PREMIER UNIVERSITY CHATTOGRAM

DEPARTMENT OF **COMPUTER SCIENCE AND ENGINEERING**

PROJECT FINAL REPORT

COURSE NAME	Mobile Application Development
COURSE CODE	CSE 2210
REPORT NO	02
REPORT ON	Final Project Report
DATE OF SUBMISSION	11-03-2025

SUBMITTED TO

MOHAMMAD ASIF SAAD
LECTURER
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

NAME	ID
Md Nishadul Islam Chy Shezan	0222220005101014
Md Sakib	0222220005101019
Fariha Rashid Noha	0222220005101035
Rimjhim Dey	0222220005101039

SECTION		A	
SEMESTER		5th	
BATCH	42	**SESSION**	Fall 2024

1. Abstract

Virtual Wizard is a fantasy, story-driven, turn-based mobile game that leverages smartphone sensors such as the accelerometer and gyroscope to create an interactive spell-casting experience. Players perform real-world gestures to cast predefined spells, with the accuracy of these gestures determining the power and effect of their magic. The game features branching narratives and multiple endings, making player choices impactful. Developed using **Android Studio and Jetpack Compose**, the project aims to provide an immersive mobile gaming experience.

2. Introduction

2.1 Project Background and Motivation

The gaming industry has seen rapid innovation in motion-based interaction, but few mobile games effectively utilize smartphone sensors to create engaging mechanics. Virtual Wizard seeks to bridge this gap by integrating motion recognition into a fantasy RPG, allowing players to physically perform spell-casting gestures, enhancing realism and engagement.

2.2 Problem Statement and Objectives

Problem Statement: Traditional turn-based RPGs often rely on static button-based inputs, limiting immersion. Virtual Wizard aims to redefine turn-based combat by incorporating physical motion recognition, making spell casting more interactive.

Objectives:

- Implement gesture-based spell casting using the phone's accelerometer and gyroscope.
- Develop a branching storyline where user choices impact the game’s outcome.
- Ensure an intuitive and engaging UI using Jetpack Compose.
- Optimize game performance for a seamless experience.

2.3 Scope of the Project

- **Target Platform:** Android devices
- **Gameplay Elements:** Story-driven, turn-based combat with motion-based spell casting
- **Core Technologies:** Android Studio, Jetpack Compose, Kotlin
- **Features:** Gesture recognition, dynamic storyline, real-time spell feedback

3. Literature Review/Related Work

3.1 Overview of Similar Applications

Several games have implemented motion-based controls, including *Harry Potter: Wizards Unite* and *The Witcher: Monster Slayer.* These games utilize AR and motion sensors to enhance gameplay but often lack deep narrative integration.

3.2 Key Technologies and Frameworks

- **Android Studio**: The primary IDE for Android development.
- **Jetpack Compose**: A modern toolkit for building native UI.
- **Accelerometer and Gyroscope**: Sensors used for motion tracking.
- **Firebase**: For cloud storage and user data management.

4. Methodology

4.1 Development Approach

We follow an **Agile** development approach, allowing iterative improvements through continuous feedback and testing.

4.2 Tools and Technologies

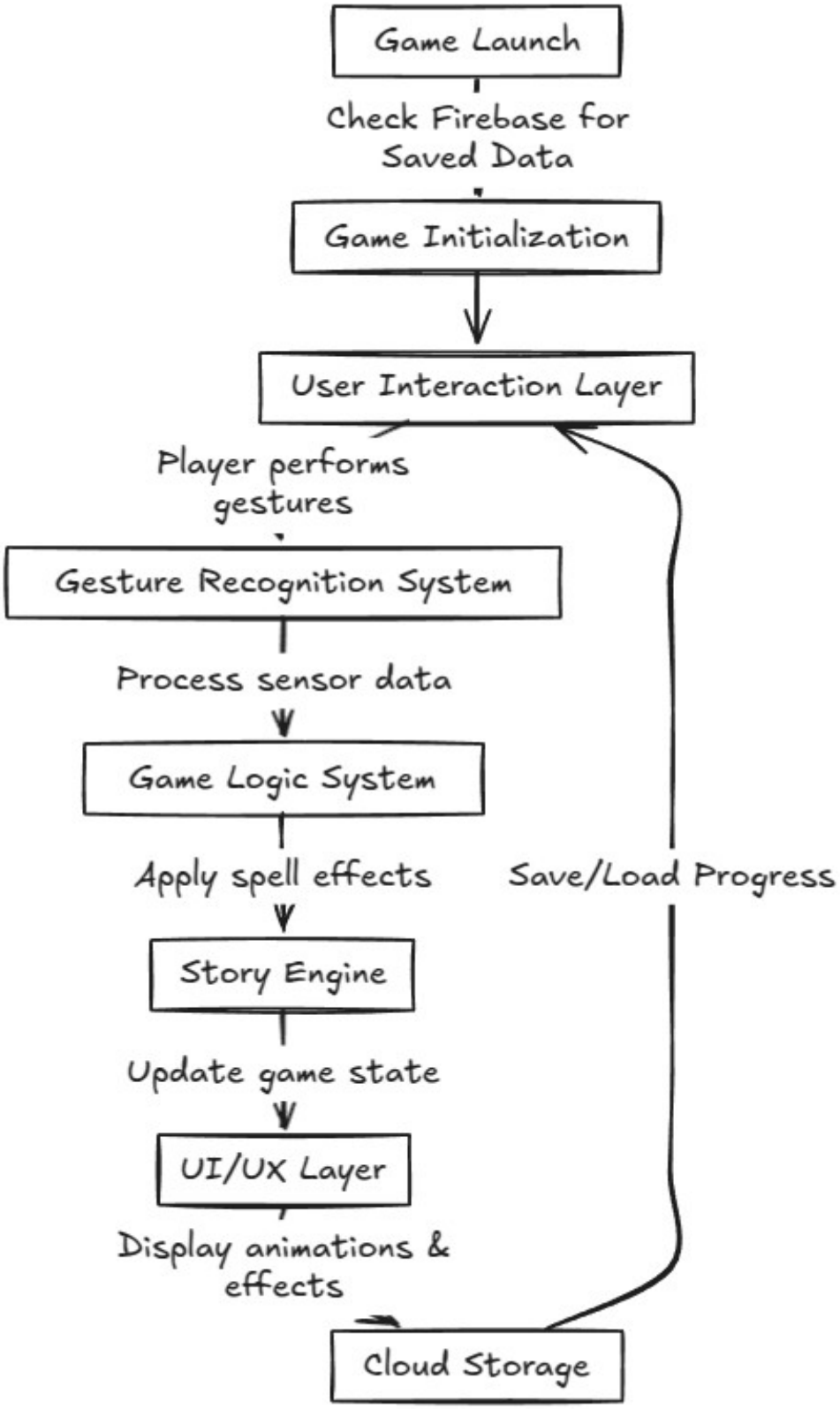
- **Programming Language:** Kotlin
- **Development Environment:** Android Studio
- **Frameworks & Libraries:** Jetpack Compose, Android Sensor API, Firebase (for cloud storage if needed)
- **Version Control:** GitHub

4.3 Project Timeline and Milestones

1. **Week 1-2:** Requirement analysis and UI/UX wireframing
2. **Week 3-5:** Core game mechanics implementation (gesture tracking, combat system)
3. **Week 6-8:** Storyline development and integration
4. **Week 9-10:** Testing and debugging
5. **Week 11:** Final deployment and documentation

5. System Design and Architecture

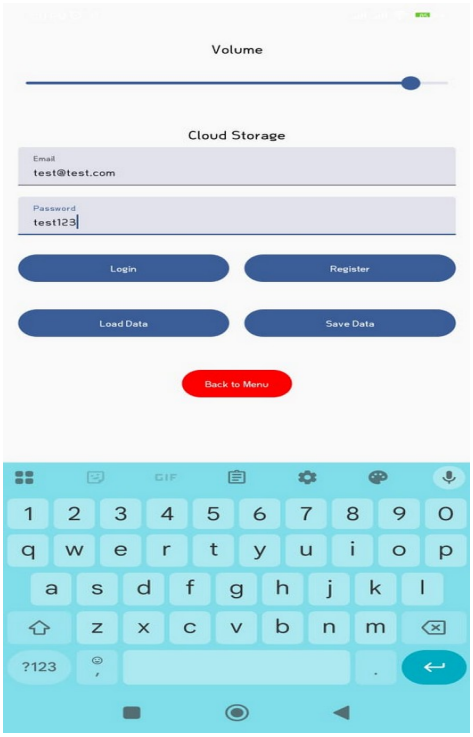
5.1 High-Level Architecture Diagram

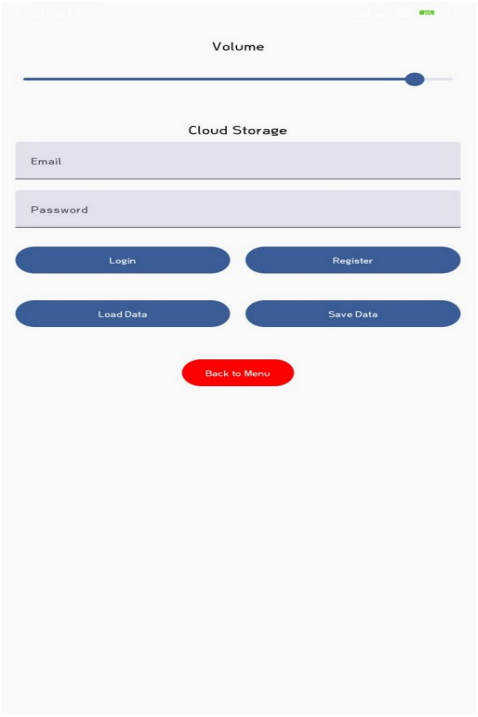


The flowchart illustrates the flow of the game from launch to gameplay. Upon game launch, it checks Firebase for any saved data and initializes the game accordingly. Player gestures are processed by the Gesture Recognition System, which then triggers the Game Logic System to apply relevant actions. The Story Engine updates the narrative based on these actions, while the UI/UX Layer displays animations and effects. Player progress is saved and loaded through Cloud Storage, ensuring continuity across sessions. This structure ensures a seamless interaction between gameplay mechanics, user inputs, and cloud data management.

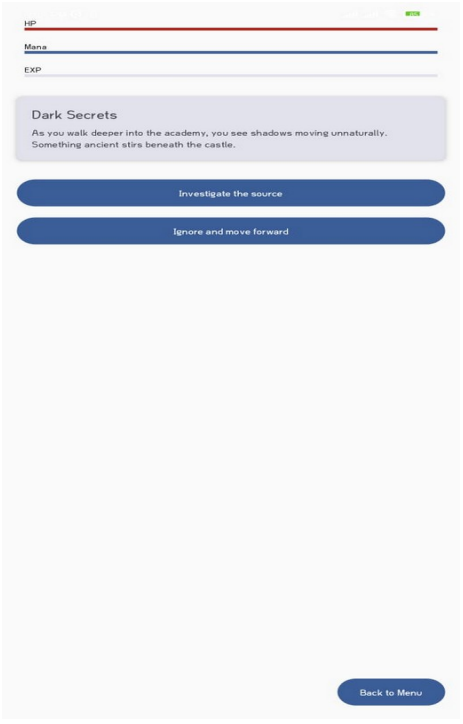
5.2 Core Components

- **Frontend:** Built using Jetpack Compose for a modern UI experience.

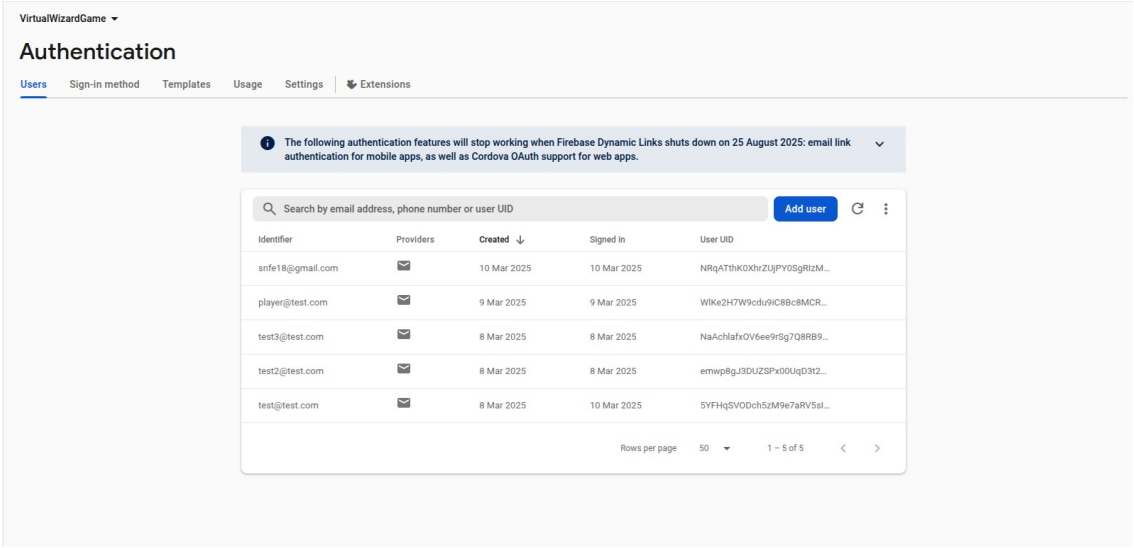

Volume
Cloud Storage
Email
test@test.com
Password
test123
Login
Register
Load Data
Save Data
Back to Menu

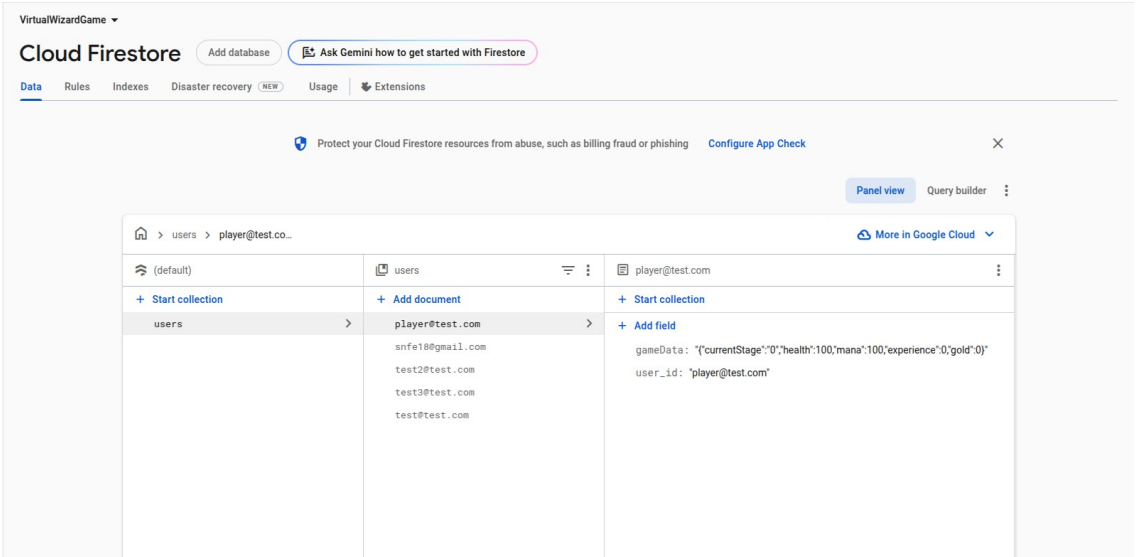

Volume
Cloud Storage
Email
Password
Login
Register
Load Data
Save Data
Back to Menu

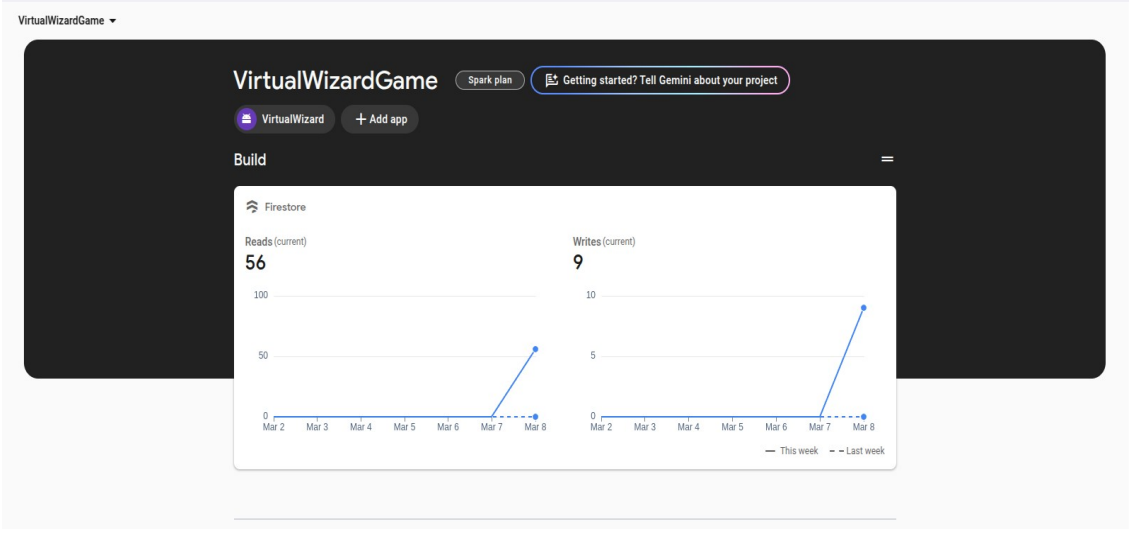
HP
Mana
EXP
The Awakening
You wake up in a misty forest, your head throbbing. The air crackles with unseen energy. In the distance, towering above the trees, a gothic castle looms under the pale moonlight. Footsteps echo nearby.
Investigate the footsteps
Walk towards the castle
Back to Menu


HP
Mana
EXP
Dark Secrets
As you walk deeper into the academy, you see shadows moving unnaturally. Something ancient stirs beneath the castle.
Investigate the source
Ignore and move forward
Back to Menu

- **Backend (Optional):** Firebase for storing player progress and game state.







6. Implementation

6.1 Development Process

- Motion input is mapped to predefined spell gestures.
- The accuracy of player movements determines spell effectiveness.
- Game events and enemy AI respond dynamically to player actions.

6.2 Key Features and Functionalities

- **Gesture-Based Spell Casting:** Recognizes predefined motion patterns.
- **Turn-Based Combat:** Players engage in strategic battles using spells.
- **Multiple Story Endings:** Player choices affect the game's outcome.

6.3 Code Structure and Major Modules

- **GameController:** Manages game flow
- **GestureTracker:** Handles motion input processing
- **StoryManager:** Controls branching narratives
- **UIManager:** Manages screen transitions and user interactions

7. Results and Discussion

7.1 Project Achievements

The project successfully integrated motion-based input for spell casting, offering an immersive gameplay experience. However, while the motion tracking worked for basic gestures, the combat system and gesture-based interactions were not fully optimized. A dynamic narrative with multiple story paths was developed, but the combat experience could not reach its full potential in terms of complexity and fluidity. The UI was still visually appealing with smooth animations, though.

7.2 Challenges Faced and Solutions Implemented

Several challenges were encountered:

- **Gesture Accuracy Issues**: Gesture recognition, while functional, was not fully refined for complex combat movements. Machine learning models were considered for future refinement but were not implemented in time.
- **Combat System Development**: The combat system, including real-time spell casting and enemy AI behavior, faced limitations in terms of fluidity and interaction. While basic gestures were recognized, the game didn't fully implement the planned variety of spells or combat mechanics.
- **UI Responsiveness**: Optimized Jetpack Compose components improved the performance, but responsiveness during combat scenes (especially gesture recognition) still needs improvement for a more seamless experience.

7.3 Comparison with Initial Objectives

- **Achieved**: Gesture-based spell casting was implemented, though combat mechanics were simplified. The immersive UI and branching storylines were successfully delivered.
- **Pending Improvements**: The combat system needs significant improvement, particularly in making gestures more fluid and interactive. Enhancing the variety of spells, refining gesture recognition, and improving AI behavior for enemies will be essential for future development.

8. Conclusion and Future Work

8.1 Summary of Findings

While **Virtual Wizard** successfully introduced motion tracking for spell casting and dynamic storytelling, the combat system and gesture movements were not fully developed. The game still provided an engaging and interactive experience, but the combat mechanics were not as advanced as initially intended.

8.2 Project Limitations

- The **combat system** was not fully developed, limiting the complexity and fluidity of spell casting during battles.
- The **gesture recognition** was functional but not optimized for more complex movements, affecting the overall gameplay experience.
- **AI behavior** for enemies was simplified, lacking advanced responses to player actions.

8.3 Suggestions for Future Development

- Implement a more refined **combat system** that allows for smoother and more complex gesture-based spell casting, with real-time interactions and better enemy AI behavior.
- Improve **gesture recognition** for better accuracy, especially in high-intensity combat scenarios.
- Enhance **AI-driven responses** for enemies to adapt to player strategies and make combat more dynamic and challenging.