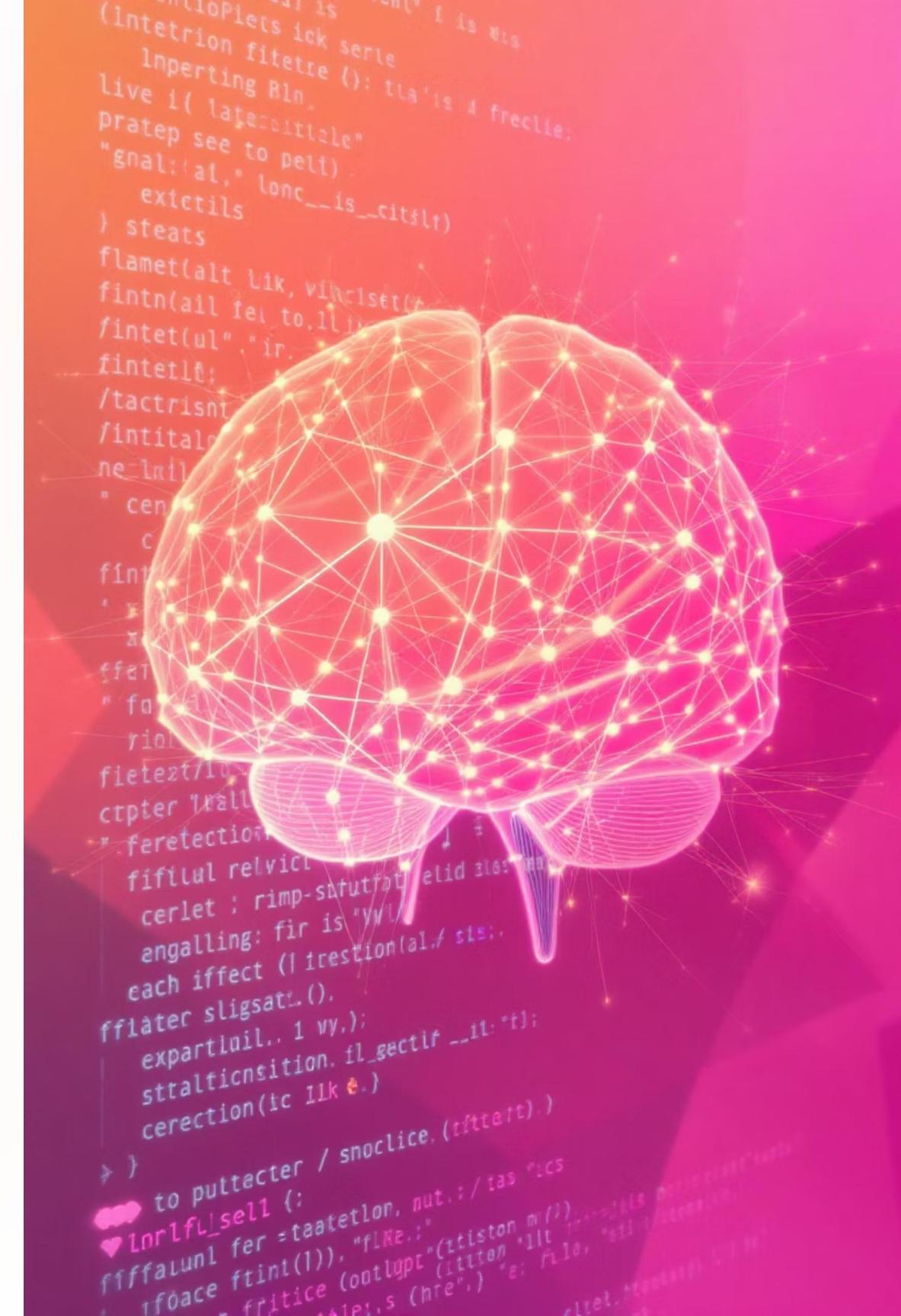
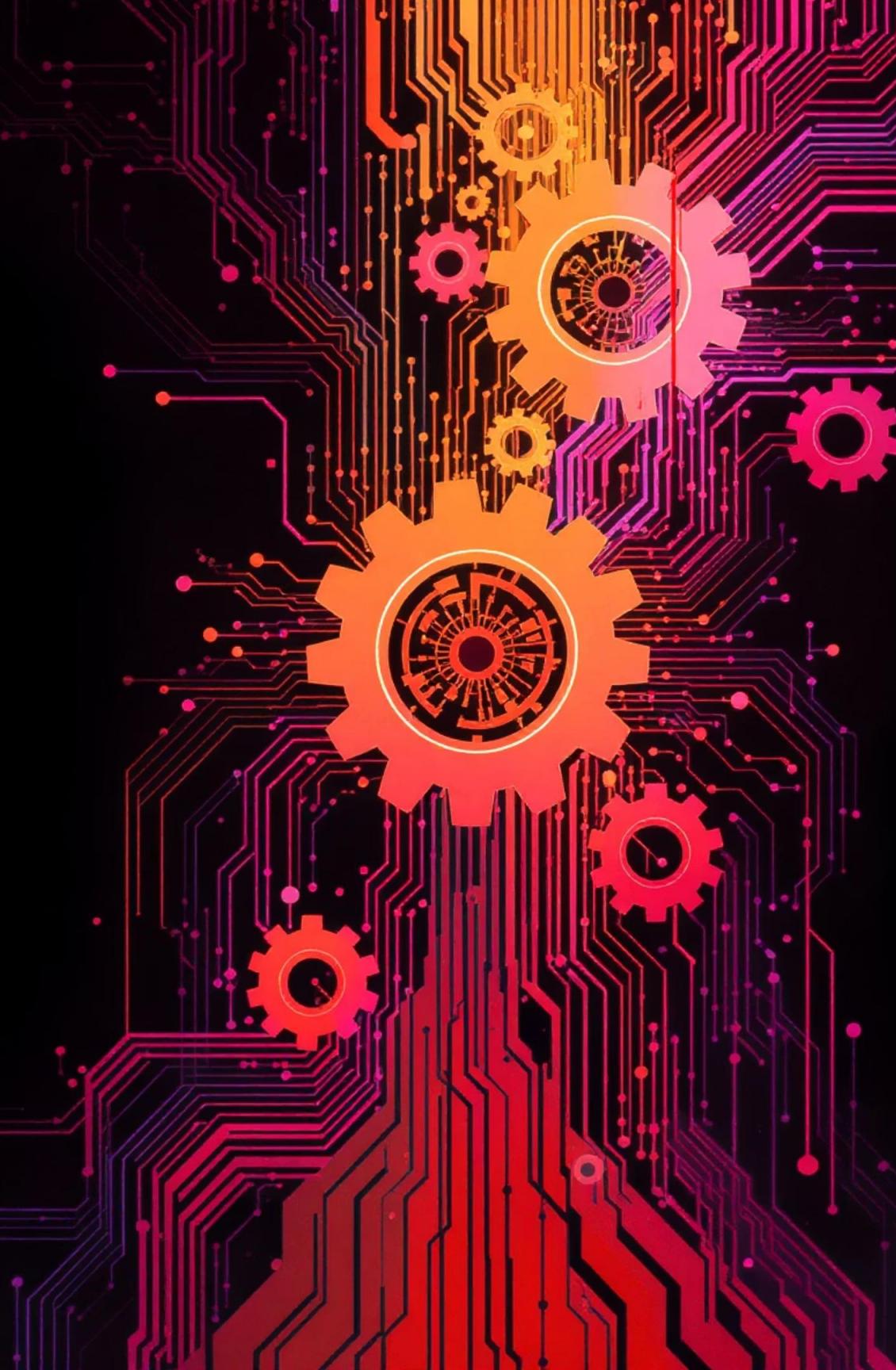


Applying AI in SDLC

A Case Study

using Streamlit





Agenda

1 Introduction

The Evolving Role of AI

2 Our Demo App

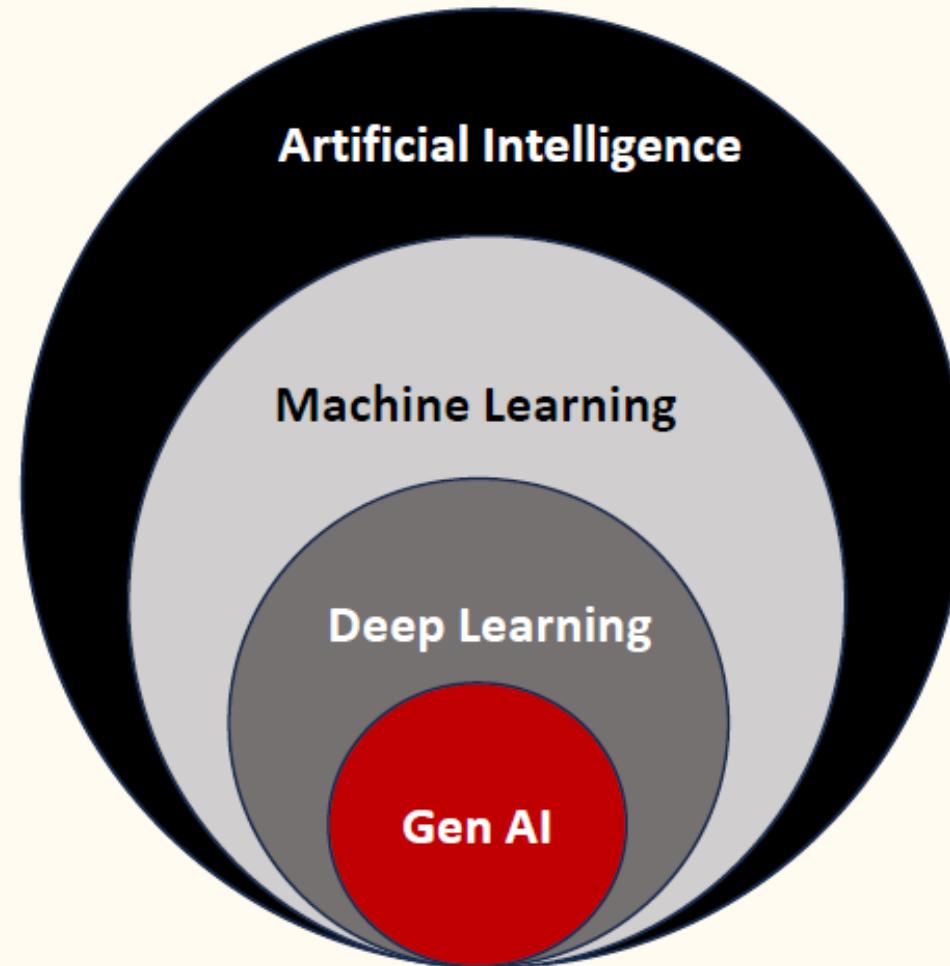
The AI-Assisted Daily Task Manager

3 Phases of SDLC with AI

- Requirements Gathering
- Design & Architecture
- Coding & Testing
- Deployment

4 Q&A

Artificial Intelligence



Artificial Intelligence

Dartmouth conf 1956

Any technique that enables computer to mimic human behavior



Machine Learning

IBM Deep Blue 1997

A subset of AI that enables computer to learn pattern from data



Deep Learning

Revival of interest in 2006

A subset of ML that works more like human brain



Generative AI

ChatGPT in 2021

Create new text, visual or auditory content based on prompt

Language Models

What are Language Models(LMs)

Language models predict the next word given input word sequence



Large Language Models (LLMs)

A large language model (LLM) is a specialized type of artificial intelligence (AI) that has been trained on vast amounts of text to understand existing content and generate original content. The model can produce text, audio, video data.

Characteristics of LLMs

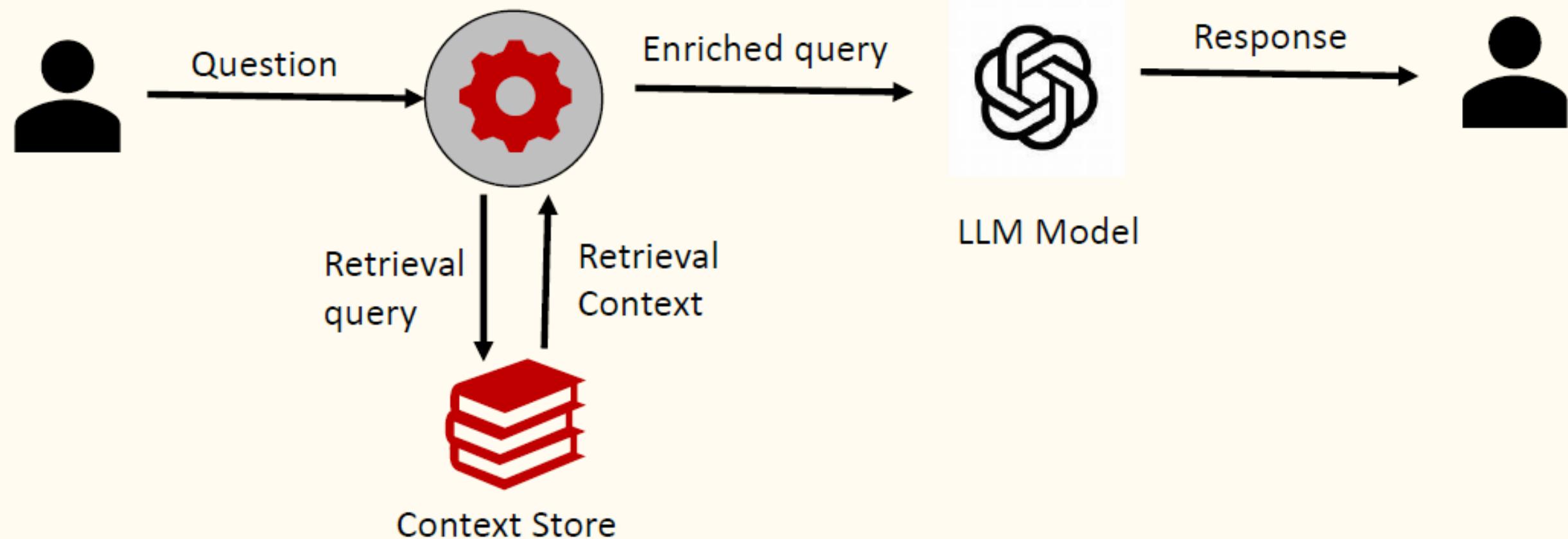
-  Large number of parameters
-  Trained on vast datasets
-  Versatile and general-purpose

Benefits of LLMs

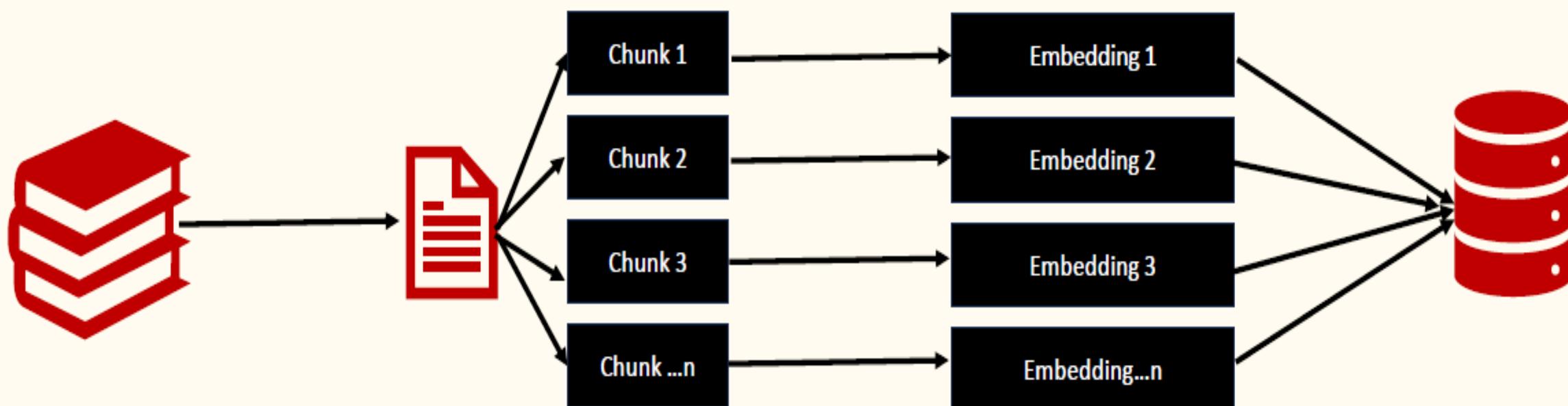
-  Perform diverse tasks
-  Adaptable through fine-tuning
-  Improve continuously with more data

RAG – Retrieval Augmented Generation

RAG is augmenting LLMs with specialized and mutable knowledge base



Vector Databases



1. Load Documents

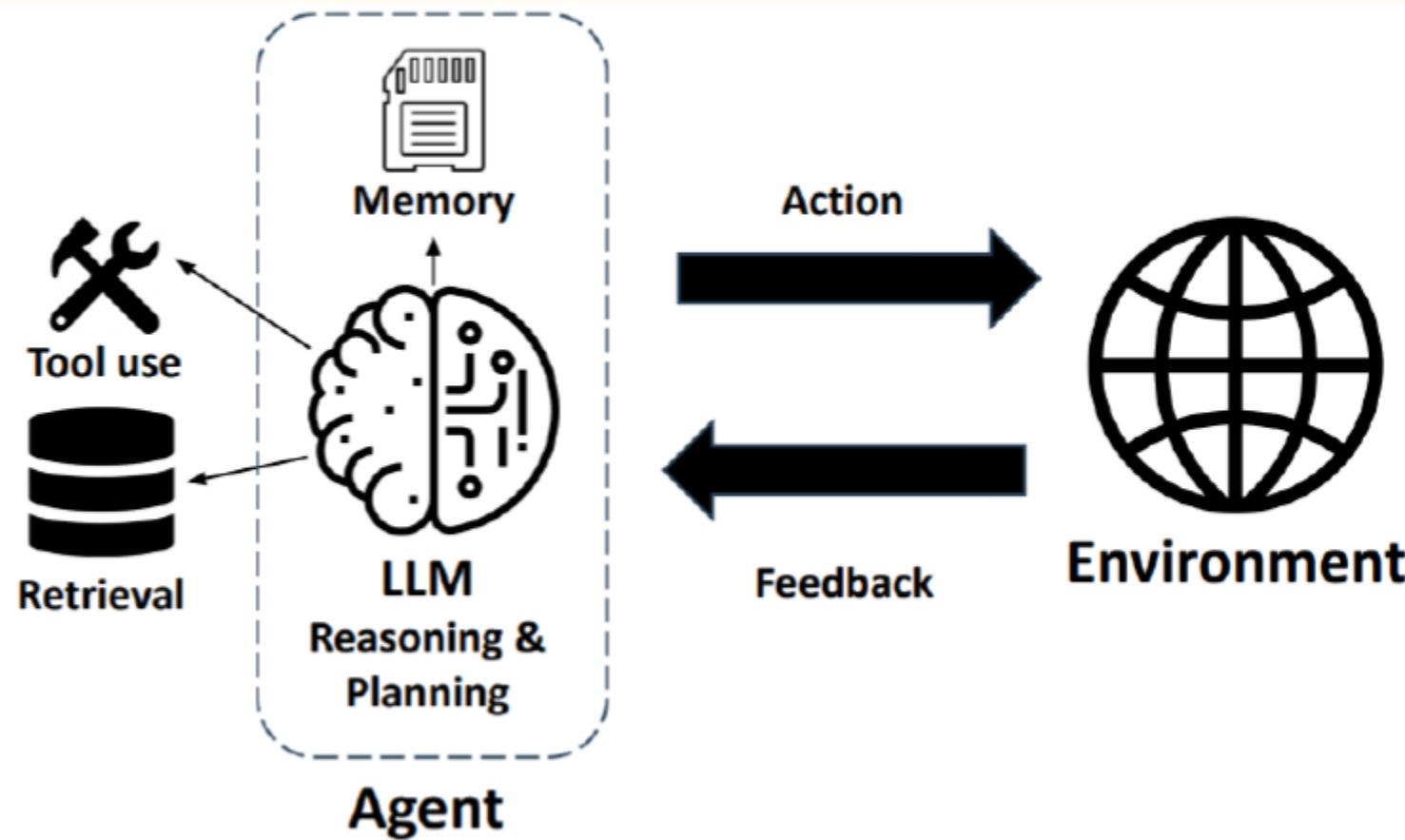
2. Chunk Documents

3. Create Embedding

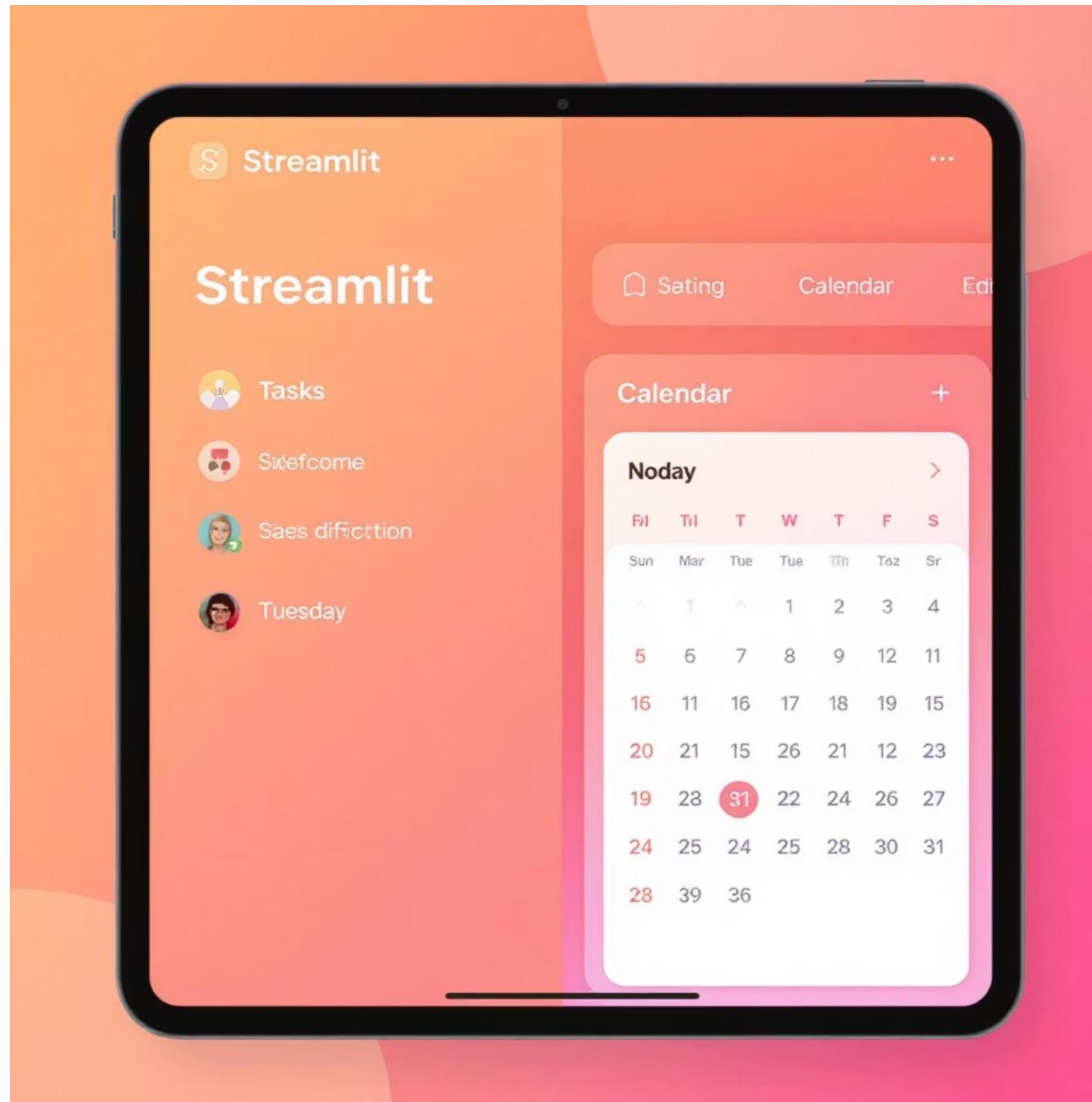
4. Load in a vector DB

Agentic AI

Agentic AI interacts with the environment



The AI-Assisted Daily Task Manager: Practical Demo



Purpose

Illustrating practical AI integration across all the SDLC phases.

Key Features

- Secure User Access & Time Tracking
- Task Management (Add, Edit, Complete)
- Flexible Reporting & Data Export to Excel

Built With

Streamlit (Python) and various other AI tools, VS Code IDE

Live Demo

<https://task-tracker-pro.streamlit.app/>

Phase 1: Requirements Gathering

Definition

Defining what the software needs to do.

Importance

Impacts project success, cost, and user satisfaction.

Traditional Challenges

- Ambiguity in specifications
- Misinterpretation between stakeholders
- Time-consuming brainstorming



AI's Role: ChatGPT for Requirements Elicitation



Idea Generation

Diverse perspectives, virtual sparring partner for concepts.



User Story Definition

Transforming concepts into actionable user stories, ensuring consistency.



Ambiguity Detection

Prompting for missing details, refining statements for precision.

Brainstorming App Features with ChatGPT

Login/Logout Functionality

Features

1. User Authentication
 - Email/username + password login
 - Optional: OAuth (Google, GitHub) for ease
2. Session Management
 - Persistent login during session using `st.session_state`
 - Timeout or manual logout option
3. User Registration
 - Signup form with validation (unique email, strong passwords)
4. Password Handling
 - Hash passwords using libraries like `bcrypt` or `passlib`

Our Approach

Used ChatGPT to kickstart our feature list.

Sample Prompt

I'm building a Streamlit app for daily task management. What are the essential features for login/logout, task tracking, and report generation? Think about user experience and data structure.

Key Insights

- Authentication & Timestamping
- Task Categories & Completion Status
- Excel Export for Reports

Generating User Stories for Specific Features

Our Approach

Refined brainstormed features into actionable user stories.

Sample Prompt

Based on the features we brainstormed for the Streamlit daily task manager, write specific user stories. Focus on the 'As a [user role], I want to [action] so that [benefit]' format.

Example User Story

"As a User, I want to record my login and logout times so I can track my daily work hours automatically."

Benefits

Faster, more comprehensive, and consistently formatted user stories.

Authentication & Account Management

1. User Registration

- As a new user, I want to sign up with my email and password so that I can create a personal task manager account.

2. User Login

- As a returning user, I want to securely log in so that I can access my personal tasks and reports.

3. User Logout

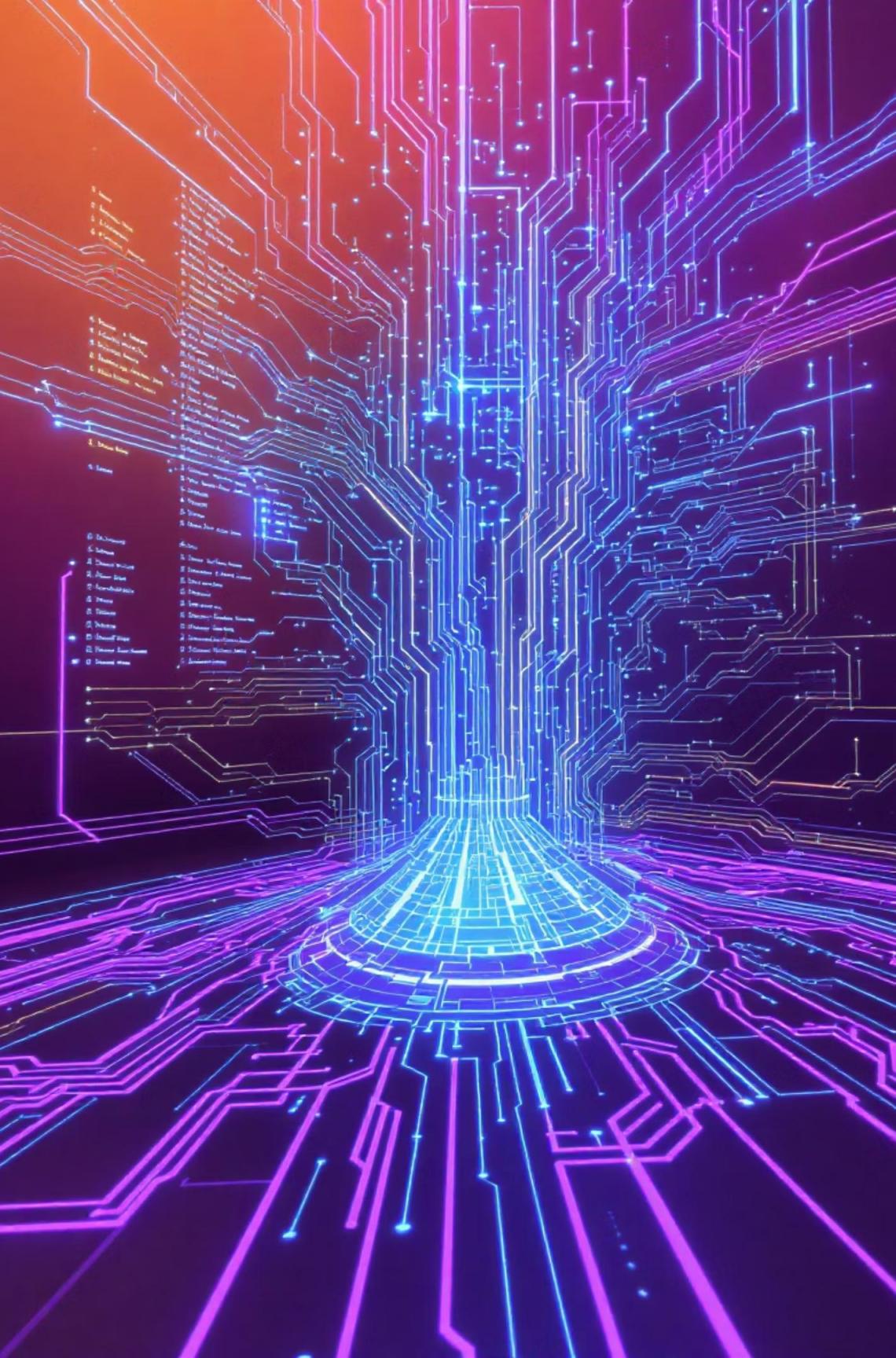
- As a logged-in user, I want to log out of the app so that my session is securely closed.

4. Password Security

- As a user, I want my password to be securely encrypted so that my data remains protected.

AI-Powered Software Design & Architecture

Exploring how AI revolutionizes software design, from conceptualization to dynamic visualization.



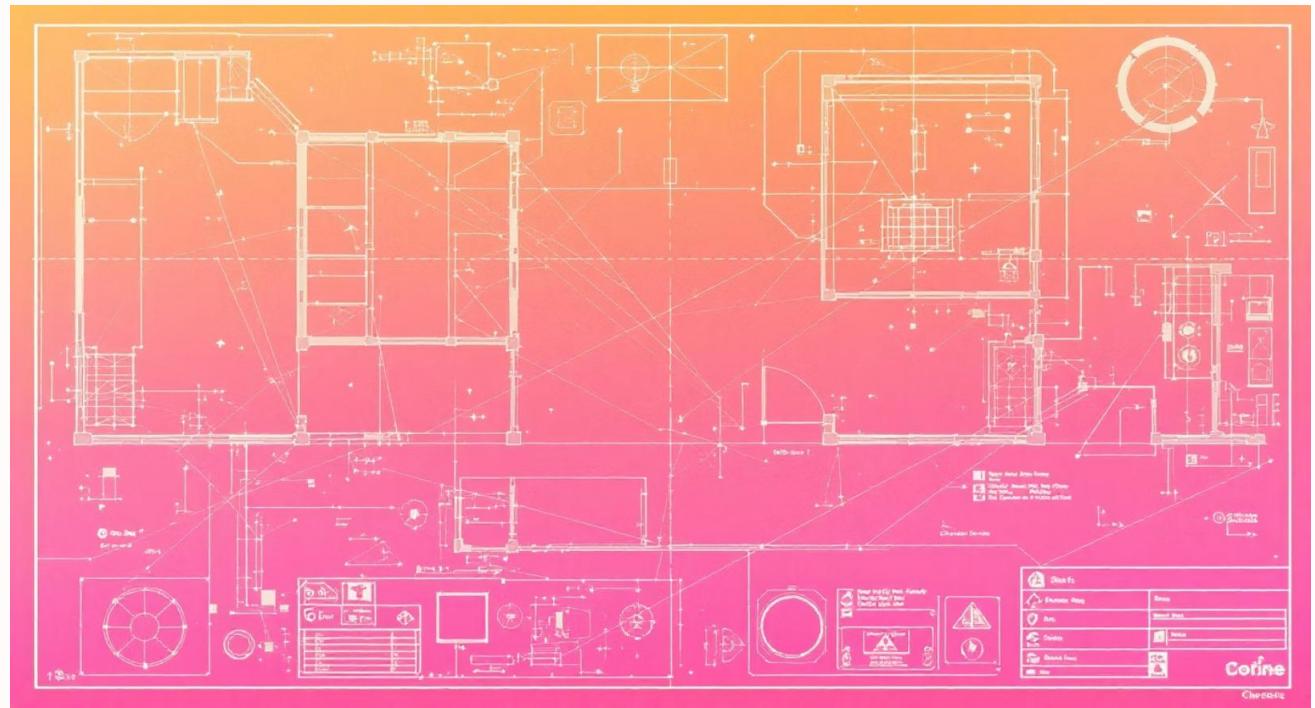
Phase 2: Design & Architecture

Definition

Translating requirements into a technical plan, outlining system components, data structures, and interactions.

Common Artifacts

- Entity-Relationship Diagrams (ERDs)
- UI/UX Flowcharts
- Component Diagrams



Importance

- Ensures shared understanding
- Identifies bottlenecks early
- Guides efficient coding



AI's Role: Dynamic Design with Text-to-Diagram



Fast Prototyping

Quickly visualize ideas without manual drawing tools.



Version Control

Diagrams as text files, easy to track changes in Git.



Enhanced Collaboration

Share text definitions, allowing others to render and understand.

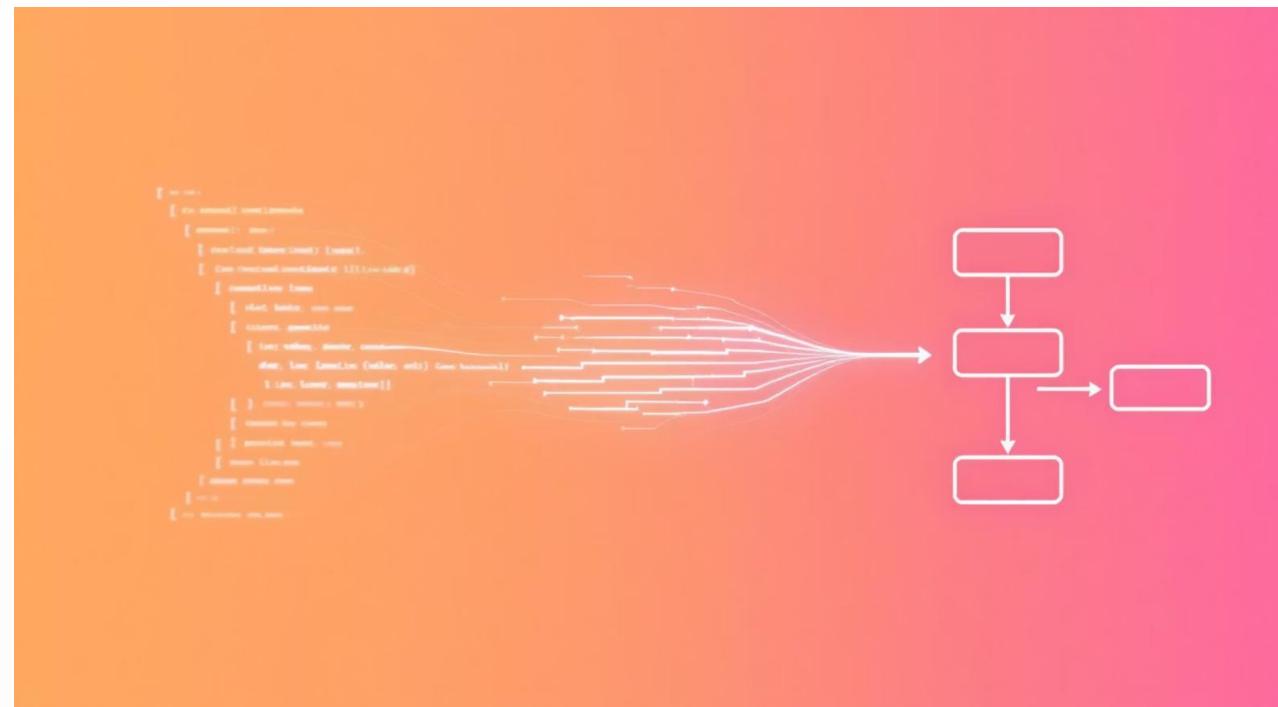


"Living Diagrams"

Easy to update as design evolves, integrated into documentation.

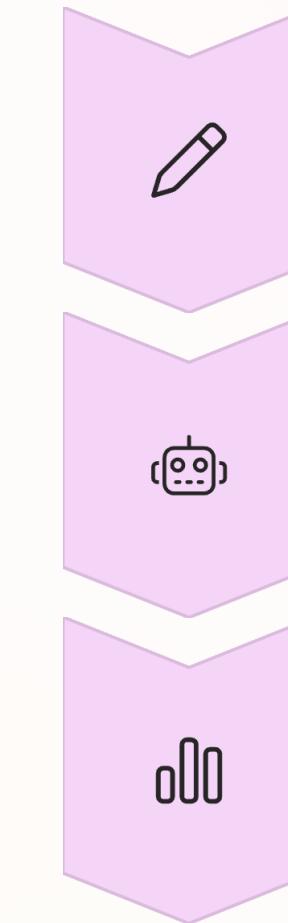
AI assists by generating PlantUML/Mermaid syntax from high-level descriptions.

AI Integration: PlantUML



Leveraging Large Language Models (LLMs) to generate diagram syntax from natural language descriptions.

This transforms static diagrams into dynamic, version-controlled assets.



Describe

High-level design concepts.

AI Generates

PlantUML

Visualize

Dynamic, living diagrams.

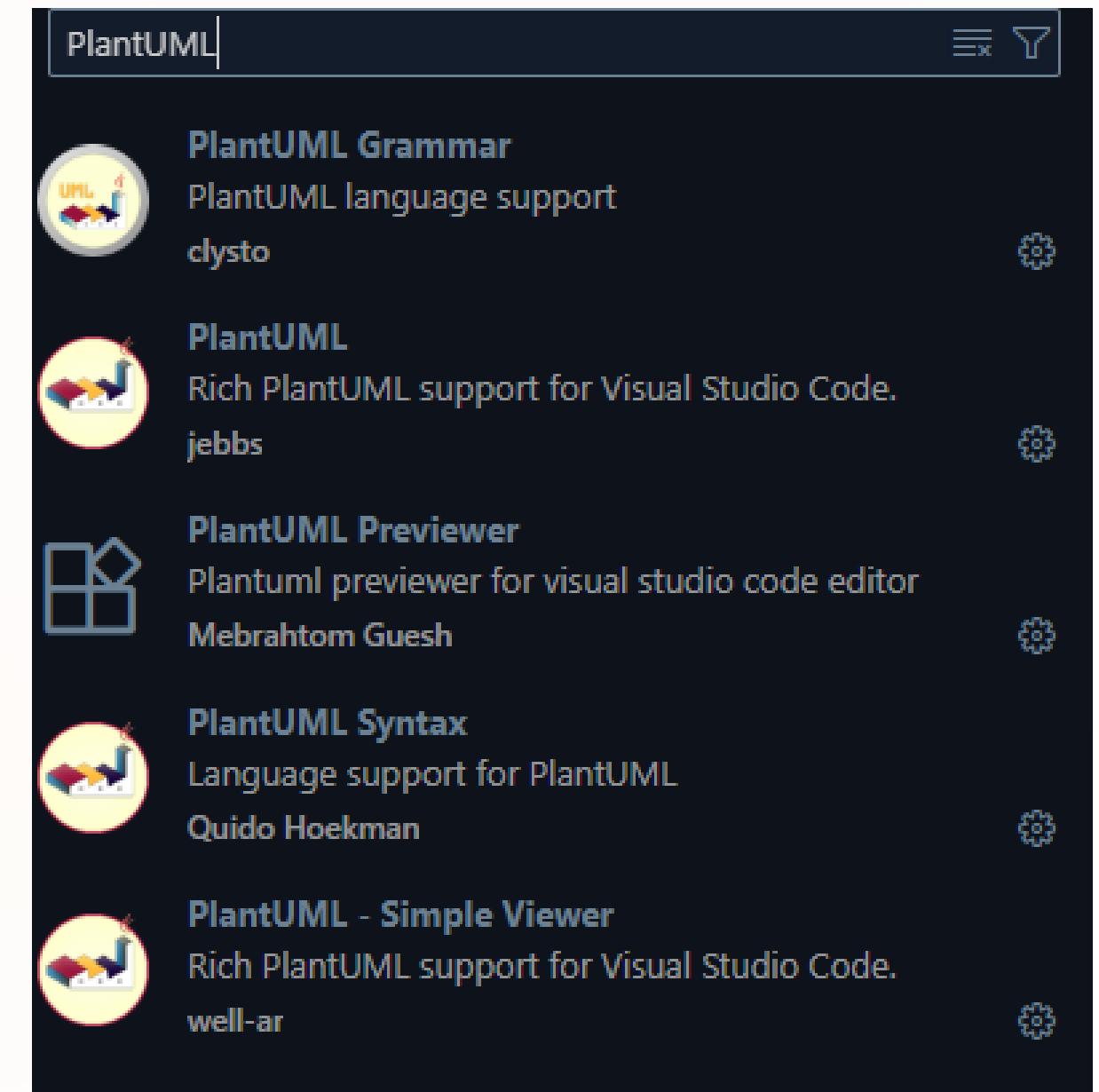
Data Model Design with PlantUML

Our Approach

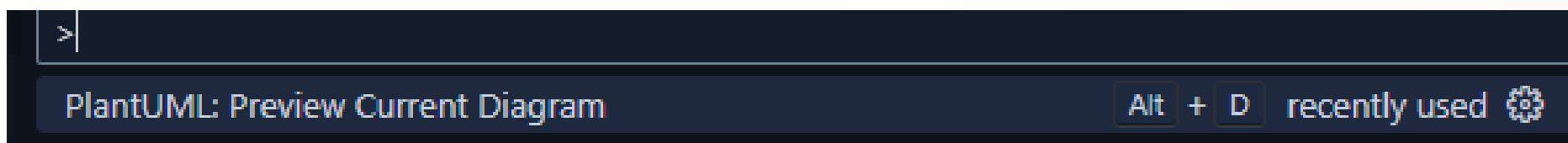
Defined the database schema for our app's users, daily logs, and tasks.

We leveraged **ChatGPT** to automatically generate **PlantUML syntax** based on the desired entities and relationships.

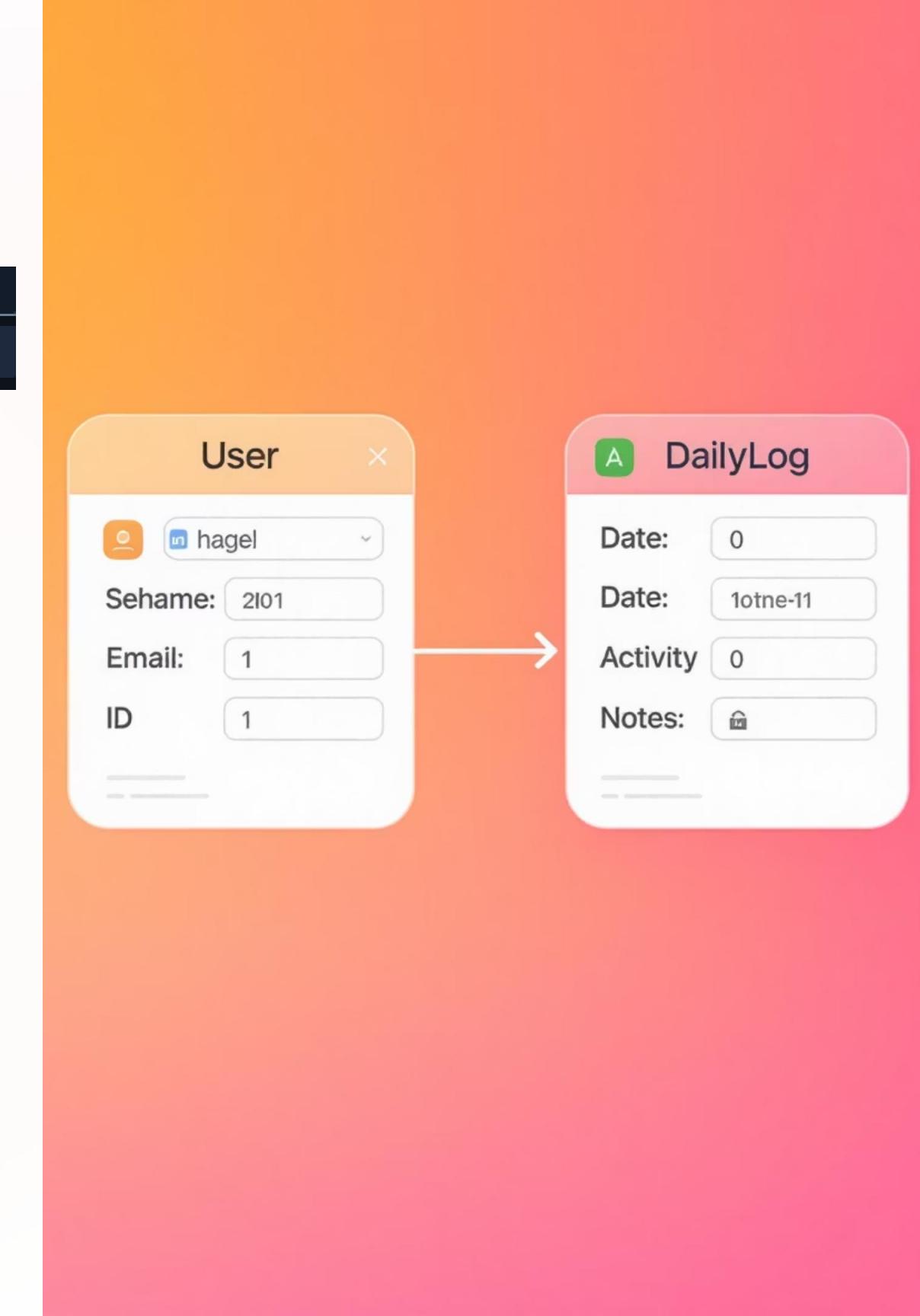
Initially, we provided our requirements, and ChatGPT produced the corresponding **PlantUML code** to create the necessary diagrams.

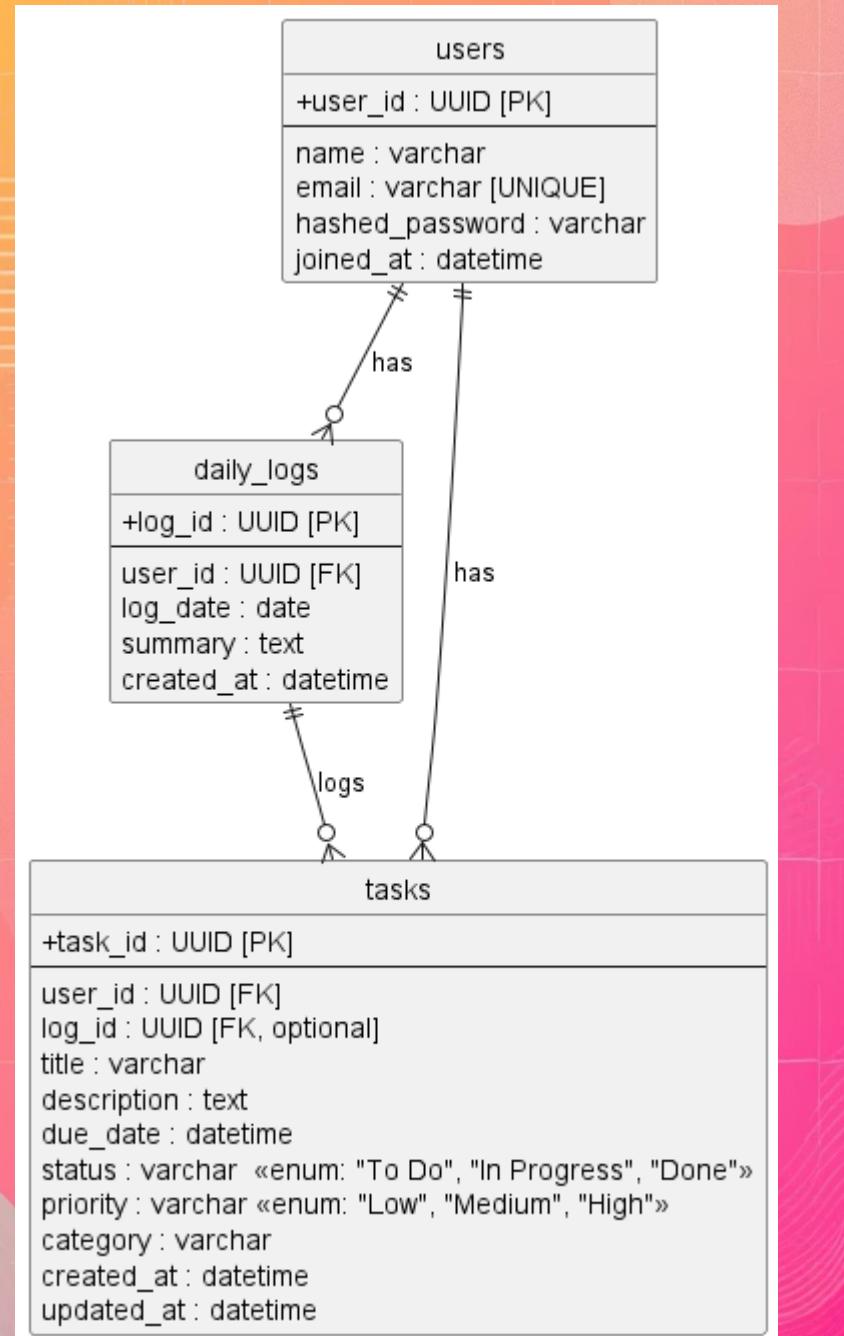


Here, We can Preview the Diagram By clicking Preferences or Ctrl + Shift + P And enter
PlantUML: Preview Current Diagram



Now, we can Export the diagram by click same as above and type “PlantUML: Export Current Diagram”





Sample PlantUML Code Snippet

PlantUML Code for Database Schema

```

plantuml

@startuml

' Style settings
hide circle
skinparam classAttributeIconSize 0

' Entity: Users
entity "users" as users {
    + user_id : UUID [PK]
    --
    name : varchar
    email : varchar [UNIQUE]
    hashed_password : varchar
    joined_at : datetime
}
  
```

Copy Edit

AI-Assisted Coding: Supercharge Your Workflow

Discover how AI tools revolutionize the coding phase, transforming challenges into opportunities for efficiency and innovation.





Phase 3: Coding - Bringing Design to Life

This phase is where your designs and requirements materialize into functional software. It's the heart of development, turning concepts into reality.

Core Definition

Writing, implementing, and integrating software components for a cohesive system.

Vital Importance

Directly translates design and requirements into functional, executable software.

Traditional Hurdles

- Repetitive boilerplate code
- Debugging complex logic
- Learning new APIs efficiently
- Maintaining code quality

AI Pair Programmers: Your Intelligent Coding Assistants

AI tools are not just helpers; they're intelligent collaborators, streamlining your coding process and boosting productivity.

Smart Code Completion

Predicts and suggests entire lines, functions, or code blocks as you type.

Code from Comments

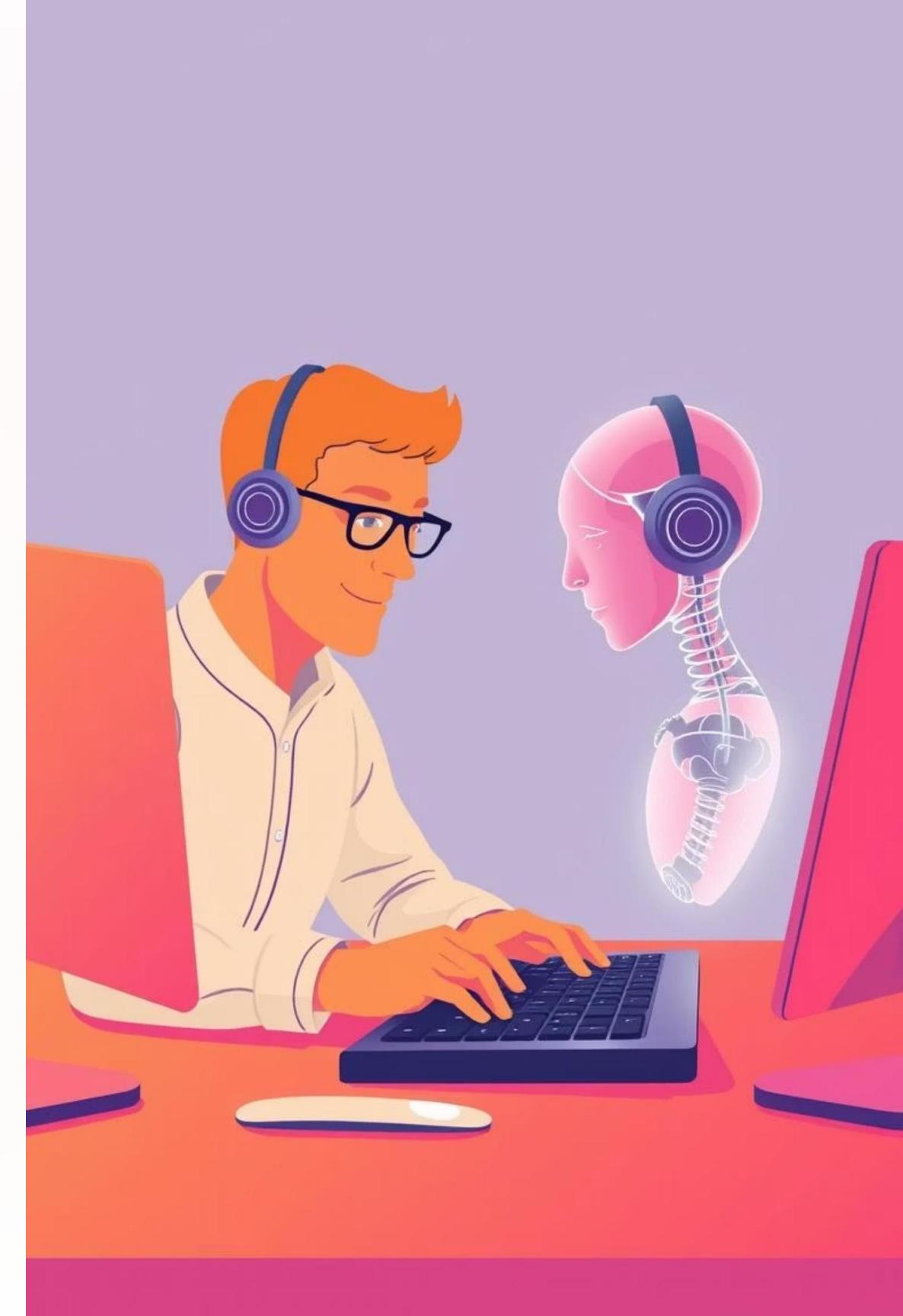
Transforms natural language comments into executable, functional code.

Boilerplate Reduction

Automates repetitive setup tasks like UI widgets or database connections.

API Usage Guidance

Recommends effective ways to use new libraries and frameworks.



Leading AI Coding Tools

Explore the top AI coding tools that are shaping the future of software development.



GitHub Copilot

Top-tier code generation and context-aware suggestions (free for students and open-source contributors).



Tabnine (Free Tier)

Context-aware code completion that learns from your code patterns for smarter suggestions.



Amazon CodeWhisperer (Individual)

Real-time, AI-powered code recommendations directly in your IDE, supporting multiple languages.



Codeium (Free Tier)

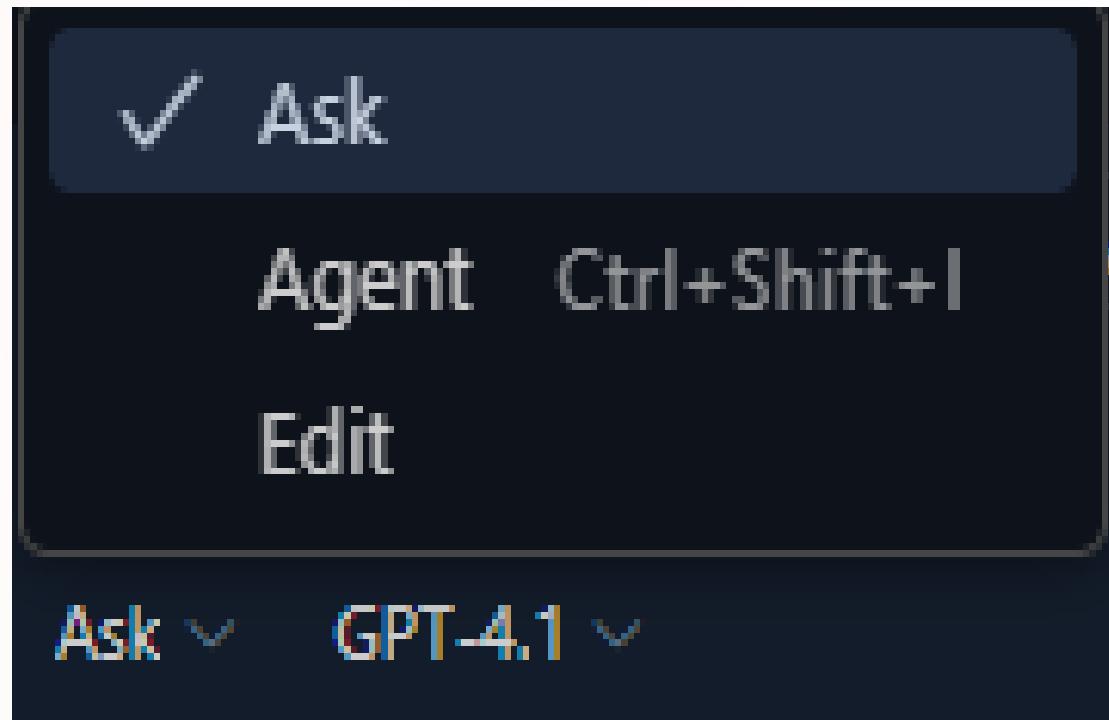
AI-powered autocomplete and code generation, designed for speed and developer productivity.

The **GitHub Copilot** has three Modes.

- Ask
- Agent
- Edit

1. Ask Mode

- Quick, conversational help—ideal for understanding code or clarifying concepts.
- No changes made; the AI provides answers based on your context



2. Edit Mode

- Lets you select specific files and scope for changes.
- You approve or reject each rewrite—and have control over edits.

3. Agent Mode

- Acts as a real-time pair programmer **inside the IDE** (VS Code).
- Plans and executes multi-step tasks: editing files, running commands, fixing runtime errors automatically.
- Synchronous interaction; AI actively iterates within your workspace

Core AI Models Powering GitHub Copilot

Copilot Models	
Claude Sonnet 3.5	1x
Gemini 2.0 Flash	1x
✓ GPT-4.1	1x
GPT-4o	1x
o3-mini	1x

1 GPT-4o

A multimodal powerhouse for fast, high-quality suggestions, ideal for code explanations, documentation, and bug investigations.

2 GPT-4.1

The default GPT-4 model, providing solid reasoning and reliable code generation, ensuring consistency across tasks.

3 O3-mini

Lightweight, ultra-responsive models perfect for quick syntax fixes, variable renaming, and small, efficient code edits.

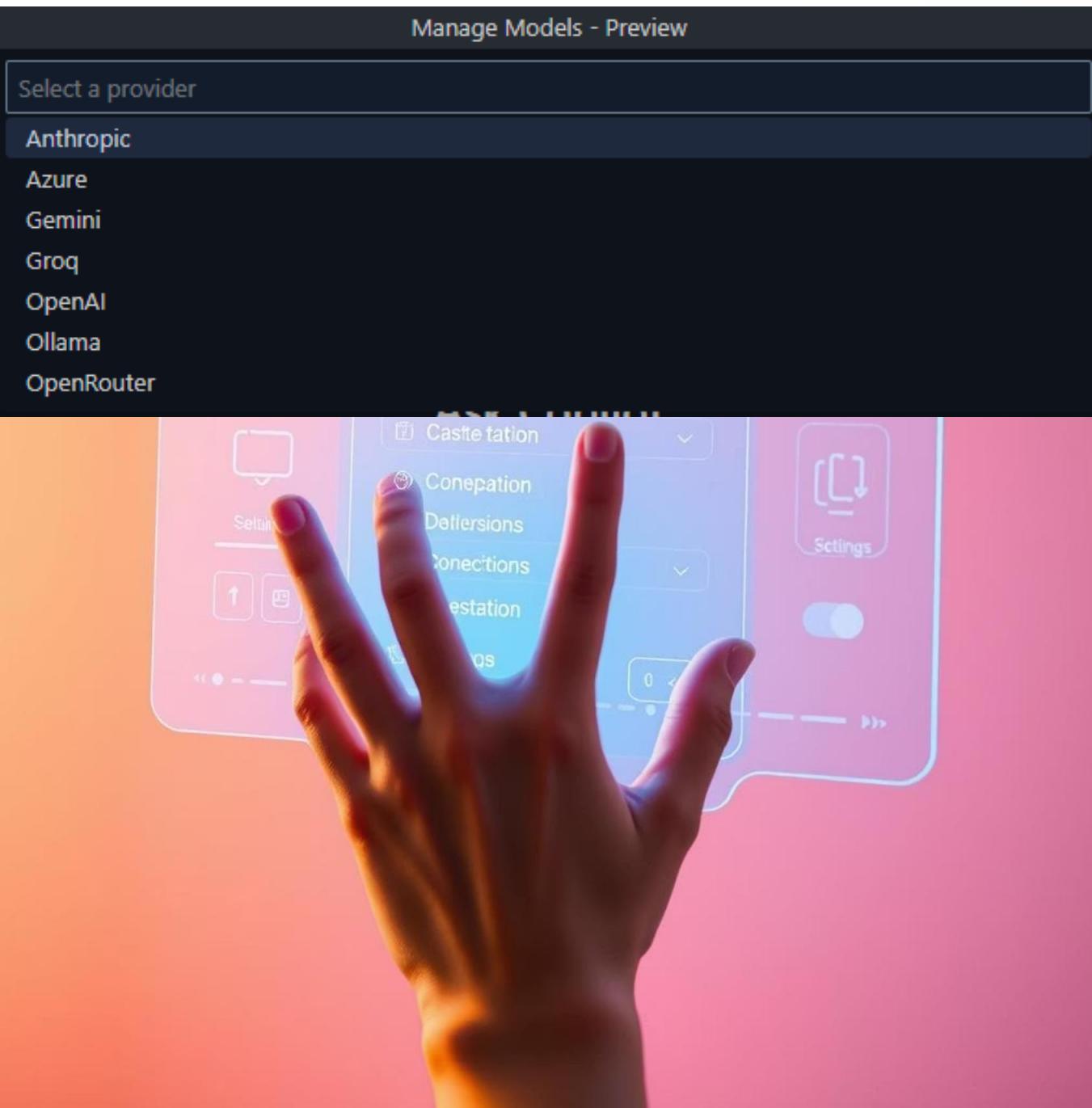
4 Claude 3.5

Excels in deep architectural reasoning, large-scale refactoring, and generating production-level, robust code solutions.

5 Gemini 2.0 Flash

Google's models optimized for rapid code completions and reliable multi-step reasoning, boosting developer productivity.

Managing AI Models in Copilot Chat



Provider Management

Add or remove model providers like OpenAI, Anthropic, Google Gemini, Ollama, OpenRouter, and Azure OpenAI via API keys.

Model Selection

View and enable a range of models, including preview variants (e.g., o1, o3-mini, o4-mini, Gemini Pro, Claude Sonnet).

Customization

Edit provider configurations through a gear icon, allowing updates to API keys and endpoints for flexible setup.

Contextual Switching

Enabled models can be easily switched via the model picker in the Chat UI or through the Settings menu.

AI for Streamlit UI & Logic Generation

Watch AI in action as it rapidly generates common UI components and logic for Streamlit applications.

Scenario 1: Login Form

- **Accelerates UI creation significantly.**
- AI suggests `st.text_input` for username/password.
- Includes basic login logic.
- Generates `st.button` for login actions.

```
import streamlit as st
import database
import hashlib

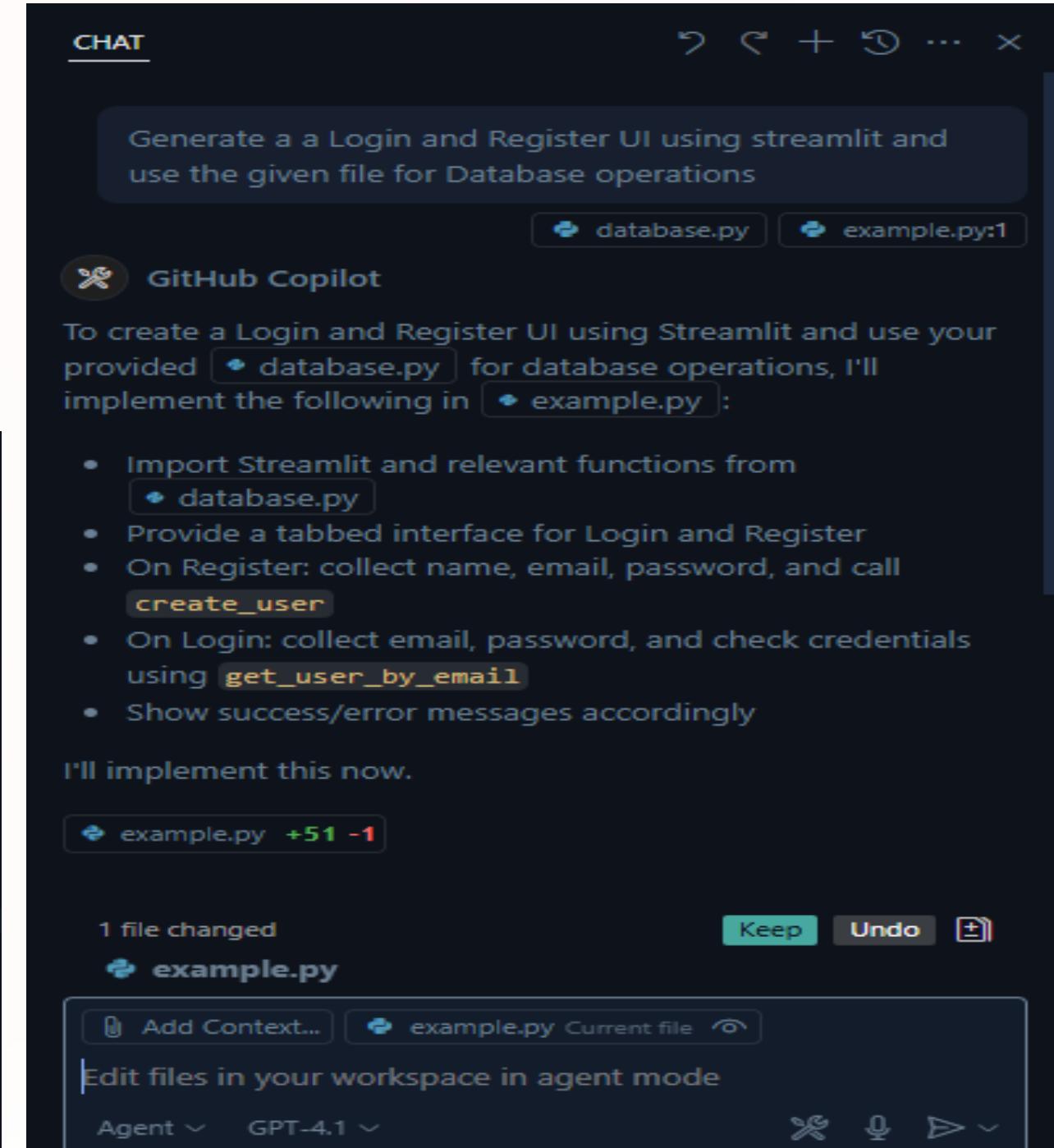
Qodo Gen: Options | Test this function
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

Qodo Gen: Options | Test this function
def main():
    st.title("AI in SDLC - Login & Register")
    tabs = st.tabs(["Login", "Register"])

    # Login Tab
    with tabs[0]:
        st.header("Login")
        login_email = st.text_input("Email", key="login_email")
        login_password = st.text_input("Password", type="password", key="login_password")
        if st.button("Login"):
            user = database.get_user_by_email(login_email)
            if user:
                user_id, name, hashed_password, email = user
                if hash_password(login_password) == hashed_password:
                    st.success(f"Welcome, {name}!")
                else:
                    st.error("Incorrect password.")
            else:
                st.error("User not found.")

    # Register Tab
    with tabs[1]:
        st.header("Register")
        register_email = st.text_input("Email", key="register_email")
        register_password = st.text_input("Password", type="password", key="register_password")
        if st.button("Register"):
            user_id = database.create_user(register_email, register_password)
            st.success("User registered successfully!")

if __name__ == "__main__":
    main()
```



Scenario 2: Task Input Form

- AI assists in generating `st.text_area`, `st.text_input`, `st.time_input`.
- Ensures correct Streamlit API usage.

The screenshot shows a GitHub Copilot interface. On the left is a code editor with `example.py` open, displaying Python code for a Streamlit application. The code includes a registration section and an "Add Task" tab with forms for task title, description, and status. On the right is a "CHAT" window where GitHub Copilot provides suggestions and implements changes.

Code Editor:

```
example.py
example.py > main
def main():
    if success:
        st.success("Registration successful! You can now log in.")
    else:
        st.error("Registration failed. Email may already be registered.")

# Add Task Tab
with tabs[2]:
    st.header("Add Task")
    task_title = st.text_input("Task Title", key="task_title")
    task_description = st.text_area("Task Description", key="task_description")
    task_status = st.selectbox("Status", ["To Do", "In Progress", "Done"], key="task_status")
    if st.button("Add Task"):
        if not task_title or not task_description:
            st.error("Title and Description are required.")
        else:
            success = database.create_task(task_title, task_description, task_status)
            if success:
                st.success("Task added successfully!")
            else:
                st.error("Failed to add task.")

if __name__ == "__main__":
    database.init_db()
    main()


```

CHAT Window:

need task title, description and status

GitHub Copilot

To add a Task Input Form and integrate it with the database, I'll do the following in `example.py`:

- Add a new "Add Task" tab in the Streamlit UI.
- Create a form for task title, description, and status.
- On submission, call a function from `database` to insert the task.

I'll update the file now.

A new "Add Task" tab with a form for task title, description, and status has been added to `example.py`, and it is integrated with the database using a `create_task` function.

If `database.create_task` does not exist yet, let me know and I can help you implement it in your `database.py` file.

Keep Undo

1 file changed

Add Context... example.py Current file

Edit files in your workspace in agent mode



AI-Powered Testing: Elevating Software Reliability



Phase 4: Testing – Ensuring Software Reliability

Definition & Importance

Testing is the critical process of verifying software meets all requirements, functions correctly, and is free of defects. It reduces bugs, improves user experience, builds trust, and significantly lowers long-term maintenance costs.

Traditional Challenges

Manual test case creation is time-consuming and repetitive. Achieving adequate code coverage and writing effective, maintainable unit tests are persistent hurdles that slow down development cycles.



AI's Role: Automated & Intelligent Test Generation



Code Logic Analysis

AI systems deeply understand code logic, identifying complex relationships and potential test scenarios that human eyes might miss.



Automated Unit Test Generation

Tools automatically propose and write robust unit tests for individual functions and methods, freeing developers from boilerplate.



Edge Case & Behavioral Understanding

AI excels at suggesting tests for unusual inputs and boundary conditions, ensuring tests truly validate the intended behavior across all scenarios.



Leading AI Testing Tools



CodiumAI (Qodo)

Focuses on generating meaningful unit tests and analyzing code behavior, providing comprehensive test suites that truly reflect code logic.



Keploy

Automatically generates tests from existing code and usage patterns, capturing real-world interactions to build accurate test cases.



GitHub Copilot

While primarily a code assistant, **Copilot** can also suggest basic unit tests and test stubs, accelerating initial test setup.

Automated Unit Test Generation with CodiumAI

We utilized CodiumAI to generate robust tests for core functions in our application, specifically targeting the `hash_password` and `verify_password` functions to ensure security and reliability.

How it Works

Qodo seamlessly integrates with VS Code. It analyzes your function signatures and logic, then proactively suggests a suite of test cases designed to cover various scenarios, including edge cases.

The screenshot shows the VS Code interface with the CodiumAI extension active. The top status bar indicates the extension is running. The menu bar has 'Sates' (States) selected. The left sidebar shows 'CodiumAI' and 'Untisnt Tests'. The main editor area displays generated test code for 'hash_password' and 'verify_password' functions, including setup code and test cases. A sidebar on the right lists suggestions for 'hash_password' and 'verify_password' functions. The bottom status bar shows the file path 'C:\Users\user\Documents\Project\src\auth\auth.go' and the line number '123'.

```
<love
<ansintess
<cntates (l{
<goorrentatatalfarten>
<cpolganclarneasstoo>
<esgties ontill:
<austen.leustiidonal: {
  detopentin:
  tozenitesst_panaAul {
    <leccbale_pastormall:
    custten
    <chesn-Pasword>
      <achale_per -cal):
      <stajoc():
    per
  }
<chase_passoval ;
<uubl.Zoaccrnalpos(bt) {
  <curllossteerIaf> -val).
  <ductitmule_intes_pepco(te>
  <lonal_entloctor(yl>
  per -val)>
  <atcha_ndrea_data(> {
}
```

CodiumAI

```
cotes
cuttion.ava beticyress flont hapertparal(
<cleths astaandsind(or {
  Yeas_rabic mabile macloes apppates>All bash.interal) - Add plop erecofalltes:
```

```
<hesalAI petrinn trant/lessuer/TVTC/Vseclussal Satead/Vaton(Passunce Slice es):
<reasctite nescnalll fiockery>
<Mustile <hasal.pk>
<uupoore <FJaniCatione>
<iposime <Ceat.assocrated (Fasdy)>
<revarte patital hassh_pocl to -olitowarivel, in rastey>
}
```

Step 1:

- **Install VS Code Extension for Qodo Gen**
- Press Ctrl + Shift + X or Go to Extensions in Left Panel.
- Type "Qodo Gen"
- **Install Qodo Gen**

Qodo Gen: AI Coding Agent

Qodo (formerly Codium) [qodo.ai](#) | 665,353 | ★★★★★(414)

Qodo Gen is a quality-first generative AI coding agent platform that offers busy developers a comprehensive AI code agent for generati...

[Disable](#) | [Uninstall](#) | [Auto Update](#) *

[DETAILS](#) [FEATURES](#) [CHANGELOG](#) [DEPENDENCIES](#)

visit [Qodo's website](#) to learn more.

Test Generation

Qodo Gen leverages advanced AI technology to generate comprehensive tests for **any programming language**. Whether you need to test a specific class, a function, or a small section of code, Qodo Gen offers the functionality to generate meaningful test cases. Refine and customize test suites by adding context, generating example tests, and auto-fixing tests.

Use Test Generation to:

- Increase test coverage
- Unfold bugs
- Generate tests for different behaviors, from happy paths to edge cases.

Start by typing `/test` in the chatbox.

Visit [Qodo's website](#) to learn more.

Installation

Identifier	codium.codium
Version	1.5.4
Last Updated	2025-06-25, 21:17:01
Size	26.96MB

Marketplace

Published	2022-12-25, 14:50:11
Last Released	2025-06-23, 20:11:25

Categories

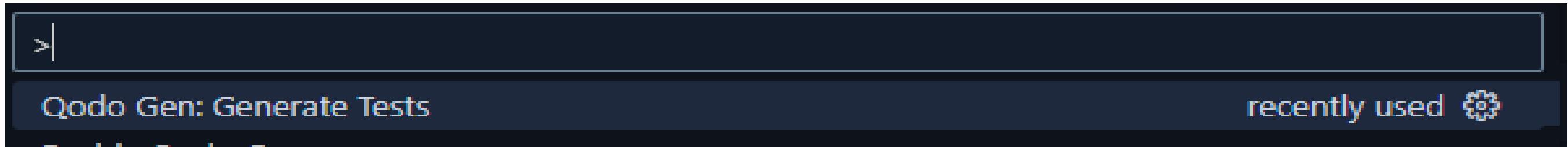
[Programming Languages](#) [Snippets](#)
[Machine Learning](#) [Testing](#) [AI](#)

Step 2: Open Qodo Gen

Press Ctrl + Shift + P

Type "Qodo Gen"

Click "Generate Tests"



Step 3: Generate Unit Tests

Qodo Gen analyzes the entire file

Automatically generates unit tests for all code

The screenshot shows the Qodo Gen interface with the following details:

- Tests**: The top navigation bar.
- Run all tests**: A button to run all tests.
- Run and auto-fix**: A button to run tests and automatically fix issues.
- HAPPY PATH**: A label indicating the current test case.
- Description**: "Function accepts a valid string password and returns a hashed string".
- Code Snippet**:

```
def test_valid_password_returns_hashed_string(self):
    password = "mypassword123"
    result = hash_password(password)
    self.assertIsInstance(result, str)
    self.assertNotEqual(result, password)
    self.assertTrue(len(result) > 0)
    self.assertTrue(bcrypt.checkpw(password.encode('utf-8'), result.encode('u')))
```
- Action Bar**: Buttons for Copy, Delete, and Regenerate.
- Feedback Box**: "What would you like to change?" with a speaker icon.
- Run and auto-fix**: Another button at the bottom right.
- EDGE CASE**: A label indicating the current test case.
- Description**: "Function handles empty string password".

Step 4: View and Run Tests

Review the generated test cases

Click "Run Tests" to execute them

See if tests pass or fail

The screenshot shows a dark-themed test runner interface. At the top right, there is a purple button labeled "Run all tests". Below it, a green button labeled "Passed" with a checkmark icon is visible. To the right of the "Passed" button is another button labeled "Run and auto-fix". On the far right, a blue button labeled "HAPPY PATH" is partially visible. In the center, there is a code editor window displaying Python test code. The code defines a test function named `test_valid_password_returns_hashed_string`. It uses the `hash_password` function to hash a password string "mypassword123". The test then asserts that the result is a string, not equal to the original password, has a length greater than zero, and matches the hashed version of the password using `bcrypt.checkpw`. Below the code editor, there are two small icons: a thumbs-up and a thumbs-down. At the bottom, there are three buttons: "Copy", "Delete", and "Regenerate".

```
def test_valid_password_returns_hashed_string(self):
    password = "mypassword123"
    result = hash_password(password)
    self.assertIsInstance(result, str)
    self.assertNotEqual(result, password)
    self.assertTrue(len(result) > 0)
    self.assertTrue(bcrypt.checkpw(password.encode('utf-8'), result.encode('u')))
```

🚀 Launching Your Streamlit Apps to the World 🚀

We'll explore how to bring your interactive data tools and AI models to users seamlessly, focusing on the powerful and user-friendly [Streamlit Community Cloud](#). Get ready to launch your creations!





Phase 5: Deployment - Delivering Value to Users

What is Deployment?

Deployment is the crucial final step of making your developed software available and operational for end-users. It's where your hard work translates into tangible value.

Why is it Important?

Without deployment, even the most innovative applications remain personal projects. It's how your project's value is truly realized and shared with the world.

Traditional Deployment Challenges

Server Management

Setting up and managing servers and underlying infrastructure.

Dependency Hell

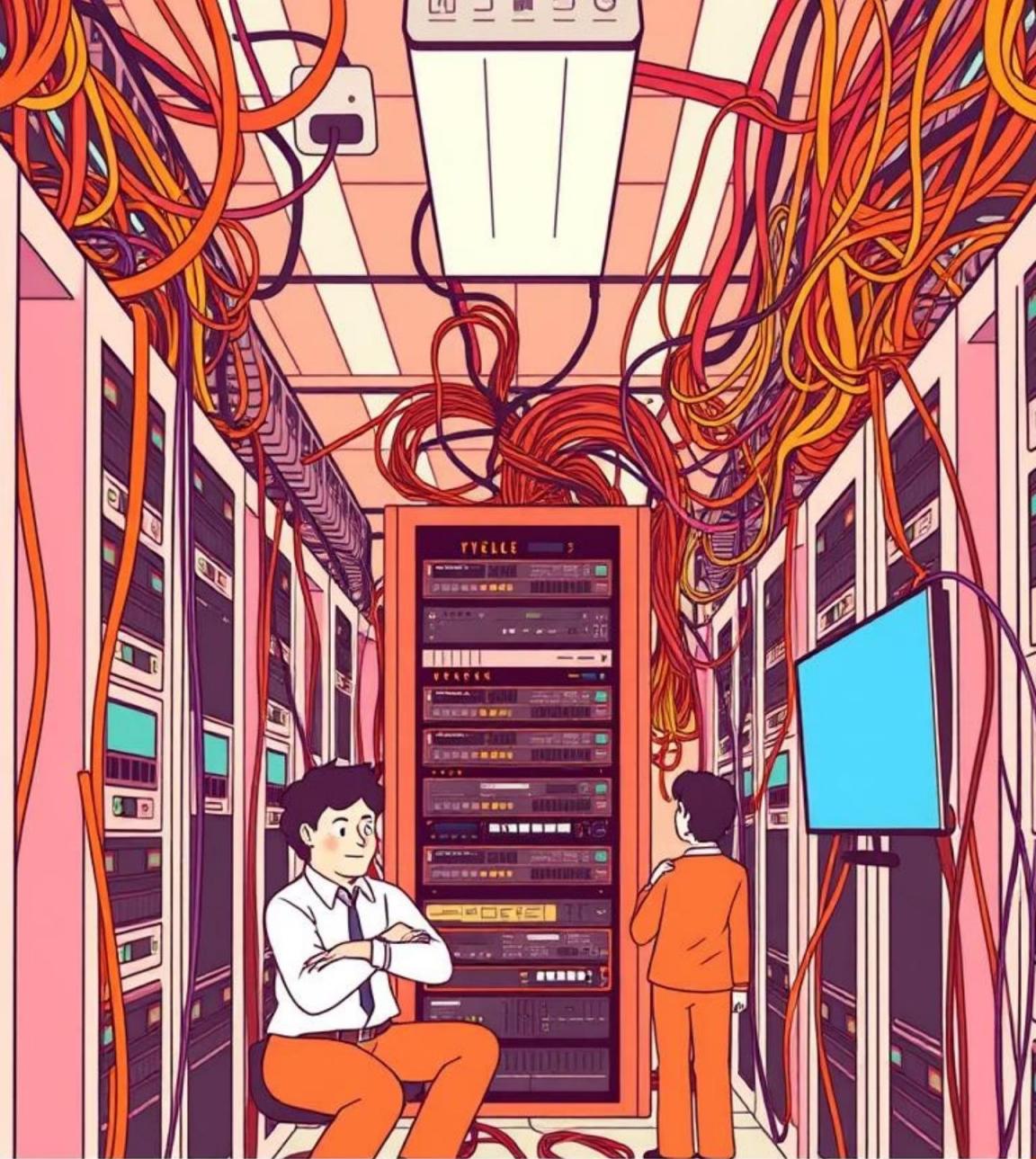
Handling diverse dependencies and ensuring consistent environments.

Scalability & Reliability

Ensuring the application scales with user demand and remains reliable.

Cost & Complexity

Often associated with significant financial cost and operational complexity.



I am appp firs the renagive.
Manecties tresfinemant bold inptirtanch.

AI-Friendly Free Cloud for Seamless Deployment

Streamlit Community Cloud (Highly Recommended)

- **Purpose-Built:** Designed specifically for Streamlit applications.
- **Zero-Config Deployment:** Connects directly to GitHub repo; builds & deploys automatically.
- **Automatic Updates:** App syncs with Git pushes for continuous delivery.
- **Free Tier:** Unlimited public apps for demonstration and personal projects.
- *Note: Apps may sleep after inactivity for resource management.*

Hugging Face Spaces

- Ideal for ML model demos and Streamlit/Gradio apps.
- Offers a free CPU tier for cost-effective development.
- Easy GitHub integration for smooth version control.
- Great for sharing models and interactive AI experiences.

Why Choose Streamlit Community Cloud?

</>

Simplified Workflow

Focus on your code, not infrastructure. Streamlit handles the complexities of deployment.



Global Accessibility

Your app gets a unique public URL, accessible to anyone, anywhere.



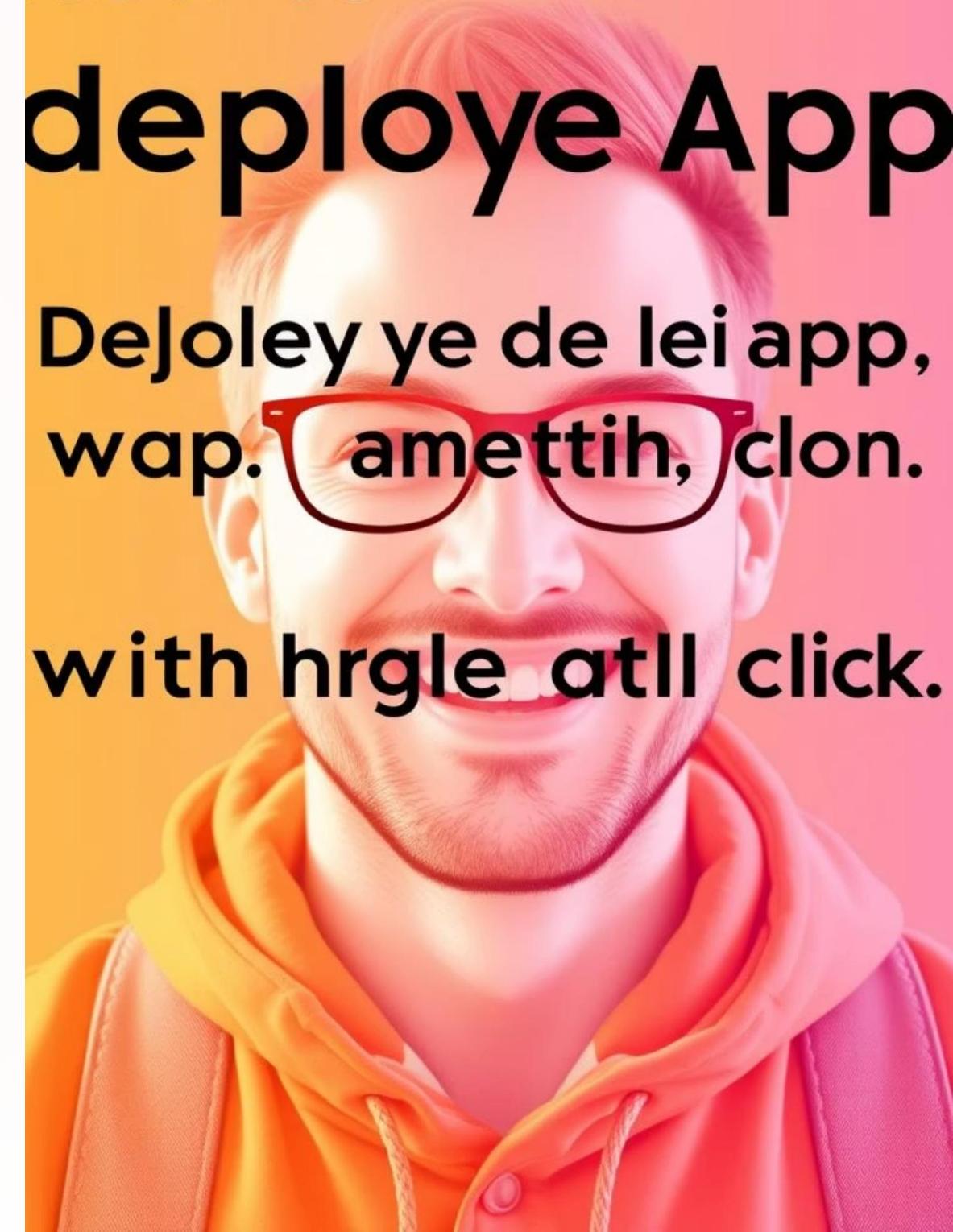
Git-Integrated

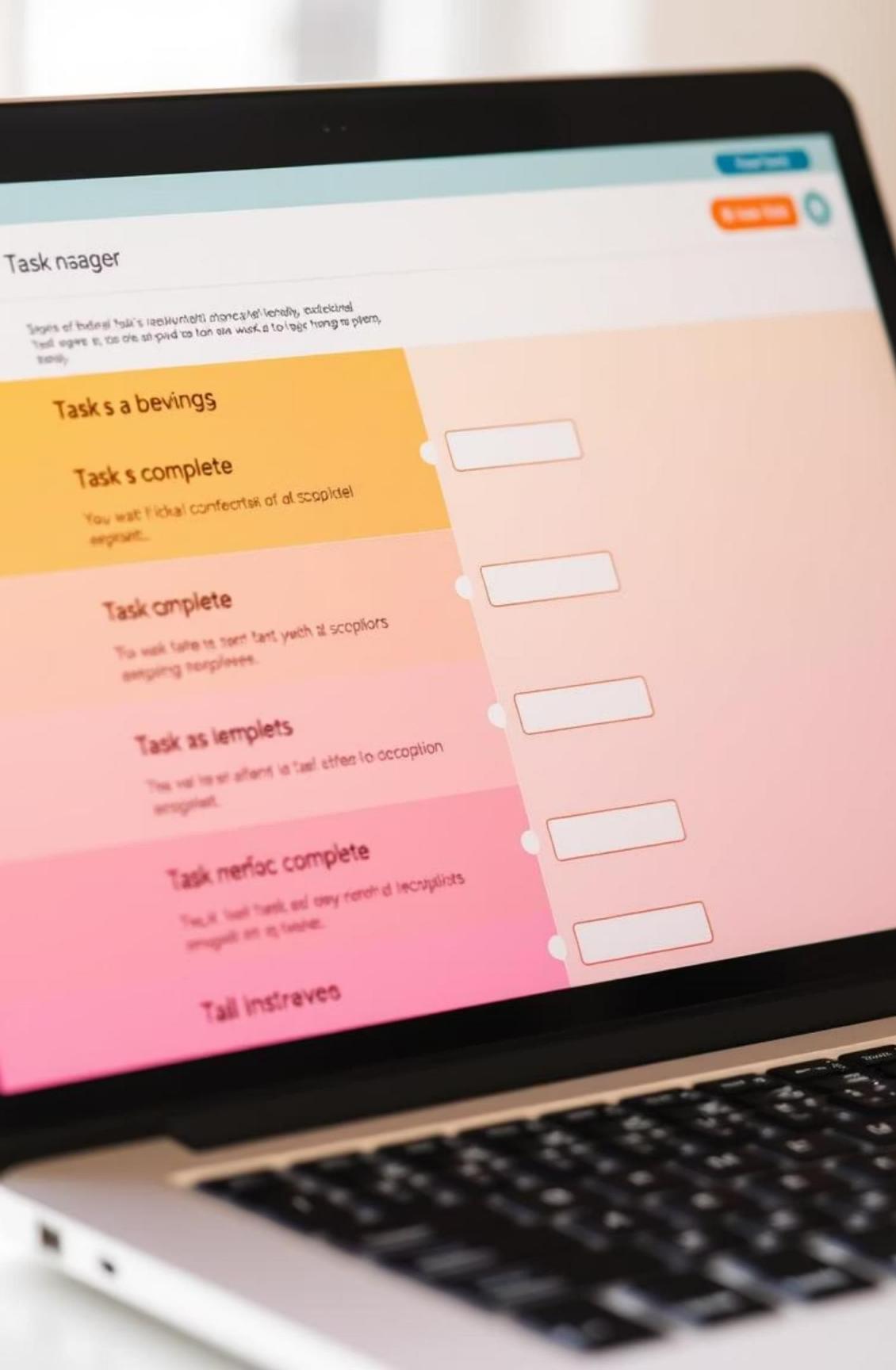
Direct connection to your GitHub repository for seamless version control and updates.

How to deploye App

DeJoley ye de lei app,
wap. amettih, clon.

with hrgle atll click.





Deploying the Daily Task Manager

Let's walk through the simple steps to get our **Daily Task Manager** app live on Streamlit Community Cloud.

Prerequisites Ready

Ensure your `app.py` and `requirements.txt` are committed to a public GitHub repository.

Connect GitHub

Navigate to share.streamlit.io and link your GitHub account.

Select & Deploy

Choose your repo, branch, and `app.py` file, then click "Deploy!"

Go to Streamlit Cloud

Visit share.streamlit.io to begin the process.

Connect to GitHub

Log in and authorize **Streamlit** to access your GitHub account.

Once connected, your GitHub repositories will be available for deployment.

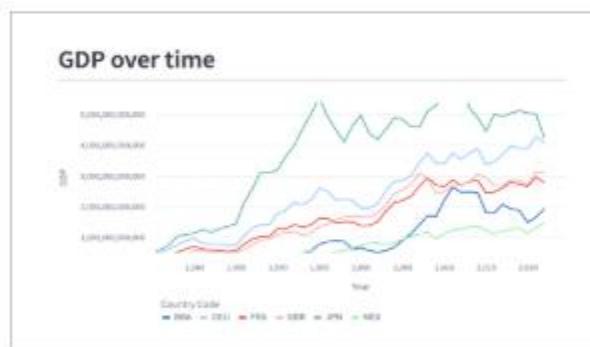
Home Page Overview

After linking GitHub, you will be directed to the **home page** where you can explore pre-built templates.

These templates are ready for deployment, making it easier to start quickly.

Get started from a template

[View all templates →](#)



GDP dashboard

[View demo](#)

A screenshot of a "Chatbot" template. It features a text input field labeled "What's up?" and a "Send" button. Below the input field, there is a message placeholder: "This is a simple chatbot that uses OpenAI's GPT-3.5 model to generate responses. To use this app, you need to provide an OpenAI API key, which you can get [here](#). You can also learn how to build this app step-by-step by [following our tutorial](#)."

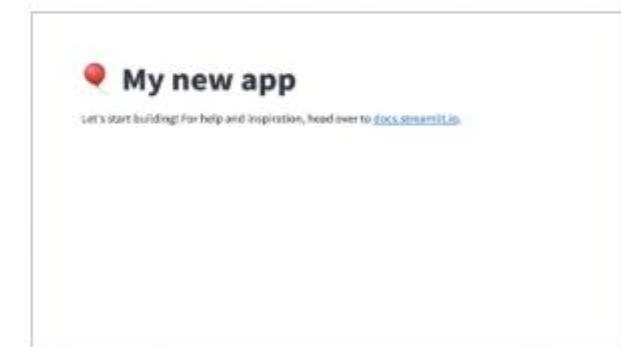
Chatbot

[View demo](#)

Existing tickets					
ID	Title	Status	Priority	Date Submitted	
TICKET-1234	Website performance degradation	Closed	Low	2023-01-27	
TICKET-1235	Database connection lost after sending notifications	In Progress	Medium	2023-01-26	
TICKET-1236	System updates causing compatibility issues	Closed	Medium	2023-01-25	
TICKET-1237	Database connection failure	Closed	High	2023-01-24	
TICKET-1238	Security vulnerability identified	Open	Medium	2023-01-23	
TICKET-1239	Website performance degradation	Closed	Medium	2023-01-22	
TICKET-1240	Customer data loss due to DDoS	Closed	Low	2023-01-21	

Support tickets

[View demo](#)



Blank app

[View demo](#)

Create a New App

In the top-right corner, click on “**Create**

App” to start a new deployment.



The screenshot shows a navigation bar with several items: "My apps" (highlighted in grey), "My profile", "Explore", "Discuss", and "Create app" (highlighted in grey). The "Create app" button is located in the top right corner of the bar.

Select “Deploy Public App from GitHub”

[← Back](#)

What would you like to do?



Deploy a public app from GitHub

My code is ready on a GitHub repo, and it is **totally awesome**.

[Deploy now](#)



Deploy a public app from a template

I want to see what kind of amazing concoctions you have for me.

[Check out templates](#)



Deploy a private app in Snowflake

I want unlimited enterprise-grade apps, with the security of Snowflake.

[Start trial →](#)

[← Back](#)

Step 1: Select Your App Repository

Choose the repository where your application code is stored.

Deploy an app

Repository 

[Paste GitHub URL](#)

Srikanth1905/Task-Tracker

Step 2: Choose Branch

Select the desired branch from your repository.

Branch

main

Step 3: Specify Main File Path

Provide the file path to the main file of your application.

Main file path

app.py

Step 4: (Optional) App URL

You can add an app URL, if necessary, but it's optional.

App URL (optional)

task-tracker-preview

.streamlit.app

Step 5: Click Deploy

Press the "Deploy" button to start the deployment process.

Domain is available

[Advanced settings](#)

[Deploy](#)

Questions?

Thank You All for the Patience & Attention!

Sadhu Sreenivas & Team