Tamkang University
Department of Computer Science and Information Engineering
2021 Project report

Project title：Scoring assistance APP for reality game

Advisor ： 張峯誠

Students： 408850112 劉姿妤

408850443 林侑霆

408850435 盧佳馨

408850393 葉柏甫

408850260 楊思妤

Abstract

Living in Yilan for three years, we found that tourism is a large part of Yilan's economic source. In order to make tourists have a better understanding of Yilan's humanities and culture, we decided to cooperate with the Department of International Tourism Management to participate in the University's Social Responsibility Program (USR), Yilan-related materials are provided by the Department of International Tourism Management, while we provide technology and game design. Since everyone goes out with a mobile phone now, our main purpose is to produce a mobile application for solving real-world puzzles through our project. This application combines GPS and QR code to make users feel in-depth. What we mainly want to do is to input data and output it as a JSON file, and then convert it into other output formats through our “readdata” program.

In this project, we use JavaScript and the Editor.js library to provide an easy-to-setup page/screen editor, through which the information blocks are exported as a JSON file. We also developed a “readdata” program to convert the JSON file to an HTML page. One contribution of the design is to make everything in a page self-contained. For example, an image is embedded as a base64 DataURL. This approach eliminates the requirement of a separate file server, which may reduce the complexity to the users.

Due to time constraints, we demonstrated the authoring tool and the conversion tool with a simple Q&A game. The game is web-page based and can be deployed on a web server or a local directory. It can be played in a HTML browser.

Table of contents

List of figures

Chapter 1 -Introduction

In recent years, although there are a lot of tourists coming to Yilan, people are always following some online attractions, and less and less people are appreciating the history and antiquities. Therefore, we hope that tourists can use Yilan train station as the starting point to appreciate the culture and customs of the whole Yilan through our live game. Due to the limited time, we decided to make a simple version first. Therefore, we will make a web version of the game with our own campus as the theme in this project.

The University Social Responsibility Practice (USR) program promoted by the Ministry of Education in recent years is very much in line with our goals. It encourages university students to participate in society, observe social problems and find ways to solve them with their own abilities and skills. Thus, we would like to participate in this program and apply our abilities in local practice to contribute to society.

In the process of developing this game, we referred to many reality games, but found that many of them are still presented in a non-realistic way, such as questions that restrict the player to find hints or answers in a specific area, but in fact the player can answer the questions anywhere, which means that the meaning of realism is lost. Also, many reality games do not have final scoring, so we want to present it in the game we want to develop this time.

In this project report (web version), we hope that the game can be played not only by local students but also by students from abroad who come to our campus. We want to set up several routes for them to choose from, so they can assess their own ability to play, and provide Chinese and English, and design characters for them to choose from. We will provide a QR code for them to walk to the designated place in order to play the game successfully, and at the end of the game, we will count the score of their answers to tell their performance in this game.

The report is organized as follows. In Chapter 2, we introduce the information related to this project, such as what is a real game, the JSON technology we used in the project, similar games and literature references. In Chapter 3, we explain our motivation for doing this project and what we want to accomplish (including detailed software requirements and our first draft). In Chapter 4, we explain what special techniques we used in using JSON. For example, in addition to the built-in text storage function of JSON, we have designed a function that can store graphics and buttons [7]. In Chapter5, we explain the development process in detail, and in Chapter6, we demonstrate the finalized web version and conclude the topic with a conclusion.

Chapter 2 -Background

Related technologies and similar games, etc. For this report, we want to present it mainly in a quiz game and hope that it will eventually be able to detect GPS and participate in the time path. Therefore, the following methods were queried. You can see the relative technology we want to use in Ch 2.1, and similar game or application in Ch 2.2.

A reality game is a series of scenes in which multiple participants can play a certain role according to the plot setting of the game designer, collect information by observing and searching objects in the scenes under a certain time limit, analyze and reason logically, and solve puzzles to complete the game. [1]

2.1 Relative technology

In this project, we use the technology of HTML, CSS, JavaScript and JSON. HTML is a typical web language and it's simple and easy to read, and is also the structure of a website. CSS is to manage the appearance of web pages. And JavaScript needs to control the content of web pages, so it should be able to manage and control every element in the website.

JSON is a plain text-based way to store and send simple structured data. You can store any data (strings, numbers, arrays, objects) in a specific format, or send more complex data via objects or arrays. Its advantages include high compatibility, easy to read and understand format, and ease of data modification.[2]

2.2 Similar game or application

Just for the question-and-answer part (not including actual path). In Figure 2.1 King of knowledge(知識王) is an question and answer game, and it also count the core, at the end, someone who's score is higher is the winner. You can also play with the server and play against each other in pairs.

For the puzzle adventure games, also divided into two types, the first is the online game, the second is to play the game in reality. In Figure 2.2 Escape room-搶救泱泱村大作戰, is one of the online puzzle adventure game, it combining the background of the epidemic and the relics of the National Palace Museum(故宮博物院), players must collect enough materials from different National Palace Museum relics to make an antidote and find the key to open the gate in order to escape and rescue the infected villagers. About the second type, most of them are played in a certain room, the player should escape the room within the time limit, the more familiar gaming term is escape room. In Figure 2.3 is the reality puzzle game, escape room-救救唐三藏 held in the National Center for Traditional Arts. We actually experienced the game for the purpose of the survey and Figure 2.4 is the ticket for the game.

Figure 2.1 King of knowledge

Figure 2.2 Escape online game

Figure 2.3 Escape reality game

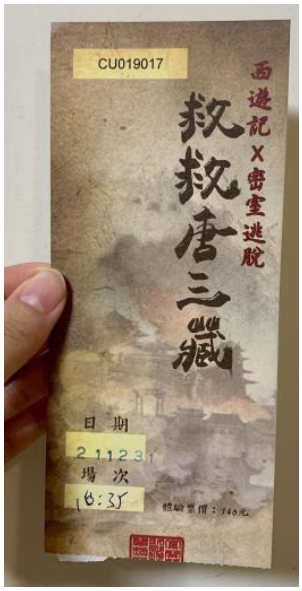


Figure 2.4 Ticket of escape reality game

Chapter 3 -Analysis

In this chapter, we would discuss the overall project overview for this project. In Ch3.1, we explain the motivation for our project. In Ch3.2, we explain the goals of the project, including the requirements for the software and the expected finished version, from initial planning, hand-drawn draft to final product. And in Ch3.3 we outline the technical problems we encountered in actual development and will briefly mention our solutions.

3.1 Motivation

About the motivation of this project, we wanted to accumulate practical experience and hope to apply it in practice, so we decided to participate in the USR project, hoping to cooperate with the Yilan County government and implement it in Yilan Railway Station. Therefore, we wanted to develop a simple quiz game that would guide tourists to the sightseeing spots and give them a better understanding of the local culture. After discussing it with the advisor professor, he told us that the Department of International Tourism Management had a similar idea and recommended that we discuss it with the professor of that department. After discussing with the professors of the Department of International Tourism Management, we decided that they would provide us with photos and information about the topic, and we would provide the technology and game design. However, due to the time constraint, we planned to make a simple webpage as the topic for this project first.

During the development process we noticed that each similar application is mostly redesigned from scratch by engineers. However, starting over requires a lot of time for design and testing, so we decided to create an editor that can convert the required data into JSON form for storage, and then use the program we developed to convert the JSON into other standard formats that we designed. For example, the web version we created is converted into the standard format of html that we designed using the above model, so we don't need to spend a lot of time to redesign a new file, just input the data to finish your web design and can be flexibly converted to the required device.

3.2 Project goal

We hope to make a reality game that can scan QR code with a cell phone to detect the location, pop up the question and answer, and finally score. However, in this project, we only develop the web version now. In fact, we have not given up developing the mobile version, our goal is still to use our project to develop a mobile version of the application, but because of the time pressure, we decided to develop a web version first as a foundation for the mobile board. Through the web version, we can do some process design, such as the design of the topic and how to present the topic and content, so the web version first cannot only show our results and solve most of the problems, but also help the development of the mobile version.

For the goal of the web version, we first want to provide a basic question and answer part

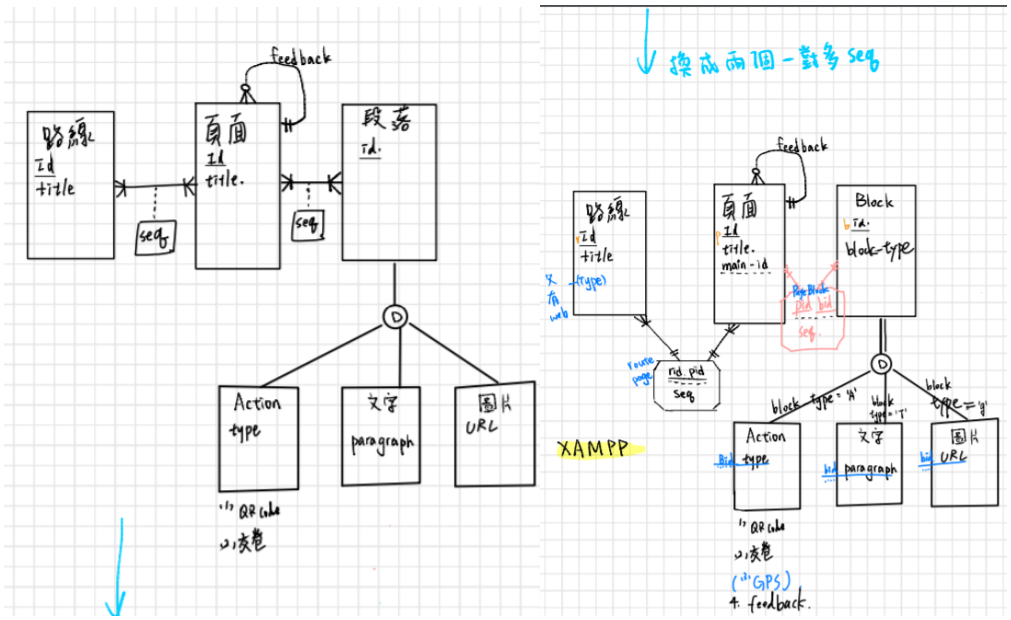
so that anyone can play it on their mobile devices. Next, we want to design many different routes for the player first, and make sure the question-and-answer part can be operated directly. Finally, if possible, we want our game to have a scoring system and it can show up at the end so that players can know about their performance.

3.2.1 Software requirements

The web version can do most of the functions first, such as the game page presentation, including the title, text, image arrangement, and ensuring the game flow without errors. The web version can be used as a complementary tool to the mobile version to do what we want to do in the mobile version, although there are some functions that cannot be done in the web version, such as scanning QR Code and integrating GPS.

Our idea is to make the reality game more in-depth, because if there is no positioning system, there will be a situation that the user can continue to solve the puzzle after leaving the place of real-life puzzle, so we use GPS to confirm the user's location on the mobile version, and the user needs to be in the correct location to join the game. In addition, GPS can be combined with the map, if someone is lost or cannot find the direction, then you can use the positioning system to guide him. The QR code scan is used to reconfirm the user's correct location to avoid the inaccuracy of the positioning system that could lead to the game being played in the wrong location and lack of realism.

3.2.2 Estimated completion



(a) (b)

Figure 3.1 website structure design

At first, we designed a database. According to Figure 3.1(a), we designed an ERD model. There are two many-to-many relationships between three main entities, they're route, page, and paragraph. Each of them all have attribute id as their primary key. In both route and page entities, they have title as their attribute. For the page entity, it has a foreign key called main-id, and it has at least one response to zero or more feedback to itself.

For the Fig 3.1(b), we change the entity name from paragraph to block since we think it's more suitable, it has disjointed relationship with action (e.g., QR code, submit button, GPS, and feedback function), words (paragraph like quotation or description) and pictures (URL). There are two sequence as weak entities between three main entities (route, page, and block).

The sequence between route and page has an one-to-one and one-to-many relationships (because a sequence will respond to one or many results to the route and page, and the route or page must apply one action to the sequence), and it has route id(rid) and page id(pid) attributes. The other sequence is the same as the previous one but the difference is that it is between the page and block entity. (And this sequence has the attribute pid and block id(bid).

However, after evaluating the customer's needs, we found that the amount of data was not so much that we needed to use a database to solve the problem. So finally, we used editor.js to transfer data, stored data with JSON and linked the game with a page string. The editor.js of JavaScript Object Notation (JSON) is used to edit the game, and it is also considered to provide practicality and convenience for editing the program partition blocks, and for subsequent users.

3.2.2.1 Mobile version

We draw on iPad with manuscripts:

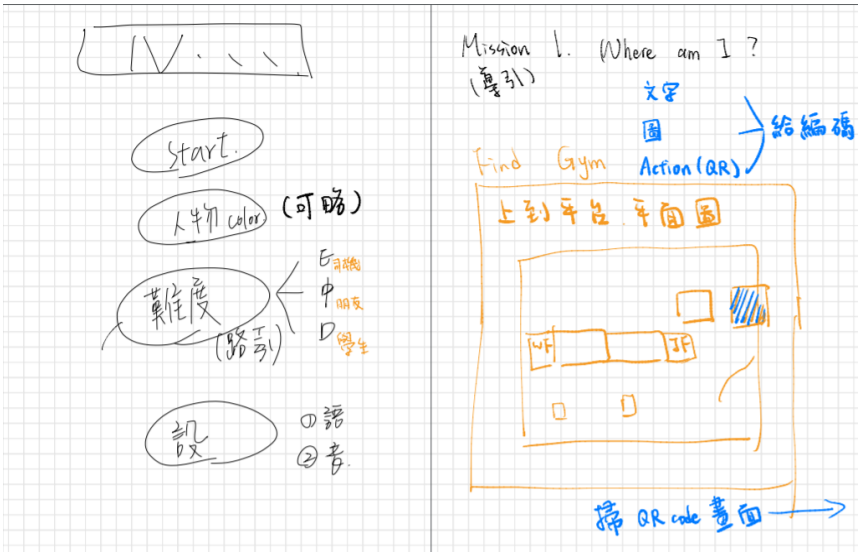


Figure 3.2 Website home page and guide page

In Figure 3.2, it is our draft for the starting page and the guide page. We designed four buttons at first (there are start, character color, difficulties and setting buttons). For the guide page, we don't know how to let the player know where they are at first. So we decide to show up a map and it may include different floors so the players can clearly know where they are. Also, we think starting with a significant landmark (the gymnasium) will help the player quickly know where they are.

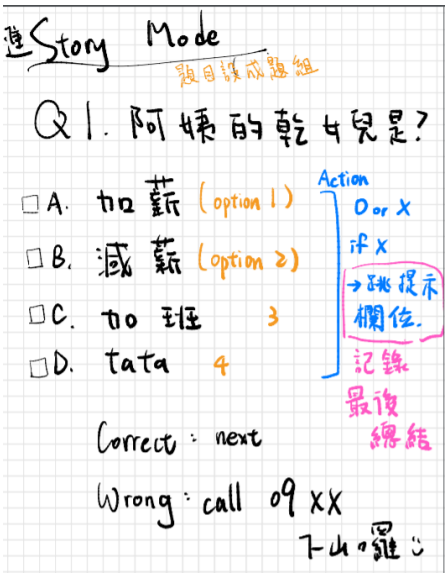


Figure 3.3 Question interface

In Figure 3.3, it is our question-and-answer part. It lists one question and four answers (multiple choice question). And in this figure, we want to show them about the hint and count the score. So, we are supposed to design that the total score will be deducted from 100 points, and if the player makes a mistake in each question, a hint will pop up until you get it right. (Set the questions into question groups and score them, and settle them at the end.)

3.2.2.2 Web version

In order to test that the game we designed works properly, we use slides to simulate:

Figure 3.4 Home page

At beginning, we design the home page and name the title 蘭陽小學堂, and add two buttons for start and setting.

Figure 3.5 Difficulty

After clicking the start from the home page, you'll see Fig 3.5. We design three difficulties for players and we have a small button right down the web page, it is a back to home page button.

Figure 3.6 Setting

After clicking the setting from the home page, you can change your language and volume. For the language, we want to do two different languages at first. And we also want to let players choose their character, but after we evaluate our ability, we decide not to do it at first. (Because we're not good at painting)

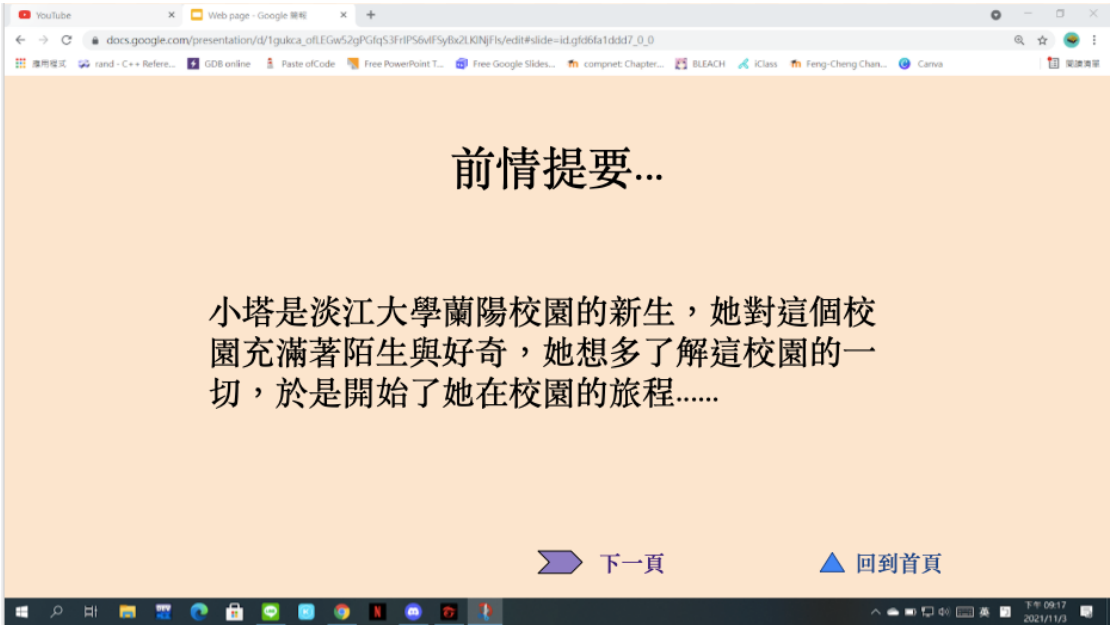


Figure 3.7 Storyline of game

After clicking the start on the homepage, it would enter the game and show the story background, and content of the game will also be in accordance with the storyline to conduct.

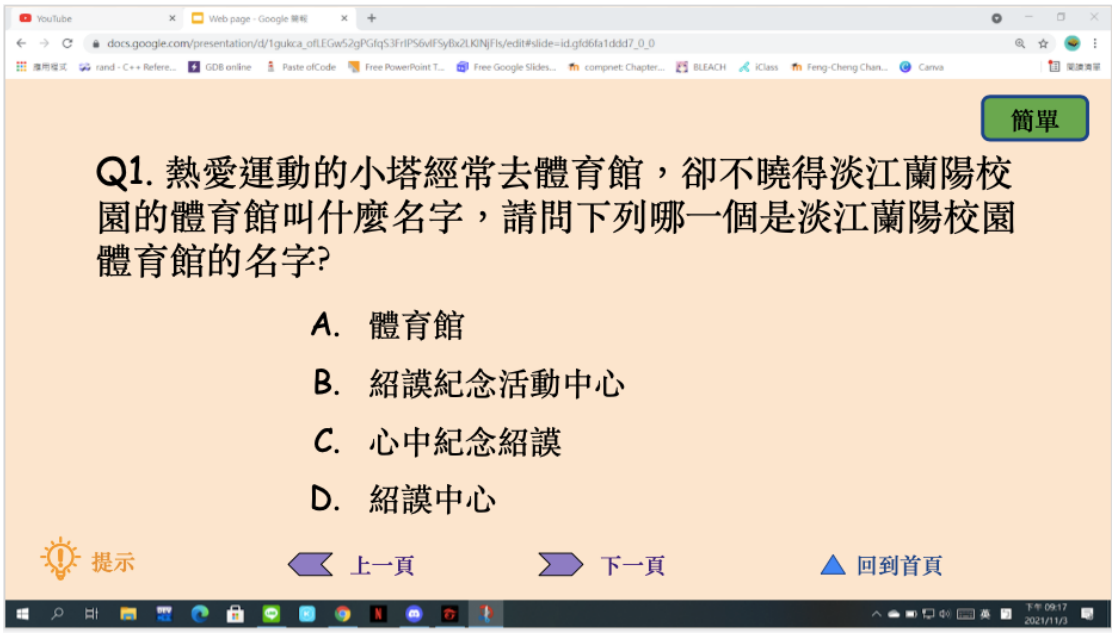


Figure 3.8 Questions of game example

Clicking the next page on the storyline page would be the questions. In the question page, we designed the next question to pop up directly after receiving the correct answer. Usually, the questions are related to the scenario within the boundaries that we limited.

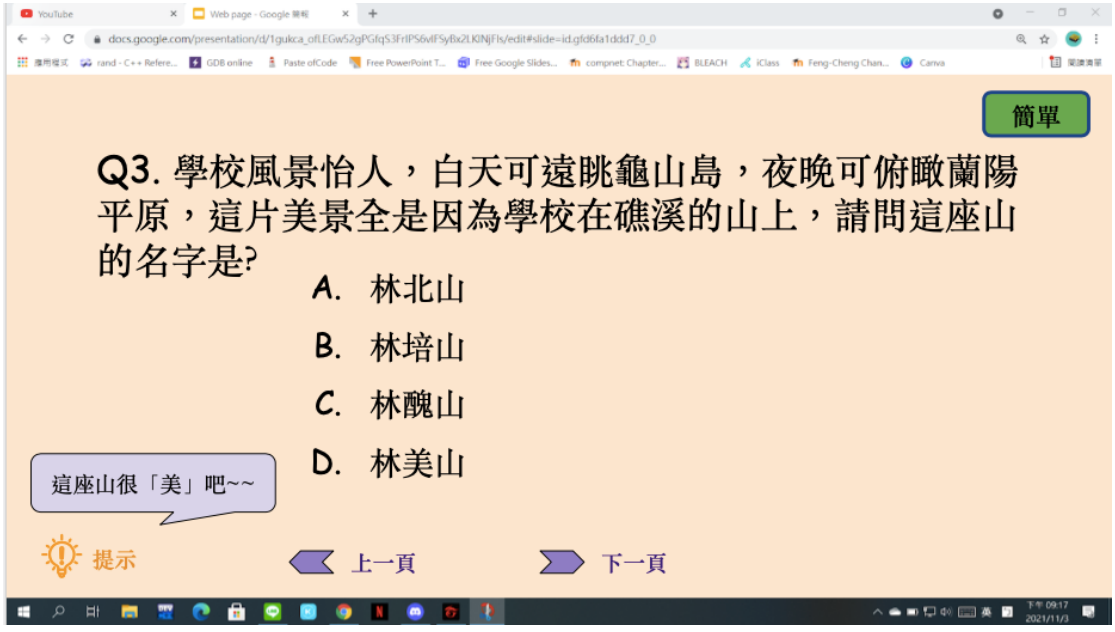


Figure 3.9 Hint of question

We have designed a hint for each question to ensure that the player can answer the question correctly and would not get stuck.

Figure 3.10 Finish button

A button for players who want to submit their answer.

Figure 3.11 The final score

After players answer all questions, we will count the score and players will see the result.

3.3 Technical solution

In order to save time in application development, we made a compiler that can automatically produce and output a standard format so that the user can convert according to the desired format, so we designed a set of conversion formulas to implement the functions of the compiler.

After doing the text version, we wanted to add the upload photo function but found that

the whole layout would run off and the photos could not be displayed properly, when we checked it was a problem with brackets ({}) and found that the upload photo function needed to set up a server to achieve it when testing.

The original buttons, text and pictures are saved as a JSON file, and when trying to convert the file, we found that the button function we wrote could not be converted to an html file. There is an official JSON to html conversion method, but it does not support the button function, so we made our own Read Data version and added a button function.

While designing the CSS, we found that the editor.js we designed did not support the no-input feature, so the CSS design did not look the way we expected. Because of the lack of input, we adopted the blank key method.

Chapter 4 -Design(solution)

In this section we will introduce the JSON and Editor.js technology used in our project. There is also a brief mention of the specified method we use to convert images in files.

4.1 JSON

JSON stands for JavaScript Object Notation. It is a standard format for presenting structured data as JavaScript objects, and although JSON is based on JavaScript syntax, it can be used independently, and many programming environments can read (parse) and generate JSON. JSON objects can be stored in their own files, which are basically text files with the filename. json, and the MIME type of application/json.[4]

4.2 Editor.js

Workspace in classic editors is made of a single content editable element, used to create different HTML markups. Editor.js workspace consists of separate Blocks: paragraphs, headings, images, lists, quotes, etc. Each of them is an independent content editable element (or more complex structure) provided by Plugin and united by Editor's Core. And Editor.js outputs JSON object with data of each block sequentially. [5]

4.3 DataURL

When a picture is converted to a JSON file in the Editor.js, a new server needs to be opened to support the conversion. Not only do we need to provide a new server, but the user also needs to learn to open our server. This will complicate the game and defeat the purpose of the game for the general public. However, displaying images is an important part of the game we are designing. We cannot use the .jpg format directly, but we can convert the image format to base64 DataURL and store it in JSON, although it will result in a very large file, the advantage is that the file is all the data and there is no need to download extra images.

A DataURL is a URI scheme that provides a way to inline data in an HTML document. It might save some time for small files, but for bigger files there are downsides in the increased HTML file size, and they are especially a problem if the image reloads on all your pages, as you can’t take advantage of the browser caching capabilities. We want use this technology so that our picture can be saved in HTML file, and when we call it, it can show up without some complicated process, such as catching pictures from server and return the result to the HTML file.

4.4 Read Data

Originally, we planned to use Editor.js to download JSON files and make them into a database, and then read the JSON files to present the content of the different pages. However, after planning the game topic, we noticed that the amount of data in the game was not so large

that we needed to use a database to support it, so we chose a simpler way to link the data together by directly converting each page of JSON files into HTML files through Read data, and then linking each page of HTML files with a button function. The result is to simply use buttons as a page conversion tool simply.

Chapter 5 -Implementation

After introducing the technology, we used in this project, we're going to introduce more detail about the specific technology in Editor.js and JSON. In Ch5.1, we show our Editor.js code used in our project report. In Ch5.2, we talk more about some special skills we use for this project in JSON, and explain why we use this technology and how we use it.

5.1 Editor.js

```
<script>
        function imgFileHandler(file) {
                return new Promise((res, rej) => {
                        let reader = new FileReader();
                        reader.onload = () => { return res(reader.result); }
                        reader.onerror = rej;
                        reader.readAsDataURL(file);
                });
        }

        var editor = null;

        function startup() {
                editor = new EditorJS({
                        holder: 'editorjs',
                        tools: {
                                image: {
                                        class: ImageTool,
                                        config: {
                                                uploader: {
                                                        uploadByFile(file) {
                                                                return imgFileHandler(file).then((data) => {
                                                                        return { success: 1, file: {url: data}}
                                                                })
                                                        }
                                                }
                                        }
                                },
                                AnyButton: {
                                        class: AnyButton,
                                        inlineToolbar: false,
                                        config:{
                                                css:{
                                                        "btnColor": "btn--gray",
                                                }
                                        }
                                },
```

Figure5.1(a) Editor.js

```
                                header:{
                                        class: Header,
                                        config:{
                                                placeholder: 'Enter a header',
                                                levels: [2, 3, 4],
                                                defautlevel: 3
                                        }
                                }
                        }
                });
        }

        function save() {
                editor.save().then((outputData) => {
                        let dl = document.createElement('a');
                        console.log(JSON.stringify(outputData));
                        dl.setAttribute('href', 'data:text/plain;base64,'+btoa(unescape(encodeURIComponent(JSON.stringify(outputData, null, '\t')))));
                        dl.setAttribute('download', 'output.json');
                        dl.style.display = 'none';
                        document.body.appendChild(dl);
                        dl.click();
                        document.body.removeChild(dl);
                });
        }
</script>
```

Figure5.1(b) The code of Editor.js

Figure5.1(a) and Figure5.1(b) are our Editor.js code. Line 22 of the code in Figure 5.1(a) to line 57 of the code in Figure 5.1(b) is used to determine which type the input value belongs to and in what format it is stored in the JSON file. Lines 59 to 70 of the code in Figure 5.1(b) are used to convert data into JSON format for storage, export and download of this JSON file.

5.2 JSON Structure

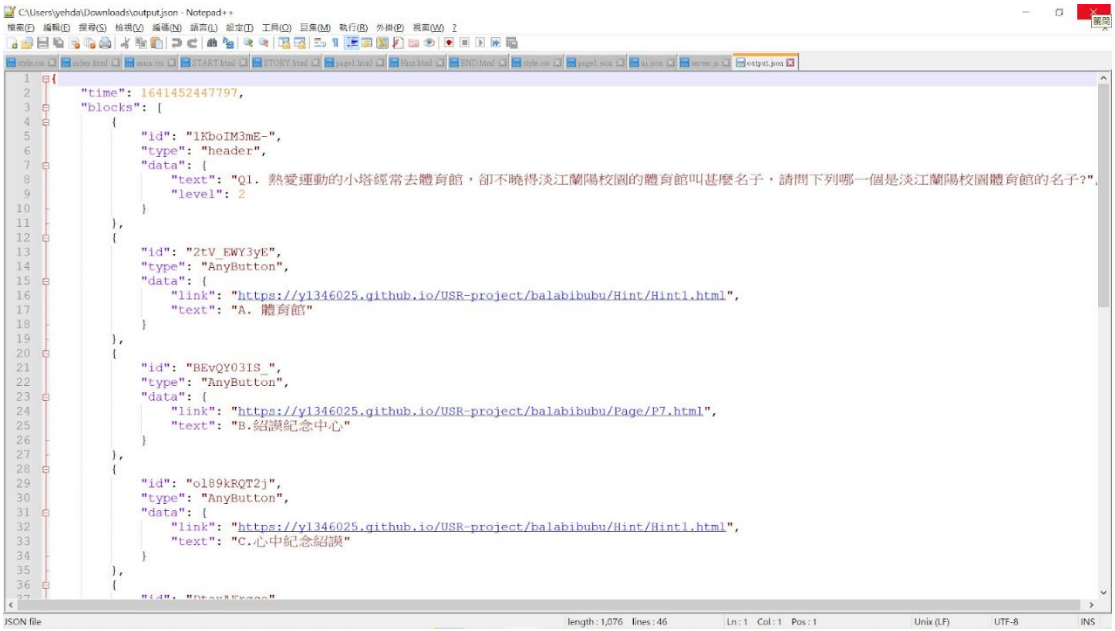


Figure 5.2 JSON structure

Figure 5.2 shows an example of a JSON structure. The "blocks" at the top of Figure 5.2 means that the data we enter is divided into different blocks. Each pair of brackets in the blocks contains the data for each different block, and the id is used to identify the different blocks. The type refers to the type of this block, e.g. header refers to the title. AnyButton refers to the button.

Our readdata function will determine what kind of block it is based on the type, and convert it to different functions depending on the type, such as header in the form of a header, and

AnyBotton in the form of a button. data refers to the data that will be displayed in this block. For example, the text in the data of the first block in Figure 5.2 is the text that will be displayed in the block, and the text in double quotes after the text is the text that will be displayed in the block. The level below the text is the text size, we set the text into three levels, level2, level3 and level4. level2 text size is h2 in html, level3 text size is h3 and level4 text size is h4. In Figure 5.2, the second block in data is written as a link. This link refers to the link that the user will go to when they press the button. The text below the link refers to the text that will be displayed on the button.

During the production process, we found that Editor.js has many official functions, such as the function of saving text, but at the same time, there are also functions that we need but not suitable for us, such as the image and button functions. These two functions in our project are our own design. Firstly, we would like to introduce the official Editor.js features used in our project. The first one is the text function, which allows you to save the input data in text format so that other files can convert it in text form when processing. The second is the title function, which allows text to be enlarged, bolded, or italicized, making it more visible and easier to read.

When developing our project, we found that our project could not rely on the official functions provided by Editor.js only, because some of its original functions could not achieve what we wanted to do or were limited by conditions, so we wrote some additional functions in our project.

The first one is the button function which we use a lot in this demo version. The button is one of the core functions of our demo real game, and we can't do without it because it is the most frequently used function. However, because the button function we use is not officially provided, and the official save() function can only download the officially released function as a JSON file, as long as the unofficial API will not be exported as a JSON file, so we need to write a set of functions that can save the format as a button. We use JSON.parse() to convert the output JSON file into a JSON object, and then determine the type of object to determine the output html format.

When we made the first version of the save() function, we found that there was no way to save Chinese, so we used the unescape() and encodeURIComponent() [6] functions to make Chinese correctly transcoded and stored in JSON files.

```
var obj = JSON.parse(data);
```

Figure 5.3 (a)Button function

```
else if(block.type == "AnyButton"){
    fs.writeSync(output,'<div class="AnyButton">\n');
    fs.writeSync(output,'<button type="button" onclick="window.location.href=\''+block.data.link+'\'">\n');
    fs.writeSync(output,block.data.text);
    fs.writeSync(output,'</button>\n');
```

Figure 5.3 (b)Button function

We have also added a picture function, initially we made a picture function by referring to the official website. However, after testing and checking, we found that the first version of the image function required a server to be set up in order to download it, so we tried various things from scratch. Finally, this version is a way to put images into JSON without setting up a server, we convert the images to base64 DataURL and then output them to a JSON file.

Because of these two additional functions, our project can be used more flexibly, not only with interactive buttons, but also with better understanding of the image function, as well as combining the two to create more diverse puzzles.

5.3 Read data

```
const {argv} =require('process');


const fs = require('fs');
    if(argv.length < 4){
        console.log('argv!!!');
        process.exit();
    };
const data = fs.readFileSync(argv[2],{encoding:'utf8',flag:'r'});
var output = fs.openSync(argv[3],'w');
var obj = JSON.parse(data);
fs.writeSync(output,'<!DOCTYPE html>\n');
fs.writeSync(output,'<html>\n');
fs.writeSync(output,'<head>\n');
fs.writeSync(output,'<meta charset="utf-8">\n');
fs.writeSync(output,'<link rel="stylesheet" type="text/css" href="style.css">\n');//這裡還要丟一個外部CSS連結

fs.writeSync(output,'</head>\n');
fs.writeSync(output,'<body>\n');
for(block of obj.blocks){

    if(block.type == "paragraph"){
        fs.writeSync(output,'<div class="Paragraph">\n');
        fs.writeSync(output,block.data.text);
        fs.writeSync(output,'\n');
        fs.writeSync(output,'</div>\n');
    }else if(block.type == "image"){
        fs.writeSync(output,'<div class="Image">\n');
        fs.writeSync(output,'<img src="');
        fs.writeSync(output,block.data.file.url);
        fs.writeSync(output,'">\n');
        fs.writeSync(output,'</div>\n');
    }else if(block.type == "AnyButton"){
        fs.writeSync(output,'<div class="AnyButton">\n');
        fs.writeSync(output,'<button type="button" onclick="window.location.href=\''+block.data.link+'\'">\n');
        fs.writeSync(output,block.data.text);
        fs.writeSync(output,'</button>\n');

        fs.writeSync(output,'</div>\n');
    }else if(block.type == "header"){
        fs.writeSync(output,'<div class="Header">\n');
        if(block.data.level==2){
            fs.writeSync(output,'<h2>\n');
            fs.writeSync(output,block.data.text);
            fs.writeSync(output,'\n');
            fs.writeSync(output,'</h2>\n');
        }else if(block.data.level==3){
            fs.writeSync(output,'<h3>\n');
            fs.writeSync(output,block.data.text);
            fs.writeSync(output,'\n');
            fs.writeSync(output,'</h3>\n');
        }else{
            fs.writeSync(output,'<h4>\n');
            fs.writeSync(output,block.data.text);
            fs.writeSync(output,'\n');
            fs.writeSync(output,'</h4>\n');
        }
        fs.writeSync(output,'</div>\n');
    }
}
fs.writeSync(output,'</body>\n');
fs.writeSync(output,'</html>\n');
fs.close(output);
```

Figure 5.4 The code of readdata

readdata.js uses the function fs.writeSync()[8] to create an html file, and then uses a 20-60 line for loop to identify the block.type in the JSON object, and then input the syntax of a specific type of html according to each type.

Chapter 6 -Demonstration

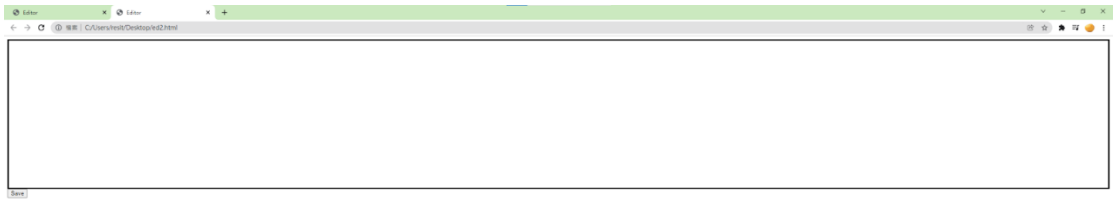


Figure6.1 (a) Editor.js

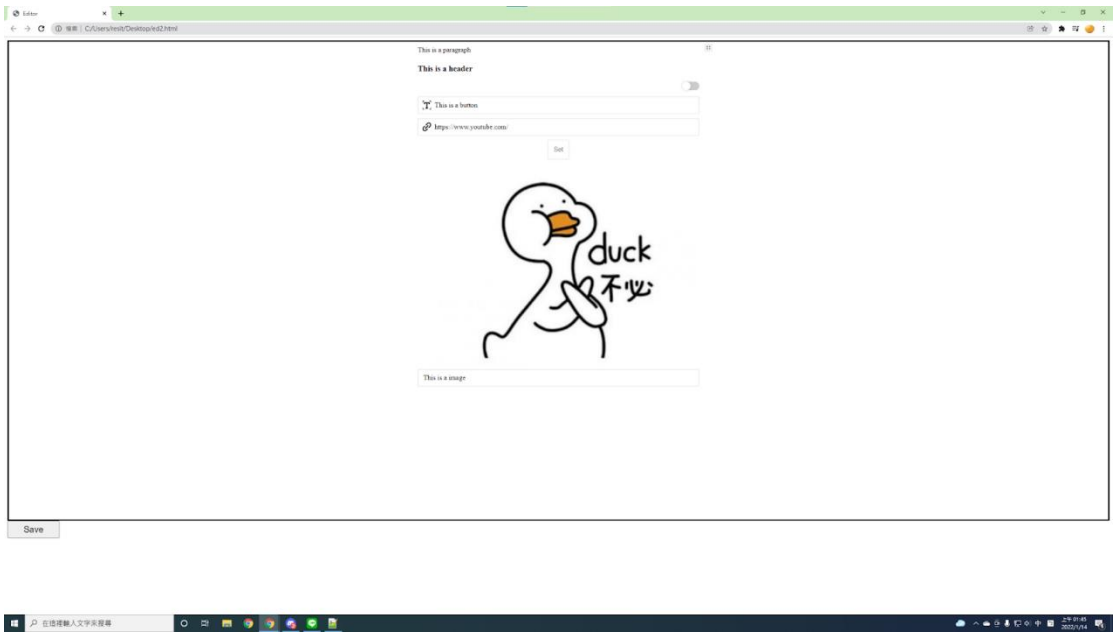


Figure 6.1 (b)Editor.js

When you open the HTML imported into Editor.js, the interface will appear as in Figure 6.1(a), edit the content you want in the black box, e.g. paragraph, header, image, button. After editing, press the save button in the bottom left corner to download the JSON file of the input data, as shown in Figure 6.1(b).

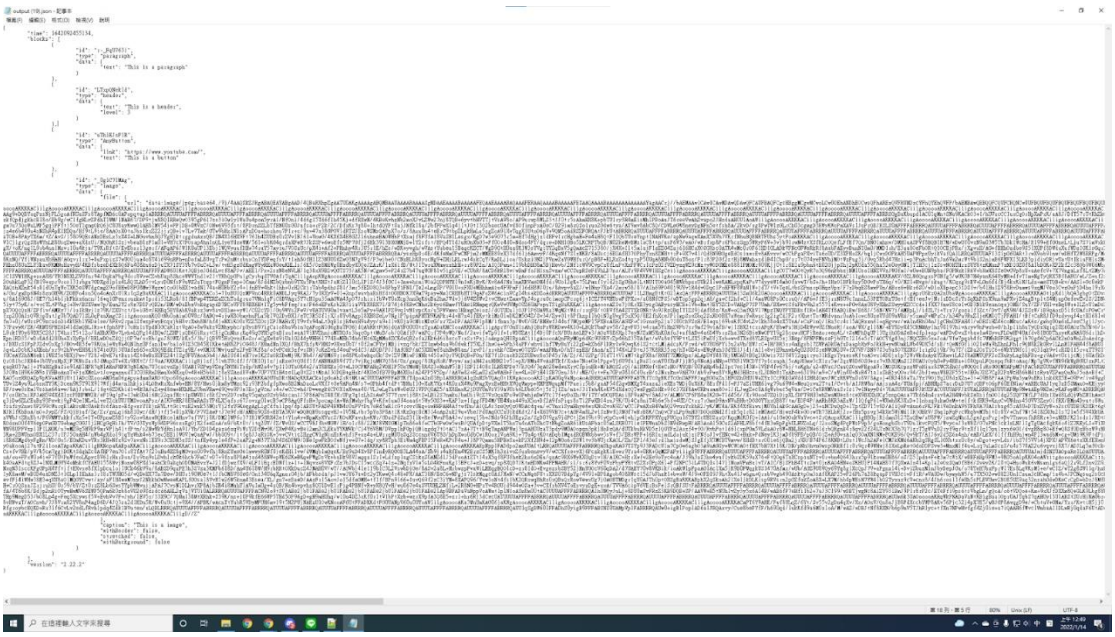


Figure 6.2 JSON

```
C:\Users\resit\Downloads>node readdata.js "output (19).json" Demo.html
```

Figure 6.3 readdata.js

Run readdata.js(input text would be like : node readdata.js "name of JSON file to convert" "name of the output html file")

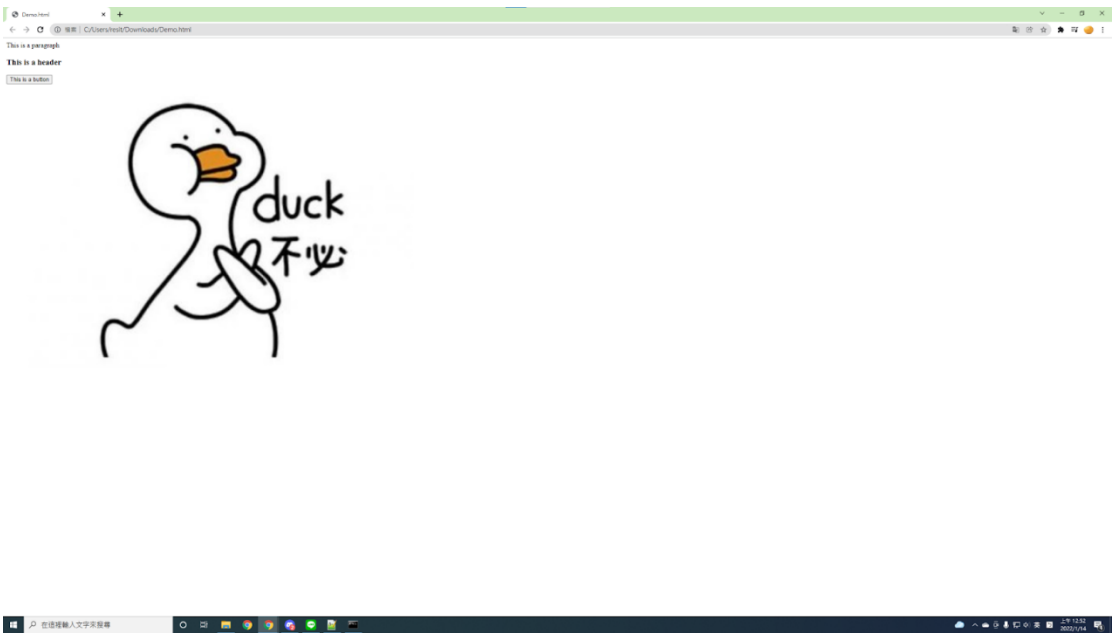


Figure 6.4 Output Demo.html

Chapter 7 -Conclusions

At first we wanted to develop a reality game, but due to time constraints, we aimed to develop a campus web version in this project, and we expect to apply it to mobile phones in the future and actually use it in local tourist spots.

In this project, we got familiar with many tools such as Editor.js and JSON. We used the features of Editor.js to allow us to partition the editor, simplify the editing page and provide more flexibility, and use many of the built-in features of Editor.js, such as editing text, title input, and creating new features of our own. We have also created new features, such as buttons for JSON, and the ability to add images to JSON files after converting them to base64 DataURL.

In this report, we believe that Chapter 3.2 Estimated completion and Chapter 6 Demonstration are worth cross-referencing. From these two chapters, we can see how we expected to look and how we finally made it. And in Chapter 5 Implementation, we can also learn how to start from scratch to final completion. Although we were not able to implement the scoring system in this project, we did implement the question-and-answer process part.

From the very beginning, when we thought of the title, what problems we encountered and how to solve them, we drew up the design, set the goals for the project and figured out how to make it happen. It was a process that started from scratch and really taught us a lot of lessons that we hadn't had before. We had to imagine what a game would do, how it would be played, what the theme would be and so on. We started with a hand-drawn script, imagining every page, step, even button and what it would do, and we worked on Editor.js and JSON syntax from the beginning to being able to build new features ourselves.

Having this project, we have slightly experienced that it is not easy for the engineers in the industry to meet the needs of the customers. Each step in the software development process is subject to many adjustments and discussions among the team members, however, the length of time spent is not indicative of good or bad results. Although there were many setbacks, the effort was worth it. We spent a lot of effort on this project, grew and improved our strengths during this experience, and gained a better understanding and knowledge of many tools. We believe that we will be able to complete the initial motivation of this project and realize the mobile version in the future.

Reference

[1] Reality game definition
http://ir.lib.ncku.edu.tw/handle/987654321/157084
[2] JSON 基本介紹
https://blog.wu-boy.com/2011/04/%E4%BD%A0%E4%B8%8D%E5%8F%AF%E4%B8%8D%E7%9F%A5%E7%9A%84-json-%E5%9F%BA%E6%9C%AC%E4%BB%8B%E7%B4%B9/
[3] 資料庫介紹與比較
https://ithelp.ithome.com.tw/articles/10206222
[4] JSON 基本介紹
https://developer.mozilla.org/zh-TW/docs/Learn/JavaScript/Objects/JSON
[5] Editor.js
https://editorjs.io/
[6] JavaScript atob / btoa 編解碼不支援 utf8 的解決方案
https://ithelp.ithome.com.tw/articles/10229587?sc=pt
[7] Editor.js-button
https://www.npmjs.com/package/editorjs-button
[8] node.js
https://nodejs.org/api/fs.html

Appendix

At first, we wanted to use databases (e.g., MySQL, PostgreSQL) for our project, but in our project, it was not suitable to use databases for storing data. The reason is that usually data is stored with databases because there is a large amount of data to be stored. In our project, there is a small amount of data. If we were to use MySQL it would face security issues and be unreadable, and if we were to use PostgreSQL it would be because it is more commercial, usually used when there is a large amount of data, and complex queries would lead to low performance. However, we need to store the data, so we discussed with our advisor and he introduced us a good way to store the data, i.e., JSON. Then we searched a lot of information about JSON on the internet, because JSON is an unknown area for us. After we learned some knowledge about JSON, we discussed it with our advisor again and decided to use JSON to store data. We wanted to use JSON to store data because it is a relatively simple way to store small amounts of data, easy to read, and flexible to use.[3]

The initial design was for a mobile application, however, we had studied web design and web programming techniques in our previous courses, so we wanted to start with the web version that we were familiar with. Afterwards, we will cooperate with the local cultural program of the Yilan government to develop a mobile version of the application. Therefore, about the language design, initially we wanted to have both Chinese and English versions, but considering the cooperation project with the Yilan County Government and our belief that Taiwanese are the main users, we decided to consider the English ability of some Taiwanese and the difficulties of senior citizens and children, so we finally decided to develop the Chinese version first.