

**Capstone Project – Cover Sheet**

ASSIGNMENT

**Instruction:**

* Marks will be allocated based on the quality of presentation, innovation and the depth of your approach.
* Referencing Code: When incorporating code or ideas derived from external sources (such as books, journals, the internet, or discussion forums), proper citation and referencing according to APA conventions within the source code are mandatory. Failure to reference code appropriately will be treated as plagiarism.
* Please ensure that you complete this cover sheet and **affix it to the first page of your project.**
* This project is intended for groups comprising 5 to 6 students.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student declaration:** | | | |
| *I declare that:* | 1. *We understand what is meant by plagiarism* 2. *The implication of plagiarism has been explained to us by our lecturer* 3. *This project is all our work and we have acknowledged any use of the published or unpublished works of other people.* | | |
| Group Leader’s Signature:  *Nic* | | Date: 12/5/2025 | |
| **GROUP NUMBER: Group 21**  **PROJECT TITLE: ProSnap – Intro to Visual Interactive Programming**  **INTAKE: UCDF2307ICT(SE)** | | |  |
| **Name** | | **Signature** | |
| **Anderson Tey Le Tian** | | *Anderson* | |
| **Chan Guan Ming** | | *Chan* | |
| **Chek Ka Nic** | | *Nic* | |
| **Chong Wei Jie** | | *Chong* | |
| **Mah Zheng Yang** | | *Mah* | |

Table of Contents

[Workload Matrix 3](#_Toc197984104)

[Acknowledgement 4](#_Toc197984105)

[Abstract 5](#_Toc197984106)

[1.0 Introduction 6](#_Toc197984107)

[1.1 Project Background 6](#_Toc197984108)

[1.2 Problem context 7](#_Toc197984109)

[1.3 Proposed solution 8](#_Toc197984110)

[1.4 Project Objectives 9](#_Toc197984111)

[1.5 Project Scope 9](#_Toc197984112)

[2.0 Project Plan 10](#_Toc197984113)

[2.1 System Development Methodology 10](#_Toc197984114)

[2.2 Gantt Chart 11](#_Toc197984115)

[3.0 System Hierarchy Chart 11](#_Toc197984116)

[4.0 Context Diagram 12](#_Toc197984117)

[5.0 Data Flow Diagrams Level 0 13](#_Toc197984118)

[6.0 Entity Relationship Diagram 14](#_Toc197984119)

[7.0 Individual Component 15](#_Toc197984120)

[7.1 Chek Ka Nic TP074894 15](#_Toc197984121)

[7.1.1 Data Flow Diagram Level 1 15](#_Toc197984122)

[7.1.2 Data Dictionary 15](#_Toc197984123)

[7.2 Anderson Tey Le Tian TP076394 18](#_Toc197984124)

[7.2.1 Data Flow Diagram Level 1 18](#_Toc197984125)

[7.2.2 Data Dictionary 18](#_Toc197984126)

[7.3 Chan Guan Ming TP076256 20](#_Toc197984127)

[7.3.1 Data Flow Diagram Level 1 20](#_Toc197984128)

[7.3.2 Data Dictionary 20](#_Toc197984129)

[7.4 Chong Wei Jie TP076546 22](#_Toc197984130)

[7.4.1 Data Flow Diagram Level 1 22](#_Toc197984131)

[7.4.2 Data Dictionary 22](#_Toc197984132)

[7.5 Mah Zheng Yang TP076666 25](#_Toc197984133)

[7.5.1 Data Flow Diagram Level 1 25](#_Toc197984134)

[7.5.2 Data Dictionary 26](#_Toc197984135)

[8.0 Screen Design and Report Design with User Manual 28](#_Toc197984136)

[9.0 Test Plan (Unit Testing and User Acceptance Testing) 33](#_Toc197984137)

[10.0 Significant Source Code with Explanation 34](#_Toc197984138)

[Conclusion 98](#_Toc197984139)

[References 99](#_Toc197984140)

# Workload Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Area** | **ANDERSON TEY LE TIAN (TP076394)** | **CHAN GUAN MING (TP 076256)** | **CHEK KA NIC**  **(TP 074894)** | **CHONG WEI JIE**  **(TP 076546)** | **MAH ZHENG YANG (TP 076666)** |
| **Cover Page** | **10%** | **60%** | **10%** | **10%** | **10%** |
| **Table of Content** | **10%** | **60%** | **10%** | **10%** | **10%** |
| **Acknowledgement** | **20%** | **20%** | **20%** | **20%** | **20%** |
| **Abstract** | **20%** | **20%** | **20%** | **20%** | **20%** |
| **Project Background** | **10%** | **10%** | **45%** | **10%** | **25%** |
| **Problem Context** | **10%** | **25%** | **45%** | **10%** | **10%** |
| **Proposed Solution** | **10%** | **10%** | **60%** | **10%** | **10%** |
| **Project Objectives** | **10%** | **10%** | **10%** | **60%** | **10%** |
| **Project Scope** | **60%** | **10%** | **10%** | **10%** | **10%** |
| **Project Plan** | **10%** | **10%** | **10%** | **60%** | **10%** |
| **System Hierarchy Chart** | **60%** | **10%** | **10%** | **10%** | **10%** |
| **Context Diagram** | **20%** | **20%** | **20%** | **20%** | **20%** |
| **Data Flow Diagram level 0** | **10%** | **10%** | **60%** | **10%** | **10%** |
| **Entity Relationship Diagram** | **30%** | **10%** | **40%** | **10%** | **10%** |
| **Screen Design and Report Design with User Manual** | **10%** | **10%** | **10%** | **60%** | **10%** |
| **Game Development** | **35%** | **10%** | **10%** | **10%** | **35%** |
| **Test Plan (Unit Testing and UAT)** | **20%** | **20%** | **20%** | **20%** | **20%** |
| **Significant Source Code with Explanation** | **10%** | **60%** | **10%** | **10%** | **10%** |
| **Conclusion** | **10%** | **10%** | **10%** | **60%** | **10%** |
| **Digital Signature** | **Anderson** | **Chan** | **Chek** | **Chong** | **Mah** |

# Acknowledgement

I would like to extend my sincere gratitude to all those who supported and guided me throughout the completion of this assignment, which was submitted as part of my studies at Asia Pacific University (APU).

Firstly, I am deeply thankful to my lecturer, Justin Gilbert A/L Alexius Silvester, for his continuous support, constructive feedback, and valuable guidance that greatly contributed to the success of this project.

I would also like to express my appreciation to my friends for their involvement in user acceptance testing. Their thoughtful feedback and suggestions were crucial in identifying improvements and enhancing the overall functionality and user experience of the project.

Lastly, I am grateful to my friends and family for their encouragement and support during this assignment. This experience has significantly enriched my learning and practical skills.

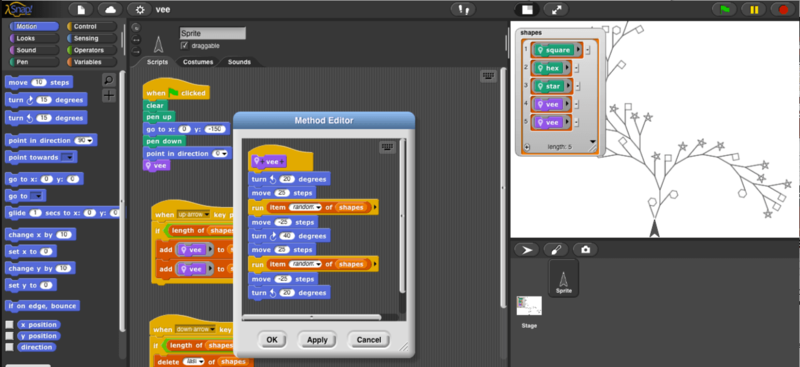
# Abstract

In this era, learning might become a huge burden for students. This is because the competition between people and people becomes truly intense. Hence, most students have to face and cope with study stress. To ease this serious problem, we have developed a gamified e-learning platform targeting university students and lecturers. This platform is Pro Snap. In this platform, lecturers are able to upload questions to a maze game with different levels of difficulty. Then, students can access the platform to play games and answer questions. They would need to answer the correct answers in order to pass the games. The main purpose of this platform is to increase the academic performance of university. We created this platform to help increase the desire of students to do revision. They would not feel as bored as before to do revision on their studies. In addition, it could reduce lecturers’ burden at the same time.

# 1.0 Introduction

## 1.1 Project Background

Snap! is a block-based visual programming language designed to introduce users, particularly young learners for educational purposes, to the fundamentals of computer science and programming concepts. It is developed by Jens Mönig and Brian Harvey at the University of California, Berkeley. Snap! is an extension of Scratch, a popular programming language for children created by the MIT Media Lab. Snap! remains the drag-and-drop interface of Scratch but it also comes with advanced features such as first-class functions, recursion, and custom blocks that makes it suitable for both new learners and more experienced programmers exploring complex programming concepts (Harvey & Mönig, 2010). The browser-based and open-source language allow users to create interactive animations, games, and simulations while learning computational thinking and problem-solving skills. Its design emphasizes on minimal hand-coding and enable users to focus on logic and structure without the need to memorize syntax which usually needed in traditional programming languages (Brian et al., 2017).



*Diagram shows an example of Snap! Visual Programming Interface*

The intention of developing Snap! are due to the need for a more reliable educational tool that could bridge the gap between beginner-friendly environments like Scratch and text-based programming languages like Python or JavaScript that are stricter on syntax. Snap! is widely used in educational settings, from secondary schools to universities, to teach concepts such as algorithms, data structures, and even artificial intelligence (Mönig et al., 2015). Snap! is a valuable resource for new learners to get familiar with computational thinking due to its ease of use and user interface that are very user-friendly. By combining the simplicity of visual programming with the robustness of traditional coding, Snap! has become a significant tool in computer science education.

As for Malaysia, visual programming languages like Snap! are introduced into educational curricular to enhance students' computational thinking and programming basics. Based on a study by Ismail in 2019, an investigation was carried out to explore the capability of primary school pupils in learning and exploring visual programming languages, and it is clear that students as young as seven could engage with visual programming with appropriate guidance from teacher. Additionally, SEGi College Kuala Lumpur collaborated with SAS Institute Inc. to introduce CodeSnaps, a coding system that utilizes a tablet and robot mainly aiming to prepare students for the demands of Industry 4.0 by equipping them with essential coding skills (SEGi University & Colleges, 2019). These initiatives show a growing emphasis on incorporating visual programming tools like Snap! to leverage programming knowledge among Malaysian students.

## 1.2 Problem context

**Limited for Complex Applications**

There are several limitations and challenges that Snap! encounters which might affect its usability. One of the key limitations is its reliance on a visual, block-based interface. Although it is very user-friendly for new learners, but it also can become restrictive for advanced users who was looking for more complex functionalities. In most situation, block-based programming environments often struggle with scalability when it is used to handle large-scale projects, as the visual representation can become cluttered and cause confusion to users (Meerbaum-Salant et al., 2013). Additionally, Snap! lacks support for certain low-level programming features such as advanced debugging tools or direct memory management, which are commonly used for higher level software development (Harvey & Mönig, 2012).

**Difficult Transition to Text-Based Languages**

Another significant challenge is the transition from Snap! to text-based programming languages. While Snap! introduces important computational thinking concepts, students may struggle when shifting to traditional languages such as Python, Java, or C++ due to differences in syntax and problem-solving approaches (Weintrop & Wilensky, 2015). Research shows that block-based programming can cause misunderstood that coding is entirely graphical instead of lines of complicated code, which cause difficulties for learners to get used to text-based environment (Bau et al., 2017).

**Not Industry Standard**

Furthermore, Snap! faces limitations in real-world applications beyond educational contexts. Unlike widely used professional programming languages, Snap! is not commonly employed for commercial software development, limiting its relevance in industry settings (Harvey & Mönig, 2012). While it is an excellent tool for learning computational thinking, it lacks integration with many APIs and external libraries that are necessary for developing complex applications. On top of that, due to the browser-based platform of Snap! , performance issues can arise when running computationally intensive tasks, as it depends on web technologies that may not be as efficient as native applications (Meerbaum-Salant et al., 2013). These limitations suggest that while Snap! is a valuable educational tool, its usage outside of learning environments remains constrained.

## 1.3 Proposed solution

Addressing the limitations of Snap! Visual Programming Language require exploration of specific strategies external integrations. Below are some potential solutions regarding on the limitation that were mentioned above, supported by relevant studies:

**Integration of Snap! with advanced features**

Integrate Snap! with text-based languages to extend its capabilities. By allowing users to incorporate textual code within Snap! projects, more complex functionalities can be achieved. This hybrid approach leverages the simplicity of visual programming while tapping into the power of traditional coding (Harvey, 2020). Furthermore, the author of the same article also voices out that development of an offline version of Snap! can significantly reduce the reliance on web browsers and internet connectivity. This approach ensures that users can access to Snap! anytime anywhere without facing compatibility issues.

**Implementation of text-based coding in Snap!**

Based on a journal article by Brian Harvey in 2020, the difficulty of transitioning Snap! learner to text-based coding language can be overcome by designing an educational pathway that gradually introduce text-based coding concepts within the Snap! environment. Moreover, highlighting Snap!'s advanced features and its potential for teaching complex programming concepts can bridge the gap between educational tools and industry practices (Harvey, 2020).

## 

## 1.4 Project Objectives

* To develop an attractive and fun e-learning platform that can be accessed anywhere easily.
* To enhance undergraduate students’ performance in their studies.
* To improve the effectiveness of learning and teaching.
* To reduce the burden of educators like teachers and lecturers.

## 1.5 Project Scope

|  |  |
| --- | --- |
| Component | Details |
| Project Objective | -To develop an attractive and fun e-learning platform that can be accessed anywhere  -To enhance undergraduate students’ performance  -To improve the effectiveness of learning and teaching.  -To reduce the burden of educators |
| Deliverables | 1.Product  Student  -Must be able to start a game for education  -Must be able to customize the difficulty of game  Lecturer  -Must be able to check students’ result  -Must be able to upload games for e-learning  2.User Manual |
| Milestones | 1. Project Planning (Week 1)  2. Risk and Analysis Mitigation (Week 2)  3. System Development (Week 3)  4. Review and Evaluation (Week 4) |
| Technical Requirements | - Any modern web browser  - A device with a reasonable processor and RAM  - An internet connection |
| Limits and Exclusion | - The e-learning system cannot be run when offline  - There is only web-based version for the system |
| Necessary Changes | - The data security of system should be improved  - Possibility to allow student save their progress while playing  - Possibility to allow lecturer to analyze student performance |

# 2.0 Project Plan

## 2.1 System Development Methodology

The system development methodology that we apply is Spiral Model. It is a risk-driven software process framework that combines design and prototyping. It was proposed by Boehm in 1988. Spiral Model contains many loops in the process and each of the loops represents each phase. In this model, the development team begins with a small set of requirements. Then, the team will be going through every phase. (Alshamrani & Bahattab, 2015)

There are four phases in Spiral Model, Planning, Risk Analysis and Mitigation, Development and Review and Evaluation.

Phase 1: Planning

In the planning phase, we communicate between team members to discuss the topic that we are doing. Then, we list out the issues and solutions. Every team member provides ideas and combines. We also distribute work among all members equally. Besides, we set up a Gantt chart to follow the scheduled duration.

Phase 2: Risk Analysis and Mitigation

In the second phase, we discuss about the potential risk of the project. For example, the risk can be failure of calculating the score obtained in the questions. This may cause the student to get an incorrect score. Solutions to the risk will be produced too.

Phase 3: Development

In Development phase, we produce the context diagram, data flow diagram, data dictionary, entity relationship diagram, pseudocode, flowcharts and source code. All elements are essential to make sure that the system can run smoothly with any problem.

Phase 4: Review and Evaluation

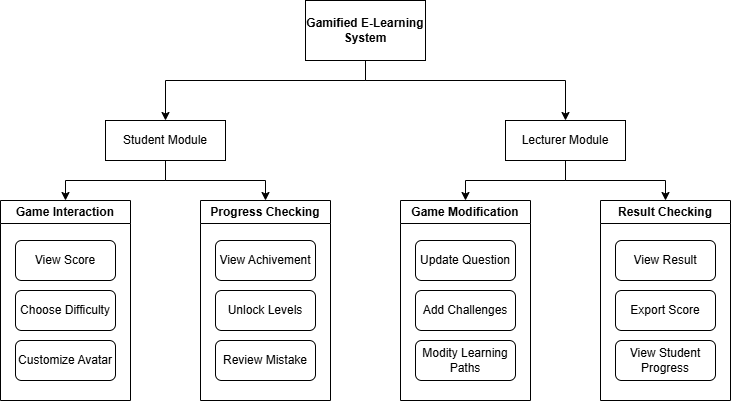
In the final phase of Spiral Model, we test the complete system. We make sure that there are no any logic error and other errors. If the system has successfully achieved the requirements, the development process is complete. (Risener, 2022)

## 2.2 Gantt Chart

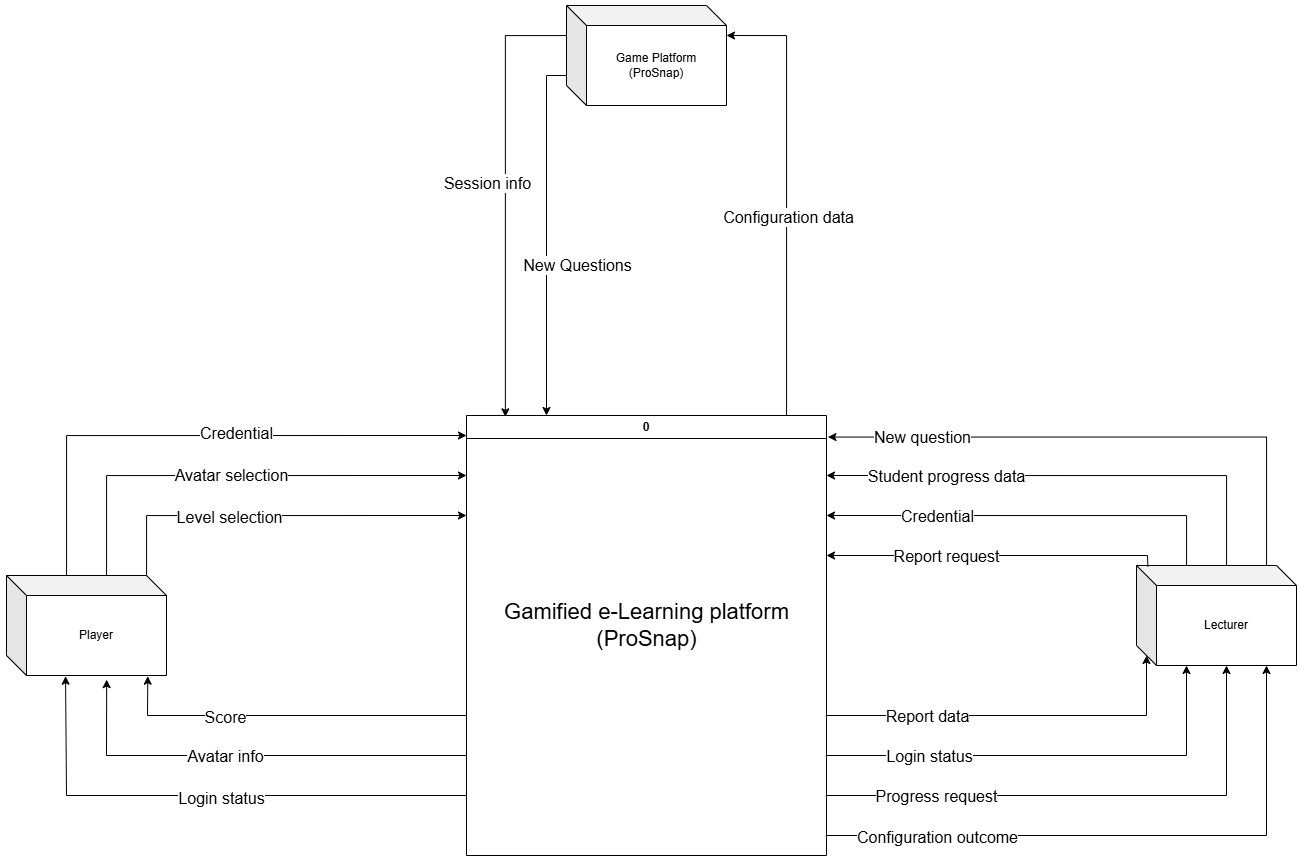
(Gantt Chart can be viewed in the following hyperlink)

<https://cloudmails-my.sharepoint.com/:x:/g/personal/tp076666_mail_apu_edu_my/EUirO0aaaOJAtE3kxpUDRKoB-CVJG-gRedIBK9HahFSWbA?e=oNAB1T>

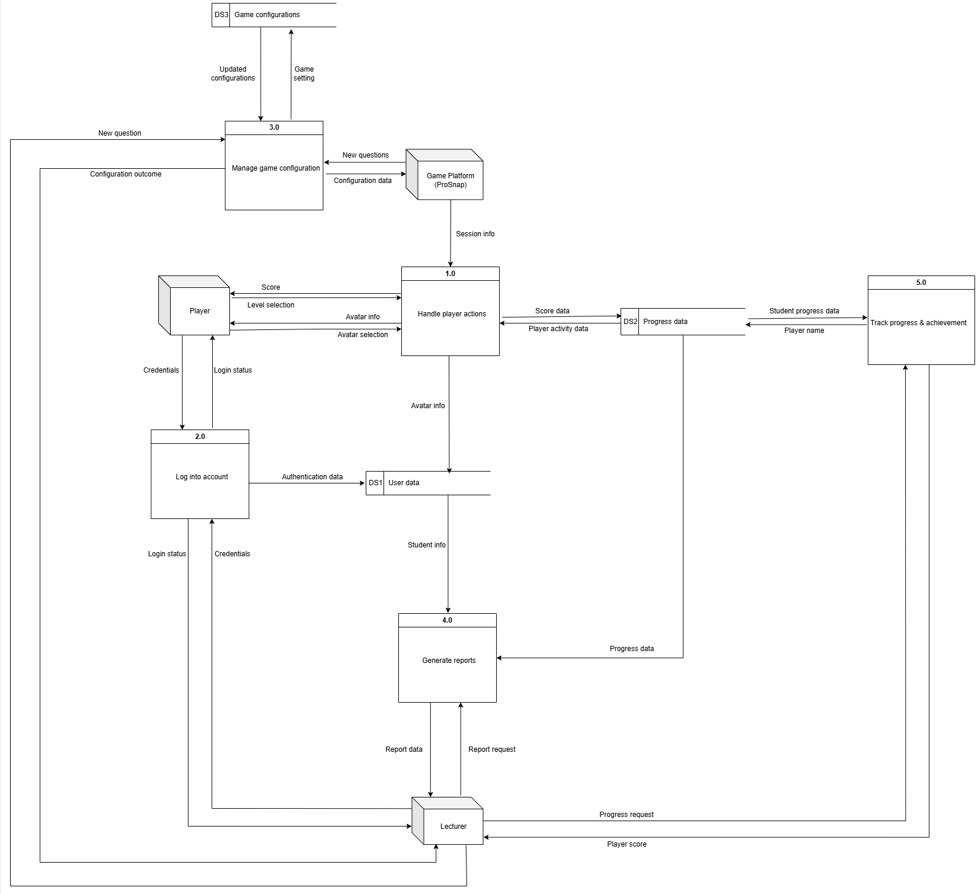
# 3.0 System Hierarchy Chart



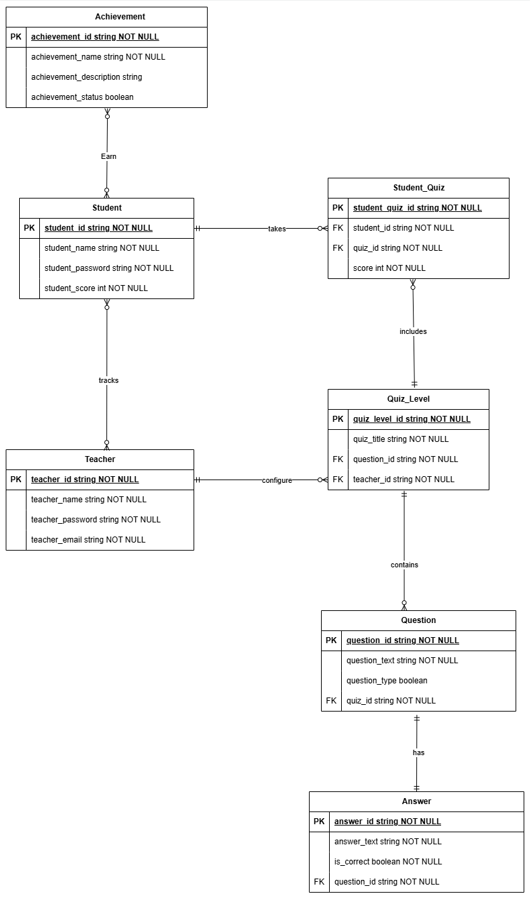
# 4.0 Context Diagram



# 5.0 Data Flow Diagrams Level 0



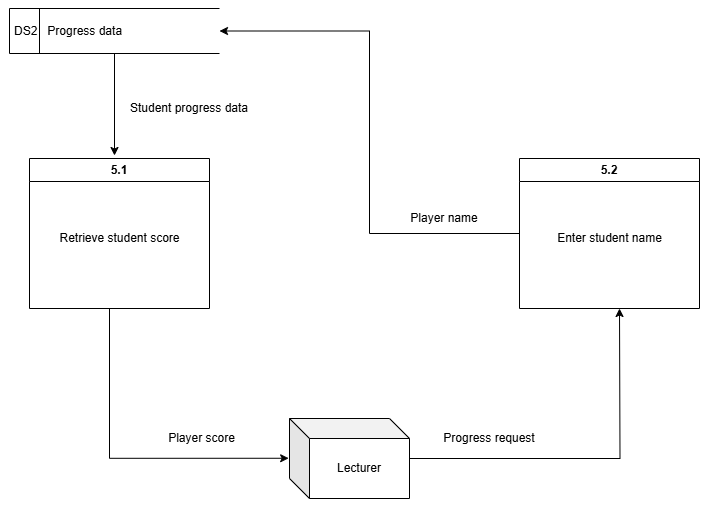
# 6.0 Entity Relationship Diagram



# 7.0 Individual Component

## 7.1 Chek Ka Nic TP074894

### 7.1.1 Data Flow Diagram Level 1



### 7.1.2 Data Dictionary

**External Entity**

**Name:** Lecturer

**Description:** Lecturer can track student progress and score

**Input data flow:** Student progress data

**Output data flow:** Progress request

**Process**

**Name:** 5.2 Enter student name

**Description:** Receive student name upon progress request by lecturer

**Input data flow:** Progress request

**Output data flow:** Player name

**Process:** START

Read in progress request

Receive player name

IF player name is invalid

Request confirmation for correct player name

ELSE

Forward player name to retrieve student progress

END ELSE

END

**Data Flow**

**Name:** Student progress data

**Description:** Contain information of student, score, etc...

**Origin/Source:** Progress data data store

**Destination/Sink:** Process5.1 Retrieve student score

**Data Structure:** Student progress data = Student ID + Student name + Total score

**Data Element:**

* **Name:** Student ID
* **Description:** Used to uniquely identify each student that has registered
* **Element Characteristics:**

**Type:** Alphanumeric

**Length:** 4

**Output format:** S001

**Data Store**

**Name:** DS2 Progress data

**Description:** To store the details of each student’s progress

**Input data flow:** Player name

**Output data flow:** Student progress data

**Data Structure:** Progress data = Progress ID + Student ID + Student name

+ [Lecturer remarks]

**Data Element:**

* **Name:** Progress ID
* **Description:** Used to uniquely identify each progress made by student
* **Element Characteristics:**

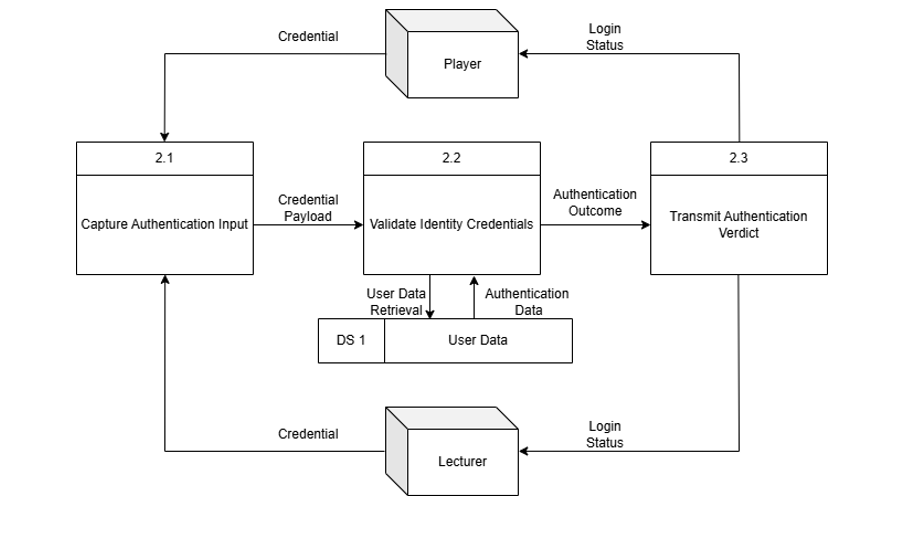
**Type:** Alphanumeric

**Length:** 5

**Output format:** PO001

## 7.2 Anderson Tey Le Tian TP076394

### 7.2.1 Data Flow Diagram Level 1



### 7.2.2 Data Dictionary

**External Entity**

**Name:** Player

**Description:** Player enter user ID and password into the system to recognise their identity for using the system

**Input data flow:** Login Status

**Output data flow:** Credential

**Process**

**Name:** 2.1 Capture Authentication Input

**Description:** Receive users’ personal information and send it to the system for login purpose

**Input data flow:** Credential

**Output data flow:** Credential Payload

**Process:**

START

Check whether credential is received;

IF receive credential,

Send it to the system;

ELSE,

Wait until credential is received;

ENDIF,

END.

**Data Flow**

**Name:** Credential

**Description:** To allow the system to verify the user

**Origin/Source:** Player/Lecturer external entity

**Destination/Sink:** 2.1 Capture Authentication Input Process

**Data Structure:** Credential = User ID + Password + Role

**Data Element:**

* **Name:** User ID
* **Description:** Used to identify the User of the system
* **Element Characteristics:**

**-Type:** Alphanumeric

**-Length:** 5

**-Output format:** AA001

**Data Store**

**Name:** DS1 User Data

**Description:** Store the information of user by ID and password

**Input data flow:** User Data Retrieval

**Output data flow:** Authentication Data

**Data Structure:** User Data = User ID + Password + Role

**Data Element:**

* **Name:** User ID
* **Description:** Used to identify the User of the system
* **Element Characteristics:**

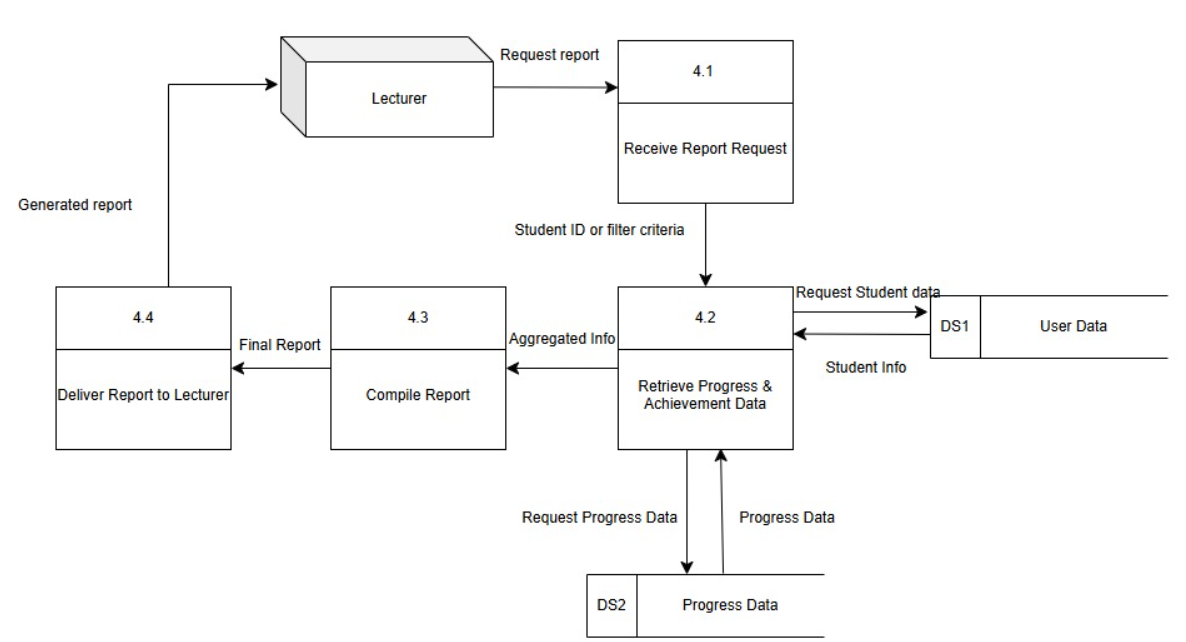
**Type:** Alphanumeric

**Length:** 5

**Output format:** AA001

## 7.3 Chan Guan Ming TP076256

### 7.3.1 Data Flow Diagram Level 1



### 7.3.2 Data Dictionary

**External Entity**

**Name:** Lecturer

**Description:** Lecturer requests student progress reports and receives compiled report data to monitor student performance.

**Input Data Flow:** Report request

**Output Data Flow**: Report data

**Process**

Name: 4.1 Receive Report Request

Description: send lecturer’s request to the next process for generate report

Input data flow: Report request from lecturer

Output data flow: Requested student ID or filter criteria

Process:

START

Wait for input from lecturer;

IF report request is received THEN

Extract student ID or filter criteria from request;

Validate lecturer access rights;

IF access is valid THEN

Forward request to data retrieval process (4.2);

ELSE

Display "Access Denied" error message;

END IF

END IF

END

**Data Flow**

Name: Report data

Description: The report that student’s score based on lecturer’s request.

Origin/Source: 4.0 Generate Reports

Destination/Sink: Lecturer

Data Structure: Report data = Student Name + Score

Data Element:

* **Name:** User ID
* **Description:** A unique identifier to identify the user of the system
* Element Characteristics:
* **Type:** Alphanumeric
* **Length:** 5
* Output format: AA001

**Data Store**

Name: DS1 User Data

Description: To store authentication and personal information of all users including students and lecturers.

Input Data Flow: Authentication data

Output Data Flow: User data, Student info, Lecturer info

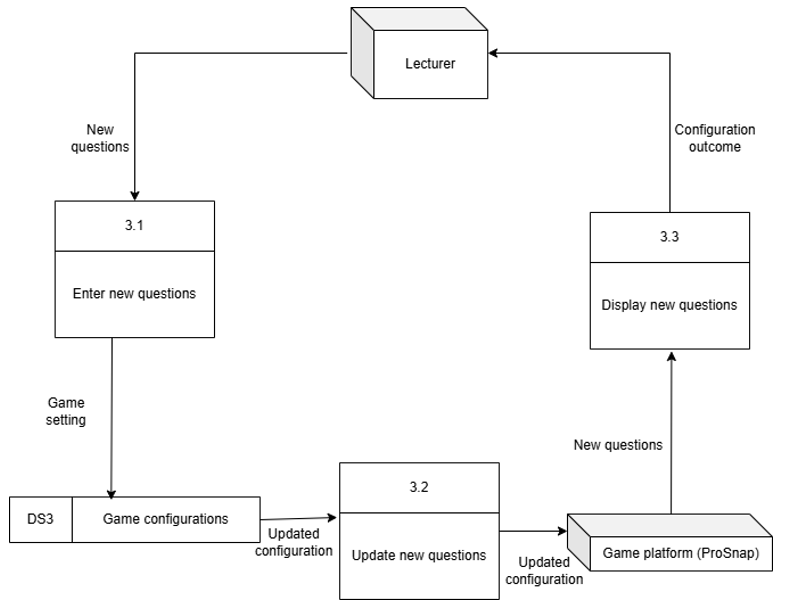
Data Structure: User = Name + Password + Email

Data element:

* Name: Password
* Description: used to verify the credentials of a user by logging in
* Element Characteristics:
* Type: Alphanumeric
* Length: 8
* Output format: 31398800

## 7.4 Chong Wei Jie TP076546

### 7.4.1 Data Flow Diagram Level 1



### 7.4.2 Data Dictionary

**External Entity**

Name: Lecturer

Description: Lecturer enters new questions into the data store and then it is displayed.

Input data flow: Configuration outcome

Output data flow: New questions

**Process**

Name: 3.3 Display new questions

Description: Receive new questions and then display the result.

Input data flow: New questions

Output data flow: Configuration outcome

Process:

START

Check whether new question is received:

If received,

Print result;

Show new questions;

ELSE,

Wait until the question is received;

END IF

END

**Data Flow**

Name: Game setting

Description: To allow lecturer to make changes to the game.

Origin/Source: 3.1 Enter new questions

Destination/Sink: DS3 Game configurations

Data Structure: Game setting = Level + Level ID + Question + Question ID

Data Element:

* Name: Level ID
* Description: Used to identify the level of the game
* Element Characteristics:
* Type: Alphanumeric
* Length: 3
* Output format: L01

**Data Store**

Name: DS3 Game Configuration

Description: Store the game setting (new question and answer)

Input data flow: Game setting

Output data flow: Updated configuration

Data structure: Game setting = Question ID + New Questions + Answer ID + New Answers

Data element:

* Name: Question ID
* Description: Use to identify the updated question.
* Element Characteristics:
* Type: Alphanumeric
* Length: 6
* Output format: Q00001

## 7.5 Mah Zheng Yang TP076666

### 7.5.1 Data Flow Diagram Level 1

A diagram of a flowchart

AI-generated content may be incorrect.

### 7.5.2 Data Dictionary

**External Entity**

**Name:** Player

**Description:** Player can choose avatar, choose difficulty and play game

**Input data flow:** Result

**Output data flow:** Player status

**Process**

**Name:** 1.3 Choose difficulty

**Description:** Allow players to choose the target score

**Input data flow:** Avatar information

**Output data flow:** Target score

**Process:** START

Read in avatar information

Receive difficulty

IF difficulty is easy

Set target score to 400

ELSE IF difficulty is medium

Set target score to 500

ELSE IF difficulty is hard

Set target score to 600

END ELSE

END

**Data Flow**

**Name:** Level information

**Description:** Contain information such as level difficulty, level selection, avatar status

**Origin/Source:** Process 1.4 Choose level

**Destination/Sink:** Process 1.2 Play game

**Data Structure:** Level information = Level difficulty + Level selection + Avatar status

**Data Element:**

* **Name:** Level difficulty
* **Description:** Used to uniquely identify difficulty selected by the player
* **Element Characteristics:**

**Type:** Alphanumeric

**Length:** 3

**Output format:** D01

**Data Store**

**Name:** DS2 Progress data

**Description:** To store the details of each student’s progress

**Input data flow:** Score

**Output data flow:** Player activity data

**Data Structure:** Progress data = Score + Total score

**Data Element:**

* **Name:** Total score
* **Description:** Used to uniquely identify the total score scored by player
* **Element Characteristics:**
* **Type:** Numeric
* **Length:** 3 - 64
* **Output format:** 7000

# 8.0 Screen Design and Report Design with User Manual

Register / Login Page:

A screenshot of a video game

AI-generated content may be incorrect.

There are two buttons on the main page, Register and Login. Users need to enter a username, email address and password to register account. Users need to select L for lecturer and S for student too. The username, email address and password cannot be repeated as previous user’s. Users need to enter the correct username and password to log into their account. If correct username and password are entered, the system shows “Login Successfully! Welcome back”.

A screenshot of a video game

AI-generated content may be incorrect.

Main Page (Lecturer):

A screenshot of a video game

AI-generated content may be incorrect.

There are three buttons in the main page, “EDIT QUESTION”, “VIEW SCORE” and “EXIT”. Lecturer can edit questions and answers when they click the “EDIT QUESTION” button.

A screenshot of a video game

AI-generated content may be incorrect.

Lecturer can view students’ score when they click “VIEW SCORE” and enter the student’s name.

**A screenshot of a video game

AI-generated content may be incorrect.**

Lecturer clicks “EXIT” button to log out their account and return to the LOGIN/REGISTER page.

Main Page (Student):

A screenshot of a video game

AI-generated content may be incorrect.

Student clicks “START GAME” button to enter the game. After clicking “START GAME” button, the system will show three avatars for user to choose.

A screenshot of a video game

AI-generated content may be incorrect.

Then, students need to choose the game level, Level 1, Level 2 and Level 3. Each level represents different subjects and has different routes of maze. There are 3 difficulties too, Easy, Medium and Hard. Easy difficult requires 400 points to pass, Medium difficulty requires 500 points to pass and Hard difficulty requires 600 points to pass.

A screenshot of a game

AI-generated content may be incorrect.

A screenshot of a video game

AI-generated content may be incorrect.

In the game, players need to start from the starting point and end at the ending point. When they meet the wolfman, they will have to answer the question. They have three chances (love symbol) to answer the questions wrongly. If they answer three times wrongly, then they will lose the games. They will get 100 points when they answer one correct question or open one chest. If they get enough points for the game and successfully go out to the ending points, they pass the game.

A screenshot of a video game

AI-generated content may be incorrect.

They can pause the game by clicking 

They can continue the game, restart the game or exit the game. If they click “RESTART”, the score will be returned to 0 and the avatar will return to the starting point. If they click “EXIT” button, they will return to the main page.

A screenshot of a video game

AI-generated content may be incorrect.

When players click “ACHIEVEMENT" button, it will show the achievement that players have made. Once the achievement has been done, it will show a tick.

A screenshot of a video game

AI-generated content may be incorrect.

Players can click A black arrow in a yellow circle

AI-generated content may be incorrect. button to return to the main page.

Players can click “EXIT” button to log out their account and return to the LOGIN/REGISTER page.

# 9.0 Test Plan (Unit Testing and User Acceptance Testing)

(Test plan is included in the following hyperlink) (UAT is submitted independently in a pdf file)

<https://cloudmails-my.sharepoint.com/:x:/g/personal/tp076666_mail_apu_edu_my/ESCzskTMQTNEg5NG2adNrD4BQdWqJ1FP2qvbXmak-vSoDA?e=yr9D9j>

# 10.0 Significant Source Code with Explanation

**Source Code Explanation:**

**Sprite 1: Main**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a game

AI-generated content may be incorrect.**

Explanation:

**Game Initialization**

* When the green flag is clicked: the sprite is hidden at the start.
* Several game variables are initialized:
* status is set to 0, indicating the game’s starting state.
* health starts at 3, meaning the player has three lives.
* no of keys starts at 0, keys can be collected during the game.
* score is set to 0 to track points earned.

**Game Over Handling**

* When health reaches 0: the score display is hidden.
* The game broadcasts "Game Over" and "endgame", to triggering other elements to stop and transition.

**General Behavior**

* When the "endgame" message is received:
* The sprite hides, indicating the game session has ended.

**Sprite 2: Stage 1**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Explanation:**

**This sprite functions as an interactive menu button in the game.**

**Visibility & Interaction**

* It hides when receiving "next" or "difficulty" broadcasts, meaning it disappears when transitioning to different parts of the game.
* It shows when receiving "Choose Level", indicating it is used for level selection.

**Hover Effect**

* Inside a forever loop, the sprite checks if it's touching the mouse pointer:
* If true, it increases to 110% size (highlight effect).
* Otherwise, it remains at 100% size.

**Quiz Initialization**

* When the green flag is clicked, the sprite sets up quiz-related variables:
* answer1 stores a list of possible answers.
* question1 contains a list of quiz questions.
* edit manages reset and cancel options.

**Positioning & Click Handling**

* The sprite starts at (-141, -30) coordinates.
* When clicked, it:
* Plays a "button" sound.
* Sets current\_stage to 1.
* Checks the player’s role (Student or other).
  + If Student, it broadcasts "difficulty" (probably showing difficulty options).
  + Otherwise, it broadcasts "next".
* Finally, it hides itself.

**Sprite 3: Stage 2**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite functions as an interactive menu button in the game.

**Visibility & Interaction**

* The sprite hides when receiving "next" or "difficulty" broadcasts, meaning it disappears when transitioning to different parts of the game.
* It shows when receiving "Choose Level", indicating it is used for level selection.

**Hover Effect**

* Inside a forever loop, the sprite checks if it's touching the mouse pointer:
* If true, it increases to 110% size (highlight effect).
* Otherwise, it remains at 100% size.

**Quiz Initialization**

* When the green flag is clicked, the sprite sets up quiz-related variables:
* answer2 stores a list of possible answers.
* question2 contains a list of quiz questions.
* edit manages reset and cancel options.

**Positioning & Click Handling**

* The sprite starts at (0, -30) coordinates.
* When clicked, it:
* Plays a "button" sound.
* Sets current\_stage to 2.
* Checks the player’s role (Student or other).
  + If Student, it broadcasts "difficulty" (probably showing difficulty options).
  + Otherwise, it broadcasts "next".
* Finally, it hides itself.

**Sprite 4: Stage 3**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite functions as a selectable quiz button in the game.

**Visibility & Interaction**

* The sprite hides when receiving "difficulty" or "next" broadcasts, meaning it disappears when transitioning to different game sections.
* It shows when receiving "Choose Level", indicating it is used for selecting quiz difficulty.

**Hover Effect**

* Inside a forever loop, the sprite checks if it's touching the mouse pointer:
* If true, it increases to 110% size (highlight effect).
* Otherwise, it remains at 100% size.

**Quiz Initialization**

* When the green flag is clicked, the sprite sets up quiz-related variables:
* answer3 stores a list of correct answers related to HTML.
* question3 contains a list of quiz questions.
* edit manages reset and cancel options for the quiz session.

**Positioning & Click Handling**

* The sprite starts at (143, -30) coordinates.
* When clicked, it:
* Plays a "button" sound.
* Sets current\_stage to 3.
* Checks the player’s role (Student or other).
  + If Student, it broadcasts "difficulty" to show difficulty selection.
  + Otherwise, it broadcasts "next", moving to the next stage.
* Finally, it hides itself.

**Sprite 5: Chest 1**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite represents an interactive chest that the player can open to earn rewards.

**Visibility & Positioning**

* When receiving "next":
* The sprite moves to different positions depending on the value of current\_stage.
* It appears at (-43, -90) for stage 1, (-43, -93) for stage 2, and (131, -134) for stage 3.
* After moving, the sprite shows itself.

**Chest Opening Mechanism**

* The chest begins closed (open\_chest1 = false).
* It switches to a closed chest costume and enters a forever loop checking interaction conditions:
* If open\_chest1 = false AND the chest is touching the Main sprite:
* The "orb" sound plays.
* The costume changes to an opened chest.
* The player earns 1 key (no of keys increases by 1).
* The score increases by 100 points.
* The chest state updates to open\_chest1 = true, preventing repeated activation.

**Game Start and End Behavior**

* When the green flag is clicked, the chest hides.
* When receiving "endgame", the chest hides, ensuring it does not interfere with the ending sequence.

**Sprite 6: Chest 2**

**A screenshot of a computer game

AI-generated content may be incorrect.**

**Explanation:**

**This sprite represents another interactive chest that the player can open to earn rewards.**

**Visibility & Positioning**

* When receiving "next":
* The sprite moves to different positions based on current\_stage:
  + Stage 1: Moves to (137, -88).
  + Stage 2: Moves to (133, 135).
  + Stage 3: Moves to (0, 134).
* After moving, the sprite shows itself.

**Chest Opening Mechanism**

* The chest starts closed (open\_chest2 = false).
* It switches to the closed chest costume and enters a forever loop checking interaction conditions:
* If open\_chest2 = false AND the chest is touching the Main sprite:
* The "orb" sound plays.
* The costume changes to an opened chest.
* The player earns 1 key (no of keys increases by 1).
* The score increases by 100 points.
* The chest state updates to open\_chest2 = true, preventing repeated activation.

**Game Start and End Behavior**

* When the green flag is clicked, the chest hides.
* When receiving "endgame", the chest hides, ensuring it does not interfere with the ending sequence.

**Sprite 7: Chest 3**

**A screenshot of a computer game

AI-generated content may be incorrect.**

**Explanation:**

This sprite represents a third interactive chest that the player can open to earn rewards.

**Visibility & Positioning**

* When receiving "next":
* The sprite moves to different positions based on current\_stage:
  + Stage 1: Moves to (-1, -2).
  + Stage 2: Moves to (1, -1).
  + Stage 3: Moves to (40, -47).
* After moving, the sprite shows itself.

**Chest Opening Mechanism**

* The chest starts closed (open\_chest3 = false).
* It switches to the closed chest costume and enters a forever loop checking interaction conditions:
* If open\_chest3 = false AND the chest is touching the Main sprite:
* The "orb" sound plays.
* The costume changes to an opened chest.
* The player earns 1 key (no of keys increases by 1).
* The score increases by 100 points.
* The chest state updates to open\_chest3 = true, preventing repeated activation.

**Game Start and End Behavior**

* When the green flag is clicked, the chest hides.
* When receiving "endgame", the chest hides, ensuring it does not interfere with the ending sequence.

**Sprite 8: Wolfman 1**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite is an interactive character that engages with the player through movement, interactions, and quizzes.

**Positioning & Visibility**

* When receiving "next", the sprite moves to stage-specific positions:
  + Stage 1: Moves to (-50, -90).
  + Stage 2: Moves to (-45, -95).
  + Stage 3: Moves to (-30, -80).
* The sprite shows itself and faces right (90°) upon reaching its position.

**Movement & Interaction Loop**

* If the player’s role is Student, the sprite enters a continuous interaction loop:
* Moves smoothly while switching costumes for animation.
* Checks for obstacles, turning back if movement is blocked.

**Quiz System**

* The sprite asks a question depending on the current stage:
  + If the answer is correct, it increases the player’s score.
  + If the answer is wrong, the "Wrong answer" message is displayed.
* The quiz is structured using lists (question1, answer1, etc.), dynamically replacing questions based on user input.
* If status = 1, the quiz interaction continues until status = 0, ensuring the quiz sequence runs correctly.

**Game Start & End Behavior**

* When the green flag is clicked, the sprite hides.
* When receiving "endgame", the sprite hides, ensuring it does not interfere with post-game events.

**Sprite 9: Wolfman 2**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite is an interactive NPC that moves across different stages and engages players in quiz-based interactions.

**Positioning & Visibility**

* When receiving "next", the sprite moves to stage-specific positions:
  + Stage 1: Moves to (41, 0).
  + Stage 2: Moves to (-91, -136).
  + Stage 3: Moves to (-107, 46).
* After reaching its position, the sprite shows itself.

**Game Start & End Behavior**

* When the green flag is clicked, the sprite hides and resizes to 80%.
* When receiving "endgame", it hides, ensuring it does not interfere with post-game events.

**Movement & Interaction Loop**

* The sprite faces right (90°) upon appearing.
* If the player's role is Student, the sprite enters a movement sequence:
* Moves in steps while switching costumes for animation.
* Waits 0.08 seconds between movements.
* If touching obstacles, it turns 180° and moves backward.

**Quiz System**

* The sprite engages the player in a quiz interaction, asking questions based on the stage:
* Stage 1:
  + Asks item (2) of question1.
  + If the answer is correct, the score increases by 100.
  + If the answer is wrong, the "Wrong answer" message is displayed, and health decreases by 1.
* Stage 2:
  + Asks item (2) of question2.
  + If the answer is correct, the score increases by 100.
  + If the answer is wrong, health decreases by 1.
* Stage 3:
* Uses a similar structure, continuing the quiz interactions.

**Sprite 10: Wolfman 3**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Explanation:**

**Quiz System & Scoring**

* If current\_stage = 3:
* It prompts the player with a quiz question stored in question3.
* If the answer matches answer3, the score increases by 100 points, and "Correct!!" is displayed for 2 seconds.
* If incorrect, the player's health decreases by 1, and "Wrong answer" appears for 2 seconds.

**Interactive Answer Editing**

* If the current stage is not 3, the game waits for status = 1 before allowing edits.
* It checks if the mouse is hovering over the sprite and clicked.
* If "ResetQuestion" is entered:
* The game asks for a new question (temp\_question) and new answer (temp\_answer).
* If any input is empty, "Invalid question or answer" is displayed.

**Dynamic Question Updates**

* Based on the stage, it replaces the first item in the corresponding question and answer lists:
  + Stage 1: Updates question1 and answer1.
  + Stage 2: Updates question2 and answer2.
  + Stage 3: Updates question3 and answer3.
* Displays "Update Successfully" for 2 seconds after applying the change.

**Sprite 11: Door**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite controls stage progression and game completion conditions.

**Interactions with the Player**

* If the sprite touches Main (the player):
* It hides the score variable.
* Sets status to 0.
* Broadcasts "endgame", triggering the game-ending sequence.

**Game End Handling**

* When receiving "endgame":
* The sprite hides, ensuring it disappears from the final screen.

**Win/Loss Conditions**

* The game checks if the player has met stage requirements:
* If no of keys < 3 OR score < target score:
  + Plays the "lost" sound.
  + Broadcasts "Game Over".
* Otherwise:
* Plays the "win" sound.
* Broadcasts "Game Passed".
* Updates the player’s total\_score by adding their current score to the stored value in username\_list.

**Stage Transitions**

* When receiving "next", the door moves to its respective stage positions:
  + Stage 1: (165, -136)
  + Stage 2: (158, -45)
  + Stage 3: (156, -90)
* After repositioning, the sprite shows itself, allowing interaction.

**Click Behavior**

* If the sprite is clicked, it:
* Sets its ghost effect to 100.
* Hides itself, likely for visual effects or transitions.

This sprite is essential for managing progression, determining win/loss conditions, and handling player interactions. Let me know if you need refinements! 🚀

**Sprite 12: Heart 1**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages visibility changes based on game conditions.

**Game Start & End Behavior**

* When receiving "endgame", the sprite hides, ensuring it does not interfere with post-game events.
* When the green flag is clicked, it hides, meaning it remains unseen at the beginning.
* If health < 1, the sprite hides, likely to reflect losing all lives.

**Visibility in Gameplay**

* When receiving "next", the sprite shows, suggesting it appears at a certain stage transition.

**Sprite 13: Heart 2**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages visibility changes based on game conditions.

**Game Start & End Behavior**

* When receiving "endgame", the sprite hides, ensuring it does not interfere with post-game events.
* When the green flag is clicked, it hides, meaning it remains unseen at the beginning.
* If health < 2, the sprite hides, likely to reflect losing all lives.

**Visibility in Gameplay**

* When receiving "next", the sprite shows, suggesting it appears at a certain stage transition.

**Sprite 14: Heart 3**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages visibility changes based on game conditions.

**Game Start & End Behavior**

* When receiving "endgame", the sprite hides, ensuring it does not interfere with post-game events.
* When the green flag is clicked, it hides, meaning it remains unseen at the beginning.
* If health < 3, the sprite hides, likely to reflect losing all lives.

**Visibility in Gameplay**

* When receiving "next", the sprite shows, suggesting it appears at a certain stage transition.

**Sprite 15: Login**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages the authentication system, handling both login and registration processes.

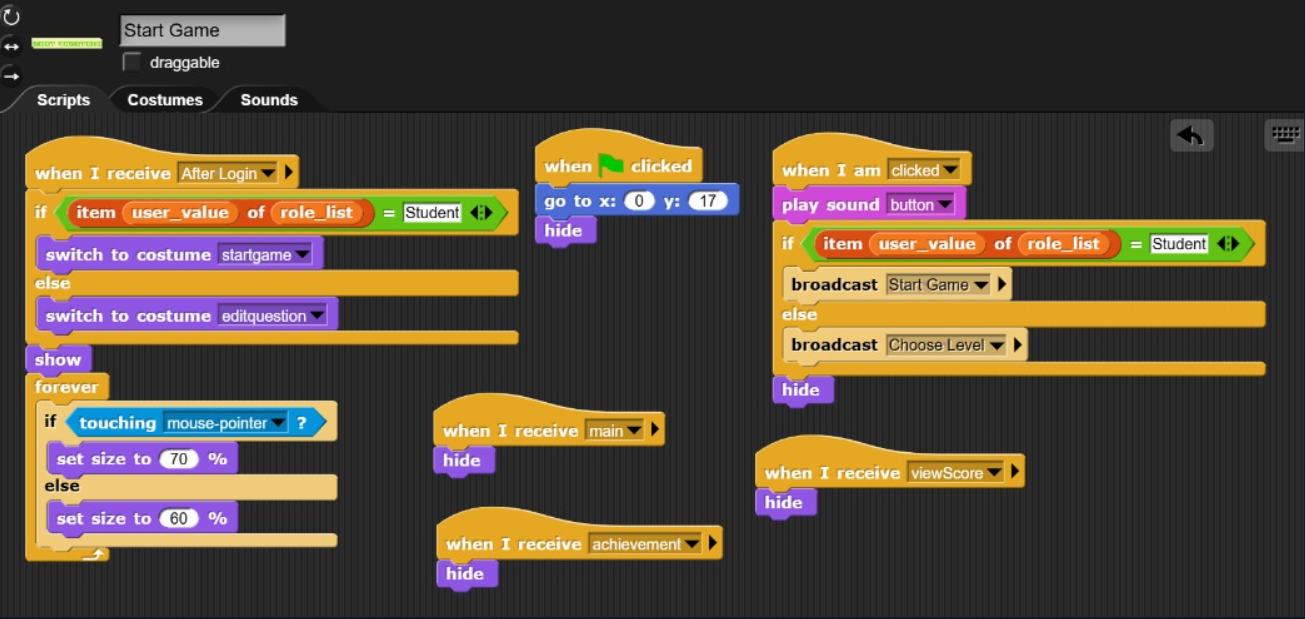
**Login Process**

* When receiving "login", the sprite hides and asks for a username, password and email address.
* It stores the username and password in temp username and temp password.
* It checks if both credentials exist in the saved lists (username\_list, password\_list and email\_list) and ensures their indexes match, confirming a valid login.
* If valid:
  + Sets user to the index of the username.
  + Switches to "welcome" costume, briefly showing a success message.
  + Broadcasts "loading", followed by "afterLogin".
* If invalid:
* Switches to "invalid" costume, briefly showing an error.
* Broadcasts "loading", followed by "main", returning to the menu.

**Registration Process**

* When receiving "register", the sprite hides and asks for:
  + A new username (username).
  + A new password (password).
  + An email address
  + A role selection (Student or Lecturer).
* The script ensures valid input:
  + If empty, it triggers an "invalid" costume and sends "main".
  + If credentials already exist, it displays "repeated", preventing duplicates.
  + If selecting Student (8) or Lecturer (12), it adds the role and score to respective lists.
  + If selecting an invalid role (0), it shows "invalidRole" and sends "negative".
  + Otherwise, it stores new user credentials, initializes achievement variables, and shows "complete".
* After registration, it broadcasts "loading", transitioning back to "main".

**Sprite 16: Start Game**

****

**Explanation:**  
This sprite functions as a "Start Game" button that reacts based on the user's role, either starting the game or leading to a question editor.

**Game Start Behavior**

* When the green flag is clicked, the sprite moves to position (0, 17) and hides itself.

**Visibility & Hover Effect**

* When receiving After Login, it checks the user's role:
  + If "Student", it switches to the “startgame” costume.
  + Otherwise, it switches to “editquestion.”
* The sprite becomes visible and enters a loop to detect mouse hover:
  + If the mouse pointer touches it, the size increases to 70%.
  + Otherwise, it stays at 60%.

**Click Behavior & Broadcasts**

* When the sprite is clicked:
  + It plays a button sound.
  + If the user is a "Student", it broadcasts Start Game.
  + Otherwise, it broadcasts Choose Level.
  + Finally, it hides itself.

**Interface Management**

* The sprite hides again when receiving messages like main, achievement, or viewScore to keep the interface clean.

**Sprite 17: Exit**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite serves as a menu button that allows users to return to the main screen.

**Visibility & Mouse Interaction**

* When receiving "After Login", the sprite shows itself and enters a forever loop:
* If the mouse pointer is touching it, the sprite increases in size to 110% for a hover effect.
* Otherwise, it remains at 100% size.

**Click Behavior**

* When clicked:
* Plays a "button" sound for feedback.
* Broadcasts "main", returning the player to the main menu.
* The sprite then hides itself.

**Game Transition Management**

* The sprite hides when receiving:
* "Start Game" – when the game begins.
* "Choose Level" – when selecting a level.
* "viewScore" – when viewing scores.
* "achievement" – when navigating to achievements.

**Game Start Behavior**

* When the green flag is clicked, the sprite moves to (0, -125), ensuring it starts at a set position.
* If "achievement" is broadcast, the sprite hides, likely indicating it does not appear in the achievement screen.

**Sprite 18: Achievement**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

**This sprite manages access to achievement tracking and score viewing based on the user's role.**

**Visibility & Appearance**

* When receiving "After Login":
  + If the user's role is Student, the sprite switches to bAchievement costume.
  + Otherwise, it switches to viewscore costume.
  + The sprite then shows itself.
* Enters a forever loop to detect mouse hover:
* If touching the mouse pointer, it increases in size to 67%.
* Otherwise, it remains at 57%.

**Click Behavior**

* When clicked:
* Plays a button sound for feedback.
* If the user is a Student, it broadcasts "achievement" to access achievements.
* Otherwise, it broadcasts "viewScore" to display scores.

**Game Transition Management**

* The sprite hides itself when receiving:
* "main" – returning to the main menu.
* "Choose Level" – transitioning to level selection.
* "Start Game" – beginning gameplay.

**Game Start Positioning**

* When N is clicked, the sprite moves to (0, -54) and hides itself.
* When the green flag is clicked, it hides, ensuring it only appears when needed.

**Sprite 19: Skin1**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite is a character selection button that allows the player to choose an avatar.

**Game Start Behavior**

* When the green flag is clicked:
* The sprite moves to (-148, -36) as its default position.
* It hides itself, ensuring it only appears when needed.

**Visibility & Hover Effect**

* When receiving "Start Game":
* The sprite shows itself and enters a forever loop to detect mouse hover.
* If touching the mouse pointer, its size increases to 70%, creating a highlight effect.
* Otherwise, it remains at 60%.

**Character Selection & Broadcasts**

* When clicked:
* Broadcasts "Choose Level", likely transitioning to level selection.
* Sets current\_avatar to 1, confirming the player's choice.
* The sprite then hides itself to ensure a clean interface.

**Scene Management**

* When receiving "Choose Level", the sprite hides, ensuring it doesn’t interfere with level selection.

**Sound Effect**

* When mouse-entered, the sprite plays "skin1" sound, enhancing interactivity with an audio cue.

**Sprite 20: skin2**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite is a character selection button that allows the player to choose an avatar.

**Game Start Behavior**

* When the green flag is clicked:
* The sprite moves to (5, -36) as its default position.
* It hides itself, ensuring it only appears when needed.

**Visibility & Hover Effect**

* When receiving "Start Game":
* The sprite shows itself and enters a forever loop to detect mouse hover.
* If touching the mouse pointer, its size increases to 70%, creating a highlight effect.
* Otherwise, it remains at 60%.

**Character Selection & Broadcasts**

* When clicked:
* Broadcasts "Choose Level", likely transitioning to level selection.
* Sets current\_avatar to 2, confirming the player's choice.
* The sprite then hides itself to ensure a clean interface.

**Sound Effect**

* When mouse-entered, the sprite plays "skin2" sound, enhancing interactivity with an audio cue.

**Sprite 21: skin3**

**A screenshot of a computer game

AI-generated content may be incorrect.**

**Explanation:**

This sprite functions as a character selection button, allowing the player to choose an avatar.

**Game Start Behavior**

* When the green flag is clicked:
* The sprite moves to (151, -30) as its default position.
* It hides itself, ensuring it only appears when needed.

**Visibility & Hover Effect**

* When receiving "Start Game":
* The sprite shows itself and enters a forever loop that detects mouse hover.
* If touching the mouse pointer, its size increases to 70%, creating a highlight effect.
* Otherwise, it remains at 60%.

**Character Selection & Broadcasts**

* When clicked:
* Broadcasts "Choose Level", likely transitioning to level selection.
* Sets current\_avatar to 3, confirming the player's choice.
* The sprite then hides itself to ensure a clean interface.

**Sound Effect**

* When mouse-entered, the sprite plays "skin3" sound, enhancing interactivity with an audio cue.

**Sprite 22: Restart**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite serves as a game restart button, allowing players to retry or continue after game completion.

**Game Start & End Behavior**

* When the green flag is clicked, the sprite hides, ensuring it is not visible at the start.
* When receiving "After Login" or "pause", the sprite hides, ensuring it does not interfere with login or pause functionality.

**Restart Mechanism**

* When receiving "Game Over":
* Moves to (0, -20).
* Switches to the "Restart" costume.
* Shows itself to indicate that the player can restart.
* Enters a hover detection loop:
* If touching the mouse pointer, its size increases to 75%, creating a highlight effect.
* Otherwise, it remains at 65%.

**Play Again Button for Game Completion**

* When receiving "Game Passed":
* Moves to (0, 20).
* Switches to the "Play Again" costume.
* Shows itself, allowing the player to continue.
* Enters a hover detection loop with the same size adjustments.

**Click Behavior**

* When clicked:
* Plays a button sound for feedback.
* Sets status to 0, resetting the game state.
* Hides itself and broadcasts "Choose Level", transitioning back to level selection.

**Visibility in Transitions**

* When receiving "next", the sprite hides, ensuring it doesn’t appear during scene changes.

**Sprite 23: Next**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite enables stage progression by changing the current\_stage when clicked.

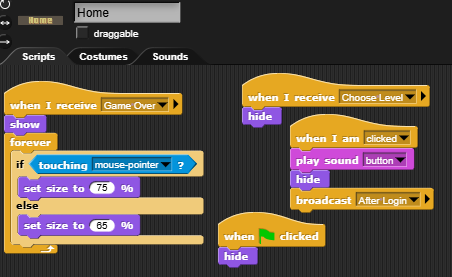
**Visibility & Hover Effect**

* When receiving "Choose Level", the sprite hides, ensuring it does not interfere during level selection.
* When receiving "Game Passed", the sprite shows itself and enters a forever loop for hover detection:
* If touching the mouse pointer, its size increases to 75%, creating a highlight effect.
* Otherwise, it remains at 85%.

**Click Behavior & Stage Progression**

* When clicked:
* Plays a button sound.
* Broadcasts "next", triggering the transition.
* If the current stage is 3, it resets current\_stage to 1.
* Otherwise, it increments current\_stage by 1 to move forward.
* The sprite then hides itself.

**Sprite 24: Home**

****

**Explanation:**

This sprite serves as a menu navigation button, allowing players to return to the home screen.

**Visibility & Hover Effect**

* When receiving "Game Over":
  + The sprite shows itself and enters a forever loop for hover detection:
  + If touching the mouse pointer, its size increases to 75%, creating a highlight effect.
  + Otherwise, it remains at 65%.
* When receiving "Choose Level", the sprite hides, ensuring it does not interfere during level selection.

**Click Behavior & Navigation**

* When clicked:
* Plays a button sound for feedback.
* Broadcasts "After Login", likely transitioning back to the main menu or login screen.
* The sprite then hides itself to ensure a clean interface.

**Game Start Behavior**

* When the green flag is clicked, the sprite hides, ensuring it only appears when needed.

**Sprite 25: Sprite**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite handles game pausing and visibility adjustments.

**Visibility & Hover Effect**

* When receiving "next", the sprite shows itself.
* Enters a forever loop for hover detection:
* If touching the mouse pointer, it increases size to 75%.
* Otherwise, it remains at 85%.

**Game Start Behavior**

* When the green flag is clicked, the sprite hides, ensuring it does not appear at the start.

**Click Behavior & Game Pause**

* When clicked:
* Plays a "button" sound for feedback.
* Hides the score variable.
* Broadcasts "endgame" and "pause" to trigger the pause state.
* The sprite then hides itself, ensuring the pause effect applies smoothly.

**Game End Handling**

* When receiving "endgame", the sprite hides, ensuring it does not interfere with post-game events.

**Sprite 26: Continue**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite enables players to resume the game after pausing.

**Visibility & Hover Effect**

* When receiving "pause", the sprite:
* Moves to (0, 65).
* Switches to the "continue" costume.
* Shows itself for interaction.
* Enters a forever loop for hover detection:
* If touching the mouse pointer, its size increases to 75%.
* Otherwise, it remains at 65%.

**Click Behavior & Game Resumption**

* When clicked:
* Plays a "button" sound.
* Broadcasts "next" to proceed with gameplay.
* The sprite then hides itself, removing the pause screen.

**Game Start & Transition Handling**

* The sprite hides itself when receiving:
  + "After Login" – ensuring it doesn’t interfere with login functionality.
  + "Choose Level" – preventing it from appearing during level selection.
* When the green flag is clicked, the sprite hides, ensuring it only appears when needed.

**Sprite 27: exit(2)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite serves as an exit function, allowing players to return to the main menu.

**Visibility & Hover Effect**

* When receiving "pause", the sprite:
* Moves to (0, -104).
* Switches to the "exit" costume.
* Shows itself for interaction.
* Enters a forever loop for hover detection:
* If touching the mouse pointer, its size increases to 75%.
* Otherwise, it remains at 85%.

**Click Behavior & Navigation**

* When clicked:
* Plays a button sound.
* Sets status to 0, possibly resetting the game state.
* Broadcasts "After Login", likely transitioning to the main menu or login screen.
* The sprite then hides itself, ensuring a clean interface.

**Game Start & Transition Handling**

* The sprite hides itself when receiving:
  + "next" – ensuring it does not appear during scene transitions.
  + "Choose Level" – preventing interference during level selection.
* When the green flag is clicked, the sprite hides, ensuring it only appears when necessary.

**Sprite 28: easy**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite allows the player to select Easy Mode for gameplay.

**Visibility & Hover Effect**

* When receiving "difficulty", the sprite shows itself and enters a forever loop:
* If touching the mouse pointer, its size increases to 60% for a hover effect.
* Otherwise, it remains at 50%.

**Game Start Behavior**

* When the green flag is clicked, the sprite moves to (-149, 22) and hides itself, ensuring it appears only when needed.

**Click Behavior & Gameplay Setup**

* When clicked:
* Broadcasts "next" to transition forward.
* Sets target score to 400, defining the difficulty level for Easy Mode.
* Hides itself, ensuring a clean interface.

**Transition Management**

* When receiving "next", the sprite hides, preventing interference during scene transitions.

**Sprite 29: medium**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite enables players to select Normal Mode, adjusting the game's difficulty.

**Visibility & Hover Effect**

* When receiving "difficulty", the sprite shows itself and enters a forever loop:
* If touching the mouse pointer, its size increases to 80% for a hover effect.
* Otherwise, it remains at 50%.

**Game Start Behavior**

* When the green flag is clicked, the sprite moves to (1, 22) and hides itself, ensuring it appears only when needed.

**Click Behavior & Gameplay Setup**

* When clicked:
* Broadcasts "next" to transition forward.
* Sets target\_score to 500, defining the difficulty level for Normal Mode.
* Hides itself, ensuring a clean interface.

**Transition Management**

* When receiving "next", the sprite hides, preventing interference during scene transitions.

**Sprite 30: hard**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite allows the player to choose Hard Mode, increasing the game's difficulty level.

**Visibility & Hover Effect**

* When receiving "difficulty", the sprite shows itself and enters a forever loop:
* If touching the mouse pointer, its size increases to 60%, creating a hover effect.
* Otherwise, it remains at 50%.

**Game Start Behavior**

* When the green flag is clicked, the sprite moves to (148, 22) and hides itself, ensuring it appears only when needed.

**Click Behavior & Gameplay Setup**

* When clicked:
* Broadcasts "next", transitioning to the next setup step.
* Sets target\_score to 800, defining the difficulty level for Hard Mode.
* Hides itself, ensuring a clean interface.

**Transition Management**

* When receiving "next", the sprite hides, preventing interference during scene changes.

**Sprite 31: Register**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite serves as a menu button that allows users to access the registration process.

**Visibility & Hover Effect**

* When receiving "main", the sprite:
* Moves to (-101, -68).
* Shows itself for interaction.
* Enters a forever loop to detect mouse hover:
* If touching the mouse pointer, its size increases to 55%, creating a highlight effect.
* Otherwise, it remains at 50%.

**Click Behavior & Navigation**

* When clicked:
* Plays a button sound for feedback.
* Broadcasts "register", triggering the registration process.
* Hides itself, ensuring a clean interface.

**Game Transition Management**

* When receiving "login", the sprite hides, preventing it from appearing during login interactions.

**Sprite 32: login(2)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite serves as a menu button that allows users to access the login process.

**Visibility & Hover Effect**

* When receiving "main", the sprite:
* Moves to (101, -86).
* Shows itself for interaction.
* Enters a forever loop to detect mouse hover:
* If touching the mouse pointer, its size increases to 55%, creating a highlight effect.
* Otherwise, it remains at 50%.

**Click Behavior & Navigation**

* When clicked:
* Plays a button sound for feedback.
* Broadcasts "login", triggering the login process.
* Hides itself, ensuring a clean interface.

**Game Transition Management**

* When receiving "register", the sprite hides, preventing it from appearing during registration interactions.

**Sprite 33: loading**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite provides visual feedback when transitioning between game states.

**Game Start Behavior**

* When the green flag is clicked:
* The sprite sets its size to 75%.
* Moves to (0, -88) as its default position.
* Hides itself, ensuring it does not appear at the start.

**Loading Screen Animation**

* When receiving "loading":
* The sprite shows itself, appearing briefly to indicate a transition.
* Waits for 1 second, allowing players to notice the loading phase.
* Hides itself, completing the transition.

**Sprite 34: Achieve1**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages achievement tracking and visibility updates based on the player's progress.

**Unlocking Achievements**

* When receiving "Game Passed":
* If health = 3, it updates achieve1 by replacing the item at user\_value with "yes", marking the achievement as unlocked.

**Visibility & Display Mechanics**

* When receiving "achievement":
* If item user\_value of achieve1 is "yes", the sprite shows itself, indicating that the achievement is displayed to the player.

**Game Transition Handling**

* When receiving "After Login", the sprite hides, ensuring it does not interfere with login interactions.

**Game Start Behavior**

* When the green flag is clicked:
* Sets its size to 8%, likely adjusting its display for a minimal UI footprint.
* Moves to (197, 65), placing it in a designated section of the screen.
* Hides itself, ensuring it appears only when the achievement system is triggered.

**Sprite 35: achieve2**

**A screenshot of a video game

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages achievement tracking based on total score conditions.

**Unlocking Achievements**

* When receiving "achievement":
* If item user\_value of total\_score is greater than 2000, the sprite shows itself, indicating that the achievement is unlocked.

**Game Transition Handling**

* When receiving "After Login", the sprite hides, ensuring it does not interfere with login interactions.

**Game Start Behavior**

* When the green flag is clicked:
* Hides itself, preventing early visibility.
* Sets its size to 8%, ensuring it remains compact.
* Moves to (197, 0), placing it in a designated section.

**Sprite 36: achieve3**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages achievement tracking based on difficulty selection.

**Unlocking Achievements**

* When receiving "Game Passed":
* If target\_score = 800, it updates achieve2 by replacing the item at user\_value with "yes", marking the achievement as unlocked.

**Visibility & Display Mechanics**

* When receiving "achievement":
* If item user\_value of achieve2 is "yes", the sprite shows itself, indicating the achievement is unlocked.

**Game Transition Handling**

* When receiving "After Login", the sprite hides, ensuring it does not interfere with login interactions.

**Game Start Behavior**

* When the green flag is clicked:
* Hides itself for initial setup.
* Sets its size to 80%, adjusting its display for visibility.
* Moves to (107, -68), placing it in a designated achievement section.

**Sprite 37: achieve4**

**A screenshot of a game

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages achievement tracking based on total score conditions.

**Unlocking Achievements**

* When receiving "achievement":
* If item user\_value of total\_score is greater than 5000, the sprite shows itself, marking this achievement as unlocked.

**Game Transition Handling**

* When receiving "After Login", the sprite hides, ensuring it does not interfere with login interactions.

**Game Start Behavior**

* When the green flag is clicked:
* Hides itself, preventing early visibility.
* Sets its size to 8%, keeping it compact.
* Moves to (197, -133), placing it in the designated achievement section.

**Sprite 38: return**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite acts as a navigation button, allowing players to return to the home screen or login interface.

**Visibility & Hover Effect**

* When receiving "achievement" or "viewScore":
* The sprite shows itself.
* Enters a forever loop that detects mouse hover:
* If touching the mouse pointer, its size increases to 60%, creating a highlight effect.
* Otherwise, it remains at 50%.

**Click Behavior & Navigation**

* When clicked:
* Plays a button sound for feedback.
* Broadcasts "After Login", returning the player to the main menu.
* Hides itself to ensure a clean interface.

**Game Transition Handling**

* When receiving "show", the sprite shows itself.
* When receiving "hide", the sprite hides itself, allowing smooth transitions.
* When receiving "clicked", the sprite moves to (-202, 135) and hides itself, likely resetting its position for future interactions.

**Sprite 39: score**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite enables users to search for a student’s total score by entering a name.

**Score Lookup Process**

* When receiving "viewScore":
* Broadcasts "hide" to clear unnecessary UI elements.
* Asks the player to enter a student’s name, providing the option to type "exit" to quit.
* The script repeats until "exit" is entered":
* If the username exists in username\_list:
  + Checks if the role associated with the username is "Student":
  + If valid, shows the total score for 2 seconds.
  + Otherwise, displays "Username invalid" for 2 seconds.
* If the username is invalid, prompts the user again until "exit" is entered.

**Game Transition Handling**

* When receiving "clicked":
* Sets ghost effect to 100%, likely making the sprite invisible.
* The sprite hides itself, ensuring it does not interfere with normal gameplay.

**Sprite 40: stage**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

This sprite manages game state transitions, soundtrack, and scene changes.

**Game State Initialization**

* When receiving "next":
* Sets status to 1, initializing the game state.
* Resets key variables:
  + health to 3
  + no\_of\_keys to 0
  + score to 0
* Shows the score variable, making it visible.

**Stage Selection & Scene Changes**

* When receiving "next":
  + Switches costumes based on current\_stage:
  + Stage 1 → level1 costume
  + Stage 2 → level2 costume
  + Stage 3 → level3 costume
* Other costume transitions based on game events:
* "pause" → Switches to paused costume.
* "Start Game" → Switches to avatar costume.
* "main" → Switches to Login costume.
* "After Login" → Switches to mainpage costume.
* "Game Passed" → Switches to win costume.
* "difficulty" → Switches to difficulty selection costume.
* "achievement" → Switches to achievement costume.
* "viewScore" → Switches to viewScore costume.
* "Game Over" → Switches to lose costume.
* "Choose Level" → Switches to level selection costume.

**Soundtrack Management**

* When the green flag is clicked:
* Enters a forever loop, continuously playing the "calm\_before\_a\_storm" soundtrack.

**Game Start & Transition Handling**

* When the green flag is clicked:
* Hides the score variable, ensuring it does not display too early.
* Broadcasts "main", setting up the initial screen.

# Conclusion

In a nutshell, we have created a gamified e-learning platform using Snap, a visual programming language. This platform is useful to help students to learn some academic knowledge with fun. This is because they would not feel bored while doing the quiz.

We assume that the university academic performance would be improved since this e-learning platform is applied among students. Besides, it also helps lecturers in universities to ease their burden. In this situation, we have successfully increased the effectiveness of teaching and learning.

However, there are still some limitations in our e-learning platform. At this point, we only have web-based system for the game. This means there is no mobile phone application version for it. Besides, it also could not run without network. Apart from that, there are also some limitations in the platform itself. At this point, we still only have one language, English. We also do not have the record for the games played by students.

With these limitations, we will manage to make some enhancements in the future. We will provide function for players to change language to at least 5 or 6 languages. Then, we will have a record system to record the games details, including student’s name, date and score. In addition, we will try to implement more avatar for players to choose.

# References

Alshamrani & Bahattab. (2015). A Comparison between Three SDLC Models Waterfall Model, Spiral Model and Incremental Model/ Iterative Model. IJCSI International Journal of Computer Science Issues, 106-111. Retrieved from <https://d1wqtxts1xzle7.cloudfront.net/36637147/SDLC-libre.pdf?1423940228=&response-contentdisposition=inline%3B+filename%3DA_Comparison_Between_Three_SDLC_Models_W.pdf&Expires=1733772954&Signature=KYRGLCNcVBdkGKvrm99rHLMmZYDX90qUnZc9CaZV8WnCi8H018xVSQro0>

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: blocks and beyond. *Communications of the ACM*, *60*(6), 72-80.

Harvey, B., & Mönig, J. (2010). Bringing “no ceiling” to scratch: Can one language serve kids and computer scientists. *Proc. Constructionism*, *1*

*Ismail, A., Yatim, M. H. M., Sahabudin, N. A., & Zain, N. Z. M. (2016). Keupayaan Murid Sekolah Rendah Mempelajari dan Menerokai Bahasa Pengaturcaraan Visual: Capability of Primary School Pupils in Learning and Exploring Visual Programing Language. Journal of ICT in Education, 3, 89-97.*

Kraleva, R., Kralev, V., & Kostadinova, D. (2019). A Methodology for the Analysis of Block-Based Programming Languages Appropriate for Children. *J. Comput. Sci. Eng.*, *13*(1), 1-10.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010, August). Learning computer science concepts with scratch. In *Proceedings of the Sixth international workshop on Computing education research* (pp. 69-76).

Risener. (2022). A Study of Software Development Methodologies. *ScholarWorks@UARK*, 1-39. Retrieved from <https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=1105&context=csceuht>

SEGi College Malaysia. (2019). SCKL is the first higher education institution to adopt CodeSnaps in Southeast Asia. Retrieved from <https://www.segi.edu.my/sckl-is-the-first-higher-education-institution-to-adopt-codesnaps-in-southeast-asia/>

Weintrop, D., & Wilensky, U. (2015, June). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children* (pp. 199-208).