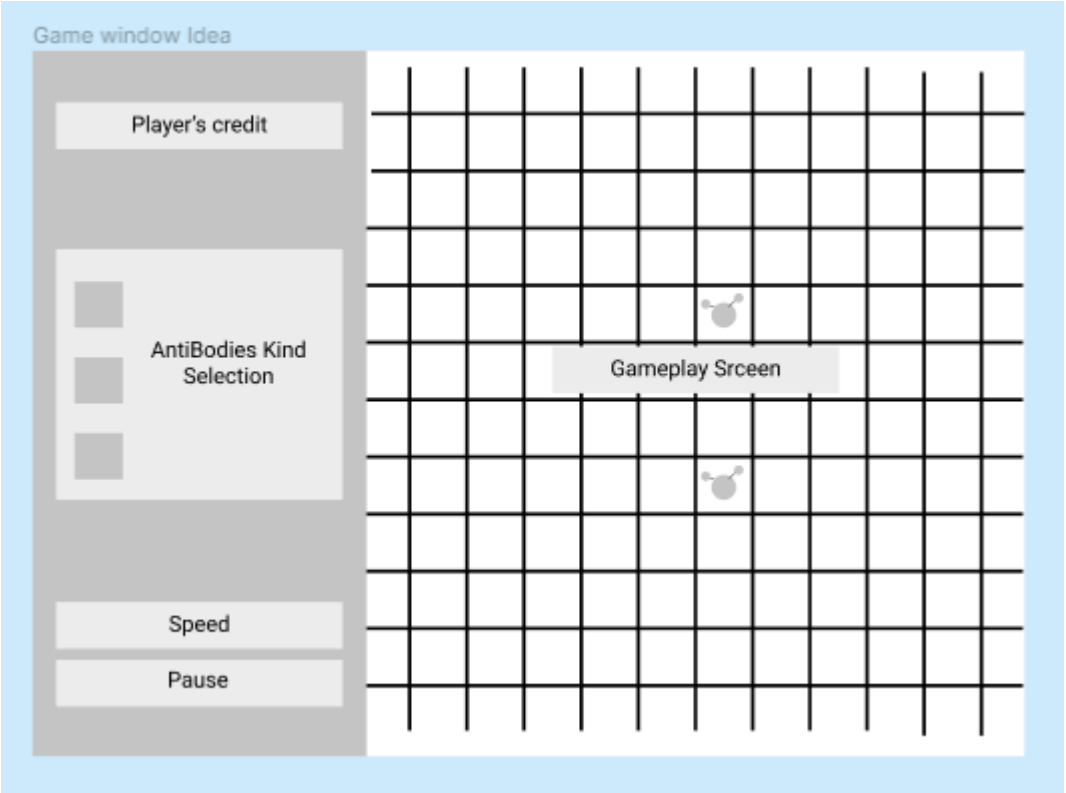
Front Cover:

CPE200: Proj: CARIN: Group: Yukon Gold: Overview Document

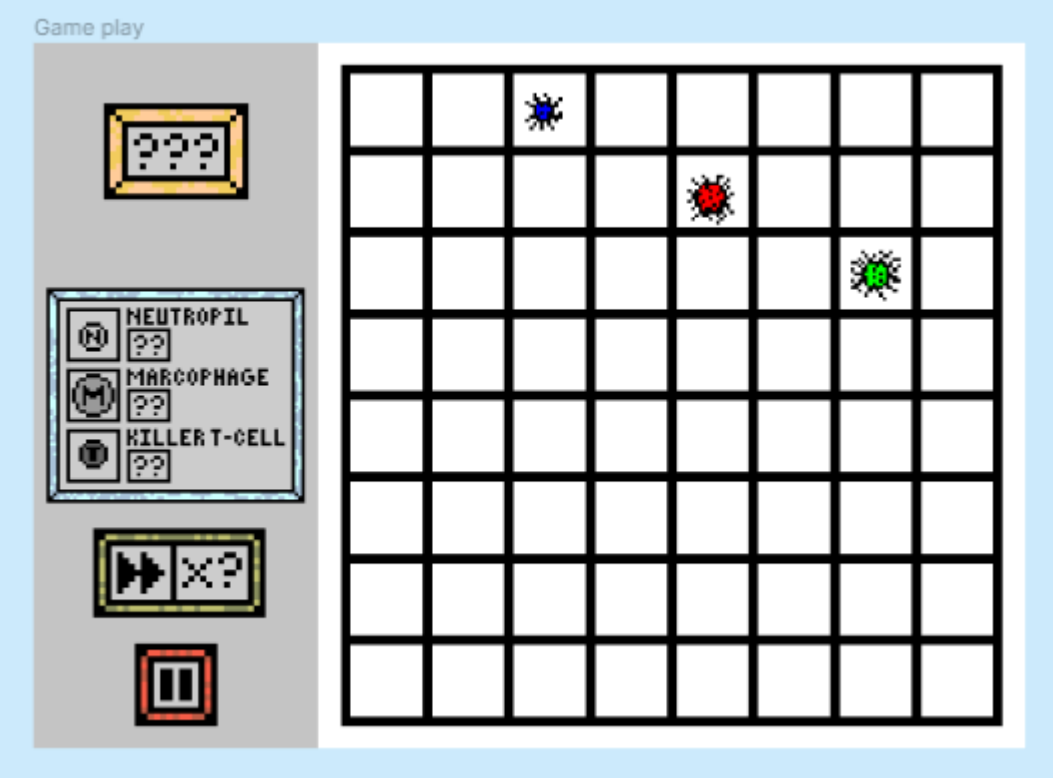
Overview Document:

Design:

Game preview and its brief description:



- PrototypeGameplayWindow



- GameplayWindowDesign01

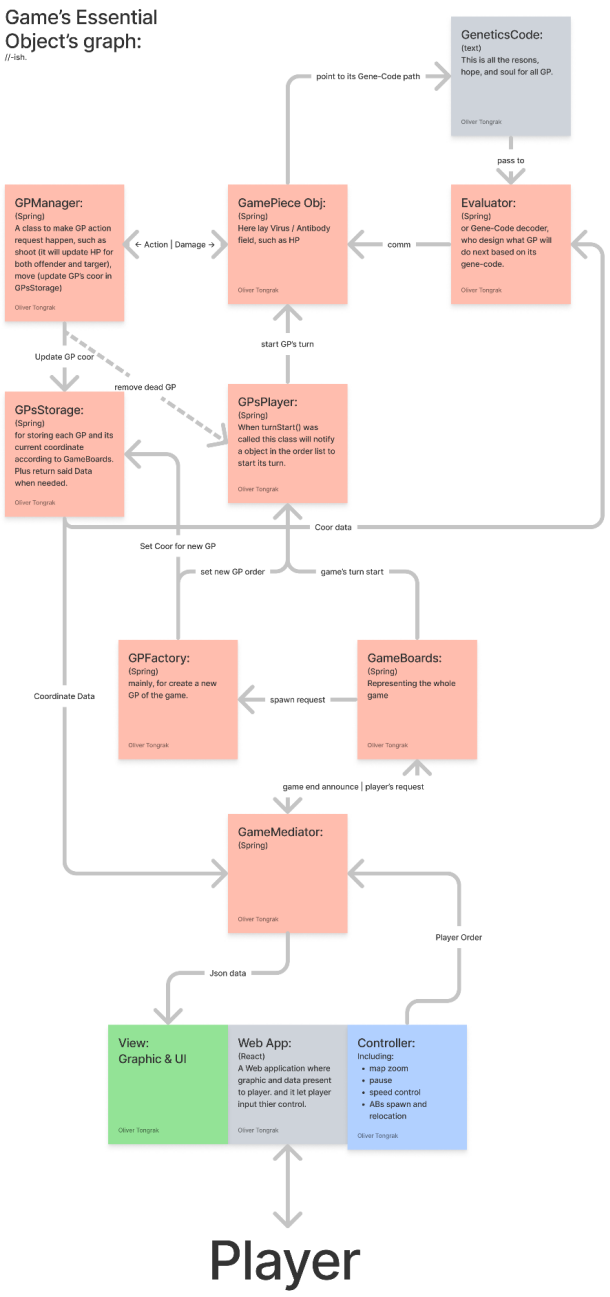
Control description:

- Speed control button: It is a button pressed to increase the game speed divided into 3 levels (x1 , x2, x4).
- Antibody Select Button: Used to select Antibody to be placed. To place an Antibody, you have to click twice. The first time will be the selection and the second time. It will be placed at the desired location.
- Zoom button: Select the section you want to expand to see.
- Pause button: Pause the game and If you haven't played the game yet You can start a new game. but if you ever play you can continue playing.

Class control:

- Class Speed control button: When pressed, it will send that it has been clicked to the back-end to adjust the speed of the game.
- Class Pause button: When pressed, it will send the back-end to pause or resume the game.
- Class Antibody Select Button: Clicking 1 is the position of the ab and clicking 2 is the position on the grid, it will send the data to the back-end.
- Class Zoom button: Select the part you want to expand and send the data to the function of The view creates a grid equal to the expanded portion.

Class Hierarchy and Objects relationship Diagram:

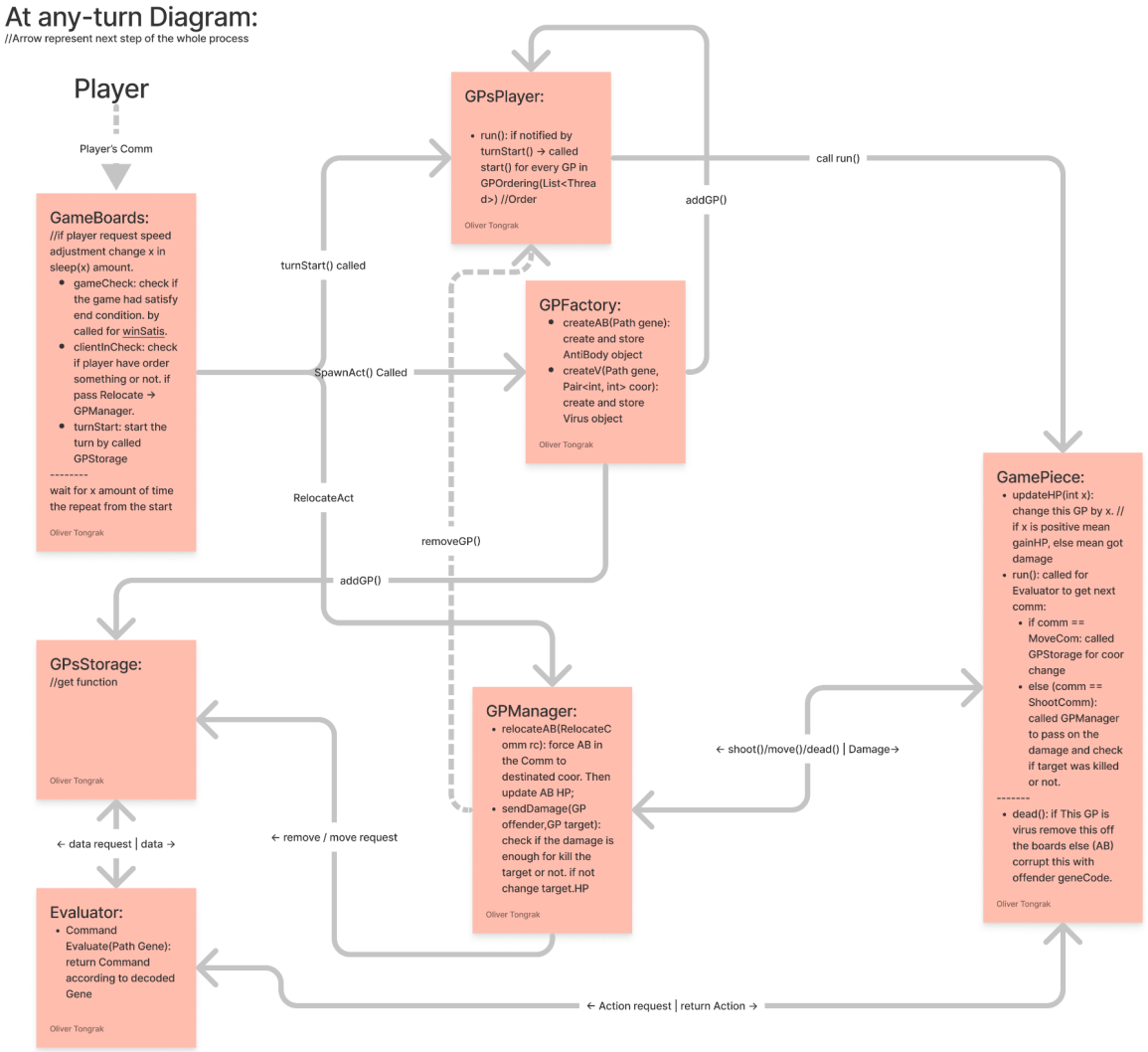


(Fig1: Game's Essential Components)

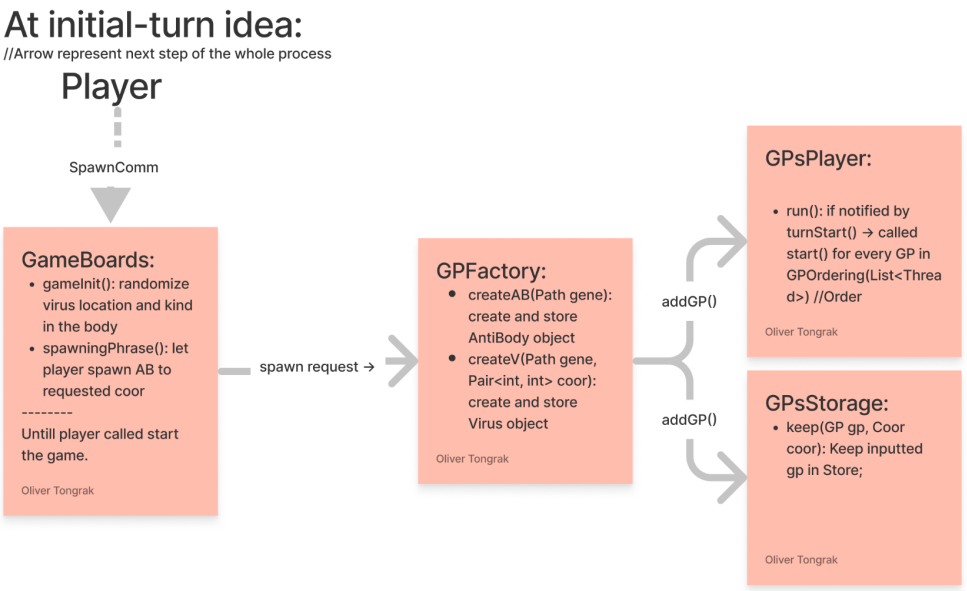
Interface / Class design:

//arrows represented "is a subClass"

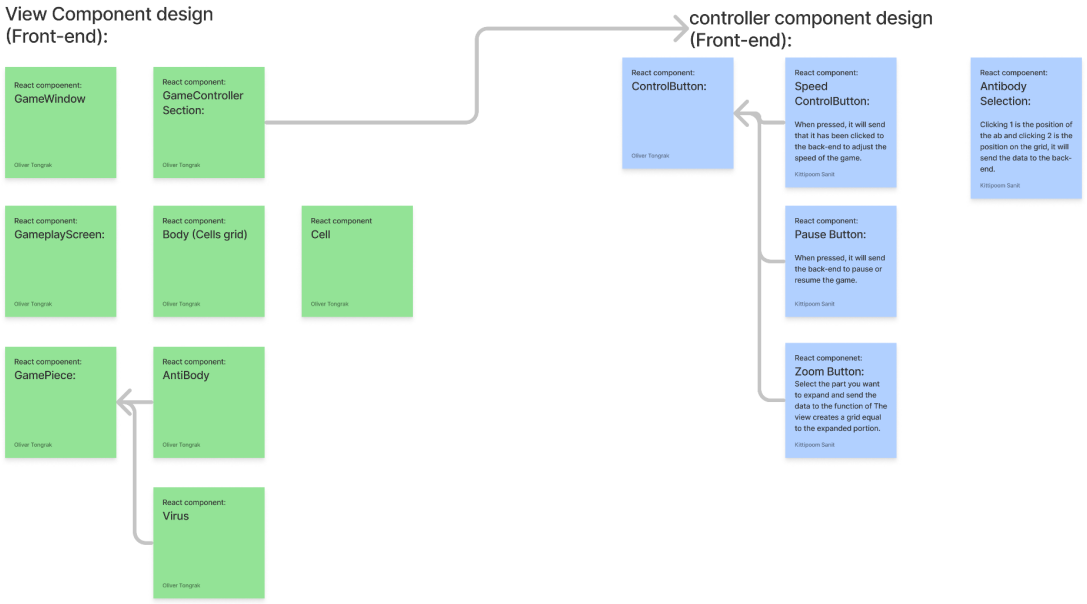
(Fig2: Interface / Class design)



(Fig3: At any-turn idea)



(Fig4: At initial-turn idea)



(Fig5: FrontEnd component design)

Architecture:

Note: GameBoards refer to: m x n body cells
GamePiece refer to: Virus or Antibody like object.
AB refer to: Antibody

model class description:

GameBoards:

Representation of the whole game. Meaning, this class receives the order from the player and sends data to the front-end. The responsibility of this class included:

Initiate the game: preparation of GameBoards and call for randomise virus and its spawn position.

Enforce the player's request : when the player's request was detected, the class will call to make the requested order happen like: AB spawning, AB relocation, game pause, game's speed adjustment.

Call for game's continuation or start: from the first GamePiece's action to the last in the GamePiece list action can be viewed as a single "turn" of the game. Meaning, this class has to call for a turn to start first before any GamePiece can take its turn to move, shoot, or even to make a player-to-GamePiece request to take effect. This step was designed for game's speed adjustability.

GPsPlayer:

Designed to be a class for controlling a Thread to "run" each GamePiece, sequentially.

GPsStorage:

Class to be used as the main storage to keep all data related to all GamePiece. The referred storage including:

List of GamePiece: to store the order of GamePiece to be used as an order for each GamePiece's turn.

Map of Coordinate and Game: to store GamePiece and where it's currently located in GameBoards.

GPManager:

Designed to act in GamePiece's behave. For example, if Virus is designed to shoot(according to its geneCode) an antibody nearby. The virus will call GPManager to send a damage report, then GPManager will update the target's HP.

GamePiece:

Representing an antibody or a virus in the boards. This class has to have a path to its genetic Code which when got the green light (got called start() by GPPlayer) it will call GPManger to send the damage to the targeted game piece.

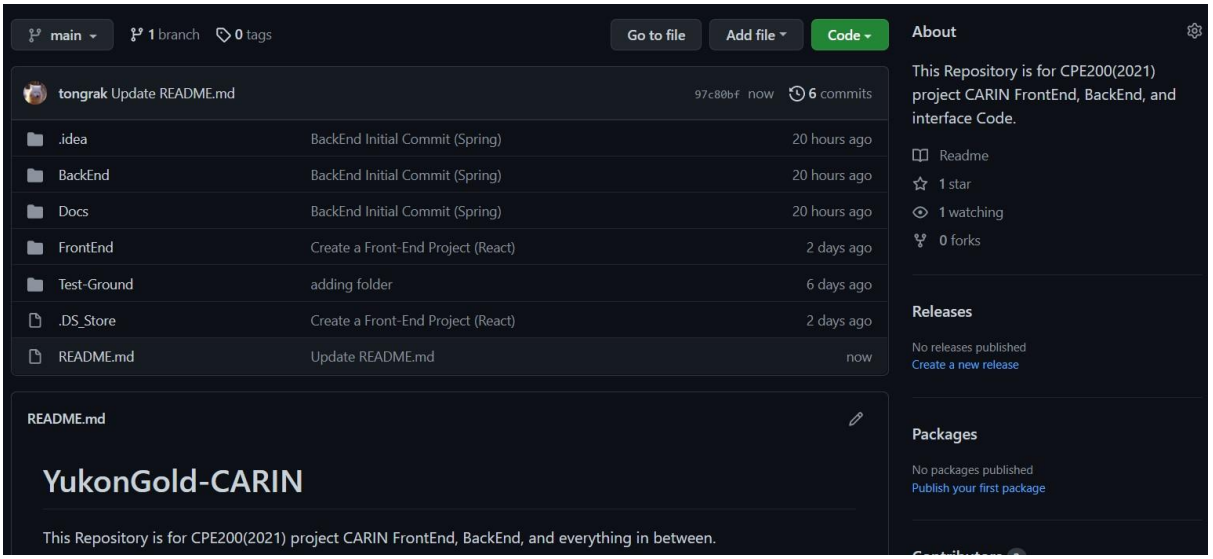
Data structures:

- **List:** for store order of game piece action turn.
- **Map alike:** A bipartite graph which each side will only connect one with another side. for storing unique Coor connected with a unique GP.

Tools:

IED : IntelliJ and VisualCode
Front-end Framework: React
Back-end Framework: Spring
Testing Framework (plus testing request): JUnit5 and Postman
Graphic Designing: Figma.

Git repository:



(tongrak/YukonGold-CARIN: This Repository is for CPE200(2021) project CARIN FrontEnd, BackEnd, and interface Code. (github.com)

Overview Document:

Testing plan:

Our tests can be separated into 2 forms: API-stuff and BackEnd-code.

API-stuff:

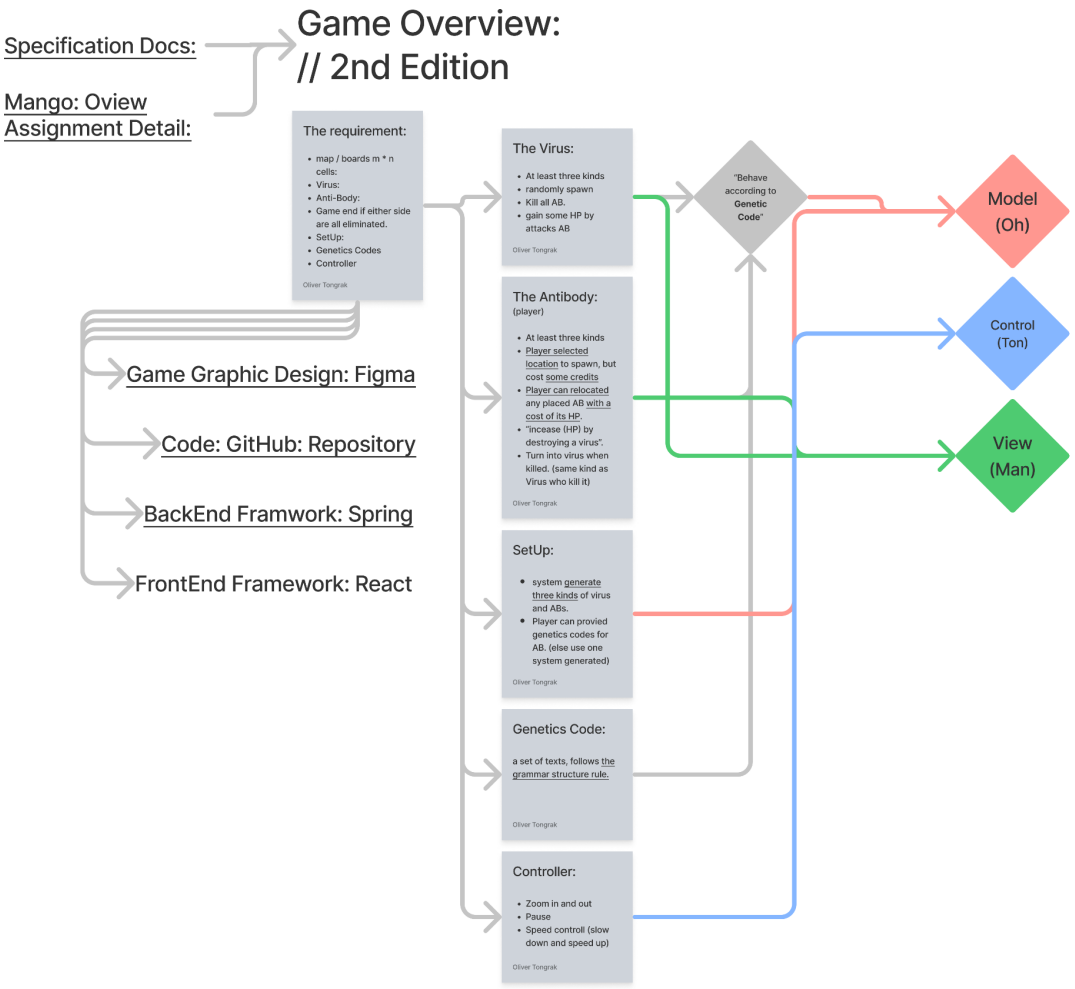
We focus our tests toward the controller section. By testing each feature to see if the request has been sended to BackEnd or not. Plus, testing for response from BackEnd if it came in predetermined form or not. We will use PostMam for testing API requests and responses.

BackEnd-code:

We planned to use the testing method which we had learned in one of the lectures. Said method advice testing each step from starting from code designing to code intergrading. Each testing session will attempt to test both Black-Box and Glass-Box tests, if possible.

Overview Document:

Work Plan:



(Fig6: First Time Game Design)

Due to time limited-time and lack of project design experience. Our group designs our work plan that will minimise work conflict. We divided our works according to the MVC idea. Which means every person working in a sipperate field which also means minimising conflict in both work and git commit. This method also ensures that every person's work cannot be struck due to pre-required work from another person.

But this project heavily requires work from the model side, which our group have come to an agreement that the person responsible for view and controller will help out with the model side on parts that have a clean pre-determined interface and method.

Overview Document:

ProblemS:

Overall, our group lacks experience in software designing, planning, let alone implementing. For said reason, there will be challenging tasks, a number of mind-numbing bugs, big miss calculations in designing and planning, and a group's meltdown in generals.

In addition, these are other problems we have found:

- Can’t remember React FrameWork Implementation, much.
- Forget about react framework knowledge.