Interim Report – Team 18

Project Title: “Codenames”

Project Sponsor: Colin Johnson (University of Nottingham)

Academic Sponsor: Colin Johnson

Contributors:
Alexandru Stoica [psyas13, 20274825]
Daniel Robinson [psydr2, 20220149]
Hongjia Xue [scyhx5, 20216915]
Shahil Pramodkumar [psysp7, 20270365]
Ing Sam Yin [hfysi2, 20189487]
Ao Li [scyla3, 20217306]
Tianxiang Song [scyts1, 20217424]

Introduction

This Interim Report will document the things that our team has done so far to achieve our goal of making a game with AI players. This report will brief the reader on what we have done and why we decided to do it that way, as well as the reflections on our methods and what we can do better in the future. This report will clarify in detail the framework that we plan on using, as well as our proposed classes and methods that will hopefully be fully realized and implemented into our game. We hope that the reader will find this report useful to gain insight on what we have decided to do.

Boardgame Background

First and foremost, before starting this project, we, as a team, had several discussions over a few days regarding our choice of project and what it entailed doing. We read the project brief given to us and made sure to iron out the details of what our primary objective was, and still is.

Based on the initial brief given, we understand that one of our main objectives is to create an online version of the board game “Codenames”. This version of the board game must allow people to play the game with/against each other, either in the same space or remotely. Our second main objective is to implement artificial intelligence (AI) players that can play with/against human players as well.

Codenames is a game whereby two teams with one spymaster each try to get all their words guessed first. The spymasters who can see the colours of all the cards must provide one word and one number that their teammates must use to guess which cards are to be chosen. A randomly chosen bomb card

instantly causes the team that picked it to lose. The skill needed as a spymaster is to correlate the different words that are of their team's colour and to provide a single word that can be assumed to relate to those words, whereas the spies must deduce what words are meant to be guessed based on the clues given by their spymaster.

Of course, as a team, we also had some ideas on how the game could be improved. On an online platform, there are many more things that can be done compared to an actual physical board game. One of the prime examples is the setup time, which takes far longer to do in-person than what could be done on an online platform. Taking advantage of the digital status of our project, we made our game extremely accessible and easy to play from a new player's perspective.

We included an observer mode, whereby AI players duke it out while human players watch the gameplay. We strongly believe that this mode can help newcomers understand how to play far faster than reading from a rulebook.

In addition, we also made the game significantly easier to play for the eyesight impaired by letting players adjust the font size for their instance of the game. We also added an option to adjust the colours of the game to account for players who are colour-blind. These options are completely local and do not affect the other players in any way yet help improve the quality of gameplay for the players who need these options.

Music and sound effects will also be added to help players feel the impact of their actions, such as soft clicks when the cursor passes over a card or when tapped. These sound options can be turned off or made softer at the players' discretion via the settings menu.

We aim to create a game that can be enjoyed by everybody. We realize that there already exist online versions of the game. However, we believe that we can take inspiration from these existing games and improve our own game by seeing what the others lack. For example, our team had the idea to add a hint system that made use of natural language processing to help struggling players by giving them additional words that relate to the cards on the game board. Our game will also differ from existing products through the usage of AI and AI players.

Requirements and Specifications Analysis

Before officially taking this project, we analysed basic requirements and applicable methods. We thought that three main components are included in the scope:

- The game mechanics
- Web development for online play
- AI players

In the phase of Requirements Elicitation, we made User Stories, Persona Boards, and a Use Case Diagram (details available in **appendix A**). The User Stories are written at the beginning to help us produce initial requirements. The Persona Board and Use Case Diagram were written afterwards by two sub teams to identify possible user types and their requirements.

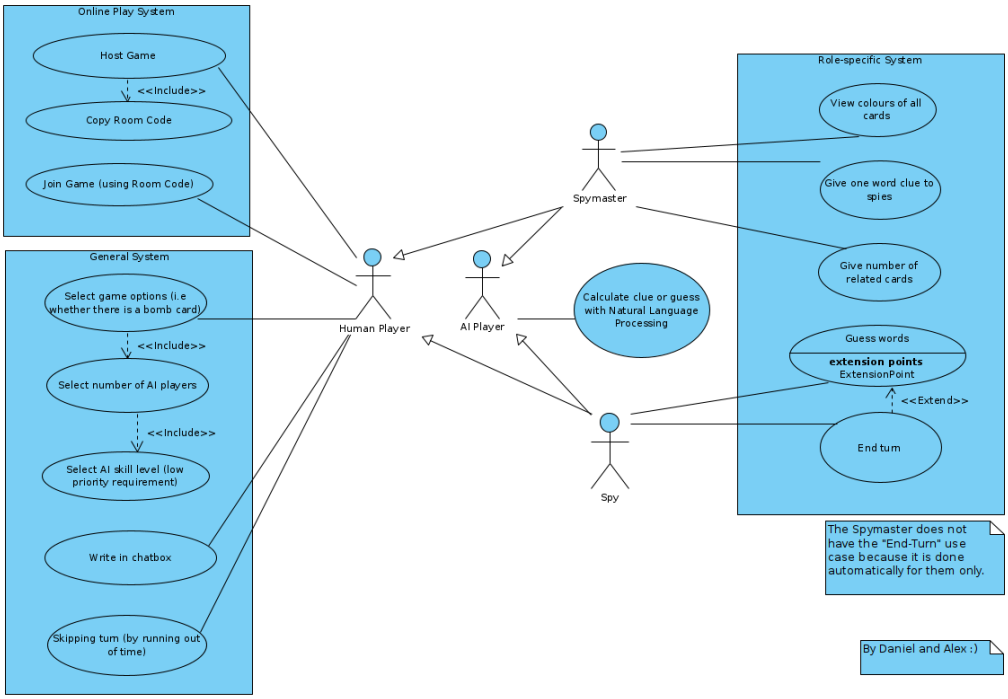

Figure.1 Use Case Diagram

Following these steps, we concluded a variety of requirements (details available in **appendix A**). The game rules have been deconstructed into the first 12 functional requirements. The further requirements have been selected to meet the demands brought by moving the board game onto the computer. Low priority requirements are also included. These are to be completed if there is sufficient time during project development.

These requirements were changed after discussion, and the controversy is primarily about whether to store user data to achieve following functions:

- Registering and storing player accounts
- Online matchmaking (playing with same-level strangers)
- Making online friends
- Leader boards
- Game record

As this project is relatively small, for now we decided not to put it on large software platforms such as Google Play or Apple Store, thus there might not be a great demand on synchronizing game record or supporting vast number of players matching their opponents at the same time. We preferred to make an instant game that hosts quickly with friends or fill the place with AI if human players are not enough.

Therefore, the requirements regarding user data are reduced in the scope. Apart from the three main components mentioned before, the scope also includes:

- Some customization options
- PC and Mobile compatibility

As this is a Traditional-Agile hybrid project, it may be possible to extend the scope, though this may only be done when the sponsor has agreed and when all requirements are completed.

In the phase of Requirements Validation, we had an Internal Review to make some updates and clarifications on requirements. We also have Focus Groups to make sure that our implementation of the game was correct, and we have not missed any steps or rules of the game. The results are available **in appendix A.**

Moreover, we have done an Activity Diagram, Sequence Diagram and State Diagram for Requirements Modelling and Specification Visualizing (details available **in appendix A&B**). The Specifications have been thoroughly gathered from these requirements to make sure the game is in line with the client's requirements.

The game can be split up into three sections for development, so that each area is encapsulated and can be worked on independently if needed. These areas are *web development, game mechanics and AI development*, and for each area there are several specifications. We also made some low fidelity prototypes to test the consistency of specifications and present how all specifications work together. (Specifications and prototypes available in **appendix B**).

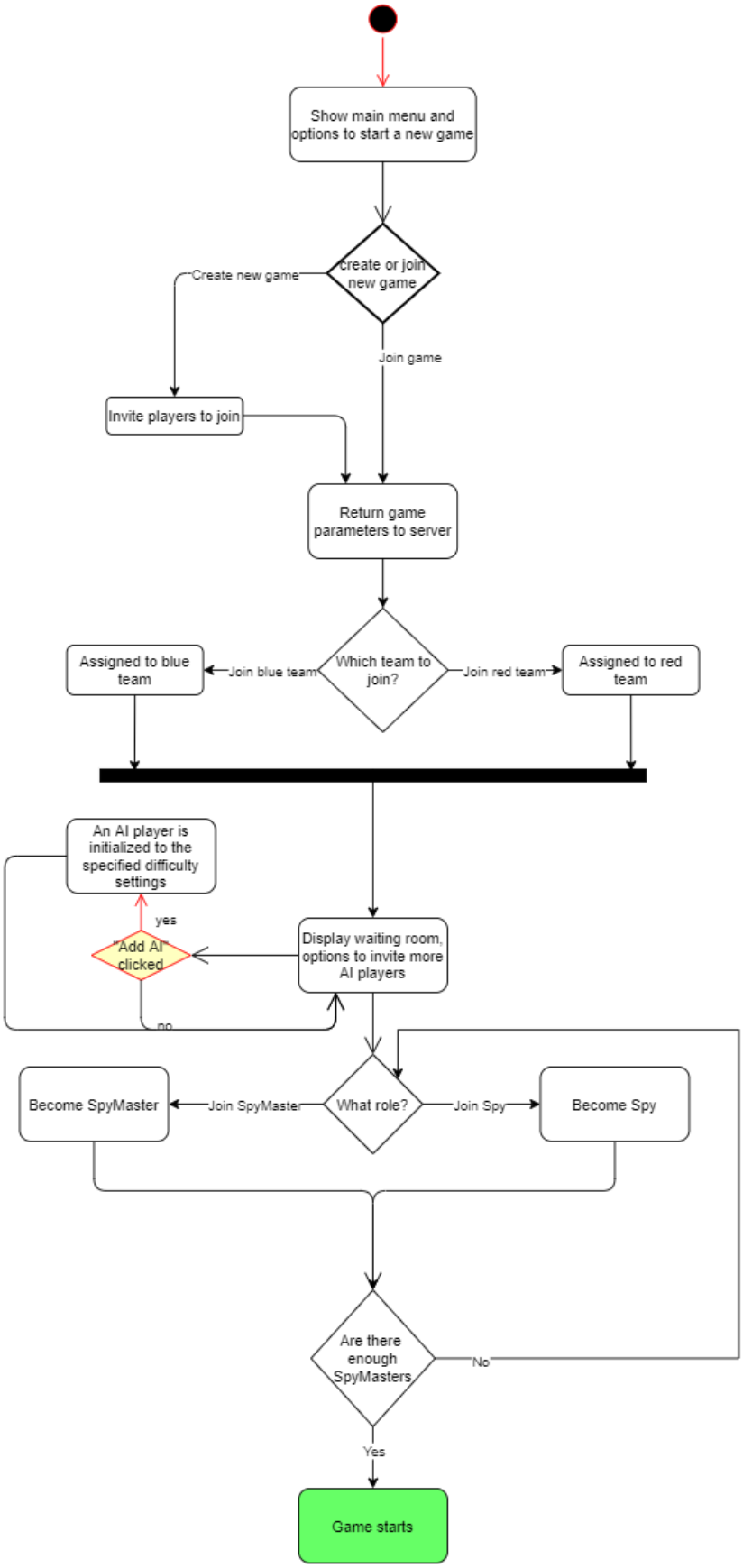

Figure.2 Activity Diagram for Main Menu

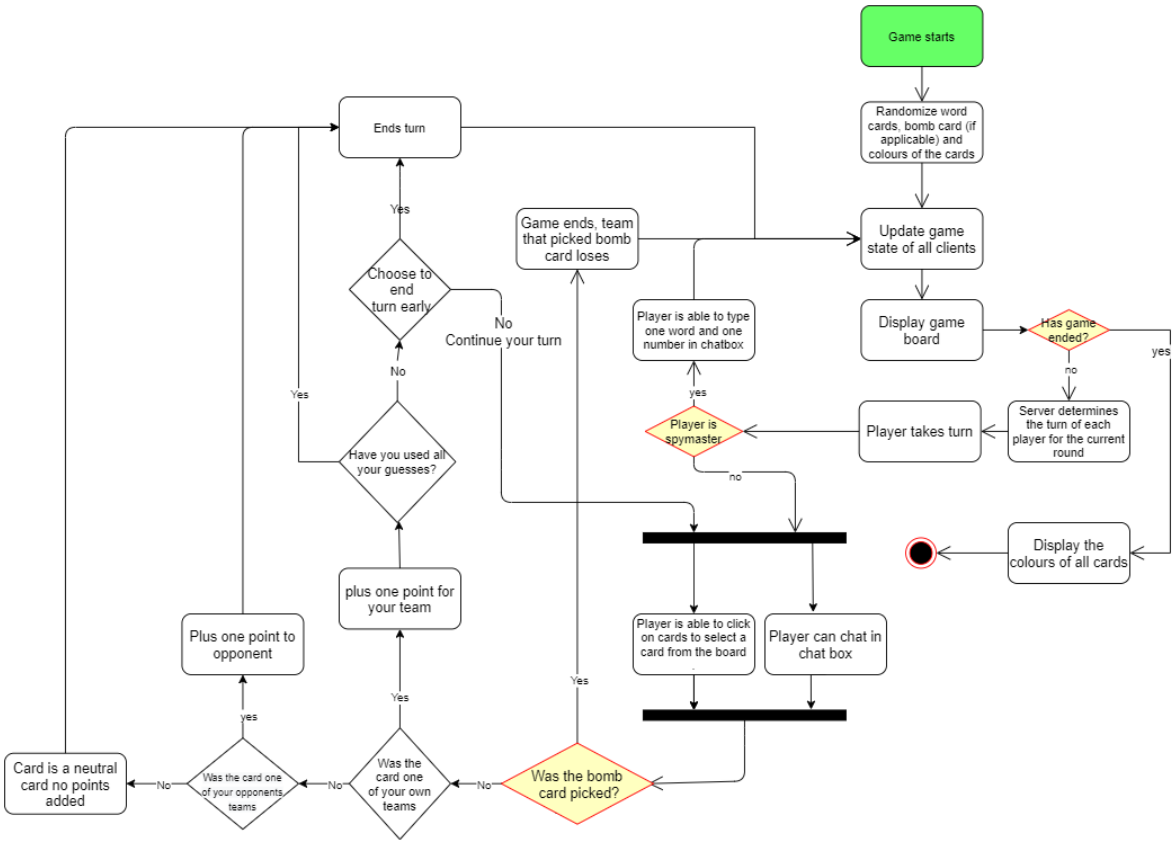

Figure.3 Activity Diagram for Game Mechanism

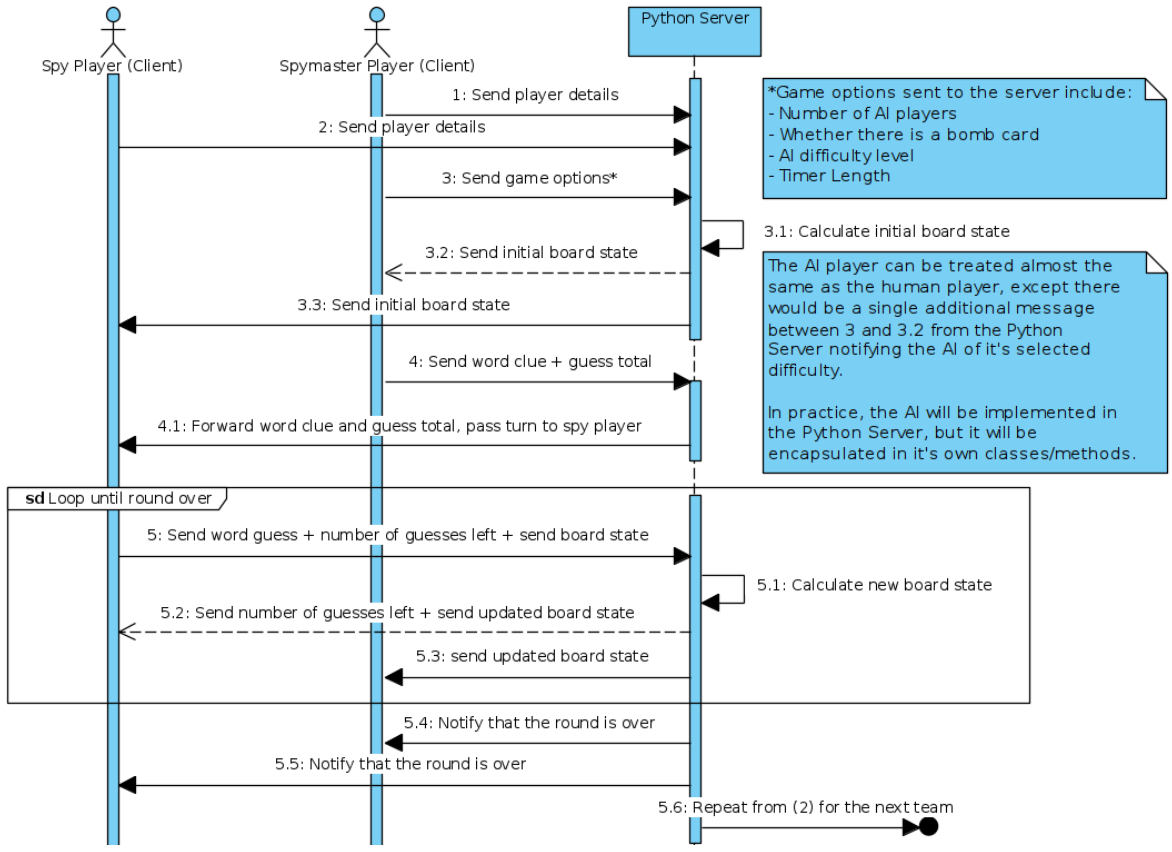

Figure.4 Sequence Diagram for Interaction between Server and Clients

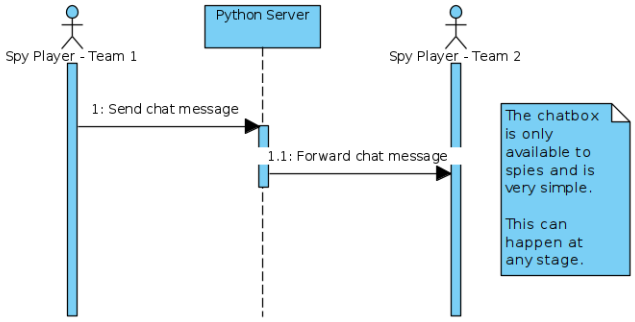

Figure.5 Sequence Diagram for Functionality of the Chat Box

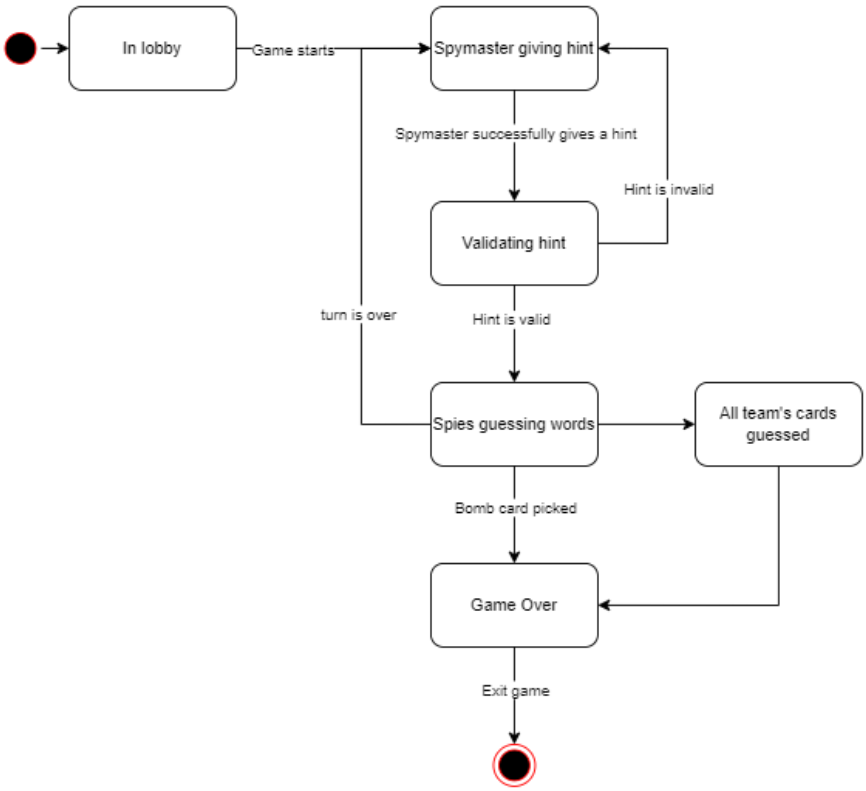

Figure.6 State Diagram

Analysis of techniques for the project

For the hardware we will use computers or mobile devices, as we decided to make this project run in the web browser. What we need to consider is the layout of the webpage on both desktop/laptop and mobile devices.

The tools and methods we planned to use to build this project include:

Programming Languages

- Python - backend language suitable for game mechanism, prediction making model, machine learning framework and web server. Its abundant libraries provide useful tools for the project.

- JavaScript - language for user interaction with graphical interface and connection between frontend and backend. It is also an event-driven client-side scripting language with relative security, widely used for client-side Web development.
- HTML/CSS - languages for Web development.

Web Server

- Flask - a micro python web framework, it is lightweight and designed to make getting started quick and easy, with the ability to scale up to complex applications.
- Socket.io - a library that enables real-time, bidirectional, and event-based communication between the browser and the server (can be implemented using Flask).

AI Techniques

- Glove/word2vec - Methods to convert word to vectors (use cosine similarity on word vectors to get word similarity). They mainly focus on context to get words similar in meaning. We do not use fastText since it focuses more on the form of words. We plan to implement two kinds of AI (Spy and Spymaster) by extracting suitable words from vocabulary using word vectors.

Game Libraries (

The following 2 libraries are not yet implemented but are noted because they may be useful in the future

- PyGame - adds more features to the excellent SDL library and allows you to create full-featured games and multimedia programs using Python, also it is highly portable and can run on almost all platforms and operating systems.
- Python-Ogre - a complete Python wrapper for the OGRE 3D engine, as well as 15 other graphics and game-related libraries for GUI, physics, special effects, sound, and more.

Reflection

We have done adequate preparation before implementing the software, including Requirement Gathering, Modelling and Validation, Specification Visualizing and Prototyping. We have also analysed feasible methods for the project to ensure it can be done well in a proper way. However, the most challenging thing for now is how to integrate the game mechanism with the web development part, since we use Python rather than JavaScript for the backend, which is something we have not done before. We plan to focus on the connection first, then we can work on separate parts.

Initial software development and testing

Architecture and design

First, we split our project into three layers: the data layer, logic layer (Game mechanism) and page layer (game web), which is similar to the MVC pattern which we are remarkably familiar with. Under this clear and reasonable pattern, we can divide our team into sub-teams thus making our job more efficient.

For the logic layer, we have decided to use Python when coding our project as it is a high-level language which uses less code to implement complex designs, and programmers can focus more on the problem itself rather than the language syntax. There are abundant frameworks and libraries for AI coding as well.

Inside the logic layer, we also plan to follow the MVC pattern. As shown in Figure.8, the logic layer has a main controller class (GameController), which is responsible for reading and setting data from the model classes (GameBoard, GameSettings, GameAI, Spy and Spymaster), and passing data to the view

classes, which is related to the webpage. The architecture of the logic layer may change due to the interface of the web server.

We are all using PyCharm for python to make sure software version is kept the same. The algorithm details for the game AI are available in **appendix C**. In terms of the choice of game platform, we chose to run our game on the web rather than the APP, because the web is more compatible with our code, and players can use the built-in browser to play our game no matter what device they use.

Inside the page layer, we have split the view into the BoardState, ChatBox and Lobby class, as shown in Figure.7 and as detailed in **appendix C.** The page layer makes use of inheritance as these classes are children of the Observer class, allowing them to be updated whenever the Update() function is called. The Server Class handles all the interactions between the Server and its Observers so that other classes can use the Server Class methods regardless of their implementation.

We are using Sockets and the socket.io library which allows clients to connect with the server and each other which will allow us to create the multiplayer side of the game. As for how current development is going, the technologies we have used are proving beneficial and have been successful in current stages of development.

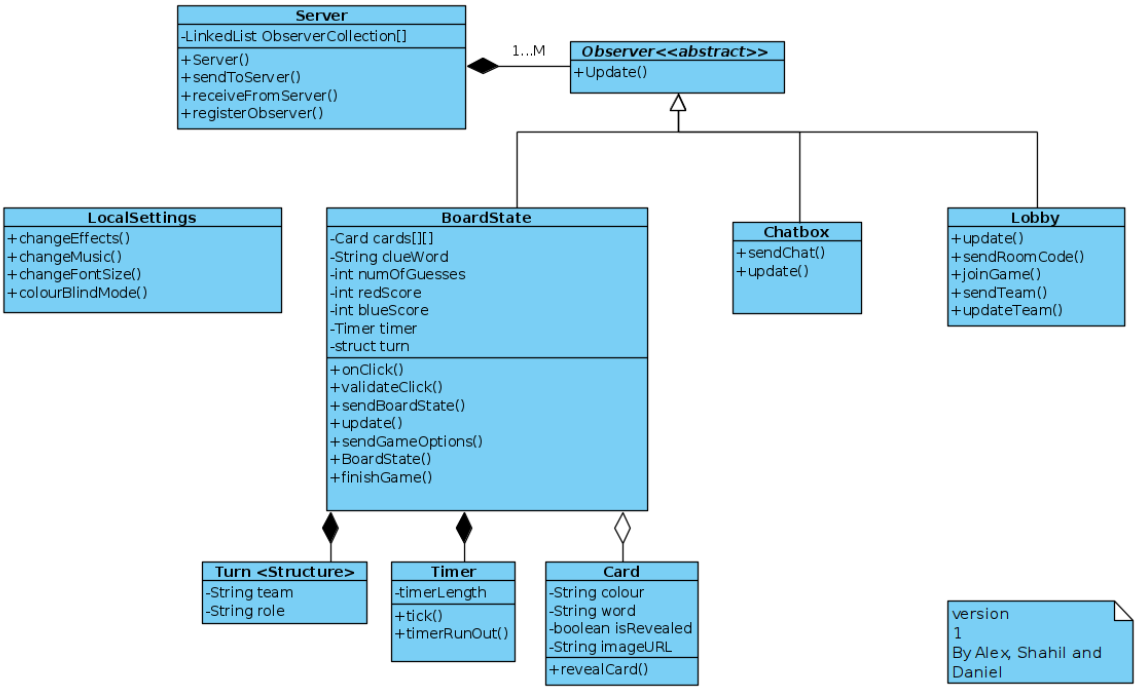

Figure.7 Class Diagram for Web Dev

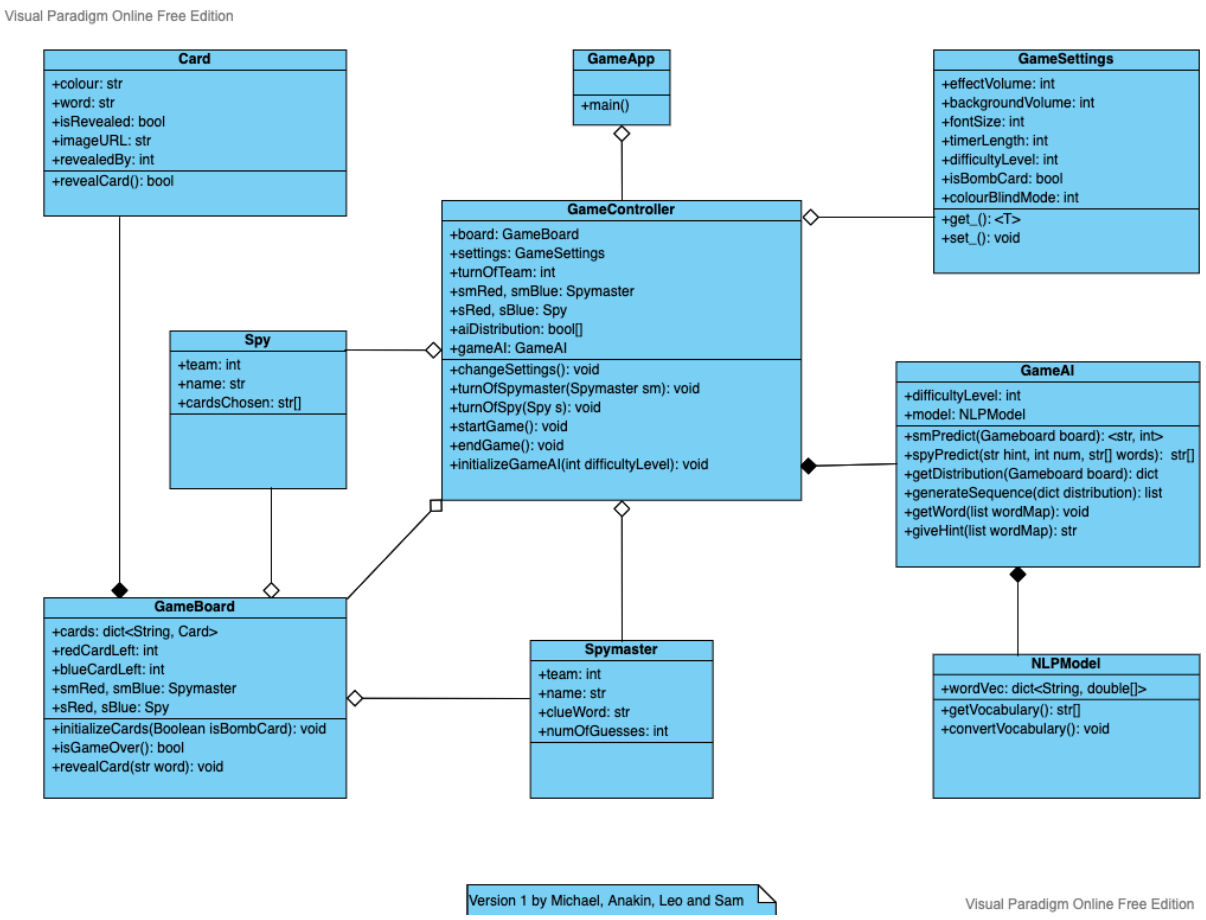

Figure.8 Class Diagram for Game Dev

For the data, we currently using a we.txt file to store word data due to its convenience, and in the future, we may update it to SQL storage for more extensive usage. For the card information such as cardName, cardId, cardType, active and colour we use a list of dicts to store it in the second position, and the first position stores the difficulty level of the AI for example:

```
[{'name': 'bark', 'type': 'neutral', 'color': '#D0D0D0', 'active': False, 'id': 1},
{'name': 'carrot', 'type': 'red', 'color': '#0080FF', 'active': False, 'id': 2},
{'name': 'missile', 'type': 'red', 'color': '#0080FF', 'active': False, 'id': 3},
etc...]
```

Git Usage

As a group we have decided that using GitLab is the best approach to completing this project to the highest standard. It allows us to collaborate as a group in every stage of our project. We have split up into groups to complete different components of the software in the most efficient way. Since the start of the project, we have been using issues and milestones to assign tasks among us with deadlines and tags to make sure all tasks were completed on time. This also allows us to communicate with each other when we need help with anything.

We made sure to make effective use of version control through branching and merging to work on key areas of software which helps speed up code development while also protecting code in other branches. We created a WebDev and GameDev branch separating the front end (WebDev) and the

back end (GameDev) where all the game mechanics are created. There is also an integration branch which will be used when merging code from both WebDev and GameDev. This allows us to make sure the code will work together properly before we merge back up to the main branch.

Testing

When testing our software, we made sure to test each class separately and the methods in the class. We did some functional testing to make sure the functions we created worked as they should and provided results as expected. As it is a web game, we have tested the looks and functionality by running the web pages seeing whether certain features and functions work as intended giving the right responses. For example, we had to do some user testing for the revealCard() function such as clicking on the card on the screen to make sure it changed to a picture to reveal the card.

Before card is clicked

After card is clicked

We have also done tests on making sure the client and server messages are received as intended. For example, in sendBoardState() we tested to make sure the protocol and message sent to the server is how we expect the message to be. We went through this before Coding on how these messages should be written so both sub teams know what needs to be sent between the client and the server.

```
server.sendToServer("sendBoardState",
    {
    "Protocol" : "sendBoardState",
    "clue" : this.clueWord,
    "numberOfGuesses" : this.numOfGuesses,
    "redScore" : this.redScore,
    "blueScore" : this.blueScore,
    "timerLength" : this.timer,
    "player" : this.player,
    "turn" : this.turn,
    "cardChosen" : `${i},${j}`,
    "cards" : this.cards
    });
```

Figure showing the protocols to send to server

Preliminary software documentation

Before we started developing our software, we made sure to create class diagrams and class descriptions for all the necessary classes, including all variables and data we need to store and the

methods we need for the software to work as intended. We used our requirements and specifications to produce a suitable method to develop our software to fulfil our requirement goals. Due to thorough requirements and specifications gathering we have been able to create a secure software design/structure which thinks about the future development of our software ensuring we minimise the risk of encountering any uncertainties or problems. Also, before starting software development, we introduced coding a Quality Manual document which takes into consideration how our code is written to ensure code legibility and reusability. We also used paired programming which helps reduce bugs in our code while increasing code quality. We thoroughly looked through the different technologies and architectures that were available to us and what they had to offer before starting development. In terms of inline documentation, we currently make use of them through comments in the code explaining the use of the methods and what they do. We are constantly using comments in our code as inline documentation to make sure all code is legible and understandable; this helps every member of the team and external views understand what the code is doing.

Reflection

One thing we can work on which we will do in the future is make more use of inline software documentation (A word/PDF document) talking about the changes and additions we made at every stage (Dated) and the functions of it. Towards the end we focused more of our energy on creating the prototype. In terms of team collaboration, we are working extremely well together and will continue to work as we are. Any technical difficulties we have had are always addressed with aid from other members.

Project management

Project Management Approaches

At the beginning of this project, we decided to use a hybrid approach of both Traditional and Agile methods. Our supervisor and sponsor agreed that those requirements were unlikely to change greatly throughout the project and hence we made sure to benefit from a thoroughly documented requirements and specification process at the start of our project. This expansive preparation means that we have a deep understanding of how the project will need to be developed in the future.

As part of our hybrid approach, we have used a variety of Agile methods where appropriate. We have used SCRUM with 2-week long sprints and retrospectives. During these retrospectives we have discussed whether we have met our deliverables, how the team has found the work and how effective our methods and tools have been in meeting our targets. Our retrospective documentation is available in **appendix F**. Further to this we have had bi-weekly meetings with our supervisor, placed either just before or after our retrospectives, where we have shown him our sprint deliverables and written any feedback in **appendix E.** We have been continually working on feedback, for example we incorporated a sliding side-panel into our design when we were asked to better consider mobile compatibility.

In addition to our bi-weekly sprint and supervisor meetings, we have held regular meetings for 2-3 days a week on average giving us a lot of time to share ideas and plan. During these meetings issues are assigned to team members using our project management tool, GitLab.

Project Management Tools

Our team decided on three different tools:

- Discord for online communication and messaging
- Microsoft Teams for file (documentation) sharing

- Gitlab for planning and managing workloads (see below)

Gitlab was the most important management tool used in this project. We decided to use it because most of the team have had experience with it in the past and because it is closely integrated with our project repository, meaning we can link issues and milestones with git commits.

During the project we have split the workload into milestones and issues and have assigned issues equally between the team so that everyone has had roughly equal amounts of work to undertake. Gitlab has also allowed us to be able to organise these issues by using a board (similar to Kanban) and by applying tags to identify features about them. A full list of issues, tags and milestones are available in **appendix H**.

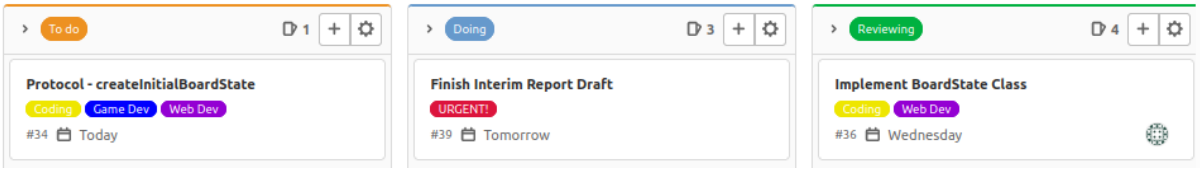

Figure.9 GitLab Issues

Team Skills and Coordination

Members of the project have been active and eager to work as a team, demonstrated by a total 2607 individual messages in our discord server (as of 13/12) and the frequency of our meetings.

Since the early stage of our project there have been two sub-teams, one handling the client-side (web-development) and the other handling the server-side (game/ai-development). The groups were split based on what people were most interested and on what people were most experienced with, meaning that both teams were well-rounded.

Each team has practiced cross-training, where team members that were more knowledgeable in an area taught others. An example is in the client-side sub-team where one team-member helped the others with the JavaScript Document-Object-Model while in exchange they helped him learn more about JavaScript Server-side events and socket libraries.

Risk Planning

In terms of risk-planning and contingency management, members can switch between sub-teams with ease because of the same code conventions between both teams, making newly viewed code quickly readable (again refer to **Appendix D** for more details). Additionally, members of one team should already have a good idea of what is happening on the other team due to the reviews we have in meetings, where both sub-teams share what they have done.

Earlier in our project one of our team members unfortunately could not work due to injury for a brief period of time (but has caught up and contributed very well afterwards). During this time, another team member was easily able to temporarily change sub-teams to cover the team which would have otherwise been left with a very imbalanced number of workers. The project was able to continue as normal. While a very unfortunate situation, the silver lining is that this does prove that our risk-mitigation is practical and successful.

Preliminary Reflection on Management

The team has worked very cohesively together and has followed everything that has been discussed in meetings. Our project management approach has been consistent, and the combination of traditional and agile methods has allowed us to have a thorough and well-documented understanding of the requirements. The one drawback of this is that we started coding relatively late, hence in hindsight we should have started a week earlier to start the coded prototype.

The use of project management tools was appropriate for the task, as were the team skills and coordination. To further improve in the future, we will have more frequent milestones and more descriptive GitLab issues.

Conclusion

In conclusion, we consider that we have structured and split up our work in a particularly good and appropriate way. In the current team compositions, each part is done by groups of people which include teammates that have experience on that subject as well as teammates that have a high interest in the subject, following the paired programming technique. As stated above we have a very flexible and risk-free team structure so we do not think that there will be any changes on this matter moving forward. In addition, the time allocated for each part of the project so far was about the right amount, except for the last sprint, for which we should have allocated ourselves a little bit more time. Fortunately, because of the great teamwork we managed to get everything done on time, regardless. Moving forward we will try to divide our work in a more feasible way, which will be fairly easy to do. Since we have already integrated the client and server side, any other changes will just be add-ons which will be easy to implement.

On a final note, we hope this document has portrayed how well development is going and how we, as a group, have been working together.

Appendix

APPENDIX A – REQUIREMENTS DOCUMENT

During the first sprint of our project, we wrote a thorough requirements document listing the requirements and organising them using diagrams.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Requirements%20Document%20V2.pdf

APPENDIX B – SPECIFICATIONS DOCUMENT

Similarly, to the requirements document above, we wrote a thorough specifications document listing the specifications and organising them using diagrams.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Specifications%20Document%20V3.pdf

APPENDIX C – CLASS SPECIFICATIONS

In our second sprint we created a class specification document that defines each class that we would like to program and their associated methods. This also includes two class diagrams.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Class%20Descriptions.pdf

APPENDIX D – QUALITY MANUAL

The quality manual outlines how code will be written, reviewed, and committed to the git repository.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Quality%20Manual.pdf

APPENDIX E – MEETING MINUTES

The notes of every meeting we had are recorded in the meeting minutes document.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Meeting%20Minutes.pdf

APPENDIX F – RETROSPECTIVE NOTES

There have been three sprints so far. The deliverables for each sprint and the comments given during the retrospectives are found in the following document.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/blob/main/docs/Sprint%20Retros.pdf

APPENDIX G – SOFTWARE REPOSITORY

The code is stored in a GitLab repository.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/tree/main

APPENDIX H – GITLAB ISSUES AND BOARD

The GitLab issues, board, and milestones that we used to keep track of our project are available below.

https://projects.cs.nott.ac.uk/comp2002/2021-2022/team18_project/-/boards/61