Gamification of Computer Based Assessments: Variant 3

6CCS3PRJ Final Project Report

Author: Joel Jolly

Course: BSc Computer Science

Supervisor: Dr. Senir Dinar

Student ID: 21067140

April 2024

Abstract

This project explores the integration of gamification principles into computer-based assessments (CBAs) and its impact on assessment practices. Gamified CBAs leverage game elements to enhance user engagement, motivation, and learning outcomes. Through comprehensive review of existing literature, this report plans to examine the foundations, design, and effectiveness of gamified CBA in the historical context.

The report highlights the role CBAs play in promoting motivation, fostering active participation, and providing meaningful feedback. Through the evidence and case studies, it offers insight into how gamified CBA can cater to diverse learning styles, promote collaboration, and enhance the assessment experience overall.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary.

I grant the right to Kings College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service.

I confirm this report does not exceed 25,000 words.

Signed: Joel Jolly

Date: April 2024

Acknowledgements

I would like to express my deepest gratitude to Dr Senir Dinar, whose guidance, encouragement, and expertise has been invaluable throughout the process of completing this report. His insightful feedback and encouragement have deeply improved my understanding of the subject matter and have inspired me to strive.

Contents

1 Introduction

1.1 Project Motivation:

Traditional assessment approaches in education and evaluation have encountered several challenges and obstacles that halt the learning process of students. There is a demand for creative solutions that will improve the educational experience overall and speed up the evaluation process so that students can benefit from their educational journey.

Conventional exams such as your traditional pen-and-paper exams or standardized tests tend to expose students to a lot of stress, anxiety and in some cases, hinder the true reflection of a student's capabilities. Seeing this difficulty is the key behind my motivation to develop a computer-based assessment game, which allows me to help foster the way we evaluate and encourage intellectual growth.

The proposed computer-based assessment game aims to bring a dynamic and interesting approach to evaluating, avoiding traditional techniques that may unwittingly hinder the learning process. I hope to make evaluations less intimidating, and reflective of a student's genuine potential by automating and gamifying the assessment process.

My shift to computer-based assessments aims to reduce stress associated with the old evaluation techniques while also making learning more accessible. I hope to empower teachers, engage students more actively and help redefine how we view and carry out assessments in the education sector altogether with this initiative.

1.2 Scope:

My vision for the project is to create an endless runner game tailored for computer-based assessments. It plans to combine the thrill of an endless runner game with the depth of educational assessments.

At the heart of my game is a commitment to creating an immersive gameplay experience creating an immersive gameplay experience that goes beyond traditional evaluation methods. I hope to change how students engage with educational content by seamlessly integrating multiple choice assessment questions into the gameplay of an endless runner. The goal is to create a blend in which students answer questions while trying to avoid obstacles, which hopefully can educate, inspire, and engage the students.

This project plans to go beyond mere entertainment; it plans to use a dynamic educational framework that adapts to the students' capabilities and caters to different levels of proficiency. I intend to break down the concept of every student going through boring formats and make education different and tailored to every student, so learning becomes an exhilarating journey. As a student plays more, not only will they find entertainment in the game, but also take a step forward in their intellectual growth.

Not only will this help students in their educational path but will also support teachers assess which student can do what in certain areas of history. Allowing them to maintain certain standards in their class and allowing them to understand each student differently.

1.3 Stakeholders:

At the heart of my endless runner game for kids learning history, there are 2 primary stakeholders; one being the teachers, who shape the learning experience, and the students, who play the game and take part in an educational adventure.

Teachers play a pivotal role in ensuring the questions they set aligns with the key stage 2 curriculum and ensures the questions asked are valuable for the students to learn from. The feedback and collaboration they get from students playing my game will be crucial in tailoring classes specific for the students.

The main stakeholders of our game are the students. They will be of key stage 2 level, meaning aged 7 to 11, and they game will be easy enough for them to interact with and play. Their engagement and comprehension will be the main key to my project. By creating a game that is interactive and enjoyable, while at the same time making it crucial to their studies, it will be well suited for my motivation set up top.

2 Background

2.1 Background Introduction:

The integration of a gamified principle into computer-based assessments (CBAs) has emerged as a promising method to enhance learning outcome, motivation and mainly engagement. This background report explores the potential a gamified CBA could hold, with the focus being on an endless runner style game. The aim is to examine the foundations, design, and effectiveness of this approach in the context of assessment practices. The endless runner style game provides a unique opportunity to make assessment more engaging and effective through its simplicity, continuous play, and progressive difficulty. This review will delve into the literature and case studies that support the use of a gamified CBA and the endless runner style game, along with discussing possible challenges with possible solutions it can hold. It will also explore how a gamified CBA can cater to diverse learning styles, promote collaboration, and enhance the overall learning experience overall. This review provides a background for understanding the evolution of assessments and the role that gamification plays into enhancing the assessment experience.

Assessments have been a cornerstone of education, providing a way to measure students' understanding, capabilities and skills. With the rise of technology, these assessments have evolved into the computer-based assessments we are used to, which offers a more interactive and engaging learning experience. The application of game-design elements in a non-game context has further enhanced the potential of these computer-based assessments. One example of said game-design elements is the endless runner style game, which gives the user a unique opportunity to make these non-game CBAs into a more engaging and effective way. However, this style of game is particularly suited to the educational context as it allows for continuous and on the stop feedback as well as multiple instances of examination, which can be adapted to cater diverse learning styles. Furthermore, the endless runner concept can be designed to promote collaboration and active participation, therefore enhancing the overall assessment experience.

2.2 Literature Review:

Through my review of existing literature, it is revealed that there is a growing body of evidence that supports the effectiveness of gamified computer-based assessments. These studies have shown that gamified CBAs can enhance user engagement, promote collaboration, and enhance the overall assessment experience altogether. Links to the literature and case studies can be found in the references page.

One of the key finds in the literature is the positive impact of gamification on user engagement. Gamified assessments have been found to increase candidate engagement, to test motivation, improve satisfaction and decrease test anxiety in several students. This is particularly important in high stakes psychometric testing environments, as test anxiety can play a crucial role in when it comes to impacting performance levels. By incorporating game elements into these assessments processes, the candidates are more likely to take their minds of the importance of the exam and tend to focus on the gaming aspect which in turn can lead to improved performance and more accurate assessment outcomes. (1)

In addition to enhancing user engagement and motivation, gamified CBA has also been found to promote collaboration. Many of these assessments have been found to incorporate collaborative elements, such as cooperative tasks or team challenges, which require these students to work together to get to a common goal. Not only does this enhance the assessment experience but it also allows for the assessment of important collaborative skills, such as cooperation, conflict resolution and communication which are ideal traits for a teacher to teach for personal development in the future. (2)

Despite these promising findings, the literature also highlights several challenges associated with gamified CBAs. These findings include the risk of bias, design complexity, cost and development difficulties and other overlaying issues. Many of these challenges however can be addressed through careful design and implementation, using effective adaptive algorithms to cater to diverse learning tyles and to help with ongoing research to address psychometric issues. (3)

There are several case studies that show the successful implementation of gamified CBAs. For instance, a study conducted by Zhihui Zhang and Jenifer Crawford investigated the motivation of EFL learners (English as a Foreign Language) in a gamified formative assessment using an application called Quizizz. The study found that Quizizz, as an example of the gamified formative assessment, can enhance the EFL learners' internalization at a higher level during their learning process. Furthermore, the overall accuracy of the students I the experimental group who managed to get 89.05% was substantially higher than the control group's accuracy which was only 74.01%. (4)

Also using the following case study, I was able to generate a framework on how my game could be made, as it presents a game-based assessment framework (GBAF) that is designed to be educator-friendly and useful for the design of assessments which can benefit the immersive learning environments. The application of the framework to the design of embedded and external (multiple-choice questions) assessments, which I plan to use in my code, is also presented in the case. The GBAF offered an easy and structured basis for the design of two distinct assessments for the game which can assist game developers as well as teachers for designing assessments around preexisting games. (5)

In conclusion, the literature provides strong support for the use of gamified CBAs. While there are challenges to be addressed, the potential benefits for this kind of approach, including enhanced motivation, collaboration, user engagement and learning outcomes, makes it a promising are for further exploration and development.

2.3 Why Endless Runner?

Endless Runner games like Subway Surfers and Temple Run have captivated millions of players across the world with their endless runner mechanics, colourful visuals, and addictive gameplay.

Subway Surfers is a 3d endless runner game, that involves the main characters, Jake, Tricky and Fresh who run away from an inspector. During their run they must avoid trains and other obstacles that come in their path, while collecting coins and special perks such as jetpacks and pogo-sticks. The gameplay takes place between 3 lanes in which the players can switch between from. Subway Surfers was one of the first mobile game to reach one billion downloads on Google Play and the first mobile game to reach four billion downloads across the Apple App Store and Google Play. It manages to have around 20 million daily active users and over 150 million monthly active users. (6)

Temple Run is also a 3d endless runner game, where the main character, Guy Dangerous, an explorer, has stolen an idol from a temple is being chased down by a group of demonic monkeys. In this game you need to run, jump, slide and collect coins. The key difference of this game from Subway Surfers is that the player can choose to can the flow of the game by swiping left and right. Temple Run has been downloaded more than a billion times by people from all over the world.

These statistics clearly highlight the immense popularity of the endless runner genre and its potential for engaging a massive audience.

In addition to Subway Surfers and Temple Run, another popular endless runner game is Alto's Odyssey. Alto's Odyssey is a follow-up to Alto's Adventure and has you reprising your role as a snowboarding hero, through endless sandy dunes. Players can build up momentum by flinging Alto into the air and can perform various stunts while speeding through atmospheric environments with stunning visual 2D visuals. For visual based gameplay I have taken inspiration from this game to be visually pleased. (7)

In the context of CBAs, the endless runner style can provide several benefits to my game. First, the simplicity of the controls combined with the low-intensity, repetitive gameplay makes the endless runner genre highly accessible and engaging, making it ideal for students at the suggested age level. Second, the continuous play and progressive difficulty inherent in endless runner games can provide a dynamic and challenging assessment experience. Third, the collection of coins and power-ups in games like Subway Surfers, Temple Run and Alto's Odyssey can be used as a form of formative assessment, which can provide immediate feedback to learners about their performance so they can adapt and improve on the spot.

Moreover, the success of Subway Surfers, Temple Run and Alto's Odyssey can be attributed t their focus on maintaining the core mechanics of the endless runner genre while avoiding unnecessary complications. For instance, Subway Surfers has continued to add more and more users to its player-base without having to create a second game or adding major changes to the game. This strategy focusing on the core endless runner mechanics while avoiding the complexity could be a valuable lesson for the design of gamified CBAs.

2.4 Challenges and Potential Solutions

Despite a gamified CBA having promising potential, there are several challenges that need to be addressed:

Risk of Bias: The first challenge is the risk of bias. Gamified assessments can be influenced by a range of factors, including the candidate's familiarity with gaming, their cultural background, and their physical abilities. This could lead to biased assessment outcomes. To address this, we can ensure that the game design is culturally sensitive and accessible to all students, regardless of their gaming experience and physical abilities.

Psychometric Issues: The second challenge is related to psychometric issues. These include increased anxiety for candidates completing in high stake exams and this could lead to performance in students being altered. Multiple-choice questionnaires could encourage students to take on a guessing mentality and having the examination go on for too long can result in the students being fatigued. Some tests can also be quite time intensive or may suffer from "basement effects" or "ceiling effects" in which case the students will have reached the upper and lower limits of measurements that the test will allow. Ongoing research will be needed to address these psychometric issues and ensure the reliability and validity of gamified CBAs.

Design Complexity: The third challenge is the complexity of designing effective gamified assessments. Creating interactive, fun games that does not feel like quizzes pretending to be games takes time and creativity. Assessments in games need to link back to the learning objectives, just like any other online learning course.

Motivation: The fourth challenge is mainly motivation. While gamification is used to enhance motivation, it is very important to note that not all users are motivated by the same things. Events such as getting questions right or collecting stars might not be enough to motivate students playing the game. To address this issue, gamified CBAs could incorporate a variety of motivational elements, such as meaningful feedback, rewards, and progress tracking.

2.5 Background Conclusion

In conclusion, gamified CBAs, particularly those incorporating the endless runner style, offer significant potential to enhance user experience, learning outcomes and the overall assessment environment. Despite the challenges, the benefits of this approach make it a promising area for further exploration and development in the field of education and assessments.

3 Requirement & Specification

3.1 Requirements

The purpose of this project is to make an endless runner game that works as a computer-based assessment which after every run provides feedback to the teacher. The teacher should be able to upload a file containing the questions which can be accessed by the player when they play the game.

3.1.1 Game Functional Requirements

These are requirements that should be implemented when the user is playing the game.

- G1 – The application should let the student move and control the player.
- G2 – The application should generate obstacles dynamically at regular time intervals.
- G3 – The application should keep track of the player's score based on the distance travelled.
- G4 – The application should be able detect collisions between players and obstacles.
- G5 – The application should display questions that the students can see and answer accordingly.
- G6 – The application should provide the user with feedback on how they did on the question that they answered.
- G7 – The application should have a way of ending the game after a certain event has occurred.
- G8 – The application should consist of a difficulty level/scoring system for when they play the game for longer.

- G9 – The game could feature a leaderboard system that tracks and displays the top scores achieved by players.
- G10 – The game could allow players to choose or customize their character's appearance.
- G11 – The game could implement power-ups that can appear randomly during the gameplay, providing temporary advantages such as invincibility, speed boost, extra points or give additional lives.
- G12 – The game must save the data of each of the run the player has played so it can be viewed by the teachers later.

3.1.2 Non-Game Functional Requirements

These are requirements that should be implemented when the user is not playing the game.

- N1 – Teachers should be able to upload a set of questions onto the application.
- N2 – Teachers should be able to save the questions they have uploaded.
- N3 – Teachers should be able to navigate through the questions they have uploaded onto the game.
- N4 – Teachers should be able to see the statistics of all the runs the player has made.
- N5 – Teachers should be able to see all the questions the students have managed to answer correctly.
- N6 – Both students and teachers should have a responsive and intuitive user interface, which allows them to navigate and interact with the said features.
- N7 – Teachers can reset a set of runs if they want to, for a time a new student is going to do the exam.
- N8 – Login/Register system so students can save their scores and can track progress.
- N9 – Have a tutorial or help section which helps students to play the game and navigate through the application.

3.2 Specifications

The table below shows the importance of each requirement. The high priority requirement is essential for the project's completion, while low priority requirements, although useful, are not critical. It is crucial to prioritize high priority requirements and address lower priority ones subsequently to ensure basic functionality.

3.2.1 Game Functional Specifications

Requirement	Requirement Detail	Specification	Importance
G1	Allow player movement and control	Implement player movement script with user input controls, mainly using left and right arrow keys	High
G2	Generate obstacles dynamically	Implement an obstacle generation script that spawns objects are regular intervals	High
G3	Track player's score based on distance travelled	Implement scoring systems linked to player movement using Z positions	High
G4	Detect collisions between player and obstacles	Using Unity collision boxes and scripts find collisions between player and obstacle game objects	High
G5	Display questions for players to answer	Integrate text UI for displaying questions during gameplay	Medium
G6	Provide feedback on player's question responses	Implement a feedback UI which displays whether the student got the question right or wrong	Medium
G7	End game after certain events	When no of lives during run are over, end the game.	High
G8	Implement difficulty level/scoring system	Integrate difficulty levels affecting obstacle speed, frequency, and player scoring	Medium

G9	Implement leaderboard system	Integrate online leaderboard functionality to display top scores	Low
G10	Allow character customization	Integrate character customization options for players	Low
G11	Implement power-ups	Create a power-up system with random appearance during gameplay	Low
G12	Save data of each of the times the student has played the game	Implement a data storage system which stores all the data about the current run	High

Table 3.2.1: Gamified CBA Application Requirements Table

3.2.2 Non-Game Functional Specifications

Requirement	Requirement Detail	Specification	Importance
N1	Allow teachers to upload questions	Implement UI and backend functionality for question uploading using a JSON format	High
N2	Enable saving of uploaded questions	Implement a storage system for uploaded question	High
N3	Enable navigation through uploaded questions	Implement UI for teachers to navigate through uploaded questions	Medium
N4	Provide statistics of each of the player runs	Implement backend tracking of player statistics and calculations and use UI for teachers to view them	High
N5	Display the questions each of the student answered, showing whether its right or not	Implement UI for teachers to view a complete list of all the questions the students have answered correctly.	Medium
N6	Ensure intuitive user interface	Design effective UI/UX adhering to Unity’s best practices for intuitive navigation	High

N7	Allows teachers to reset runs	Implement a functionality which allows teachers to reset each of the players runs	Medium
N8	Implement a login/register system	Develop a login/register system that allows both teachers and students to create accounts and log in securely	Low
N9	Have a tutorial on how the game functions	Design and implement a tutorial mode that guides users through the game's mechanics	Low

Table 3.2.2: Gamified CBA Application Requirements Table

3.2.3 Limitations

Some limitations for the applications might include:

- Ensuring user engagement and retention may be challenging, especially if the game lacks compelling gameplay mechanics or fails the capture users' interest over time. Continuous updates and engagement from the class may be necessary to maintain user interest and keep them playing all the time.

- The system's storage capabilities might be limited, especially on old generation devices where storage system is often constrained. This limitation may impact the amount of data that can be stored locally, such as saved runs for each of the player or question locations.

- The system may be limited to specific platforms that only support Unity, such as Windows, macOS, Linux, iOS, Android, and certain web platforms. Compatibility with other platforms may require additional development effort.

- The performance of the game may be limited by the hardware specifications of the devices running it. Having high end graphics or complex gameplay mechanics may not be feasible on older or lower-powered devices.

4 Design

The design section outlines the step-by-step process of how the program functions in a simplified manner, supported by easy-to-understand figures. This overview helps in grasping the system's basics, preparing us for its implementation, which we'll further explore in the implementation section.

4.1 Use Cases:

We use use-cases to identify the interactions between the actor and the system. We have 2 actors in our system, the teacher, and the student. In the use cases below, we can show both users are modelled as actors in the use case diagrams. It is used to provide functionality and gives the main functions so the program can run as it should.

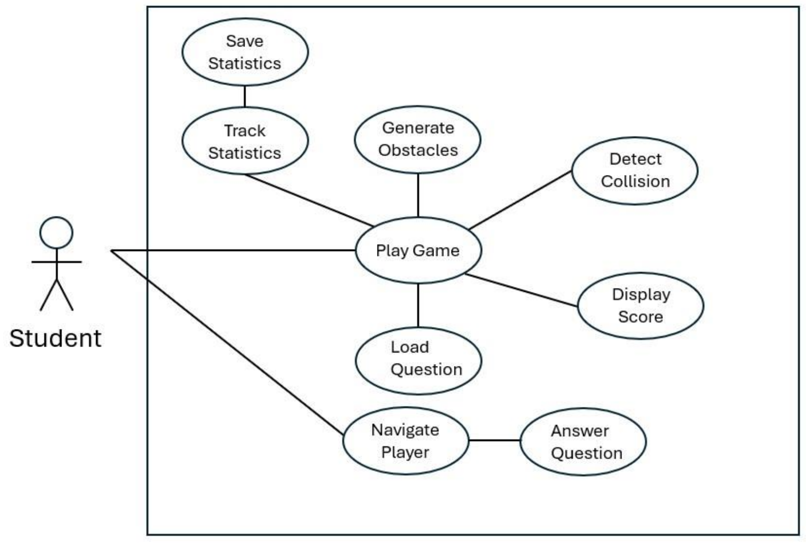


Figure 4.1.1: Gamified CBA Application Use-Case 1 Diagram

The student launches the game application and selects the option to start a new game. Which is the initial interaction that triggers the use case. Once the game starts, the system generates obstacles dynamically at regular intervals. This is the part of the game mechanics that should provide a sort of challenge for the player. The obstacles include the questions that the student must answer during their run. The student controls the player character to navigate through the obstacles. After the game has been running for a while, gates will pop up with question ahead, in which the student must answer by selecting the correct gate. If the student got it right a message will pop up which states whether they got it right or wrong. These are the main action the student performs during the game. As the student navigates through the game, the system keeps track of the student's score, number of questions they answered and the number of lives the student has. If the player collides with an obstacle, the system detects the collision and ends the game session if all the lives have been reduced to 0. This is the part of the game that saves all the statistics of the run which is then used so the teacher can have an analysis. After the game ends, the system allows them to restart the game by putting them into to the main menu.

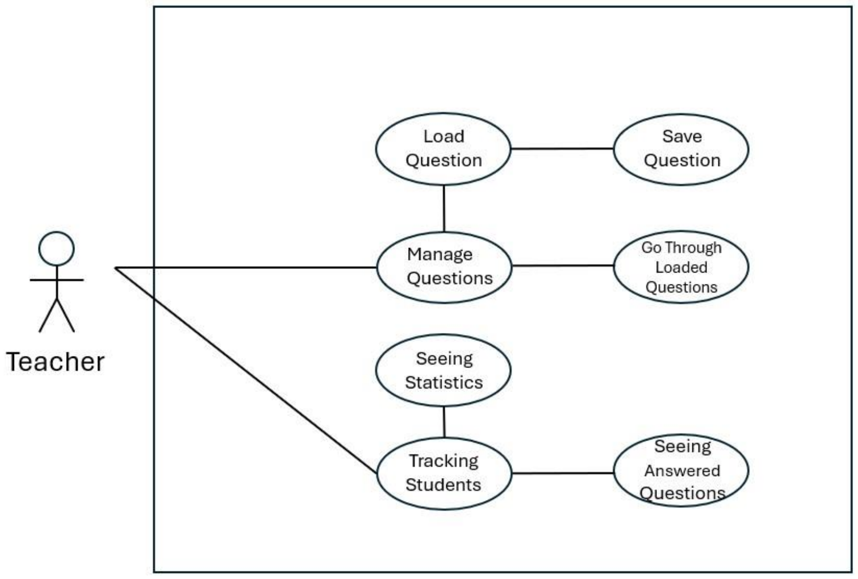


Figure 4.1.2: Gamified CBA Application Use-Case 2 Diagram

The teacher accesses the teacher dashboard of the application and selects the part which allows them to manage questions. This is the initial interaction that triggers this use case. The system allows the teacher to upload a set of question. The teacher has the option to then save the question they just loaded into the database. This is how teachers add new questions to the system. After uploading the questions, the teacher can then navigate through the uploaded questions using the provided interface. This allows the teacher to review the questions and make any necessary adjustments. The navigation of questions also shows the correct answers in the set of the questions. The teacher can then back out and then see what the tracking students' section. This is where the teacher can view statistics of all runs the player has made. This provides the teacher with information about the student's performance in the game, this can include the score, questions answered and improvement from previous runs. The system also displays all the questions the students managed to answer correctly. This gives the teacher insight into what students have learned.

4.2 System Architecture:

The system architecture serves as a roadmap for a system, breaking it down into subsystems and showing their interdependencies. A three-tier architecture diagram divides the system into three layers: the presentation layer (user interface), the logic layer (core functionality) and the data management layer (data storage and retrieval). This structure ensures each part of the system plays a specific role, enhancing understanding and maintenance while allowing for flexibility and scalability. It's a visual representation of how the system is organized and how data flows through it, providing valuable insights for developers.

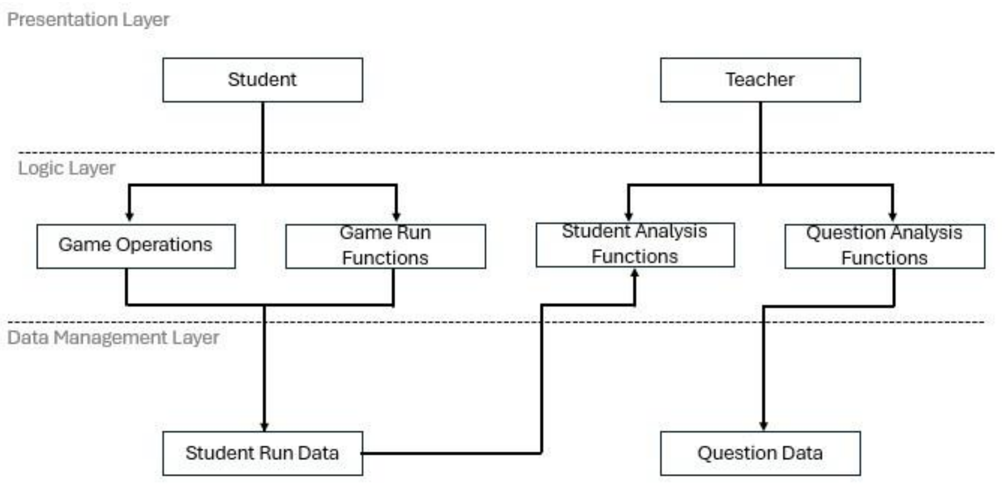


Figure 4.2: Gamified CBA Application Architecture Diagram

The architecture diagram (Figure 4.2) reveals there are two users in the presentation layer (them being Student and Teacher users), four system logic layer functions (Game Operations, Game Run Functions, Student Analysis Functions and Question Analysis Functions) and two data repositories (Student Run Data and the Question Data).

In the presentation layer, the student user is in charge of navigating the player and moving the player around in order to avoid obstacles and go into the correct lanes to answer the questions. The teacher user allows view questions and helps with navigating through each of the runs the students have made.

The logic layers hold the all the main algorithm involved to make the game functional and running smoothly and the analysis part of the students. The game run functions include all the functions required to make the game functional. This includes the spawning of obstacles, all features allowing player movements, question gate generation and so on. It is the main functions that are required to make the game run as an endless runner game. The game operations include all the algorithms that are used to save data of each run which include keeping track of the scores, questions answered and lives the player has. It is also beneficial to loading up questions the teacher has originally set and assigning it to the gates generated by the game run functions. The student analysis functions use the student run data to present statistics about each of the run the user has made. It is required to show the questions they have stored. And finally, the question analysis functions are used so the teachers can load and save the questions that are used and then it presents the questions so the teacher can see them and check if they are correct.

The Data management layer holds the data repositories which stores information regarding the run. It is also used to store the questions which are going to be used in the logic layers by its algorithms and functions.

4.3 Data Flow:

This part of the Design structure shows how the data is going to be loaded and used in the project. They will show the two distinct data repositories that are shown In Figure 4.2, which makes the data flow diagram important for our project.

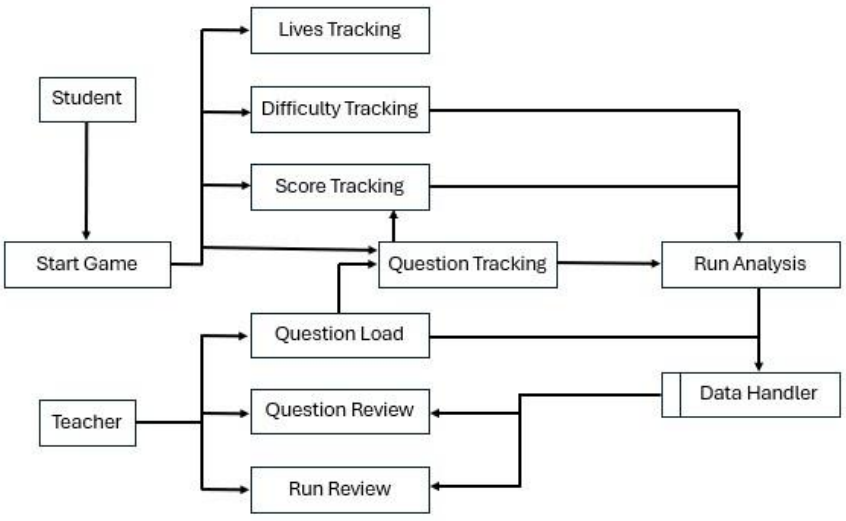


Figure 4.3: Gamified CBA Application Data Flow Diagram

Figure 4.3 shows that the on the student side of data handling, they must start the game. Upon starting the game all four main information regarding lives tracking, difficulty tracking, score tracking and question tracking. The lives tracking manages to keep the data regarding all the lives the player has during the run of the game. This is the only part of data tracking which does not get saved by the run analysis. The difficulty, score and question tracking are all constantly changing throughout the game. The score increases as the player get more and more distance. The difficulty tracking changes the score points gained according to the questions answered correctly. The question tracking is used to see how many questions have been answered and out of those how many questions correctly. The data for the question tracking is loaded from the question load feature from the teachers end. All three data tracking features are then put through the run analysis and is sent to the data handler which then stores them to each of the run. The information from the run is saved as a JSON file which keeps track of the score, question data and the question difficulty.

The teacher side of data handling mainly uses the questions that have been data handled and saved for each of the run. They use the runs to provide the teacher with a run review with each of the runs created and also a question review of which questions the students got right and wrong. The question load part of the diagram shows that the teachers load the question data, which consists of the questions which are going to be displayed in the game. These questions are can then be used for question tracking purposes on the student side or they can be saved by the data handler so it can be used for another time.

4.4 Graphical User Interface:

The user interface of our system serves as the primary means for users to interact with my gamified CBA application. It's crucial that the interface manages to strike a balance between simplicity and functionality to ensure user engagement without overwhelming them with too many options.

Given that my application is a game, one of our major focuses would be on the graphical user interface (GUI). The GUI design must cater to the gamified CBA, providing users with an intuitive and visually appealing experience that facilitates seamless interaction with the questions tasks that they must answer.

To achieve this, the GUI design should incorporate elements that are easy to navigate and understand, guiding users through the assessment process effectively. Clear and concise instructions, along with basic controls, will help the users navigate the application without feeling lost or confused.

Furthermore, while simplicity is paramount, it's crucial to ensure that the GUI encompasses the full functionality of the gamified CBA. This includes features such as having the ability to answer questions, viewing progress, receiving feedback, and interacting with gamified elements that the game consists of.

4.4.1 Start-Up GUI

When opening the application, the student is faced with playing the game, once they click the play game option, they are taken into the game in which the character will start moving forward. The start-up screen also allows the teachers to access the teacher's report too. From there they can navigate through the application to change questions or see the performance of students. There is also a settings options in the starting page as well which will show the students the basic controls to play the game as well as reset options so teachers can reset the runs and reset the questions as well.

4.4.2 Game GUI

When the player has entered the game, the student is automatically put into the game environment. When they are loaded into the game, the student can see the number of lives they have, their score as well as the question they must answer at the top of the screen. While the game continues moving forward, obstacles start spawning in, in which they must move the player, using the arrow keys, and avoid these said obstacles. Every time they collide with an obstacle, they lose a life which is then updated on the screen. While the game is running for a while, 4 gates will appear on the screen. The student must pick the correct lane which has the answer allocated to it. If they managed to pick the correct lane, they will gain score points and a correct answer message will pop up on the screen. However, if they manage to get the wrong answer, they will lose a life and then an incorrect answer message will pop up on the screen. As soon as they pass the gate a new question will pop up and have till the next gate spawning to choose the correct answer. These mechanisms will continue it all the lives the player has, has been used up and returned to 0 in which the student will be taken back to the main menu, where they have an option to restart the game again.

4.4.3 Teacher - Question Load GUI

When the teacher opens the question load page, they would be greeted with a load and save Json file options, along with a question list they can shuffle through. The teacher can then import a Json file which has been serialized with our specific format, in which the questions will be loaded. The teacher after loading the file has 2 options; they can either shuffle through the questions they have selected to see if they are satisfied with the import, or they can save the Json file which will then be used in the game.

4.4.4 Teacher - Student Review GUI

After a student has completed a run, the teachers can review the students' statistics on the student review page. Here they can see specific details about the run like the score, score boost difficulty, the percentage of questions and so on. There is also an improvement percentage visible here that allows the teacher to see how much they have improved from the last run. Each run the student has made can be shuffled through using the "next" and "back" buttons. On each of the run's page there is also a button which opens the all the students have managed to answer. Along with the questions is also a section which shows whether the students got the question correctly. As you shuffle through the runs, the allocated questions would change accordingly.

4.5 Third Party Content

This project includes some third-party content which allows it to meet the requirements and specifications needs so the project can run effectively.

4.5.1 Unity Game Assets

For the main character in my game, I utilized a Unity game asset sourced from the Unity Asset Store. This asset provided a pre-designed character model with animations, allowing for easy integration into the game environment. This character model with appropriate animations, allowed for an easy integration based on its compatibility with the game's visual style and thematic elements. Importing the ready-made assets from the Unity Asset Store only saved development time but ensured high-quality character design and animation.

Other assets in the game, including the tiles and obstacles, were made by me using an external software called Asset Forge and was imported into the game, as the Unity Asset Store lacked the required assets I was looking for.

4.5.2 Standalone File Browser

For file browsing purposes, I have integrated a standalone file browser component sourced from a third-party provider. This file browser module enabled teachers to navigate through files on their system and is extremely crucial for loading up the required Json file needed for the use of the game. The decision to incorporate a standalone file browser was driven due to its robust features, its familiar UI, and its ability to support the necessary file types needed for the project. By using the file browser, I was able to enhance the game's functionality while maintaining a streamlined development process. (8)

5 Implementation & Testing

In this section, we go through the development and implementation of the project, building upon the design's plans outlined in Chapter 3. We may have to explore any deviations from the original design, specifications, and requirements. The section begins by discussing my development approach and its functionalities, followed by an examination of implementation issues, and resulting changes to the system and specification. Finally, we conclude with the testing results after the implementation features were fully implemented.

5.1 Development Approach

In this project I have adopted an iterative development approach, allowing for flexibility and continuous improvement. During the development phase, the focus was mainly on small, manageable units called work packages. Each work package represented a specific set of system features or functionality. As I progressed through the project, I completed these work packages in groups, which correlates to the requirements that can be found in **Table 3.2.1** and **Table 3.2.2 Requirement Tables** of Chapter 3. Keep in mind that due to time, some of the low priority requirements could not be fulfilled. The order in which they were developed are shown below along with the requirement codes along with it in brackets.

1. A basic graphical user interface so both students and teachers can interact with a display (N6)
2. Framework and development of the endless runner game (G1, G2, G4)
3. Question upload feature that will allow teachers to upload and save questions (N1, N2, N3)
4. Question answer feature that will be used in-game for students to answer (G5, G6)
5. To be able to reset the same after a set number of lives has been lost (G7)
6. Data tracking and storing of information regarding game attributes, like score and questions (G3, G8, G12)
7. Teacher run tracking analysis which lets the teachers view the run of the students (N4, N5)

8. Setting features that provides help to the student and allows users to reset question location and runs (N7, N9)

The work packages above show the order of completion where tasks 1-7 are seen as high priority and task 8 has been seen as a low priority. One of the main reasons they were done in the order above was mainly due to the code being reliant of the information of the tasks above it. For example, doing step 6 without the need of step 2 would nearly be impossible as without the game being functional, half of the other parts of the code would be rendered useless. **Fig 4.1 Use Cases, Fig 4.2 System Architecture** and **Fig 4.3 Data Flow** were all used as part of the implementation process for the development of the system.

5.2 Application Functionalities

As the game is run using unity there are several scripts that I would have to use to connect the objects with the code. These methods that are found in these scripts are used to control how each of the attribute moves. You can find the scripts below along with a description of how they function:

1. **GameFlow.cs**: This script manages the flow of the game, including spawning obstacles, handling player interactions, updating statistics and displaying certain messages. It is designed for the game environment which allows the player to navigate through obstacles and answer questions. Upon initialization in the 'Start' method, the script sets up parameters for spawning tiles and starts coroutines for spawning tiles dynamically as the player progresses through the game. Additionally, it sets up another coroutine to destroy premade tiles after a certain delay. In the 'Update' method, the script continuously updates the player's position and score based on their movement and performance. It also checks if the player is close to a gate, indicating a point where a question should be triggered. The CheckIfPlayerAtGate method is responsible for detecting if the player is near a gate. If so, it triggers a question to be presented to the player by interacting with the QuestionHandler script. It also ensures that questions are not continuously triggered by using a flag to control question changes. The QuestionChecker method handles the player's response to a

question. It increments the statistics based on whether the answer was correct or incorrect and displays a pop-up message accordingly. The script also includes coroutines for resetting flags after a dela and destroying game objects that move out of the player's view, ensuring efficient memory management and game performance. Finally, the DifficultyChanger method changes the game's difficulty dynamically based on the player's performance in answering questions, thereby providing a more tailored and engaging experience for the player.

2. **QuestionHandler.cs**: This script serves as a crucial component in managing questions for the game. Upon initialization, it loads questions from a JSON file, the path of which is saved and retrieved from another JSON file. This method ensures dynamic question loading and flexibility in updating questions externally. The script includes methods to display the current question and its options on the UI. It checks if the questions list is properly initialized and displays the question text along with its options. Additionally, the script determines the index of the correct answer option for the current question, facilitating the evaluation of player responses. Error handling is implemented throughout the loading process to address potential issues such as missing files or corrupted data. Moreover, the script allows for the cycling through questions by incrementing the current question index, ensuring a continuous loop of questions during gameplay.

3. **PlayerRun.cs:** This script governs the behaviour of the player character within the game environment. Upon initialization in the Start method, it sets up initial parameters such as player position and life count and displays this information on the UI. Additionally, it triggers the player's movement forward at the beginning of the game, ensuring immediate engagement for the player. In the Update method, the script continuously checks for player input to move the player character sideways using the left and right arrow keys. It restricts sideways movement based on the player's current position to prevent movement beyond the game boundaries. Furthermore, the script allows the player to jump by pressing the spacebar, initiating a vertical movement with a return to the group after a short delay. The Update method also monitors the player's lives run out. This reset includes restoring the player's statistics, positioning, and

transitioning to the menu scene. The script contains methods for obtaining the player's position, reducing the player's life count upon collision with obstacles and resetting the game state. Also, it includes coroutines to handle movement animations smoothly, providing visual feedback to the player during actions such as sideways movement and jumping.

4. **PlayerStatistics.cs:** This script manages and track various statistics related to the player's performance throughout the game. Upon initialization in the Start method, it sets up initial values for statistics such as the number of questions answered correctly, total questions asked, question difficulty level, player score count and a list to track questions. This script includes methods to retrieve statistics such as the player's score, total questions asked, and the number of questions answered correctly. Additionally, it provides functionality to update the score difficulty of questions, update the list of questions asked, increment the player's score, and increment the counters for correct answers and questions asked. One of the key functionalities of the script is the UpdateRunToJson method which is responsible for updating the game's data and serializing it into JSON format. It creates a new GameData object containing the current statistics, adds it to a list of previous game data, and serializes the list into JSON format. The JSON data is then written to a file for storage. It also consists of a ResetStats method which is responsible for resetting all the statistics to their initial value making it ready for the next game. This script also consists of a simple game data structure called GameData which is used to store game-related statistics such as score, questions asked, questions answered correctly, score difficulty level and a list of questions. It serves as a container for organizing and storing the player's progress data in a structured manner.

5. **TeacherReport.cs:** This script is responsible for generating a report displaying various statistics from the game runs. Upon initialization in the Start method, it reads game data a JSON file and deserializes it into a list of GameData objects. It then displays the statistics of the current game run using the DisplayCurrentRun method. The DisplayCurrentRun method updates the UI elements with information from the current game which includes the following: run number, number of questions asked, number

of questions answered correctly, score question difficulty, the score, improvement percentage, and percentage of correct answers. It also parses the list of questions asked during the run and formats them for display. The ParseQuestionList method parses the list of questions from the question list of the game data into a readable format, displaying each question with its correctness (correct or incorrect). The CalculateImprovement and CalculatePercentage are both used to calculate the improvement from which the student made the previous runs and the percentage of questions they answered correctly, respectively. The MoveToNextRun and MoveToBackRun methods enable navigation between different game runs, updating the current index accordingly and displaying the statistics for the corresponding run.

6. **JsonFileLoader.cs:** This script is responsible for loading and displaying JSON files containing the questions for a quiz game. It features functionalities to load and save JSON files, display questions as well as their options and navigate through questions. Upon initialization, the Start method invokes the resetQuestion method which is used to load a base question file already made in the game, from a JSON file. The LoadJsonFile method allows users to select a JSON file containing questions, which are then deserialized and validated. If the questions are valid, they are updates and displayed. The SaveJsonFile method saves the current questions to a JSON file. The LoadQuestionsFromJson method is used to read the location of the JSON file containing questions from another JSON file. It then loads the questions from the specified JSON file and adds to the questions list. The DisplayCurrentQuestion method displays the current question and its options on the UI. Navigation between questions after it has been loaded are down using the MoveToBackQuestion and MoveToNextQuestion fcuntiosn, which increment and decrement the current question index respectively. The SetOptionText method sets the text of each of the options and is used to highlight the correct answer to green.

7. **BoulderMovement.cs and BoulderCollisionHandler.cs:** Both these scripts handle the movement and collision of the boulder as the boulder is the only obstacle in the game that moves towards the player within the game environment. The BoulderMovement script controls the boulder’s movement, utilizing a Rigidbody

component and by adjusting its velocity moves it towards the players direction in the z axis. BoulderCollisionHandler handles the collisions between obstacles that it touches and as it moves down the lane. Upon collison, identified through the OnTriggerMethod, the script destroys the game object that it touched, providing the user with a realistic collision behaviour and obstacle destruction.

8. **SettingsMenu.cs:** This script is used for the configuration options accessibility to the teachers, enabling them to reset certain game parameters. It provides functionality to reset the runs file, effectively clearing all previous game data stored in the runs.json file. This action involved initializing an empty list of GameData objects, serializing it into JSON format, and overwriting the existing gile with the updated data. Similarly, the script facilitates the reset of the question location file, location.json, by updating its content to point to a base questions file named question.json in the Resources folder. This also involves creating a JSON object specifying the new path, serializing it, and writing it back to the original file. Through these methods, teachers can manage and customize their game experience according to their preferences.

5.3 Implementation Issues

Throughout the software development process, various challenges occurred, which involved my attention to resolve them. As issues arise, the development must address them, which may result in the modifications of the system's design. When confronted with issues during project development, the primary remained gameplay, ensuring that solutions were implemented using gameplay elements and enabling balance with speed while doing so. Thie approach ensure that problem-solving efforts remained aligned with the core gameplay experience and avoiding the risk of implementing insecure or non-essential feature fixes.

5.3.1 Infinite Question Generation

One of the major features of an endless runner question was that there needs to be an infinite number of questions. This ensures the player has access to a diverse pool of questions without exhausting system resources or causing performance issues. My original plan to overcome this was to acquire questions from ChatGPT using OpenAI features. This soon became ineffective as to use OpenAI, you had to pay a certain amount for each word generation which didn't seem to be worth it. As a result, to overcome this challenge, I had to use a traditional method of reusing the questions that the teacher has originally set. This method seemed effective to the cause of the computer-based assessment as it lets the teacher see whether the student is making improvements throughout the course of the game.

5.3.2 Question Difficulty Changer

Providing players with the ability to adjust the difficulty of questions adds depth to the gameplay experience. However, dynamically changing questions difficulty levels while maintaining balance was challenging. This tied to the issue mentioned above where the access of new question generation according to the questions answered was simply not cost effective. As a result, instead of changing the difficulty of questions, the number of points given due to the questions answered was implemented. This means as the percentage of questions answered correctly directly correlated with the number of points they received during the run. This ties well into the CBA aspect of the game as it meant the student will be motivated to work more on the getting more questions correct as they will be rewarded with more points.

5.3.3 Gameplay Speed Adjustments

One major issue I faced while developing the game was maintaining the game speed with the player movement There were points in the game where the game was not able to keep up with the rate at which questions were spawning and how fast the player was progressing forward. As a result, I had to fine tune values which allowed the game to keep balance and work smoothly. All question changes as well as collisions with objects have been well handled.

5.3.4 Object Generation Issues

It must be ensured that there is a consistent and appropriate generation of objects (e.g. spikes, statues, boulders) within the game environment. Utilizing the GameFlow script which handles object generation and destruction was crucial for this process. It helped with making sure objects were deleted right after the player has passed them. However, there were lots of instances where they would either deleted far too early or the wrong obstacles all together. To fix this the DestroyItem method is used to destroy these said object so the game can run smoothly and control the storage and game performance.

5.4 GUI Implementation

The GUI (Graphical User Interface) implementation was a crucial aspect of the project, ensuring a user-friendly and visually appealing interface for interacting with the game. This section outlines the key components and strategies employed in the GUI implementation:

1. Design Considerations: The GUI design was meticulously crafted to align with the overall aesthetic and theme of the game which is based on an ancient Egypt environment. Considerations were made for readability, usability, and consistency across different screens and devices.

2. Unity UI System: The Unity UI system was used to create and manage the graphical elements of the interface. Various UI components such as TextMeshProUGUI, Buttons, Panels and Images were utilized to construct interactive and informative GUI elements. UI elements were organized into canvases and hierarchies to facilitate efficient rendering and interaction handling.

3. Dynamic Content Generation: The GUI accommodated dynamic content generation, such as displaying questions, options, scores, and game statistics. Content was dynamically updated based on gameplay events, user input and system states to provide real-time feedback and information.

4. User Input Handling: User input was managed through interactive UI elements such as buttons, sliders, and input fields. Event handling mechanisms were also implemented to respond to user interactions which triggers appropriate game actions or UI updates.

5. Responsive Layout: The GUI layout was designed to be responsive, adapting to different screen sizes and resolutions. Techniques such as anchoring, layout groups and flexible UI components were employed to ensure a consistent presentation across various devices and aspect ratios.

6. Integration with Game Mechanics: The GUI seamlessly integrated with the underlying game mechanics provide essential feedback, instructions, and controls to the player. Elements such as such as score counters, life counts, questions and options are incorporated into the GUI to enhance gameplay clarity and immersion.

7. Testing and Iteration: Extensive testing and iteration were conducted to refine the GUI design, functionality, and reliability. User feedback, usability testing and playtesting sessions were utilized to identify usability issues, improve navigation flow, and optimize visual presentation.

5.4.1 Student GUI

The following figures shows the GUI for what the game should look like after the player has started.

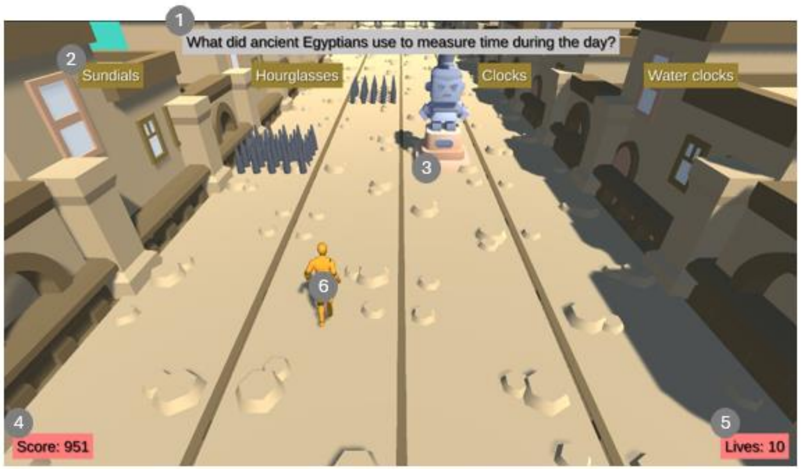

1
What did ancient Egyptians use to measure time during the day?
2
Sundials
Hourglasses
Clocks
Water clocks
3
6
4
Score: 951
5
Lives: 10

Figure 5.4.1: GUI Gameplay Design 1

1. Question Display: When the student loads into the game, a question is displayed at the top of the screen which the student can read. They have around 10 seconds to choose an answer to the question.
2. Options Display: When the question is loaded, the options are also loaded below it. There are 4 options which align with the 4 lanes. The lane the student chooses determines the option they want to stick with.
3. Obstacles: These are what the students want to avoid while playing the game, colliding with the player would result in a decrease in a life point the player has.
4. Score: This text displays the score the player has accumulated throughout the run. As the player moves forward, the score automatically increases.
5. Lives: This text displays the number of lives the player has. The default number of lives is 10, which decreases through certain events in the game.
6. Player: This is the character the player must control; they have an option to switch between 1 of 4 lanes using the arrow keys. They can also jump, but only over the spikes. Miss timing a jump or colliding with an obstacle will result in the obstacle being destroyed and a life point being lost.

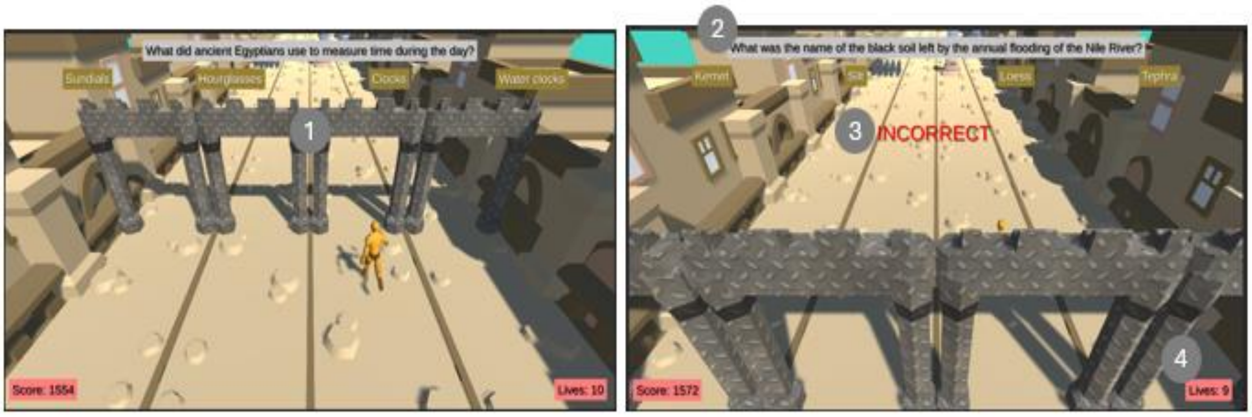

What did ancient Egyptians use to measure time during the day?
Sundials
Hourglasses
Clocks
Water clocks
1
Score: 1554
Lives: 10
2
What was the name of the black soil left by the annual flooding of the Nile River?
Kemet
Silt
Loess
Tephra
3
INCORRECT
4
Score: 1572
Lives: 9

Figure 5.4.1: GUI Gameplay Design 2

1. Gate: The gate is the part of the game where the student must lock in their answer. As they enter through the gate, the game decides whether they got the question right or wrong. This event is used to use to update several stats.
2. Question Change: Once the student has passed the gate, the question is then changed to a different question, regardless of if the student got it right or not. It shuffles into the

next question in the list of questions the teachers have provided. The student has then till the next gate spawn to choose an appropriate answer.

3. Correctness Display: Once the student has a text GUI will appear on screen which shows whether the student got the right or wrong answer. A “CORRECT” message will appear if they manage to get the question right and an “INCORRECT” message will pop up if they manage to get the question wrong. This message will stay on the screen for approximately 3 seconds.
4. Life Change: Depending on the event that has occurred, the life count changes. In this instance a life was lost due to the question being answered incorrectly.

5.4.2 Teacher GUI

The following figures shows the GUI for what the application looks like for the teacher. This includes both the question loading and saving features along with the run analysis of each of the student.

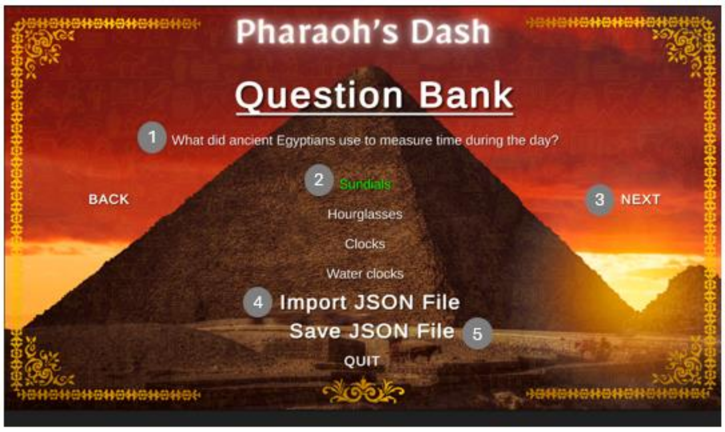
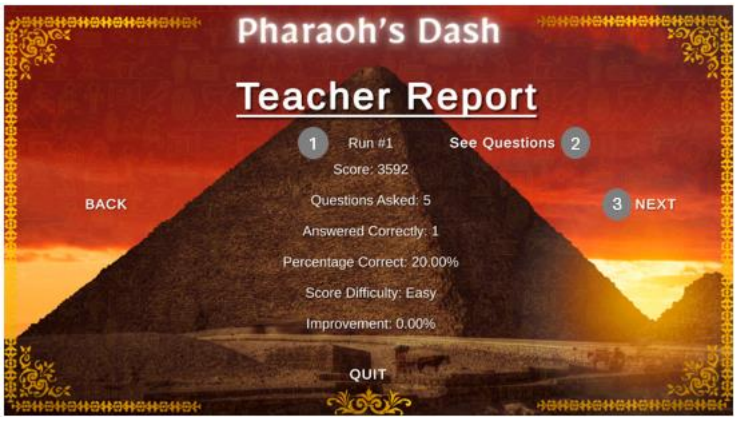


Figure 5.4.1: GUI Teacher Design 1

1. Question: When initially loading up the question bank page, which lets the teacher load the questions in, there is a default set of questions that are already loaded up. When the teacher uploads a new Json file, this question will change accordingly to what the Json file consists of.
2. Options: Similar to the question part this part of the GUI shows the teacher which options the student can choose between. I have also included so that the correct

answer of the options is highlighted in green. This lets the teacher know which option is the correct just in case they are checking all the questions faster.

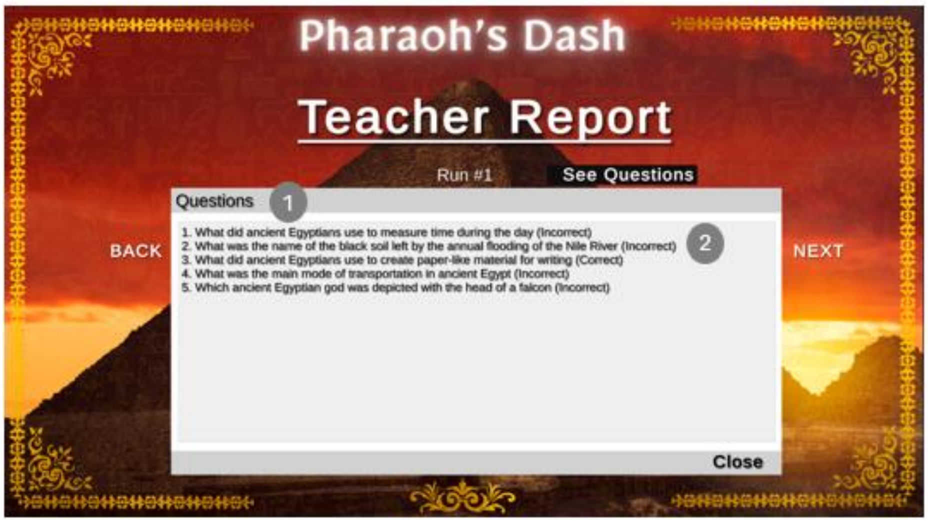
3. Question Shuffling: This feature simply lets the teacher shuffle through the set of questions and options they have.
4. Importing JSON file: This button lets the teacher to import a Json file. Upon clicking this button, it opens a file directory they can search through which lets them select the correct file they require. If the question set is valid, all the question and options will change accordingly, if not then the default questions will remain present.
5. Saving JSON file: This button allows the teacher to save the Json file they have loaded. This button is only functional if a new set of questions were loaded in. Nothing will happen if a question has not been loaded in or an invalid question set has been loaded in.



Score: 3592
Questions Asked: 5
Answered Correctly: 1
Percentage Correct: 20.00%
Score Difficulty: Easy
Improvement: 0.00%

Figure 5.4.1: GUI Teacher Design 2

1. Run Analysis: When loading the report, all the following statistics are loaded up: Run number, score count, questions asked, questions answered correctly, percentage of questions answered correctly, score difficulty and rate of improvement. If there hasn't been a run or the teacher decides to reset the runs, then the runs will remain empty.
2. See Questions Options: When clicking this button, a pop up will appear, which will show the all the questions the students have answered in that respective run.
3. Run Shuffling: This button along with the back button on the other side of the screen, are used to shuffle through all the runs the student have made.



1. What did ancient Egyptians use to measure time during the day (Incorrect)
2. What was the name of the black soil left by the annual flooding of the Nile River (Incorrect)
3. What did ancient Egyptians use to create paper-like material for writing (Correct)
4. What was the main mode of transportation in ancient Egypt (Incorrect)
5. Which ancient Egyptian god was depicted with the head of a falcon (Incorrect)

Figure 5.4.1: GUI Teacher Design 3

1. Question Analysis Page: This is pop up that is displayed when the teacher presses the See Questions button. It consists of a list of all the questions that the student has answered during that respective run.
2. Correctness of Questions: Along with the list of all questions, there is a bracket and within that bracket is whether the student has got that question right or wrong. This is intended to help the teacher see which questions the student right and wrong.

5.5 JSON File Implementation

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is commonly used for storing data in my game design. It provides a structured way to represent data in a human-readable format, making it easy to parse and manipulate programmatically. It also a format lets teachers make questions in a quick and easy way. The JSON data structures provided consists of the format shown below.

1. Question Bank Format: This format represents a collection of questions and their corresponding options and correct answers. Each question is represented as an object containing the question text, an array of options and the correct answer. For example:

```
{
        "question": "What did ancient Egyptians use to measure time during the day?",
        "options": ["Sundials", "Hourglasses", "Clocks", "Water clocks"],
        "correctAnswer": "Sundials"
},
```

This structure format allows for easy retrieval and manipulation of questions and answers within the game. It is also the same format the teachers must load up into the game. If the Json file doesn't match the format, it will be rendered useless. It is recommended that the teacher uploads more than 20 questions so the student can enjoy playing with a variety of questions.

2. Player Statistics Format: This format records player performance statistics, such as scores, number of questions asked, number of questions answered correctly and question. Each entry in the Json array represents a gaming session, detailing the player's performance metrics and the list of questions asked during that session, along with their correctness. For example:

```
{
        "score": 3592.0,
        "questionsAsked": 5,
        "questionsCorrect": 1,
        "questionDifficulty": "Easy",
        "questionList": ";What did ancient Egyptians use to measure time during the day?Incorrect;What was the
        name of the black soil left by the annual flooding of the Nile River?Incorrect;What did ancient Egyptians
        use to create paper-like material for writing?Correct;What was the main mode of transportation in
        ancient Egypt?Incorrect;Which ancient Egyptian god was depicted with the head of a falcon?Incorrect"
},
```

This structured format enables tracking and analysis of player progress and performance over time, facilitating improvements to the gaming experience.

3. Location Format: This Json is only used as a method to store the location of the Json file of the questions that has been uploaded by the teachers. Otherwise, a default location will be stored here instead which points to the default question file that can be found in the resources folder.

5.6 Testing

Testing played a pivotal role in ensuring the functionality, reliability, and user experience of our game system. This process aimed to validate that the system met the requirements that were made in the chapter 3, Requirements and Specifications, of the report. Different forms of testing were conducted to assess various aspects of the system's performance.

5.6.1 Non-functional Testing

Non-functional testing focused on evaluating the quality attributes of the game to ensure its effectiveness and usability. Key aspects of this type of testing include:

1. Efficiency: The efficiency of the game was assessed in terms of its performance and resource utilization. Testing included measuring factors such as loading times, frame rates and memory usage to ensure that the game ran smoothly and under varying conditions. Optimization techniques, such as code refactoring and asset compression and deletion, were applied based on testing results to enhance the game's performance and minimize resource consumption.
2. Usability: Usability testing focused on evaluating the game's user interface (UI) design, navigation flow and overall user experience. Testing this means interacting with the game to assess the intuitiveness of controls, clarity of instructions and accessibility of features. Feedback from usability testing helped identify areas for improvement such as redesigning menu layouts, adding tutorial prompts in settings or adjusting controls to enhance player satisfaction.
3. Reliability: Reliability testing aimed to verify that the game operated consistently and predictably under normal and stress conditions. Test scenarios included simulating heavy user loads and prolonged gameplay sessions. Any instances of crashes, freezes or unexpected were doing and majority of them was sorted out during debugging.

5.6.2 Requirement-Based Testing

To make sure that the game meets its functional and non-functional requirements as outlined in Tables 3.2.1 and 3.2.2, comprehensive testing procedures were conducted. During this phase it allowed me to test out the functionality, reliability, usability, and performance of the game across various scenarios.

The following allowed me to test the metrics of each functional requirement, which makes for a proper implementation and integration of the game:

1. G1 – Allowing player movement and control was tested using the use of left and right arrow keys to verify the responsiveness and accuracy of movement controls. Situations which involved players moving out of boundaries were checked in the process.
2. G2 – Generating obstacles dynamically was testing in the obstacle generation script to ensure that the objects spawn during regular intervals of the game. It was also used to verify that there was a variety and randomness of obstacles.
3. G3 – To test the tracking of player's score based on distance travelled we had to make sure there was consistency in score rate increase during different scenarios of the game. Which was used to verify if the scoring system was accurately designed to increase score efficiency.
4. G4 – To test detect collisions between player and obstacles, I had to use unity collision boxes and scripts to detect collisions between the player and obstacle game objects. Test collisions detection under different conditions, like when players do a mid-lane switch for example.
5. G5 – Testing for display questions for players to answer I must integrate text UI for displaying questions and simply had to verify if the questions are displayed correctly and clearly visible for the user to read.
6. G6 – To test out the feedback on player's question responses I had to implement a feedback UI which displays whether the player's response was correct or not. The test feedback system was used to check for the accuracy of the result and clarity.
7. G7 – To test for ending the game after certain events was tested using various game ending scenarios. Two main scenarios being when players collided with objects or when players got a question wrong. This was used to ensure the game ended appropriately.

8. G8 – Integrating a scoring system that reflected on the questions answered correctly was difficult to test. However, it was done using different difficult score bonuses individually to ensure balance and progression.
9. G12 – To test for the saving of data for each of the run, I had to implement have multiple data savings and loading functionality to ensure persistence and accuracy.
10. N1 – To test for teachers uploading questions I had to verify that questions were uploaded correctly and can be accessed by the game. There were also tests which involved uploading different file types and files with incorrect formats.
11. N2 – To test for the saving of uploaded questions was relatively easy as I had to make sure if data was saved securely and whether it could be retrieved easily.
12. N3 – To test for the navigation through the uploaded questions, I had to verify that navigation of the controls was intuitive, and all questions were accessed easily. Also making sure if it could handle huge amounts of input.
13. N4 – To test for player statistics, I had to implement backend tracking of player statistics and calculations, and testing the UI was relevant for teachers to view player statistics and ensure accuracy of data presentation.
14. N5 – To test the display of the questions the students have answered, showing whether its they got it right or wrong, I had to initially test the UI functionality so they can see a full list of all the questions. I also had tests to verify that correct and incorrect answer are displayed.
15. N7 – To test for resetting runs and questions, I had to implement a functionality that allowed teachers to reset the files no matter the contents in it. I also had to test that this did not result in relevant data to the game was accidentally removed, along with it affecting other game elements.

5.6.4 Bugs

During testing phase, it came to my attention that there were two notable bugs in the game.

Firstly, there were incident where the lives would randomly fluctuate up and down. At random scenarios during the game there were incidents the player character would collide with an obstacle which would cause the life count to go up. I believe this was an issue caused by the UI display and overlapping script calls which did resulted in an incorrect update of the UI for the life text. Although the display would fluctuate the number of lives in game would still go down correctly.

Secondly, there were issues when spawning obstacles as multiple obstacles were spawning at the same at the same spot, which creates unintended challenges for the player. This error was due to a bug that was caused in the GameFlow script of the code. A result of this bug meant if students collided with this a double spawned object, then it would take out two lives from the tally instead of one.

6 Evaluation

6.1 Application Evaluation

The gamified computer-based assessment (CBA) represents an attempt to revolutionize the educational experience a student has through gamification principles and attributes. This project evaluation delves into the various facets of development, testing, and implementation, providing an overall assessment of the project's overall success and potential areas for improvement.

The project's primary objective revolved around creating an engaging educational platform that seamlessly integrated various gameplay elements with curriculum-based assessments. Through research, planning and execution, I have managed to create the objectives into a tangible game and that have several game functionalities of it within the application. However, a critical evaluation is required to see the extend of which the application meets the educational needs and goals outlined in the project scope.

Technical implementation of the game plays a pivotal role in its overall development. A thorough evaluation of the development tools including unity, programming languages and frameworks used was essential to maintain the efficiency and scalability of the codebase.

The user experience is a fundamental aspect of any software application, particularly ones that are designed for educational purposes. Evaluating the application's interface design, navigation flow, responsiveness and accessibility allowed a better usability and appeal to both teachers and students.

At the core of the gamified CBA application lies its educational value and ability to facilitate learning outcomes. Evaluating the alignment between game mechanics and educational objectives, as well as the relevance and accuracy of the content that is being presented is crucial. There also is an importance of assessing student engagement, motivation and knowledge retention through studies and qualitative feedback which offered valuable feedback for when making the project.

6.2 Project Evaluation

The approach we took during development allowed us to be flexible and meet requirements effectively. Careful planning was essential for creating a well-developed system, and the design stage played a pivotal role in achieving that.

The background research conducted during the start of the project came in critical for understanding the project's understanding the project's challenges and finding solutions. If more research was done, it could have reduced the issues we faced during implementation. Dealing with these challenges took extra time, along with the unfamiliarity of using unity and the programming language well. For instance, figuring out how to control assets on the scene in the application required additional research and significantly impacted development time.

During the design phase, we created diagrams and graphical designs for the system's user interface. This was helpful when it came to implementation as it meant the developers did not have to imagine how the interface structure would look like, as they could easily follow the designs shown. However, for GUI implementation it could have been better if there was a design for button layout and overall display structure so as the developer, we would know what everything would look like. The data flow diagrams, and use-case diagrams weren't explicitly clear on how the user should structure the code, as there was no activity or interface linking them and was left to the developer to decide on the approach.

During the implementation phase we managed to use short development sprints which ensured we completed focus on one work package at a time. This ensured thorough development, implementation and testing before moving onto the next part of the project. This approach allowed the developer to have refreshing tasks by introducing them to new work packages upon completing the previous ones. The system was built feature by feature through the addition of work packages until all medium and high-level requirements were met. Despite aiming to finish the project two weeks before the submission deadline, unforeseen bugs and circumstances had caused to extend the timeline, ultimately meeting the deadline. The estimated time frames to complete the tasks were also taken in account when it came to meeting the deadline.

The testing process allowed issues to be fixed faster than expected and ensured that the application met all the requirements that were set before it. It was mainly used to detect errors early in the code, highlight any issues and find a resolution in code and work packages. This proactive approach of testing managed to minimize errors in the integrated system. While there were overarching bugs and issues in the code, it did help with significantly reducing the likelihood of errors.

The final product of the application resulted in a well set up gamified computer-based assessment that both teachers and students can use to test out their capabilities. Easy and engaging for the student and convenient for the teacher to mark and get results.

6.3 Project Limitations

While the project had considerable potential and promise, due to time constraints and other factors, there were limitations that halted the project from reaching a whole new level entirely. Some limitations are as follows.

One of the primary limitations of the application might have been the limited variety of questions or coverage it had for the students. Due to the lack of AI and other resources available int eh development phase, the application contains a finite number of questions, leading to a lack of diversity and potential repetition. This limitation could impact the overall learning experience a student might have while playing the game.

Another limitation of the application may be the challenge in maintaining user engagement and motivation over time. While the gamification is used to enhance user engagement, there is always a risk where students are not interested or might be tired due to repetitive gameplay, particularly if gameplay mechanics are repetitive and predictable.

Technical constraints, such as platform compatibility or resource limitations, could also lead to significant challenges for the application. There may be issues that might come about due to compatibility issues with certain operating systems or hardware configurations, which limits the reach of the application. As the project was tested in a Windows operating system, it is difficult to tell how well it will perform under other OS systems.

7 Professional Issues

7.1 Legal Issues

Due to copyright infringement, there must be an assurance that all assets used in the endless runner game, which includes graphics must be original or properly licensed to avoid any potential copyright violations. There is also a concern when it comes to respecting the intellectual property rights of other games and game developers to avoid potential legal disputes that may come along due to the similarities in game mechanics or design elements.

7.2 Ethical Concerns

As the project is a game, we must acknowledge the potential for addiction or dependency on the game. There may be incidents where students are unable to stop playing the game and might need to incorporate features such as limiting playtime or providing periodic breaks. When playing the game, we must also try and make a game environment that is inclusive and ensures that all players feel welcomed and respected when playing the game.

7.3 Social Implications

Some social issues that may rise when playing the game is risk of excessive gameplay leading to social isolation and loneliness, particularly with the younger players that we have targeted, and we must try our best to encourage balanced gaming habits that prioritize real-world social interactions. Also, a toxic environment in the gaming field can occur, due to other players scoring more than others can cause the game to not have a positive and supportive gaming environment which is needed for all types of games.

7.4 Professional Issues

We must adhere to rigorous testing and quality assurance protocols to identify and address bugs, glitches, or other performance issues, so we can ensure a polished and enjoyable gaming experience for players. In the future the potential of having ongoing development and improvement of the game might be necessary with user feedback and trends in the industry, will demonstrate a dedication to the development of the game.

8 Conclusion

In conclusion, the gamified computer-based assessment marks an advancement in educational technology, combining the curriculum-based assessments to enhance student engagement and learning outcomes with gamification principles. While this project has demonstrated success in implementing game mechanics and ensuring the robustness of technical aspects, there are still areas for improvement and have many bugs that could be fixed in the future, including diversification of question. Moving forward, future user feedback and embracing new technologies will allow the application to evolve, making it an even more impactful tool for both teachers and students. Ultimately, the gamified CBA application represents a promising step towards the future of interactive learning experiences, showcasing the transformative potential of technology in education.

9 References

1. TestPartnership. [Online] 2019-gamification-literature-review.pdf (testpartnership.com)

2. Arxiv. [Online] [2207.07369] A Systematic Literature Review of Game-based Assessment Studies: Trends and Challenges (arxiv.org)

3. ResearchGate. [Online] (PDF) A Systematic Literature Review of Game-Based Assessment Studies: Trends and Challenges (researchgate.net)

4. Springer. [Online] EFL learners' motivation in a gamified formative assessment: The case of Quizizz | Education and Information Technologies (springer.com)

5. Bera Journal. [Online] Measuring learning in digital games: Applying a game-based assessment framework - Udeozor - 2024 - British Journal of Educational Technology - Wiley Online Library

6. Orcunnisli. [Online] Endless Runners: Understanding Previous User Experiences (orcunnisli.com)

7. GamingScan. [Online] Best Endless Runner Games 2024 [Ultimate List] - GamingScan

8. Standalone File Brower [Online] https://github.com/gkngkc/UnityStandaloneFileBrowser/blob/master/README.md