Vibe Coding Session

In-Depth Analysis: Hands-On Vibe Coding Session

Instructors: Anupam Purwar (Lead Facilitator)
Core Philosophy: Democratizing development by using AI assistants to translate natural language intent into functional code, drastically reducing the barrier to entry for building software.

1. Technical Stack & Ecosystem Context

The session wasn't just about using a single tool, but about integrating a modern, AI-powered development ecosystem:

- **Primary AI Tool: GitHub Copilot**
  - **Role:** An AI pair programmer, deeply integrated into the IDE.
  - **How it Works:** It goes beyond simple code completion. It processes **natural language instructions** (e.g., "create a function to fetch user data from an API") and generates entire code blocks, functions, or even files. This is the core of "vibe coding" – describing the *what*, not the *how*.
  - **Corporate Relevance:** its widespread use in corporate environments, signaling that this is a practical, industry-relevant skill, not just a novelty.
- **The IDE: Visual Studio Code (VS Code)**
  - **Why it's Important:** VS Code is the central hub. It's not just a text editor; it's an **Integrated Development Environment (IDE)**. This means it has built-in utilities for:
    1. File and folder management (creating the project structure).
    2. Running and executing code.
    3. Integrating terminals, debuggers, and extensions (like Copilot).
    4. The instructors emphasized that Copilot's magic happens *within* this powerful context-aware environment.
- **The LLM Power: Gemini API & Google Cloud**
  - **Strategic Mention:** While Copilot assists with *writing* code, the **Gemini API** was introduced as the way to *power* the application's intelligence.
  - **The Architecture:** The implied stack is:
    1. **VS Code + Copilot:** To build the application structure and logic.
    2. **Gemini API:** To provide the core AI/LLM brain for the agent (e.g., processing user queries, generating responses).

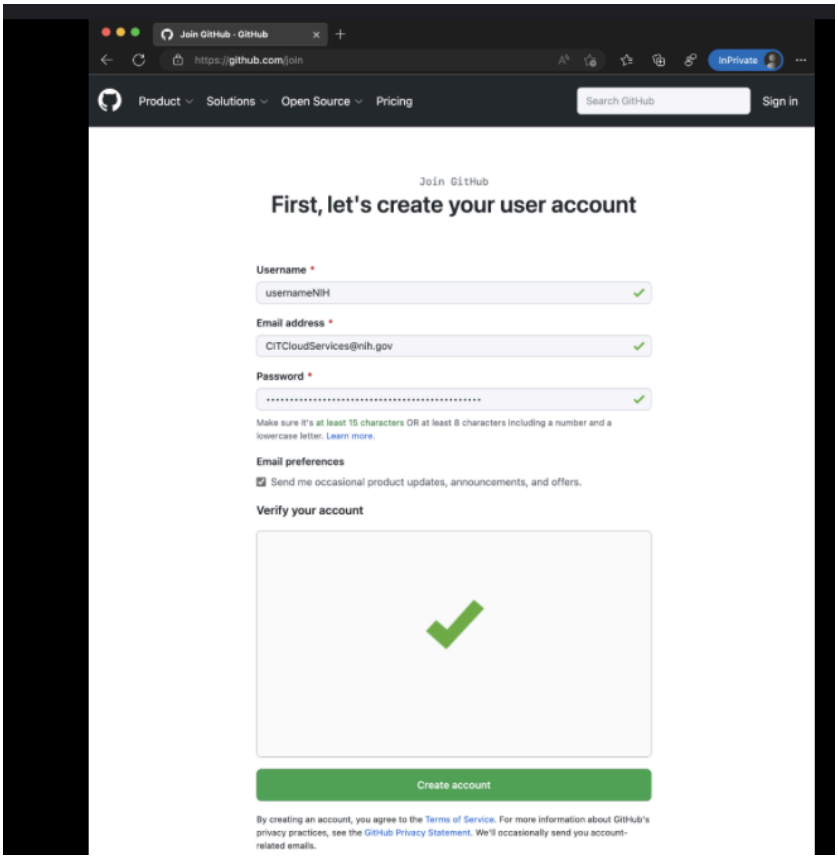
3. **Google Cloud:** The suggested platform for hosting, scaling, and serving the application reliably. This introduces participants to cloud concepts.

2. Deconstructing the "Hands-On" Instructions

The instructions,:

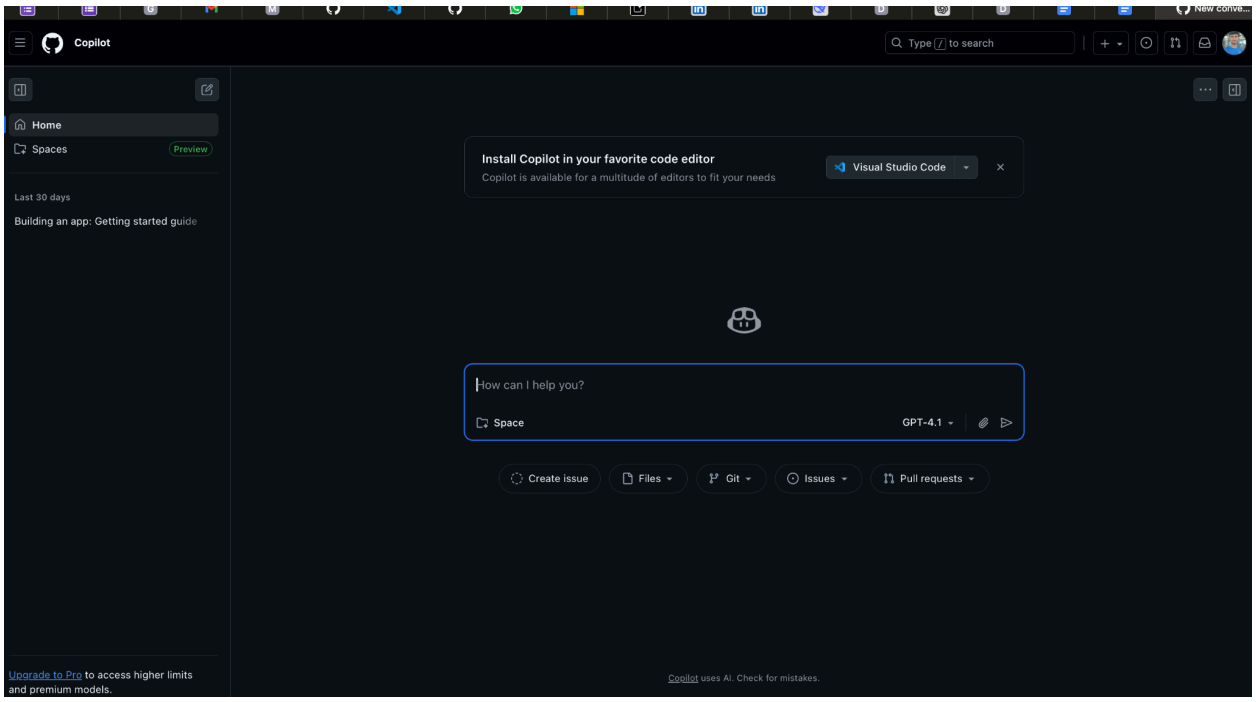
- **Prerequisites (To be done BEFORE coding starts):**

1. **Create a GitHub Account:** This is fundamental for version control, collaborating on code, and often for authenticating services like Copilot.



2.
3. Github sign up - github.com/signup

Vibe Coding Session

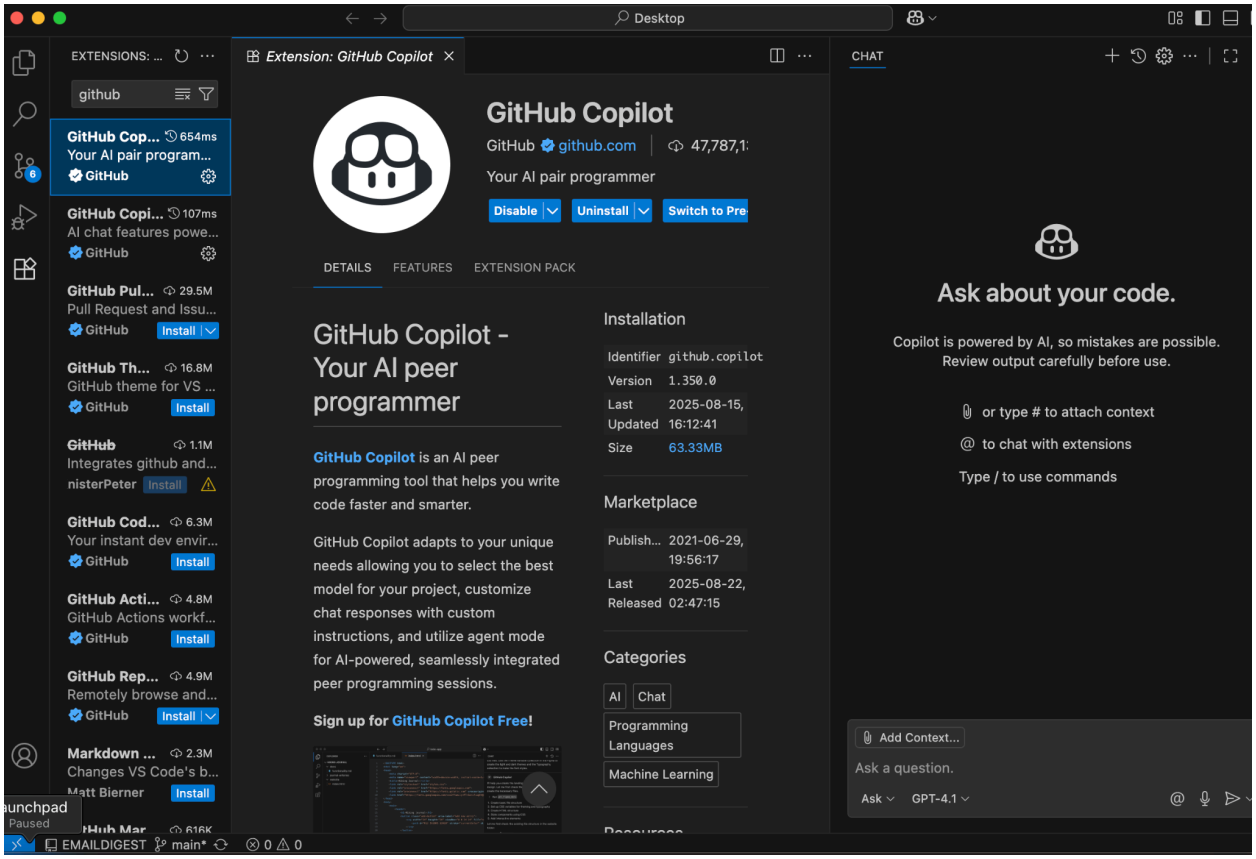


Github copilot sign-up setup - https://github.com/copilot

Download VS Code: This is the non-negotiable workspace. The session assumed a "follow-along" format, making this step essential for participant engagement.

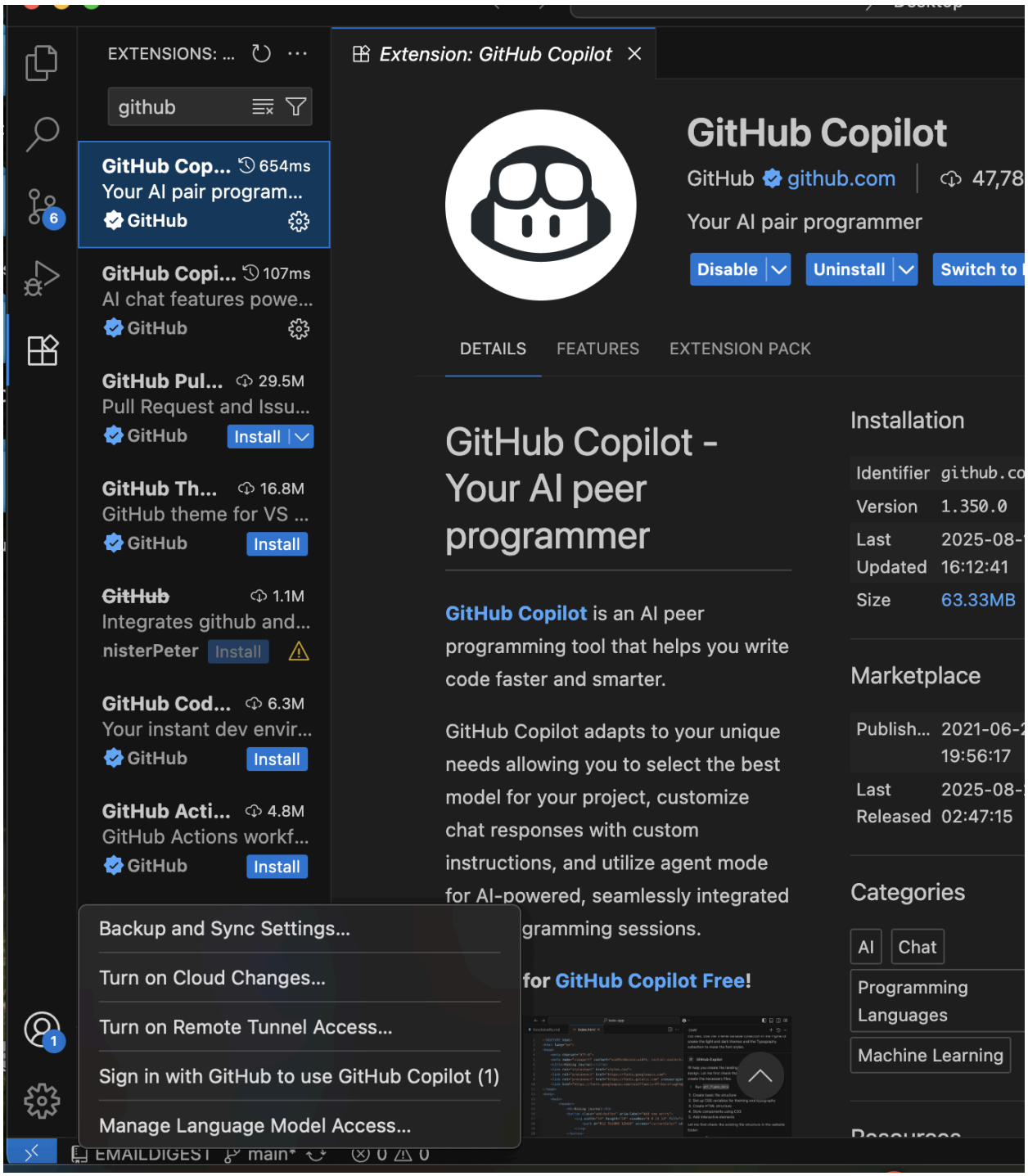
https://code.visualstudio.com/

Vibe Coding Session



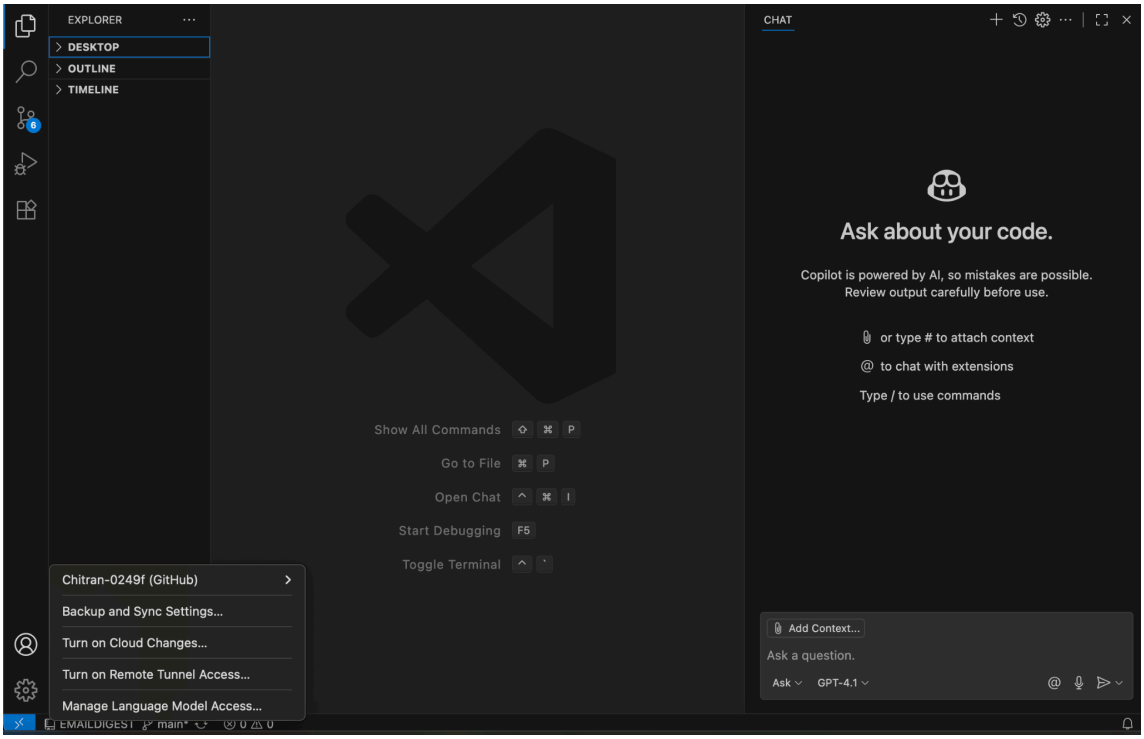
Enable Githb Copilot extension

Vibe Coding Session



Sign up with github account on copilot extension

Vibe Coding Session



Steps to be followed after setup:

1. After signing up, verify your account from the bottom-left account icon.
2. On the right-side chat window, select a model (preferably *Claude Sonnet* for coding tasks).
3. In the **Ask** dropdown, select **Agent** mode.
4. Provide the following prompt to build an application:
   "Build a simple AI app that uses Gemini-2.5-Pro to read a PDF and answer user questions based on its content."
5. Paste your Gemini API key into the environment variables, or provide the API key directly in the chat window and instruct the model to include it in the code.
6. Test, debug, iterate, and improve your app!

🎉 Congratulations — you've built your first AI application!

Vibe Coding Session

3. The "Vibe" and Pedagogical Approach

specific teaching methodology:

- **Acknowledgment of Real-World Constraints:** acknowledged that setup (installations) can take 15-20 minutes, which is a common hurdle in live technical sessions.
- **Interactive and Responsive:** The session was designed to be a starting point. The offer for a **follow-up session** was a key takeaway**.** This future session is framed as participant-driven: *"the inputs or the requirements can come from all from participants."*

4. Vision for Future Sessions & Advanced Concepts

The closing discussion laid out a roadmap for more complex projects, moving from a basic agent to a production-ready system:

- **Database Integration:** Moving from a static app to one that persists and recalls data (e.g., user histories, preferences).
- **Analytics (Google Analytics):** Adding the ability to measure user interaction with the AI agent, a critical skill for product development.
- **Deployment and Sharing:** The ultimate goal: taking the locally hosted app and putting it on the internet so that others can use it. This covers a vast area of development ops (DevOps) and cloud deployment.

5. Key Takeaways & Implications

- **The Paradigm Shift:** Coding is evolving from writing syntax to **curating and instructing AI**. The skill is shifting towards precise prompting, architectural thinking, and problem decomposition.
- **Low Barrier, High Ceiling:** This approach allows beginners to quickly build impressive projects, while the advanced topics (databases, deployment, analytics) provide a path for experienced developers to integrate AI into complex systems.
- **Community-Driven Learning:** The recurring Saturday sessions and the request for participant input foster a community of learning rather than a one-way tutorial, making the knowledge transfer more effective and sustainable.
