Project Report: Exploring AI Tools in Software Development

Student Name: Minav Karia

Course: AI Receptive Software Development

Date: October 14, 2025

Part 1: Introduction (10 points)

1.1 Selected AI Tools

For this project, I have chosen to work with two prominent AI tools that represent different paradigms of AI assistance in software development:

1. **GitHub Copilot:** An AI pair programmer that integrates directly into the code editor (Visual Studio Code, in my case).
2. **ChatGPT (GPT-4 Model):** A versatile conversational AI developed by OpenAI, accessed through its web interface.

1.2 Introduction to the Tools

GitHub Copilot

GitHub Copilot is a cloud-based artificial intelligence tool developed by GitHub and OpenAI. It is designed to function as an "AI pair programmer," providing real-time code suggestions and auto-completions directly within an Integrated Development Environment (IDE).

- **Purpose:** The primary purpose of Copilot is to increase developer productivity by automating the writing of boilerplate code, suggesting entire functions or code blocks, and helping developers navigate new libraries and frameworks more quickly. It aims to reduce the time spent on routine coding tasks, allowing developers to focus on more complex problem-solving.
- **Primary Features:**
    - **Context-Aware Code Completion:** Copilot analyzes the context of the code being written—including comments, function names, and surrounding code—to provide highly relevant suggestions.
    - **Multi-Language Support:** It supports a wide array of programming languages, including Python, JavaScript, TypeScript, Ruby, Go, and more.
    - **From Comments to Code:** A developer can write a comment describing the desired logic, and Copilot can generate the corresponding code block.

    - **Boilerplate Reduction:** It excels at generating repetitive or standard code, such as class definitions, API calls, or unit test structures.
    - **Learning and Exploration:** It can be a powerful tool for learning a new language or framework, as it suggests idiomatic ways of writing code.

ChatGPT (GPT-4)

ChatGPT, powered by OpenAI's GPT (Generative Pre-trained Transformer) models, is a large language model (LLM) designed for natural language understanding and generation. While not exclusively a coding tool, its advanced reasoning and code generation capabilities make it invaluable for software development.

- **Purpose:** In a development context, ChatGPT serves as a multi-purpose assistant. It can be used for brainstorming, high-level architectural design, debugging complex issues, generating documentation, writing test cases, and explaining code. It acts as a conversational partner that can help solve problems that are broader than just writing the next line of code.
- **Primary Features:**
    - **Conversational Interface:** Developers can ask questions and refine requests in plain English, allowing for a natural and iterative problem-solving process.
    - **High-Level Brainstorming:** It can suggest different architectural patterns, technology stacks, or approaches to solving a complex problem.
    - **Code Generation & Explanation:** It can generate code snippets, functions, or even entire scripts based on a description and, crucially, explain how the code works.
    - **Debugging and Error Analysis:** A developer can paste an error message or a problematic code block and ask for an explanation and a potential fix.
    - **Documentation and Content Creation:** It is highly effective at writing technical documentation, such as `README.md` files, API docs, and comments.

Together, GitHub Copilot and ChatGPT provide a comprehensive AI-assisted development experience. Copilot excels at the micro-level, in-editor tasks, while ChatGPT excels at the macro-level, conceptual and explanatory tasks.

Part 2: Experimentation (30 points)

2.1 Project Task: Creating a Simple Weather Web Application

To test the capabilities of these AI tools, I undertook a small project: to build a simple web application using Python's **Flask** framework. The application's goal is to:

1. Present a user with a web form to enter a city name.
2. Fetch real-time weather data for that city from a third-party API (OpenWeatherMap).
3. Display the weather information (temperature, humidity, description) back to the user.

This project was chosen because it involves several common development tasks: setting up a web server, handling user input, making external API calls, parsing JSON data, and rendering data in an HTML template.

2.2 Documentation of the Process

Step 1: Project Scaffolding and Brainstorming (ChatGPT)

I started with a blank slate. My first interaction was with ChatGPT to outline the project structure and necessary steps.

- **My Prompt to ChatGPT:**
  "I want to build a simple Flask web application that gets the weather for a city using the OpenWeatherMap API. What are the main steps and what should my project file structure look like?"
- **How ChatGPT Assisted:** ChatGPT provided a clear, step-by-step plan and a recommended file structure:

```
None

/weather_app
|-- app.py             # Main Flask application logic
|-- templates/
|   |-- index.html     # HTML form and results display
|-- static/
|   |-- style.css      # CSS for styling (optional)
|-- .env               # To store the API key securely
|-- requirements.txt   # Python dependencies
```

- It also listed the necessary libraries (`Flask`, `requests`, `python-dotenv`) and provided

the `pip install` commands. This saved me the initial mental overhead of planning and setup.

Step 2: Writing the Core Flask Application (GitHub Copilot)

Inside `app.py`, I leveraged GitHub Copilot heavily.

- **Process:** I started by writing comments describing what I needed.
    1. I wrote `# Import necessary libraries`. Copilot immediately suggested `from flask import Flask, render_template, request` and `import requests`.
    2. I wrote `# Initialize Flask app`. Copilot generated `app = Flask(__name__)`.
    3. The most impressive part was the main route. I wrote the comment: `# Define the main route that handles both GET and POST requests`. Copilot then generated almost the entire Flask route function.
- **Example of Copilot's suggestion:** After I typed `def index():`, Copilot suggested the following block, which was about 90% accurate. I only had to tweak the API URL and the JSON parsing logic slightly.
  Python

```
None

# (Copilot's suggestion after I wrote the function signature)
if request.method == 'POST':
    city = request.form.get('city')
    api_key = 'YOUR_API_KEY' # I replaced this later
    url =
f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={
api_key}&units=metric'
    response = requests.get(url)
    weather_data = response.json()
    # ... logic to handle data ...
    return render_template('index.html', weather=weather_data)
return render_template('index.html', weather=None)
```

-

Step 3: Creating the HTML Template (ChatGPT and Copilot)

For `templates/index.html`, I needed a form and a section to display results.

- **My Prompt to ChatGPT:**
  "Create a simple HTML page using Bootstrap 5. It should have a form with one text input for a 'city' and a submit button. Below the form, it should have a section that only displays if weather data is available. Use Jinja2 templating to display the city name, temperature, and weather description."
- **How ChatGPT Assisted:** ChatGPT provided a complete, well-structured HTML file with Bootstrap CDN links, the form, and the correct Jinja2 conditional blocks (`{% if weather %}`) and variable placeholders (`{{ weather.main.temp }}`). This saved me at least 15-20 minutes of looking up Bootstrap classes and Jinja2 syntax.
- **Copilot's Role:** While editing the HTML in VS Code, Copilot assisted with small tasks like closing tags and auto-completing class names.

Step 4: Debugging and Refinement (ChatGPT)

I encountered a `KeyError` when I entered an invalid city name. The `weather_data` JSON returned by the API was an error message, not the expected weather structure.

- **My Prompt to ChatGPT:**
  "My Python code is getting a KeyError when I enter a fake city name like 'asdfg'. The error is `KeyError: 'main'`. Here is my code: [I pasted the Flask route function]. How can I fix this?"
- **How ChatGPT Assisted:** ChatGPT explained that the API returns a different JSON structure for errors (e.g., `{'cod': '404', 'message': 'city not found'}`). It suggested adding a check for the response status code or the `cod` key in the JSON before trying to access `weather_data['main']`. It provided the corrected Python code, which worked perfectly.

2.3 Challenges Encountered

1. **Over-reliance on Copilot:** At one point, I accepted a suggestion from Copilot that included a logic error. It suggested a variable name that was not yet defined in my code, causing a `NameError`. This highlighted the need to always review and understand the AI-generated code, not just blindly accept it.
2. **Generic ChatGPT Suggestions:** For the initial HTML, ChatGPT's suggestion was functional but stylistically very basic. I still needed to apply my own CSS and design sense to make it look presentable.
3. **API Key Security:** Both tools initially placed my API key directly in the `app.py` file. I had to use my own knowledge to implement a more secure method using a `.env` file and the `python-dotenv` library. The AI tools didn't proactively suggest this best practice.

Part 3: Analysis and Reflection (30 points)

3.1 Effectiveness Analysis

The use of AI tools was highly effective in accelerating the completion of this project. I estimate that the project, which took about 45 minutes, would have taken closer to 2 hours without these tools.

Strengths:

- **Massive Speed Increase:** The single greatest strength was the raw speed. GitHub Copilot's ability to generate entire blocks of boilerplate code from a single comment or function name was a game-changer. ChatGPT's ability to generate a full HTML template from a prose description was equally impressive.
- **Reduced Cognitive Load:** I didn't have to spend mental energy recalling the exact syntax for a Flask route, the structure of a Bootstrap form, or how to parse JSON in Python. This allowed me to stay focused on the overall logic and goal of the application.
- **Excellent Debugging Partner:** ChatGPT's ability to analyze an error message and code snippet and provide a clear explanation and fix is invaluable. It's like having a senior developer on call to explain your mistakes.
- **Democratization of Knowledge:** For a developer new to Flask or APIs, these tools can bridge knowledge gaps in real-time, functioning as an interactive, on-demand tutor.

Weaknesses:

- **Potential for Subtle Bugs:** AI-generated code is not infallible. As I experienced, it can introduce subtle bugs, use incorrect variable names, or miss edge cases. The code *looks* plausible, which can make debugging more difficult if not reviewed carefully.
- **Lack of Architectural Insight:** The tools are excellent at tactical, line-by-line or function-by-function implementation. However, they lack a true understanding of the project's high-level goals. They won't question your approach or suggest a better architectural pattern (e.g., "Maybe you should separate your routes into a blueprint for this larger application").
- **Security and Best Practices:** The tools do not always default to the most secure practices. As seen with the API key, the developer's expertise is still required to ensure the code is robust, secure, and maintainable.
- **Risk of Stifling Learning:** Over-reliance on these tools could prevent junior developers from learning the fundamental principles behind the code. If you never have to write a `for` loop by hand because the AI always does it for you, you may not fully grasp how it works.

3.2 Reflection

My experience with this project has solidified my view that AI tools are not a replacement for human developers, but rather powerful force-multipliers.

Integration into My Workflow:

I see myself adopting a hybrid workflow. I will use **ChatGPT** at the beginning of a project for brainstorming, planning, and generating initial templates. It will also be my go-to tool for debugging errors I don't immediately understand and for generating documentation. I will use **GitHub Copilot** continuously during the coding process to handle boilerplate, auto-complete functions, and accelerate the minute-to-minute task of writing code. The key is to treat the AI's output as a "first draft" that I, the developer, am responsible for reviewing, refining, and approving.

Impact on the Future of Software Development:

I believe these tools will fundamentally change the landscape of software development in several ways:

1. **Shift in Developer Skills:** The value of a developer will shift from being a master of syntax to being a master of system design, architecture, and critical thinking. The most important skills will be prompt engineering (the ability to ask the AI the right questions), code review, and the ability to integrate AI-generated components into a cohesive, secure, and efficient system.
2. **Increased Productivity and Prototyping Speed:** The time from idea to a working prototype will be drastically reduced. This will allow for more rapid innovation and experimentation.
3. **Focus on Higher-Order Problems:** By automating the more tedious aspects of coding, AI will free up developers to spend more time on creative problem-solving, user experience, and business logic.
4. **A "Golden Age" for Senior Developers:** Senior developers who can effectively guide and review the output of AI tools will become even more valuable. Their experience and intuition will be critical in steering projects and avoiding the pitfalls of AI-generated code.

In conclusion, AI tools like Copilot and ChatGPT are transformative. They do not make software development "easy," but they do make developers more powerful. The future belongs to the developers who learn to effectively partner with these AI assistants.