# AI Tool Setup Report

**Huaye Li**

## 1. Introduction

Week 3 of the Personalized Movie Recommendation System project centered on configuring the essential development tools, platforms, and libraries required to implement and deploy the system. This technical groundwork supports the integration of machine learning (ML), natural language processing (NLP), web development, and user interface design. The chosen toolchain was based on compatibility with Python, ease of rapid prototyping, and support for collaborative and AI-assisted development.

## 2. Development Environment

### 2.1 IDE: PyCharm

The primary integrated development environment was **PyCharm Community Edition**, selected for its deep integration with Python workflows, virtual environment management, code intelligence, and plugin ecosystem.

**Key benefits utilized:**

- Real-time syntax checking and linting
- GitHub plugin for version control
- Integrated terminal and debugger
- Support for virtual environments (venv)

### 2.2 Version Control: Git and GitHub

The project repository was initialized on **GitHub** to track code changes, manage development branches, and share documentation. Key practices included:

- Weekly commits aligned with milestone completion
- Feature branches for individual modules (e.g., feature/tfidf, feature/ui)
- Markdown README for repository documentation

## 3. AI-Powered Tools

### 3.1 GitHub Copilot

The **GitHub Copilot** plugin was installed in PyCharm. It uses OpenAI Codex to suggest intelligent code completions based on context.

**Use cases included:**

- Writing boilerplate functions for poster fetching, TF-IDF computation
- Suggesting fallback logic for missing poster data

- Drafting data preprocessing pipelines

**Observed benefits:**

- Accelerated prototyping
- Reduced manual repetition
- Increased accuracy in function scaffolding

**Limitations:**

- Occasionally offered incomplete or contextually incorrect suggestions
- Required manual validation and editing

### 3.2 OpenAI ChatGPT

**ChatGPT (GPT-4)** was used extensively as a dynamic assistant for problem solving and conceptual clarification.

**Example queries:**

- "Best practices for building a content-based recommender with TF-IDF?"
- "How to structure cosine similarity matrices?"
- "Streamlit interaction tips for layout and responsiveness"

It served as both a code assistant and research resource, offering clarity and confidence in complex decisions.

### 3.3 Uizard for Wireframing

**Uizard**, a web-based wireframing tool, was used to prototype the user interface layout before implementing it in Streamlit.

**UI elements designed:**

- Homepage layout with title, subtitle, search fields
- Movie recommendation cards (poster + title + overview)
- Responsive alignment for user ID input and feedback

Prototypes were exported as reference images to inform UI styling and layout decisions in the app.

## 4. Libraries and Tools

### 4.1 Core Python Libraries

- **pandas** – Data manipulation and merging
- **scikit-learn** – TF-IDF vectorization, cosine similarity
- **requests** – API calls to fetch movie posters
- **re** – Regex string cleaning
- **pickle** – Model serialization and storage

### 4.2 Web Interface: Streamlit

**Streamlit** was used to build the interactive interface. Its ability to integrate Python logic directly into a browser-accessible UI made it ideal for quick iterations.

**Features used:**

- st.text_input, st.number_input for real-time search
- st.columns, st.container for displaying results
- Custom HTML + CSS styling injected with st.markdown

### 4.3 Visualization Tools (Prepared for Week 4)

- **matplotlib** and **seaborn** – User rating distribution and trend plots
- **WordCloud** – Overview-based keyword clouds

## 5. Installation and Environment Setup

### 5.1 Local Setup (macOS)

- Python version: 3.9.13
- Virtual environment setup using venv
- Dependency installation with pip
- Requirements tracked in requirements.txt

### 5.2 OMDb API Integration

API calls to OMDb were used to retrieve movie posters based on imdb_id. An API key was securely passed into function calls and responses were cached to avoid rate limits.

## 6. Challenges and Solutions

**Challenge 1: Library Version Conflicts**

- *Problem:* Conflicts between scikit-learn and older versions of numpy
- *Solution:* Upgraded to compatible versions and locked packages in requirements.txt

**Challenge 2: Streamlit Layout Customization**

- *Problem:* Native layout controls were limited
- *Solution:* Custom CSS with embedded HTML resolved layout issues

**Challenge 3: Copilot Suggestion Errors**

- *Problem:* Suggested incorrect variable references or ambiguous logic
- *Solution:* Manual review and debugging ensured accuracy

## 7. Reflections

The setup phase demonstrated the practical value of AI-assisted development. GitHub Copilot and ChatGPT reduced time spent on routine tasks and enabled deeper focus on higher-level system architecture. Meanwhile, tools like Uizard and Streamlit allowed for fast visualization and feedback without requiring full-stack web development knowledge.

However, reliance on AI tools necessitated critical oversight. While useful, suggestions from Copilot and ChatGPT were not always accurate or context-appropriate.

Overall, the tool setup phase provided a strong technical base for future development. The environment is now fully configured and the foundational architecture is ready for model integration and data visualization.

## 8. Next Steps

In Week 4, the focus will shift to:

- Performing thorough data preprocessing (normalization, filtering, cleaning)
- Conducting exploratory data analysis (EDA)
- Visualizing trends in ratings, genres, overview lengths, and more
- Preparing the cleaned dataset for modeling with TF-IDF and cosine similarity

## Appendix: Tools and Versions

| **Tool / Library** | **Version** |
|---|---|
| Python | 3.9.13 |
| PyCharm | 2023.1 |
| GitHub Copilot | Latest plugin |
| ChatGPT | GPT-4 (Web) |
| Streamlit | 1.27.2 |
| pandas | 1.5.3 |
| scikit-learn | 1.2.2 |
| requests | 2.28.1 |
| seaborn | 0.12.2 |
| WordCloud | 1.9.3 |
| Uizard | Web-based |
| OMDb API | Public endpoint |