Specification for a browser-based app “Kids multiple-choice game for use by English Teachers”

Project Proposal

Drafted: 4 February 2020 by James Marriott

Agreed on: not agreed

Table of Contents

[1 Introduction 2](#_Toc63326953)

[2 Overview 2](#_Toc63326954)

[2.1 Audience overview 2](#_Toc63326955)

[2.2 Deliverables 2](#_Toc63326956)

[3 Project Requirements 2](#_Toc63326957)

[3.1 Core Functionality – Gameplay Screen 2](#_Toc63326958)

[3.2 Options Page 2](#_Toc63326959)

[4 Design 3](#_Toc63326960)

[4.1 Visual approach 3](#_Toc63326961)

[4.2 Layouts 3](#_Toc63326962)

[4 Key Components 5](#_Toc63326963)

[4.1 Welcome / Options Screen 5](#_Toc63326964)

[4.2 GamePlay Screen 5](#_Toc63326965)

[5 Technical development 6](#_Toc63326966)

[5.1 Team 6](#_Toc63326967)

[5.2 Techincal approach 6](#_Toc63326968)

[6 Project Management 7](#_Toc63326969)

[6.1 Project Phases 7](#_Toc63326970)

[6.2 Next Steps 7](#_Toc63326971)

# 1 Introduction

According to the British Council, there are more than 12 million English Teachers worldwide with 1.5 billion students. Many of these teachers, teach online. As such, there is a significant secondary market for tools to help online teachers.

The purpose of this project is to develop a flexible multiple-choice game that online English teachers can use in the classroom.

The primary purpose of this project is as a learning exercise for trainee Front-End Developers working to developing their collaborative coding skills.

This project is not intended to be a commercial application.

# 2 Overview

## 2.1 Audience overview

The primary intended audience for this project is employers looking to hire Front End Developers. The app will be a portfolio project for Scrimba students who developed the app to be displayed on their GitHub pages and/ or personal websites.

The secondary audience of this app is for online English teachers. The author of this document (James Marriott) is a front end developer with experience as an online English Teacher. As such he has contacts in the teaching community who will be willing to help test the app and feedback on bugs/ browser issues/usability/ functionality. The app may later be used by teachers in real-world settings.

## 2.2 Deliverables

A browser-based app compatible with all major browsers (excl. I.E.). The app will allow English teachers to write multiple choice questions for two students and play a game in class to see who can get the most answers correct. As this is intended for online distance learning purposes, the teacher will load the app into her browser and share her screen with the students. As such, all interaction will be with the game will be made by the teacher (now referred to as ‘the user’.)

# 3 Project Requirements

## 3.1 Core Functionality – Gameplay Screen

The core functionality is the gameplay page. The app allows two students to play in turn for ten questions. Each student will be offered a question and three possible answers. If correct, the student’s score increments, and gameplay moves to the next player. At the end of the questions, a winner or draw is declared. The game can then be reset. Visually, this page should be very attractive for children utilising primary colours/animations/transitions/ cute characters.

## 3.2 Options Page

This page allows the user to set game player names and their token. This options page will allow the teacher to edit the default questions. By default, questions will be hardcoded into a JSON file. On the options page, the teacher should be able to set students names and change the questions. The question input interface should take in each question and three possible questions.

# 4 Design

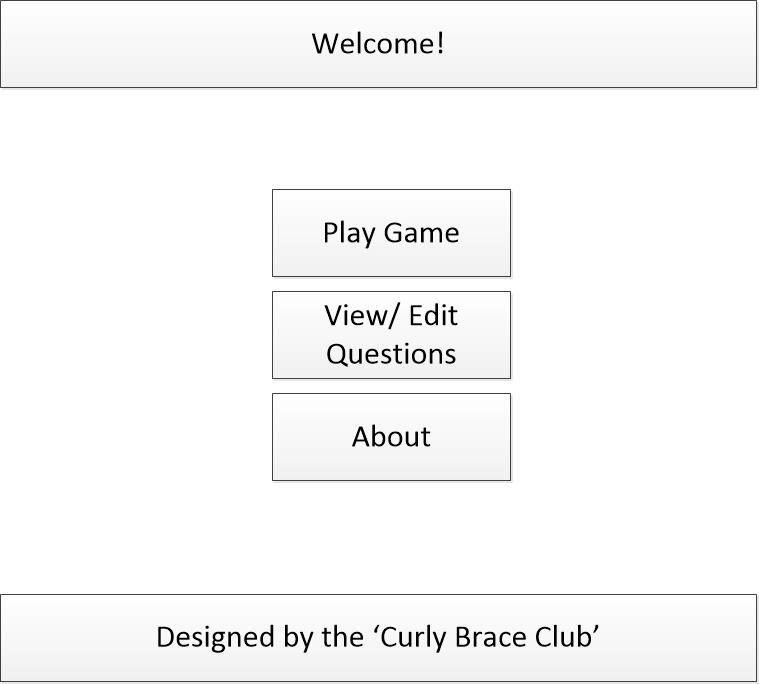
## 4.1 Visual approach

As the intended audience for this project is English Learners of school-age children (5 – 15) the gameplay interface should be appealing to children. This would include the use of bright primary colours, emojis, curved buttons etc, child-friendly fonts.

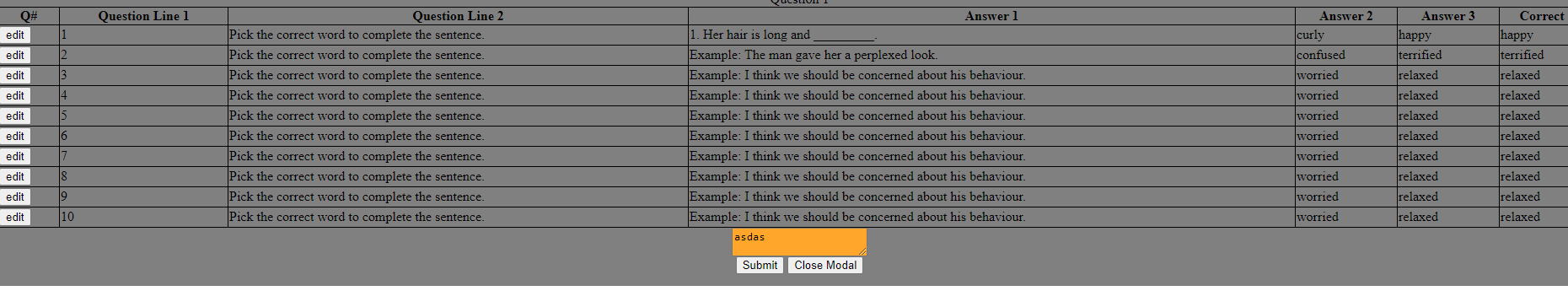
## 4.2 Layouts

There are two main pages for this app. A welcome / options screen and the gameplay screen. Here is a very rough idea of layouts.

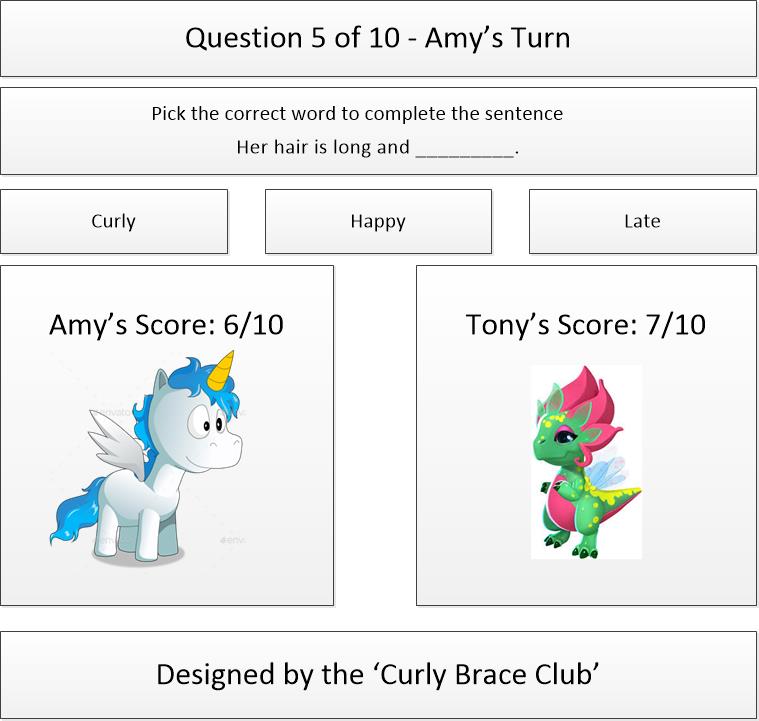
**Figure 1 –welcome screen**

****

**Figure 2 – Question Edit Modal**

****

**Figure 3 – Gameplay screen**



# 4 Key Components

## 4.1 Welcome / Options Screen

Welcome/ Options page.

1 – Set student names and tokens. This options page will allow the teacher to edit the default questions. By default, questions will be hardcoded into a JSON file. On the options page, the teacher should be able to set students names and change the questions. The question input interface should take in each question and three possible questions.

In a commercial application, there would be an option to log in and save question sets. However, as a development exercise and to avoid any need for back-end development these questions will be saved in local storage. The teacher can cut and paste pre-written questions. The questions will be saved in local storage. This will demonstrate skills in handling and validating strings.

These questions can be cut and paste from a word processor (or typed into a box.) The user will enter the questions into a text box in the following format:

‘%'Pick the correct adjective to complete the sentence.%'Her hair is long and \_\_\_\_\_\_\_\_\_.%curly%happy%late%1.’

A special character ‘%’ (or similar) would indicate a break. So, there will be two question line inputs. Three possible answers and the correct answer. These will be converted into five strings (2 question lines, 3 possible answers) and 1 integer (1, 2 or 3) indicating the correct answer.

This input will be saved into an object format such as.

let questions = {

1 : [{

'ID': 1,

'question-line-1': 'Pick the correct word to complete the sentence.',

'question-line-2': 'Her hair is long and \_\_\_\_\_\_\_\_\_.',

'answers': ['curly', 'happy', 'late'],

'answer': 1

}],

2 : [{

'ID': 2,

… etc.

## 4.2 GamePlay Screen

The gameplay screen should load up the questions from the JSON file in random order.

Each student takes a turn to answer a question. The question box changes to ‘Correct! <Answer> is the right’ or ‘Sorry! The correct answer is <answer>.

If correct the student gets a point. Play moves to the next player.

(Optional functionality – If the student gets the wrong answer, it could be possible to remove that answer and allow the student to choose between the last two options. Then the student would get half a point. Each question could be worth ten points. If correct on the second go, the student gets awarded 5 points.)

The play continues until all questions have been asked. A message congratulates the winner.

# 5 Technical development

## 5.1 Team

7 members of ‘the curly brace club’ coding team with the following discord handles working in pairs (and one three).

1. JamesRM: AdelFattakhova.
2. KaranT: Parthnarang: Janedoe.
3. Wigsters52: The-Helvetic-Scenario

Once approved. The project will be broken down into modular parts and tracked using a shared repository on GitHub. (Any team member who is unfamiliar with github can take the very short and straight forward FEND Scrimba - Module 10 Learn Git)

Here is the gituhb repo <https://github.com/jamesmarriott/English-Learning---Kids-Game>

### 5.2 Technical approach

From a technical point of view, this project will be more performant in React. However, as an exercise in fundamental skills (HTML/ CSS/ JS) it may be better to use vanilla JS. The project could then be tackled in two phases.

Phase 1 – Develop the code using vanilla JS.

Phase 2 – Refactor the code in React.

This demonstrates fundamental skills in

* Vanilla JS
* CSS
* HTML
* Refactoring Code
* React

However, the team may decide to use React from the outset. This is something that needs to be discussed and agreed.

# 6 Project Management

### 6.1 Project Phases

Phase 1 – Design layouts. Identify color scheme (using CSS hex codes), typography – font names (desired/ and fall back), sizing (H1, H2, P, etc.) – emojis / user icons.

Phase 2 – Create a static page using HTML and CSS. Create a JSON file with default content.

Phase 3 – Identify independent javascript modules and colloboratively build components using either in React or vanilla JS (as decided by the team.) These should adhere to the ‘separation of concerns’ principle so that modules such as rendering and game play logic are in separate files.

Phase 4 – Testing.

## 6.2 Next Steps

Hopefully, this document has explained the project and how it could be tackled.

The team should now:

1. Consider this document and make suggestions/ changes to the proposal.
2. Agree on the technical approach – specifically to use React or not.
3. Agree on the division of labour and assign responsibilities and tasks.
4. Edit and finalise this document.

-0-