Fantasy Adventure Game

Project Report: CLI-to-Web Expansion with GitHub Copilot and PythonAnywhere Deployment

Microsoft AI Engineer Program

December 2025

Prepared by: Jim Cockerham

Table of Contents

Executive Summary

This project delivered a role-playing, text-based fantasy adventure game in two forms: (1) a command-line (CLI) Python game and (2) a Flask-based web version. Development began with a basic narrative premise and a straightforward Python structure (simple data containers plus a main loop). GitHub Copilot was then used to accelerate feature implementation, refactoring, and edge-case handling. Finally, the web version was deployed to PythonAnywhere for hosted access.

Project Background and Narrative

The story follows a classic hero's journey: the player explores locations, gathers equipment, and ultimately defeats the Dragon of Shadowmere. Progression is gated by obtaining the Crystal Sword from the Crystal Cave before confronting the dragon in the Dragon's Lair.

Objectives

- Implement a playable RPG-style adventure loop using core Python constructs (variables, lists/dictionaries, loops, conditionals, and functions).
- Add modular features such as turn-based combat, inventory, and a shop system.
- Adapt the CLI game into a web application while preserving the core mechanics.
- Deploy the web version to PythonAnywhere.

Initial Python Architecture (CLI)

Baseline File Set

The project was organized around separate CLI and web entry points, supported by documentation and deployment metadata:

```
fantasy_adventure_game.py     # CLI version
fantasy_adventure_web.py      # Web version (Flask)
requirements.txt              # Dependencies
Procfile                      # Deployment config (optional, platform-dependent)
README.md                     # Repository overview
Fantasy_Adventure_Game_Documentation.md  # Supporting documentation
```

Game-State Data Model

The CLI version relied on simple, readable containers: a dictionary for player stats, lists for inventory and location definitions, and flags to track quest progress (e.g., whether the Crystal Sword has been acquired and whether the dragon has been defeated).

Functional Decomposition

Gameplay was decomposed into modular functions for input validation, status display, location encounters, combat, shopping, and item usage. A main game loop orchestrated progression by repeatedly presenting choices, resolving outcomes, and updating state.

GitHub Copilot Contributions

GitHub Copilot was used as an accelerator on top of an existing structure: it suggested reusable patterns, generated scaffolding for features, and helped identify refactoring opportunities. All Copilot-generated code was reviewed and adjusted for correctness, theme consistency, and user experience.

Figure 1. Reusable input-validation helper

Copilot proposed the common “while True” loop pattern to validate player choices.

This reduced repeated input-handling code throughout the CLI menus and combat loop.

```
def get_valid_input(prompt: str, valid_options: list) -> str:
    while True:
        user_input = input(prompt).strip().lower()
        if user_input in valid_options:
            return user_input
        print("Invalid choice. Try again.")
```

Figure 2. Random encounter selection pattern

Copilot suggested using random.choice() for encounter selection in the Whispering Forest.

This improved replayability while keeping the logic approachable for a course-level project.

```
encounter = random.choice(["goblin", "wolf", "fairy", "treasure"])
```

Figure 3. Hardening Copilot-generated logic (edge-case safeguards)

Copilot suggestions required tightening to avoid runtime errors or unintended outcomes.

Representative safeguards are shown below.

```
player["health"] = min(player["health"] + heal, player["max_health"])
if not inventory: return  # guard against empty inventory
damage = max(1, raw_damage - player["defense"])
```

Feature Set Implemented

Locations

- Village of Elderbrook: starting area with shop and quest hints.
- Whispering Forest: random encounters (e.g., goblin, wolf, fairy, treasure).
- Crystal Cave: boss encounter (Giant Bat) and Crystal Sword reward.
- Dragon's Lair: final boss encounter (requires the Crystal Sword).

Combat System

Combat is turn-based with player choices such as Attack, Use Potion, and Flee. Damage is randomized within a controlled range and mitigated by defense. Flee attempts have a probabilistic success rate to balance risk and player agency.

Items and Shop

The shop supports purchasing items that directly modify survivability and combat capability.

Item	Price	Effect
Health Potion	15 gold	Restores 30 health
Iron Sword	25 gold	+5 Attack
Leather Shield	20 gold	+3 Defense
Magic Amulet	40 gold	+20 Max Health

Web Expansion (Flask)

The web version preserves the core game mechanics while adapting interaction to HTTP requests. Because a web app does not maintain a continuous terminal session, game state is stored per user using Flask session data.

Figure 4. Session-based state management (web version)

Each player's state is stored in the Flask session (e.g., stats, inventory, flags).

This keeps concurrent players isolated and enables multi-step gameplay through browser interactions.

Deployment to PythonAnywhere

The final web application was deployed using PythonAnywhere's Flask web app workflow. At a high level, deployment consists of creating a PythonAnywhere account, uploading the project files, configuring a new Flask web app (Python 3.11), and pointing the WSGI configuration at the application entry point.

1. Create a PythonAnywhere account.
2. Upload the Flask app files (including the web entry point and requirements).
3. Create a new Flask web app (Python 3.11) from the PythonAnywhere Web tab.
4. Configure the WSGI file to import and run the Flask app.
5. Reload the web app and verify the hosted URL renders correctly.

Validation and Known Issues

During final review, the repository README lists the CLI run command as “python dnd_adventure_game.py,” while the documentation lists “python fantasy_adventure_game.py.” For a clean submission, these filenames and instructions should be aligned so graders and users can run the CLI version without ambiguity.

Conclusion

This project demonstrates a complete, course-appropriate RPG loop implemented with core Python constructs, extended into a Flask web app with session-managed state. GitHub Copilot accelerated development by suggesting reusable patterns and refactors, while developer review ensured correctness and thematic consistency. Deployment to PythonAnywhere provided a straightforward path to hosting and sharing the finished web version.

References

Fantasy Adventure Game Documentation. (2025). Fantasy_Adventure_Game_Documentation.md.

Fantasy Adventure Game Repository README. (2025). README.md.

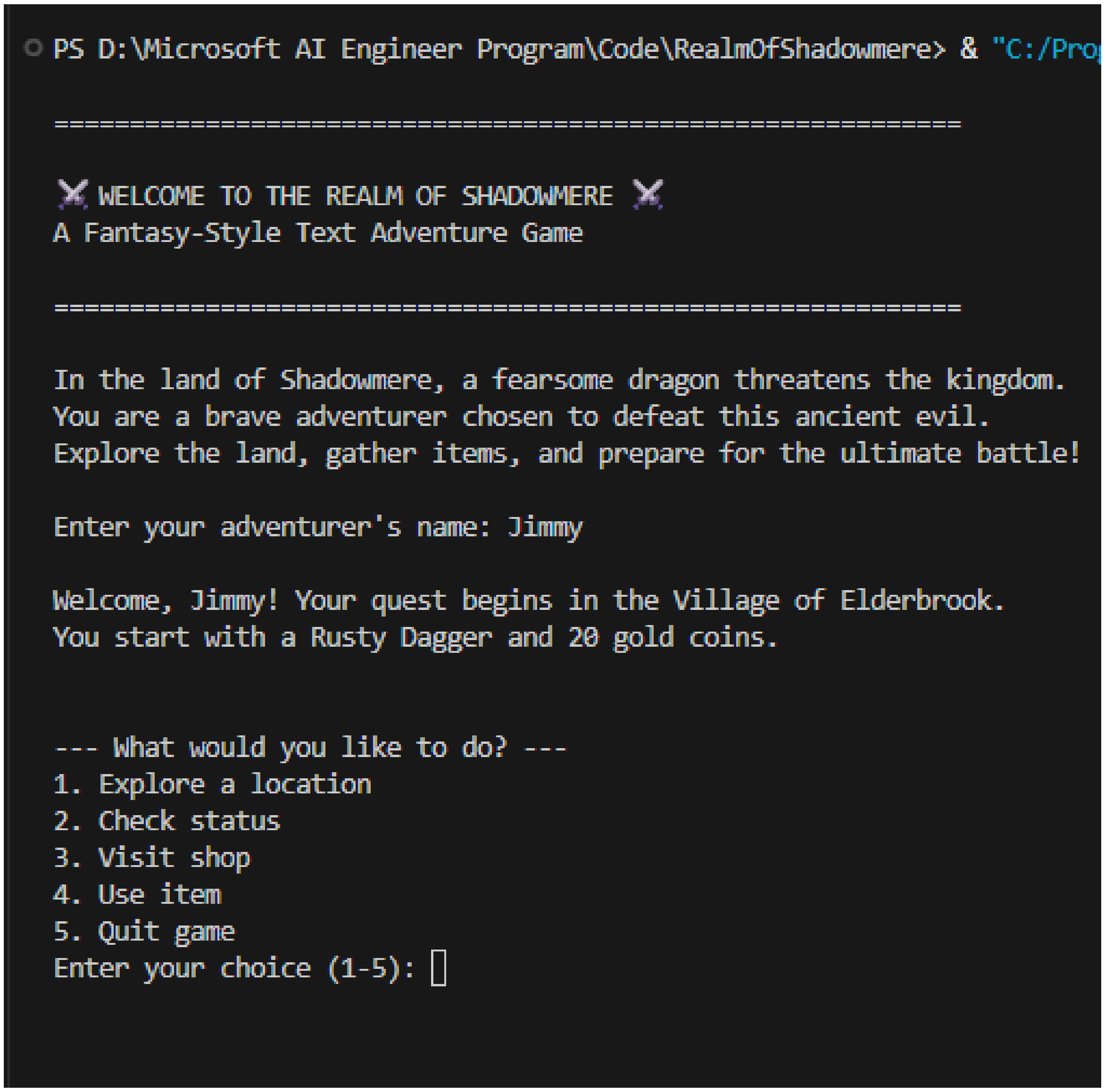
PythonAnywhere. (n.d.). Flask hosting documentation (accessed December 2025).

Appendix: Screenshots

This appendix contains implementation evidence screenshots for the CLI and hosted web versions.

Figure D-1. CLI Run — Game Launch, Player Initialization, and Main Menu

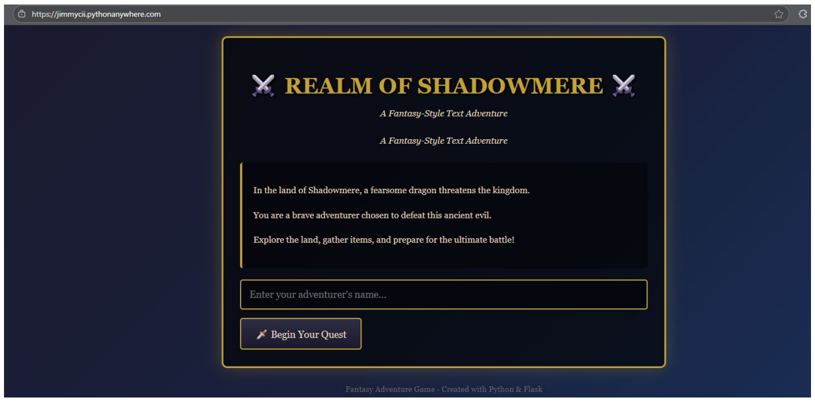
Caption: Terminal output from running the CLI version locally, showing the start banner, narrative introduction, player name entry, initial inventory/gold assignment, and the primary action menu.



```
PS D:\Microsoft AI Engineer Program\Code\RealmOfShadowmere> & "C:/Pro

============================================================

⚔️ WELCOME TO THE REALM OF SHADOWMERE ⚔️
A Fantasy-Style Text Adventure Game

============================================================

In the land of Shadowmere, a fearsome dragon threatens the kingdom.
You are a brave adventurer chosen to defeat this ancient evil.
Explore the land, gather items, and prepare for the ultimate battle!

Enter your adventurer's name: Jimmy

Welcome, Jimmy! Your quest begins in the Village of Elderbrook.
You start with a Rusty Dagger and 20 gold coins.


--- What would you like to do? ---
1. Explore a location
2. Check status
3. Visit shop
4. Use item
5. Quit game
Enter your choice (1-5):
```

Figure D-2. Hosted Web UI — Game Landing Page (PythonAnywhere)

Caption: Browser-based interface for the Flask version of the game, showing the deployed URL, story introduction, player name input, and "Begin Your Quest" action.



https://jimmycii.pythonanywhere.com

⚔️ REALM OF SHADOWMERE ⚔️

A Fantasy-Style Text Adventure

A Fantasy-Style Text Adventure

In the land of Shadowmere, a fearsome dragon threatens the kingdom.

You are a brave adventurer chosen to defeat this ancient evil.

Explore the land, gather items, and prepare for the ultimate battle!

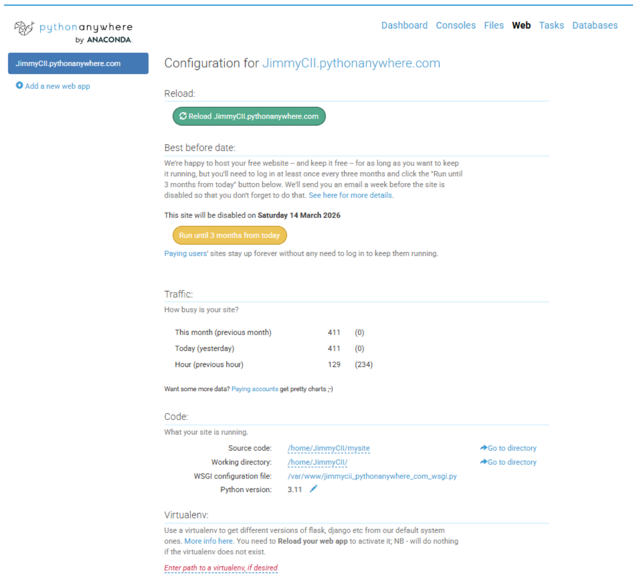
Enter your adventurer's name...

Begin Your Quest

Fantasy Adventure Game - Created with Python & Flask

Figure D-3. PythonAnywhere Web App Configuration (Deployment Evidence)

Caption: PythonAnywhere Web tab showing the hosted domain, reload control, source/working directory, WSGI entry file, and Python runtime version used for the Flask deployment.



pythonanywhere by ANACONDA
Dashboard Consoles Files Web Tasks Databases
JimmyCII.pythonanywhere.com
Add a new web app

Configuration for JimmyCII.pythonanywhere.com

Reload:

Reload JimmyCII.pythonanywhere.com

Best before date:

We're happy to host your free website -- and keep it free -- for as long as you want to keep it running, but you'll need to log in at least once every three months and click the "Run until 3 months from today" button below. We'll send you an email a week before the site is disabled so that you don't forget to do that. See here for more details.

This site will be disabled on **Saturday 14 March 2026**

Run until 3 months from today

Paying users' sites stay up forever without any need to log in to keep them running.

Traffic:

How busy is your site?

This month (previous month)	411	(0)
Today (yesterday)	411	(0)
Hour (previous hour)	129	(234)

Want some more data? Paying accounts get pretty charts ;-)

Code:

What your site is running.

Source code:	/home/JimmyCII/mysite	Go to directory
Working directory:	/home/JimmyCII/	Go to directory
WSGI configuration file:	/var/www/jimmycii_pythonanywhere_com_wsgi.py	
Python version:	3.11	

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. More info here. You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired