# BUGLUAY POP QUIZ GAME

BY

| | | |
|---|---|---|
| MISS RUNCHANA | SEESUNG | 6388052 |
| MISS NANTANAT | SUDLAR | 6388106 |
| MR. SIRANUT | AKARAWUTHI | 6388117 |

SECTION 2

ADVISOR
SANIT SANGHLAO

A Project Submitted in Partial Fulfillment of
the Requirements for

ITCS424 Wireless and Mobile Computing

Faculty of Information and Communication Technology
Mahidol University
2/2022

# ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this academic paper on the BuGluay Pop Quiz Game. Furthermore, we would like to express our appreciation to the institution or organization that provided us with the necessary resources to carry out this research.

Miss Runchana Seesung<br>
Miss Nantanut Sudlar<br>
Mr. Siranut Akarawuthi

**BUGLUAY POP QUIZ GAME**

| | |
|---|---|
| MISS RUNCHANA SEESUNG | 6388052 ITCS/B |
| MISS NANTANAT SUDLAR | 6388106 ITCS/B |
| MR. SIRANUT AKARAWUTHI | 6388117 ITCS/B |

B.Sc. (INFORMATION AND COMMUNICATION TECHNOLOGY)

PROJECT ADVISOR: SANIT SANGHLAO

## ABSTRACT

THE BUGLUAY POPQUIZ GAME IS A PART OF THE WIRELESS AND MOBILE COMPUTING COURSE. THE OBJECTIVE OF THIS PROJECT WAS TO PROVIDE FREQUENT MEMORY REFRESHES AND ENCOURAGE PEOPLE TO ACQUIRE KNOWLEDGE. THE SCOPE WILL COVER PROVIDING A LIST OF QUESTIONS FOR PLAYING AND COMPETING IN A GAME WITH OTHER PEOPLE AT THE SAME TIME.

# CONTENTS

Page

# LIST OF TABLES

Page

# LIST OF FIGURES

Page

# CHAPTER 1
# INTRODUCTION

In this chapter covers six topics: motivation, problem statement, objectives of the project, scope of the project, expected benefits, and organization of the document. These topics are written to help the audience understand the goal of the BuGluay Pop Quiz Game, particularly the issues that must be resolved and the processes involved. In general, pop quiz is a game or competition in which people answer questions about pop music, or a television show that includes such a game. Pop quizzes can provide frequent memory refreshes and encourage people to acquire knowledge. [1] Therefore, this project aims to allow two users to play a game with two mobile phones connected.

## 1.1 Motivation

The COVID-19 pandemic affects people in every sector, including the attention span of students. Attention span is essential for effective learning because it is a key to a child's learning process. [2] Online learning decreases the attention span of many students, so it is more difficult for them to focus on the lessons. In order to make students become interested in classes and gain acknowledgement, learning games such as missing word, quiz, crossword, matchup, etc. are perfect options to be used for converting dull tasks into interesting activities. Exciting activities can get people interested and help them concentrate. In the meantime, quiz games can impart knowledge and act as stress relievers or even icebreakers at schools, offices, or homes.

## 1.2 Problem Statement

A pandemic increases stress and impairs one's ability to focus and learn something new at the same time. Consequently, there is less inspiration for people to build general knowledge these days.

## 1.3 Objectives of the Project

- To help students improve quality of long-term learning
- To build general knowledge
- To entertain people in every age range based on interests

## 1.4 Scope of the Project

Allowing players to join the lobby and compete in the BuGluay Pop Quiz Game. People participate in the game by selecting the correct answers within the time limit and are rewarded with points for each correct answer. At the end of the game, which will be after players answer a specific number of questions, there will be a result page that calculates the score each player gets. One correct answer gives 100 points.

## 1.5 Expected Benefits

- People can enjoy the multiplayer quiz game.
- People gain some general knowledge, such as geography, from playing the game.

## 1.6 Organization of the Document

This document consists of 6 chapters including:

1. Introduction – The first chapter in the project that includes motivation, problem statement, object of the project, scope of the project, and expected benefit.
2. Background – The second chapter in the document that clarifies the description of BuGluay project.
3. Analysis and Design – The third chapter includes the system architecture overview, the system structure chart, the process analysis and design, the data flow diagram, the process description, the data stores, the data element, the database analysis and design, the ER-diagram, etc.
4. Implementation – The hardware and system environment are covered in detail in the fourth chapter, which also demonstrates the implementation strategy.

5. Testing and Evaluation – The fifth chapter covers unit tests, system integration tests, and test scenarios of the project.
6. Conclusion – The final chapter is a summary that discusses the project's developer and user benefits, problems and limitations, and future work.

# CHAPTER 2
# BACKGROUND

This chapter mainly focuses on the literature review. Readers will learn more about the project-related facts that we have collected and analyzed.

## 2.1 Literature Review

Creating a multiplayer trivia game with Flutter can be an interesting project that will provide participants with hours of amusement. Because of Flutter's adaptability, it is possible to create a well-designed and user-friendly interface that engages players. To ensure that the game is both interesting and challenging, it should include areas such as science, history, literature, pop culture, and others. It is also important to correctly select the questions to ensure accuracy, up-to-date material, and a suitable level of difficulty. A scalable architecture is required to handle many participants while maintaining performance. Overall, this project has a lot of potential to be an enjoyable and educational experience for trivia fans.

# CHAPTER 3
# ANALYSIS AND DESIGN

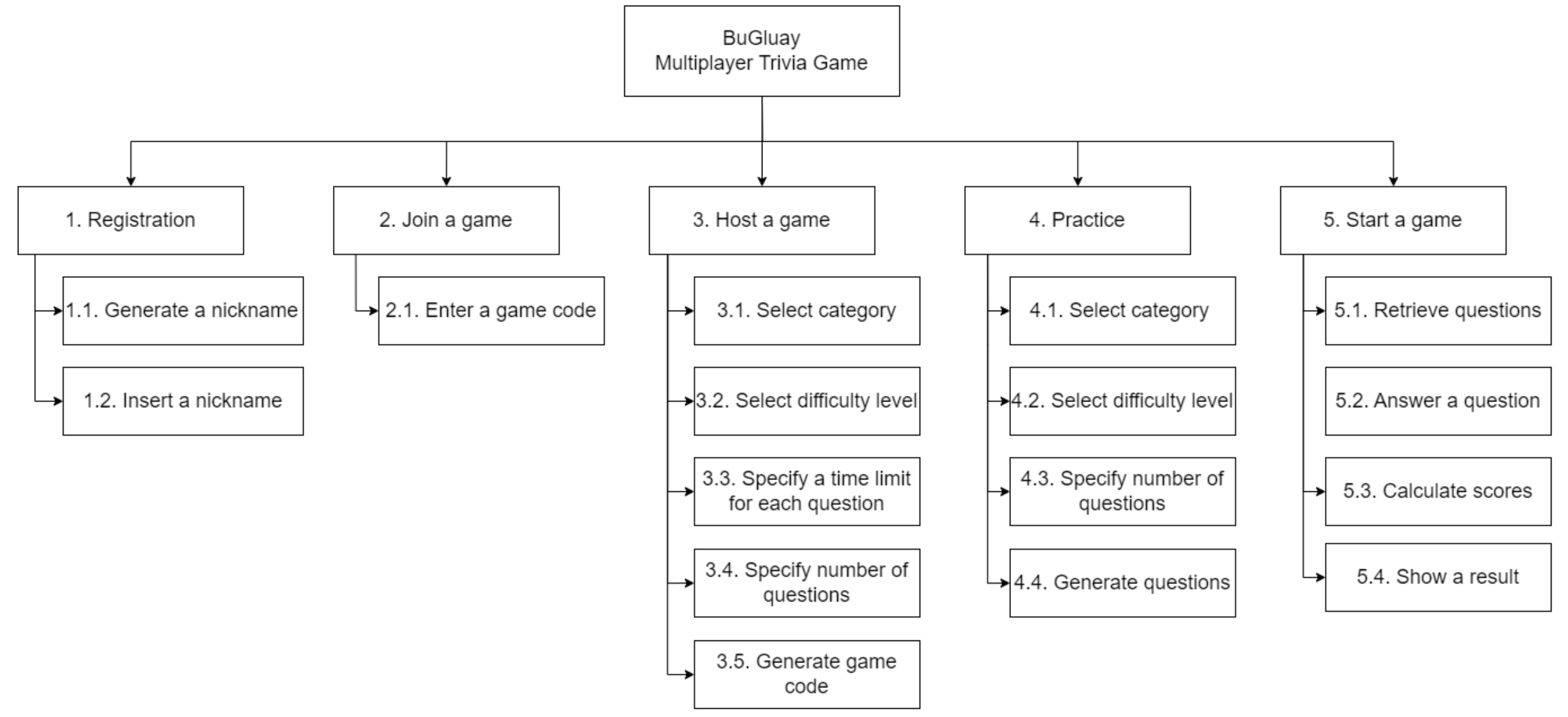
This chapter includes our application's analysis and design. The reader will learn more about our system architecture, which includes a system structure chart, a data flow diagram, and an interface design.

## 3.1 System Architecture Overview

A Pop Quiz Game designed with Flutter will have three major components: the Client, Server, and Database. The Client component oversees rendering the user interface, receiving user input, and making requests to the Server. The Server component will take client requests, respond with necessary data, and manage game logic. The Database component will hold all game data, such as questions, answers, user profiles, and game history. To ensure the game's scalability, a load balancer can be used to distribute incoming requests evenly among servers. A RESTful API will handle communication between the Client, Server, and Database, and a strong authentication system will be created to prevent unwanted access to the game and user data.

## 3.2 System Structure Chart

The key components of the multiplayer trivia game are shown in this structure chart, which includes the registration, joining, hosting, and practicing a game together with starting a game. Developers can use this structure chart to understand the general system architecture as well as the interactions between the game's various components and sub-components.



| | |
|---|---|
| **Project:** BuGluay Pop Quiz Game<br>**System:** BuGluay | **Major Advisor:** Aj. Sanit Sanghlao |
| **Description:** This figure represents the system architecture of a multiplayer trivia game, including the main components and sub-components for creating, joining, practicing, and playing games. | |

**Figure 1: System Structure Chart of BuGluay Pop Quiz Game Application**

The detailed description of each subsystem is shown below:

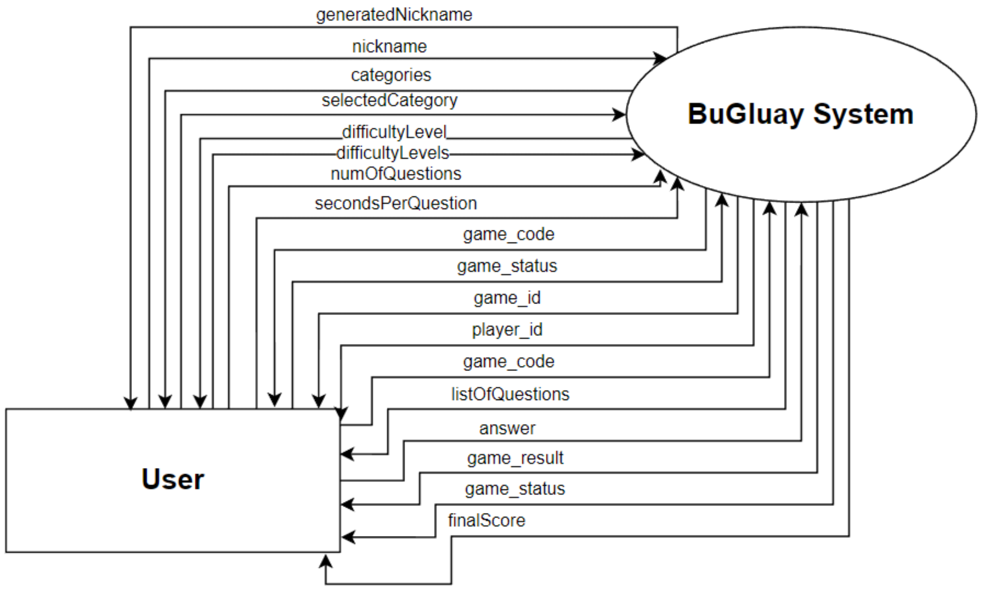
- Registration – The first step of using BuGluay Pop Quiz Game is to insert a nickname for playing a game.
    - Generate a nickname – This section is to generate random nickname for users as the beginning.
    - Insert a nickname – If the users would like to use their own nickname, this section also allows users to input nickname.
- Join a game – This section is for joining a created room with another player.
    - Enter a game code – Users must enter a game code to join a room.
- Host a game – This section allows players hosting a game to play with other players.
    - Select category – Users who host a game must select the category of questions.
    - Select difficulty level – Users who host a game must select the difficulty level of questions.
    - Specify number of questions – Users who host a game must specify the number of questions with an integer number.
    - Specify a time limit for each question – Users who host a game must specify a time limit in a second unit with an integer number.
    - Generate game code – The system automatically returns game code for other users to join.
- Practice – This section lets users practice playing or answering questions before starting an actual game.
    - Select category – Users who play a game in single player mode must select the category of questions.
    - Select difficulty level – Users who play a game in single player mode must select the difficulty level of questions.
    - Specify number of questions – Users who play a game in single player mode must specify the number of questions with an integer number.
- Start a game – This section lets players answer the same questions in the same room they joined.

- Retrieve questions – The system retrieves questions based on the category and difficulty level.
- Answer a question – Users can select one answer for each question.
- Calculate scores – After answering every question in a game, the system will take a while to calculate scores.
- Show a result – Once the system finishes calculating scores, users will be able to see a result page with rankings and scores of each player.

## 3.3 Process Analysis and Design

### 3.3.1 Data Flow Diagram

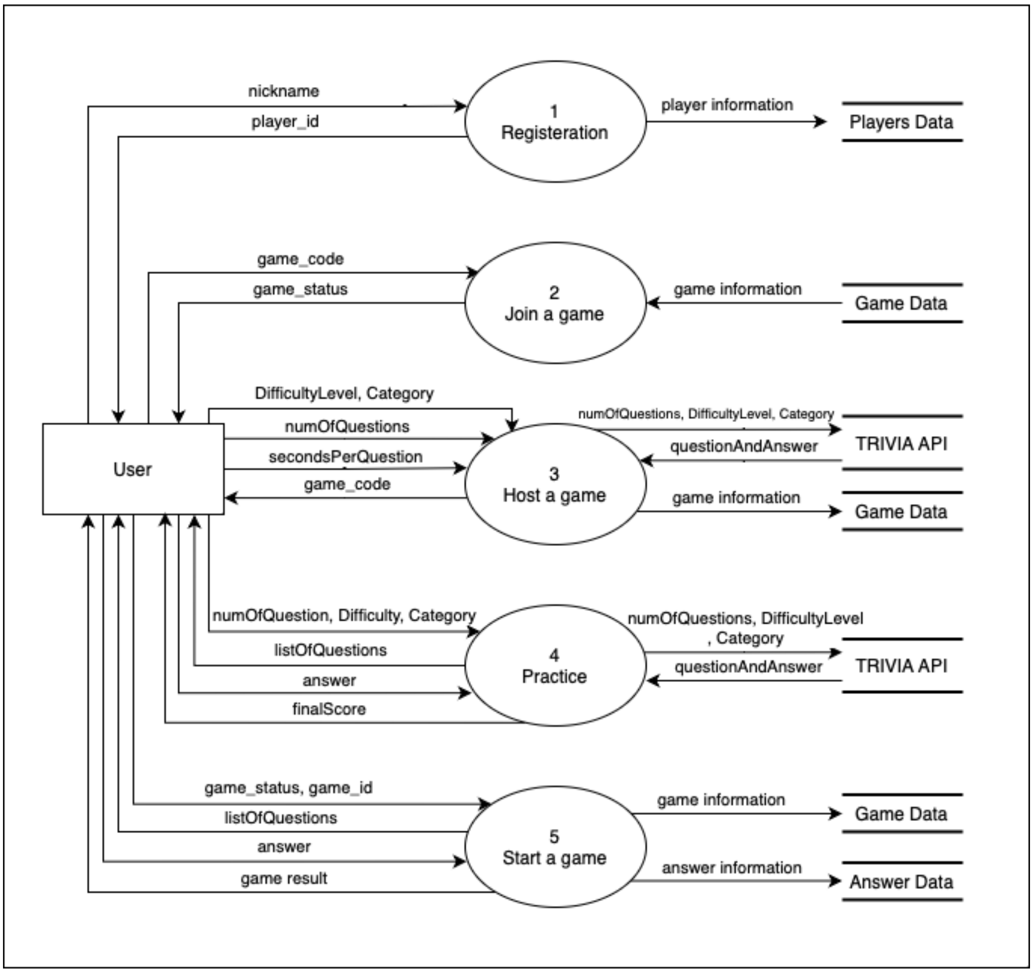
A visual representation of the data flow between the actors and the BuGluay system is provided by the data flow diagram shown below.



| | |
|---|---|
| **Project:** BuGluay Pop Quiz Game<br>**System:** BuGluay | **Major Advisor:** Aj. Sanit Sanghlao |

**Description:** This figure illustrates how users communicate with the system and what data is inbound and outbound respectively.

**Figure 2: Context diagram of BuGluay Pop Quiz Game**



| | |
|---|---|
| **Project:** BuGluay Pop Quiz Game<br>**System:** BuGluay | **Major Advisor:** Aj. Sanit Sanghlao |
| **Description:** This figure illustrates how users communicate with the system and what data is inbound and outbound respectively. | |

**Figure 3: Data flow diagram Lv. 1 of BuGluay Pop Quiz Game**

### 3.3.2 Data Dictionary

A data dictionary is a way to document and describe Processes, Data Stores, and Data Elements (Data Flow) that occur in a Data Flow Diagram (DFD). It is composed of 3 parts as shown below.

- Process Descriptions
- Data Stores
- Data Elements

#### 3.3.2.1 Process Description

This section will provide a detailed description of each process that exists in this system. It includes Inbound Data, Outbound Data, and Logic Summary.

**Table 3.1: List of all Processes**

| No. | Process | Name | Description |
|---|---|---|---|
| 1 | 1 | Registration | Process for registration |
| 2 | 1.1. | Generate a nickname | Process for nickname generator |
| 3 | 1.2. | Insert a nickname | Process for inserting a nickname |
| 4 | 2 | Join a game | Process for joing an existing game room |
| 5 | 2.1. | Enter a game code | Process for entering a game code |
| 6 | 3 | Host a game | Process for creating a game room |
| 7 | 3.1. | Select category | Process for selecting a category |
| 8 | 3.2. | Select difficulty level | Process for selecting difficulty level |
| 9 | 3.3. | Specify number of questions | Process for specifying number of questions |
| 10 | 3.4. | Specify a time limit for each question | Process for specifying a time limit for each question |
| 11 | 3.5. | Generate game code | Process for displaying game code |
| 12 | 4 | Practice | Process for practicing |
| 13 | 4.1. | Select category | Process for selecting a category |
| 14 | 4.2. | Select difficulty level | Process for selecting difficulty level |

| No. | Process | Name | Description |
|---|---|---|---|
| 15 | 4.3. | Specify number of questions | Process for specifying number of questions |
| 16 | 4.4. | Generate questions | Process for retrieving questions |
| 17 | 5. | Start a game | Process for starting a game |
| 18 | 5.1. | Retrieve questions | Process for retrieving questions |
| 19 | 5.2. | Answer a question | Process for answering a question |
| 20 | 5.3. | Calculate scores | Process for calculating scores |
| 21 | 5.4. | Show a result | Process for showing a result |

**Table 3.3.2: Process Description of *Registration***

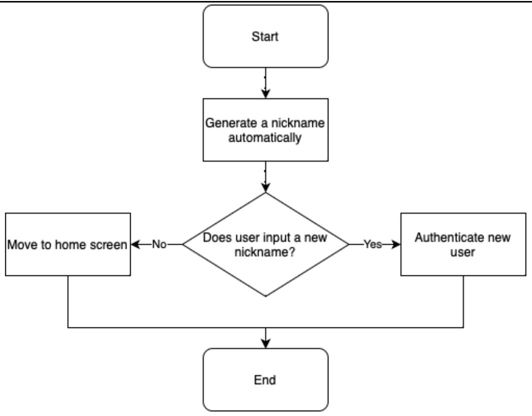
| | |
|---|---|
| **Process Name** | 1- Registration |
| **Description** | Process for registration |
| **Inbound data** | • Nickname |
| **Outbound Data** | • GeneratedNickname |
| **Logic Summary** |   Start<br>Generate a nickname automatically<br>Does user input a new nickname?<br>No<br>Move to home screen<br>Yes<br>Authenticate new user<br>End |

**Table 3.3: Process Description of *Joining a game***

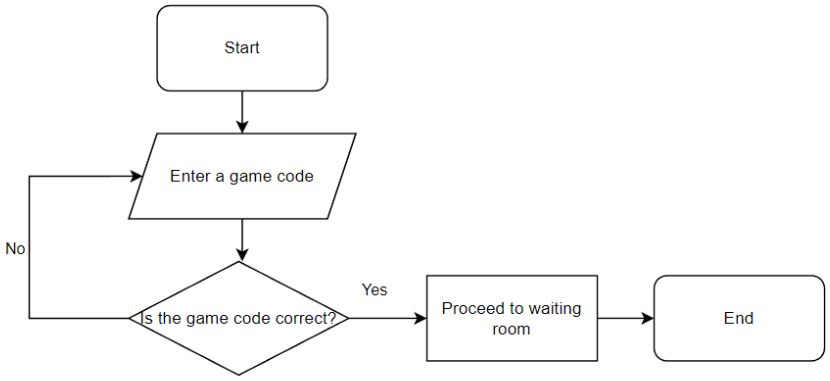
| **Process Name** | 2- Join a game |
|---|---|
| **Description** | Process for joing an existing game room |
| **Inbound data** | • Game_code |
| **Outbound Data** | • Game_status |
| **Logic Summary** |   Start<br>Enter a game code<br>No<br>Is the game code correct?<br>Yes<br>Proceed to waiting room<br>End |

**Table 3.3.4: Process Description of *Hosting a game***

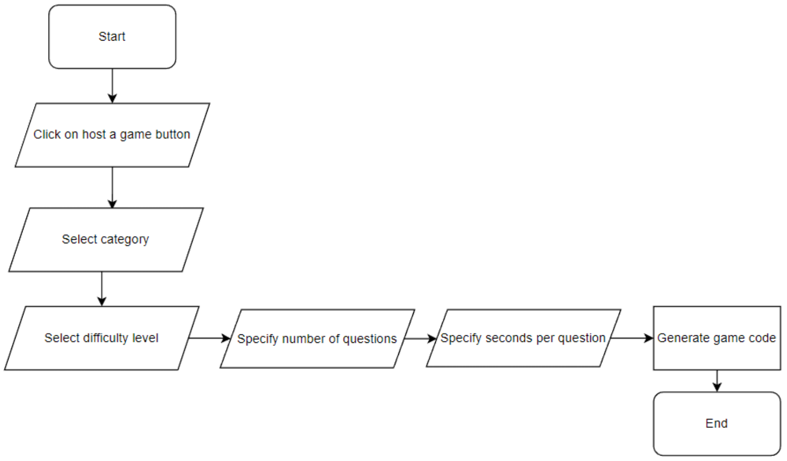
| **Process Name** | 3 – Host a game |
|---|---|
| **Description** | Process for creating a game room |
| **Inbound data** | • selectedCategory<br>• difficultyLevels<br>• numOfQuestions<br>• secondsPerQuestion |
| **Outbound Data** | • game_code |
| **Logic Summary** |   Start<br>Click on host a game button<br>Select category<br>Select difficulty level<br>Specify number of questions<br>Specify seconds per question<br>Generate game code<br>End |

**Table 3.3.5: Process Description of *Practicing***

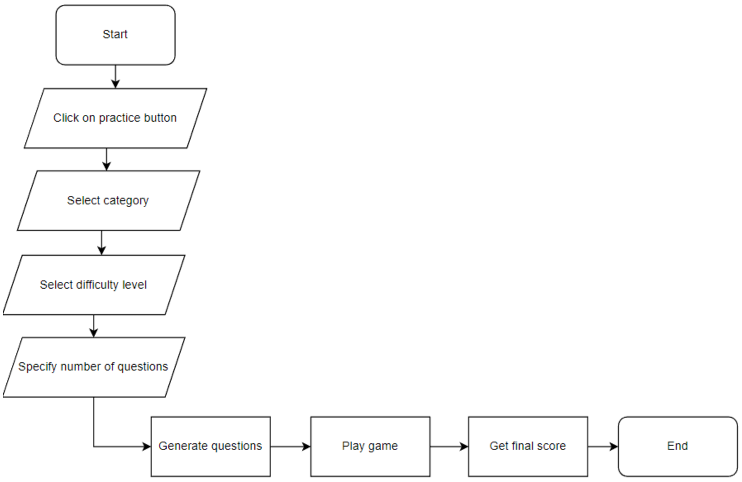
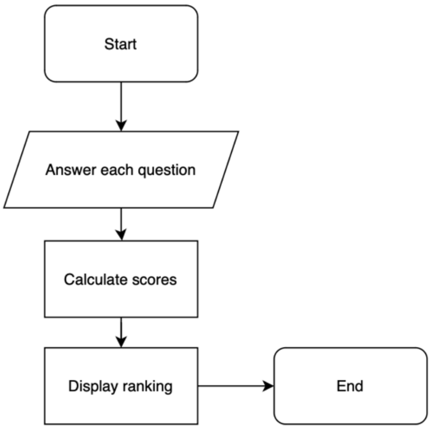
| | |
|---|---|
| **Process Name** | 4 - Practice |
| **Description** | Process for practicing |
| **Inbound data** | • selectedCategory<br>• difficultyLevels<br>• numOfQuestions |
| **Outbound Data** | • listOfQuestions<br>• finalScore |
| **Logic Summary** |   Start<br>Click on practice button<br>Select category<br>Select difficulty level<br>Specify number of questions<br>Generate questions<br>Play game<br>Get final score<br>End |

**Table 3.3.6: Process Description of *Starting a game***

| | |
|---|---|
| **Process Name** | 5 – Start a game |
| **Description** | Process for starting a game |
| **Inbound data** | • Answer |
| **Outbound Data** | • listOfQuestions<br>• finalScores<br>• Game_result |
| **Logic Summary** |  Start<br>Answer each question<br>Calculate scores<br>Display ranking<br>End |

### 3.3.2.2 Data Stores

This section describes the data stores that exist in the data flow diagram and consists of the Data Store Name, Description, Inbound Data, and Outbound Data.

**Table 3.3.7: List of all Data Stores**

| No. | Data Store | Name | Description |
|---|---|---|---|
| 1 | Game_DB | Game Database | Collect information about each game, players, answers of each player, and questions for each game. |

**Table 3.3.8: Data Store Description of *Game_DB***

| | |
|---|---|
| **Data Store Name** | Game_DB - Game Database |
| **Description** | Collect information about each game, players, answers of each player, and questions for each game. |

| | |
|---|---|
| **Inbound data** | • second_per_question<br>• status<br>• nickname<br>• answer<br>• answer_time |
| **Outbound Data** | • question<br>• score<br>• correct_answer<br>• wrong_answer_1<br>• wrong_answer_2<br>• wrong_answer_3 |

#### 3.3.2.3 Data Element

This section describes the data elements or data flows that exist in this system. The table below contains the list of all data elements belonging to their data element name, starting process/source/data store, and ending process/source/data store.

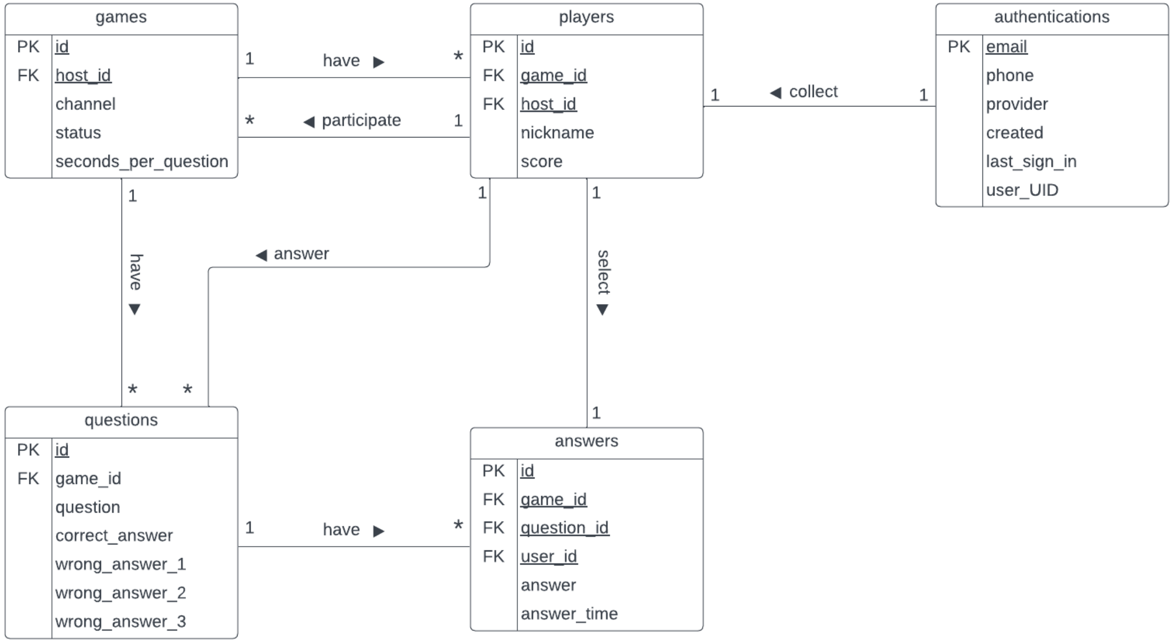
**Table 3.3.9: List of All Data Elements**

| SEQ | Data Element Name | From Process/Source/Data Store | To Process/Source/Data Store |
|---|---|---|---|
| 1 | Nickname | Registration | Start a game |
| 2 | Player_id | Registration | Start a game |
| 3 | Game_code | Join a game / Host a game | Start a game |
| 4 | Game_status | Join a game | Start a game |
| 5 | Questions | Host a game | Start a game |
| 6 | Number of questions | Host a game | Start a game |
| 7 | second_per_question | Host a game | Start a game |
| 8 | Number of questions | Practice | Start a game |
| 9 | Answer | Start a game | Start a game |
| 10 | Score | Start a game | Start a game |

## 3.4 Database Analysis and Design

In this topic, we will cover the topic of database analysis and design such as ER-Diagram, Relational Diagram, and Interface design. We have also explained the steps of using our applications.

### 3.4.1 ER-Diagram



| | |
|---|---|
| **Project:** Bugluay Pop Quiz Game<br>**System:** Buglauy | **Major Advisor:** Aj. Sanit Sanghlao |
| **Description:** This figure is a ER-Diagram that shows the relationship between each table in Bugluay's database | |

**Figure 4: ER-Diagram**

### 3.4.2 Relational Schema

This section describes the attributes of the tables in the database. The attribute notation is shown below.

- **Attributes** – which are bold and underlined are the Primary Keys
- *Attributes* – which are Italic are the Foreign Keys
- ***Attributes*** – which are bold, italic and underlined are both Primary Keys and Foreign Keys

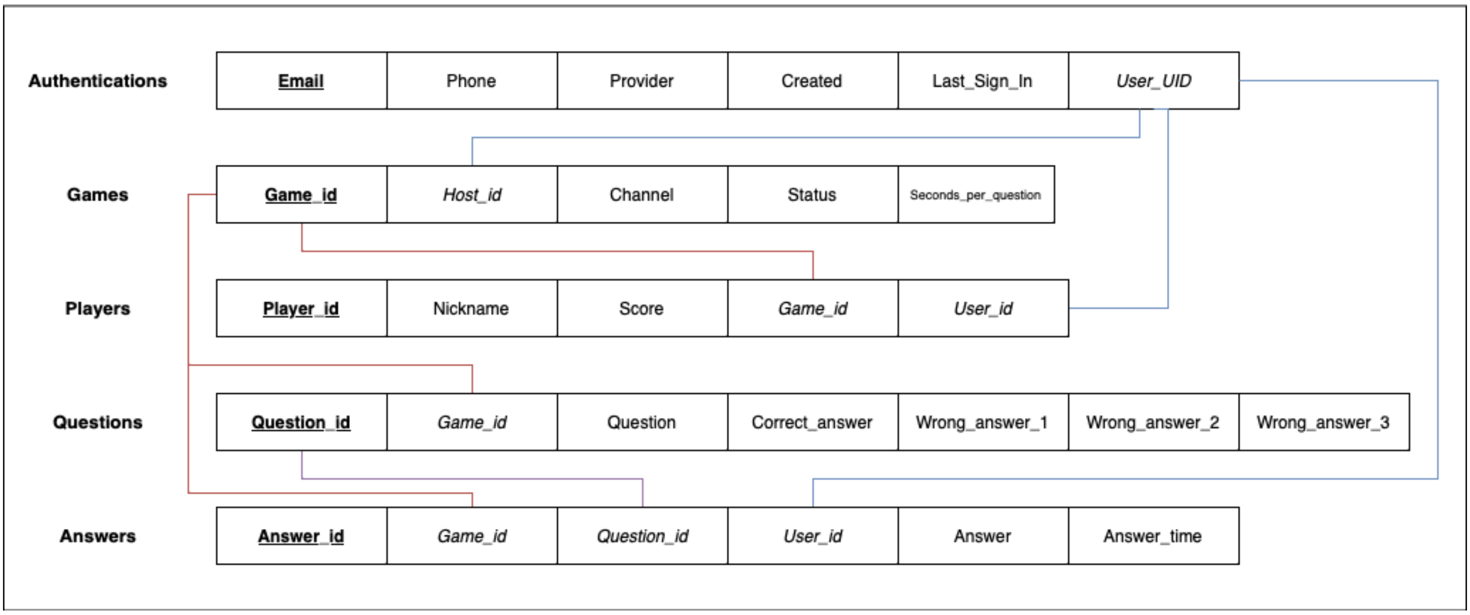
Tables in this system can be divided into 3 groups as follows:

- Master File Table
- Base File Table
- Transaction File Table

**Table 3.3.10: List of all Tables in Our System Database**

| Table# | Table Name | Table Type | Description |
|---|---|---|---|
| 1 | Authentications | Base | This table stores the authorized user from registration process. |
| 2 | Games | Master | This table stores the game information. |
| 3 | Players | Master | This table stores registors player information. |
| 4 | Questions | Master | This table stores retrieved questions from Trivia API. |
| 5 | Answers | Master | This table stores the answer of the user. |

1. Relational Schema of Master File Tables
   - Games
   - Players
   - Questions
   - Answers
2. Relational Schema of Base File Tables
   - Authentications



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Authentications** | **Email** | Phone | Provider | Created | Last_Sign_In | *User_UID* | |
| **Games** | **Game_id** | *Host_id* | Channel | Status | Seconds_per_question | | |
| **Players** | **Player_id** | Nickname | Score | *Game_id* | *User_id* | | |
| **Questions** | **Question_id** | *Game_id* | Question | Correct_answer | Wrong_answer_1 | Wrong_answer_2 | Wrong_answer_3 |
| **Answers** | **Answer_id** | *Game_id* | *Question_id* | *User_id* | Answer | Answer_time | |

### 3.4.3. File Structure

This section shows the details of each file component including field name, field description, field data type, field length, null value, primary key and foreign key.

**Table 3.3.11: File Structure of *Authentications***

| | | |
|---|---|---|
| **Table Name:** | Authentications | **Table #1** |
| **Table Type:** | Base | |
| **Description:** | This is the file structure of 'authentications' that collects authorized players information | |

| Field Name | Type | Length | Description | Key | Reference | Null |
|---|---|---|---|---|---|---|
| Email | text | 42 | Store an email of each user during authentication. | PK | - | - |
| Phone | text | 10 | Store a phone number of each user during authentication. | - | - | - |
| Provider | text | 5 | Store a type of information that was used for authentication | - | - | - |
| Created | datetime | 8 | Store a datetime when the authorization was created. | - | - | - |
| Last_Sign_In | datetime | 8 | Store a datetime of the last time when this authorized user signed in. | - | - | - |
| User_UID | uuid | 16 | Store a unique uid of the user. | - | - | - |
| | | | | | | |
| | | | | | | |
| | Total | 89 | Bytes | | | |

**Table 3.3.12: File Structure of *Games***

| | | | |
|---|---|---|---|
| **Table Name:** | Games | | **Table #2** |
| **Table Type:** | Master | | |
| **Description:** | This is the file structure of 'games' that collects game information | | |

| Field Name | Type | Length | Description | Key | Reference | Null |
|---|---|---|---|---|---|---|
| Game_id | int8 | 8 | Store the ID of each game. | PK | - | - |
| Host_id | uuid | 16 | Store the ID of the host who created the game. | FK | Auth (User_UID) | - |
| Channel | uuid | 16 | Store a unique channel of the game room for identification purpose. | - | - | - |
| Status | game_status | 15 | Store the status of current game. | - | - | - |
| Seconds_per_question | int4 | 4 | Store the time for answering of each question in this game. | - | - | - |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | Total | 59 | Bytes | | | |

**Table 3.3.13: File Structure of *Players***

| | | |
|---|---|---|
| **Table Name:** | Players | **Table#3** |
| **Table Type:** | Master | |
| **Description:** | This is the file structure of ‘players’ that collects every registered player information. | |

| Field Name | Type | Length | Description | Key | Reference | Null |
|---|---|---|---|---|---|---|
| Player_id | int8 | 8 | Store a unique id of the player. | PK | - | - |
| Nickname | text | 20 | Store a nickname of the player. | - | - | - |
| Score | int4 | 4 | Store personal scores in one game room. | - | - | - |
| Game_id | int8 | 8 | Store the ID of each game. | FK | Games (Game_id) | - |
| User_id | uuid | 16 | Store the ID of the authorized user. | FK | Auth (User_UID) | - |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | Total | 56 | Bytes | | | |

**Table 3.3.14: File Structure of *Questions***

| | | |
|---|---|---|
| **Table Name:** | Questions | **Table#4** |
| **Table Type:** | Master | |
| **Description:** | This is the file structure of ‘questions’ that collects retrieved questions from Trivia API. | |

| Field Name | Type | Length | Description | Key | Reference | Null |
|---|---|---|---|---|---|---|
| Question_id | int8 | 8 | Store the ID of the question. | PK | - | - |
| Game_id | int8 | 8 | Store the ID of each game. | FK | Games (Game_id) | - |
| Question | text | 120 | Store a question that was generated in each game. | - | - | - |
| Correct_answer | text | 120 | Store the correct answer to that question. | - | - | - |
| Wrong_answer_1 | text | 120 | Store the wrong answer. | - | - | - |
| Wrong_answer_2 | text | 120 | Store the wrong answer. | - | - | - |
| Wrong_answer_3 | text | 120 | Store the wrong answer. | - | - | - |
| | | | | | | |
| | Total | 616 | Bytes | | | |

Table 3.3.15: File Structure of *Answers*

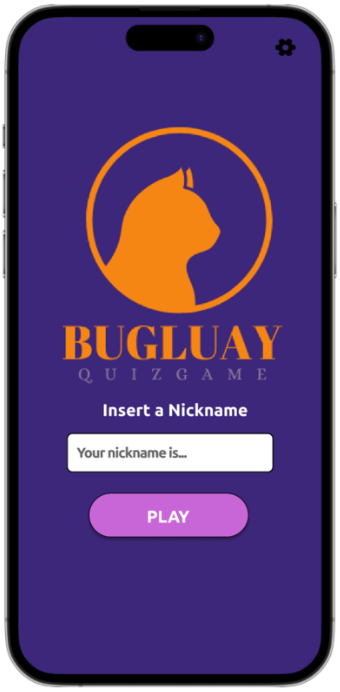
| | | |
|---|---|---|
| **Table Name:** | Answers | **Table#5** |
| **Table Type:** | Master | |
| **Description:** | This is the file structure of ‘answers’ that collects answers to each question from players. | |

| Field Name | Type | Length | Description | Key | Reference | Null |
|---|---|---|---|---|---|---|
| Answer_id | int8 | 8 | Store the ID of each answer. | PK | - | - |
| Game_id | int8 | 8 | Store the ID of each game. | FK | Games (Game_id) | - |
| Question_id | int8 | 8 | Store the ID of each question. | FK | Question (Question_id) | - |
| User_id | uuid | 16 | Store the ID of the authorized user. | FK | Auth (User_UID) | - |
| Answer | text | 20 | Store the answer. | - | - | - |
| Asnwer_time | datetime | 8 | Store the time that this answer is selected. | - | - | - |
| | | | | | | |
| | | | | | | |
| | Total | 68 | Bytes | | | |

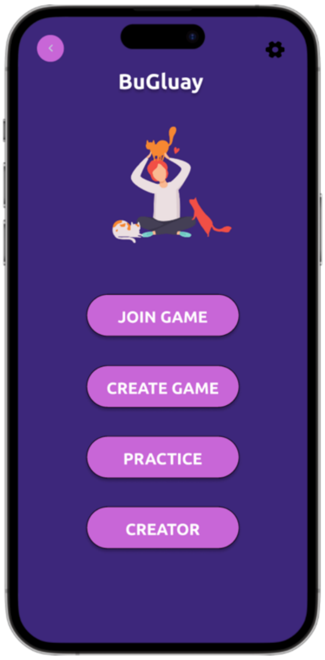
## 3.5 I/O Design

This section explains the design of the Input and Output User Interface. The section consists of two parts, the interface design and the transition diagram showing transition through the system.

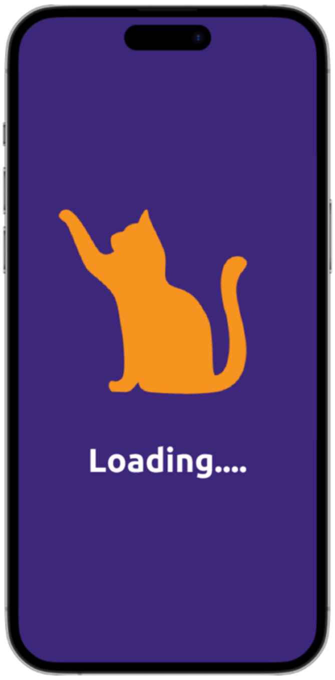
### 3.5.1 Interface Design



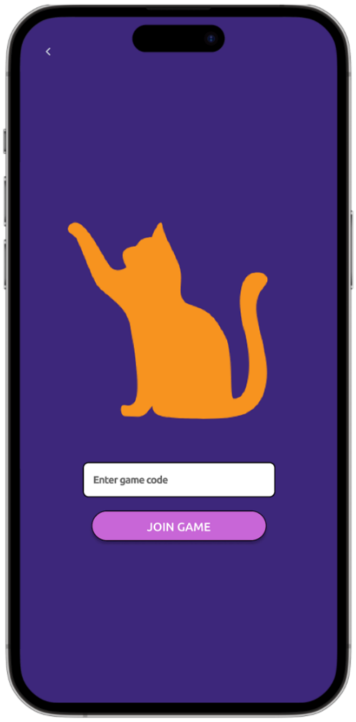
This page is the first page of BuGluay Pop Quiz Game. Users have to insert Nickname to start playing a game in the application.



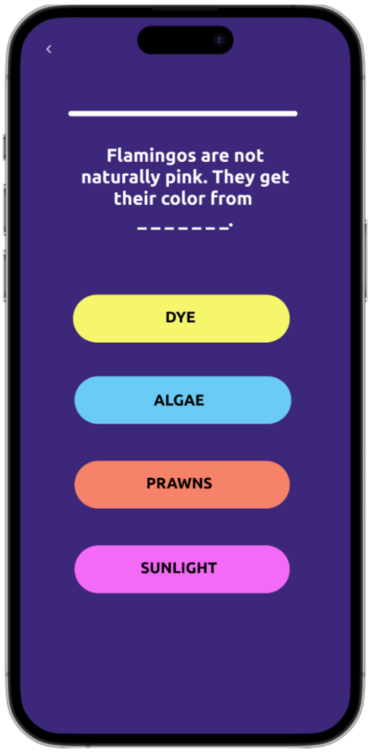
After finished insert a nickname, users will be linked to this page automatically. The page will show four options that users have to select which are join a game, create a game, practice, and creator.



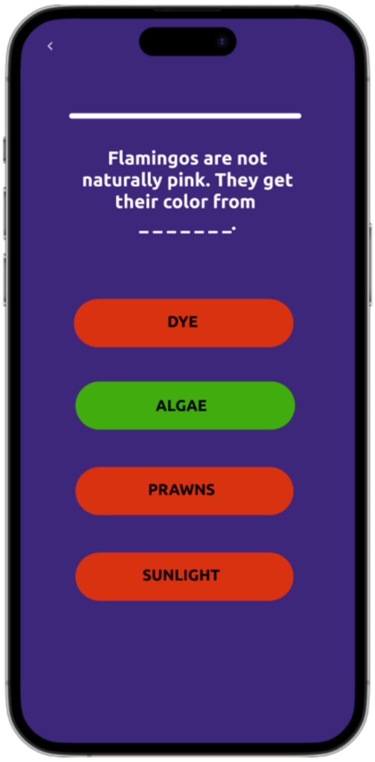
When the user finished selecting the option that they want, this loading page will be shown to the user for a second.



This page will be shown for the user who selects the join game option which has entered game code button to let users enter the game code to start a quiz game.

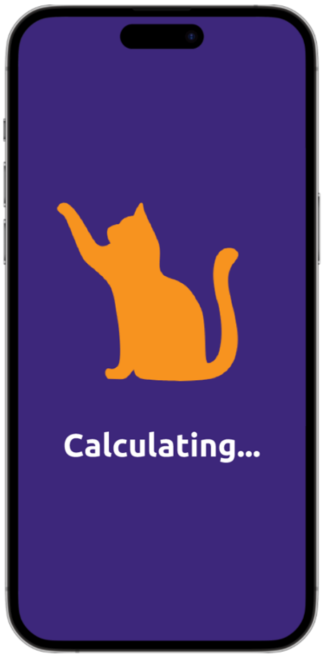

Flamingos are not naturally pink. They get their color from _______.

This page is the example question that will appear in BuGluay Pop Quiz game. It will have a time limit in the bar at the top of the page and four choices per question.

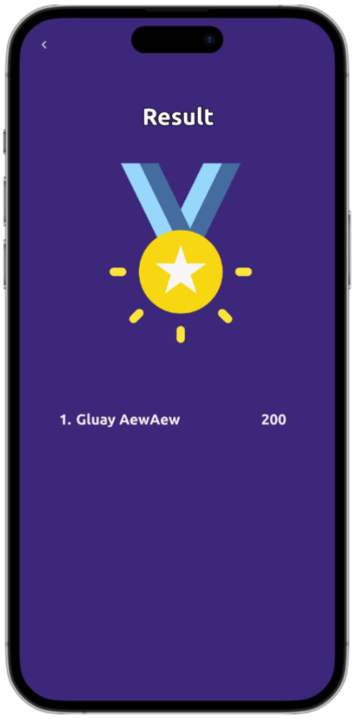


Flamingos are not naturally pink. They get their color from _______.

This page is the answer page that will show automatically when users finish selecting their answer. Besides, the correct answer was set to green color and incorrect answers were set to red color.



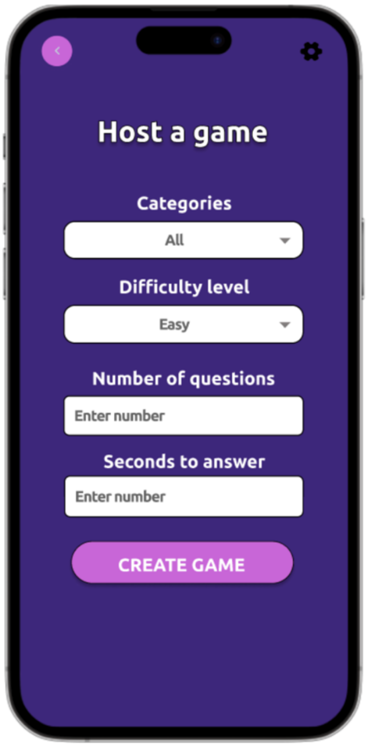
This page is for calculating the score of the user which will appear when the user finished answering all the questions.



Result

1. Gluay AewAew 200

After finishing calculating the score, the result page that has a ranking along with the point will be shown to the user.



Host a game

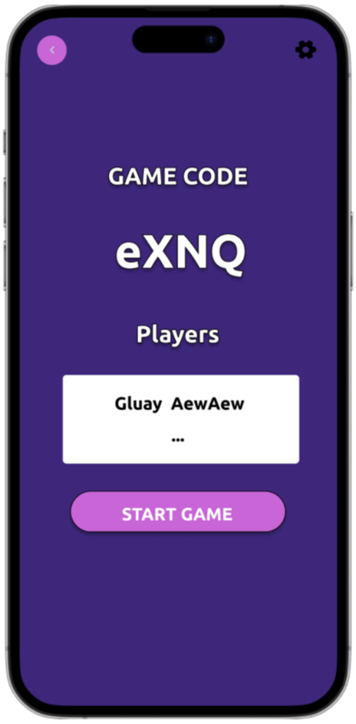
Categories: All

Difficulty level: Easy

Number of questions: Enter number

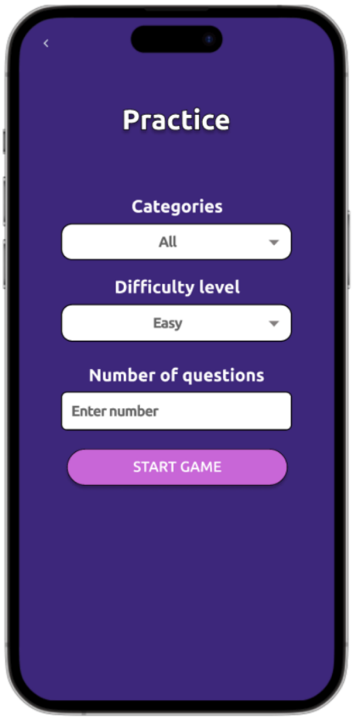
Seconds to answer: Enter number

This page was created for the user who selects the host game option. They have to choose categories, difficulty level, enter the number of questions and seconds to answer in order to create a game.



Gluay AewAew
...

This page will be shown to the user who hosts a game. It will show the game code and name of the player who enters a game code to join the game.

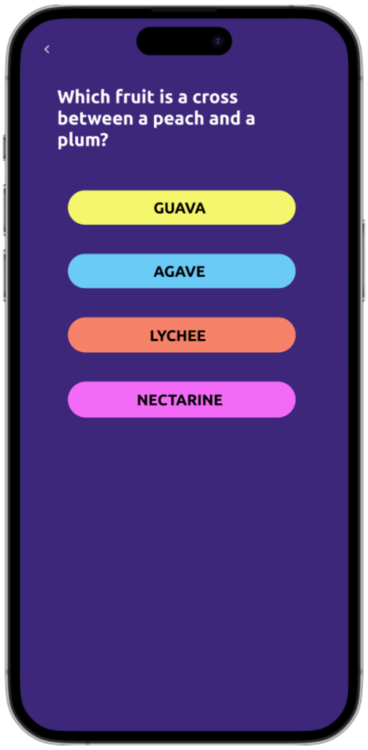


Categories
All

Difficulty level
Easy

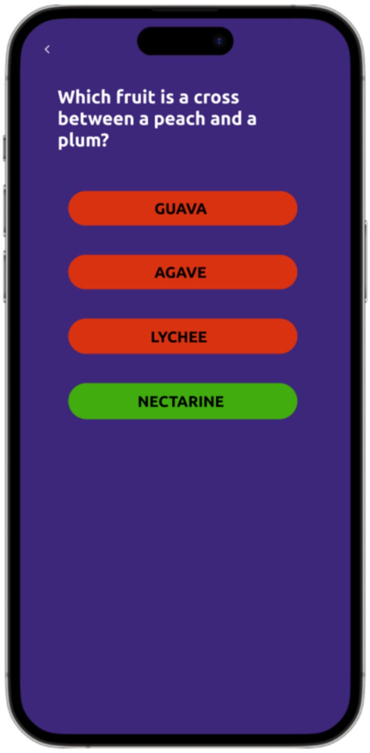
Number of questions
Enter number

This page was created for the user who selects the practice option which they can select categories, difficulty level, and enter the number of questions that they have to practice by themselves before starting to practice a quiz.



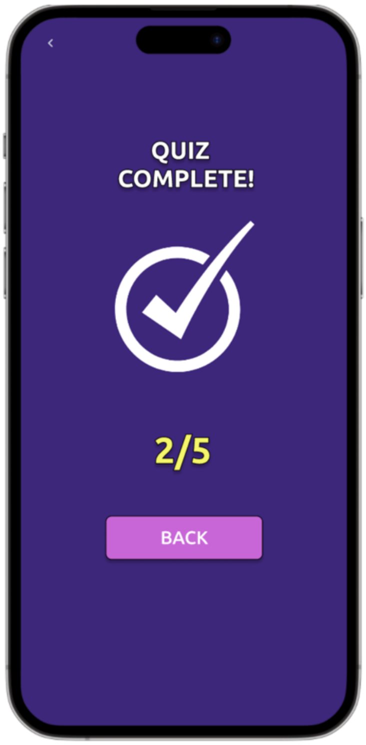
Which fruit is a cross between a peach and a plum?

This page is the example practice question that will show to the user and is different from the real pop quiz because it does not have a time limit for each question.

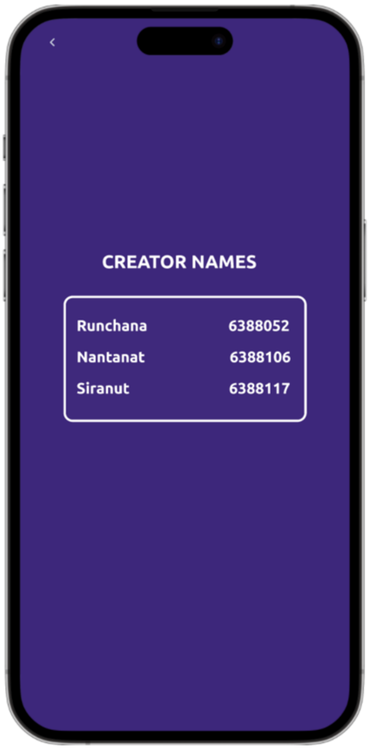


Which fruit is a cross between a peach and a plum?

This page is the answer page that will be shown automatically when users finish answering the question.



This page has the score that users get which will be shown when the user finishes doing a practice.



| | |
|---|---|
| Runchana | 6388052 |
| Nantanat | 6388106 |
| Siranut | 6388117 |

This is the creator page that has the name of the people who create the Bugluay Pop Quiz Game.

### 3.5.2 Transition Diagram

Before accessing the main menu, users will need to fill in their nickname that will be used thoughout the game. From the main menu, users can select “Create Game” to enter the number of questions and seconds to answer. The program will take the user to the waiting room. Alternatively, users can select “Join Game” and enter a room code. Moreover, users can choose to play as a single player by accessing the “Practice” menu. When all users are ready, the host can start the game, which will then display questions from the Trivia API website. After the game, a leaderboard will be shown, and users can choose to return to the main menu to start a new game or perform other actions. The transition diagram provides a clear visualization of the flow of the application and user interactions.

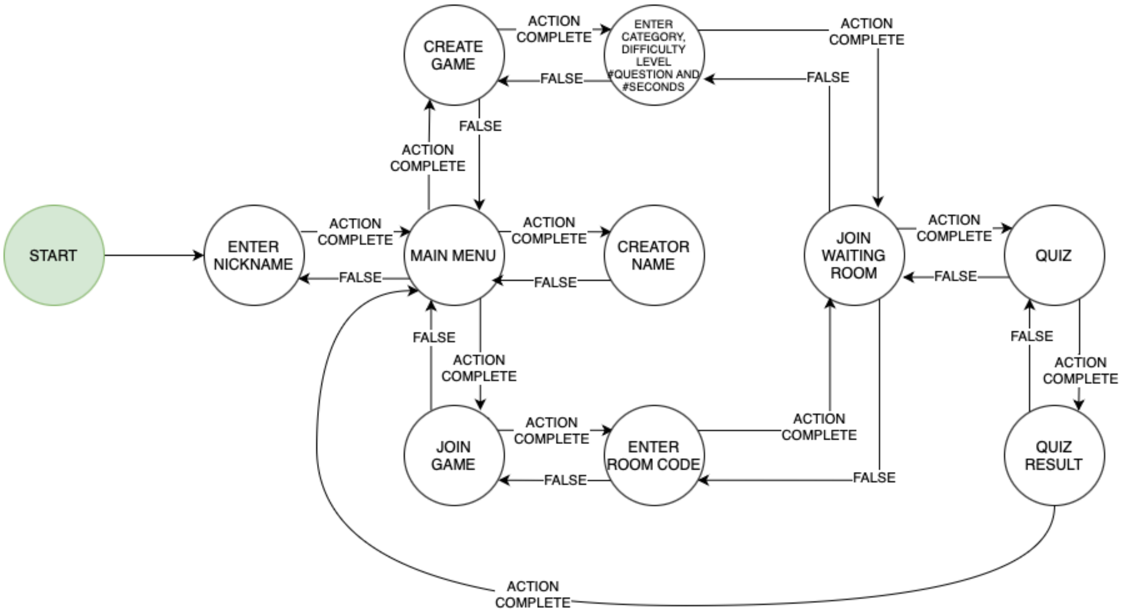


**Figure 5: Transition Diagram**

# CHAPTER 4
# IMPLEMENTATION

This chapter discusses the environment of the hardware and software, as well as the implementation guide and techniques of BuGluay Pop Quiz system.

## 4.1 Hardware and System Environment

- Operating System and Utilities Applications
    - Laptop hardware: Apple M2 chip, 8-core CPU, 10-core GPU, 16-core Neural Engine, 8GB unified memory, 256GB SSD storage,
- Web Server Software & Database Management System
    - Supabase

Supabase is a platform that allows developers to handle backend operations for their web or mobile applications. It provides a wide range of features that allow you to manage your data and logic seamlessly. One of the key features that Supabase provides is a database management system (DBMS), which allows developers to store and retrieve data from their applications. With Supabase, you can set up a PostgreSQL database with minimal effort, and access it through an API or a SQL client.

Supabase also provides a user authentication system, which enables developers to add authentication to their application with just a few lines of code. This system is flexible and customizable, allowing you to create custom login and registration flows, set user roles and permissions, and integrate with third-party authentication providers such as Google, Facebook, and GitHub.

In addition to these features, Supabase provides real-time data streaming, file storage, and serverless functions. With real-time data streaming, you can subscribe to changes in your database and receive updates in real-time. File storage enables you to store files in the cloud and access them through an API, while serverless functions allow you to run custom code in response to events in your application.

- Programming and Scripting Tools
    - Visual Studio Code

Main program for implementation is the Visual Studio Code from the Microsoft Corporation. Visual Studio Code supports hundreds of languages and provides syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. It is convenient for editing, building, and debugging, so it leads to faster development. Furthermore, the implemented program on Visual Studio Code can be integrated with both Android and iOS emulators. VS Code interfaces with build and scripting tools to carry out routine activities, speeding up daily workflows. Git is also supported by VS Code, allowing users to work with source control without leaving the editor, including viewing diffs of pending changes.

Visual Studio Code has great performance when running Flutter and allows users to run it anywhere. Users can run simulators by clicking on the device list in the bottom-right corner and selecting the device from the information bar. When using widgets, VS Code displays the layout of widgets on the left, which helps in the visual organization of the program. Users can deploy code and resolve issues while a simulator is active. All the standard functionality, such as breakpoints, is supported because VS Code supports Flutter. Moreover, there are approximately 240 plugins related to Flutter in VS Code.

- Dart Programming Language

Dart is the principal programming language that is used throughout this project. Dart is a client-oriented programming language that enables users to construct quick apps on any platform, hence it is utilized as our primary programming language. It is generally known as the most productive programming language for cross-platform development as well as an adaptable runtime platform for app frameworks. Dart is specifically built with client development in consideration, prioritizing high-quality production and development experiences across a variety of compilation targets (web, mobile, and desktop). As a result, Dart could be capable of assisting people in developing a pop quiz game that requires a connection between two phones.

In addition, Dart is the foundation for Flutter as well. With its language and runtimes, Dart not only drives Flutter apps but also assits programmers in formatting, examining, and testing code.

- Components
    - This section describes the components of the application “BuGluay”

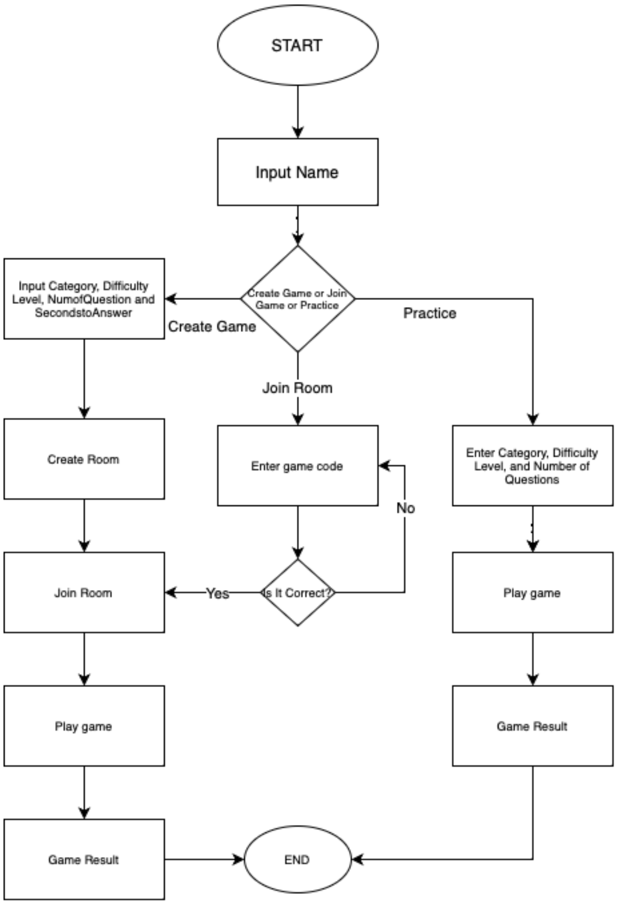

START
Input Name
Create Game or Join Game or Practice
Create Game
Join Room
Practice
Input Category, Difficulty Level, NumofQuestion and SecondstoAnswer
Create Room
Enter game code
Enter Category, Difficulty Level, and Number of Questions
No
Join Room
Yes
Is It Correct?
Play game
Play game
Game Result
Game Result
END

**Figure 6: Flow Diagram**

## 4.2 Implementation Guide and Techniques

### 4.2.1 Connecting the Project to Supabase

For implementing this project, it needs to set some plugins such as Supabase and Flutter SDKs. Supabase is a powerful open-source Backend as a Service (BaaS) that provides a range of services including database management, authentication, and file storage. To connect your Flutter app with Supabase, you need to add the Supabase Flutter plugin to your project. Here's an example of how to initialize Supabase in your Flutter app:

```
import 'package:supabase_flutter/supabase_flutter.dart';

final supabase = await Supabase.initialize(
	url: 'https://your-project-url.supabase.co',
	anonKey: 'your-anon-key',
);
```

Flutter, on the other hand, is a popular mobile app development framework that allows developers to build high-performance mobile apps for both Android and iOS platforms. With Flutter, you can create beautiful and responsive user interfaces using a single codebase. Together, these technologies provide a robust foundation for building scalable and feature-rich mobile apps that can interact with the Supabase backend.

### 4.2.2 Fetching Triva Quiz from the-trivia-api.com

To pull the trivia data for our project, we need to use an API provided by another website, the-trivia-api.com. To do this, we created a TriviaRepository class with a getQuestions method that takes the desired number of questions as a parameter. In the method, we first construct a URL with the desired number of questions using the Uri.https constructor, passing in the website's authority and endpoint for the questions API. We then use the http package to send a GET request to the constructed URL and receive a response containing the questions in JSON format. To decode the JSON response, we use jsonDecode from the dart:convert package and convert the JSON data into a list of TriviaQuestion objects using the map function. Here is the code for the getQuestions method:

```
class TriviaRepository {
  static const _authority = 'the-trivia-api.com';

  // TODO Select categories and difficulty.
  Future<List<TriviaQuestion>> getQuestions(int numOfQuestions) async {
    final url = Uri.https(
        _authority, 'api/questions', {'limit': numOfQuestions.toString()});
    final response = await http.get(url);
    final questions = jsonDecode(utf8.decode(response.bodyBytes)) as List;
    return questions
        .map((q) => TriviaQuestion.fromJson(q as Map<String, dynamic>))
        .toList();
  }
}
```

Overall, this allows us to fetch the trivia questions we need to populate our quiz game from the-trivia-api.com with ease.

# CHAPTER 5
# TESTING AND EVALUATION

This chapter covers how the researchers test some features of this application. This chapter is divided into two parts. The first part is the Unit tests, second is the System integration tests. Therefore, the readers will get to know more about the testing process, the result of the testing, the example scenarios, and a detailed explanation for using the application.

## 5.1 Unit Tests

For the unit tests, we selected some important and critical processes for formal unit testing. The selected processes include:

- Process Number 1: Registration
- Process Number 2: Join a Game
- Process Number 3: Host a Game
- Process Number 5: Start a Game

### 5.1.1. Test Performed on Process Number 1: Registration

Registration function is similar to a typical user registration that must be done before using a system. This is the first interaction between users and the system. In this unit test, the name that users registered should be recorded in a database with a unique user ID.

| Operation Performed | Condition Tested | Actual Result |
|---|---|---|
| **User inputs a nickname** | The registered nickname of the user is recorded in a table ‘players’ in a database. | Pass |

| | | |
|---|---|---|
| **User generates random nickname** | The random nickname function should automatically provide a random nickname as the default value when the user opens an application. | Pass |

Table 5.1.1. Registration

### 5.1.1 Test Performed on Process Number 2: Join a Game

For the join a game function, there is one field for users. It is mandatory for users to input a game code generated by the host player so that they can join an existing game lobby room.

| Operation Performed | Condition Tested | Actual Result |
|---|---|---|
| **User enters valid game code** | Game code exists | Pass |
| **User enters invalid game code** | Game code does not exist | Pass |
| **User enters game code while game is already in progress** | Game is already in progress | Pass |

Table 5.1.2. Join a Game

### 5.1.2 Test Performed on Process Number 3: Host a Game

Hosting a game function includes customizing a setting of a game, which are number of questions and seconds to answer. Therefore, there are two fields in this section. The first field is the number of questions, and another field is seconds to answer. Both fields accept only numeric values.

| Operation Performed | Condition Tested | Actual Result |
|---|---|---|
| **User creates a new game** | User is authenticated and new game session is generated. | Pass |
| **User creates a new game with invalid number of questions** | The number of questions accepts only a numeric value. | Pass |

| | | |
|---|---|---|
| **User creates a new game with invalid seconds to answer** | The second to answer accepts only a numeric value. | Pass |
| **Users select a category** | The questions are from a specific category | Pass |

Table 5.1.3. Host a Game

### 5.1.3 Test Performed on Process Number 5: Start a Game

During a game session, players are required to choose an answer for each question. There are four options to choose from. Each question has only one correct answer, which displays as a green box after answering. The correct answer gives 100 points to the player. Apart from that, an incorrect answer displayed as red. Once every question is answered, the system calculates a total score for each player and displays it on a scoreboard with rankings.

| Operation Performed | Condition Tested | Actual Result |
|---|---|---|
| **Trivia questions are pul1led from the API** | API responds with trivia questions | Pass |
| **User answers a question correctly** | User selects correct answer, and the score is increased by 100 points and shown as green. | Pass |
| **User answers a question incorrectly** | User selects incorrect answer, and the system displays incorrect answer as red. | Pass |
| **Timer works properly** | Timer is set for the correct amount of time | Pass |
| **Answer is shown after the end of the timer** | Correct answer is displayed | Pass |
| **Scoreboard is shown correctly** | The ranking of the players is displayed correcly | Pass |

Table 5.1.4. Start a Game

## 5.2 System Integration Test

This activity is performed after the system is completely integrated. The purpose of this testing is to check whether the system can operate correctly according to the required functions or not.

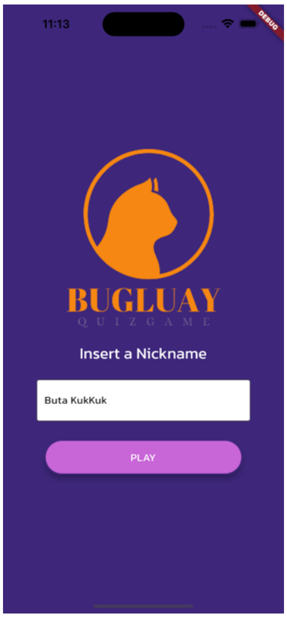
### 5.2.1 Test Scenario

In order to test all functional aspects of the system thoroughly, we had set up a test scenario which consisted of 8 phases as shown below.

- Registration
- Select menu
- Host a game
- Join a game
- Customize a game
- Customize a game room
- Answer questions
- Score board

Moreover, the test scenario can be used as a user guideline because it covers all the steps necessary in order to use our system. The details of each phase are shown in the next section.

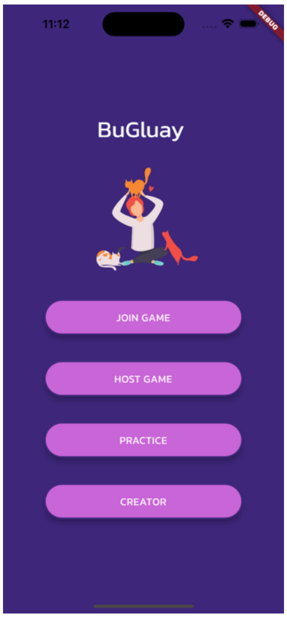
### 5.2.1.1 Registration: Users log in to the system



Buta KukKuk

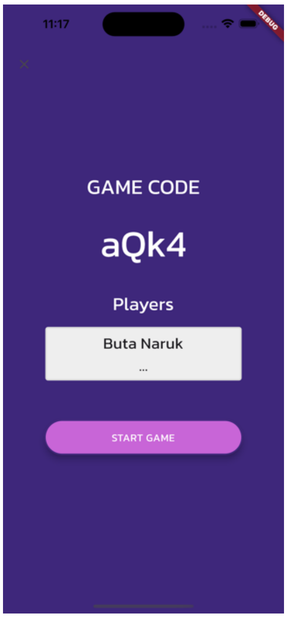
- The ‘Registration’ section starts with the page that asks users to fill in a nickname, which acts as a user authentication.
- In this scenario, the users can choose whether they want to use random name that was automatically generated from the system or specify their desired names in order to start playing a BuGluay pop quiz game.

### 5.2.1.2 Select menu: Users choose to join, host, or play a solo game.

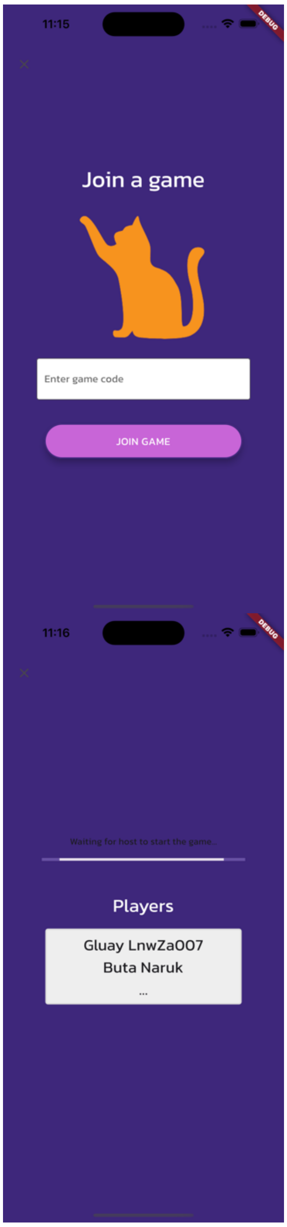


- The ‘select menu’ section starts with a page that provides three main options for users to select. The last button called creator is for displaying a list of creator names.
- In this scenario, the users choose only one option at a time, which will lead them to a specific page.

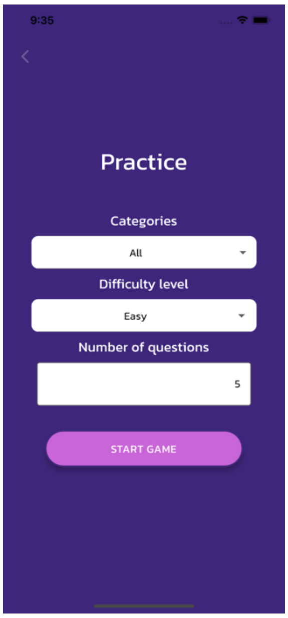
#### 5.2.1.3 Host a game: Users create a room to let other players join the game

| | |
|---|---|
| <br>11:17<br>DEBUG<br>GAME CODE<br>aQk4<br>Players<br>Buta Naruk<br>...<br>START GAME | • In this scenario, the users must customize four options in the game, which are category, difficulty level, number of questions, and seconds to answer.<br>• The generated game code will be used for allowing other players to join the same room as the host player.<br>• The host player presses on a start game button to start a game. |

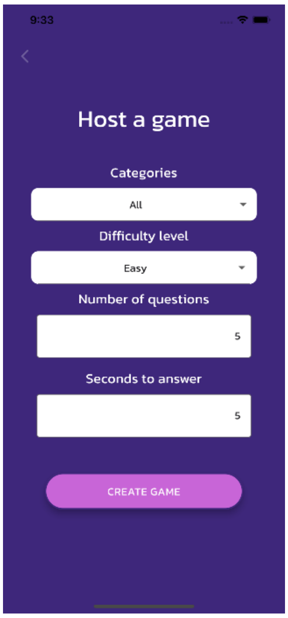
#### 5.2.1.4 Join a game: Users input the game code from a host

| | |
|---|---|
| <br>11:15<br>Join a game<br>Enter game code<br>JOIN GAME<br>11:16<br>Waiting for host to start the game...<br>Players<br>Gluay LnwZa007<br>Buta Naruk<br>... | • The 'join a game' section is to let users join a game room that was created by another player.<br>• There is only one text field on this page that requires user to fill in. The code game that was generated from the host will be a mixing set of numbers and alphabets.<br>• After the user input a game code, the user must press on a join game button. The BuGluay system will lead the user to a lobby room and display a list of current joined players.<br>• If the game code is invalid, the system will alert an error message. |

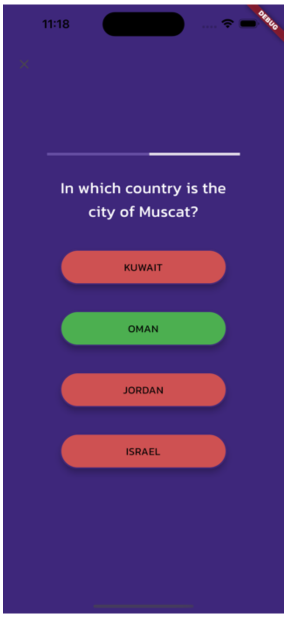
### 5.2.1.5 Customize a game: Users customize the category, difficulty level, and number of questions they would like to answer.

| | |
|---|---|
| <br>9:35<br>Practice<br>Categories<br>All<br>Difficulty level<br>Easy<br>Number of questions<br>5<br> | - The 'Customize a game scenario asks users to select category and difficulty level and input integer number of questions they would like to answer. In practice session, the users can specify only number of questions. |

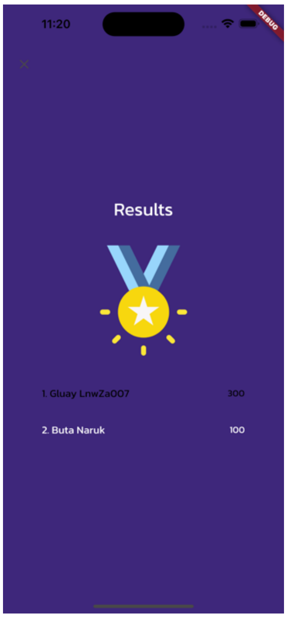
### 5.2.1.6 Customize a game room: Host player inputs the category, difficulty level, number of questions, and time limit.

| | |
|---|---|
| <br>9:33<br>Host a game<br>Categories<br>All<br>Difficulty level<br>Easy<br>Number of questions<br>5<br>Seconds to answer<br>5<br> | - This scenario asks a user who hosts a game to specify four values in four fields. The first one is the category. The second field is difficult level. Another one is the number of questions and the last one is seconds to answer, which will be used during a game, respectively. Default values for each game are questions from all categories, easy level, five questions, and five seconds.<br>- The host player presses on a 'create game' button to open a lobby room with generated game code. |

#### 5.2.1.7 Answer questions: Users choose a correct answer and get a score

| | |
|---|---|
|  <br>In which country is the city of Muscat? | • This scenario is to check if each question can show a correct answer and store each player’s answers or not. The list of answers will be used for a score calculation afterward.<br>• There are four choices to choose, but the users can select only one answer for each question.<br>• Correct answer is shown as green color and gives 100 points, while incorrect answers are shown as a red color and gives zero points. |

#### 5.2.1.8 Score board: Users view a result of the game

| | |
|---|---|
|  <br>Results<br>1. Gluay LnwZa007 300<br>2. Buta Naruk 100 | • In this scenario, users can only view a result of the game they recently played. A score board shows calculated scores of each player and ranking. |

# CHAPTER 6
# CONCLUSIONS

The project's conclusion is covered in this chapter. It outlines the advantages, issues, restrictions, and the researchers' perspectives on future study.

## 6.1 Benefits

This topic will explain the benefits of the project to the audience.

### 6.1.1 Benefits to Project Developers

- Creativity
- Opportunity for Collaboration
- Skill development in Flutter and Dart
- Learn how to use supabase including authentication feature

### 6.1.2 Benefits to Users

- Entertainment
- Stress Relief
- Improve knowledge and skills
- Improve cognitive skills: Attention and concentration

## 6.2 Problems and Limitations

- Require internet access: The application requires the user to connect to the internet because the system must send and retrieve data to an online database.
- Cannot customize theme and avatar.

## 6.3 Future Work

- Provide several languages, such as Thai, Japanese, Chinese, etc.
- Add sound effects and background music.
- Develop a function to collect users’ match history and statistics.

REFERENCES

[1] C. Pappas, "8 Innovative Ways To Use Pop Quizzes," eLearning Industry, 30 September 2018. [Online]. Available: https://elearningindustry.com/innovative-ways-use-pop-quizzes-in-online-training. [Accessed 17 February 2023].

[2] O. I. School, "Does your child have a good attention span?," Oakridge International School, 8 July 2022. [Online]. Available: https://www.oakridge.in/oakridge-international-schools/blogs/does-your-child-have-a-good-attention-span/#:~:text=Attention%20span%20is%20the%20key,has%20taken%20from%20the%20lesson.. [Accessed 18 February 2023].

# APPENDIX A
# LINK TO GITHUB PROJECT

https://github.com/pkbsa/bugluayquiz_flutter

# BIOGRAPHIES

| | |
|---|---|
| **NAME** | Miss Runchana Seesung |
| **INSTITUTIONS ATTENDED** | Suksanari, 2019: |
| | High School Diploma |
| | Mahidol University, 2023: |
| | Bachelor of Science (ICT) |
| | |
| **NAME** | Miss Nantanat Sudlar |
| **INSTITUTIONS ATTENDED** | Saint Joseph Convent, 2019: |
| | High School Diploma |
| | Mahidol University, 2023: |
| | Bachelor of Science (ICT) |
| | |
| **NAME** | Mr. Siranut Akarawuthi |
| **INSTITUTIONS ATTENDED** | Bangkok Christian College, 2019: |
| | High School Diploma |
| | Mahidol University, 2023: |
| | Bachelor of Science (ICT) |