Product Design Document

Mystery Trivia

**Authored by:**

Mohammed Chokr

Austin Jeffery

Jason Marrone

Mohammed Rubel

# Revision History

| **Date** | **Description** | **Authors** | **Comments** |
| --- | --- | --- | --- |
| 2/25/2023 | Version 1 | Mohammed Chokr  Austin Jeffery  Jason Marrone  Mohammed Rubel | First draft |
| 3/2/2023 | Version 2 | Mohammed Chokr  Austin Jeffery  Jason Marrone  Mohammed Rubel | Revised and edited as requested by the TA |
| 3/6/2023 | Version 3 | Mohammed Chokr  Austin Jeffery  Jason Marrone  Mohammed Rubel | Revised and edited as requested by the TA |

# Document Approval

The following document has been accepted and approved by the following:

| **Signature** | **Printed Name** | **Title** | **Date** |
| --- | --- | --- | --- |
|  | Mohammed Chokr | Back-end Lead | 3/2/2023 |
|  | Austin Jeffery | Team Lead | 3/2/2023 |
|  | Jason Marrone | Front-end Lead | 3/2/2023 |
|  | Mohammed Rubel | Documentation Lead | 3/2/2023 |

[**Revision History 2**](#_31i46yn8p2ns)

[**Document Approval 2**](#_f5zs5c3xc7xy)

[**1. Purpose 4**](#_5jp5tbpxo15v)

[**2. General Overview 4**](#_rufnb8jwjp3l)

[2.1 Assumption: 4](#_edbzynkd987e)

[2.2 Constraints: 4](#_3gu773iqknq6)

[2.3 Standards: 5](#_7ucaw34swxxp)

[**3. Architecture Design 5**](#_1ps83pfjg24w)

[3.1 Hardware Architecture 5](#_dz4s1rw8eox9)

[3.2 Software Architecture 5](#_wslygqi2m2f)

[3.3 Security Architecture 6](#_1wp0umz53pi5)

[3.4 Communication Architecture 7](#_o6mvr176w7bg)

[**4 System Design 8**](#_1bglyc3r3tm8)

[4.1 Use-cases 8](#_8k7ttx6mkbl9)

[4.2 Sequence Diagram 17](#_y3xe3ncr0q0u)

[Fig. 4.2.1 Logging into the application. 17](#_bb0yvthk0lrx)

[Fig. 4.2.2 Saving user progress. 17](#_nuqs1f1oey36)

[Fig. 4.2.3 Registering for an account 18](#_gedx79o6jxtz)

[4.3 Data Flow Diagram 18](#_up6cv3aavj2h)

[Fig. 4.3.1 Full data flow diagram. 18](#_zq9dy4zan1x)

[Fig. 4.3.2 First half data flow diagram. 19](#_d37zbuhwqhc2)

[Fig. 4.3.3 Second half data flow diagram. 20](#_h4gru3p02usj)

[4.4 Database Design 21](#_eivlq9dvxl0l)

[4.5 Class Diagram 21](#_lqnmy2ejpobp)

# Purpose

The Product Design Specification document will discuss and show a blueprint for how the application should be built and operated. This document will serve as a guide for developers which will outline the architecture of how the system will work together and communicate. This document will provide a clear understanding and help developers have a clear guide on the architecture and system design. The Product Design Specification document will help identify potential issues or challenges that may happen during the development process and by doing so, this can help the development team reduce the risk of delay. This document will be used as a reference for modifications in the future to the application. In addition this document will show and discuss different ways the application communicates and links with one another.

# 2. General Overview

## 2.1 Assumption:

One of our general assumptions is that our end-users will be young kids who could practice their beginning English, Mathematics, and Reading skills. Another general assumption is that our program will be simple-to-use. Finally, another general assumption is that the user experience should be great. These assumptions will be used to address any constraints within the questions of the game and the gameplay.

## 2.2 Constraints:

The constraint is that the questions must be simple worded and not too complex for the end-user. This means that the design of the game must be simple to use and easy for our users to navigate. In addition, our game must be fun and challenging for the user so that they can enjoy the game, so adding enemies for the player to avoid and high scores to enhance competition must be added.

## 2.3 Standards:

Our general standard is that our targeted audience will enjoy our game. This means that we hope our product will be easy to use by having multiple choice questions and answers where users can choose from which has four options and a hint button that’ll help reduce the possible choices provided instead of entering answers. This game is going to be challenging by having multiple levels with a variety of questions and a mysterious maze that you have to get through while collecting chests and avoiding enemies.

Abiding by these standards and being mindful of the assumptions and constraints previously outlined are an amazing starting point for using this document meaningfully.

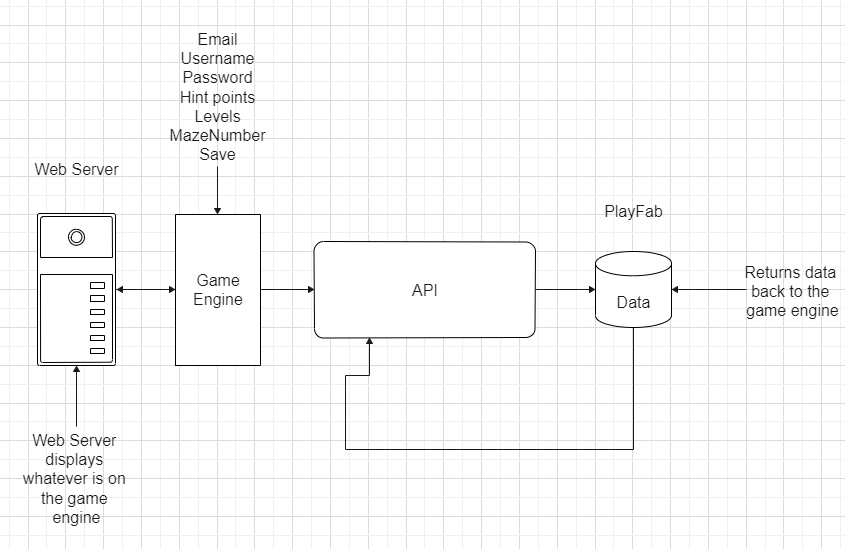
# 3. Architecture Design

## 3.1 Hardware Architecture

This is not applicable to this project.

## 3.2 Software Architecture

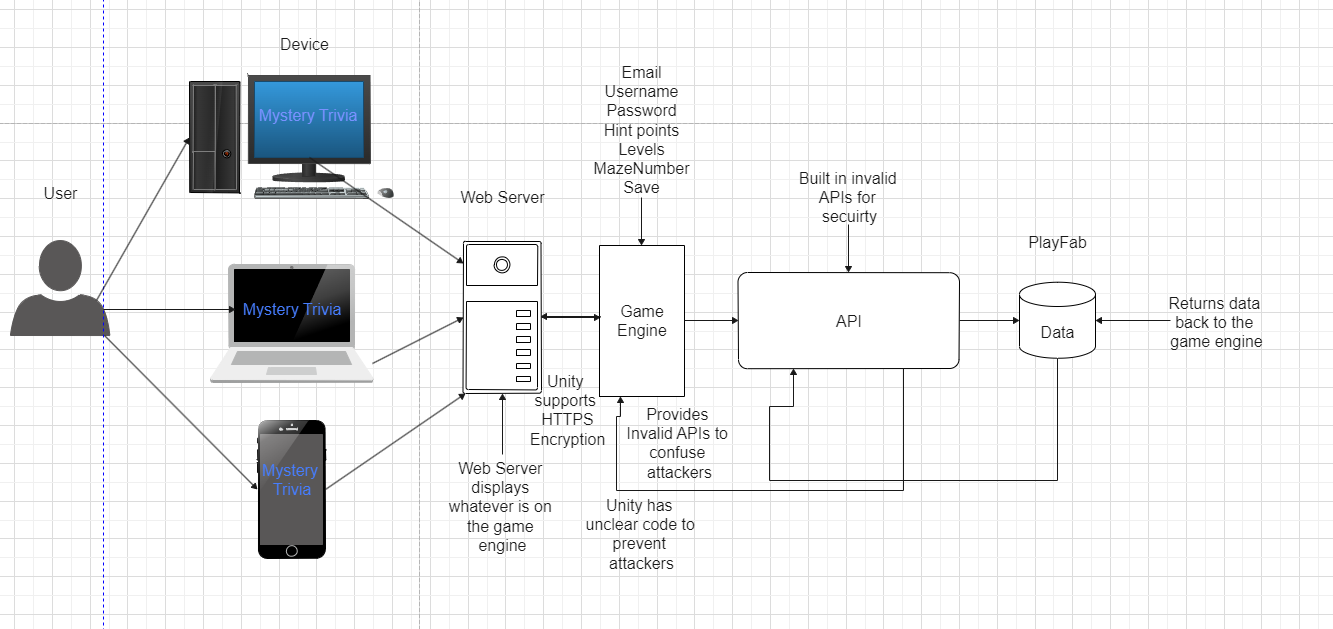
The software architecture of our system has been designed and structured into four distinct layers, each with its unique purpose and functionality. This is composed of the Web server which acts as the primary entry point for all users. The website is responsible for giving the user easy access to play the game. The game engine, which runs on unity, is the main component that provides built in APIs that enables the game to work with external APIs.The game engine will also enhance the users overall experience. The API will connect to our fourth layer which is the database we are using called PlayFab. PlayFab will constantly communicate with our game engine to ensure all data is safely stored and read. PlayFab is the backbone of our system that enables the system to constantly send and receive user data. This is what keeps our data securely stored and accessible at any time.



*Fig. 3.2.1*

## 3.3 Security Architecture

The security architecture for the game application is protected by the game engine unity. Unity automatically makes their code unclear to make it more difficult for hackers to reverse engineer and exploit. Unity provides encryption APIs that allow developers to encrypt sensitive data such as player credentials and game assets. This application will be used through a web browser and with that being said it does not need to secure the game application through the web because Unity supports HTTPS encryption for network traffic and will communicate with the game engine. Unity provides protection through their engine as they work off of invalid APIs that help prevent common types of attacks.



*Fig. 3.3.1*

## 3.4 Communication Architecture

The communication Architecture is based on the Unity maze game creating a connection to the PlayFab database. Once they are connected, the maze game will be able to send data such as a user's email, username, hint points, level, and maze number to be stored within the database. By using PlayFab API with Unity reference, it will allow data storage as well as user authentication for the maze game. Unity will use UDP protocol to display the game to the user through the webpage.

# 4 System Design

## 4.1 Use-cases

| **Use Case ID:** | UC-1 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Answering questions | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Austin Jeffery |
| **Date Created:** | 2-22-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | This use case describes what happens when the question pop up is initiated. | | |
| **Trigger:** | | The player character touches a lock. | | |
| **Preconditions:** | | 1. The user has chosen a category. 2. The user is in the maze scene. | | |
| **Postconditions:** | | The user has gained 50 points and the next time the user interacts with that lock, they will be able to destroy it and free that passage. | | |
| **Normal Flow:** | | 1. The player collides with a lock, which will stop their movement. 2. Question pop-up appears and provides a question and four available answers. 3. The user selects an answer with their mouse. 4. If the answer is correct, it will light up green. If it is incorrect, it will light up red and disappear causing the player to interact with the lock again. 5. The question pop-up animation disappears and the player will be able to move. | | |
| **Alternate flow:** | | N/A | | |
| **Frequency of Use:** | | Whenever a lock object is interacted with. | | |
| **Assumptions:** | | The user has sufficient resolution to view the full question popup. | | |

| **Use Case ID:** | UC-2 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Changing volume option | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Jason Marrone |
| **Date Created:** | 2-22-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | This use case describes what happens when the volume slider is clicked. | | |
| **Trigger:** | | The user clicks the volume slider. | | |
| **Preconditions:** | | The application is in maze view. | | |
| **Postconditions:** | | The application’s output volume will be adjusted to match the input. | | |
| **Normal Flow:** | | 1. The application adjusts the appearance of the volume slider. 2. The application adjusts the volume to match the new slider position. . | | |
| **Alternate Flow:** | | None. | | |
| **Frequency of Use:** | | Whenever the user desires a different volume. | | |
| **Assumptions:** | | The user has properly functioning audio hardware. | | |

| **Use Case ID:** | UC-3 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Player movement | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Jason Marrone |
| **Date Created:** | 2-22-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | This use case is for player character movement. | | |
| **Trigger:** | | 1. The ↑ arrow is pressed. 2. The ← arrow is pressed. 3. The ↓ arrow is pressed. 4. The → arrow is pressed. 5. The w key is pressed. 6. The a key is pressed. 7. The s key is pressed. 8. The d key is pressed. | | |
| **Preconditions:** | | The user is not answering a question. | | |
| **Postconditions:** | | The user is in a new position. | | |
| **Normal Flow:** | | 1. The player character moves at a constant speed in a direction determined by the key press for as long as the same input key is pressed. | | |
| **Alternative Flows:** | | 1. The player character collides with an enemy. 2. Move to UC-5. 3. The player character collides with a powerup. 4. The constant speed is altered to a larger constant for 5 seconds. 5. Return to normal flow. | | |
| **Frequency of Use:** | | Whenever the user wishes to move. | | |
| **Assumptions:** | | The user will not collide with a wall object. | | |

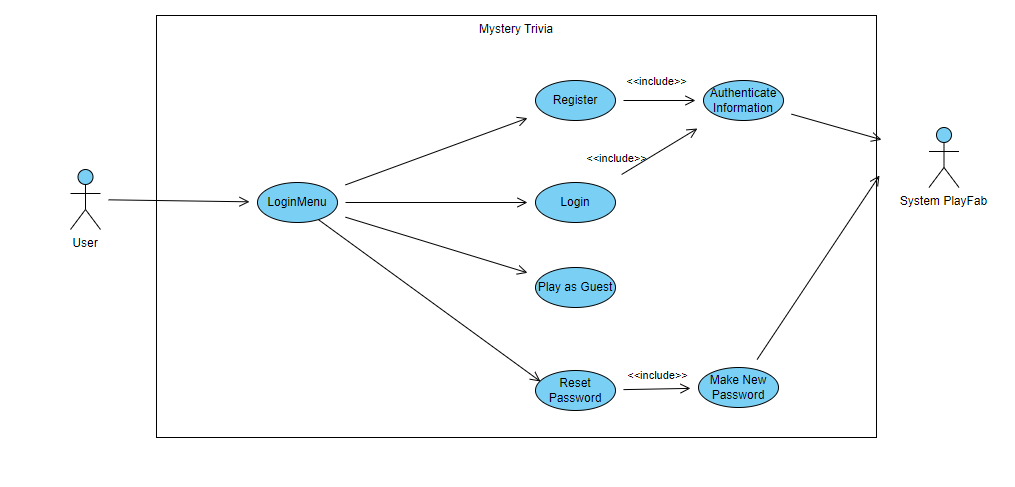
| **Use Case ID:** | UC-4 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Chest collection case | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Austin Jeffery |
| **Date Created:** | 2-22-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | This use case is for chest collision. | | |
| **Trigger:** | | The player character collides with a chest. | | |
| **Preconditions:** | | The user is in the maze view. | | |
| **Postconditions:** | | The chest sprite will disappear and the hint points will be updated. | | |
| **Normal Flow:** | | 1. The user will collide with a chest sprite. 2. The users will gain 150 hint points. 3. The chest sprite will be destroyed. | | |
| **Alternate Flow:** | | None. | | |
| **Frequency of Use:** | | Whenever the user collides with a chest. | | |
| **Assumptions:** | | None. | | |

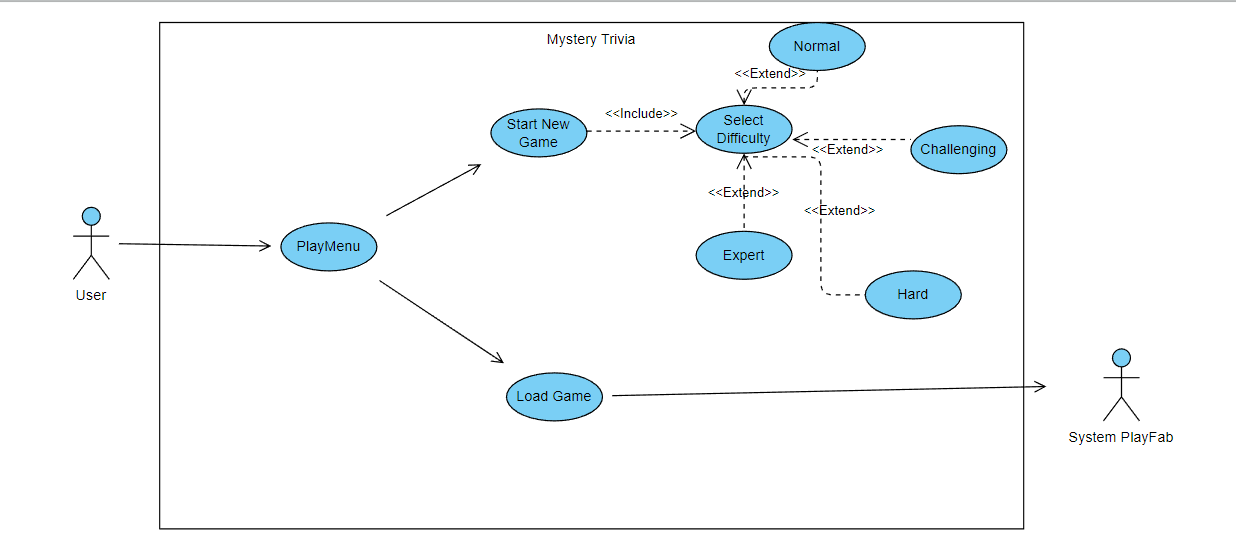
| **Use Case ID:** | UC-5 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Player defeat | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Austin Jeffery |
| **Date Created:** | 2-22-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | What happens when the player character touches an enemy. | | |
| **Trigger:** | | The player character collides with an enemy. | | |
| **Preconditions:** | | The user does not have a question open while in the maze view. | | |
| **Postconditions:** | | 1. The player has been placed at the beginning of the maze.  2. The user’s hint points have been set to their original points. | | |
| **Normal Flow:** | | 1. The player collides with an enemy. 2. The player death animation will play. 3. The player will spawn at the start of the maze with the hint points that they started with. | | |
| **Alternate Flow:** | | None. | | |
| **Frequency of Use:** | | Whenever the player character collides with an enemy. | | |
| **Assumptions:** | | None. | | |

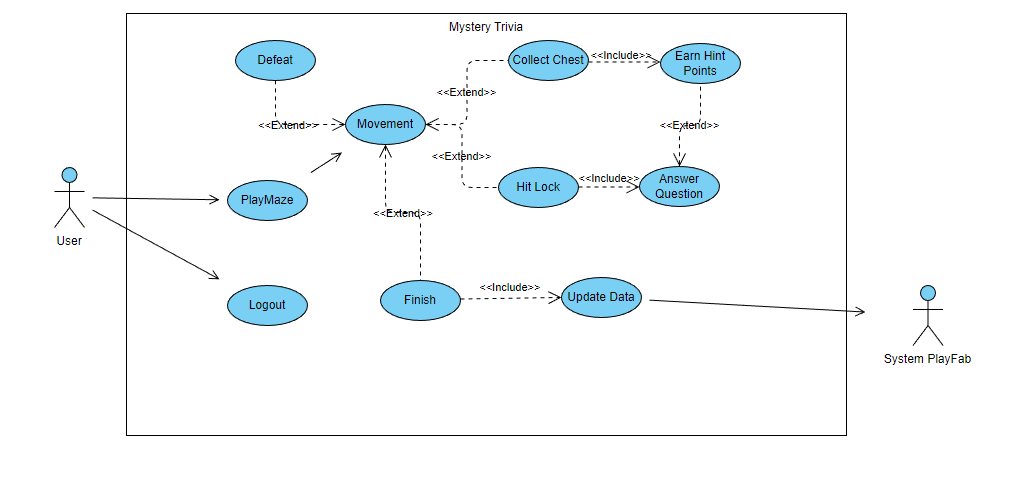
| **Use Case ID:** | UC-6 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | User login | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Austin Jeffery |
| **Date Created:** | 2-28-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | What happens when the User wishes to log into the application. | | |
| **Trigger:** | | The user clicks on the login button. | | |
| **Preconditions:** | | The user has text in the email and password text fields. | | |
| **Postconditions:** | | 1. The user will be shown either error messages if the incorrect email or password were used. The input text fields will also shake to grab their attention. 2. The user has entered the correct information and will be displayed a success message and be taken to the play menu screen. | | |
| **Normal Flow:** | | 1. The user has entered their email and password. 2. The user clicks on the login button. 3. The user will be taken to the play menu screen. | | |
| **Alternate Flow:** | | None. | | |
| **Frequency of Use:** | | Whenever a user logs into the application. | | |
| **Assumptions:** | | None. | | |

| **Use Case ID:** | UC-7 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | User registration | | | |
| **Created By:** | Jason Marrone | | **Last Updated By:** | Jason Marrone |
| **Date Created:** | 2-28-2023 | | **Last Revision Date:** | 3-2-2023 |
| **Actors:** | | User | | |
| **Description:** | | What happens when the User wishes to log into the application. | | |
| **Trigger:** | | The user clicks the register button. | | |
| **Preconditions:** | | None. | | |
| **Postconditions:** | | 1. A new account will be stored in playfab containing the entered data. | | |
| **Normal Flow:** | | 1. The user fills the email, password, and confirm password fields. 2. Playfab checks to see if that email is taken. 3. An email is sent to the entered address. 4. The user opens the entered email account. 5. The user finds the confirmation email. 6. The user selects the confirmation link. | | |
| **Alternate Flow:** | | None. | | |
| **Frequency of Use:** | | When the user decides to make an account. | | |
| **Assumptions:** | | 1. The user has an email account. | | |

| **Use Case ID:** | UC-8 | | | |
| --- | --- | --- | --- | --- |
| **Use Case Name:** | Reset User Password | | | |
| **Created By:** | Austin Jeffery | | **Last Updated By:** | Jason Marrone |
| **Date Created:** | 3-2-2023 | | **Last Revision Date:** | 3-6-2023 |
| **Actors:** | | User | | |
| **Description:** | | In case a user forgets their password associated with their email, they will be able to request to reset their password by clicking on the reset password button and filling out their email. Once they enter their email and submit, an email will be sent where they will click on a link to enter a new password. | | |
| **Trigger:** | | Click on reset password button | | |
| **Preconditions:** | | They are on the reset password screen. | | |
| **Postconditions:** | | 1. They will be notified with a message saying an email has been sent. 2. They will be brought back to the login page where they must enter their new password. | | |
| **Normal Flow:** | | 1. The user clicks on the reset password button. 2. The user enters their email and submits. 3. A notification will say an email has been sent for them to reset their password 4. The user goes to their email and opens the email where they will have to click on a link. 5. The link will take them to a website where they will enter their new password 6. They will go back to the login screen and enter their new password to login. | | |
| **Alternate Flow:** | | 1. The user emails one of the developers listed on the website 2. The developer manually changes it through the database | | |
| **Frequency of Use:** | | Anytime a user forgets their email and wishes to reset it. | | |
| **Assumptions:** | | None. | | |

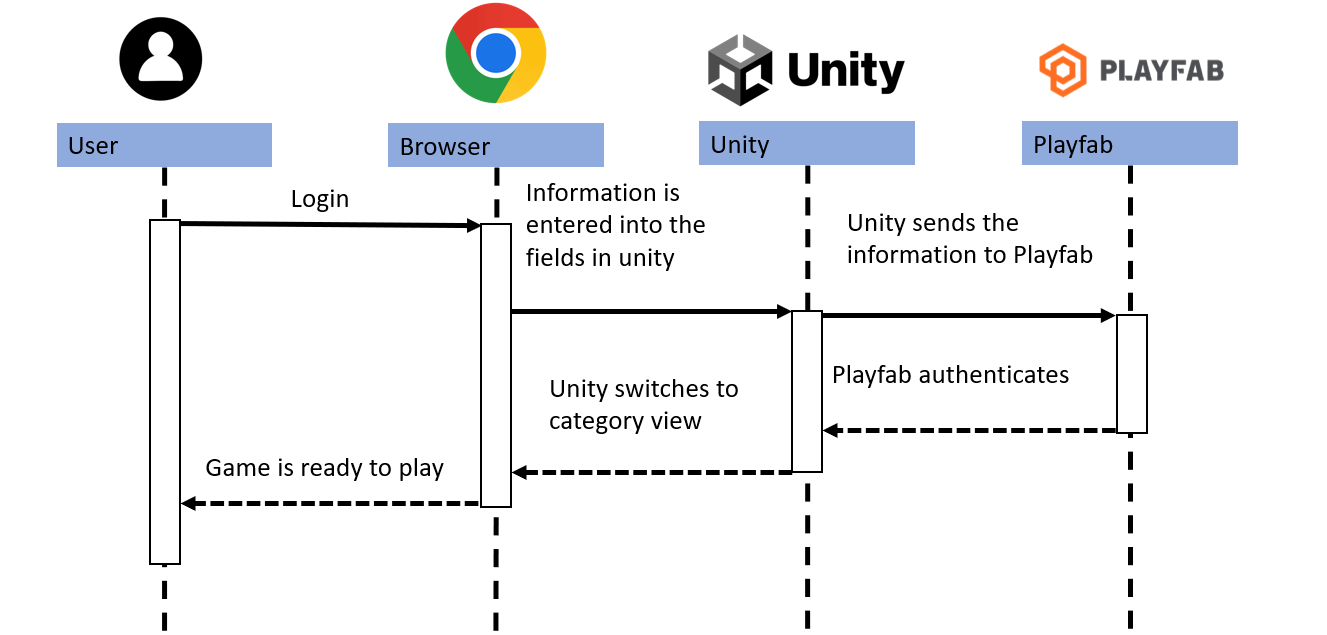
*Fig. 4.4.1*

*Fig.4.4.2*

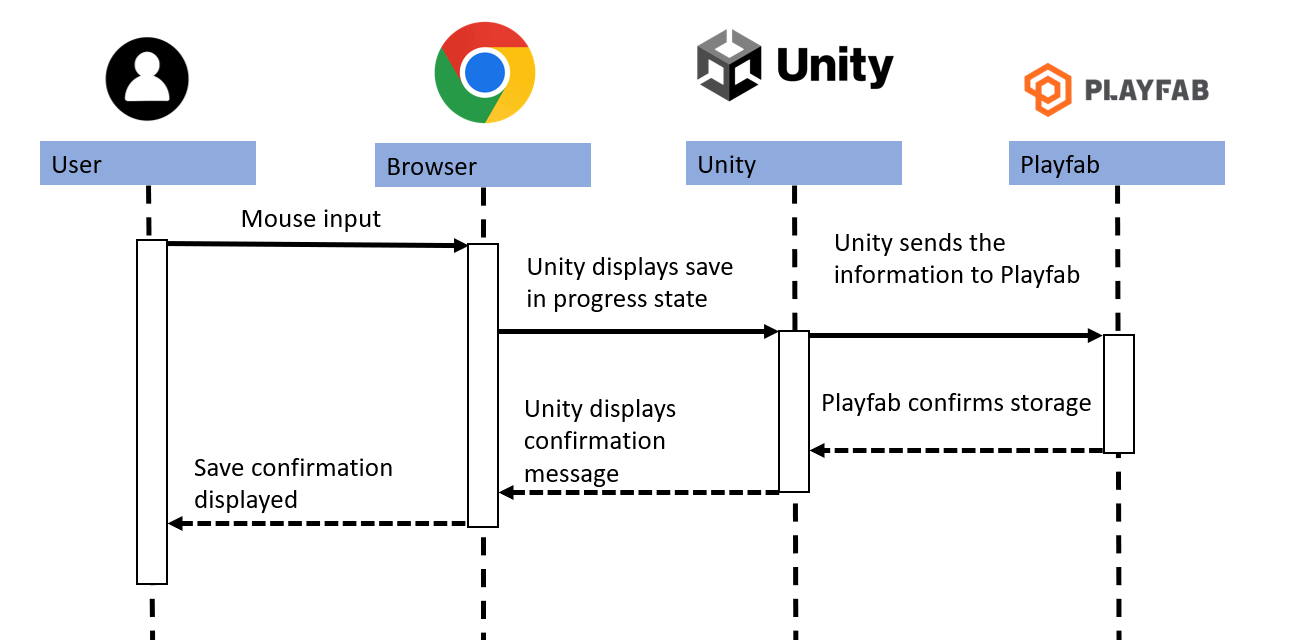
*Fig.4.4.3*

## 4.2 Sequence Diagram

### Fig. 4.2.1 Logging into the application.



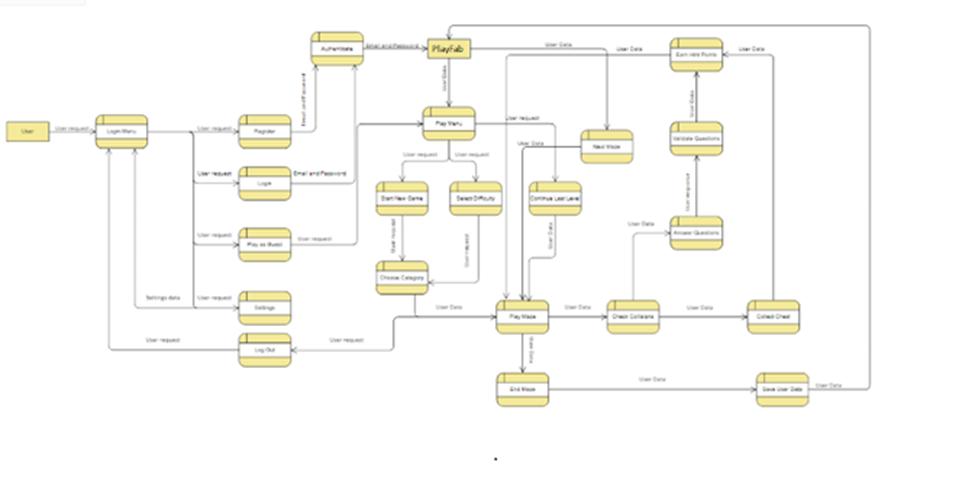
### Fig. 4.2.2 Saving user progress.



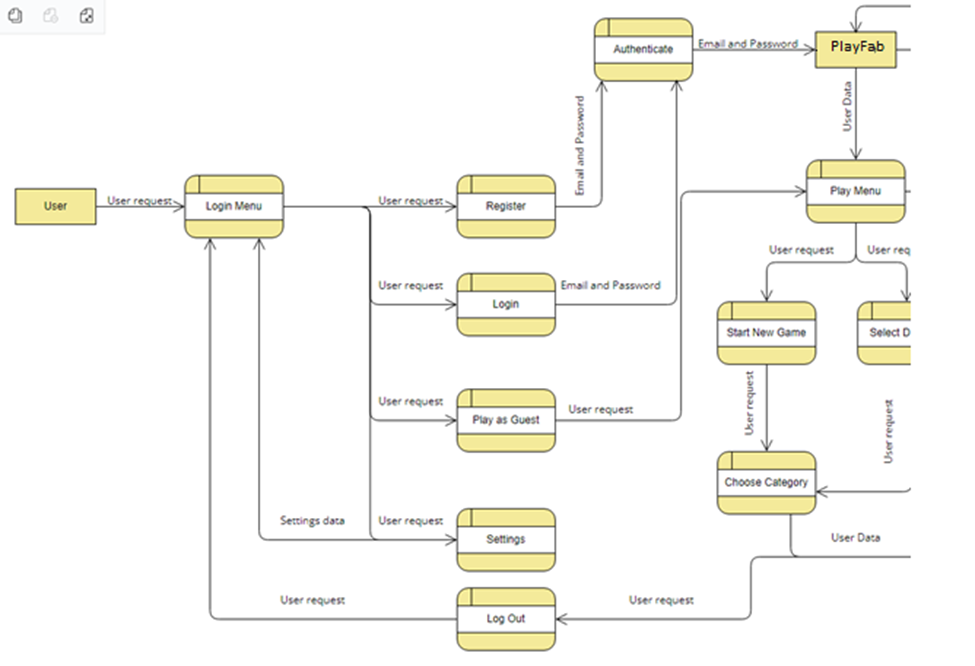
### Fig. 4.2.3 Registering for an account

## 4.3 Data Flow Diagram

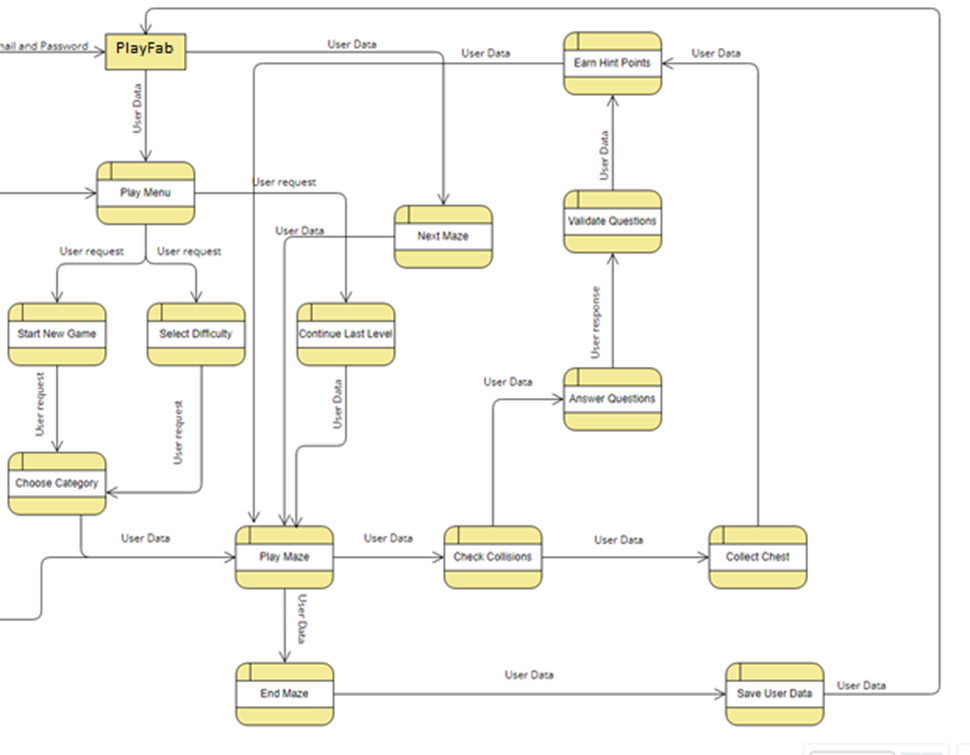
### Fig. 4.3.1 Full data flow diagram.



### Fig. 4.3.2 First half data flow diagram.



### Fig. 4.3.3 Second half data flow diagram.



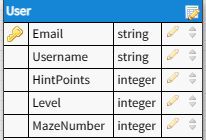
## 

## 

## 

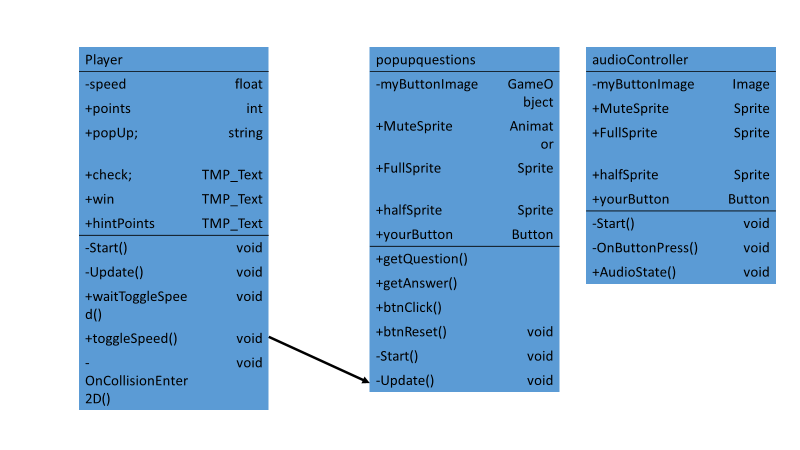
## 4.4 Database Design

As for our database, it is a NoSQL database where it only has one table to hold information for the user. This is to keep track of the users current progress and hint points for when they decide to come back to play again. We are using NoSQL as we don’t have much data to store besides the users progress. Using PlayFab we are able to store the users data so that when they leave and decide to join again, they will be able to pick up where they left off.



*Fig. 4.4.1*

## 4.5 Class Diagram



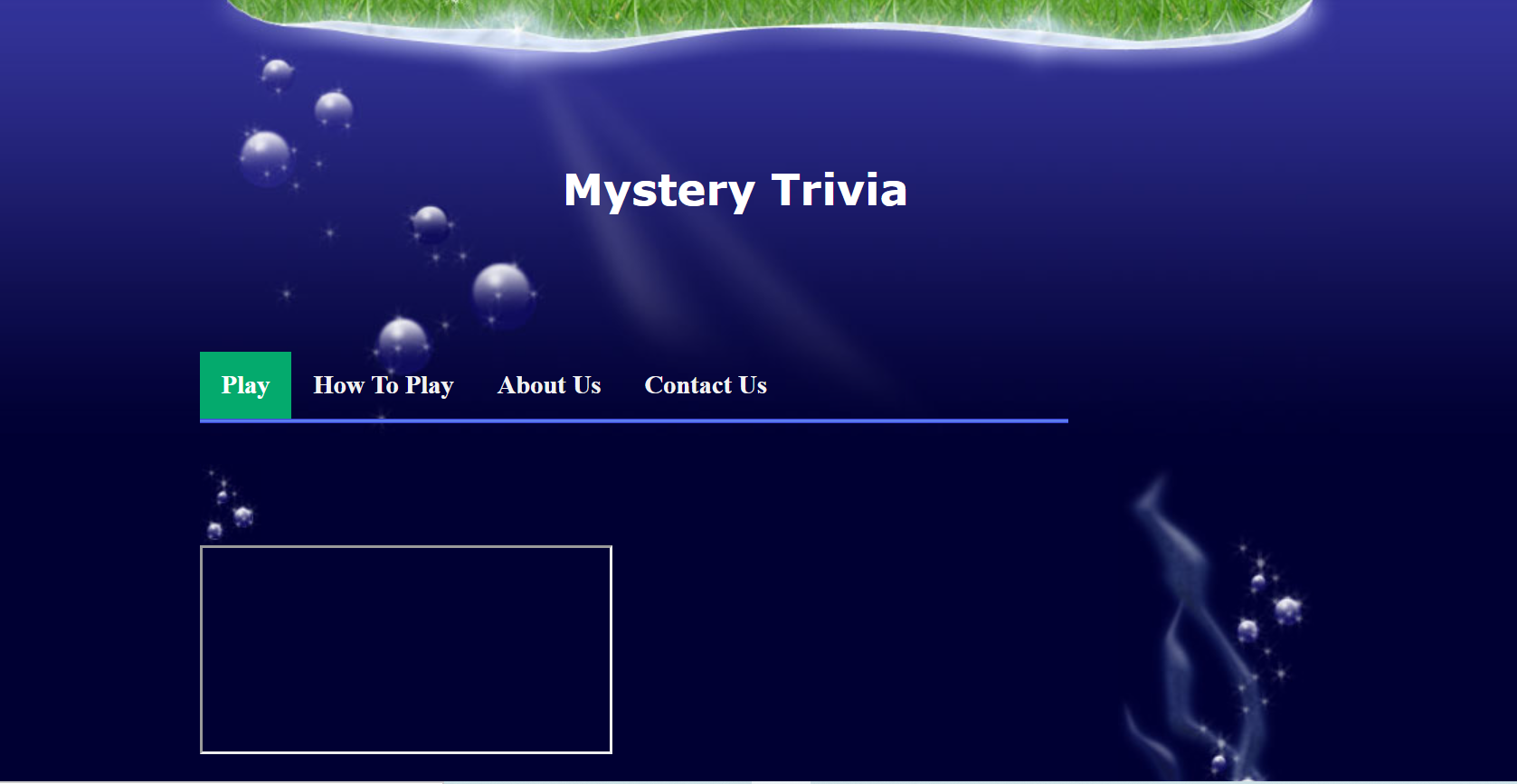
*Fig. 4.5.1*

6.7. Applications Program Interface:

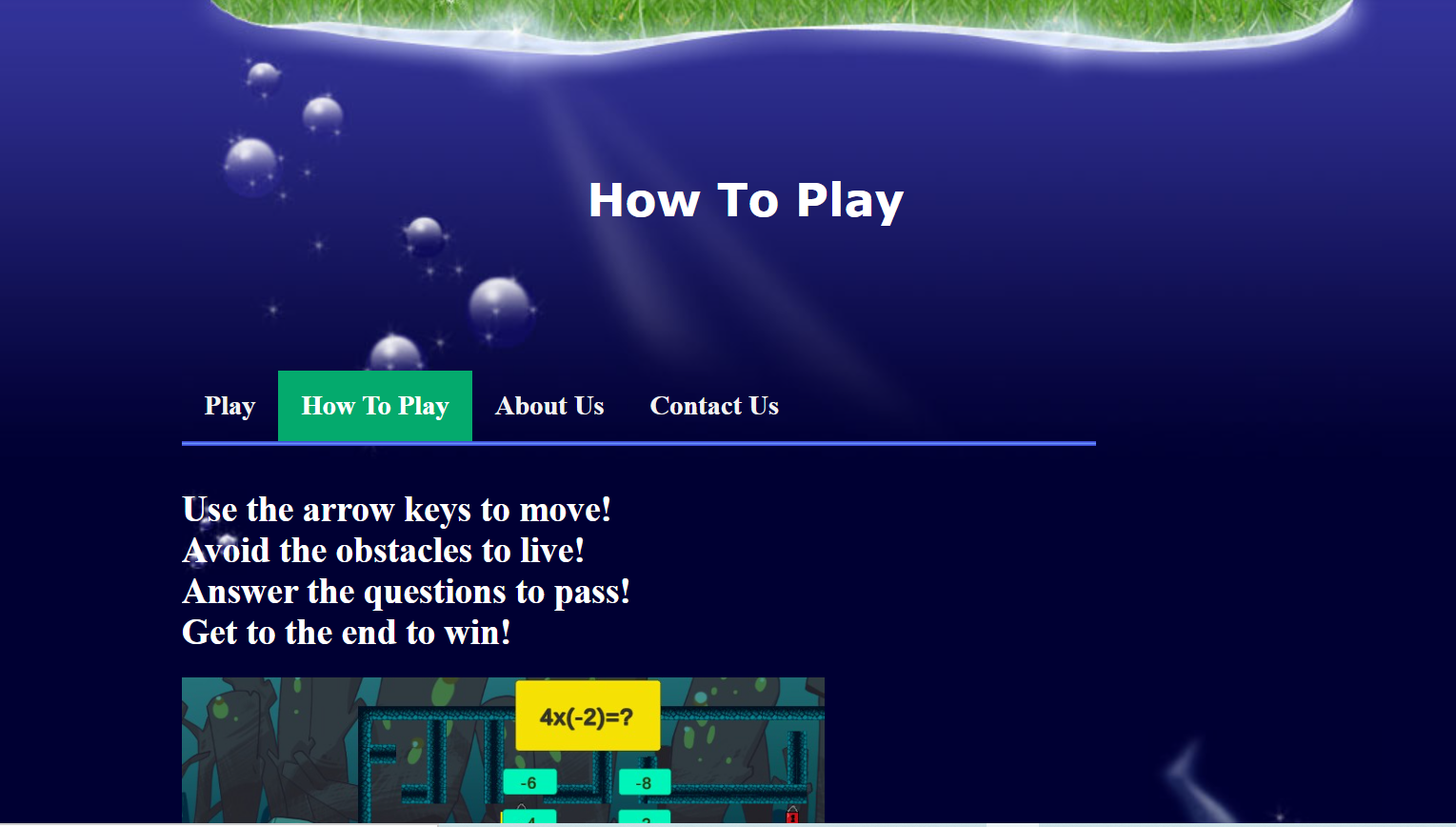
PlayFabAPI: https://learn.microsoft.com/en-us/gaming/playfab/

Unity API: <https://docs.unity3d.com/ScriptReference/PlayerSettings.WebGL.html>

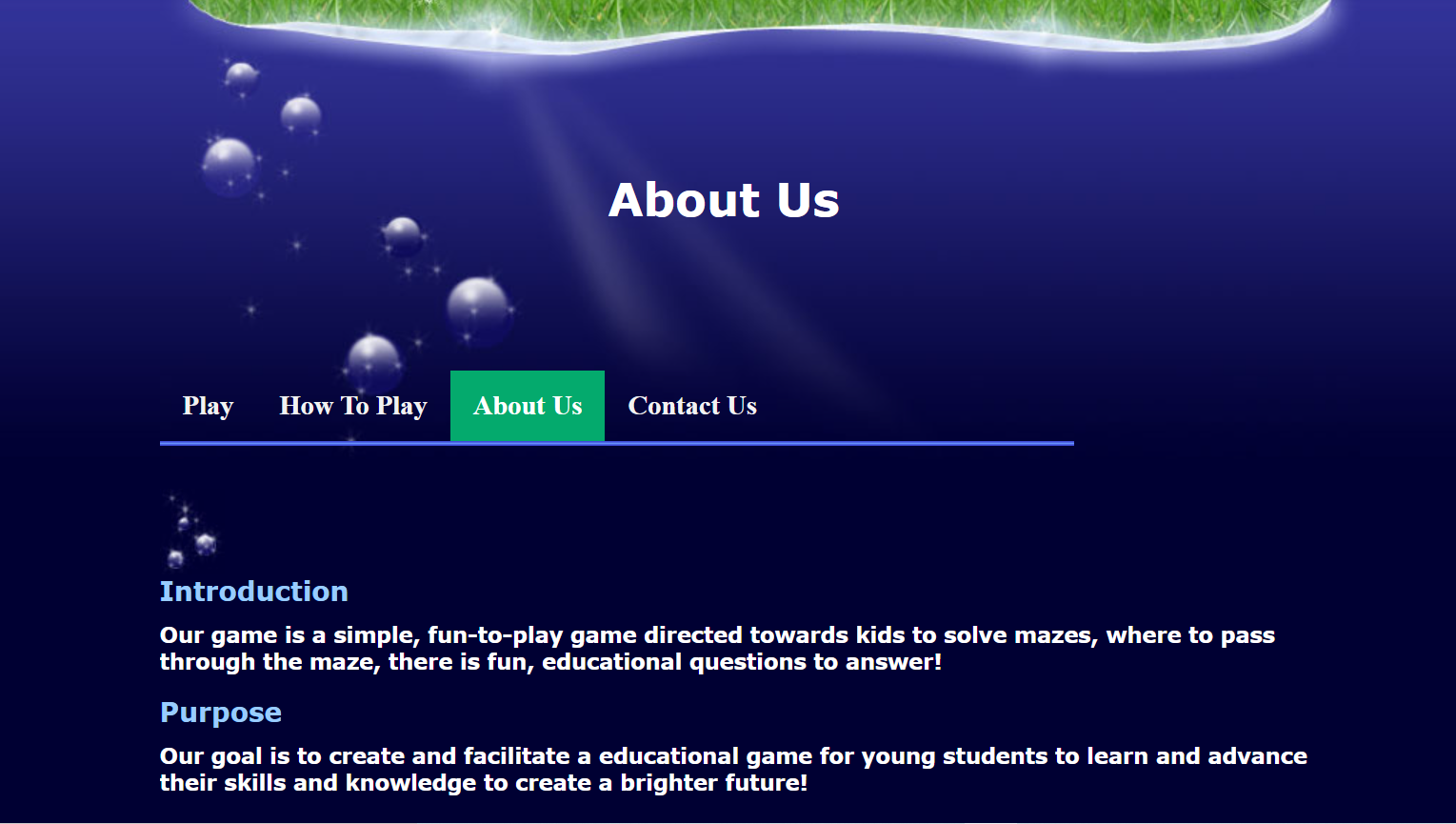
6.8. User Interface Design:



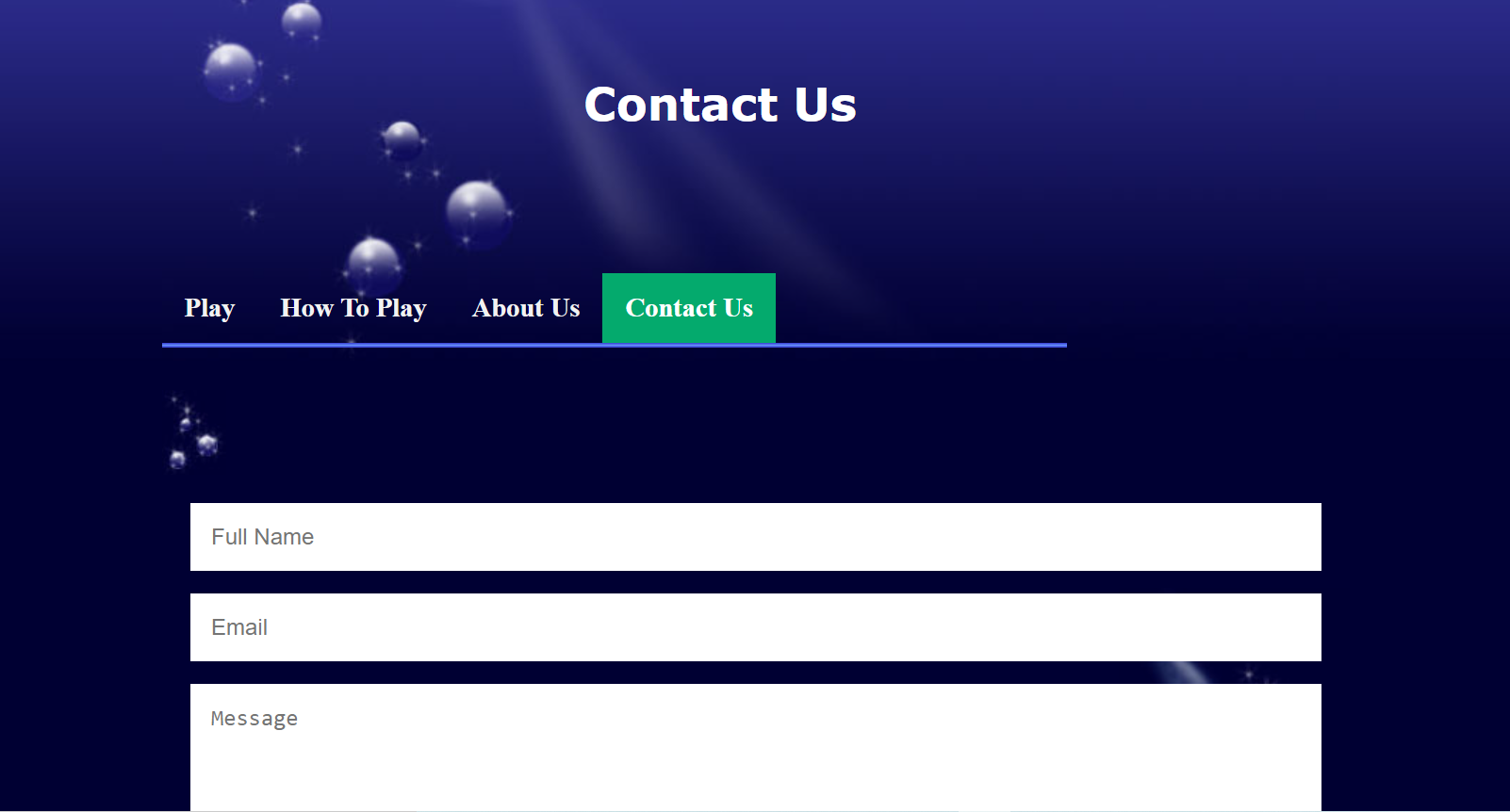
*Fig. 6.8.1*



*Fig. 6.8.2*



*Fig. 6.8.3*



*Fig. 6.8.4*

**

*Fig. 6.8.5*

**

*Fig. 6.8.6*

**

*Fig. 6.8.7*