Software Requirements Specification

Hex Army

Prepared by Wizard Monkeys

University of North Texas - CSCE 5430

Contents

Group members :

No.	Name	Student ID	Email
1	Jeremy Glebe	11452290	JeremyGlebe@my.unt.edu
2	Haiyi Wang	11528159	haiyiwang@my.unt.edu

Version: 3.0

Document History

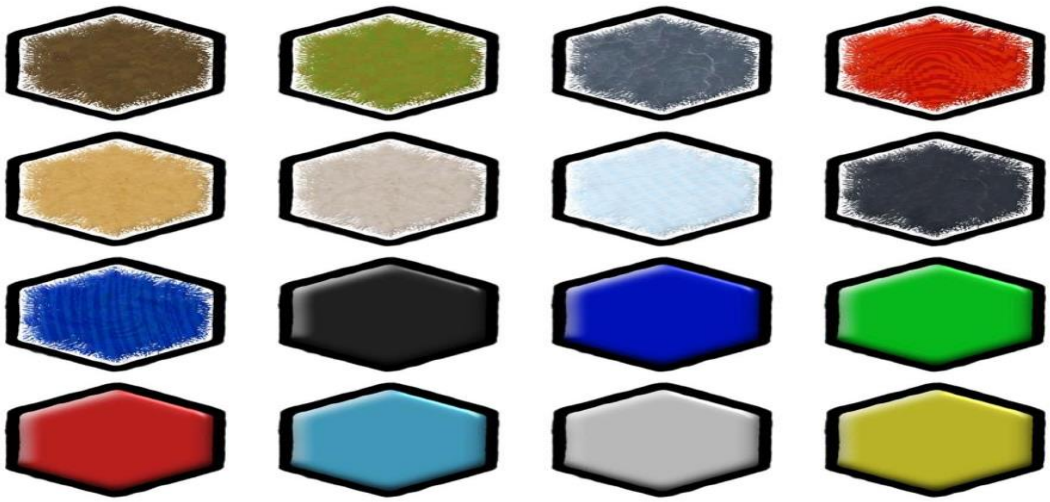
Version	Date	Author	Comments
3.0	19th Nov, 2021	Wizard Monkeys	Deliverable 5

1. **General Diagram:**

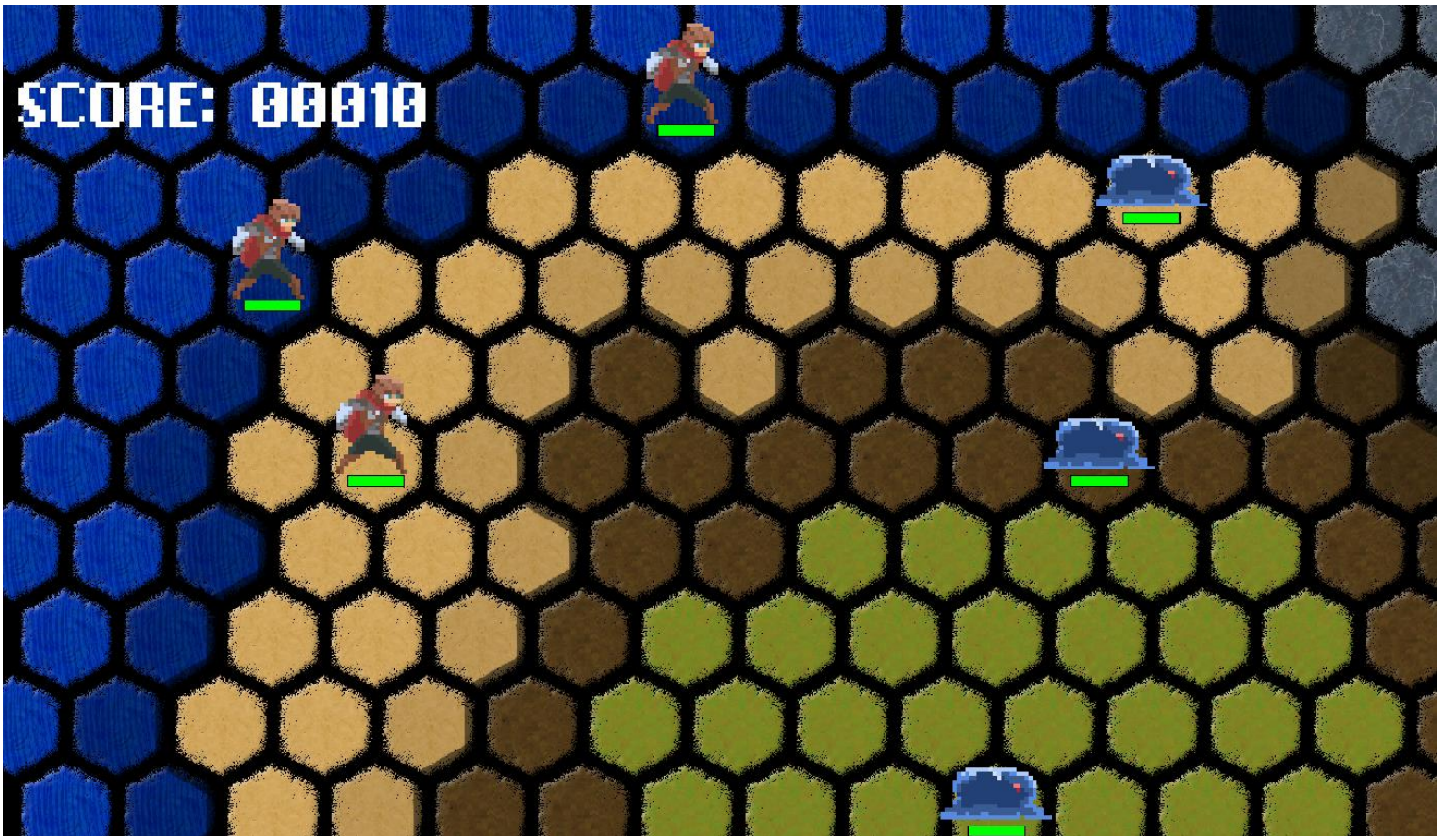
Main page

The game is fitted with multiple hexagon grids with different colors. This leads to an effective layout for the game that not only looks good but also puts emphasis on ease of use. The player and enemies are separated by the different colors.

Hexagonal Grid:



Character Image:



2. **User manual:**

Description:

Controls:

There are two types of controls available for the users.

They are:

I) by using the mouse pointer, the player can move.

II) by using the arrows of the keyboard.

By pressing the arrow keys allows the player to move respective to the location of the arrow key towards the enemies.

1. By clicking the up arrow, the player moves upwards.
2. By clicking the down arrow, the player moves downwards.
3. By clicking the right arrow, the player moves towards the right side.
4. By clicking the left arrow, the player moves towards the left side.

Note:

The player can initially move from a standing point to any direction on a hexagonal grid.

Clear instructions on how to compile/run both your program and your test cases (the program must compile/run):

Install dependencies (must be done once initially and any time a new dependency is added to project)

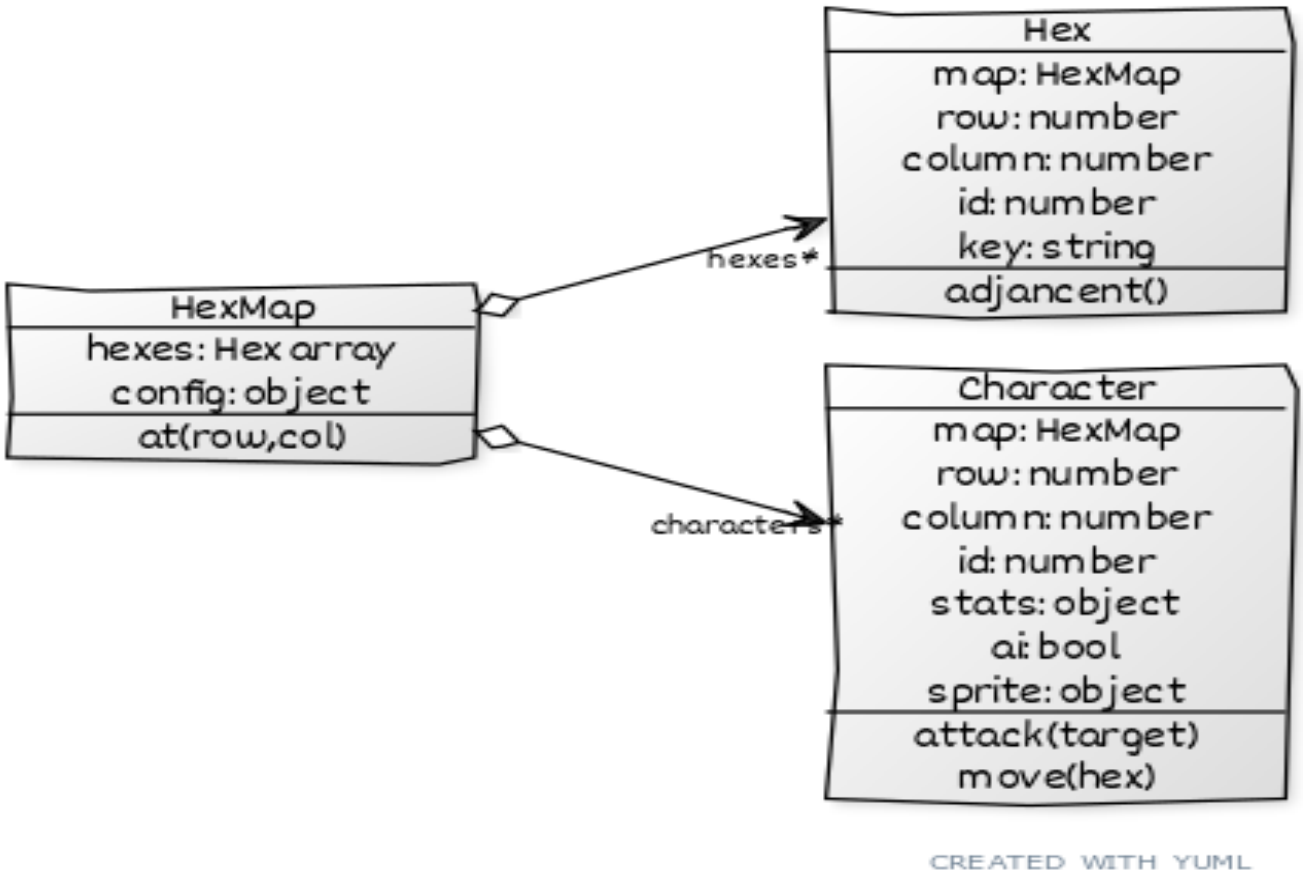
npm install

Run the project in a server (connect with your browser to test)
npm run serve # Use this command
npm run test # OR this, both commands are the same
Run the project in a server open to LAN devices (can be useful for mobile testing)
npm run serve:lan
Just build the project to www/ folder (production mode)
npm run build

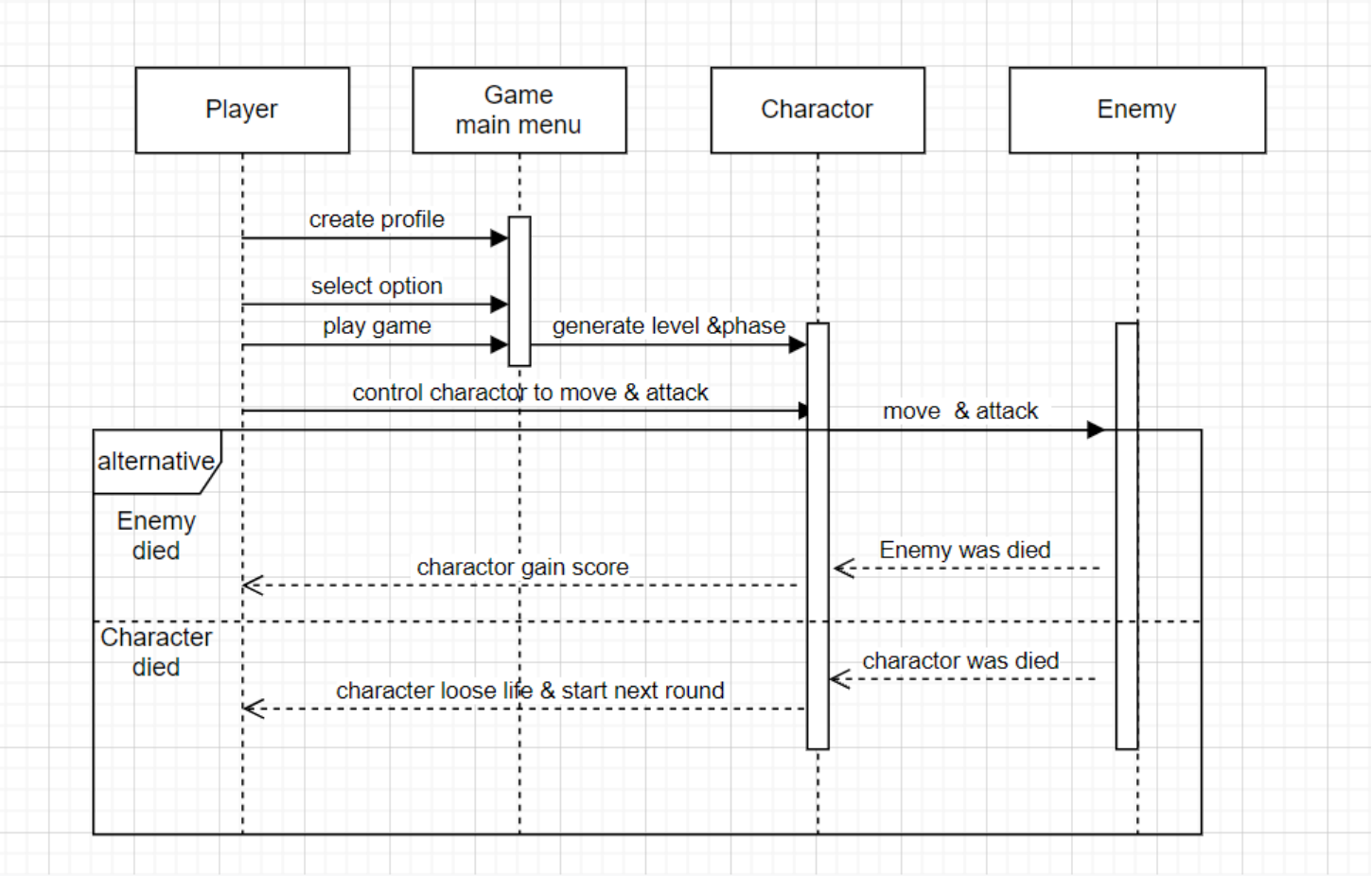
#clone the repo
git clone https://github.com/jeremyglebe/5430-Software-Engineering.git
#use visual studio code to compile and run the code
code .
run start.js file

3. UML design

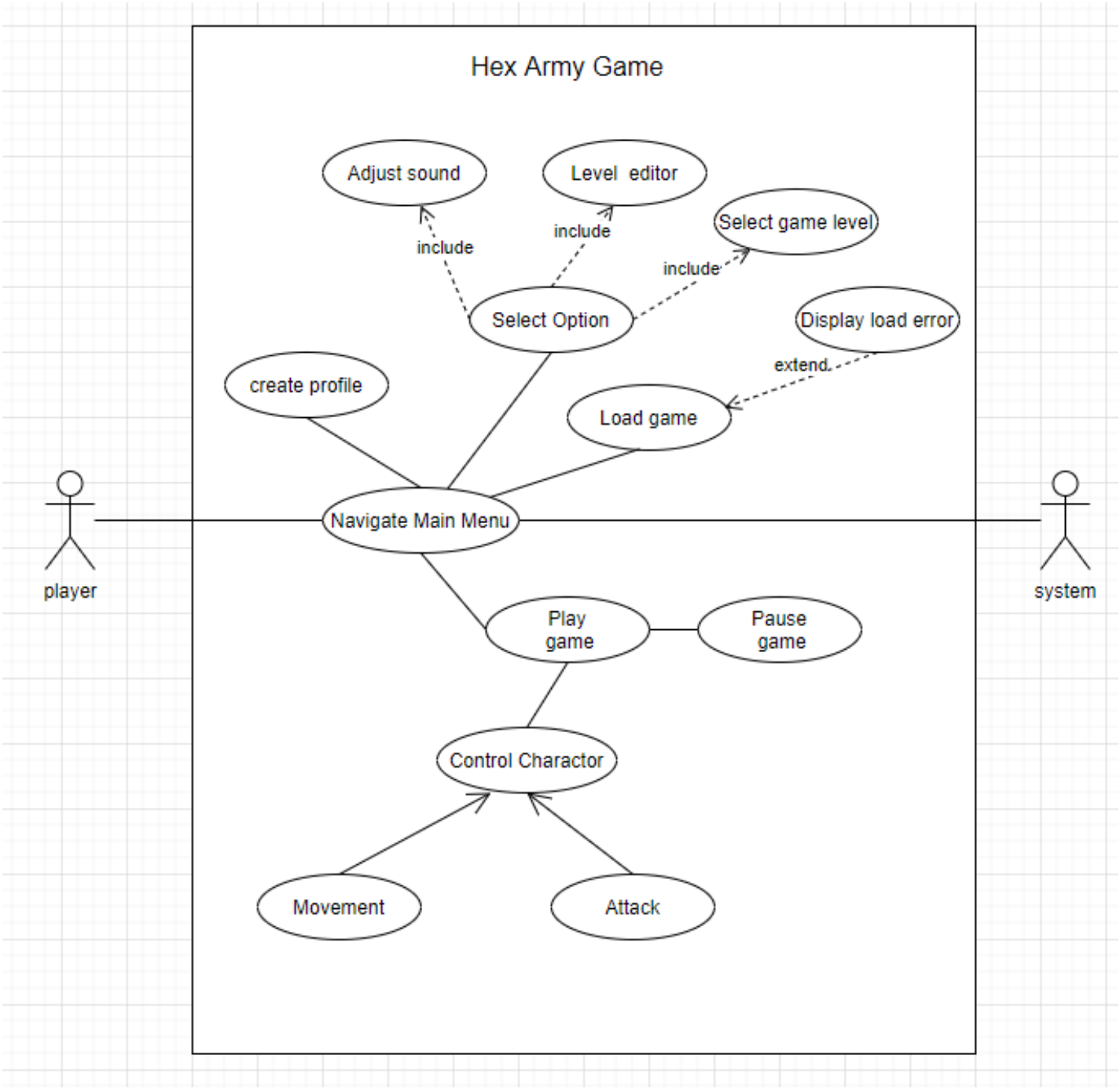
3.1 Class Diagram:



3.2 Sequence Diagram:



3.3 Use case Diagram:



4. Software Requirement Specifications:

4.1 Functional Requirements:

1. The game board(User Interface):

- Created hexagon-based digital board
- Created 20*20 tile grid
- Created 2-dimensional table of the board with odd rows visually stagger to display as hexagons

2.Avatars/Characters:

- Created character of the player and enemy
- Created character will display animations when moving and attacking
- Created health bar of the player and enemy
- Created the weapon of the player and will display animations when attacking
- Created the score when the player gain

3.Inputs

- Accomplished the moving that player can move to any of their six neighbours and could not exceed the board
- Inputs outside of the board's grid should be ignored
- Adjacent characters can attack each other, and a prompt is provided to the player to do so

4.User main page

- Created the main page, the player could choose to play and level editor.
- Create the setup of the board, the player can choose the colour, appearance, size of the board.

5.Sounds

- Added the sound of some backgrounds
- Adder the sound of some clicks and buttons
- Added the sound of the player attack

6.Level Structure

- Levels should contain a unique map of hexagonal tiles
- Levels should contain one or more enemies, with a positive correlation between level number and number of enemies
- The number of tiles an enemy can move each turn should have a positive correlation with level numbe

4.2 Non-Functional Requirements:

4.2.1 Performance Requirements

a. The application should respond to user quickly to user input.

b. Have error handling mechanism that deals with slow internet or no internet issues.

c. Our baseline device for performance is the iPhone 7; Devices with slower processors or less memory are not supported.

d. The game should run at 30 frames per second on the baseline device

4.2.2 Security Requirements

a. Protecting the application data.

b. Make sure that the application data should not be stealed.

4.2.3 Usability Requirements

a. Ensure each navigation works like how they should

b. Ensure that user interface is user friendly and intuitive.

4.2.4 Reliability Requirements

a. The system should not crash on common use cases

b. Connection issues should resolve cleanly and display errors on screen

4.2.5 Maintainability Requirements

a. Project source code should follow OOP paradigm

b. Project source code should utilize JSDoc comment structure

c. Developers should utilize the GitHub repository for version control

4.2.6 Portability Requirements

a. The system will be portable to any device with WebView, such as Android, iOS, and most desktop devices

4.3 Interface Requirements:

4.3.1 User Interface:

A frontend will be created with html, CSS and Java Script

Initially, the game can be started by giving a player name, after inputting the player's name it leads to the new page where the player can select the controls and the sounds of the game and start playing the game. It also consists of an exit button if the player wants to leave.

The hex game is the game that takes place within the hexagonal grids. The user is considered as a player who continuously moves towards the enemies and kills them respectively. This is a one-on-one play where the player moves towards the enemy by selecting the path and attacks the enemy after the player kills the enemy the next one starts attacking. If the player defeats all the enemies in level one and heads to the next level. For each kill, the player can gain a certain amount of score.

The goal of the game is to kill the enemy who was placed in random places on the hexagonal grid. The game gets tougher while getting to the next levels.

4.3.2 Hardware Interface:

Our application can be accessed and can be opened in devices like laptops, computers as well as mobile phones.

4.3.3 Software Interface:

a. Backend

JAVA SCRIPT

b. Development Languages:

HTML,CSS,JAVASCRIPT

5. Test Cases:

5.1 Test Cases for phase 1

List a set of test cases used for testing the working program including descriptions of tests (e.g., what functionality they test, and inputs/outputs for them).

Test case:

1. Test all hexagonal grids, walls and characters
   - Make sure the characters can move to each and every hexagonal
   - No character is moving outside the grid
   - When two characters came across make sure they don't pass through each other
2. Test the main page
   - Check all the buttons and their functionality
   - Make sure all the buttons serve their purpose
3. Test for health bar
   - Check if the health is being reduced for every attack
   - Make sure both players and enemy are attacking each other
   - No two enemies should fight with each other
4. Test the controls
   - Test every possible control button
   - Make sure character moves according to the controls
   - Test the whole GUI (colours, buttons, appearance, etc,.)

5.2 Test Cases for phase 2

List a set of test cases used for testing the working program including descriptions of tests (e.g., what functionality they test, and inputs/outputs for them).
Test case:

1. Test all hexagonal grids, walls, player, and enemy

- Make sure the enemy can move up, down, left, right by computer.
- No enemy is moving outside the grid, if they meet the border, they will move to opposite direction.
- Check the setup of different enemy with different score in different level.
- Check if the player restart at one hex when it loses its life.

2. Test the player and enemy attack attribute

- Make sure the number of attack and the possibility of attack successfully could make right function (Ex:input different attribute value ,output different damage value)
- Check the attack attribute of player and enemy have different value in different level.

3. Test the main page

- Check the level choose button and page
- Check all the buttons and their functionality
- Make sure all the buttons serve their purpose

4. Test for health bar and score

- Check if the health is being reduced for player loose in the attack with enemy.
- Check if the health is reduced when the enemy touched player first.
- Check if the score is being increase when player destroyed one enemy.
- Make sure the number of the adding score is right with different enemy destroyed.

5. Test the sound

- Check if the sound of the different background work normally

6. Test the controls

- Test every possible control button
- Make sure character moves according to the controls
- Test the whole GUI (colours, buttons, appearance, etc,.)

5.3 Test Cases for phase 3

List a set of test cases used for testing the working program including descriptions of tests (e.g., what functionality they test, and inputs/outputs for them).
Test case:

1. Test all hexagonal grids, walls, player, and enemy

- Check the height system and editor

2. Test the player and enemy attack attribute

- Check the attack cooler combat animations, every attack will have the number of attack animations.
- Check the unit's health bar changes colour with damage

3. Test the main page

- Check user login in to play game

4. Test for health bar and score

- Check the units health bar changes colour with damage

Unit test:

Input: attack = 1, defense = 1 ,damage = 0

Output(consloe.log)

Rolling Attack: Hit

Rolling Attack: Miss

Rolling Defense: Block

Rolling Defense : Fail

Damage = 0 or Damage = 1

Health Bar colour will change with damage.

If (Rolling Attack: Hit and Rolling Defense: Block):

Damage = 0; Health Bar colour don't change

If (Rolling Attack: Hit and Rolling Defense: Fail):

Damage = 1; Health Bar colour change

If (Rolling Attack: Miss and Rolling Defense: Block):

Damage = 0; Health Bar colour don't change

If (Rolling Attack: Miss and Rolling Defense: Fail):

Damage = 0; Health Bar colour don't change

5. Test the controls

- Test every possible control button

- Make sure character moves according to the controls
- Test the whole GUI (colours, buttons, appearance, etc.)

5.4 System tests

When we finished the program, we tested function and non-function program.

- Input ‘npm i’ and ‘npm run server’, program can run in localhost
- User uses google account to login in game
- User choose to play game
- User control player to move up, down, left, right
- Enemy could move up, down, left, right
- When player move to enemy, he can attack and destroy the enemy, healthy bar would change
- User could choose different level to play
- The program could run in different device

6. Features have successfully implemented and any limitation

Implemented:

1. Created the hex map
2. Created the characters of player and enemies
3. Created the animation of characters moving
4. Created the health bar of player and enemy
5. Completed the player attack with enemy
6. Completed the change of health bar of enemy are attacked
7. Completed the background sounds

Limitation:

1. The enemy moving is not so smart

   The enemy moving is control by computer, it just moves up, down, left, right, or static. They can’t move smart. We considered that enemy moving should connect with player, they should find the shortest path with player in the map. But considering the algorithms of finding shortest path and request of the huge amount of calculation in real time. We gave up this thought. We could consider it in the next phase.

7. Brief reflection on what has been accomplished (Phaser 3)

1. Unit health bar
    - Created the units health bar changes colour with damage.

2. Enemy move & attack
    - Add the animations of attack, it will display the animation of the number of damage when player attack
    - Completed units are restricted on where they can move
3. User login
    - Add user use google account to login
4. Control
    - Add some more control
    - Created tile height system and editor

8. Member Contribution Table:

Member	Contribution description	Overall Contribution (%)
Jeremy Glebe	Completed user login, height system and editor, heath bar change with colour and add some control	60
Haiyi Wang	Change colour of the health bar and size of damage, Completed the deliverable 5 and representation slide.	40

9. **References**

https://photonstorm.github.io/phaser3-docs/