SCHOOL OF TECHNOLOGY

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

UNIT: BIT 04105

TITLE:KCAU ONLINE VOTING SYSTEM

KAREN NJOKI KIARA

23/04687

Email: 2304687@students.kcau.ac.ke

(PROJECT TEST PLAN)

SUPERVISED BY: DR. EDWIN OMOL

FINAL YEAR PROJECT

This* PROJECT TEST PLAN *is submitted IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF the award of BACHELORS OF SCIENCE IN INFORMATION TECHNOLOGY in KCA University

Table of Contents

1.0 Introduction

This document outlines the comprehensive test plan and detailed test procedures for the KCAU Online Voting System, an Android application developed using Java and Firebase services (Authentication and Firestore). The system facilitates electronic voting for university elections, ensuring secure and reliable student participation.

1.1 Goals and Objectives

The primary objectives of the testing process are;

- To verify the correct functioning of the voting system
- Ensure that each student can vote only once per election
- Confirm that only one candidate can be selected per position
- Validate accurate vote storage in Firebase Firestore,
- Guarantee proper authentication and access control.

Additionally, the system's reliability, performance, and security will be thoroughly assessed.

1.2 Statement of Scope

Features to be Tested include user registration and login via Firebase Authentication, loading of election positions and candidates from Firestore, enforcement of one vote per position using RadioGroup controls, prevention of double voting, submission of votes to Firestore, redirection to a Thank You page post-voting, and display of appropriate Toast messages. Features excluded from testing are external integrations beyond Firebase services.

1.3 Major Constraints

The system's dependency on Firebase services necessitates an active internet connection. Testing will be conducted exclusively on Android devices and emulators. Firestore security rules may influence data accessibility, and real-user testing is limited due to the academic scope of the project.

2.0 Test Plan

2.1 Software (SCI's) to be Tested

The software under test is the KCAU Online Voting System Android Application, comprising components such as

- LoginActivity
- RegisterActivity
- VoteActivity
- ThankYouActivity
- Firebase Authentication
- Firebase Firestore Database.

Exclusions include Firebase's internal infrastructure and Android OS internal services.

2.2 Testing Strategy

The testing approach encompasses four levels: Unit Testing, Integration Testing, Validation Testing, and High-Order (System) Testing.

2.2.1 Unit Testing

Individual components will be tested independently, focusing on user authentication logic, vote submission methods, double voting prevention logic, candidate loading, and position loading methods. Tests will verify correct data retrieval from Firestore, proper RadioGroup selection behavior, accurate Toast message displays, and seamless navigation between activities.

2.2.2 Integration Testing

Integration tests will validate interactions between the Android app and Firebase Authentication, Firestore Database, and between VoteActivity and ThankYouActivity. The integration sequence includes authentication, loading positions, loading candidates, vote submission, and post-submission redirection.

2.2.3 Validation Testing

Validation ensures the system meets user requirements: successful student login, visibility of all election positions, single candidate selection per position, prevention of multiple votes by the same student, and correct vote storage under the "votes" collection.

2.2.4 High-Order Testing

System-level tests include Recovery Testing, Security Testing, Stress Testing, Performance Testing, and Alpha Testing, conducted by the developer and selected student testers.

2.3 Testing Resources and Staffing

Resources include Android Studio, Firebase Console, Android Emulator, physical Android devices, and internet connectivity. Staffing comprises the developer, Karen Kiara, and selected student testers.

2.4 Test Work Products

Deliverables consist of the Test Plan Document, Test Cases Document, Test Log, Bug Reports, and the Final Test Report.

2.5 Test Record Keeping

Test outcomes will be documented in a Test Log. Firebase Console will verify stored votes. Errors and bugs will be recorded and addressed, with screenshots maintained as testing evidence.

2.6 Test Metrics

Metrics tracked include the number of test cases executed and passed, defects identified and fixed, system response times, and the success rate of double-vote prevention.

2.7 Testing Tools and Environment

The test environment comprises Android Studio, Java SDK, Firebase Authentication,

Firebase Firestore, Android Emulator, and physical Android devices. Tools used include Logcat for error tracking, Firebase Console, and manual test scripts.

2.8 Test Schedule
Unit Testing will occur after each module's development, Integration Testing after module connections, Validation Testing upon full system completion, and High-Order Testing prior to final project submission.

3.0 Test Procedure
3.1 Software (SCI's) to be Tested
The KCAU Online Voting System Android Application is the focus, excluding external third-party systems and Firebase's internal backend services.

3.2 Testing Procedure
Manual and functional testing techniques will be employed. Each feature will be tested sequentially and verified via the Firebase Console.

3.2.1 Unit Test Cases
Component: VoteActivity

3.2.1.1 Stubs and/or Drivers
No external stubs are required; Firebase test data will be utilized.

3.2.1.2 Test Cases

- Load positions from Firestore
- Load candidates under each position
- Select one candidate per position
- Attempt submission without selecting all positions
- Submit vote successfully
- Attempt double voting

3.2.1.3 Purpose of Tests
To ensure correct UI rendering, enforce single selection per position via RadioGroup, verify accurate vote storage, and prevent double voting.

3.2.1.4 Expected Results
Positions and candidates load correctly; only one RadioButton per position is selectable; a Toast message appears if not all positions are selected; vote documents are created under elections → currentElection → votes → userId; users are redirected to ThankYouActivity after voting; double voting attempts trigger a "You have already voted" message.

3.2.2 Integration Testing

3.2.2.1 Testing Procedure
Login user, load election data, select candidates, submit vote, and verify vote in Firebase.

3.2.2.2 Stubs and Drivers
Firebase test database entries.

3.2.2.3 Test Cases and Purpose
Authentication combined with vote submission, Firestore write operations, and activity redirection.

3.2.2.4 Expected Results
Successful login, correct vote storage, prevention of duplicate votes, and proper navigation between activities.

3.2.3 Validation Testing

3.2.3.1 Testing Procedure
Simulate real student actions: register, login, vote, and attempt a second vote.

3.2.3.3 Expected Results
System behaves as required, disallowing double voting and multiple candidate selections per position.

3.2.3.4 Pass/Fail Criterion
Pass: All functional requirements met without critical errors.
Fail: Vote storage failures, double voting possible, or multiple candidates selectable per position.

3.2.4 High-Order Testing (System Testing)

3.2.4.1 Recovery Testing
Simulate internet disconnection during voting; system should display an error without crashing.

3.2.4.2 Security Testing
Attempt double voting and unauthorized access; access should be denied.

3.2.4.3 Stress Testing
Multiple users vote simultaneously; Firestore should handle requests without data loss.

3.2.4.4 Performance Testing
Measure vote submission time; expected to be under 3 seconds.

3.2.4.5 Alpha Testing
Selected students test the system before final deployment.

3.2.4.6 Pass/Fail Criterion
System passes if no major security flaws exist, no data corruption occurs, and all votes are accurately recorded.

3.3 Testing Resources and Staffing

Developer: Karen Kiara
Testers: Selected students
Tools: Android Studio, Firebase Console

3.4 Test Work Products

Test Cases, Bug Report Log, Test Log, and Final Testing Report.

3.5 Test Record Keeping and Test Log

A digital test log will maintain records of test dates, tester names, test case IDs, results (Pass/Fail), and remarks. All logs will be securely stored and backed up.