# Set Up for Developers

1. Ask Venki to invite you as a collaborator on the GitHub repository. Accept the invite.
2. Install VS Code
3. Install latest Python 3.13+
4. Install the Basic Git software
5. Inside VS code install extensions: GitHub Copilot, GitHub Repositories, GitHub Pull Requests, see below:
6. A screenshot of a computer

   AI-generated content may be incorrect.
7. Inside VS code, Shift+Control+P (pallete), and search for GitHub: Clone
8. Clone the ‘LUMA\_Timesheet\_Automation’ repository. You should see the below:

A screenshot of a computer

AI-generated content may be incorrect.

1. Click on main.py and you should see the code as below:

A screen shot of a computer program

AI-generated content may be incorrect.

1. Open a new Terminal Window:

A screen shot of a computer

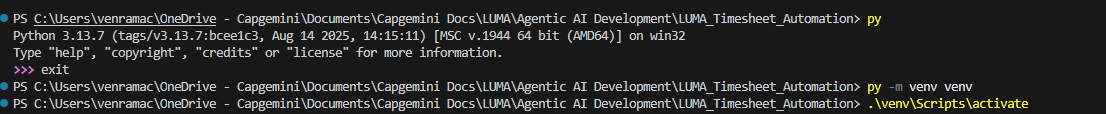
AI-generated content may be incorrect.

1. Make sure you are in the dir where you cloned the repository: With ‘dir’ cmd should see

A screenshot of a computer

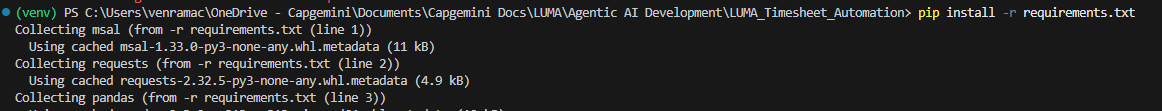
AI-generated content may be incorrect.

1. Create a virtual environment using:



Once the venv is created, you should activate it using the above ‘activate’ cmd

1. Your prompt in the terminal should have, the green (venv) text in front as seen here:

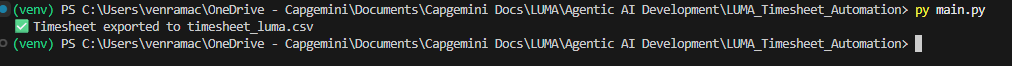


1. Install all the packages needed for the tool using the above pip command
2. The final packages when installed should show the below screen

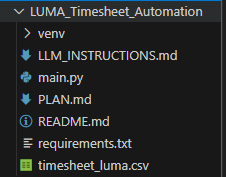
A black screen with white text

AI-generated content may be incorrect.

1. Run the main.py command using:



1. You should see the csv file created in the file system as below:



1. Open it up and you should see:

A screenshot of a computer

AI-generated content may be incorrect.

1. Make sure you have some Outlook meeting entries titled ‘LUMA Timesheet Entry’ and some text in the body.
2. Make some changes to the main.py code and save the file ‘Control-S’:
3. Do a ‘git status’ as below:

A computer screen with white text

AI-generated content may be incorrect.

1. It shows that there were changes in the repo dir. Main.py had changes and a new file ‘timesheet\_luma.csv’ got added (output of our py run)
2. Just check in the changes to the py code:

A black screen with white text

AI-generated content may be incorrect.

1. You added (add) using the git add command. You committed and provided a message using the ‘git commit’ cmd and then finally pushed the changes to the remote repo using ‘git push’.
2. Changes you pushed are visible in the GitHub web site account folder as below:

A screenshot of a computer

AI-generated content may be incorrect.

1. Congratulations, your laptop is now set to develop python applications using VS Code.

# Mount the SharePoint Folder on your Windows Explorer

We need to mount the SharePoint directory for our project LUMA since Munish/ben create the common xls of all meetings in a shared folder there. We need to access that and use its entries in our own timesheet xls

Start with going to the following folder in the IE browser

A screenshot of a computer

AI-generated content may be incorrect.

Click on the three horizontal dots besides ‘Edit in grid view’

A screenshot of a computer

AI-generated content may be incorrect.

Click on the ‘Sync’ menu as shown above, Let it do its thing and you should see in your MS Explorer

A screenshot of a computer

AI-generated content may be incorrect.

Expand the one for ‘Task and Time Keeping for LUMA’ by clicking on the arrow to its left, you should see all our timesheets as below:

A screenshot of a computer

AI-generated content may be incorrect.

The folder where the common meeting list is saved in a xls is at:

A screenshot of a computer

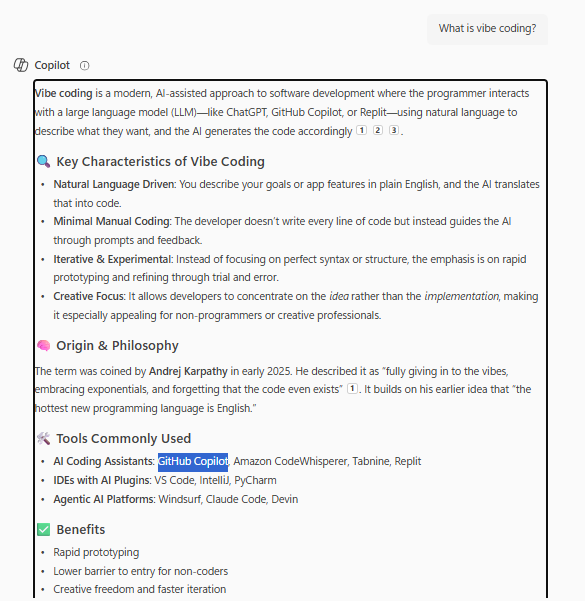
AI-generated content may be incorrect.

This is the directory and file name we will use in the python code so that we don’t have to copy or download this to our local drive. Just mount the Sharepoint and use the code.

# Vibe coding Start

A screenshot of a chat

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

/asyncpr

├── PLAN.md # Must-haves, nice-to-haves, out-of-scope

├── LLM\_INSTRUCTIONS.md # Prompts and guidance for AI agents

├── /src

│ ├── /backend # FastAPI or Spring Boot

│ ├── /frontend # React or Blazor

│ └── /shared # Common utilities

├── /evals

│ ├── image-processing-eval.md

│ ├── business-name-detection-eval.md

│ └── feedback-generation-eval.md

├── /tests # User journey validations

├── /docs # API specs, schemas, business rules

├── /scripts # Utility scripts

├── /mcp-servers # Local server configs for Claude

└── README.md

**Workflow Steps**

**1. Define the Vision**

* Write a clear **PLAN.md** with goals and constraints.
* Include **user stories** and **acceptance criteria**.

**2. Start with Evaluations (Evals)**

* Define what “good” output looks like before coding.
* Examples:
  + Receipt image → Extracted business name
  + Business name → Valid business email
  + Customer feedback → Structured JSON

**3. Prompt GitHub Copilot**

* Use natural language like:

“Create a FastAPI endpoint that accepts a receipt image and returns the business name.”

* Let Copilot generate the code, then refine iteratively.

**4. Use Claude for Planning**

* Ask Claude to:
  + Review architecture
  + Suggest refactors
  + Validate business logic

**5. Build Section-by-Section**

* Implement one feature at a time.
* Validate against Evals.
* Commit only when it passes.

**6. Containerize & Orchestrate**

* Use Docker to containerize backend and frontend.
* Optionally orchestrate with Docker Compose or Kubernetes.

**🧠 Mindset Tips**

* Treat AI tools like **interns**: powerful but needing guidance.
* Reset early if AI output drifts.
* Keep Git branches clean and use GitHub Actions for validation.
* Refactor relentlessly for clarity and modularity.

A screenshot of a chat

AI-generated content may be incorrect.

I saved the Plan.md and LLM\_instructions.md in my local file system for later use

A screenshot of a computer

AI-generated content may be incorrect.

**🚀 Next Step: Set Up Your Environment**

Would you like help setting up:

1. A **Python project** in VS Code with GitHub Copilot?
2. **Microsoft Graph API access** (app registration, permissions)?
3. A **starter script** to fetch and filter meetings?

Let me know which you'd like to tackle first, and I’ll guide you through it!

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Follow all the instructions, they take up a lot of time if you are already familiar with coding in python using VS Code.

1. Installed VS code

2. Installed latest python in my Cap laptop

A yellow object with text

AI-generated content may be incorrect.

3. Installing GitHub desktop gave me a lot of problems. Not sure what I did wrong.

A screenshot of a computer error

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Finally installing only Git not Github desktop

 A computer screen shot of a computer screen

AI-generated content may be incorrect.

Why did I install this? Probably need to keep it running in the background

A screen shot of a computer

AI-generated content may be incorrect.

Install just Git from: <https://git-scm.com/download/win> (accepted all defaults)

A screenshot of a computer error

AI-generated content may be incorrect.

Exit and restart VS Code..error disappears

A black screen with white text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

Use Shift+Cntrl+P, get the pallet, Get the GitHub Copilot: Sign In

And sign in to your GitHB account. Now test the auto complete by creating a python code and see if it completes the code:

A screenshot of a computer program

AI-generated content may be incorrect.

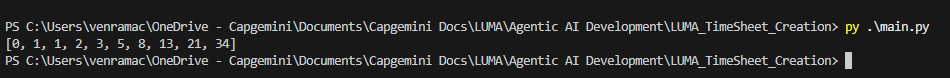
The greyed out text the code suggested by AI, Hit TAB to accept it.

All the suggested code gets accepted and it suggest the next ‘print line’

A screen shot of a computer program

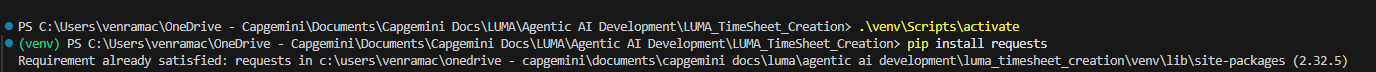
AI-generated content may be incorrect.

Hit TAB, Control+S to save the file and run it in the terminal.



Congrats, you created the first py code with GitHub Copilot suggesting all the code and tested it and it ran correctly.

Check if the path is set and the venv is always activated before you start coding



(venv) in green should be in front of the prompt.

Next Steps:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

A computer screen with a blue suitcase on it

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Mount SharePoint web site as a network drive in your laptop:**

A screenshot of a computer

AI-generated content may be incorrect.

# Integrating with Sharepoint

1. Check if One Drive is loaded and working.
2. Click on the “…” besides the high level dir ‘LUMA – LUMA AMI HES”
3. Click on Sync…You should see this on your File Explorer
4. A screenshot of a computer

   AI-generated content may be incorrect.
5. The Sharpoint site is mounted on your File Explorer and one can access the files and dir as if it is on your local drive.

# Debugging Outlook

Upgraded to the latest Outlook and the version #s is below….

A screenshot of a computer

AI-generated content may be incorrect.

**Recommended Alternative: Microsoft Graph API**

Since COM automation is deprecated in the New Outlook, Microsoft recommends using the **Microsoft Graph API**, which is:

* **Cloud-based** and works with both desktop and web versions of Outlook.
* **Secure and modern**, using OAuth2 for authentication.
* Capable of accessing **emails, calendar events, contacts**, and more.

**Example: Reading Emails with Microsoft Graph API**

You can use the official Python SDK:

You'll need to:

* Register an app in **Azure Active Directory**.
* Get **client ID**, and set up permissions (like Mail.Read, Calendars.Read).
* Use DeviceCodeCredential or AuthorizationCodeCredential for authentication

2

.

**🛠️ Other Options**

If you're in a corporate environment and still have access to the **classic Outlook desktop app**, you can:

* **Revert to the older version** (if allowed by your IT admin).
* Use win32com as before.

But long-term, Microsoft is phasing out the classic client, so Graph API is the future-proof solution.