ENGR302 Requirements and Plan

Team Members: S5 - Alex Wells, Phillip Sparrow, Thomas Green, Bernard Del Puerto, Max McPhee, Oshi Werellagama

Game Name: Agile Adventures

Delivery Date
Requirements and Planning: 26/07/2024
Progress Report: 16/8/2024
Design Review: 04/09/2024

PLAN

Project Objective
Our group will make a multiplayer ‘Among Us’ inspired game where promotion and encouragement of the group to work together is vital to complete tasks. Tasks will be scattered across the map and are themed around software engineering. The game's purpose is to teach first year computer science and cybersecurity students on how to use Kanban boards effectively and as a unit.

Gameplay
There will be a map of the ECS labs where up to five players will be able to freely move between areas to complete tasks. Some tasks will include: fixing pipeline error, load server, make project plan, make AND-OR-XOR Gates, review tasks. These tasks will be positioned on a Kanban board, where players will need to observe and interact with it to both find what tasks are to be done and what tasks are already completed.

Risks and Mitigations

1. Risk: Scope Creep
    - Description: Uncontrolled changes or continuous growth in a project's scope.
    - Mitigation: Establish a clear project scope at the beginning and use a change control process for any modifications. Regularly review scope with the team to ensure alignment.
2. Risk: Missed Deadlines
    - Description: Delays in project milestones and deliverables.
    - Mitigation: Set realistic deadlines, break down tasks into manageable parts, use the SMART system for goals and use project management tools to monitor progress. Have regular meetings to identify and address any potential delays early.
3. Risk: Communication Breakdown
    - Description: Lack of effective communication among team members leading to misunderstandings.
    - Mitigation: Fostering an open communication culture amongst the team. Use

discord group chat and comments on GitLab issues to keep us collaborating, schedule regular meetings and ensure everyone is on the same page.

4. Risk: Quality Issues
    - Description: Deliverables not meeting the required quality standards.
    - Mitigation: Implement a quality assurance process, conduct regular code reviews, and perform thorough testing. Use feedback loops to continuously improve quality.

Next Steps (due 09/08/24)

1. Now that the requirements have been formalized below, we can create issues on GitLab to start assigning the 'must have' tasks with timeframes for each requirement specified.
2. Create sprite and map design for our game.
3. Setup server and client scripts so we can start working on web-based game dev.
4. Setup base javascript code with phaser framework to represent our webgame on localhost. This would be deployable from the server and client scripts.

REQUIREMENTS

Functional Requirements

Must Have (Non-negotiable product needs that are mandatory for the team)

Game will be designed to run on a local connection (localhost), meaning that all game operations, communications, and data exchanges occur within a local network without the need for an external server. This involves setting up the game so it can be launched and played by users within a local network, promoting quick iterations and immediate feedback during development. The game should be programmed with the intention to be integrated with a server in future development (the code architecture and data handling should be structured to support this).
The game is compatible with major web operating systems including; Safari, Chrome, Firefox, Linux, Windows, Mac.
The game is designed to be multiplayer. This promotes collaboration and interaction, aligning with the goal of teaching Agile engineering principles through team-based tasks. To start a game session, a minimum of 5 players is required. This requirement encourages players to gather a team before initiating a game, reinforcing the team-building aspect.
Central to the game is a Kanban board that visually represents the workflow. This board will be identical and synchronized for all players in a game session, ensuring everyone sees the same tasks and task status in real time. The board has four columns representing different workflow stages: TODO (tasks to start), ONGOING (tasks in progress), COMPLETED (finished tasks), and REVIEW (tasks under review). This setup teaches players how tasks progress through the Kanban process.

Players navigate the game world using the keys ‘W’, ‘A’, ‘S’ and ‘D’ to move up, down, left and right respectively. Players start tasks by pressing 'T' when near a task. The Kanban board is accessed with 'B', allowing easy task monitoring and management. Players can use their mouse to click on tasks, drag them to different columns on the Kanban board, and interact with various game elements.

Should Have (Important initiatives that are not vital, but add significant value)

The game architecture will support up to 50 clients simultaneously. This means that up to 10 game sessions can run concurrently, each with 5 players. This capability ensures that multiple teams can play the game at the same time without performance degradation.
The game’s code and server will be designed to handle multiple game rooms efficiently. This includes managing ‘network traffic’, synchronizing game states across players, and ensuring smooth gameplay even with a high number of concurrent users. Optimization will be part of the development process to ensure that having up to 50 clients does not negatively impact performance. The target frame rate of 60 fps (minimum of 30 fps) must be maintained across all game sessions.
Players will interact with Kanban board to manage tasks. They can move tasks between columns to reflect their progress. For example, moving a task from TODO to ONGOING when they start working on it. Since Kanban board is shared among all players in a this will teach players to coordinate, prioritize and progress simulating a real-world Agile environment.
The game is suitable for players aged 12 and up, ensuring content, language and complexity are appropriate and engaging for both younger and older players.

Could Have (Nice to have initiatives that will have a small impact if left out)

Players can customize key bindings, such as switching to arrow keys for movement, ensuring accessibility for different preferences. A proximity detection system highlights tasks when players are close, helping identify interactive objects. On-screen buttons provide intuitive access to the Kanban board and other functions, especially for players who prefer mouse-based interactions. The settings menu allows players to reassign keys for movement, task interaction, and other actions.
Game has an implemented scoring system which tracks each player's performance within a game session. Points can be awarded based on the quickest time to complete tasks or the entire game session. This encourages players to work quickly and effectively. Scoring is team-based, with all players on a team receiving the same score if they collectively complete tasks, promoting collaboration. In scenarios where players can ‘die’ or be removed from the game, the score can be based on the performance of surviving players. This adds a layer of challenge and strategy to the game.

The UI is designed to be simple and user-friendly minimizing complexity to ensure that it is accessible to younger players. Clear icons, straightforward menus and intuitive navigation are key aspects.
Game includes helper functions such as tips, hints and contextual assistance to guide players through various game mechanics and tasks. These functions help new players understand how to play the game without feeling overwhelmed.
Implement a ‘How to Play’ section which will be included in the main menu. This section provides a detailed overview of the game’s objectives, controls, and mechanics through instructions.

Will Not Have (Initiatives that are not priority for this specific time frame)

Possible verbal help and other accessibility elements can be added to players with visual and hearing disabilities and can be added at later stages.
The game will monitor and record each client's connection status to manage active players and detect disconnections. It will also store client progress, including tasks completed, current tasks, and overall game state. Logs of previous sessions, scores, and actions taken will be stored for review and analysis.
All client information will be private and accessible only to the respective client. Proper data encryption and secure access controls will protect personal information. Clients can delete their stored information, clear cached data, and permanently remove personal info through a user-friendly data management interface.

Non-Functional Requirements

Must Have (Non-negotiable product needs that are mandatory for the team)

Game will be coded/implemented using Python 3.12.3 to develop the server and client (backend) components. This will be used to boot up the server to host the game and the clients who will be the game users. The actual game will be coded using Javascript (ECMAScript 2024) using Phaser.js framework.
Team will use VSCode as the developer tool. Will make use of Git and GitLab to store, track and document project. Use of GitLab components will help create a timeline of our project through the use of issues, branches, and review of work.
Game is presented with a top-down view of the map. This view allows players to see their surroundings, navigate the environment, and locate tasks easily. Map is designed to include various locations where tasks are placed, encouraging exploration and strategic movement. Map is intuitive.

Should Have (Important initiatives that are not vital, but add significant value)

Each player is represented by a unique sprite, allowing for easy identification. Objects located on the map represent tasks as an interactive object within the game environment, making it visually engaging and easy to recognize.
Implementation of safety cases to handle failures in code. Exception handling should be implemented so our system remains stable and ensures that our game is kept to a high standard and positive public experience.
User interface must be intuitive and easy to navigate for new users.

Could Have (Nice to have initiatives that will have a small impact if left out)

Each player will have a unique identifier, which can be their real name or chosen game ID. This helps personalize the experience and track individual progress within a game. Game will record the score each client achieves in a session. This can be used for individual performance tracking and comparison across sessions.
Have helper sections and tutorials at the start of the game for explaining what a Kanban board is, what it is used for, and how it is important. Have tutorials on what the tasks in the game represent in terms of software engineering and why they are prevalent.
Ensure that the game is not prone to web-based cybersecurity attacks. For example, input validation could be implemented to prevent execution of code initiating a XSS attack.
Implement CI/CD pipelines in GitLab to build, test and format the code.

Will Not Have (Initiatives that are not priority for this specific time frame)

When running on the web, when resizing the window then the web-browser will react to this and refit the game.