Project Statement of Work

Title

Treasure Hunt

Team

Status 417:

Grayson Morris, Callista Robinson, Jacob Cole

Objective

The objective of this project is to develop a multi-player online game using socket programming, which will accommodate up to four players simultaneously. The game is a unique blend of Chess and a Choose Your Own Adventure format, where players start from the corners of a 10x10 grid and move one square at a time (up, down, left, right, or diagonally) to locate hidden treasures. Players may also encounter items that can affect opponents by causing them to lose turns or be eliminated from the game through a health and damage system.

The project will be divided into two main components: the client and the server.

Client Component:

- **Technology**: Python, Sockets, and Flask
- **Functionality**: The client will interface with the game through a web-based user interface (UI) hosted locally using Flask. Its primary role is to handle player interactions within the game. This includes moving characters on the grid and sending messages to other players (either broadcast or targeted).
- **Networking**: The client will establish a TCP connection with the server, capable of handling a client's potentially dynamic IP address. It will manage network communication to maintain real-time updates of the game state and interactions (individual and all player chat) between players.

Server Component:

- **Technology**: Python, Sockets
- **Functionality**: The server will be responsible for the core game logic, including tracking and updating the game state, managing player actions, and enforcing game rules and handling errors gracefully. It will handle connections from multiple clients, ensuring that each player's actions are synchronized, and the game state is consistently updated across all clients.

- **Networking**: The server will establish and maintain TCP connections with clients, handling connection setup, maintenance, and teardown. It will manage data exchange between clients, ensuring reliable communication and real-time updates. Network resilience and error handling will be key aspects of server operation to provide a smooth gaming experience.

This project will integrate basic networking concepts such IP addressing and socket programming to create a simple but playable game.

Scope

Inclusions:

- Game Development – Implementation of game logic, rules, simple and clean/ simple web UI
- Networking – Client and Server to connect players using sockets (custom network protocol)
- Documentation – List of documents to help users set up and play the game

Exclusions:

- Long Term Support (LTS) – Not to continue the project after this class
- Advanced Networking – No cross-platform support for mobile devices (Linux, PC, Mac ONLY)
- Advanced Security – User authentication and encryption schemes

Deliverables

Code:

- Client Python Script
- Server Python Script
- Flask Python Script
- HTML Web Page

Documentation:

- Software Bill of Materials (SBOM)
  - Dependencies
  - Libraries
- User Manual
  - About
  - Rules
  - FAQ
  - Installation
- Design Documentation
  - UML

- README (Digital User Manual)
  - About
  - Rules
  - FAQ
  - Install
- Weekly Report
  - Completed/Uncompleted Tasks

Timeline

Key Milestones:

This project will involve 6 (0-5) sprints. Each sprint will be two weeks long (#5 will be three weeks due to fall break)

- Sprint 0: Team formed, tools setup for each member, submit SOW (By: Sept 22)
- Sprint 1: Basic TCP Client/Server built (By: Oct 06)
- Sprint 2: Game messaging protocol implemented and tested, able to manage multi-client connections (By: Oct 20)
- Sprint 3: Multi-player functionality implemented, synchronization of states across clients is successful and tested. (By: Nov 03)
- Sprint 4: Game play and state maintenance implemented and tested (By: Nov 17)
- Sprint 5: Error handling and processing implemented and tested (By: Dec 6)

Task Breakdown:

Tasks will be created and added to the team GitHub page in each sprint. Then tasks will be broken down and distributed via team member skill set and availability. Each team member is expected to contribute 5 hours per week to the project.

1. Team setup/idea generation – 5 days
2. Client base script– 1 Week
3. Server base script– 1 Week
4. Messaging protocol – 10 Days
5. Multiplayer functionality - 10 Days
6. Game logic – 1 Week
7. State maintenance – 1 Week
8. Web UI – 5 Days
9. Error Handling – 5 Days
10. Testing – 2 Days

Technical Requirements

Hardware:

- Server (Static/Reserved IP)
- 4 Clients
- 5 Port Switch (Off-Campus Development)

Software:

- Microsoft VS Code
- Python 3.10+
- HTML 5
- CSS
- Flask, Socket, Threading, Requests, OS (Libraries)
- MacOS (Sonoma), Linux (Debian based), Windows 11

Assumptions

We are assuming that the CS120 lab will have machines available to act as the server for final presentations and submission. As well as that all team members will have access to Visual Studio Code on their personal devices (not relying on the CS Lab Machines). Lastly, we will assume that the University will have a stable internet connection throughout the project's lifecycle.

Roles and Responsibilities

Lead Server Developer – Grayson Morris

- Manage server side of project
- Frame out server code
- Test server code
- Help other developers where needed

Lead Client Developer – Jacob Cole

- Manage client side of project
- Frame out client code
- Test client code
- Help other developers where needed

Lead Web Developer – Callista Robinson

- Manage web and UI side of the project
- Frame out HTML/CSS code
- Frame out Flask routine

- Help other developers where needed

Communication

All communication will be facilitated through a Microsoft Teams channel and GitHub using the Teams Integration app. A brief (2 Sentences +/-) summary of completed and non-completed Tasks will be submitted through a shared Word document in the Teams channel.

Additional Notes

Grayson: In Person

Jacob: Online

Callista: Online