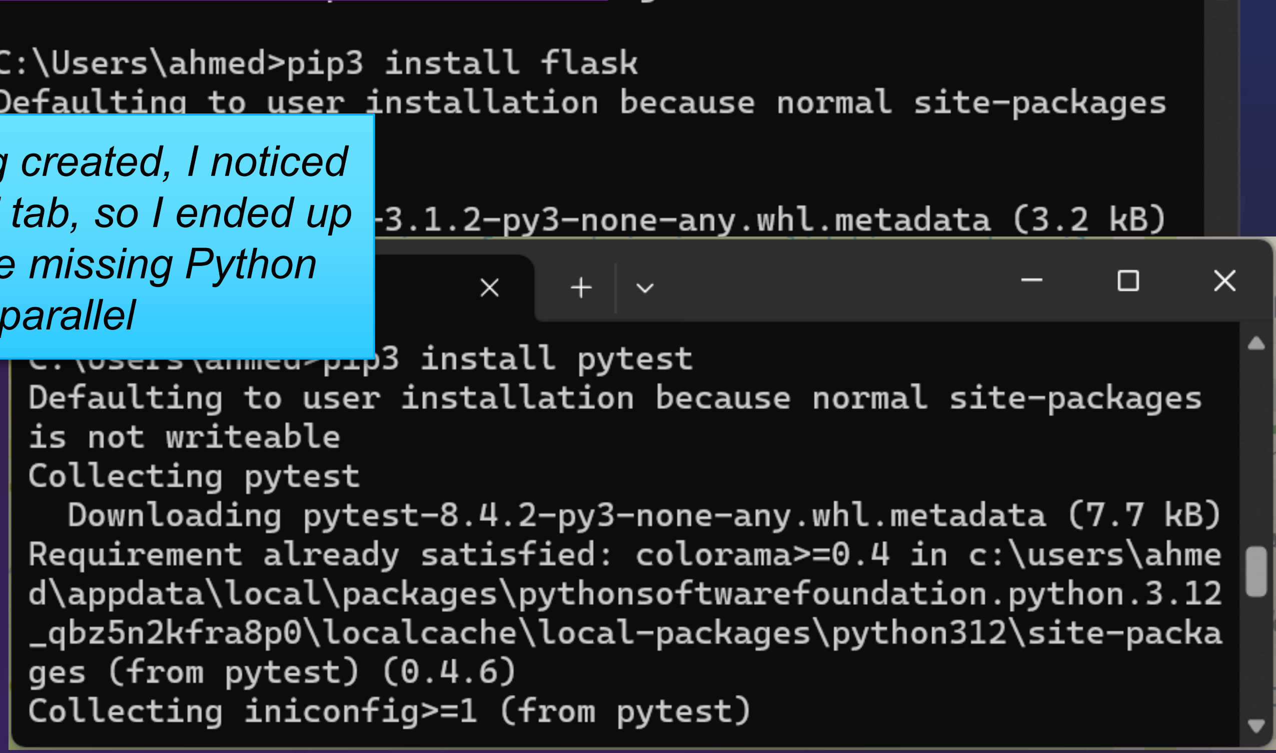
mplates/registration.html

to-approve: ☐ Enabled ☐ Read ☐ Edit ✓ Safe Commands

*And since I didn't enable "auto-approve" it's confirming with me first before performing any write operations to my project folder*



Import "flask" could not be resolved Pylance(reportMissingImports) [Ln 10, Col 6]

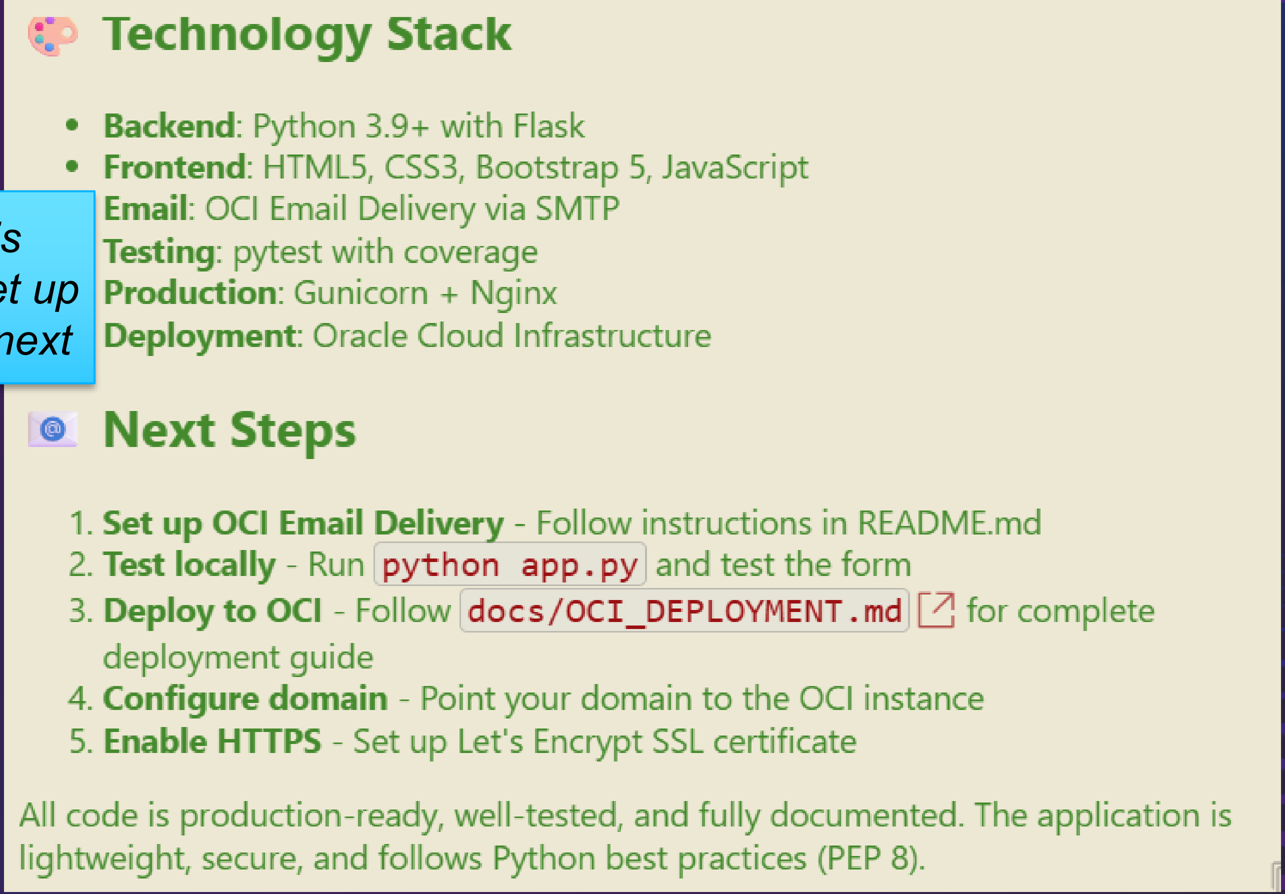


```
.26200.6901]
rights reserved.

C:\Users\ahmed>pip3 install flask
Defaulting to user installation because normal site-packages

-3.1.2-py3-none-any.whl.metadata (3.2 kB)
```

```
C:\Users\ahmed>pip3 install pytest
Defaulting to user installation because normal site-packages
is not writeable
Collecting pytest
  Downloading pytest-8.4.2-py3-none-any.whl.metadata (7.7 kB)
Requirement already satisfied: colorama>=0.4 in c:\users\ahme
d\appdata\local\packages\pythonsoftwarefoundation.python.3.12
_qbz5n2kfra8p0\localcache\local-packages\python312\site-packa
ges (from pytest) (0.4.6)
Collecting iniconfig>=1 (from pytest)
```

*While the code is being created, I noticed issues in the “Problem” tab, so I ended up manually installing the missing Python modules in parallel*

Great! The code is created, so let me set up OCI Email Delivery next



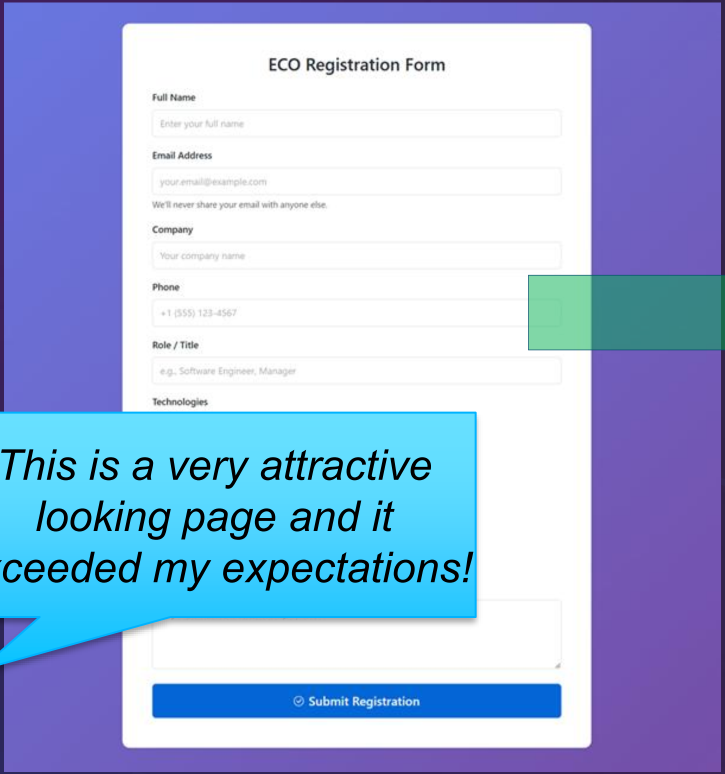
## 🎨 Technology Stack

- **Backend**: Python 3.9+ with Flask
- **Frontend**: HTML5, CSS3, Bootstrap 5, JavaScript
- **Email**: OCI Email Delivery via SMTP
- **Testing**: pytest with coverage
- **Production**: Gunicorn + Nginx
- **Deployment**: Oracle Cloud Infrastructure

## 📧 Next Steps

1. **Set up OCI Email Delivery** - Follow instructions in README.md
2. **Test locally** - Run `python app.py` and test the form
3. **Deploy to OCI** - Follow `docs/OCI_DEPLOYMENT.md` for complete deployment guide
4. **Configure domain** - Point your domain to the OCI instance
5. **Enable HTTPS** - Set up Let's Encrypt SSL certificate

All code is production-ready, well-tested, and fully documented. The application is lightweight, secure, and follows Python best practices (PEP 8).



### ECO Registration Form

Full Name

Enter your full name

Email Address

your.email@example.com

We'll never share your email with anyone else.

Company

Your company name

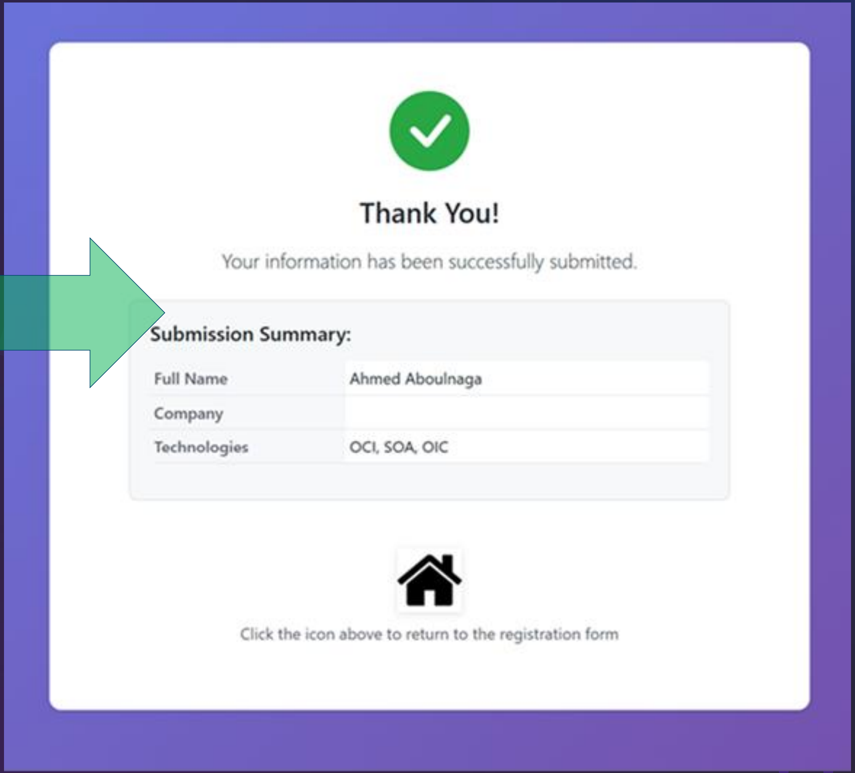
Phone

+1 (555) 123-4567

Role / Title

e.g., Software Engineer, Manager

Technologies



### Thank You!

Your information has been successfully submitted.

**Submission Summary:**

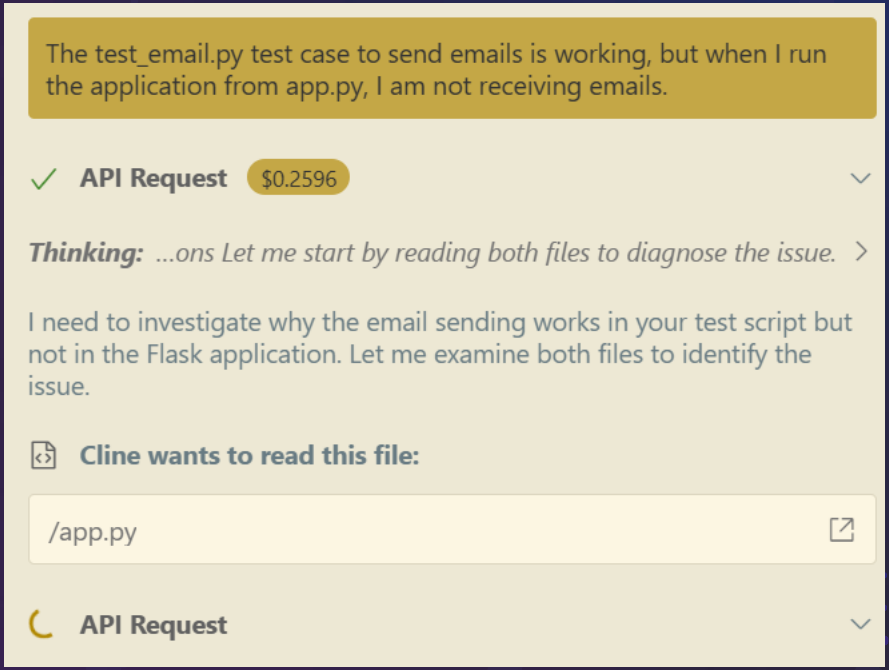
| | |
|---|---|
| Full Name | Ahmed Aboulnaga |
| Company | |
| Technologies | OCI, SOA, OIC |

Click the icon above to return to the registration form

*This is a very attractive looking page and it exceeded my expectations!*

*Email is not working so let me ask Cline to help troubleshoot*

*It's taking too long, has it hung?*



The test_email.py test case to send emails is working, but when I run the application from app.py, I am not receiving emails.

✓ **API Request** $0.2596

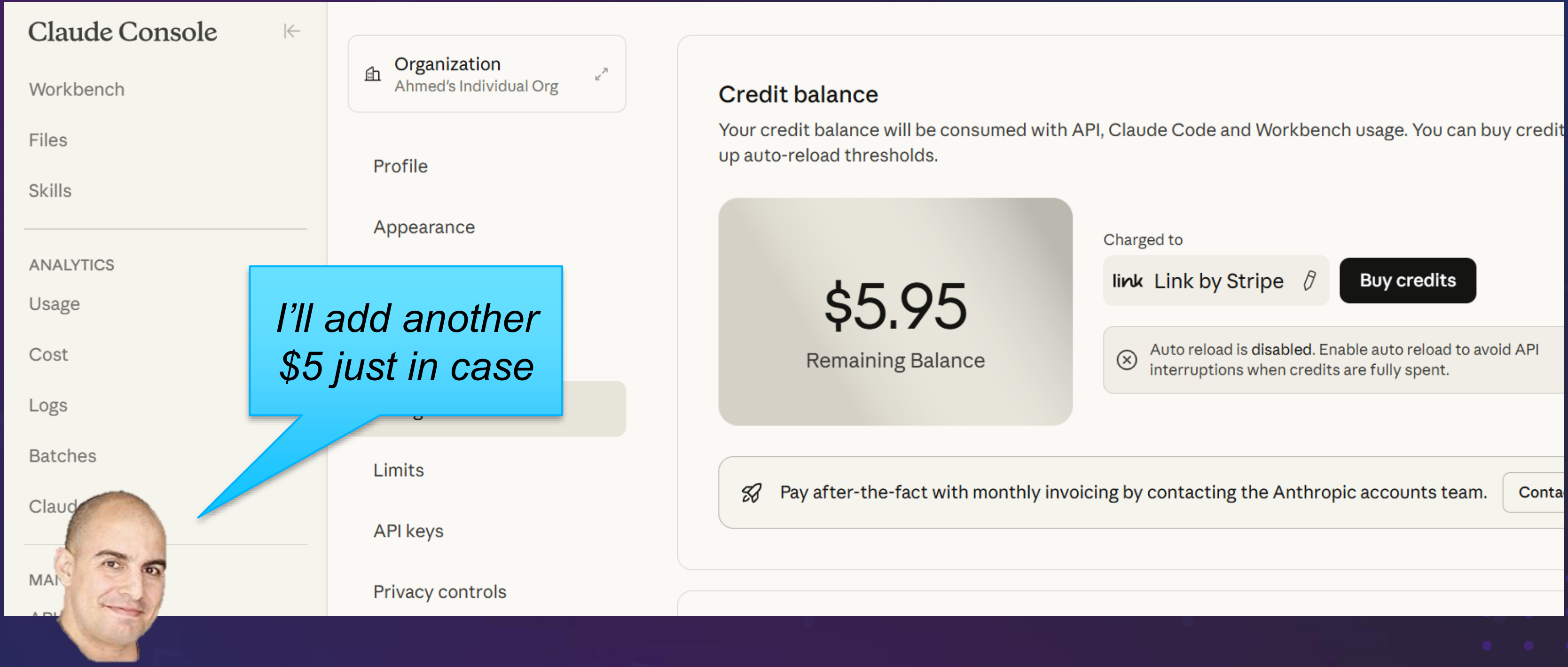
***Thinking:*** *...ons Let me start by reading both files to diagnose the issue.*

I need to investigate why the email sending works in your test script but not in the Flask application. Let me examine both files to identify the issue.

**Cline wants to read this file:**

/app.py

**API Request**



## Credit balance

Your credit balance will be consumed with API, Claude Code and Workbench usage. You can buy credit

up auto-reload thresholds.

$5.95

Remaining Balance

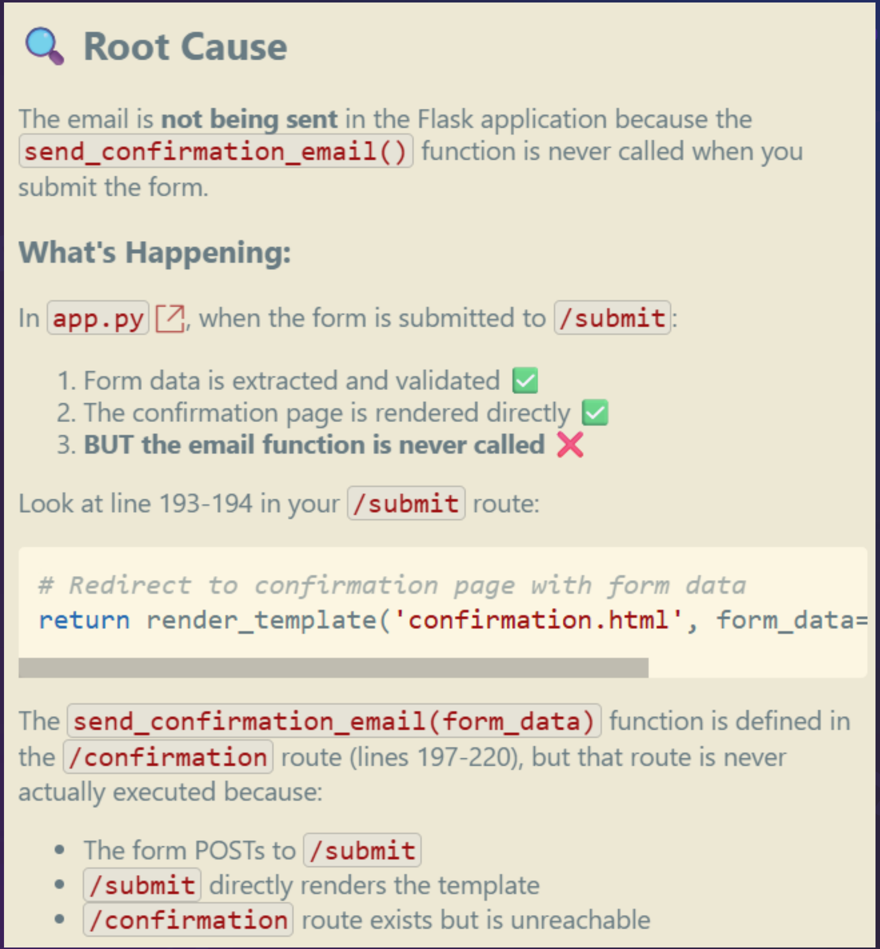
Auto reload is **disabled**. Enable auto reload to avoid API interruptions when credits are fully spent.

Pay after-the-fact with monthly invoicing by contacting the Anthropic accounts team.

*I'll add another $5 just in case*

*It just took a long time thinking, that’s all*

*But it figured out the issue*



## 🔍 Root Cause

The email is **not being sent** in the Flask application because the `send_confirmation_email()` function is never called when you submit the form.

### What's Happening:

In `app.py`, when the form is submitted to `/submit`:

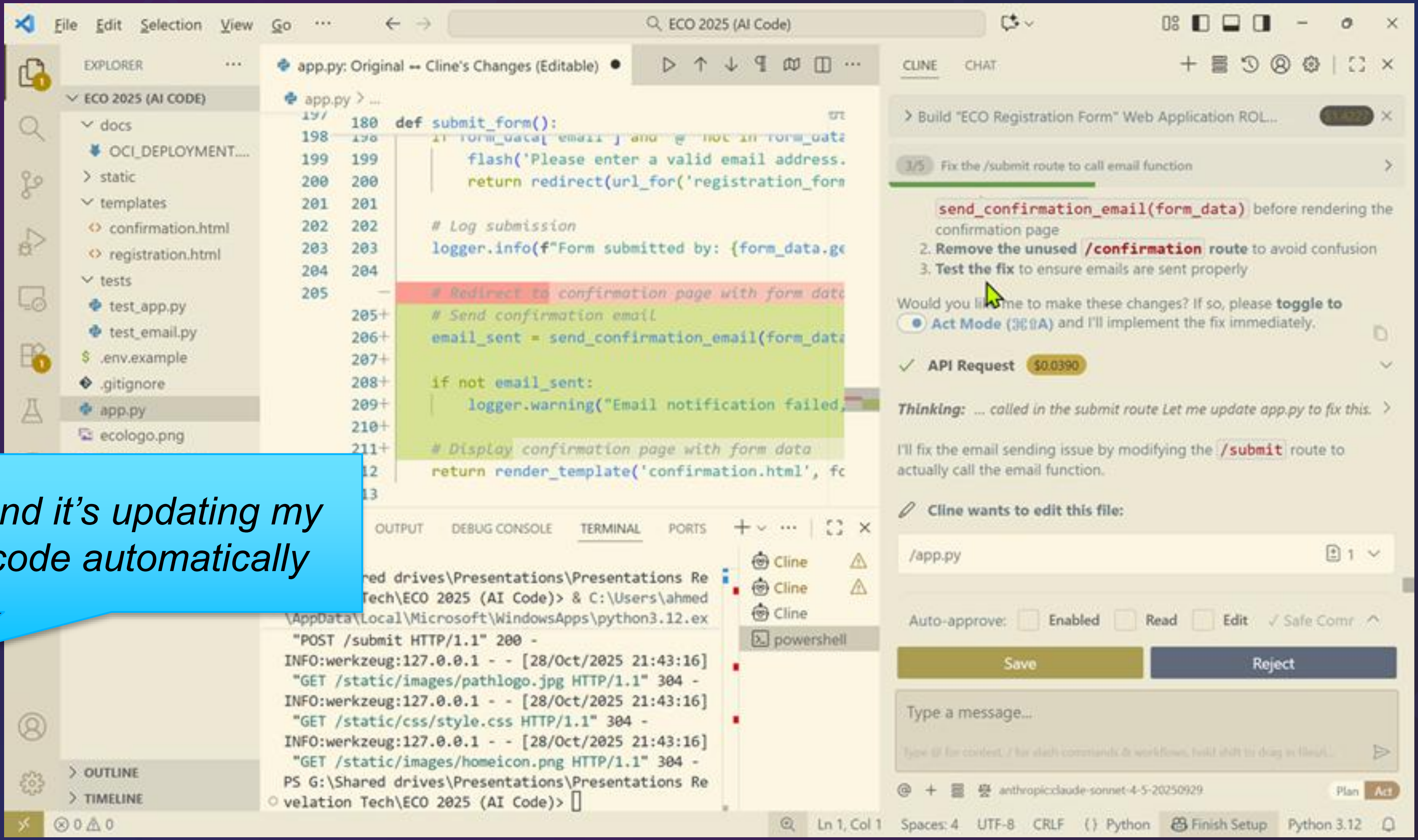
1. Form data is extracted and validated ✅
2. The confirmation page is rendered directly ✅
3. **BUT the email function is never called** ❌

Look at line 193-194 in your `/submit` route:

```
# Redirect to confirmation page with form data
return render_template('confirmation.html', form_data=
```

The `send_confirmation_email(form_data)` function is defined in the `/confirmation` route (lines 197-220), but that route is never actually executed because:

- The form POSTs to `/submit`
- `/submit` directly renders the template
- `/confirmation` route exists but is unreachable



```
def submit_form():
        if form_data['email'] and '@' not in form_data
            flash('Please enter a valid email address.
            return redirect(url_for('registration_form

        # Log submission
        logger.info(f"Form submitted by: {form_data.ge

-       # Redirect to confirmation page with form data
+       # Send confirmation email
+       email_sent = send_confirmation_email(form_data
+
+       if not email_sent:
+           logger.warning("Email notification failed,
+
+       # Display confirmation page with form data
        return render_template('confirmation.html', fo
```

```
red drives\Presentations\Presentations Re
Tech\ECO 2025 (AI Code)> & C:\Users\ahmed
\AppData\Local\Microsoft\WindowsApps\python3.12.ex
 "POST /submit HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2025 21:43:16]
 "GET /static/images/pathlogo.jpg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2025 21:43:16]
 "GET /static/css/style.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [28/Oct/2025 21:43:16]
 "GET /static/images/homeicon.png HTTP/1.1" 304 -
PS G:\Shared drives\Presentations\Presentations Re
velation Tech\ECO 2025 (AI Code)>
```

> Build "ECO Registration Form" Web Application ROL...

3/5 Fix the /submit route to call email function

`send_confirmation_email(form_data)` before rendering the confirmation page

2. **Remove the unused `/confirmation` route** to avoid confusion
3. **Test the fix** to ensure emails are sent properly

Would you like me to make these changes? If so, please **toggle to Act Mode (⌘⇧A)** and I'll implement the fix immediately.

✓ **API Request** $0.0390

***Thinking:*** *... called in the submit route Let me update app.py to fix this.*

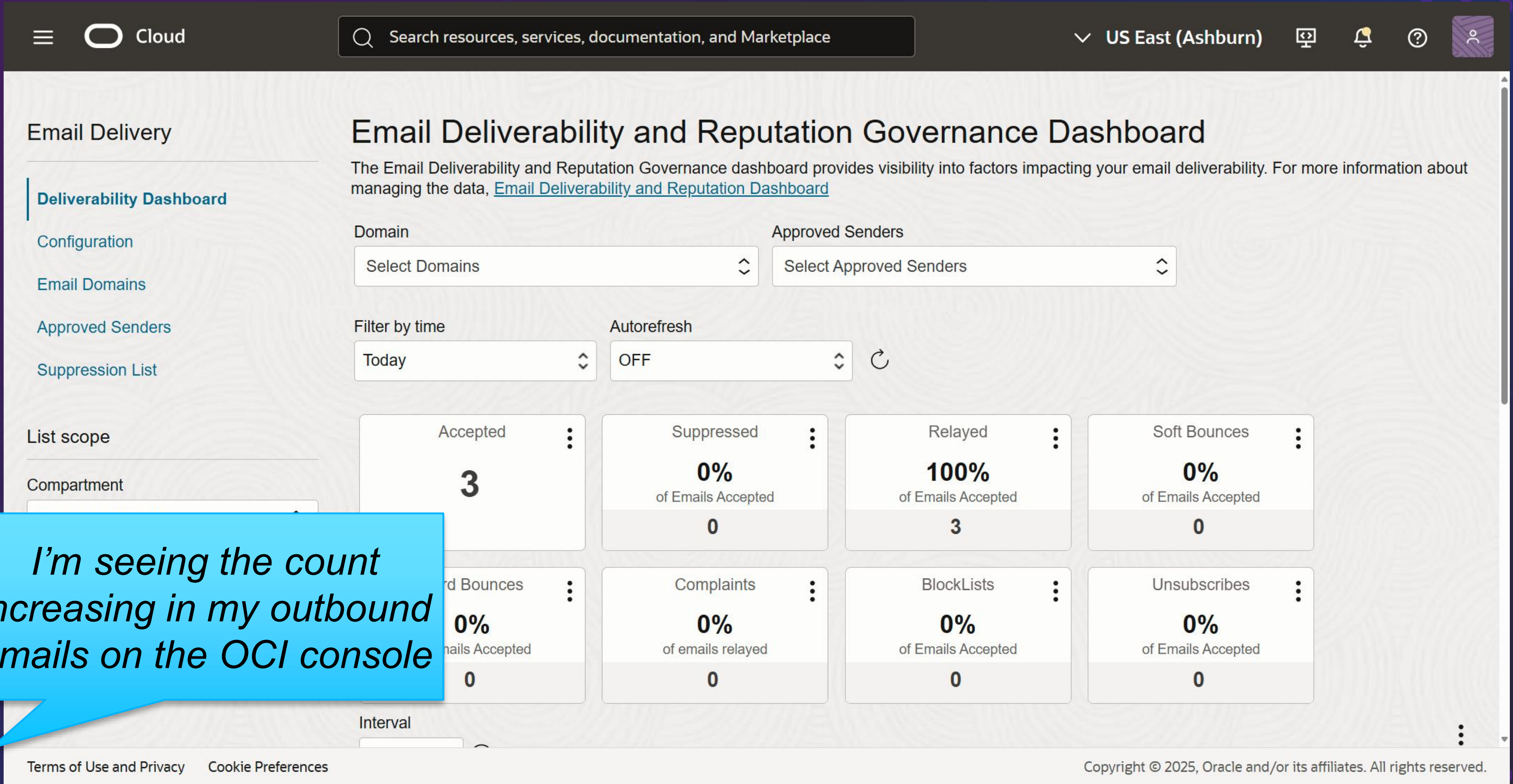
I'll fix the email sending issue by modifying the `/submit` route to actually call the email function.

**Cline wants to edit this file:**

/app.py

Auto-approve: ☐ Enabled ☐ Read ☐ Edit ✓ Safe Comr

*And it's updating my code automatically*



Email Delivery

**Deliverability Dashboard**

Configuration

Email Domains

Approved Senders

Suppression List

List scope

Compartment

# Email Deliverability and Reputation Governance Dashboard

The Email Deliverability and Reputation Governance dashboard provides visibility into factors impacting your email deliverability. For more information about managing the data, Email Deliverability and Reputation Dashboard

Domain: Select Domains

Approved Senders: Select Approved Senders

Filter by time: Today

Autorefresh: OFF

| Accepted | Suppressed | Relayed | Soft Bounces |
|---|---|---|---|
| 3 | 0%<br>of Emails Accepted<br>0 | 100%<br>of Emails Accepted<br>3 | 0%<br>of Emails Accepted<br>0 |

| …d Bounces | Complaints | BlockLists | Unsubscribes |
|---|---|---|---|
| 0%<br>…ails Accepted<br>0 | 0%<br>of emails relayed<br>0 | 0%<br>of Emails Accepted<br>0 | 0%<br>of Emails Accepted<br>0 |

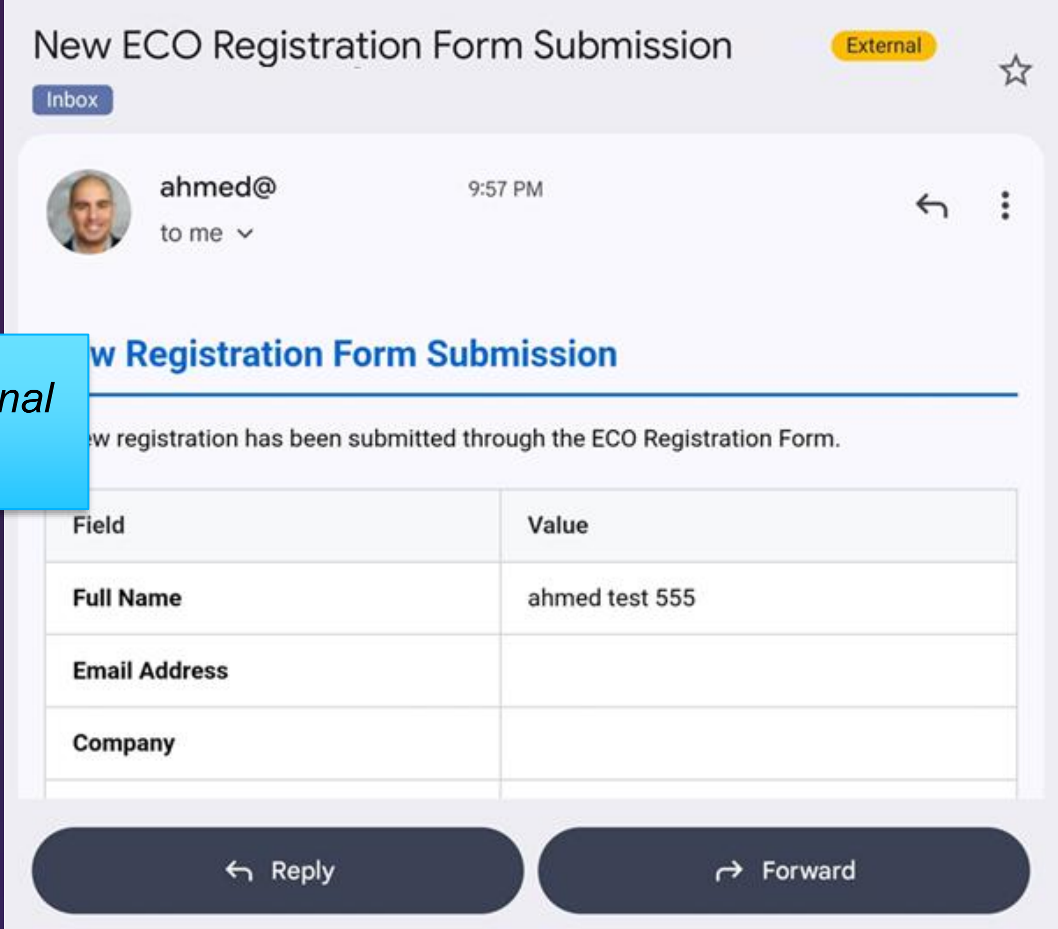
Interval

Terms of Use and Privacy   Cookie Preferences

Copyright © 2025, Oracle and/or its affiliates. All rights reserved.

*I'm seeing the count increasing in my outbound emails on the OCI console*

*Email looks professional and great!*



New ECO Registration Form Submission

ahmed@

to me

9:57 PM

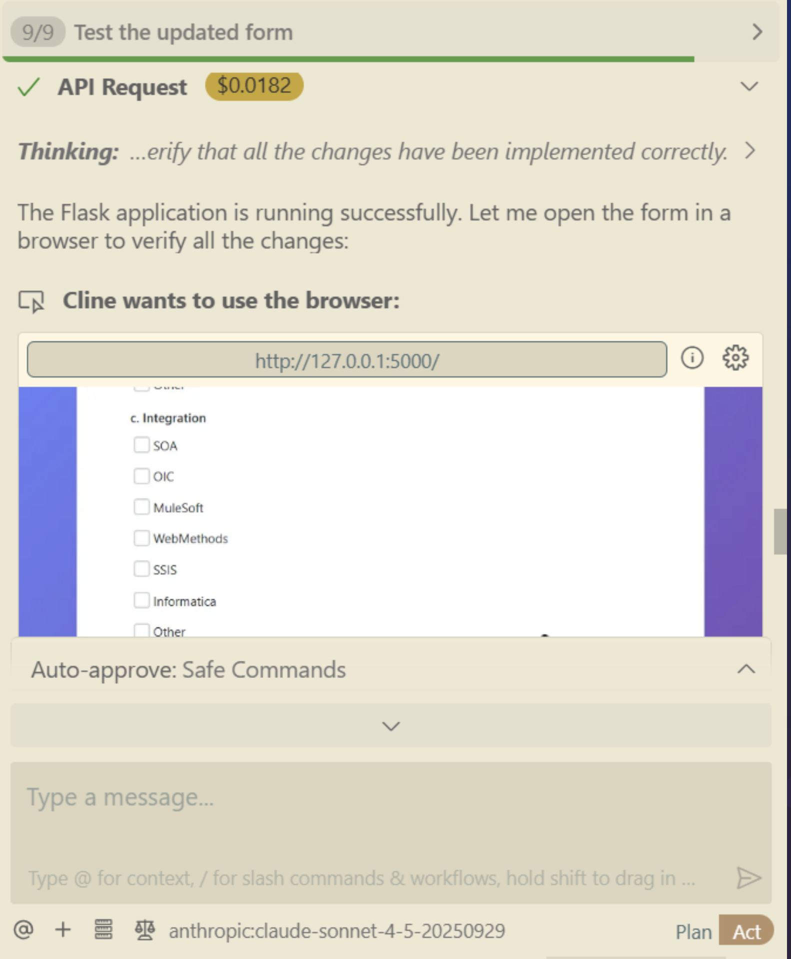
**w Registration Form Submission**

w registration has been submitted through the ECO Registration Form.

| Field | Value |
|---|---|
| **Full Name** | ahmed test 555 |
| **Email Address** | |
| **Company** | |

*Total cost was $1.5116*

*I forgot to mention that Cline can also test the web app locally*



9/9 Test the updated form

✓ **API Request** $0.0182

***Thinking:*** *...erify that all the changes have been implemented correctly.*

The Flask application is running successfully. Let me open the form in a browser to verify all the changes:

**Cline wants to use the browser:**

http://127.0.0.1:5000/

c. Integration

☐ SOA

☐ OIC

☐ MuleSoft

☐ WebMethods

☐ SSIS

☐ Informatica

☐ Other

Auto-approve: Safe Commands

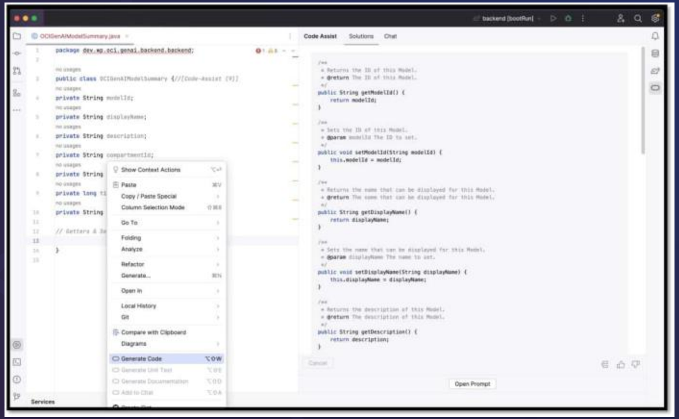
Type a message...

Type @ for context, / for slash commands & workflows, hold shift to drag in ...

# 3. FINAL THOUGHTS

# ORACLE CODE ASSIST

- Announced May 2024 (in beta)
- AI-code companion
- Leverages LLMs running OCI
- Optimized for Java, Python, C#, PL/SQL, etc.



```
package dev.mp.oci.genai.backend.backend;

no usages
public class OCIGenAIModelSummary {//[Code-Assist (9)]
no usages
private String modelId;
no usages
private String displayName;
no usages
private String description;
no usages
private String compartmentId;
no usages
private String
no usages
private long ti
no usages
private String

// Getters & Se

}
```

```
/**
 * Returns the ID of this Model.
 * @return The ID of this Model.
 */
public String getModelId() {
    return modelId;
}

/**
 * Sets the ID of this Model.
 * @param modelId The ID to set.
 */
public void setModelId(String modelId) {
    this.modelId = modelId;
}

/**
 * Returns the name that can be displayed for this Model.
 * @return The name that can be displayed for this Model.
 */
public String getDisplayName() {
    return displayName;
}

/**
 * Sets the name that can be displayed for this Model.
 * @param displayName The name to set.
 */
public void setDisplayName(String displayName) {
    this.displayName = displayName;
}

/**
 * Returns the description of this Model.
 * @return The description of this Model.
 */
public String getDescription() {
    return description;
}
```

**Generate code to help improve productivity**

Build applications faster and improve your productivity—because your time is precious.

**Generate test cases to help increase coverage**

Increase test quality with better test coverage spanning a greater variety of test inputs.

**Annotate code to help improve code readability**

Improve maintainability with AI-generated API documentation or comments.

**Analyze code to help optimize code quality**

Identify bugs and inefficiencies and produce opinionated code based on best practices.

**Tailor suggestions based on your guidelines**

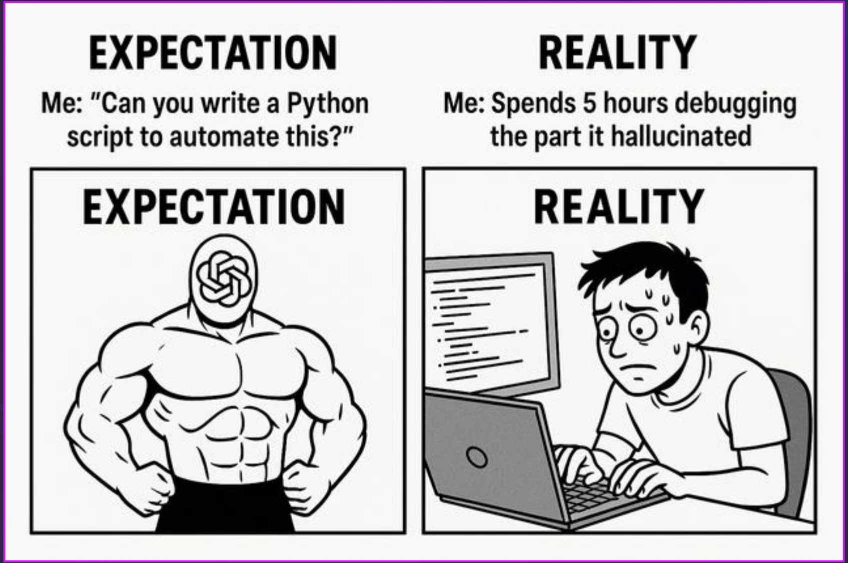
Align your code with your organization's existing codebase, which helps facilitate reviews.

**Provide suggestions for AppDev on OCI**

Maximize utility and boost effectiveness of applications that run on OCI and use OCI services.

https://www.oracle.com/application-development/code-assist/

# FINAL THOUGHTS


EXPECTATION
Me: "Can you write a Python script to automate this?"
REALITY
Me: Spends 5 hours debugging the part it hallucinated
EXPECTATION
REALITY

- Pick a tool and master it
- Enhance your prompting
- Remain critical and curious
- Stay in control
- Prioritize security
- Always refine AI-generated code
- Consider measuring ROI (utilization, impact, cost) (though takes effort)

# BLOG LINKS

- My initial experiences with Tabnine AI code assistant (hopeful)
  https://chronicler.tech/my-initial-experiences-with-tabnine-ai-code-assistant-hopeful/
- Impressive Autonomous AI Code Generation with Cline
  https://chronicler.tech/impressive-autonomous-ai-code-generation-with-cline/
- Lightning speed AI code generation with Diffusion LLM
  https://chronicler.tech/lightning-speed-ai-code-generation-with-diffusion-llm/
- Comparing Cline and GitHub Copilot for Autonomous AI Code Generation
  https://chronicler.tech/comparing-cline-and-github-copilot-for-autonomous-ai-code-generation/
- Creating an API key on Anthropic
  https://chronicler.tech/creating-an-api-key-on-anthropic/
- Configuring Cline on Visual Studio Code
  https://chronicler.tech/configuring-cline-on-visual-studio-code/
- What the AI coding assistants get right, and where they go wrong
  https://www.infoworld.com/article/4003742/what-the-ai-coding-assistants-get-right-and-where-they-go-wrong.html

# THANKS!