Project Statement of Work (SOW)

Project Title:

A Titillating Turtle Terminal Trivia Game.

Team:

Nelson Wu

Project Objective:

Goal of the project is to teach users about turtle facts but do so in an interactive game. A trivia style game will be presented. Two players connect to a server. Each player answers questions about a turtle fact, whoever gets the most points correct after 10 questions wins.

Scope:

Inclusions:

- The server will contain a local repository of trivia questions and their correct answers.
- The game will require the connection of two clients or users in order to play the game
- 10 questions will be presented to both users. The server will wait until both players respond before proceeding.
- There will be a timeout feature of 30 seconds. If the user has not selected an answer within 30 seconds, that user will not receive any points for that question. The server scores the other responding user and then proceeds to the next question.
- After 10 questions, a winner will be determined by the most number of correct answers.
- A draw will be determined if both users answered the same number of correct answers. Both players will be determined winners.

Exclusions:

- The game will not handle more than 2 players connecting at once.

Deliverables:

- A working game written in python will be delivered
- Proper documentation in GitHub will be delivered for future contributors. A simple schematic diagram will be included in the README to demonstrate the client-server architecture

Timeline:

Key Milestones:

- Set up a TCP Client Server (Oct 6)
- Develop Game Message Protocol, Manage Client connections (Oct 20)
- Multi-player functionality, Synchronize state across clients (Nov 03)
- Game play, Game State (Nov 17)
- Implement Error Handling and Testing (Dec 6)

Task Breakdown:

Sprint 1: TCP Client Server (Sept 22-Oct 06)

- Task 1: Set up a basic Python socket server using TCP (3-4 hours)
- Task 2: Develop a client-side socket connection (3-4 hours)
- Task 3: Implement a simple connection flow between two clients and the server (5-6 hours)
- Task 4: Test connections and ensure reliable data transfer (4-5 hours)
  - Total for Sprint 1: *15-19 hours*

Sprint 2: Develop Game Message Protocol, Manage Client Connections (Oct 06-Oct 20)

- Task 1: Define a custom game message protocol (4-5 hours)
- Task 2: Implement server ability to handle multiple clients (5-6 hours)
- Task 3: Handle player registration, connection management (3-4 hours)
- Task 4: Code basic server responses to game commands (5-6 hours)
  - Total for Sprint 2: *17-21 hours*

Sprint 3: Multi-player Functionality, Synchronize State Across Clients (Oct 20-Nov 03)

- Task 1: Synchronize game state across players (4-5 hours)
- Task 2: Implement trivia question flow (5-6 hours)
- Task 3: Handle timeouts, player turns, connection loss (5-6 hours)
- Task 4: Implement scoring system and end-game state (4-5 hours)
    - Total for Sprint 3: *18-22 hours*

Sprint 4: Game Play, Game State (Nov 03-Nov 17)

- Task 1: Expand game logic to handle full trivia question set (4-5 hours)
- Task 2: Implement full game cycle (5-6 hours)
- Task 3: Provide real-time feedback to players (4-5 hours)
    - Total for Sprint 4: *16-20 hours*

Sprint 5: Implement Error Handling and Testing (Nov 17-Dec 06)

- Task 1: Error handling for disconnections, timeouts (5-6 hours)
- Task 2: Write test cases for game scenarios (5-6 hours)
- Task 3: Test game scalability and performance (4-5 hours)
- Task 4: User acceptance testing (5-6 hours)
    - Total for Sprint 5: *19-23 hours*

Technical Requirements:

Hardware:

Server:

- CPU: 1-2 cores (any basic processor)
- RAM: 512 MB - 1 GB
- Storage: 10-20 MB
- Network: Basic internet connection

Client:

- CPU: Single-core or higher
- RAM: 256 MB - 512 MB
- Storage: 10-20 MB
- Network: Basic internet connection

Software:

Programming Language

- At least Python3: Python 3.x
- Select, socket, and threading may be used and are part of Python's standard library.

Software Tools

- Python IDE: VS Code
- Version Control: GitHub

Operating Systems

- Cross-platform: Windows, Linux, macOS

Assumptions:

The project assumes stable network connectivity for real-time communication between the clients and server, whether over a LAN or the internet, without interference from firewalls or port restrictions. The server will have minimal load, handling only two players at a time, with basic hardware and Python 3.x is assumed available on both the server and clients. The game is designed for two players, taking turns to answer predefined trivia questions stored locally or in a simple database. Development assumes cross-platform compatibility for the Python codebase, using Git for version control and collaboration.

Roles and Responsibilities:

- In this solo project, the individual will assume multiple roles: as the Project Manager, they will plan the project timeline, define milestones, and ensure progress aligns with deadlines. As the Developer, they are responsible for writing the socket programming code, implementing game logic, and managing client-server interactions. In the Tester role, they will create and execute test cases to ensure that the game functions correctly, handle error cases, and debug issues. Additionally, they will handle documentation and ensure that the code is properly version-controlled using tools like Git.

Communication Plan:

- Since this is a solo project, communication and decision-making will be self-managed. Regular updates will be documented in a project log to track progress. Key milestones and decisions will be reviewed weekly to ensure the project is on track, with adjustments made as necessary. This self-discipline ensures accountability and organized progress throughout the project.

Additional Notes:

- None