QuizIsFun.com
Online Educational Game Tool

Supervisor: Lau Vincent
Member: Leon Yip Long Yin (3035270745)
Submission Date: 04/13/2019
Abstract

Education with games has become more popular. Children love games and games can definitely become a useful tool for education. There are popular educational games and tools which has over 10 million of monthly players. At that same time, school has been thinking of new ways to make education fun. However, there exist no educational tools which help primary teachers to do dictations and quizzes in a fun way. Therefore, our team decides to develop a gamified educational tool which allows teachers to use games to organize quizzes and dictations in a fun way.
# Table of Contents

Abstract .......................................................................................................................... 2
List of Figures .................................................................................................................. 4
List of Tables ................................................................................................................... 4

1. **Introduction** ........................................................................................................... 6
   1.1. Background ........................................................................................................... 6
   1.2. Previous Work ..................................................................................................... 7
   1.3. Objective ........................................................................................................... 10
   1.4. Outline of the Report .......................................................................................... 11

2. **Methodology** ......................................................................................................... 12
   2.1. Client Side .......................................................................................................... 12
   2.2. Server Side ......................................................................................................... 17
   2.3. Production System ............................................................................................. 19
   2.4. Gameplay Ideas .................................................................................................. 21

3. **Progress** ............................................................................................................... 23
   3.1. Stage One ........................................................................................................... 23
   3.2. Stage Two .......................................................................................................... 25
   3.3. Schedules and Milestones .................................................................................. 42
   3.4. Future Plans ....................................................................................................... 43

4. **Limitations and Difficulties Encountered** .............................................................. 44
   4.1. Memory Leak Problem ....................................................................................... 44
   4.2. High Network Problem ...................................................................................... 44
   4.3. High CPU Usage Problem ................................................................................ 45
   4.4. Poor Graphics Problem ..................................................................................... 45
   4.5. Data Structure Design Challenge ...................................................................... 45

5. **Conclusion** ............................................................................................................. 46

Reference ......................................................................................................................... 47
List of Figures

• Figure 1. Homepage of Education.com
• Figure 2. Game Page of Education.com
• Figure 3. Homepage of CoolMathGames
• Figure 4. Game Page of CoolMathGames
• Figure 5. Kahoot during Quiz
• Figure 6. Kahoot during Answer Review
• Figure 7. Ping Delay / Latency
• Figure 8. Preview of Graphic Style
• Figure 9. Stage One: User Interface (Main Menu) – Draft
• Figure 10. Stage One: User Interface (In-Game) – Draft
• Figure 11. Stage One: User Interface (In-Game Classmate Chatting) – Draft
• Figure 12. Stage One: User Interface (In-Game Teacher Panel) – Draft
• Figure 13. Stage One: Graphical Representation of Player Controls – Draft
• Figure 14. Collision Detection Function – Sample Code
• Figure 15. Get Distance Function – Sample Code
• Figure 16. Movements of Object – Sample Code
• Figure 17. Object Animation and Status – Sample Code
• Figure 18. File System Hierarchy of Production Server (/root) – Demo
• Figure 19. File System Hierarchy of Production Server (/root/client) – Demo
• Figure 20. Active Processes in Production Server – Demo
• Figure 21. Stage Two: User Interface – Main Menu
• Figure 22. Stage Two: User Interface – Log In / Sign Up Panel
• Figure 23. Stage Two: User Interface – Join Room Panel
• Figure 24. Stage Two: User Interface – Create Room Panel
• Figure 25. Stage Two: User Interface – Load Question Set Panel
• Figure 26. Stage Two: User Interface – Start Game Panel
• Figure 27A. Stage Two: User Interface – In-Game (Student)
• Figure 27B. Stage Two: User Interface – In-Game (Teacher)
• Figure 28. Stage Two: Get Player Progress Function
• Figure 29. Stage Two: Save Player Progress Function
• Figure 30. Stage Two: Is Valid Password Function
• Figure 31. Stage Two: Add User Function
• Figure 32. Stage Two: Is Username Taken Function
• Figure 33. Stage Two: Data Structure For Rooms
• Figure 34. Stage Two: Functions For Creating Question Sets
• Figure 35. Question Set Example In Client
• Figure 36. Example Table in MongoDB - Account
• Figure 37. Example Table in MongoDB - Progress
• Figure 38. Inside “allQSets” of Progress
• Figure 39. Example of Biology Quiz Data Structure
• Figure 40. Example of English Dictation Data Structure
• Figure 41. Example of Math Quiz Data Structure

List of Tables

• Table 1. Global Market Share of Desktop Internet Browsers (2015 – 2018)
• Table 2. Project Schedule
1. Introduction

1.1 Background

There has been an increase in popularity for playing online games on the Internet. Gaming in both mobile and PC (personal computer) market has been growing rapidly and it is estimated to have reached an astonishing number of 2.6 billion audience (The UK Interactive Entertainment Association, 2017). This shows that games are very popular and are widely accepted by people.

Along with the growing trend of gaming, there is also a new type of games which can be played in browser, namely HTML5 games. Some of the popular games are Agar.io and Slither.io. For instance, “Agar.io” was the seventh search term in Google (Jon, 2016), which was slightly behind "Ronda Rousey" and "Paris". For both Agar.io and Slither.io, there were more than 300 million monthly players during their peak (Rank2Traffic, 2018) (Rank2Traffic, 2018), as well as countless of YouTube videos related to them having over 10 million views. It is clear that HTML5 games are being a trend and people enjoy playing them.

In addition, gamifying education is getting more and more popular. More people, including businessmen and teachers, are starting to integrate education with gaming in order to make learning more fun for children. A growing opportunity in education gamification is observed. Some of the notable examples of education gamification are Education.com, CoolMathGames and Kahoot. Interestingly, most of the educational games are created with HTML5, which can be explained with the advantages of HTML5.

Apart from its huge popularity, HTML5 applications are also great at being cross-platform and cross-device. In order words, the same HTML5 applications can be used in your smartphones, Desktop or Laptop computers. The cost of maintenance and update is efficient compared to traditional applications where a different version is needed for each device.

In this project, we aim to realize the possibility of games in assisting traditional education for primary school. In other words, we will be creating educational games as tools for teachers during lessons. The educational tools will be specifically used for quizzes and dictations, to make them more fun.
1.2 Previous Work

Currently, there are existing gamified educational tools available in the market for children. Some of the most notable ones are Education.com, CoolMathGames and Kahoot.

Education.com is an online platform with hundreds of educational games. It is one of the most popular educational site for pre-kindergarten to primary-five children, with over 10 million monthly users. The website includes about 400 games of different subjects, such as Math (e.g. addition, subtraction, multiplication), English (e.g. reading, writing, typing) and coding. Parents or teachers are required to pay a monthly membership fee in order to allow children to use the service. For children, they can go to the website and play any educational games they like. As for teachers and parents, they can track the progress and history of children, as well as assigning games in the website as homework.

Figure 1. Homepage of Education.com

![Figure 1. Homepage of Education.com](image1)

Figure 2. Game Page of Education.com

![Figure 2. Game Page of Education.com](image2)
CoolMathGames is a website with hundreds of educational mathematics games and has over 20 million monthly users. The games in the website are all web games made with either HTML5 or Flash. Mathematical topics including pre-algebra, algebra and pre-calculus are covered. Unlike the previous example, this website is completely free and relies on advertisement for revenue.

*Figure 3. Homepage of CoolMathGames*

![CoolMathGames Homepage](image)

*Figure 4. Game Page of CoolMathGames*

![CoolMathGames Game Page](image)

However, the above two examples aren’t multiplayer. Teachers cannot use them to interact with students during lessons. They are more like homework where students will complete the work by themselves. Our project will allow teachers and students to see each other in the game.

Moreover, the above examples don’t allow teachers to create their own questions, thus won’t be customized to individual teachers’ syllabus. On the other hand, our project allows teachers to create their own set of questions and correct answers.
Kahoot is a game-based learning platform, used as educational technology in schools and other educational institutions. It has over 40 million monthly users and is used from teaching primary children to lecturing university students. It provides multiple-choice quizzes with interactive gameplay for teachers to use during classes. There are score trackers for each student during each quiz, as well as a leaderboard system to check the ranking of students. Kahoot has both web and mobile version.

Figure 5. Kahoot during Quiz

![Kahoot Quiz](image1.png)

Figure 6. Kahoot during Answer Review

![Kahoot Answer Review](image2.png)

Out of the above examples, Kahoot will be the most similar to our project as both of us will be educational technology to assist teaching during classes. However, our project will be more gamified, more appealing to children and has more features.

In addition, Kahoot won't have as many features as our project. Kahoot only have Multiple Choice questions while my project will have more question types.
1.3 Objective

The project aims to develop a fun educational technology for primary education, to provide a fun way for dictation and quiz. In other words, a game for dictation and quiz.

1.3.1. Real-Time Multiplayer

The game will be real-time multiplayer, which means players can see others’ actions with minimal delay. Players will be able to interact with each other in different ways. The server should be able to handle 100 concurrent users.

We aim to reduce the ping delay (i.e. time needed for a message to travel from user’s computer to the server) to less than 0.5 second. This can be achieved by optimizing algorithm and data transfer.

*Figure 7. Ping Delay / Latency*

1.3.2. Cross-Browser Compatible

Users should be able to access the game with major web browsers and with different screen size. This is to make sure the game is accessible by 97.44% of the market: Google Chrome (67.66%), Firefox (10.96%), Internet Explorer (6.97%), Safari (5.13%), Microsoft Edge (4.24%) and Opera (2.48%).

*Table 1. Global Market Share of Desktop Internet Browsers (2015 – 2018)*

1.3.3. Children Friendly

The user interface and game theme will be of 2D cartoonish graphic style as our target audience will be children of age between 5 and 13. We will make sure our game is attractive to children in terms of graphics and gameplay.

In addition, extra effort will be made to make sure the content and graphics of the game will be suitable for children. Inappropriate scenes (e.g. violence, blood) will not be allowed.
Below is a preview of the graphic style (Figure 8) we are going to use for the project.

*Figure 8. Preview of Graphic Style*

### 1.4. Outline of the Report

In this report, we will discuss the progress of the project. First, we will discuss the background, previous work and objective of the project in the introduction section. Following with the methodology and technical details of the project. The overall progress of the project will be discussed, as well as a section related to difficulties and challenges. Lastly, we will conclude the progress report.
1. Methodology

The project is divided into three major components.

2.1. Client Side

This component will be responsible for presenting the game to the user, with the help of user interface, image and graphical representation of the game objects (e.g. monsters and terrains) and functions (e.g. health bar). All of the game images, graphic and sound will also be here for optimal performance.

I. User Interface

As seen in Figure 9 below, there will be some buttons on top-left and top-right corner in the User Interface. Those buttons will be responsible for basic functionalities such as toggling of music and sound effects, full screen option, switching of servers, etc.

A white box is placed in the center for players to choose their desired name. As well as a big and clear play button to speed up navigation and thus improve UX (i.e. user experience).

We will be using simple-looking graphical icons, which synergizes with the 2D cartoonish background of the game.

*Figure 9. Stage One: User Interface (Main Menu) – Draft*
Figure 10. Stage One: User Interface (In-Game) – Draft

Figure 11. Stage One: User Interface (In-Game Classmates chatting) – Draft

Figure 12. Stage One: User Interface (In-Game Teacher Panel) – Draft
II. Graphical Representation

A number of graphical representations is also presented in the game. This is to keep the simplicity of the game, as well as performing their respective functions effectively and efficiently such as player controls in Figure 10.

Other graphical representation such as Health Bars, Experience Bars and Loading Icons will also be implemented.

Figure 13. Stage One: Graphical Representation of Player Controls – Draft

III. Music and Sound Effects

We will be adding sound effects to the game. This will help improve the quality of the game and make the game more realistic with sound. Sound that are appropriate for the game will be used and they should correspond with the animation and graphics of the game. There will be sound effects for both in-game objects (e.g. throwing items, magic wands when casting magic) and user interface objects (e.g. buttons when clicked, icons when hovered). For instance, sound of magic should be played at the exact moment when player presses the cast-magic button. At the same time, magic should be casted. For user interface objects, again, sounds will have to play at the precise moment when a certain event is triggered (e.g. clicking a button). Any misalignment will make the game look unprofessional. This not be tolerated as it will have a negative effect on game quality and user experience.
In addition, various music and soundtracks will be added to the game. Different soundtrack will be played in different scenarios. For example, there will be a unique soundtrack for Main Menu, and several other soundtracks for different places (e.g. marketplace, beach). The music should be chosen such that the atmosphere of that scenario is perfectly reflected by the music, such as a joyful music for a party place and an exciting music for a battleground. When well implemented, music can help bring user experience to a new level and improve gameplay drastically. When switching scenarios, there should be a smooth transition from one soundtrack to another, so that player experience will not be affected. When music is about to be switch, there will first be a decrease in volume of the current music, following by an increase in volume of the new music. In other words, the current music will fade away naturally and then the new music will be introduced. The same technique should also be used when a song finishes playing and a new one is needed to be played.

IV. Input and Output System

There are mechanisms for the client side to collect player inputs (e.g. keyboard buttons and mouse) and send to the server for computation and then finally sending the information back to players (i.e. client side). In order words, data will be sending and receiving between client side and server. Examples of server computation include:

- After player A presses “left”, will there be any obstacles like trees on left of player A which will prevent him from moving?
- After player A presses “left”, should player B be able to see or attack him?

For details regarding server computations, we will cover them in section 2.2 Server Side. From the above example, we can see that a simple “left” movement will take a lot of computations and optimizations. After the game collects player inputs and completed the required computations, it will send back information to all of the players to notify them the changes.

There are other input and output systems, including: Movement in other directions, clicking of buttons, chat, throwing items, etc. The system will mainly be created by Socket.io.
VI. Position System for Objects

As this is going to be a 2D game, the system we are going to use to represent the position of objects will be (x,y), where a larger x means “more” right and a larger y means “more” down. This will be different with normal graphs in mathematics where a larger y usually mean “more” up. Both methods for “y” will be fine, but our choice would be better suit for web games as most websites usually scroll down for more content instead of up. In this way, when we add new game contents, y will be of positive value instead of negative.

VI. Implementation

A mix of different programming languages will be used to create the client side. The programming languages include: JavaScript, HTML and CSS.

Images and graphical designs will be created and modified with Paint.net, a professional software for image editing.

Music and sound effects will be edited with Audacity, a free-to-use, open source, professional audio editor and recorder.
2.2. Server Side

This module will be responsible for most, if not all, of the calculations, computations and algorithms of the game. Algorithms such as Collision Detection and Physics System will be implemented and executed here. This is to ensure the correctness and uniformity of data, as putting all of the crucial calculation of data in one place will remove all sort of error resulted by delay.

I. Collision Detection

This part of code will be used most often throughout the game. Collision is essential for almost any functions, from player touching a stone, to shooting bullets at enemies, to feeding food to animals.

Therefore, the code must be written in high quality, in terms of code readability (i.e. easy to read by the team and other programmers), reusability (i.e. easy to reuse the code in other places instead of creating another similar code) and maintenance (i.e. easy to be modified, updated and managed).

A simplified version of the code is shown in Figure 11 and Figure 12.

*Figure 14. Collision Detection Function – Sample Code*

```javascript
self.checkCollision = function(a,b){
    if(a.map === b.map && a.id !== b.id && a.getDistance(b.x,b.y) < Math.pow(a.radius+b.radius,2))
        return true;
    return false;
}
```

*Figure 15. Get Distance Function – Sample Code*

```javascript
self.getDistance = function(x,y){
    return Math.pow(self.x-x,2) + Math.pow(self.y-y,2);
}
```
II. General Physics

The physics of the game is also important. The set of physical rules will determine how fast an object can move (e.g. speed limit), how objects will interact with each other (e.g. animals get hurt when touching lava), how objects will behave in the environment (e.g. gravity), etc.

**Figure 16. Movements of Object – Sample Code**

```javascript
self.changeDirection = function(angle){
    self.spdX = Math.cos(angle/180*Math.PI) * self.spd;
    self.spdY = Math.sin(angle/180*Math.PI) * self.spd;
    self.angle = angle; //for rotating image
    if (self.angle<0)
        self.angle+=360;
}
```

**Figure 17. Object Animation and Status – Sample Code**

```javascript
self.changeAnimation = function(type,second){
    if (self.animationCounter<second*FPS){
        self.animation=type;
        self.animationCounter=second*FPS;
    }
}

self.toggleStatus = function(type,second){
    if (type==1 && self.statusShieldCounter<second*FPS){ //shield
        self.statusShield=1;
        self.statusShieldCounter=second*FPS;
    }
}
```

III. Implementation

A mix of different programming languages and tools will be used to create the server side, including Javascript, Node.js, Socket.io, Express and MongoDB.

We have chosen Javascript over other programming languages, such as C++, for building backend server. This is because Javascript is compatible with Node.js, which is easy, efficient and has many useful tools for building web applications like Socket.io and Express.
2.3. Production System

This component will be responsible for keeping the game available to the public. The production system’s job is to ensure the 100% uptime of the game, as well as the game running smoothly without problems.

I. File System Hierarchy of Production Server

Tens of files and folders will be used in the production system. They will be responsible for storing all the required files, such as text (.txt, .js, .html), images (.png and .jpg) and sound (.mp4) files. Figure 15 and Figure 16 show a demo of the file system hierarchy.

**Figure 18. File System Hierarchy of Production Server (/root) – Demo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Changed</th>
<th>Rights</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td></td>
<td></td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.cache</td>
<td></td>
<td>11/15/2018 7:40 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.gnupg</td>
<td></td>
<td>9/25/2018 2:44:42 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.npm</td>
<td></td>
<td>9/25/2018 2:44:40 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.pm2</td>
<td></td>
<td>10/17/2018 10:49:34 AM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.ssh</td>
<td></td>
<td>10/17/2018 10:49:52 AM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.cache</td>
<td></td>
<td>9/25/2018 2:50:26 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.client</td>
<td></td>
<td>11/25/2018 9:59:09 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>node_modules</td>
<td></td>
<td></td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.bash_history</td>
<td></td>
<td>11/18/2018 3:18:33 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.bashrc</td>
<td></td>
<td>4/9/2018 7:10:28 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.profile</td>
<td></td>
<td>8/17/2015 11:30:33 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.app.js</td>
<td>60 KB</td>
<td>11/18/2018 2:51:19 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.app2.js</td>
<td>60 KB</td>
<td>11/18/2018 2:51:37 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
</tbody>
</table>

**Figure 19. File System Hierarchy of Production Server (/root/client) – Demo**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Changed</th>
<th>Rights</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
<td></td>
<td>10/17/2018 10:49:31 AM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.img</td>
<td></td>
<td>11/16/2018 10:12:07 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.sound</td>
<td></td>
<td>10/23/2018 6:21:47 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.ads.txt</td>
<td>4 KB</td>
<td>10/23/2018 7:23:33 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.immediate.html</td>
<td></td>
<td>11/21/2018 1:42:32 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.index.html</td>
<td>108 KB</td>
<td>11/25/2018 9:35:37 PM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.rocket.js</td>
<td>96 KB</td>
<td>8/5/2017 11:32:34 AM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
<tr>
<td>.style.css</td>
<td>5 KB</td>
<td>10/20/2018 1:28:19 AM</td>
<td>rw-r--r--</td>
<td>root</td>
</tr>
</tbody>
</table>
II. Active Processes in Production Server

The production system consists of a Virtual Private Server (Linux Ubuntu 18.04), with some running software including Nginx, Process Manager (PM2) and MongoDB. Several active processes (i.e. running applications) are required in the production server. Figure 17 shows a demo of the processes.

![Figure 20. Active Processes in Production Server - Demo](image)

III. Domain Name

Lastly, a domain name will also be included. This is essential for the publishing of the project. Currently, the official name and domain name of the project has not been decided.
2.4. Gameplay Ideas

Below are gameplay ideas which are not yet implemented. There will be 3 game modes, including:

- Fill in the Blanks
- Multiple Choice Questions
- True or False Questions

Of course they will be "gamified", details will be explained below.

I. Fill in the Blanks

- Teacher and Students enter the game (same as the demo), let's say we have 1 teacher and 20 students.
- Teacher say one question, for example:
  - "5 x 6 = "
  - "I _ a teacher. (am/is/are)"
- Students can type the answer in the chat.
  - If answer correctly, they get score.

II. Multiple Choice Questions

- Teacher and Students enter the game (same as the demo), let's say we have 1 teacher and 20 students.
- Teacher say one question, for example:
  - "46 + 21"
  - "I _ a teacher. (am/is/are)"
- There will be different "zones" representing different answers.
- Students can move their character to the "zone", this means they select that answer.
  - If answer correctly, they get score.
III. True of False Questions

- Teacher and Students enter the game (same as the demo), let's say we have 1 teacher and 20 students
- Teacher say a statement which is either True Or False, for example:
  - "46 + 21 = 1"
  - "Bicycl" (This is a wrong English spelling of bicycle)
- This statement will become a monster
- Students have a "True" gun and a "False" gun to shoot at the monster
  - If statement is true, use "True" gun. If not, use "False" gun
  - If answer correctly, they get score

IV. Advantages of the game

- Make quizzes and dictations more fun
- A tool for educational class activity
- Allow teachers and students to interact closely when doing quiz=dictation
- Make quiz and dictations competitive
- Kids will have more motivation to study

V. Content Generated By Users

As the questions and answers will be created by users (i.e. teachers or parents), there will be no limitations with the subject of the tool. In other words, users can create gamified quizzes about Math, English and other subjects as they wish.

- Teachers
  - Create a new “room” and the user will receive the “teacher” role
  - Teacher can create questions, set correct and wrong answers, etc
  - After teacher clicks Submit button, in-game objects will be created and game will start. Depends on the game mode, objects can be monster, zones, etc
- Students
  - Enter code to join a “room”
2. Progress

Now, we will discuss the progress of the project.

3.1. Stage One

During Stage 1 of the project (i.e. semester 1), the following work of the project are completed.

I. Background Research

Background research about the browser gaming industry, as discussed in the introduction section, is done. An initial understanding is obtained, and we now have a clearer picture on the latest gaming trends and market demand. This helps solidifying the direction of the project.

II. Game Conceptualization

Graphical design (i.e. user interface, icons, images) were conceptualized, drafted and semi-implemented as a demo, as shown in Section 2.1. After we have a clear direction on the design of the game, we will be able to continue with the implementation.

Features and functions of the game were decided. For example, game modes and their respective gameplays were conceptualized, as discussed in Section 2.4.
III. Implementation

- **Client Side**

  70% of the client-side implementation is completed, with over 2,700 lines of code, tens of images and graphical icons. Details mentioned in Section 2.1 are completed works.

- **Server Side**

  40% of the server-side implementation is completed, with over 2,200 lines of code. Many optimization works have been done, including code and algorithm improvements. Work included game functions and computations, physics of the game, objects, input and output systems and optimization. Details mentioned in Section 2.2 are completed works.

- **Production System**

  80% of the production system is completed. Details mentioned in Section 2.3 are completed works.

- **Database System**

  0% of the database system is completed. Database system will be built in Stage 2.
3.2. Stage Two

During Stage 2 of the project (i.e., semester 2), which is also the final stage, the following work of the project are completed.

I. Client Side

100% of the client-side implementation is completed, with over 3,200 lines of code, tens of images and graphical icons.

- User Interface
  - Main Menu
    Includes:
    1. Buttons (Log In, Sign Up, Create Room, Join Room)
    2. Functional Buttons (Fullscreen, Music, Sound)
    3. Input Field (Nickname)
    4. Select Character

Logic:
When any input from users is registered, such as clicking of buttons or selecting characters, data will be sent from client to server via socket communication with the help of Socket.io. Data is first transformed into JSON object, and then sent to the server.

After receiving the JSON object, the server will validate the correctness of the data, such as the type (e.g., string, boolean). After validation, the server will use the data to perform functions such as Create Room and Sign Up (details of the functions will be explained in the Server Side section).

After the operations, the resulting data will be sent back to client if needed. Again, the data will be converted into JSON object, and then received by client.
Design:
Most of the User Interface design are done with CSS, HTML5, Document Object Model (DOM) and JavaScript. As well as PNG images for graphics and icons, and MP3 sound for sound effects and music.

CSS are mostly responsible for the color, placement and style of the UI objects (e.g. div). Certain animation such as the “shakiness” of the QuizIsFun.com logo is also done by CSS. Some of the most often used CSS methods are:

- margin, padding, width, height, background-color
- position, top, bottom, left, right
- color, font-size, text-decoration, text-align
- mouse hover

JavaScript are used for the client-side programming. Functions such as showing/hiding certain div, going to the next section (e.g. Log In, Create Room), using cookie sessions and cache, etc.

Figure 21. Stage Two: User Interface – Main Menu
Log In / Sign Up Panel

Includes:

1. Input Field (Email, Password)
2. Buttons (Log In, Sign Up)

Notes:

1. After clicking “Log In/Sign Up” button, the JSON object which contains the values of Email and Password will be sent from client to server.
2. For “Log In”, server will respond with one of the following:
   - Log In Successfully
   - Wrong Email/Password
3. For “Sign Up”, server will respond with one of the following:
   - Sign In Successfully
   - Email/Password Length Too Short
   - Email Taken

Figure 22. Stage Two: User Interface – Log In / Sign Up Panel
- **Join Room Panel**
  
  **Includes:**
  1. Input Field (Enter Code)
  2. Button (Join Room)
  
  **Notes:**
  1. After entering the code and click “Join”, the JSON object which contains the value of Code will be sent from client to server.
  2. Server will check if the game with the respective Code exists.
  3. Server will respond with one of the following:
     - Join Room Successfully
     - Room Does Not Exist

*Figure 23. Stage Two: User Interface – Join Room Panel*
Create Room Panel

Includes:

1. Input Fields (Question Set Name, Questions, Choices)
2. Grey Buttons (Delete Question)
3. Blue Buttons (Question Type, MC = Multiple Choice, T/F = True or False)
4. Answer (Green “O” = Correct, Red “X” = Wrong)
5. Functional Buttons (Add Question, Create Room, Save, Load)

Notes:

1. Users can create as many Questions as they want, there is no limit.
2. Click “Save” to save the current Question Set.
3. Click “Load” to view all of the users’ created Question Sets.
4. Click “Create Room” to use the current Question Set and start a game.

Figure 24. Stage Two: User Interface – Create Room Panel
Load Question Set Panel

Includes:

1. Buttons (Question Set Name, Number of Questions)
2. Grey Buttons (Delete Question Set Permanently)
3. Scrollbar, in case there are many Question Sets

Notes:

1. Users can load Question Sets
2. Users can delete their selected Question Set, this action is permanent

Figure 25. Stage Two: User Interface – Load Question Set Panel
Start Game Panel

Includes:

1. Room Code (Tell others the code to join)
2. Button (Start Game)

Notes:

1. Others can join this room if they enter the code, which is 2104 in the example below
2. When User clicks “Start Game”, all Users in this room will start their game

Figure 26. Stage Two: User Interface – Start Game Panel
o In-Game (After Game Start)

Includes:

1. Question, Answer Choices, Timer
2. Leaderboard
3. Current Question, Total Number of Questions in the game
4. Minimap
5. Teacher (Pause/Fast-Forward the game)

Notes:

1. Interact with Objects (e.g. pig, rabbit) in the game to change your answer
2. After each question ends, players with the correct answer gets +100 score
3. When players get certain score, they unlock new items (e.g. Snowman, Chocolate)
4. Users with the highest scores will be shown in Leaderboard
5. Teacher can pause the game or fast-forward to aid teaching

Implementation of the game client:

The client-side of the game has a framerate of 60 frame per second (fps), which means any objects in the game update their positions and perform actions 60 times per second. We have used the technique of “requestAnimationFrame();”, which is a JavaScript method to consistently perform actions, like a setInterval loop. As a result, objects will be drawn in the game client 60 times per second with the loop, and with the most updated object information.

The server-side framerate is about 20 fps, so we have used the Interpolation technique to map the 20 fps of server into the 60 fps of client. The mapping is done by creating intermediate object states in client, which is illustrated by the example below:
<table>
<thead>
<tr>
<th>Time</th>
<th>Object info in Server</th>
<th>Object info in Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>t = 0</td>
<td>(x,y) = (0,0)</td>
<td>(x,y) = (0,0)</td>
</tr>
<tr>
<td>t = 1</td>
<td>Absent</td>
<td>Need to do calculation</td>
</tr>
<tr>
<td>t = 2</td>
<td>(x,y) = (2,2)</td>
<td>(x,y) = (2,2)</td>
</tr>
</tbody>
</table>

For the client-side at $t = 1$, we can find the value of (x,y) by doing $(2/2,2/2) = (1,1)$. Thus, even with the incomplete information from server-side, we can compute additional information in client-side. This is essential as not only will server resources be minimized; the user experience of the UI can also be maximized. In other words, we can get faster animation even with a slower server.

Another important function for the client-side is to receive all crucial information from the server, which include data of all objects in the screen (e.g. position, animal type), room information (e.g. time, question). After receiving the information, they will be stored in the client-side. Thus, computations in later times can be much faster as all the required data are already stored in the client-side.

*Figure 27A. Stage Two: User Interface – In-Game (Student)*
For the case of Teachers, there are two extra buttons to help with teaching, Pause and Fast-Forward buttons. The teacher is also represented by the red flag. Teachers can pause the game for as long as they wish. And to unpause, simply click the Pause button again and the game will resume. The Fast-Forward button will skip to the next question, so it will be of good help to the teacher there is not enough time in class.

**Figure 27B. Stage Two: User Interface – In-Game (Teacher)**
II. Server Side

100% of the server-side implementation is completed, with over 2,200 lines of code. Many optimization works have been done, including code and algorithm improvements. To recap, the following are work done in Stage 1:

- Game functions and computations
- Physics of the game
- Objects data structure and functions
- Input and output systems
- Optimization

Below are work in Stage 2:

- Infrastructure for communication between Client and Server
  - Similar implementation with Stage One
- Infrastructure for communication between Server and Database
  - Function to get user data from Database

*Figure 28. Stage Two: Get Player Progress Function*

```javascript
var getPlayerProgress = function(email, cb){
  db.progress.findOne({email:email},function(err,res){
    cb({allQSets:res.allQSets});
  });
}
```

  - Function to save user data to Database

*Figure 29. Stage Two: Save Player Progress Function*

```javascript
var savePlayerProgress = function(data, cb){
  cb = cb || function(){
    db.progress.update({email:data.email},data,{upsert:true},cb);
  }
}
• Infrastructure for Users to Log In
  o Function to check if the password is correct

**Figure 30. Stage Two: Is Valid Password Function**

```javascript
var isValidPassword = function(data, cb) {
  db.account.findOne({email: data.email, password: data.password}, function(err, res) {
    if (res) {
      cb(true);
    } else {
      cb(false);
    }
  });
}
```

• Infrastructure for Users to Sign Up
  o Function to create the data for new users

**Figure 31. Stage Two: Add User Function**

```javascript
var addUser = function(data, cb) {
  db.account.insert({email: data.email, password: data.password}, function(err) {
    savePlayerProgress({email: data.email, allQSets: []}, function() {
      cb();
    });
  });
}
```

  o Function to check if the Email is taken

**Figure 32. Stage Two: Is Username Taken Function**

```javascript
var isUsernameTaken = function(data, cb) {
  db.account.findOne({email: data.email}, function(err, res) {
    if (res) {
      cb(true);
    } else {
      cb(false);
    }
  });
}
```
• Infrastructure for Load / Save Question Sets
  o Data structure for Rooms

**Figure 33. Stage Two: Data Structure For Rooms**

```javascript
socket.on('createRoomData', function(data){
    var room = {};
    room[0] = data.state; //data
    room[1] = socket.id;  //owner
    room[2] = 0;          //started (0=not started)
    room[3] = 0;          //leaderboardUsername
    room[4] = 0;          //leaderboardScore
    room[5] = 1;          //question now
    room[6] = {};         //list of players joined

    RoomList[data.roomCode] = room;

    Player.list[socket.id].room = data.roomCode;

    console.log(RoomList[data.roomCode]);
    socket.emit('createRoomDataResponse',{roomCode:data.roomCode});
    Player.list[socket.id].room = data.roomCode;

    RoomList[data.roomCode][6][socket.id] = socket.id;
});
```

  o JSON data are transmitted between Client and Server using Socket.io communication

**Figure 34. Stage Two: Functions For Creating Question Sets**

```javascript
socket.on('createQSetSave', function(data){
    Player.list[socket.id].allQSets = data.allQSets;
});
```

```javascript
socket.on('createQSetLoad', function(data){
    socket.emit('createQSetLoadResponse',{allQSets:Player.list[socket.id].allQSets});
});
```

III. Production System

100% of the production system is completed.

- MongoDB is set up in the Virtual Private Server successfully, with 100% uptime.
- The project is hosted in the Virtual Private Server with 100% uptime.
III. Database System

100% of the database system is completed. Functions such as Account System, Room Creation System and Question Sets System are utilizing the MongoDB database system.

It was a challenge to develop an effective data structure for storing Question Sets. Below are the requirements for the design:

- There can be 0-infinite Users
- There can be 0-infinite Rooms
- Each Room can have 0-infinite Users
- Each User can have 0-infinite Question Sets
- Each Question Set can have 0-infinite Questions
- Every data (e.g. Users, Questions) can be added/removed anytime

In practice, we will set a number limit for infinite, because of hardware and algorithm limitations.

We have visualized the data structure and put them in the following way:

- User (email, password, score, room)
- Question Sets (number of questions, name, owner)
- Questions (question number, question type, answers, correct choice)
- Rooms (owner, participants, time now, current question, question set used, room code)
- Objects (room)

We will have to use dynamic data structure to solve the problem. This is another advantage of JavaScript, where Object Lists can be used efficiently. Objects can be encapsulated within another Object and there are no limits to how many layers data are encapsulated. Object data will be sent between MongoDB and Server in JSON format.
Below is an example of the Graphical Representation of Question Set data in Client, and the exact same data but stored in MongoDB.

**Figure 35. Question Set Example In Client**

![Question Set Example In Client](image)

**Figure 36. Example Table in MongoDB - Account**

```json
{ "_id" : ObjectId("5cb18b1f1ad9a61308c0ce5d"), "email" : "example@email.com", "password" : "secretPassword" }
```

- Objectld = The Id of the Object
- Email = The email of the User, which is “example@email.com”
- Password = The password of the User, which is “secretPassword”
Figure 37. Example Table in MongoDB - Progress

```
{ "id" : ObjectId("5cb18b1eb9d56703852d2075"), "email" : "example@email.com", "allQSets" : { "Question Set 1" : { "1" : [ "1", "What is 1 + 1?", "2", "3", "4", "5", "1" ], "2" : [ "1", "What is 5 x 2?", "29", "30", "31", "32", "2" ], "3" : [ "1", "What is 8 - 7?", "0", "1", "2", "3", "1" ], "qSetName" : "Question Set 1" } } }
```

- ObjectId = The Id of the Object
- Email = The email of the User, which is example
- allQSets = The Object List which contains the data of the Question Sets created by the User

Figure 38. Inside “allQSets” of Progress

```
"Question Set 1" : { "1" : [ "1", "What is 1 + 1?", "2", "3", "4", "5", "1" ] }
```

- “Question Set 1” = Name of the Question Set
- “1” = Question Number (i.e. Question 1)
- “1” = Type of the Question, where 1 = Multiple Choice and 2 = True or False
- “What is 1 + 1?” = Question
- “2”, “3”, “4”, “5” = Answers (i.e. A = 2, B = 3, C = 4, D = 5)
- “1” = Correct Answer (i.e. [1,2,3,4] maps to [A,B,C,D] and [1,2] maps to [True,False])

Below are more examples to illustrate the structure of the Question sets:
**Figure 39. Example of Biology Quiz Data Structure**

- Biology Quiz (True/False):
  - 1: (7) ["1", "Spider is a type of insect.", "True", "False", "", "", "2"]
  - 2: (7) ["1", "Turtle is omnivore.", "True", "False", "", "", "1"]
  - 3: (7) ["1", "Omnivore does not eat vegetables.", "True", "False", "", "", "2"]
  - 4: (7) ["1", "[Bonus] Do you like Ms. Chan's class?", "Yes", "No", "", "", "1"]

- “Biology Quiz (True/False)” is the Name of the Question Set
- “1,2,3,4” are Question Numbers
- “Spider is a type of insect”, “Turtle is omnivore” are Questions for question 1 and 2
- “True”, “False”, “”, “” represent answers in question 1
- Because there are 2 empty strings, this means question 1 only have two choices, which are True and False
- The last integer represents the correct answer. For the case of question 1, “2” means “False” is the correct answer

Below are more examples of different type of quizzes that can be supported by our project.

**Figure 40. Example of English Dictation Data Structure**

- English Dictation (Animal):
  - 1: (7) ["1", "Which word is spelled wrongly?", "Sheep", "Cat", "Cow", "Dog", "1"]
  - 2: (7) ["1", """"Lion" is spelled correctly.", "True", "False", "", "", "1"]
  - 3: (7) ["1", "Which spelling is correct?", "Hippopotamus", "Hippo", "Potamus", "Hipotamus", "1"]
  - 4: (7) ["1", "Which animal lives under the sea?", "Rabbit", "Octopus", "Cat", "Mouse", "2"]
  - 5: (7) ["1", "Which spelling is correct?", "Elephant", "Elefant", "", "", "1"]

- qSetName: "English Dictation (Animal)"

**Figure 41. Example of Math Quiz Data Structure**

- Math Quiz (Multiplication):
  - 1: (7) ["1", "2 x 3 - ?", "4", "5", "6", "7", "2"]
  - 2: (7) ["1", "2 x 3 x 10 - ?", "60", "70", "80", "90", "1"]
  - 3: (7) ["1", "(2 + 3) x 4 - 1 - ?", "19", "20", "21", "22", "1"]

- qSetName: "Math Quiz (Multiplication)"
### 3.3. Schedules and Milestones

Below is the complete schedule of the project development.

**Table 2. Project Schedule**

<table>
<thead>
<tr>
<th>Date</th>
<th>Deliverables of Phase 1 (Inception)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sept 30</td>
<td>• Detailed project plan</td>
</tr>
<tr>
<td></td>
<td>• Project web page</td>
</tr>
<tr>
<td>Oct</td>
<td>Conceptualize the details of the game</td>
</tr>
<tr>
<td></td>
<td>• Gameplay</td>
</tr>
<tr>
<td></td>
<td>• Mechanics</td>
</tr>
<tr>
<td>Nov</td>
<td>GUI and graphic design</td>
</tr>
<tr>
<td></td>
<td>• In-game User Interface</td>
</tr>
<tr>
<td></td>
<td>• Main Menu and Other Sections</td>
</tr>
<tr>
<td></td>
<td>• Object Images</td>
</tr>
<tr>
<td>Dec</td>
<td>Construction</td>
</tr>
<tr>
<td></td>
<td>• Input/Output System of Players</td>
</tr>
<tr>
<td></td>
<td>• Basic Game Mechanics (e.g. movement, chat, collision)</td>
</tr>
<tr>
<td></td>
<td>• Construction of Objects in-game (e.g. monsters, weapons)</td>
</tr>
<tr>
<td></td>
<td>• Addition of Sound Effect and Music</td>
</tr>
<tr>
<td></td>
<td>• Production System</td>
</tr>
<tr>
<td>Jan 20</td>
<td>Deliverables of Phase 2 (Elaboration)</td>
</tr>
<tr>
<td></td>
<td>• Preliminary implementation</td>
</tr>
<tr>
<td></td>
<td>• Detailed interim report</td>
</tr>
<tr>
<td>Feb</td>
<td>Completion of Client Side, Server Side and Production System</td>
</tr>
<tr>
<td>March</td>
<td>Optimization and Testing</td>
</tr>
<tr>
<td>April 14</td>
<td>Deliverables of Phase 3 (Construction)</td>
</tr>
<tr>
<td></td>
<td>• Finalized tested implementation</td>
</tr>
<tr>
<td></td>
<td>• Final report</td>
</tr>
</tbody>
</table>
3.4. Future Plans

Below are the future planning for the direction and possibilities of the project.

I. Uploading Question Sets & Browsing Other Users’ Question Sets

For people who likes creating teaching materials and education, they can use this feature effectively. For example, one can use other’s public question sets and create a room instantly. With this feature introduced, systems such as Comments, Votes and Ratings for Question Sets can also be added later.

II. More Question Types

Currently, there are only True-or-False and Multiple-Choice questions. More question types, such as Fill-In-The-Blanks can be introduced.

III. More Customizations for Room Owners

Customizations, such as allowing Room Owners to change images of objects in the room, create new objects, and giving participants items (e.g. hat, cloths, weapons, toys, pets) for getting scores, can be added.

IV. Host In Public

After the project’s beta version is finished, it will be published to the Internet with an official domain name. The general public can create an account and start using the application. Teachers can create Question Sets and Rooms and students can join the room and play immediately.
3. Limitations and Difficulties Encountered

Several difficulties and challenges occurred during the initial stage of the project. This section is about the problems with which we have encountered and the methods we used to solve the problem.

4.1. Memory Leak Problem

During the construction of the Game Specific Framework as mentioned in the progress section, we have faced a memory leak problem related to Socket.io. Memory was 100 MB when the application started running and increased to over 2 GB after 3 days.

We found this problem during the game testing. The game was tested with fake players and ran for 3 days. After several hours of debugging, we managed to fix the problem. In fact, the problem was caused by a mistake while implementing Socket.io code.

We have randomized the “socket number” which we shouldn’t be doing. Therefore, by removing the function responsible for randomization of the “socket number”, the error is fixed. Memory would be always below 200 MB no matter how long the application was run, a 10 times improvement is obtained.

4.2. High Network Problem

The average network of the game was over 2 Mbps, which is very high and will affect game performance negatively.

After several algorithmic changes and fixes, the problem was alleviated. Instead of sending data of all objects, only objects near players are sent. In addition, the number of times of “socket.emit()” was reduced as it is very heavy.

As a result, the average network is less than 0.3 Mbps at all times. This is a 7 times improvement.
4.3. High CPU Usage Problem

The average CPU usage was 100% for the server which has exceeded the limit, resulting in the game failing to run properly.

To solve this problem, we have decreased the frame per second (FPS) of the server from 30 to 12. Moreover, unnecessary collision detections were removed.

As a result, the average CPU usage was reduced to below 50%. This is a 3 times improvement.

4.4. Poor Graphics Problem

As mentioned in 4.3, the FPS was lowered. However, this has caused a problem of poor graphics. In addition, movements looked unsmooth.

The interpolation technique was used to solve this problem. Interpolation is the technique of generating intermediate data between two time frames. For instance, when we have data (i.e. x = 1) of time t = 1 and data (i.e. x = 2) of time t = 5, interpolation technique could be used to generate artificial data for time between t = 2 to t = 4, which is x = 1.25, x = 1.5 and x = 1.75.

With the data of 12 FPS, intermediate data were constructed which made the game look like with a higher FPS. As a result, graphics looked better and movements looked smooth as if FPS became 60. This is a 5 times improvement.

4.5 Data Structure Design Challenge

The data structure is quite complex, with a lot of variables and objects such as Users, Question Sets, Questions, Rooms and Objects. In addition, the data has to be compatible for all three systems, Client, Server and Database.

Therefore, the design of data structure has delayed the progress of the project. Fortunately, we have allocated additional time for unexpected incidents. Thus, we finally managed to design a feasible solution for the problem without affecting the overall progress of the project. We have undergone numerous trial and errors, debugging and optimization on this issue.
4. Conclusion

Our project is going to be a gamified educational tool for children.

We have discussed about the background of the market of educational games and tools, some examples and that the market is huge for our project.

Our objective is to build an online gamified tool which is real-time multiplayer, cross-browser compatible, children-friendly and educational. The tool will be used by teachers as a fun way for dictations and quizzes.

We have presented the methodology of our project, which is consisted of three major components (i.e. client side, server side and production side). Client side includes user interface, graphics and sounds of the game. Server side, on the other hand, is responsible for calculations and algorithms of the game. Lastly, production side is to keep the project available to the public.

There will be three game modes for users which will be completed in the upcoming stages, including: Fill in the Blanks, Multiple Choice Questions and True or False Questions.

The report also discussed the progress of our team. We have completed several milestones, such as background research, game conceptualization, implementation (i.e. client, server and production side), as well as some debugging and optimization.

All of the problems encountered are solved thanks to the hard work of our team. Currently, the project is slightly behind schedule. We are looking forward to the next stage of the project. And hopefully, everything will be on track.
Reference


