

FARMGUARD: A MOBILE GAME TO COMBAT PHISHING SCAMS TARGETING RURAL FARMERS

SUPERVISOR: MR. FREDRICK OGORE

NAME: ALLAN NGUGI

DECLARATION

I declare that this project report is my original work and has not been submitted, either in part or in full, for the award of a degree or diploma in any other institution. All sources of information used have been duly acknowledged.

Name: Allan Ngugi

Signature: 

Date: 16th August, 2025

ABSTRACT

Cybercrime is also victimizing vulnerable communities, like rural farmers, with phishing scams. The scams typically present themselves in the form of spoof text messages proposing government subsidies, complimentary agricultural inputs, or urgent payment demands (*Cyber Security Report Q1 2024-2025.Pdf*, n.d.). The FarmGuard project is a mobile gamified learning system that seeks to educate rural farmers about phishing scams and equip them with the skills to recognize and avoid them. The system utilizes scenario-based game play to simulate real scam messages so users can practice identifying scams in a safe environment. Built with Flutter for cross-platform compatibility and integrated into Firebase Authentication and Cloud Firestore, the system maintains user score tallies and ranks users on a leaderboard to encourage engagement. Testing confirmed the system was simple to use and effectively improved awareness of phishing. This report covers the complete system development cycle, from planning to implementation, testing, and assessment.

TABLE OF CONTENTS

CHAPTER ONE	5
1.1 Introduction.....	5
1.2 Background Information.....	5
1.3 Problem Statement.....	6
1.4 Main Objective	8
1.5 Specific Objectives	8
CHAPTER TWO	9
2.0 Literature Review	9
CHAPTER THREE	10
3.0 Methodology	10
3.1 Preliminary Investigation.....	12
3.2 Design	14
CHAPTER FOUR.....	19
4.0 System Analysis and Design	19
4.1 Data Flow Diagram.....	20
4.2. Entity Relationship Diagram	21
CHAPTER FIVE	22
5.1System Code	22
The main.dart.....	22
Login Screen.....	23
Signup Screen.....	24
How to Play Screen	25
Scenario Screen	26
Scenario Data	28
Feedback Screen.....	30
Leaderboard Screen.....	32
5.2 Screenshot Images.....	33
Login Screen	33
Signup Screen.....	33
Home Screen.....	34
How to Play Screen	34
Scenario Screen	35

Feedback Screen (Player has chosen the right option)	35
Feedback Screen (Player has chosen the wrong option).....	36
Leaderboard Screen.....	36
CHAPTER SIX	37
6.0 Conclusions	37
6.1 Recommendations	37
6.2 Final Reflection	38
REFERENCES.....	39

CHAPTER ONE

1.1 Introduction

Agriculture remains the backbone of Kenya's economy, with over 70% of rural dwellers involved in agriculture (*2023-Economic-Survey.Pdf*, n.d.). The rapid financial services digitalization, particularly via mobile media like M-Pesa, has inadvertently exposed rural farmers to cyber-crime, particularly phishing scams. These scams usually manifest in the shape of faked SMS messages, faked loan proposals, and imitation agro-dealer promotions, leading to huge financial losses to smallholder farmers (*Cyber Security Report Q1 2024-2025.Pdf*, n.d.)

1.2 Background Information

While the city dwellers have had numerous digital literacy drives and good exposure to cybersecurity courses, rural folks are well behind. Poor exposure to digital awareness platforms, poor literacy levels, and sluggish internet connections render rural farmers at an extremely vulnerable stage.

For this, the project is about developing FarmGuard—a mobile application developed using Flutter and Firebase—intended to simulate real phishing scam experiences faced by farmers daily in real life. That is to facilitate cybersecurity awareness from passive reading to active learning through experiential gamification methods improving user engagement and retention.

The game will educate users through simulated instances of common phishing strategies such as the use of M-Pesa subsidy links pretence, agro-vet transaction pretence on WhatsApp, and government impersonation calls. Through interaction with the simulations, users will be educated on how to identify suspicious activity, feel confident in handling such situations, and reduce the risk of becoming a scam victim.

1.3 Problem Statement

Despite Kenya's strong digital uptake, rural farmers are highly susceptible to phishing attacks due to low levels of digital literacy, limited cybersecurity outreach, and a shortage of localized education resources. Referring to the Cyber Security Report Q1 2024–2025 from the Communications Authority of Kenya (2024), mobile scams in rural parts of the country have risen sharply, where scammers are exploiting user knowledge deficits and the informal nature of farmer communication channels such as SMS and WhatsApp.

The farming industry is among the top contributors of Kenya's GDP, with over 70% of rural families depending on it for livelihood (*2023-Economic-Survey.Pdf*, n.d.). The extensive use of digital platforms such as mobile money and smartphone applications has created avenues through which scammers have been able to exploit unsuspecting users who have not received any formal training in nefarious digital manipulation. Farmers are normally victims of misinformation offering fake government subsidy programs, discounted loans, or subsidized agro-vet products at low prices in exchange for upfront payments or secret personal details.

Current cybersecurity awareness programs, including national TV broadcasts, SMS messages, and printed guides, have been urban-centric and failed to reach rural multitudes. These initiatives typically use technical jargon, lack interactivity, and do not include the experiential learning necessary for retention and long-term behavior change. Moreover, due to infrastructural limitations, like poor internet connectivity in rural regions, the majority of farmers are unable to access online training modules or live awareness sessions.

This persistent weakness created a trust deficit in rural users of digital platforms, limiting additional digitization and financial inclusion. Lacking efficacious, tangible, and reachable training interventions, the communities remain at risk. Therefore, there exists an imperative to find a solution permitting low literacy levels, offline capability, and behavioral learning by practice.

FarmGuard, a game based on Flutter and Firebase analytics, is a new solution to the issue. It makes users experience phishing attacks in a game-like simulated environment, thereby offering users a means of simulation of actual threats in a safe environment. The players

engage with real-like materials that warn them of warning signs, respond to spam inquiries, and build safe digital habits. This method not only raises awareness but also implements permanent defensive function against the cyber-attacks they are undergoing.

Ultimately, this problem deserves its notice presently since cyber fraud is growing steadily and has reached a point of extending to unsuspecting communities. FarmGuard aims to fill the gap between rural empowerment and cybersecurity awareness by promoting resistance against phishing attacks through an interactive, informative, and efficient platform.

1.4 Main Objective

To design an interactive mobile game on Flutter and Firebase that educates rural Kenyan farmers how to identify and avoid common phishing scams.

1.5 Specific Objectives

- I. To develop an Android game developed on Flutter that simulates common phishing scams against Kenyan farmers.
- II. To implement visual learning tools, including scam warning icons and color-coded buttons, to enhance user interaction and comprehension.
- III. To utilize Firebase for tracking anonymized user achievements, including quiz score and leaderboard data.
- IV. To evaluate the effect of the game on farmers' competence in recognizing and avoiding phishing attacks.

The increasing phenomenon of phishing regarding rural farmers requires innovative learning methods tailored to their unique contexts. Traditional approaches, such as pamphlets and seminars, are limited to access and attendance. Gamification has been successful in enhancing learning performance, particularly in cybersecurity education. The project uses mobile gaming as a platform, as it offers engaging and interactive experiences, to address the digital literacy gap and enable farmers to protect themselves against cyber fraud.

CHAPTER TWO

2.0 Literature Review

Phishing has become more advanced, riding on the speedy process of digitalization in developing countries. In Kenya, rural communities' extensive usage of mobile money services and smartphones has rendered farmers an easy target for phishers (*Cyber Security Report Q1 2024-2025.Pdf*, n.d.). The most prevalent methods are fake SMS presenting government subsidies, agro-dealer deals sent through messaging applications, and fake loan requests.

The rural digital divide exacerbates the problem. Insufficient access to digital education and resources makes farmers unable to identify and respond to phishing (Tinubu et al., 2023). Initiatives like PHISHGEM have highlighted the requirement for digital literacy to empower rural communities.

Gamification has been a successful technique in the realm of cybersecurity learning. Evidence has shown that game-based learning can significantly improve users' skills to detect and defend against phishing attacks (Tinubu et al., 2023). Phishing has been tackled by games like "PHISHGEM" and proved the ability of mobile games to expand awareness about phishing with interactive and interactive content.

Although numerous digital literacy materials and cybersecurity education campaigns exist world over such as PHISHGEM, a role-playing game for general phishing training, or traditional outreach activities in Kenya such as SMS fraud warnings and TV advertising campaigns, FarmGuard stands out in modeling its approach specifically for the rural Kenyan farmers. Unlike PHISHGEM, which was designed for a general public, FarmGuard simulates common phishing attacks on Kenyan farmers, namely M-Pesa subsidy scams, government relief and fertilizer scams.

CHAPTER THREE

3.0 Methodology

The idea for the project arose from a gap analysis: while the majority of cybersecurity awareness campaigns target urban residents through social media and television, rural communities—who are more vulnerable due to low digital literacy—remain underserved. Observing this, there was an examination on existing gamified awareness software like PHISHGEM (Tinubu et al., 2023), which showed that mobile games can be effective phishing educational platforms.

This project takes an iterative, user-centered development process motivated by the demand for accessible cybersecurity awareness among rural farmers. The FarmGuard game was envisioned after reading several academic publications and government reports about the increased phishing scams on Kenyan farmers.

The resources that were used were as follows:

- I. ***Flutter SDK***: This will be used for the cross-platform development.
- II. ***Firebase***: This will be used for backend operation, such as user progress tracking.
- III. ***Figma***: This will be used for designing and prototyping the user interface.

The development process adapted was more of a simple agile process. It involved the following steps:

1. ***Conceptualization:***

The application includes the main features such as visual alert of scams, quiz-based interface, and tracking progress.

2. ***Design Phase***

The UI mockups were completed using Figma, and simplicity was stressed accompanied by color-coded decision buttons (green = safe, red = risky) and pop-up descriptions.

3. ***Development Phase***

Flutter was used to implement the core game logic, navigation, interaction

mechanism, and quiz functionality. Firebase Firestore was added for storing data such as user data and leaderboard data.

4. ***Simulation & Testing***

Simulations were done. The prototype was tested, which resulted in small changes to the UI and logic sequence.

5. ***Refinement & Packaging***

After integrating the testing, the game was optimized for performance, ready for review. The analytics and scoring logic was finalized, and documentation was prepared to be presented.

3.1 Preliminary Investigation

The user requirements of the FarmGuard game were created in such a way as to encompass the needs of smallholder farmers with perhaps minimal digital literacy and exposure to sophisticated mobile phones. The needs are such that the game not only ends up being educational and enjoyable but also usable and relevant to the rural users in Kenya.

User Needs Identified

- I. ***Simplicity*** – The game must present a simple, easy-to-understand interface with minimal words to cater to users who are not computer savvy.
- II. ***Realistic Scenarios*** – The content must include phishing simulations that represent real-life scam tactics experienced in the agricultural sector.
- III. ***Immediate Feedback*** – Players must receive instant feedbacks to their actions so that they may learn efficiently.
- IV. ***Performance Monitoring*** – Like scores or progress indicators are needed to encourage users and demonstrate improvement over time.

The methods of collecting user information was:

- I. ***Secondary Data:*** National statistics and cybersecurity publications (e.g., KNBS, CAK) were utilized to support the simulated research to ensure the project remains based on real threats and user contexts.

Functional Requirements

These specify the system's functionality required to meet user expectations.

- I. The system must simulate five phishing scenarios common with Kenyan farmers.
- II. Users must receive immediate feedback once they have responded to a scam scenario (e.g., "Correct!" or "You got scammed!").

- III. The system should utilize color-coded buttons (e.g., red for danger, green for safe).
- IV. Firebase will be used to store scores and user performance data anonymously.
- V. High scores will be shown in a leaderboard without collecting personal information, other than the name.

Non-Functional Requirements

These are the performance qualities of the system.

- I. Accessibility: Visual elements must be large, color-coded, and easy to use for people with low literacy.
- II. Lightweight: APK size must be minimal in order to install and run on low-end Android phones.
- III. Security and privacy: No personal information will be exposed; all user information will be anonymized, except for the name when displaying a user on the leaderboard.

3.2 Design

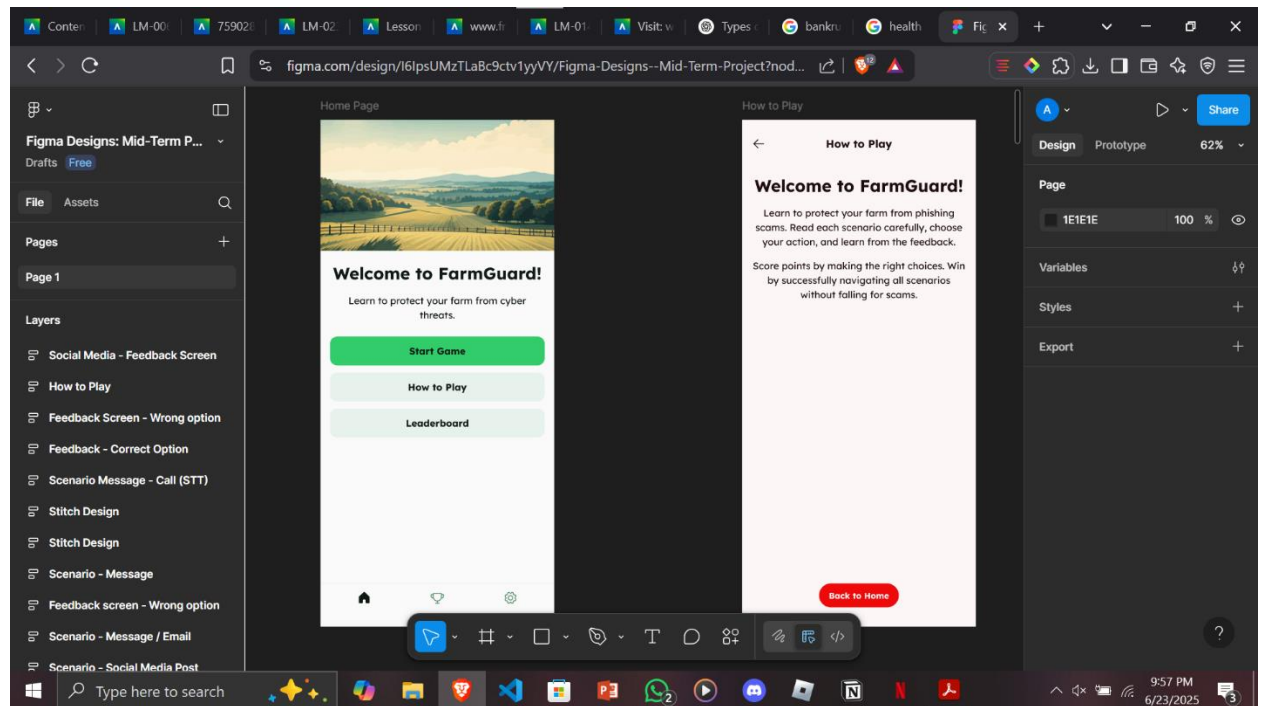
Design phase of FarmGuard project includes creation of an interactive, user-focused, and educational mobile game that mimics real phishing attacks. System architecture design and user interface (UI) design are included in this phase while prioritizing making the final application suitable for rural users without undermining technical realizability within the time frame of the project.

User Interface (UI) Design

The UI design was developed in the simplicity, clarity, and accessibility model to assist low-digital-literacy users. The wireframes and interactive mockups were constructed in Figma to visualize screen flow, navigation structure, and content layout.

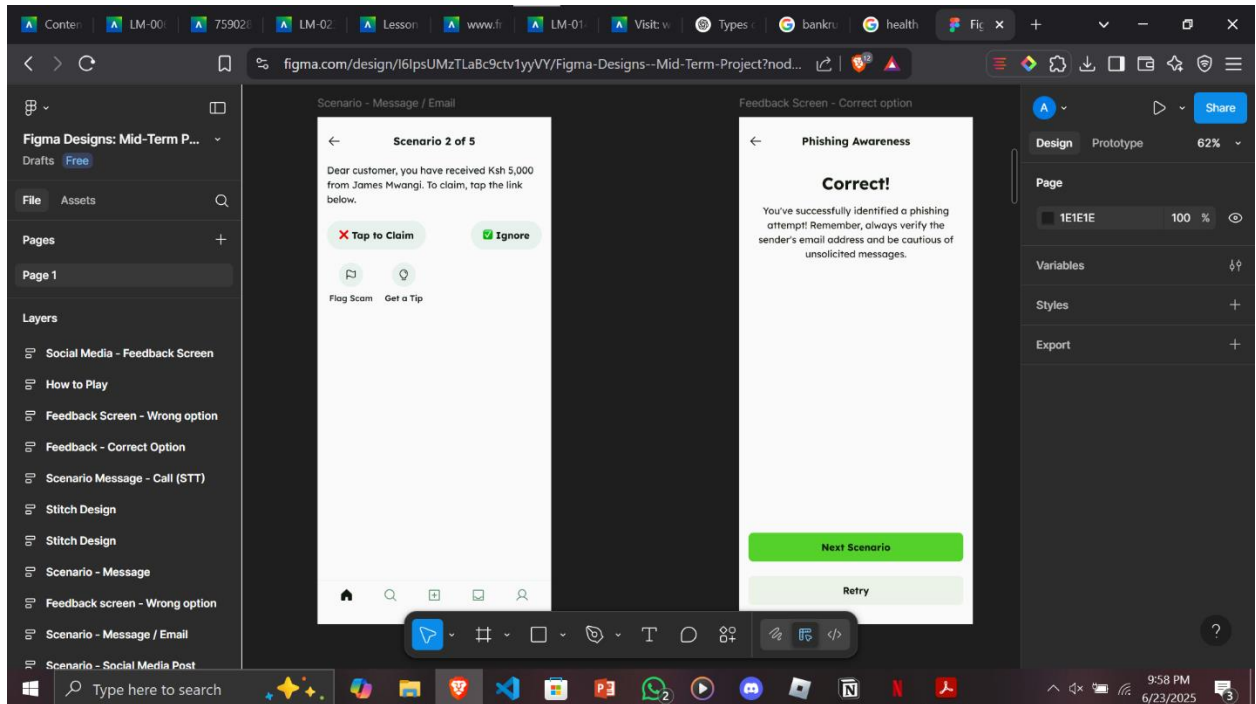
The figma screens developed were as follows:

Home and How to Play screens



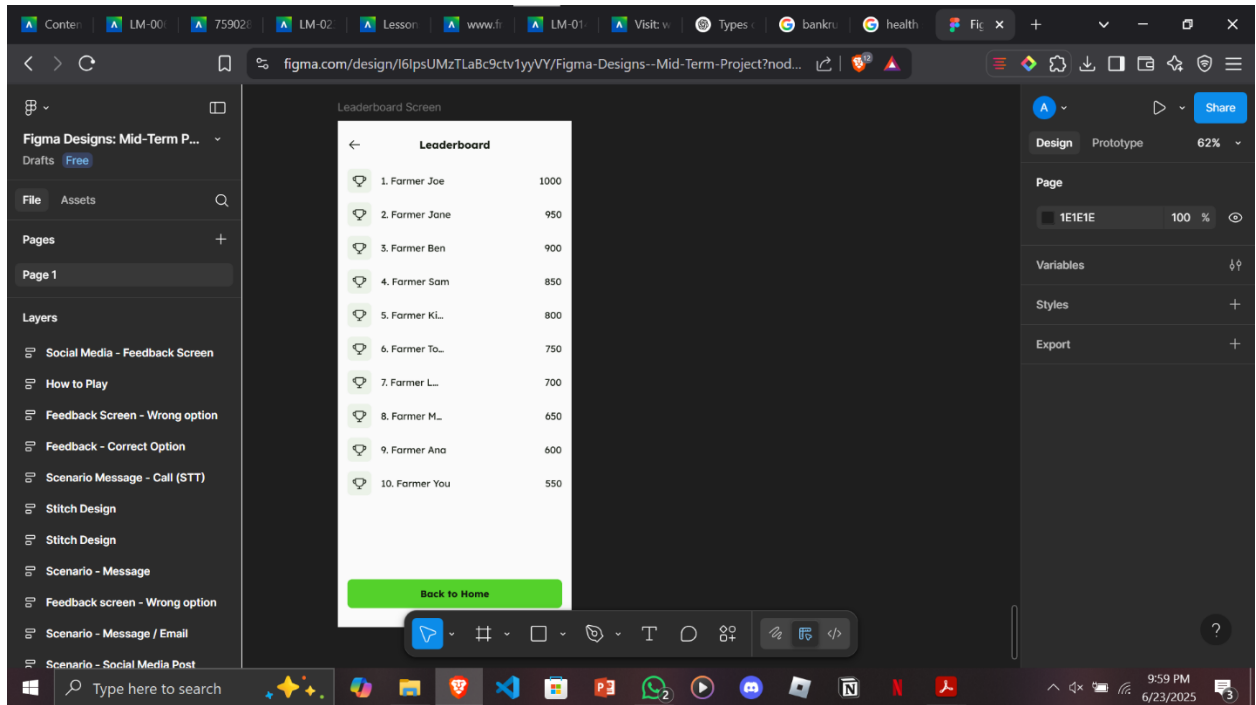
The image above were two screens designed; the one on the left being the home screen; and on the right being the how to play screen.

Scenario and Feedback Screens



The image above shows an example of the scenario screen and the consequent feedback screen.

Leaderboard Screen



The image below shows the design of the leaderboard screen.

Few key features of the UI design are:

- I. Color-Coded Buttons: Green for safe options and red for potentially scammy or risky activity, enabling quick decision-making.
- II. Scam Alert Icons: Visual cues (e.g., exclamation marks, warning signs) to help the user recognize suspicious indicators.
- III. Simple Navigation: Each screen has obvious backward arrow, a forward button and minimal text, relying heavily on icons and images to communicate.
- IV. Game Flow and Scenario Design

The game consists of a series of scenarios mimicking frequent scams as well as true verifiable message sources such as:

- I. Mock M-Pesa vouchers,
- II. An authentic farming subsidy announcement from the Kenyan state,
- III. Emergency funds from a government relief scheme,

- IV. Mock fertilizer advertisements
- V. An authentic e-farming workshop notice of registration announcing no payments were required.

The inclusion of real government-provided messages alongside the fake ones was intentional, designed to strengthen users' real world discrimination capacity. Through exposure to real and fake examples, players would be in a position to avoid over-generalizing that all unsolicited messages are scams, hence fostering critical evaluation instead of blanket rejection. This adjustment brought realism to the simulation and made the training representative of the diversity of the real world communication threats rural farmers encounter.

System Architecture and Backend Designing

The application is on a modular architecture with a clear front-end (Flutter-based) and back-end service separation. This is dealt with by Firebase. The setup is as follows:

Frontend (Flutter)

- I. User interface widgets
- II. Assistant
- III. Game logic and random scenario generation

Backend (Firebase Firestore & Analytics)

- I. Saving quiz score and user progress
- II. Leaderboard component for high scores

The methodology followed on FarmGuard emphasizes accessibility, usability, and user experience. With sequential design, mock user testing, and technological choices like Flutter and Firebase, the project is set to be a valuable learning tool for phishing awareness. Every phase of development—from the conceptualization to design—was

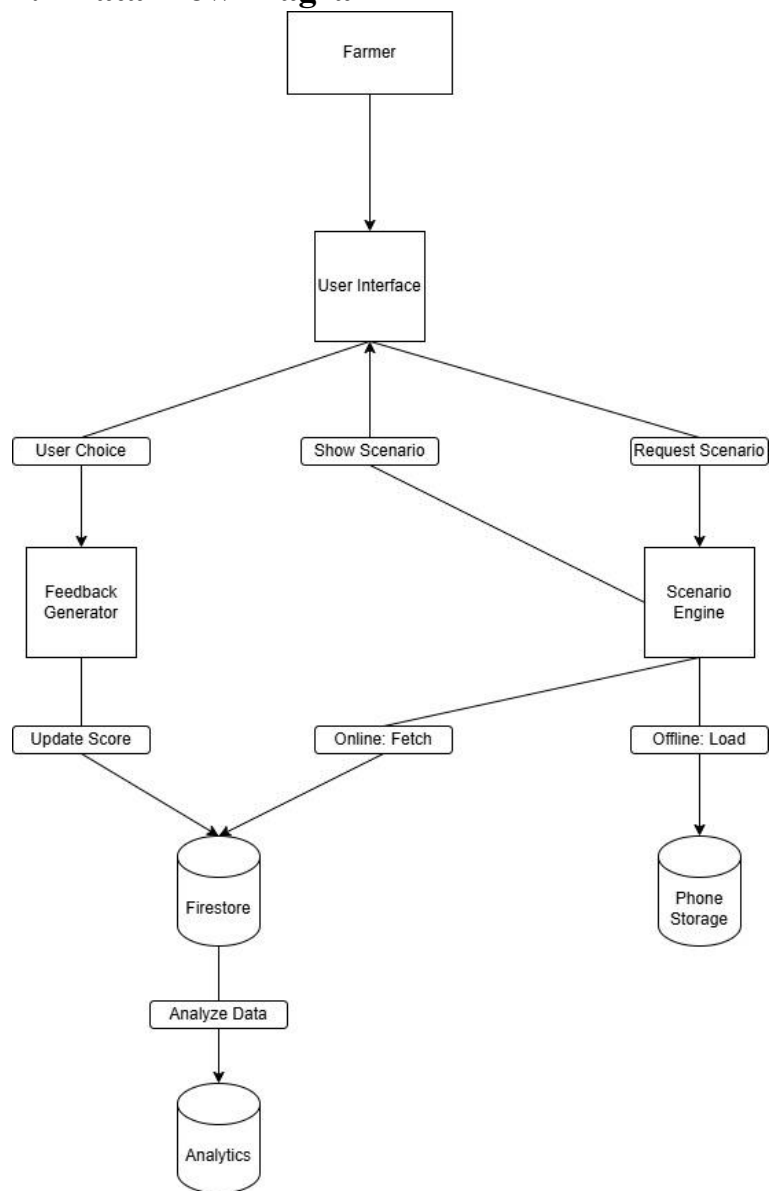
guided by real-world feasibility, laying a strong foundation for effective digital learning in rural Kenyan farming communities.

CHAPTER FOUR

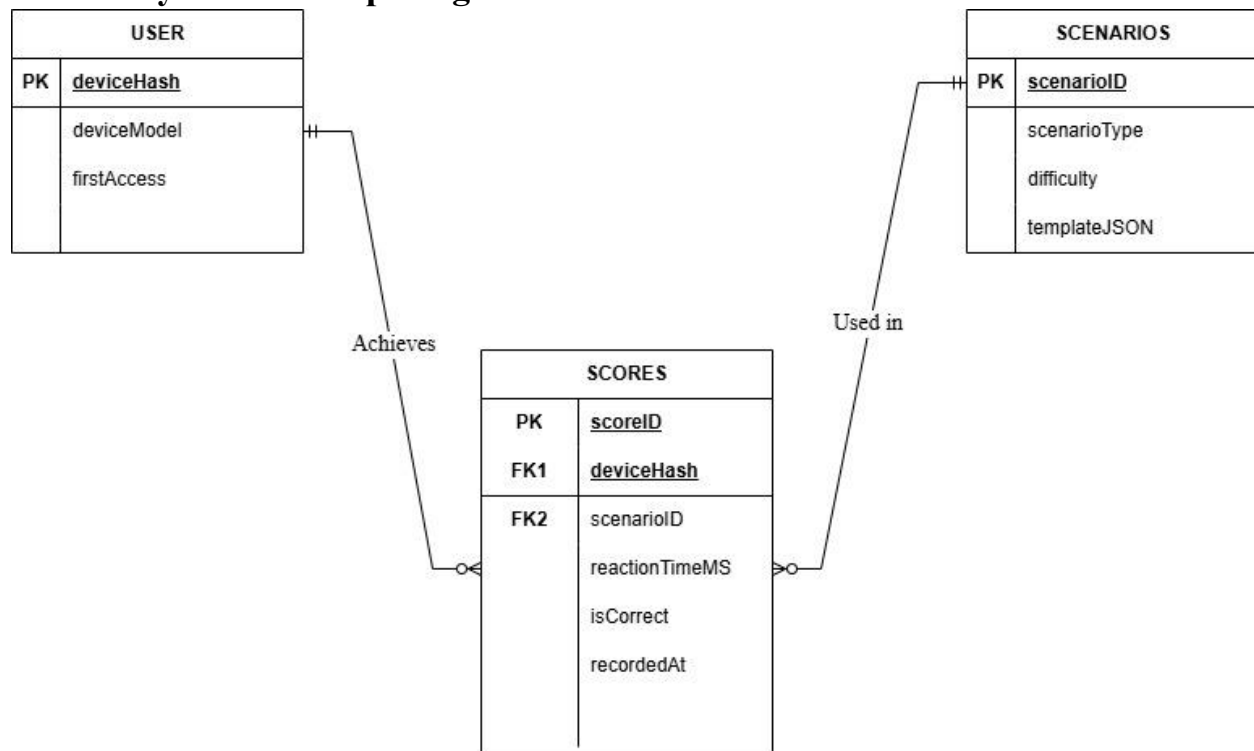
4.0 System Analysis and Design

FarmGuard mimics phishing scenarios through an interactive mobile GUI while collecting user learning data for analysis and tracking progress. The architecture is separated into the user interface, processing logic, and backend storage to provide improved scalability and maintainability.

4.1 Data Flow Diagram



4.2. Entity Relationship Diagram



CHAPTER FIVE

5.1 System Code

The following chapter contains major highlights of the code for the FarmGuard application

The main.dart

This is the entry point of the application. It contains imports of the other screens and tells the app where to start through the use of routes.

main.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'screens/login_screen.dart';
import 'screens/signup_screen.dart';
import 'screens/scenario_screen.dart';
import 'screens/how_to_play_screen.dart';
import 'screens/leaderboard_screen.dart';
import 'screens/feedback_screen.dart';
import 'screens/auth_wrapper.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const FarmGuardApp());
}

class FarmGuardApp extends StatelessWidget {
  const FarmGuardApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'FarmGuard',
      theme: ThemeData(useMaterial3: true),
      initialRoute: '/',
      routes: {
        '/': (context) => const AuthWrapper(),
        '/login': (context) => const LoginScreen(),
        '/signup': (context) => const SignupScreen(),
        '/scenario': (context) => const ScenarioScreen(),
        '/how-to-play': (context) => const HowToPlayScreen(),
        '/leaderboard': (context) => const LeaderboardScreen(),
        '/feedback': (context) => const FeedbackScreen(),
      },
    );
  }
}
```

```

    },
  );
}
}

```

Login Screen.

This is the login part of the application. Users who already have an account use their email and password to log into the application. Once they submit their credentials, the `auth_wrapper` file is the one responsible for authentication of the user, and if their credentials are correct, they are then taken to the home screen. Below is a highlight of the login screen code:

`login_screen.dart`

```

void _loginUser() async {
  setState(() => _isLoading = true);

  try {
    await _auth.signInWithEmailAndPassword(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );

    // Redirect to home screen
    Navigator.pushReplacementNamed(context, '/');
  } on FirebaseAuthException catch (e) {
    String message = '';
    if (e.code == 'user-not-found') {
      message = 'No user found for that email.';
    } else if (e.code == 'wrong-password') {
      message = 'Wrong password. Try again.';
    } else {
      message = 'Login failed. Please try again.';
    }

    ScaffoldMessenger.of(
      context,
    ).showSnackBar(SnackBar(content: Text(message)));
  } finally {
    setState(() => _isLoading = false);
  }
}

```

Signup Screen

This is part of the application where users who do not have an account and wish to play FarmGuard register. The credentials that are needed are: full names, email and password. The full names are essential in enabling the user to be saved in the Firestore and display their names on the leaderboard upon completion of the scenarios.

signup_screen.dart

```
class SignupScreen extends StatefulWidget {
  const SignupScreen({super.key});

  @override
  State<SignupScreen> createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  Future<void> _signup() async {
    try {
      // Create the user
      UserCredential userCredential = await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
          email: _emailController.text.trim(),
          password: _passwordController.text.trim(),
        );

      // Update the user's displayName
      await userCredential.user!.updateDisplayName(_nameController.text.trim());

      // Reload the user to ensure displayName is applied
      await userCredential.user!.reload();

      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Account created successfully!')),
      );
    }
  }
}
```

How to Play Screen

How to Play screen displays the important information that one needs to know in order to play the game. It provides the instructions such as avoiding to click suspicious links and always verifying the message source.

how_to_play_screen.dart

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('How to Play')),
    body: Padding(
      padding: const EdgeInsets.all(24.0),
      child: ListView(
        children: [
          Text(
            'Welcome to FarmGuard!',
            style: GoogleFonts.openSans(
              fontSize: 22,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 20),
          Text(
            'This game helps you learn how to spot phishing scams.',
            style: GoogleFonts.openSans(fontSize: 16),
          ),
          const SizedBox(height: 20),
          Text(
            'How it works:',
            style: GoogleFonts.openSans(
              fontSize: 18,
              fontWeight: FontWeight.w600,
            ),
          ),
          const SizedBox(height: 10),
          bulletPoint(
            'You\'ll see messages that look like real SMS or WhatsApp alerts.',
          ),
          bulletPoint('Decide if it\'s a scam or safe.'),
          bulletPoint(
            'Click "Ignore" for safe behavior or "Tap to Claim" if you trust it.',
          ),
        ],
      ),
    ),
  );
}
```

```

bulletPoint('Get instant feedback and learn how to stay safe.'),
const SizedBox(height: 20),
Text(
  'Tips:',
  style: GoogleFonts.openSans(
    fontSize: 18,
    fontWeight: FontWeight.w600,
  ),
),
const SizedBox(height: 10),
bulletPoint('Never accept or click any strange links.'),
bulletPoint('No one should ask you for money to get a subsidy.'),
bulletPoint('Check message sources before trusting them.'),
],

```

Scenario Screen

This is the file where the game takes places. It takes the current scenario from a list and displays it to the player. The player is given the two options, where they are supposed to choose the correct one.

scenario_screen.dart

```

void handleAnswer(String selectedOption) {
  // Determine if user was correct:
  // If scenario is a scam → correct choice is "Ignore"
  // If scenario is NOT a scam → correct choice is "Tap to Claim"
  final bool isCorrect =
    (scenario.isScam && selectedOption == 'Ignore') ||
    (!scenario.isScam && selectedOption == 'Tap to Claim');

  Navigator.pushNamed(
    context,
    '/feedback',
    arguments: {
      'result': isCorrect,
      'index': index,
      'score': isCorrect ? score + 10 : score,
    },
  );
}

```

```

// Extract source and message body
String source = '';
String messageBody = scenario.message;

if (scenario.message.contains(':')) {
  source = scenario.message.split(':')[0];
  messageBody = scenario.message.split(':').skip(1).join(':').trim();
}

return Scaffold(
  appBar: AppBar(
    title: Text('Scenario ${index + 1} of ${scenarioList.length}'),
    backgroundColor: const Color(0xFF34C759),
  ),
  body: Padding(
    padding: const EdgeInsets.all(20),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        const SizedBox(height: 20),

        // Sender box
        Container(
          padding: const EdgeInsets.symmetric(vertical: 8, horizontal: 12),
          decoration: BoxDecoration(
            color: Colors.blueGrey.shade50,
            borderRadius: BorderRadius.circular(8),
          ),
          child: Text(
            source.isNotEmpty ? source : "Unknown Sender",
            style: GoogleFonts.openSans(
              fontSize: 14,
              fontWeight: FontWeight.bold,
              color: Colors.blueGrey,
            ),
          ),
        ),
        const SizedBox(height: 10),

        // Message bubble
        Container(
          padding: const EdgeInsets.all(16),
          decoration: BoxDecoration(
            color: Colors.white,

```

```

        borderRadius: BorderRadius.circular(12),
        boxShadow: const [
          BoxShadow(color: Colors.black12, blurRadius: 5),
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(messageBody, style: GoogleFonts.openSans(fontSize: 16)),
          const SizedBox(height: 10),
          Text(
            "Today • 10:04 AM",
            style: GoogleFonts.openSans(
              fontSize: 14,
              color: Colors.grey,
            ),
          ),
        ],
      ),
    ),
  ),
),
const SizedBox(height: 30),

```

Scenario Data

This file contains all the scenarios in the application.

scenario_data.dart

```

final List<Scenario> scenarioList = [
  Scenario(
    message:
      "+254722334556: M-PESA - You've won a 10,000 KES fertilizer voucher!
Click: http://mpesa-freevoucher.com",
    isScam: true,
    correctAnswer: 'Ignore',
    feedbackCorrect: '✔ Correct! You ignored a classic phishing link.',
    feedbackWrong:
      '✗ This is a phishing link disguised as a prize. Never click such
links.',
  ),
  Scenario(
    message:

```

```

        "GovAlert: Your farming subsidy has been processed. No action needed.",
        isScam: false,
        correctAnswer: 'Tap to Claim',
        feedbackCorrect:
            '✔ Correct! This was a genuine government message. Genuine government
messages do not contain links that direct you to suspicious websites or
downloads',
        feedbackWrong:
            '✘ This message was safe. Government alerts often don't require
action.',
    ),
    Scenario(
        message:
            "FarmPay: Quickly claim urgent funds from gov relief program:
http://farmrelief-kes.info",
        isScam: true,
        correctAnswer: 'Ignore',
        feedbackCorrect: '✔ Correct! Suspicious links should always be ignored.',
        feedbackWrong:
            '✘ Suspicious links are classic signs of phishing. Do not click.',
    ),
    Scenario(
        message:
            "AGRO-DEALER: New offer! Pay KES 1500 now for 3 bags of fertilizer. Tap
to confirm.",
        isScam: true,
        correctAnswer: 'Ignore',
        feedbackCorrect: '✔ Correct! Scammers often push urgency for payments.',
        feedbackWrong:
            '✘ Scammers often push urgency for payments. Always verify before
acting.',
    ),
    Scenario(
        message:
            "KE-GOV: Your registration for e-farming workshop was successful. No
payments needed.",
        isScam: false,
        correctAnswer: 'Tap to Claim',
        feedbackCorrect:
            '✔ Correct! Legitimate workshops do not request money after
confirmation.',
        feedbackWrong:
            '✘ Legitimate workshops will not ask for payments after confirmation.',
    ),
];

```

Feedback Screen

This is the screen that provides accurate feedback depending on what the user chose when playing each scenario. It gives feedback that the player's choice was right if they chose the correct option and wrong if they chose the wrong option.

feedback_screen.dart

```
class FeedbackScreen extends StatelessWidget {
  const FeedbackScreen({super.key});

  Future<void> _saveScore(int score) async {
    final user = FirebaseAuth.instance.currentUser;
    if (user == null) return;

    final leaderboardRef = FirebaseFirestore.instance
      .collection('leaderboard')
      .doc(user.uid);

    final docSnapshot = await leaderboardRef.get();

    if (docSnapshot.exists) {
      final currentScore = docSnapshot.data()?['score'] ?? 0;
      if (score > currentScore) {
        await leaderboardRef.update({'score': score});
      }
    } else {
      await leaderboardRef.set({
        'name': user.displayName ?? user.email?.split('@').first ?? 'Unknown',
        'score': score,
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as Map;
    final bool isCorrect = args['result'] ?? false;
    final int index = args['index'] ?? 0;
    final int score = args['score'] ?? 0;
  }
}
```

```

final Scenario currentScenario = scenarioList[index];

return Scaffold(
  appBar: AppBar(
    title: const Text('Feedback'),
    backgroundColor: isCorrect ? Colors.green : Colors.red,
    centerTitle: true,
  ),
  body: Stack(
    children: [
      // Full screen confetti background if correct
      if (isCorrect)
        Positioned.fill(
          child: Lottie.asset(
            'assets/animations/confetti.json',
            repeat: false,
            fit: BoxFit.cover,
          ),
        ),

      // Feedback content
      ListView(
        padding: const EdgeInsets.all(24),
        children: [
          const SizedBox(height: 40),

          Center(
            child: Icon(
              isCorrect ? Icons.check_circle : Icons.error,
              size: 100,
              color: isCorrect ? Colors.green : Colors.red,
            ),
          ),

          const SizedBox(height: 24),

          Center(
            child: Text(
              isCorrect ? '✔ Well Done!' : '✘ You Got Scammed!',
              style: GoogleFonts.openSans(
                fontSize: 24,
                fontWeight: FontWeight.bold,
              ),
              textAlign: TextAlign.center,
            ),
          ),
        ],
      ),
    ],
  ),
);

```

```
),
```

Leaderboard Screen

Upon completion of all the scenarios, the player is taken to the leaderboard to view the number of points they have gained. Each scenario the player gets correct is 10 points. The player can also view where they stand against other players in order to have an overview of where they are in terms of being knowledgeable in phishing scams.

leaderboard_screen.dart

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_fonts/google_fonts.dart';

class LeaderboardScreen extends StatelessWidget {
  const LeaderboardScreen({super.key});

  Future<void> _saveScore(int score) async {
    final user = FirebaseAuth.instance.currentUser;
    if (user == null) return;

    final leaderboardRef = FirebaseFirestore.instance
      .collection('leaderboard')
      .doc(user.uid);

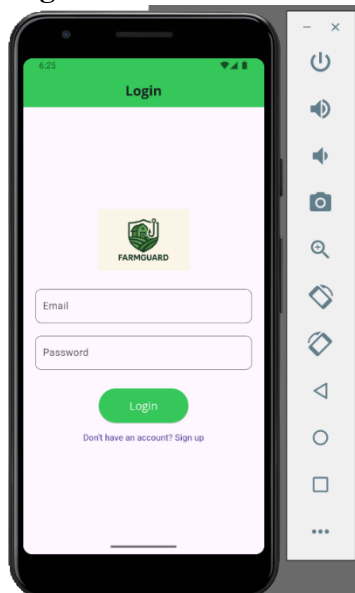
    final docSnapshot = await leaderboardRef.get();

    if (docSnapshot.exists) {
      final currentScore = docSnapshot.data()?['score'] ?? 0;
      if (score > currentScore) {
        await leaderboardRef.update({'score': score});
      }
    } else {
      // Create new entry
      await leaderboardRef.set({
        'name': user.displayName ?? user.email?.split('@').first ?? 'Unknown',
        'score': score,
      });
    }
  }
}
```

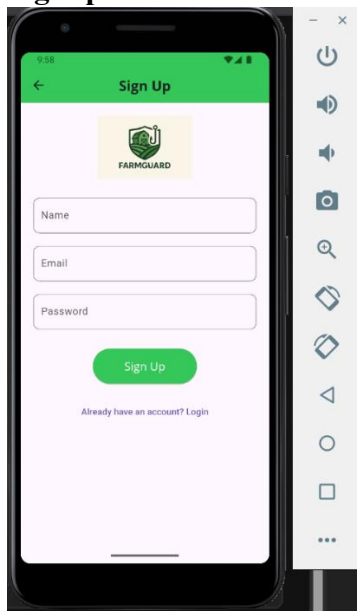
5.2 Screenshot Images

Here are the screenshots of the various screens of the FarmGuard application.

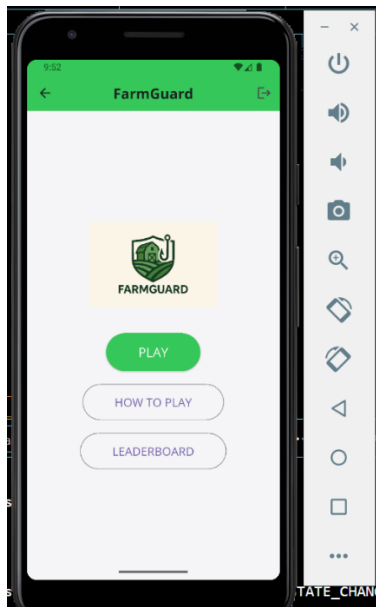
Login Screen



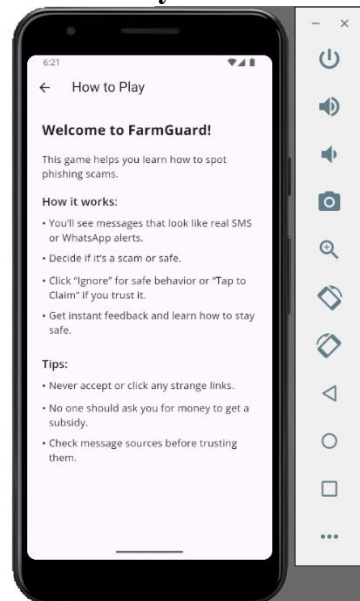
Signup Screen



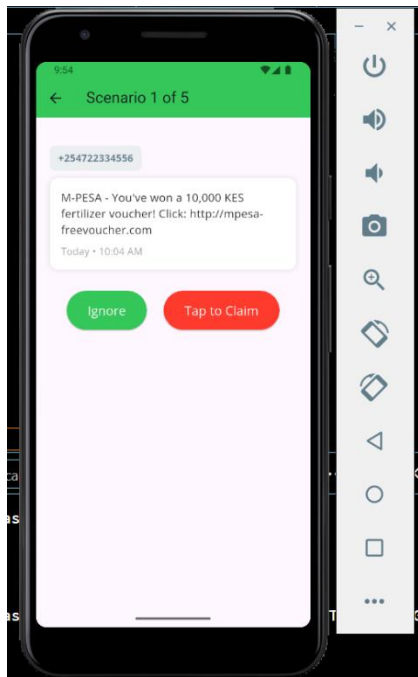
Home Screen



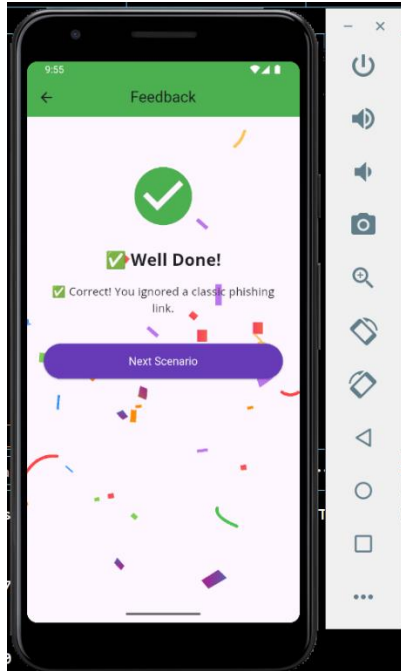
How to Play Screen



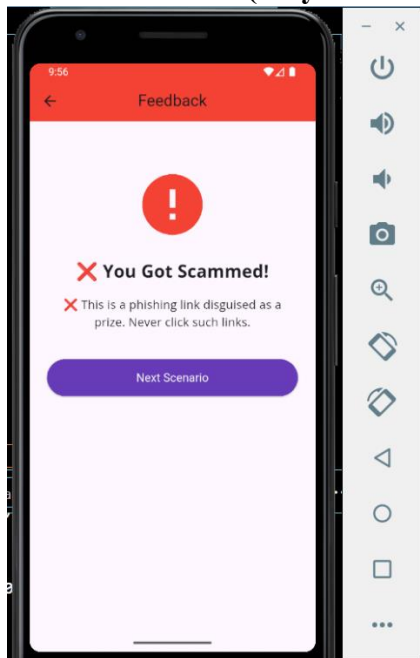
Scenario Screen



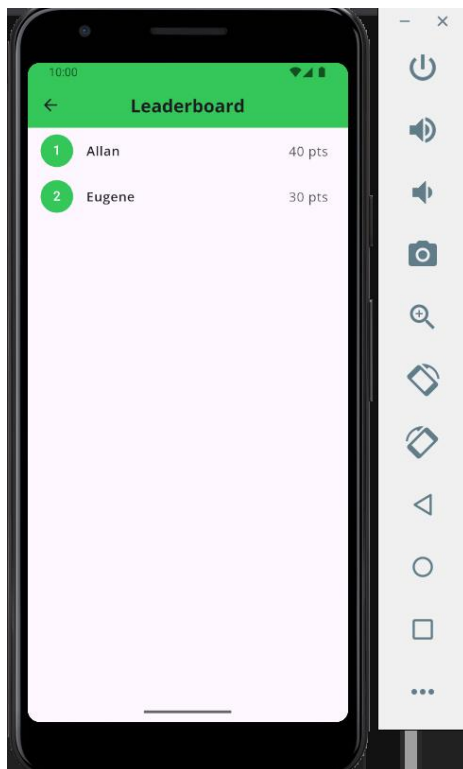
Feedback Screen (Player has chosen the right option)



Feedback Screen (Player has chosen the wrong option)



Leaderboard Screen



CHAPTER SIX

6.0 Conclusions

FarmGuard mobile application was successfully deployed and used as an education platform for rural farmers in Kenya to fight phishing scams. By the use of gamification elements and real-life examples of the most prevalent scams, the platform has been capable of sensitizing users and improving their ability to detect scams.

All the core modules of the system — authentication, scenario gaming, and leaderboard — were functioning in harmony in an attempt to provide a rich and engaging experience.

Firebase Authentication enabled secure and stable user management, and Cloud Firestore enabled real-time score updating and leaderboard rankings.

Lastly, FarmGuard bridges the gap between rural outreach and cyber education, offering an in-the-field solution for an evolving social engineering epidemic in agriculture.

6.1 Recommendations

While the project was a success, there are several areas of enhancement which can make FarmGuard more usable and available:

1. Increase Scenario Database

Add more scam scenarios, including emerging threats such as cryptocurrency investment swindles, fake grant applications, and deepfake voice scams.

Add language-specific variations to accommodate different Kenyan dialects and Swahili users.

2. Include Voice and Audio Guidance

Apply audio narration to situations to assist farmers with limited literacy.

Include warning tones for scam warning to improve the learning process.

3. Offline Mode Functionality

Local caching to be used so situations could be replayed offline, with results synced when back online.

4. Data Analytics Dashboard

Develop an admin dashboard to monitor user performance, usual mistakes, and revolts of scam trends.

5. Partnership with Agricultural Stakeholders

Partner with agricultural cooperatives, SACCOs, and government entities to promote FarmGuard through training sessions, farmer's forums, and rural ICT centers.

6.2 Final Reflection

This project has demonstrated the strength of technology to empower underprivileged groups.

Idea generation to deployment of FarmGuard required multi-disciplinary collaboration — combining software development, UI/UX design, cybersecurity expertise, and human-centered design thinking.

FarmGuard is a place to start to develop future initiatives in the prevention of cybercrime through education and to which other regions and languages could be expanded. It is a vital tool that can aid in the fight against digital cybercrime against rural farmers.

REFERENCES

2023-Economic-Survey.pdf. (n.d.).

Cyber Security Report Q1 2024-2025.pdf. (n.d.).

Tinubu, C. O., Falana, O. J., Oluwumi, E. O., Sodiya, A. S., & Rufai, S. A. (2023). PHISHGEM: A mobile game-based learning for phishing awareness. *Journal of Cyber Security Technology*, 7(3), 134–153. <https://doi.org/10.1080/23742917.2023.2167276>