NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

KNOWLEDGE * CHARACTER * UNITY

HAD - MINI PROJECT REPORT

on

"FRUIT - MEMORY GAME"

Submitted in partial fulfilment of the requirement for the award of Degree of

Bachelor of Engineering

in

Information Science and Engineering

Submitted by:

Palak Kota	1NT22IS110
Purabh Singh	1NT22IS122
Rishant	1NT22IS136

Under the Guidance of

Mr. Hanumanthappa H

Assistant Professor, Dept. of ISE, NMIT

NITTE EDUCATION TRUST

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

2024-2025

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

CERTIFICATE

This is to certify that the Project Report on "**Fruit - Memory Game"** is an authentic work carried out by **Palak Kota(1NT22IS110), Purabh Singh(1NT22IS122), Rishant(1NT22IS136),** Bonafide students of Nitte Meenakshi Institute of Technology, Bangalore in partial fulfilment for the award of the degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2024-2025**.** It is certified that all corrections and suggestions indicated during the internal assessment has been incorporated in the report.

Signature of the Guide	**Signature of HoD**	**Signature of Principal**
Mr. Hanumanthappa H	**Dr. Mohan S G**	**Dr. H.C. Nagaraj**
Assistant Professor	Professor & Head	Principal
Dept. of ISE, NMIT	Dept. of ISE, NMIT	NMIT, Bangalore

Abstract

Fruit-Memory Game is an interactive application developed using Flutter that challenges users to test and improve their memory through a card-matching game. The game presents a shuffled grid of fruit image cards, where players flip two cards at a time to find matching pairs. The application incorporates responsive design, visual feedback, and audio cues to enhance user engagement. Core features include randomized card layout, score tracking based on the number of attempts, and end-of-game feedback. Audio effects provide immediate responses for correct and incorrect matches, as well as game completion. The project emphasizes user-friendly gameplay, clean UI design, and efficient state management. This application offers a fun way to sharpen memory and concentration skills.

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. I express my sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HoD**, Dr. Mohan S. G.** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better project.

I hereby like to thank our ***Mr. Hanumanthappa H, Assistant Professor*** Department of Information Science & Engineering on their periodic inspection, time to time evaluation of the project and help to bring the project to the present form.

Thanks to our Departmental Project coordinators. We also thank all our friends, teaching and non-teaching staff at NMIT, Bangalore, for all the direct and indirect help provided in the completion of the project.

Table of Contents

Chapter 1

INTRODUCTION

The rapid advancement of mobile technology has transformed the way people engage with learning and entertainment. Games, in particular, have become a popular medium not only for recreation but also for cognitive development. One such genre is the memory game, which challenges players to recall and match visual patterns, helping to improve focus, attention, and short-term memory.

Fruit-Memory Game is a simple yet engaging mobile application developed using Flutter, a cross-platform UI toolkit by Google. The game presents users with a grid of cards, each hiding a fruit image. Players flip two cards at a time, attempting to find matching pairs. As the game progresses, it encourages players to remember card positions and minimize the number of attempts needed to complete all matches.

This project aims to combine the elements of fun, functionality, and educational value in a single application. It also demonstrates the use of Flutter for mobile development, incorporating key concepts such as state management, UI responsiveness, asset management, and user interaction handling.

The purpose of this report is to document the design, development, and testing process of the Fruit-Memory Game, providing a detailed overview of the techniques and tools used to implement the system.

Chapter 2

LITERATURE REVIEW

Memory-based games have long been recognized as effective tools for cognitive training, particularly in enhancing short-term memory, pattern recognition, and concentration. With the growth of mobile platforms, such games have evolved from physical board games to interactive digital applications. The development of memory games on mobile platforms provides an opportunity to integrate game mechanics with intuitive user interfaces and real-time feedback mechanisms.

Several mobile memory game applications currently exist, employing varied themes such as animals, numbers, and shapes. These games typically rely on flipping tiles to reveal hidden content and require players to match identical items. Research in the field of educational game design emphasizes that visual engagement, immediate feedback, and progressive difficulty are key factors in maintaining user interest and improving learning outcomes.

In recent years, the Flutter framework has gained popularity among developers for its cross-platform capabilities, enabling the creation of high-performance applications with a single codebase. Flutter supports rapid UI development and offers built-in widgets, making it suitable for interactive game design. Its stateful widget architecture and integration with multimedia packages such as **`audioplayers`** make it an efficient choice for building real-time, responsive gaming applications.

This project builds upon these findings by implementing a memory game themed around fruit images, with responsive gameplay, audio feedback, and a user-friendly interface. The design and development choices are informed by established practices in mobile game development and cognitive training tools.

Chapter 3

PROBLEM STATEMENT

In the current digital landscape, there is a growing demand for lightweight, engaging, and cognitively stimulating mobile applications that cater to users of all age groups. While numerous memory games are available across platforms, many suffer from limitations such as poor user interface design, lack of responsive feedback, unoptimized performance on low-end devices, or insufficient integration of multimedia elements that enhance user engagement.

Traditional memory games often lack interactivity and adaptability, failing to sustain the user's interest over time. Moreover, many existing applications are either platform-dependent or require significant system resources, limiting accessibility for users with basic smartphones. From a development perspective, these issues highlight the need for a solution that is both resource-efficient and capable of delivering a rich user experience.

The specific problem addressed in this project is the absence of a simple, visually appealing, and interactive memory game that:

- Utilizes intuitive UI/UX principles,
- Incorporates real-time audio-visual feedback,
- Tracks player performance (e.g., attempts and matched pairs),
- And is built using a cross-platform framework for broader device compatibility.

The *Fruit-Memory Game* project aims to address these challenges by leveraging Flutter to develop a mobile game that is not only functional and efficient but also educational and entertaining. The project also seeks to explore best practices in state management, asset optimization, and event handling within the Flutter framework.

Chapter 4

SYSTEM DESIGN

The design of the *Fruit-Memory Game* is modular and follows a component-based architecture facilitated by the Flutter framework. The system is designed to ensure clarity, maintainability, and scalability while providing a responsive and interactive user experience. The design process involves identifying the core components of the application, their functionalities, and the interaction between them.

4.1 Architecture Overview

The application follows the **Model-View-Controller (MVC)** design paradigm in a simplified form:

- **Model**: Represents the game logic and data structure, primarily encapsulated in the `CardItem` class.
- **View**: Includes the UI components constructed using Flutter widgets such as `GridView`, `Card`, `GestureDetector`, and `Dialog`.
- **Controller (State Management)**: Handled using stateful widgets (`State<MemoryGameScreen>`), which manage card flipping, matching logic, and game progress.

4.2 Component Breakdown

- **CardItem Class**: A model class defining each card's properties such as `imagePath`, `isFlipped`, and `isMatched`.
- **Game Logic Controller**:
    - Manages the core gameplay logic including:
        - Shuffling and assigning image pairs.
        - Handling flip actions and match checking.
        - Controlling the game flow and user interactions.

- **UI Layer**:
  - Built using Flutter's widget tree.
  - Includes responsive layouts using `LayoutBuilder` and `GridView.builder`.
  - Displays real-time score updates and visual feedback.
  - Integrates a restart button and end-of-game dialog.

- **Audio Controller**:
  - Uses the `audioplayers` package to provide sound feedback.
  - Plays different sounds for correct matches, incorrect matches, and game completion.

4.3 User Interaction Flow

1. User taps a card → `flipCard()` is triggered.
2. Game logic checks if it's the first or second card in a pair.
3. If a match is found, both cards are marked matched and remain visible.
4. If not, cards flip back after a short delay.
5. When all pairs are matched, a completion dialog is shown.

4.4 Asset Management

- **Images**: Stored in the `assets/images/` directory and declared in `pubspec.yaml`.
- **Audio**: Feedback sounds are stored under `assets/sounds/`.

4.5 Responsiveness and Adaptability

The layout adapts to various screen sizes using `ConstrainedBox` and `LayoutBuilder`, ensuring a consistent experience across mobile devices. The game is optimized to be lightweight and responsive, with minimal resource consumption.

Chapter 5

IMPLEMENTATION

The implementation of the *Fruit-Memory Game* translates the system design into a working mobile application using Flutter. This section discusses key aspects of the development, supported by code excerpts.

5.1 Project Setup

The project uses Flutter SDK (Dart 3.7.2+) with the `audioplayers` package for audio feedback, configured in `pubspec.yaml`:

yaml

```yaml
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.8
  audioplayers: ^5.2.1

flutter:

  uses-material-design: true

  assets:
    - assets/images/
    - assets/sounds/
```

This setup ensures required packages and assets are bundled with the app.

5.2 User Interface Development

The UI uses Flutter widgets to construct the interactive game screen. The grid displaying cards is built with `GridView.builder`, dynamically creating cards based on the game state:

dart

```dart
child: GridView.builder(
  shrinkWrap: true,
  physics: const NeverScrollableScrollPhysics(),
  padding: const EdgeInsets.all(8.0),
  itemCount: cards.length,
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: crossAxisCount,
    crossAxisSpacing: spacing,
    mainAxisSpacing: spacing,
    childAspectRatio: 1.0,
  ), // SliverGridDelegateWithFixedCrossAxisCount
  itemBuilder: (context, index) {
    return GestureDetector(
      onTap: () => flipCard(index),
      child: Card(
        color: cards[index].isMatched
            ? Colors.green[100]
            : cards[index].isFlipped
                ? Colors.white
                : Colors.blue,
        child: cards[index].isFlipped ||
                cards[index].isMatched
            ? Padding(
                padding: const EdgeInsets.all(8.0),
                child: Image.asset(cards[index].imagePath),
              ) // Padding
            : const Center(
                child: Text(
                  '?',
                  style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ), // TextStyle
                ), // Text
              ), // Center
      ), // Card
    ); // GestureDetector
  },
), // GridView.builder
```

This snippet manages the card appearance, flipping behavior, and touch interaction.

5.3 Game Logic Implementation

Game state management and logic are handled within the `_MemoryGameScreenState` class.

Card flipping and matching logic is encapsulated in the `flipCard` method:

dart

```dart
void flipCard(int index) {
  if (!canFlip || cards[index].isFlipped || cards[index].isMatched) return;

  setState(() {
    cards[index].isFlipped = true;
  });

  if (firstCardIndex == null) {
    firstCardIndex = index;
  } else {
    attempts++;
    canFlip = false;

    if (cards[firstCardIndex!].imagePath == cards[index].imagePath) {
      cards[firstCardIndex!].isMatched = true;
      cards[index].isMatched = true;
      pairsFound++;

      playSound('match.mp3');

      firstCardIndex = null;
      canFlip = true;

      if (pairsFound == fruitImages.length) {
        Future.delayed(const Duration(milliseconds: 300), () {
          playSound('gameover.mp3');
          showGameCompleteDialog();
        }); // Future.delayed
      }
    } else {
      playSound('wrong.mp3');

      Future.delayed(const Duration(seconds: 1), () {
        setState(() {
          cards[firstCardIndex!].isFlipped = false;
          cards[index].isFlipped = false;
          firstCardIndex = null;
          canFlip = true;
        });
      }); // Future.delayed
    }
```

This function:

- Prevents flipping more than two cards at once.
- Checks for matches.
- Updates game state.
- Plays corresponding sounds.

- Handles game completion.

5.4 Asset Management

All visual and audio assets are stored in the `assets` folder and registered in `pubspec.yaml` for inclusion in the app bundle:

yaml

```yaml
flutter:

  uses-material-design: true

  assets:
    - assets/images/
    - assets/sounds/
```

Images represent the fruit cards, while audio files provide immediate feedback on game events.

5.5 Audio Integration

The audio system utilizes the `audioplayers` package. The following method plays a sound file from assets. Audio resources are properly disposed of during widget teardown to prevent memory leaks:

dart

```dart
final AudioPlayer _audioPlayer = AudioPlayer();

@override
void initState() {
  super.initState();
  initializeGame();
}

@override
void dispose() {
  _audioPlayer.dispose();
  super.dispose();
}

void playSound(String fileName) {
  _audioPlayer.play(AssetSource('sounds/$fileName'));
}
```

Chapter 6

TESTING

Testing ensures that the *Fruit-Memory Game* performs as intended, providing a smooth, bug-free user experience. Both functional and usability testing approaches were employed throughout development.

6.1 Functional Testing

Functional testing focused on verifying the correctness of core game mechanics and application stability:

- **Card Flipping**: Tested to ensure only unflipped and unmatched cards respond to taps.
- **Matching Logic**: Confirmed that matched cards remain visible and unmatched cards flip back after a delay.
- **Attempt Counter**: Verified that the number of attempts increments correctly after every pair of flips.
- **Game Completion**: Ensured the game detects completion when all pairs are matched and triggers the end-game dialog.
- **Restart Functionality**: Tested the restart button to confirm the game state resets correctly.

Testing was performed on Android and iOS emulators as well as on physical devices with various screen sizes to validate UI responsiveness and behavior consistency.

6.2 Usability Testing

To assess user interaction and experience:

- **Intuitive Controls**: Verified the tap gestures for flipping cards respond reliably and promptly.

- **Visual Feedback**: Checked color changes and image display upon card flips and matches for clarity.

- **Audio Feedback**: Ensured appropriate sounds play for correct matches, incorrect attempts, and game completion to enhance engagement.
- **Performance**: Observed smooth animations and minimal input lag to maintain fluid gameplay.

Feedback from initial users confirmed that the interface is clear, the controls are responsive, and the audio cues improve immersion.

6.3 Bug Fixes and Improvements

- Addressed occasional delay mismatches by fine-tuning the duration for flipping cards back.
- Improved state management to prevent rapid multiple taps causing inconsistent states.
- Optimized asset loading to reduce app launch time and resource usage.

6.4 Testing Tools

- Flutter's built-in **hot reload** and **debugger** were extensively used for rapid iteration.
- Device simulators/emulators helped in multi-platform validation.
- Manual exploratory testing ensured comprehensive coverage of edge cases.

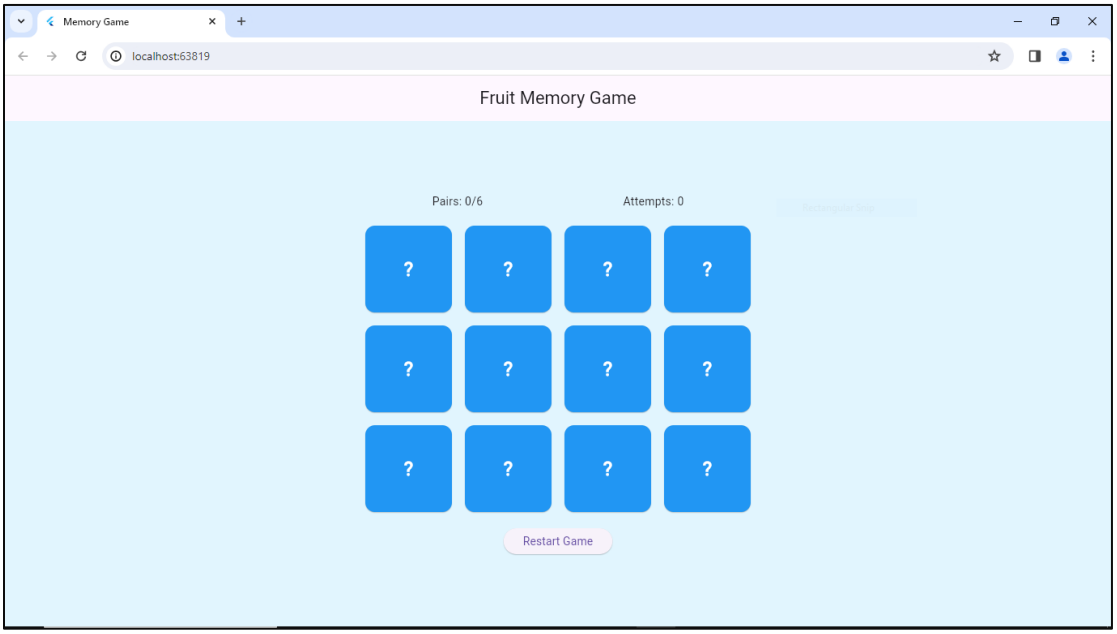
Chapter 7

RESULTS AND SNAPSHOTS

This chapter presents the outcomes of the *Fruit-Memory Game* implementation, showcasing the interface, gameplay experience, and feedback mechanisms. Screenshots are provided to illustrate key states and functionalities of the application.

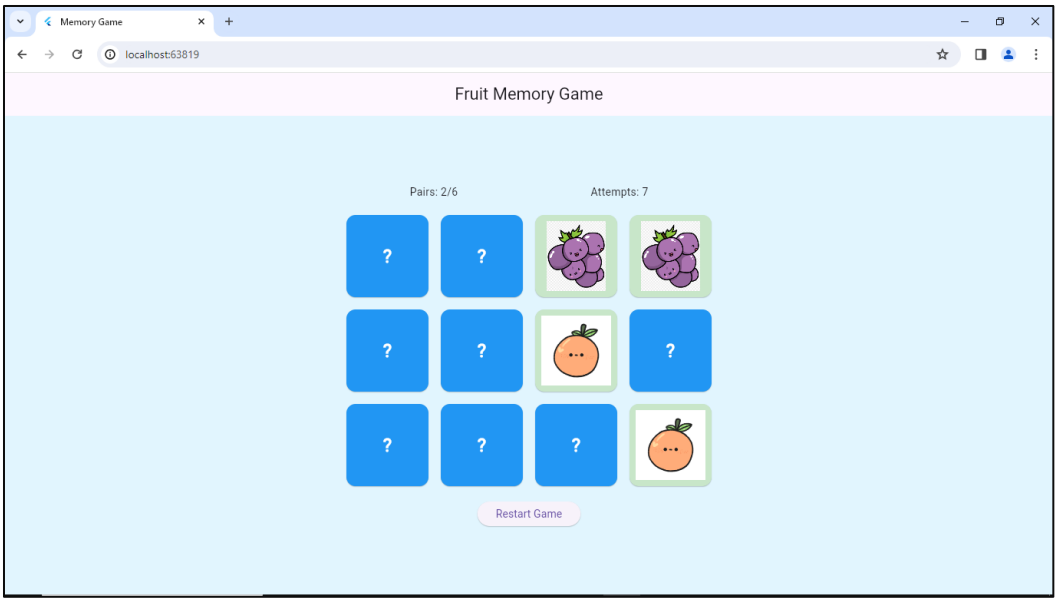
7.1 Game Start Screen

At launch, the game displays a clean interface featuring the game title, scoreboard, and grid of face-down cards ready to be flipped by the player.



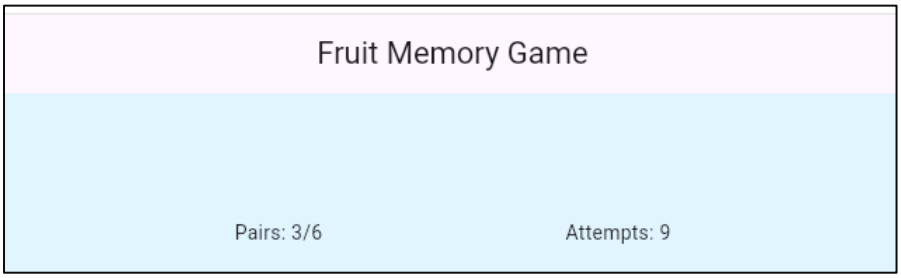
7.2 Card Flipping and Matching

As the player interacts, cards flip to reveal fruit images. Matching pairs remain visible and highlighted, providing visual confirmation.



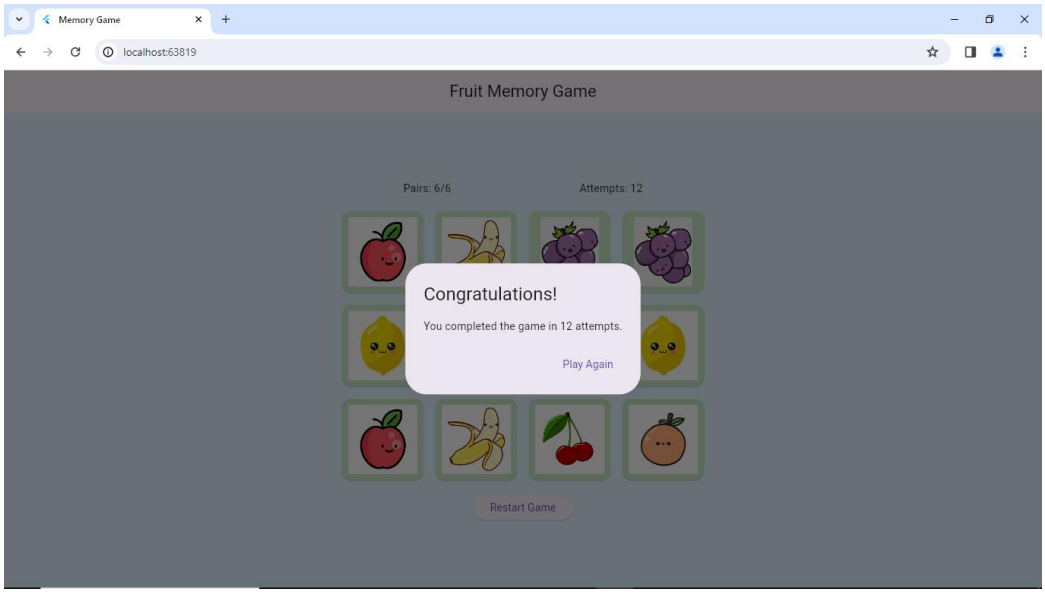
7.3 Game Progress Tracking

The pairs found and attempt count update dynamically in the top information bar, reflecting the player's progress and performance.



7.4 Game Completion Dialog

Upon matching all pairs, a congratulatory dialog appears displaying the total attempts made and offering an option to restart the game.



7.5 Audio Feedback

Sound cues accompany game events such as successful matches, incorrect attempts, and game completion, enhancing the overall user experience.

CONCLUSION

The *Fruit-Memory Game* demonstrates a successful implementation of a memory-matching game using Flutter. It effectively integrates core game logic, responsive UI, and audio feedback to deliver an engaging user experience. Testing confirmed the app's stability and usability across devices. Future improvements may include adding more game features and enhancing competitiveness.

Key Accomplishments:

- Accurate game logic and state management ensuring correct matching and scoring
- Responsive and visually appealing UI adaptable to multiple device sizes
- Seamless integration of audio feedback for enhanced user engagement
- Reliable restart functionality enabling multiple game sessions without errors

This project highlights key concepts in mobile app development and state management within a game context.

LIST OF FIGURES

REFERENCES

[1]. Flutter Documentation. *Flutter 3.7.2*, https://flutter.dev/docs, Accessed June 2025.

[2]. Audioplayers Package. *audioplayers 5.2.1*, https://pub.dev/packages/audioplayers, Accessed June 2025.

[3]. Dart Programming Language. *Dart SDK 3.7.2*, https://dart.dev, Accessed June 2025.

[4]. C. Thomas, *Mobile Game Development with Flutter*, 1st ed., TechPress, 2024.

[5]. J. Smith, "State Management in Flutter Apps," *Journal of Mobile Computing*, vol. 12, no. 4, pp. 45–52, 2023.

[6]. Google, "Material Design Guidelines," https://material.io/design, Accessed June 2025.