Group S5 ENGR302 Progress Report

Report Outline

Introduction

- **Section One: Motivation** -> Provides background information that illuminates the “big picture” and relevance of the project.
- **Section Two: Problem Statement** -> Problem has been identified and stated clearly with concrete evidence to demonstrate its existence

Background Research

- **Section Three: Literature Review** -> State-of-the-art and existing solutions to the problem, including their advantages and disadvantages. Concepts and other theoretical underpinnings of the problem/solution.
- **Section Four: Tools and Methods** -> Identifies relevant programming languages, hardware and/or software libraries, frameworks, development kits and tools that are being used in the development. Discuss how these tools will benefit the development process. Development process has been selected with relevant justifications for the choice of the methodology.

Development Progress

- **Section Five: Requirements** -> Requirements specification covering all aspects of the system that need to be developed.
- **Section Six: Design** -> Explanation of the system architecture, along with its components and interfaces with external systems. Requirements and constraints are used appropriately to drive design choices. Tests have been made to ensure the design is feasible. Prototype has been made.

References

- **Section Seven: References** -> References used throughout to backup our discussions and statements made for Motivation, Problem Statement, Literature Review, Tools and Methods.

Members

Philip Sparrow - sparrophil
Thomas Green - greenthom
Bernard Del Puerto - delpubern
Max McPhee - mcpheemax
Alexander Wells - wellsalex1
Oshi Werellagama - werellosha

Section One

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.

There will be an ECS lab map 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.

Section Two

Agile development and principles are incredibly important to software engineering. Agile development aids to *“ensure that development teams complete projects on time and within budget”* [1]. Furthermore it can *“help reduce the risks associated with complex projects”* [1]. Kanban boards, which are a type of agile development, are highly beneficial. This is evidenced by the fact that *“the visual representation of the work makes it easy to see what the development team needs to do and who is responsible for each task”* [1], furthermore the kanban board can *“help to reduce confusion and conflict within the development team”* [1].

Thus it is clear that knowledge of agile development, particularly kanban boards are essential to the career of a software engineer, digital engineer, etc. The BE(Hons) program claims that *“Our innovative and flexible programmes will prepare you for a successful career as a digital engineer”* [2]. The foundation of this program starts from the first year. A **problem identified** by our team is that we did not feel we were adequately educated on agile development during our first year despite it being such a crucial concept. The concept, if mentioned, was described in a way that was beyond our grasp at the first year level. This is evidenced by the fact that we still struggled to correctly explain certain agile principles beyond the first year. We all strongly feel that had this concept been taught in a more fun, straightforward, interactive way the concepts would’ve ‘clicked’ earlier and become ingrained in our memory and practice. Thus to rectify this problem for future students our team has decided to incorporate agile development principles, in particular kanban boards into a fun, educational multiplayer game.

Section Three

As explained above, our team identified a problem with the way agile concepts were taught and tasked ourselves with creating a game to teach the kanban methodology in a fun, straightforward, educational way. Instead of starting completely from scratch, our team investigated existing solutions to the problem. Our intention was to identify and incorporate the advantages of these existing solutions into our own solution, while trying to eliminate disadvantages we identified in these solutions.

Our game takes inspiration from the existing game ‘Among Us’. Among Us has a strong emphasis on communication as evidenced in [3]. Among Us has *“communication skills automatically built and refined throughout gameplay”* [3], this is a quality that we want to achieve within our own game. A disadvantage of ‘Among Us’ in the context of what our team wants to achieve is that it does focus on the elimination of ‘crewmates’. This individualistic quality is something we do not want to incorporate into our own game, instead focusing on a more collaborative nature.

An existing game that intends to teach players about kanban methodology is ‘Kanban Pizza’. Kanban Pizza is not web based, rather it is used as an interactive in-person game to teach agile principles in a fun way. In Kanban Pizza various team members work to assemble a mock pizza. The catch of Kanban Pizza is that an ‘inspection’ can occur at any point. During ‘inspection’ unused pizza ingredients result in deduction of points while fully formed pizzas result in addition of points [4]. Kanban Pizza has a focus on teaching the theory of constraints [4], this is a methodology for “*identifying the most important limiting factor (i.e., constraint) that stands in the way of achieving a goal and then systematically improving that constraint until it is no longer the limiting factor”* [5]. In our game we could incorporate the teaching of the theory of constraints, through this inspection or review aspect. An obvious limiting factor of the existing kanban pizza game is that it is not web based, hence we would have to identify how best to adapt the advantages of this existing game into our desired format.

Another game that aims to teach agile methodologies in a fun way is retropoly. This is a game similar to the traditional board game monopoly, with 5 steps - preparing, rewarding, remembering, debating and closing. Essentially the team will carry out a task, then discuss their performance of said task, debating and discussing ways to improve. The aim is *“at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly”* [6] which is the 12th principle of agile. In regards to disadvantages retropoly is primarily focused on teaching the scrum methodology, but the game developers state it is easily adaptable to teach other specific methodologies. Since our team is focused on teaching the kanban methodology, we would have to adapt accordingly. Another disadvantage is that Retropoly is not web-based, so we need to explore ways to incorporate advantageous game elements that are suitable for our web-based implementation. The team oriented aspect of retropoly is something we would like to adapt into our own implementation (as opposed to having members compete with one another). We feel that focusing on a team is a more suitable way to educate students on agile as opposed to a competition. Retropoly having reflective

intervals is also something worthwhile to adapt into our own game. It would be educational to show players how performance improves after reflecting upon prior performances.

In addition to games that teach various agile methodologies another solution to educate students would be through existing Kanban tools such as Trello [7] and Jira [8]. These are tools commonly used in industry to manage tasks and workflows. Advantages of using existing digital tools to educate first year students is that they would be better prepared for professional work. However the disadvantage is that since these tools are used within industry they will be an overwhelming introduction to agile methodology for many first years. Tools such as Trello and Jira have an extensive range of features, which is a bit complex for a user who does not yet understand agile principles. Furthermore, learning by using existing industry tools, although highly educational, will not be very fun. Thus we can incorporate some of the advantageous kanban features used within Trello and Jira, such as the way the kanban board is visualised in these tools. However we should aim to simplify such that our game remains interactive and engaging.

Section Four

We are using GitLab to enable all the team members to collaborate in every phase of the project. GitLab offers tracking from planning to creation which will provide an outlined lifecycle to achieve our project objective. We have chosen this tool as it provides a *"comprehensive set of features, including issue tracking, code review, wiki, and more, all integrated into a single platform"* [9]. As evidenced in [9] GitLab *"simplifies collaboration and reduces the need for multiple tools, enhancing productivity"*. Furthermore prior courses in the BE Hons programme have taught us to use GitLab, hence we are choosing a tool that we feel most familiar with.

We have chosen to use the programming language JavaScript over other options such as Python as JavaScript *"works directly in browsers"*, whereas Python is "*good for making games that you download and play on your computer*" [10]. JavaScript was the best choice for where we want the game to be played. Furthermore JavaScript is *"great for simpler 2D and mobile games"* [10], compared to a language such as C++ which "*might be better for a big 3D game"* [10] . Furthermore JavaScript has many *"tools and helpers, like Phaser or Three.js, that make making games easier"* [10]. Having these tool options was another reason JavaScript was chosen.

Javascript software libraries that we are using Frameworks runtime Node v20.16.0. Node.js and npm v10.8.1 modules that we are using are;

- concurrently 8.2.2 -> execute numerous tasks simultaneously, all while avoiding the necessity of generating new threads for each task,
- cors 2.8.5 -> used to share resources across different servers,
- express 4.19.2 -> to build a single page, multipage, and hybrid web application,
- phase 3.80.1 -> an open source html5 game framework used to create 2d games,
- socket.io 4.7.5 -> TCP connections to use open connections for real time communication,

- vite 5.4.0 -> provides out of the box support for common web patterns.

For our development process our team has chosen to take an iterative approach. This is an approach taught in ENGR301 [11], thus a reason for choosing this is that all members of our team are familiar with this methodology. Another reason for choosing this is that *"many engineering teams use the iterative process to develop new features, implement bug fixes, or A/B test new strategies"* [12] thus it is a good preparation for when we eventually work within industry. Furthermore, the benefits of an iterative approach over a waterfall process is that there is *"increased efficiency, increased collaboration and increased adaptability"* [12]. These are the reasons we chose the iterative approach as our development methodology.

Section Five

MosCow Prioritization

Must Have: Initiatives that must not lack in our product.
Should Have: Essential initiatives but not vital in our product.
Could Have: Initiatives that are nice to have in our product.
Won't Have: Initiatives that are not the priorities in our product

Requirements

Functional Requirements	
MUST HAVE	
Functional	- The game must be designed to run on a local connection (localhost). Local network users can play the game. - The game must be multiplayer (2- 6 players) - The game must have a kanban board that visually represents the workflow/tasks to be carried out. - The kanban board is shared between all members, updating 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 to review). - Players must be able to navigate the game with movement keys. - The kanban board must be accessible by each player. - The game displays a map.
SHOULD HAVE	
Functional	- The server should handle multiple game rooms efficiently. - The game should be suitable for any audience.

	- Should have on-screen buttons that provide intuitive access to the Kanban board and other functions, especially for players who prefer mouse-based interactions. - Resizing the window or the web-browser should dynamically resize the game. - The game should have various tasks for players to perform. - A player should be able to select their own sprite / character. - Tasks are positioned at points on the map. - The game should have a main menu, where players can join a game, How to Play (explaining game), create game settings. - The game should have 5 playable tasks. Users will need to complete all of them to end the game.
COULD HAVE	
Functional	- Game will have settings that will include; customise key bindings, turn off audio, turn off sound effects. - The game could have a proximity detection system for identifying tasks. - The game could have an implemented scoring system based on team performance. - Game could have a ‘How to Play’ in the settings, which will provide a detailed overview of the game’s objectives and controls. - Should have a button menu for saving and exiting the game state. - The game could have sound effects depending on player/task actions. - The game could have music which changes depending on the game state. - The game could have sprite movements depending on the direction of movement.
WON’T HAVE	
Functional	- The game will monitor and record each client's connection status to manage active players and detect disconnections. - It will store user progress and settings on the game. - All client information will be private and accessible only to the respective client.

Non-Functional Requirements	
MUST HAVE	
Non-functional	- The must be game is compatible with major web operating systems including; Safari, Chrome, Firefox, Linux, Windows, Mac. - The must game promotes collaboration and interaction. - The game must be fun. (expand, fun for who, how)

	- The game must teach the user about agile software engineering, specifically kanban boards.
SHOULD HAVE	
Non-functional	- Players should gather a team before initiating a game, reinforcing the team-building aspect. - The game should handle unexpected errors without crashing.
COULD HAVE	
Non-functional	- No matter how many users are playing, the game performance should not be impacted. - The game should feel smooth, maintaining a minimum frame rate.
WON'T HAVE	
Non-functional	- The game is not prone to web-based cybersecurity attacks. - The game should support multiple languages. - Add accessibility elements for players with verbal and visual impairments.

Section Six

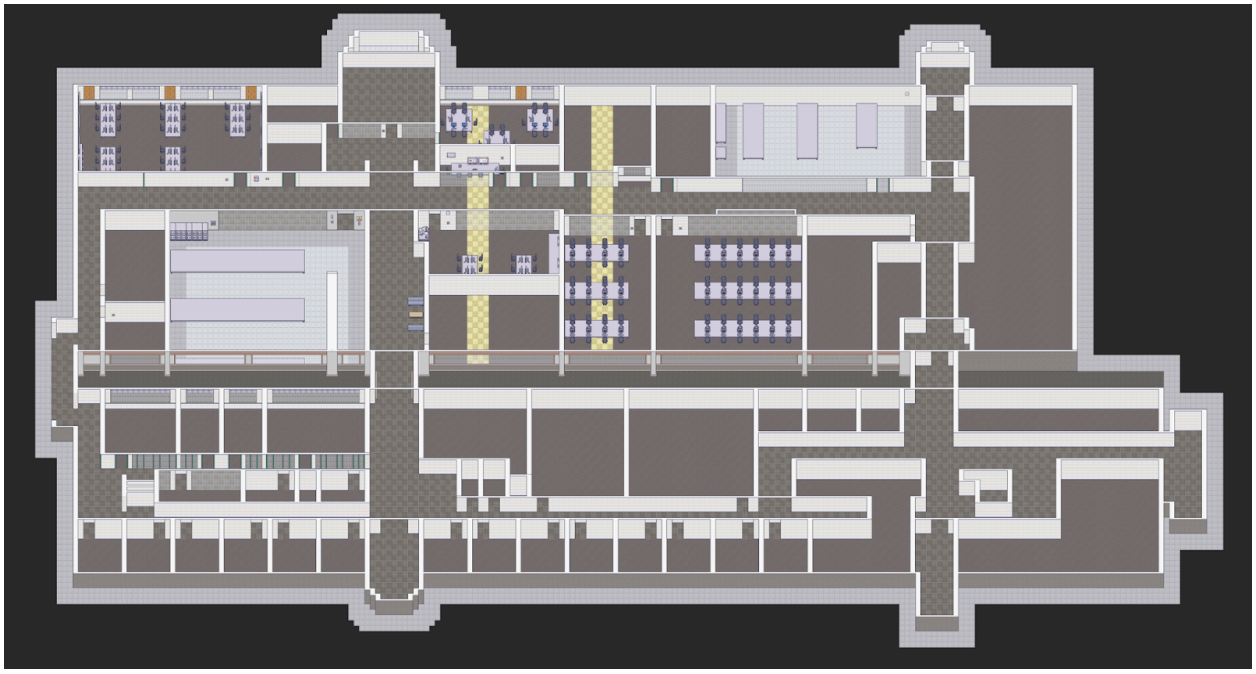
Design Outline
All game design functions, components, layout. Separated in respective areas for implementation for context on what needs to be done for each component.

Movement and Controls	Players are able to move on the map using keys; 'W'(Up), 'A'(Left), 'D'(Right), 'S'(Down).
	Players can start tasks by pressing 'T' when near a task.
	Settings are available by pressing 'tab' or 'EXC' key?
	The KanBan board is accessed by pressing 'B' or pressing a button.
KanBan Board	Interacting with the KanBan board is done with the mouse to click on tasks, drag them to different columns.
	To view the Kanban board, the user presses 'B' or selects the Kanban button with the mouse.
Tasks	Have proximity detection on tasks, where the tasks will highlight when players are near enough to interact.
	Interaction with the tasks themselves will involve mouse, keyboard interaction respective to the different tasks.

	Tasks when accessed will be presented as a pop-up on the game screen.
	Players can exit the task if they do not wish to carry it out.
	Once task has been completed, it will present a ‘Task Completed’ across the screen
	Last task, all players have to be present together in a room, in order to finish playthrough
	Tasks will include: fixing pipeline error, load server, make project plan, make AND-OR-XOR Gates, review tasks. These will be scattered at random in different rooms across the board.
Actual Tasks	Fixing Pipeline Error: Presented with a full-screen pop-up. Displays a failed pipeline. User presses the “run again” button to re-run the pipeline. It has a 1 in 5 chance of passing the pipeline. Once passed then the task is complete.
	Load Server: Presented with a full-screen popup. Presented with a failed server running on a Windows Xp. Have to click (i.e. spam the spacebar) to cycle through warnings/errors to increase the progress bar until the server is efficiently running. Once efficiently running the task is complete.
	Project Plan: Presented with a full-screen popup. Presented with blocks that identify the phases of a project lifecycle. User needs to reorder the blocks to present a “clear project structure/outline”. Once ordered correctly, the task is complete.
	AND-OR-XOR Gates: Presented with a full-screen popup. Users are presented with a logic gate output. Users must drag the correct gate (presented as drag and drop blocks) into the logic gate to make that output work. Once correct selection is made the task is complete.
	Add stories in sprints: Presented with a full-screen popup. Will be presented with an empty box called “Sprint”, and boxes with stories. It will involve players dragging stories into the sprint. Once the sprint is complete, the task is complete
	Coding: Presented with a full-screen popup. Will be presented with an empty class. Players will need to input (spam) keys on the keyboard until the class is complete and ‘compiles’ properly. Once the class has compiled correctly, the task is complete.
	Talk to Lecturer Task: Presented with a full-screen popup.

	This will involve the players to go find the teacher/stakeholder and interact with them to complete the tasks above.
Main menu	Have a button in the main menu for creating a game. When a user clicks it will create a new game session which will be publicly seen by players in the main menu. When the player creates the game they will be given a pop-up where they can provide what they would want to call their player.
	Have a button in the main menu to join a room. When the user clicks it will give them a list of joinable rooms for them to go into and participate with the current members in that room. Before the player enters the lobby they will be presented with a pop-up where they can provide what they would want to call their player.
	Have a button in the main menu to open settings. This will have the options for the user too; customise their key bindings, and turn off or up sfx and game music.
	Customising key buttons will enable the user to input which key/s they want to rebind for the specific game action.
	Turning the music up or off will involve a slide bar for the users to interact with for their personal preference.
	Have a button in the main menu, for how to play. This will present a pop-up which will have different sections for how players are supposed to move and navigate on the board, how to carry out tasks, what the game is teaching them, and how to work together as a team effectively.
Lobby (Before starting the game)	When the user enters the lobby they will be presented as a default sprite. The user can change their skin/character sprite from a specified selection by pressing left or right arrow keys beside their character sprite.
	The user who created the game room/code can start the game when their friends/members have all joined. When all members have joined this person will press the ‘Start Game’ button. This user cannot press when there is less than 2 people in a room
	The user can leave the game room by pressing the ‘leave room’ button
	Each user in the lobby will be able to move and interact with each other before play begins.

Map Design:



Prototype Goals

Prototype objectives addressed in early stages of requirements and design planning. Prototype goals outline key functionalities of initial game server and client setup that were fundamental to achieving how our game is meant to be set up, executed, and played.

Stage One Goals	1. Create game that can be seen by other users
	2. Join same room
	3. Sent object to server corresponding game room (GR)
	4. Server acts accordingly to received object
	5. Host multiple game room on localhost
	6. Send simultaneous inputs to multiple GR's
	7. Check that GR's act
	8. Check that GR's act accordingly without error
Stage Two Goals	9. Disconnect Client
	10. Client joins back with the same GR and player info
	11. Store KB board

	12. Manipulate and delete KB board
	13. Send KB board to users
	14. Relay inputs to multiple users in GR

Prototype Video
Unlisted Youtube video showing prototype achieved. Successfully achieves all stage one goals outlined in prototype goals table above.
https://www.youtube.com/watch?v=u229B4_o7UY

Prototype Evaluation
Successfully achieved all stage one goals in the prototype created. This included creating a game room that can be seen by others, where one user can create a game room with a user provided name for the room. This will load that user into their respective room (0.00-0.05). Once in the game room the user will be able to move around on the map in and out of different game rooms (0.06-0.16). If a game room is active, this will enable another player to observe that respective game, through pressing join in the main menu. This will present a pop-up which will show the given user, the available game rooms that the server is actively hosting currently. The user will be able to choose this room where they will get taken to that specific game room (0.16-0.19). From here that user will be able to move around on the map in and out of different game rooms (0.19-0.28). As seen in the video, both users that are joined to the game room, can effectively move around, which is synchronised/presented on both of the browsers running the game(0.28-0.37). Creation of our prototype enabled us to meet the goals and objectives laid out in our initial requirements and analysis, which were; efficient creation of game rooms, playable map with player movement through the use of sprite animation, efficient server and client connections run through the use of sockets, and a visual display of the game that can be further developed with stated game logic and components, that can be tested efficiently and easily.

Due to time constraints and external workload we could not achieve stage two goals which mostly revolved around our game functionality and objective components. This has now been placed in our next steps for the project.

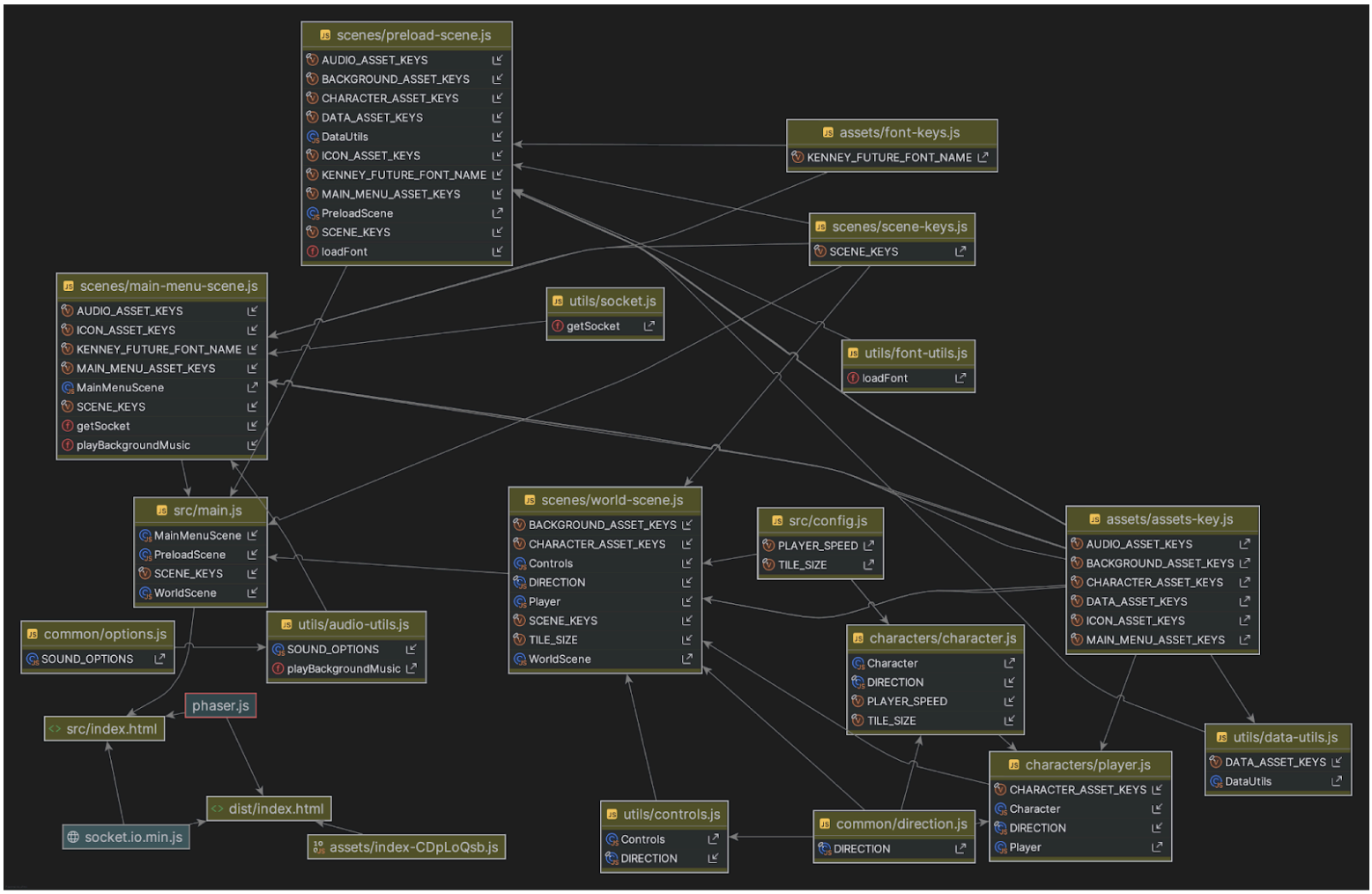
CRC Cards

Highlights class, responsibilities and collaborators for planned, implemented and future classes for the game. Responsibility highlights on what each class/component will handle in respect to game functionality. Collaborator highlights what the respective class will be using/interacting with to achieve functionality.

Class	Responsibility	Collaborator
server	- Handle Time - Store And Handles KanBan Board - Send Task Status To Clients - Handles Game Rooms - Handles Update Function	- KanBan Board
kanbanBoard	- Stores To Do Tasks - Stores Doing Tasks - Stores QA Tasks - Stores Done Tasks - Handle Changes of Task status	
character	- Stores current Phaser Scene - Stores Asset Key - Stores Player Position - Stores Asset Frame - Handles is Moving - Stores and Handle changes of current direction	- Config - Direction
player <extends character>	- Handles player movement and idle animations	- Character asset keys
font-utils	- Handles loading font from file	
data-utils	- Handles getting the animation config from json file	
audio-utils	- Handles sound effects - Handles background music	
controls	- Handles player input (i.e. movement) - Handles locking player movement	- Direction
direction	- Handles player directions enum (i.e. UP, DOWN, etc.)	

direction	- Handles player directions enum (i.e. UP, DOWN, etc.)	
options	- Stores users settings	
asset-keys	- Acts as a getter for keys of loaded assets	
font-keys	- Acts as a getter for keys of loaded fonts	
scene-keys	- Acts as a getter for keys of loaded scenes	
main-menu-scene	- Handles scene for the Main Menu Frame - Handles updating the room (current rooms active) - Handles creating game rooms	- Asset keys - Scene keys - Font keys - Audio-utils - options - socket
preload-scene.js	- Handles loading the assets into cache - Handles creating animations from a JSON file and storing into cache - Starts main menu scene	- Scene keys - Asset keys - Font keys - Font-utils - data-utils
scene-keys.js	- Handles scene keys (for client to access)	
world-scene.js	- Handles presenting the main game - Handles sending updates to the server - Handles player disconnecting from room - Handle task positioning on the map - Knows the current room of the client	- Kanban Board - Scene keys - Asset keys - Player - Controls - Directions - config
kanbanboard.js	- Handles request and receiving of the data from the server - Handles update of the KanBan board - Handles recursing current KanBan board status for client rooms	- Task handler
task-handler.js	- Generic class to handle tasks - Calls specific class that handles task - Hold current state of tasks	

UML diagram



Next Steps

Outlines the main next steps to be implemented carrying on from prototype implementation.

Due	Description
23/08	Disconnect Client from pause menu
30/08	Client joins back with the same game room and player info
30/08	Store KanBan board
06/09	Manipulate and delete KanBan board
06/09	Store KanBan board to users
06/09	Relay inputs to multiple users in game room

06/09	Implement collision logic
13/09	Complete game logic
13/09	Start on the task games and pop-ups
27/09	Implement Time Logic and Leaderboard
27/09	Implement Menu Logic

Section Seven

[1] "What is Agile Development," OpenText. [Online]. Available: https://www.opentext.com/what-is/agile-development#:~:text=Agile%20development%20is%20important%20because,team%20and%20the%20product%20owner.

[2] "School of Engineering and Computer Science," Victoria University of Wellington. [Online]. Available: https://www.wgtn.ac.nz/ecs.

[3] M. De, "Among Us skills for the real world," Medium. [Online]. Available: https://medium.com/@mohnishde/among-us-skills-for-the-real-world-0d617b3e70c1#:~:text=Every%20game%20will%20teach%20players,the%20meeting%20during%20each%20round.

[4] E. El-Badry, "Kanban Pizza: the Game that Keeps Giving," LinkedIn. [Online]. Available: https://www.linkedin.com/pulse/kanban-pizza-game-keeps-giving-ehab-el-badry/.

[5] "Theory of Constraints," Lean Production. [Online]. Available: https://www.leanproduction.com/theory-of-constraints/#:~:text=The%20Theory%20of%20Constraints%20is,referred%20to%20as%20a%20bottleneck .

[6] "Retropoly game instructions," Agile Practice. [Online]. Available: https://www.agilepractice.ro/wp-content/uploads/2017/04/Retropoly-game-instructions.pdf

[7] "Kanban Template," Trello. [Online]. Available: https://trello.com/templates/engineering/kanban-template-LGHXvZNL.

[8] "Kanban Boards," Atlassian Jira. [Online]. Available: https://www.atlassian.com/software/jira/features/kanban-boards.

[9] "GitHub vs GitLab: Why should I choose GitLab?", GitLab Forum, Aug. 2023. [Online]. Available: https://forum.gitlab.com/t/github-vs-gitlab-why-should-i-choose-gitlab/93945/6.

[10] D. Deven, "Introduction to Game Development with JavaScript," daily.dev, Jun. 16, 2023. [Online]. Available: https://daily.dev/blog/introduction-to-game-development-with-javascript#:~:text=If%20you're%20making%20a,your%20game%20will%20be%20played

[11] “ENGR 301: Methodology,” Victoria University of Wellington, 2024. [Online]. Available: https://ecs.wgtn.ac.nz/foswiki/pub/Courses/ENGR301_2024T1/LectureSchedule/Methodology.pdf

[12] "What Is the Iterative Process? (Definition and Guide)," Asana, 2023. [Online]. Available: https://asana.com/resources/iterative-process.