**CHAPTER ONE**

**INTRODUCTION**

**1.1 Background of the Study**

Voting is the process that allows the general public or the people to choose their leaders and articulate views on how they will be governed.

Real-time refers to a system in which input data is processed within milliseconds so that it is available virtually immediately as feedback to the process from which it is coming.

E-Voting refers the application of electronic technology to cast and count votes in an election. One of the fundamental mechanisms for democracy is election. It is the way to collect the public opinions to form a democratic government. The traditional process of election is quite tedious, time consuming and has a cumbersome procedure in preparation and collation phases. To overcome these difficulties Real-time Electronic voting system (REVS) is introduced. REVS continues to grow as long as the world becomes more dependable on the new technologies. REVS provides a lot of benefits than traditional voting systems. It tries to enable efficient and secure elections, inexpensive because its resources are reusable and does not require any geographical proximity of voters, and it provides better scalability for large elections.

In various Nigerian tertiary institutions, student elections are carried out every session. There are basically three arms of government in which student representatives or executives are usually elected; the Students Union Government (SUG), Departmental and Hostel levels. They are elected by students only. In other words, for any students-election, students of any given tertiary institution vote their fellow students who have shown interest in the above-listed posts. Elections have been in use to resolve various questions in the past 2000 years. Through the participation of a population, election allows public decisions to be made.

Election first started from the oral voting system to raising of hands, to the Kudavolai system (formerly utilized in the ancient India). In the ancient Greece, people would put either a white or a black ball/stone in a bucket. Oral voting was then substituted by the paper ballot first in Rome (139 BC) according to Douglas Jones.

And Nigeria still makes use of this paper-based voting systems. The voting systems had experienced continuous innovation which started as early as 1892 with the introduction of voting technology like the lever arch machine, the Optical-scan machine, and the punch card systems. Recent innovation saw evolutionary technology like the Direct Recording Electronics (DREs), Kiosk, Telephone, internet voting systems, and most recently is the mobile phone voting systems .

Most student electoral bodies since inception, still make use of obsolete paper-based voting systems characterized by filling manual forms.

**1.2 Statement of the Problem**

The voting/polling process by students of higher institution (Kaduna Polytechnic) seems to be cumbersome since there are thousands of students. So many cases of authentic students not participating in the voting process due to unfavourable voting time, conditions, environment, unbearable queue or the mammoth crowd at the voting place which is not accommodated in the period scheduled for voting. Before anyone can vote, he/she must be accredited. Taking a look at the accreditation process below:

**STEP 1**: Go to the polling centre with your students’ identification card and join the queue.

**STEP 2**: Present your card to the Independent Students Electoral Commission (ISEC) official for verification.

**STEP 3**: Your finger will be marked with ink to show that you have been accredited. And to think that this must be done for individual voters before they can actually vote could be really discouraging. Even after voting, malicious officers in charge eventually tamper with the results. It's after effect is violence, as the given population is dissatisfied with the tampered results. Such scenario could be totally avoided if students (voters) votes online using real-time e-voting application. This, allows the voters to vote from anywhere in the globe and see the result almost immediately the votes are casted, saving time and avoiding the cost of moving to polling area.

**1.3 Aim and Objectives of the Study**

**The aim of this project is to build a** mobile based real-time e-voting application

**Objectives**

The objectives of this research work are as follows:

1. The student data set will be extracted from the department based on criteria’s involving payment of school fees, as registration is not performed on the site since the registration is automated.
2. Modern technology like flutter will be employed in creating interactive user interface and experience.
3. To ensure effectiveness and efficiency several system test will be carried out.

**1.4 Scope of the Study**

This project pays close attention to the voting system in the Nigeria Tertiary Institutions already in existence, to ensure the students’ votes count, for transparency and fairness in the elective positions. Due to a large number of institutions, we will be limiting the scope of this work to computer science department ,Kaduna polytechnic. A brief description of the institution and their voting methods are given in the preceding subsections.

**1.5 Limitation of the Study**

Core limitations in carrying out the work include those related to logistics. The very major one includes:

1. Financial constraints
2. Time constraints

The materials reviewed as well as the scope of the study were also limitations. Though, the available resources and materials were optimized.

**1.6 Significance of the Study**

**The development of this project is not done for just a test of knowledge, principally, the significance of the work is to terminate the limitation of the voting to ballot papers,** vote-rigging during the election, overcrowded voting centres**, long after vote counting.** This real time mobile based voting app seeks to resolve the above problems **as these have been a serious problem to students and Electoral committee.**

**1.7 Project Organization**

**This project is made up of five chapters which will be organized in the following order;**

**Chapter one summarizes the introductory study on Electronic voting in real time, including the background of study, statement of the problem, aim and objectives, significance of the study, scope of the study and limitation of the study.**

**Chapter Two is an overview of related and relevant literature on the topic.**

**Chapter Three describes the research methodology used in this project. That is, specific methods which were used in order to achieve the objectives of the system.**

**Chapter Four displays the data analysis and gives concise details of how the system is to be implemented.**

**Chapter Five represents the summary, conclusion and recommendation of the project.**

**1.8 Definition of Terms**

**E-VOTING:** E-Voting also known as Electronic Voting, it is basically a voting procedure that allows a voter to cast their vote electronically through different machines and devices in an easy and secure manner. E-voting can eliminate fake votes, speed up the electoral process, increase accessibility and make voting more appropriate for citizens.

**REVS:** REVS stands for Real-time electronic voting system. It refers to a system in which input data is processed within milliseconds so that it is available virtually immediately as feedback to the process from which it is coming.

**SUG: SUG stands for Student Union Government. It is an organization in higher institutions responsible for protecting and defending the rights of students on campus. They organize leisure activities, provide welfare services and represent student’s political interest.**

**DRE: DRE stands for Direct Recording Electronics. This represents a voting technology** that electronically stores votes, and on which voters use interfaces (pushbutton, touchscreen, or dial) to record their votes. The votes are stored in a memory cartridge, diskette, or smart card and added to the votes of all other voters.

**ISEC: ISEC stands for Independent Students Electoral Commission. This body represents a committee of students who coordinate the elections of the student association.**

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 Introduction**

The purpose of this chapter is to show how the problem under consideration relates to prior research, current practice, or other fields of knowledge by citing relevant works by other researchers who have dealt with a similar issue. Furthermore, this chapter will include a synthesis of current research on the issue, highlighting areas of agreement, disagreement, and gaps in the literature, to establish the significance of the project topic in the field and to recommend opportunities for future study.

**2.2 Literature Review**

**Over the previous years, society and numerous days by day parts of life have been changed technologically.**

**Selvarani et al (2017) noted in a study on secure voting system through SMS and using smart phone application. The study** proposed a system developed to select their candidate through smart phone application. This process consists of three steps: online registration of voter, vote casting of voter and display of results, through the concept of SMS (short messaging service). **Voters can more effectively cast their ballots online at any time and from any location. Since every vote counts and each of those votes is significant, it is crucial to increase security throughout the entire process.** This also prevents the double voting since the databases are maintained in the encrypted form.

Kshetri & Voas(2018) carried out a study on Block chain-enabled e-voting (BEV). The study pointed out that (BEV) could reduce vote fraud and increase voters access. Eligible voters cast a ballot anonymously using a computer or smartphone. BEV uses an encrypted key and tamper-proof personal IDs.

**Okpara R. et al (2018) carried out a study on development of a mobile android voting application for tertiary institutions in Nigeria. The study's goal was to find solutions to issues with voting in tertiary institutions, such as vote rigging during elections, congested voting locations, inaccessible and unsafe polling places, and unskilled staff, among others. The system was implemented as a mobile app based voting system, using Web UI, Server and Android User Interface.**

**Amit & Abdullahi (2019) carried out a study on design and development of real-time E-voting system with high security features. The point of the study was to exhibit an electronic voting system (E-Voting) to be connected to organizations constituent body. The software application was developed utilizing web API’s and the concept of Dynamic systems development method (DSDM) was used together with object oriented methodology.**

Visalakshi et al (2020) carried out a study on QR code voting in election. The study’s goal was to suggest **a real-time Quick Response code capture system for use with electronic voting on our mobile device. Each voter is provided with a unique QR code which is scanned by a mobile scanner application and used by voters to cast their votes.** **The software’s aim is to eliminate the use of the existing voter machine, and to reduce errors done during voting.**

Ajish & AnilKumar (2021) performed a study on secure mobile internet voting system using biometric authentication and wavelet based AES. The study looked at presenting a secure mobile internet voting system in which a biometric method authenticates the voter. The biometric image can either be encrypted at the mobile device or sent to the server or process the biometric image at the mobile device to generate the biometric template and send it to the server. The wavelet-based AES algorithm was used to speeds up the encryption process and reduces the mobile devices CPU utilization.

**Sherine et al (2022) performed a study on development of an efficient and secured e-voting mobile application using android. In the aforementioned study an Android app was made that has a 3-step security procedure before voting in order to prevent phishing attempts. With the use of a mobile device, students may cast their votes online at any time and from any place. The application was made and published using Android Studio. The following is a development that is in the process of being developed. The outcome of this research is the development of a student-friendly mobile application that provides them with three degrees of voting security.**

**2.3 Summary of Related Literature Reviews**

|  |  |  |
| --- | --- | --- |
| **Author & Year** | **Title & Description** | **Merit and Demerits** |
| Selvarani et al (2017). | Secure Voting System Through SMS and using Smart Phone Application.  The study proposed a system developed to select their candidate through smart phone application. And they can cast their votes and display results, through the concept of SMS (short messaging service). | Voters can more effectively cast their ballots online at any time and from any location with their mobile phones.  The system is strictly mobile based. |
| Kshetri & Voas (2018) | Block Chain-Enabled e-Voting (BEV)  The study pointed out that (BEV) could reduce vote fraud and increase voter’s access. Eligible voters cast a ballot anonymously using a computer or smartphone. | The system will reduce vote fraud and increase voters’ confidence and access.  The system may lack scalability due to influx of users. |
| Okpara R. et al (2018) | Development of a Mobile Android Voting Application for Tertiary Institutions in Nigeria.  The study's goal was to find solutions to issues with voting in tertiary institutions, such as vote rigging during elections, congested voting locations, inaccessible and unsafe polling places, and unskilled staff, among others. | Voting can be carried out at any time and from the comfort of one’s home.  It requires internet connection. |
| Amit & Abdullahi (2019) | Design and development of real-time e-voting system with high security features.  The point of the study was to exhibit an electronic voting system (E-Voting) to be connected to organizations constituent body. The software application was developed utilizing web API’s and the concept of Dynamic systems development method (DSDM) was used together with object-oriented methodology. | Users can view polling results in real-time.  The system may lack scalability due to influx of users. |
| Visalakshi et al (2020) | QR Code Voting in Election.  The study’s goal was to suggest a real-time Quick Response code capture system for use with electronic voting on our mobile device. Each voter is provided with a unique QR code which is scanned by a mobile scanner application and used by voters to cast their votes | The system will minimize the amount of time used during voting exercises.  Poor user interface. |
| Ajish & AnilKumar (2021) | Secure Mobile Internet Voting System Using Biometric Authentication and Wavelet Based AES.  The study looked at presenting a secure mobile internet voting system in which a biometric method authenticates the voter. The biometric image can either be encrypted at the mobile device or sent to the server or process the biometric image at the mobile device to generate the biometric template and send it to the server | The system offers high security with biometric authentication.  The system is strictly mobile based |

**CHAPTER THREE**

**METHODOLOGY AND DESIGN**

**3.1 Introduction**

**Research methodology should be effective enough to ensure fulfilment of the defined objectives through particular components such as techniques of data collecting and design. Methodology is a process of rigorous study or inquiry specifically in order to uncover new fact or information. By combining the aforementioned, it will be possible to create a platform that is extremely reliable, quick, and practical.**

**3.2 Method of Data Collection**

**It is essential to acquire data and facts about the current system before implementing any system since one has to understand what is happening. Two techniques were used to conduct this study.**

1. **Observation of the Work Environment**
2. **Documentation**

**3.2.1 Observation of the Work Environment**

**This strategy was used to collect information and data for this study by observing how the manual system functioned. Through close inspection, the system's obvious flaws were found.** **The context in which the observation is conducted can be changed in a variety of ways by using the observational technique.**

**3.2.2 Documentation**

A secondary form of data acquisition is documentation. Journals, manuals, previous projects, publications, and other sources are used in this approach. This type of data collecting is employed because it provides a foundation for comparison with previous research. This includes the internet, a tool for gathering data. The internet was utilized to find information on topics that seemed challenging or unclear.

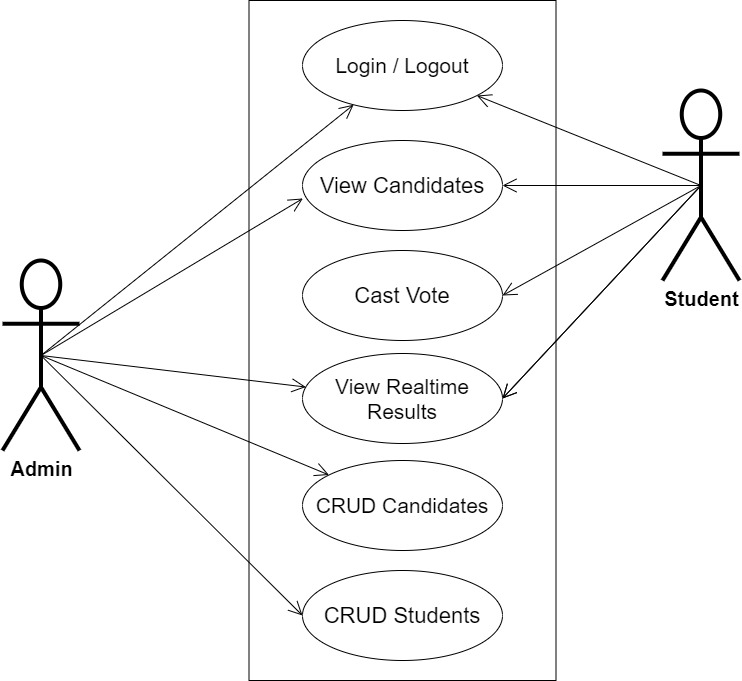
**3.3 System Modelling**

Any interaction between groups of components that work together to accomplish a single goal is referred to as a system. Visual models of the object-oriented software-intensive systems may be made using a set of graphic notation techniques that are part of the Unified Modeling Language, which is employed in this modern system design.

Use Case Diagram, Class Diagram, and Activity Diagram are some of the UML diagrams used in this new design.

**3.3.1 Use Case Diagram**

In systems analysis, use cases diagrams are a powerful tool for identifying, outlining, and organizing system requirements. The use case diagrams show a variety of potential interactions between people and systems in a certain context that are connected to a specific objective. Use cases are collections of interactions between systems and users.

The objective of a Use Case Diagram is to offer a graphical overview of a system's functionality in terms of actors, their goals (expressed as use cases), and any dependencies between those use cases.

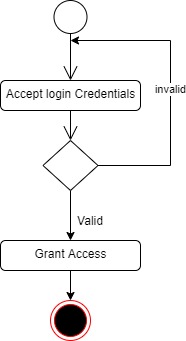
**Fig 3.1 System Use Case Diagram**

**3.3.2 Activity Diagram**

An activity diagram, like a flowchart or a data flow diagram, visually illustrates a series of events or the flow of control in a system, but it acts more like an enhanced version of both.

**Login**

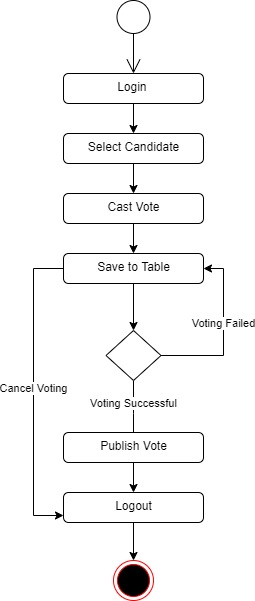
The process for gaining access to the system is depicted in the diagram below; in order to gain access, the email address and password must be accurate.



**Fig 3.2 Login Activity Diagram**

**Voting**

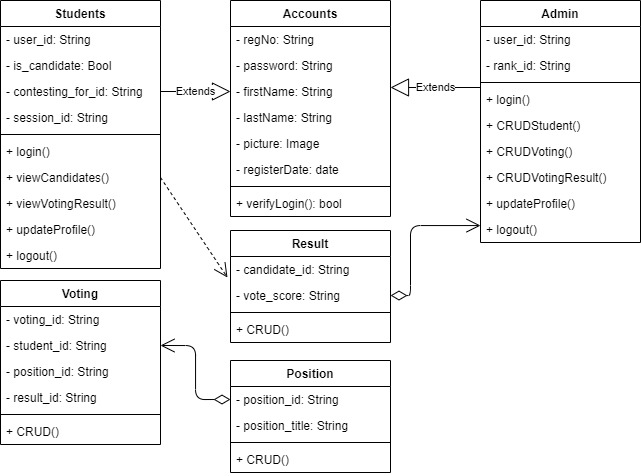
The process for voting for a candidate is depicted below, to vote one has to select from each category of available positions.



**Fig 3.3 Voting Activity Diagram**

**3.3.3 Class Diagram**

Class diagrams are visual representations of a system's static structure and composition that adhere to the Unified Modelling Language principles (UML). It is one of the most often used UML diagram kinds. Class diagrams make it simpler to explain all of the classes, packages, and interfaces that comprise a system, as well as how these components are interconnected.



**Fig 3.4 Class Diagram**

**3.4 Database Design**

Input specification is the logical explanation of how data is stored in the computer's memory. SQL standards are vital for guaranteeing that structured data is uniform and independent of applications due to the flexibility experienced when using the system, as well as the simplicity of accessing and reading the data and insuring applicability throughout the internet. The following are some of the input specifications used in this project effort.

1. Accounts Table: contains basic information about all system users.
2. Voting Table: contains the voting record for the system.

**Table 3.1 Account Input Specification Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **LENGTH** | **DESCRIPTION** |
| Reg\_no | Varchar | 150 | Reg\_no for login (case insensitive) |
| Password | Varchar | 150 | Access Code (case sensitive) |
| Firstname | Varchar | 150 | User first name |
| Lastname | Varchar | 150 | User last name |
| Picture | Varchar | 100 | User profile picture |
| RegisterDate | Varchar | 100 | The date the user was registered |
| acct\_id | Varchar | 64 | A unique string for identifying users |

**Primary key:** acct\_id

**Table 3.2 Voting Input Specification Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **LENGTH** | **DESCRIPTION** |
| Student\_id | Varchar | 150 | Reference to the student model |
| Position\_id | Varchar | 150 | Reference to the position model |
| Result\_id | Varchar | 150 | Reference to the result of the voting |
| Voting\_id | Varchar | 64 | A unique string for identifying votes |

**Primary key:** voting\_id

**3.5 Output Design**

This declares and displays the outcome of the given input. The automated system's output is dependent on its input. The output specification is listed below.

**Table 3.3 Account Output Design Table**

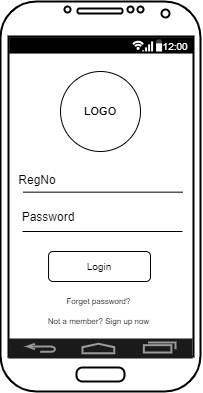
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Reg\_no** | **Password** | **Firstname** | **Lastname** | **RegisterDate** | **Picture** | **Acct\_id** |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX | XXXX | XXXX | XXXX |

**Table 3.4 Voting Output Design Table**

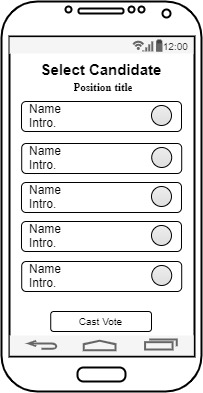
|  |  |  |  |
| --- | --- | --- | --- |
| **Student\_id** | **Postion\_id** | **Result\_id** | **Voting\_id** |
| XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX |
| XXXX | XXXX | XXXX | XXXX |

**3.6 Input & User Interface Design**

This is a graphic depiction of the system interface; it will be designed to be user-friendly, responsive, and visually beautiful. Furthermore, it will be fully secured, thus logging in will be required to see various levels of the information. To help with the designs, a mid-fidelity wireframing program called Draw.io is employed.



**Fig 3.5 User Login Screen**



**Fig 3.6 Voting Screen**

**3.7 System Requirements**

Every piece of software that is created has preset system requirements that it must meet in order to run at its best. The system requirements, on the other hand, are the bare minimum of hardware and software needed for the system to operate as intended.

**3.7.1 Hardware Requirement**

System Hardware Requirement Include:

1. Minimum of 8 GB of RAM (Random Access Memory) installed.
2. Minimum of intel core i5 required.
3. Minimum of 250GB HDD (Hard Disk Drive).

**3.7.2 Software Requirement**

The software requirements include:

1. At least windows 10 OS (Operating System).
2. Flutter Installation.
3. Vs. Code / Android studio installation.
4. Emulator installation.

**3.8 Choice of Programming Language**

The proposed design will be implemented using flutter for its user interface (frontend) while Django will be used for the backend, Sqlite3 will be used for its database and Django REST Framework for it REST-full API, combination of the above modern technology forms the technology for this research work.

**CHAPTER FOUR**

**System Implementation Evaluation**

**4.1 Introduction**

This section describes in detail how the new system will be implemented in order to assure its efficacy. It illustrates instances of functional (new) systems as well as how the system will be implemented.

* 1. **System Testing and Evaluation**

The developed system should be tested for a variety of reasons. For example, only via testing will we be able to detect and address any problems in the new system. Unit and integration testing were used in this project to confirm the design's efficacy and efficiency, as well as to ensure the new system satisfies its functional requirements and is error-free.

**Unit Testing**

specific units or single components of the system are examined individually in this part to confirm that specific phases function properly and without problems.

**Integration Testing**

The program was tested via integration testing, in which all of the components were integrated and worked as one. The connection between the different components was examined to ensure that they are correctly integrated and that the units can function as a unit.

**4.3 System Installation**

In order to use the proposed application on any computer system, the following steps need to be taken:

1. Make sure, android studio, JDK, and Android emulator are installed on the system.
2. Copy your project folder to any location of your choice.
3. Open the project folder in Visual Studio Code
4. In the terminal run “flutter pub get” to get all the dependencies in the pubspec.yaml file
5. Select the Android emulator as the device to be used.
6. Locate the main.dart file and run the file in debug mode.

**4.4 Security Measures**

Since the scope of the application is public, literally all the information is made available to any user (students and admin), but some functionalities are restricted to the admin, functionalities that have to do with creating the student accounts, creating the candidates, managing the voting periods etc are restricted from the general student. The restriction is carried out by using passwords when the application is accessed.

**4.6 Sample Outputs**

These describe and give the pictorial representation of the program or software; it shows and gives clear understanding of the design, and displays all the interfaces.

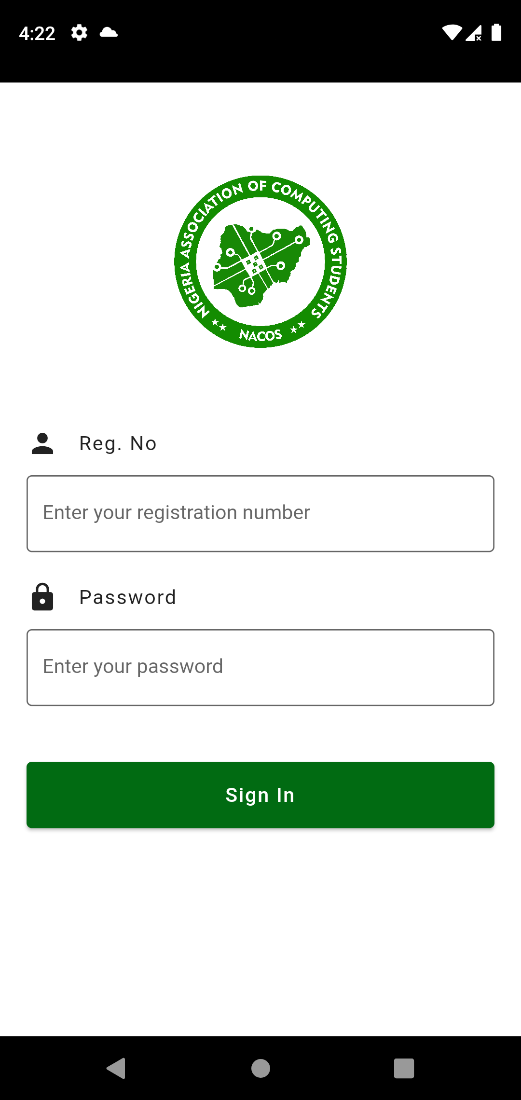


Fig 4.6.1: Splash Screen Fig 4.6.2: Login Screen

**Fig 4.6.1 Splash Screen**: This is the first screen displayed to every user that wishes to make use of the application.

**Fig 4.6.2 Login Screen**: The screen grants users access (students, admin) to the application only if the correct credentials are provided.

**Administrator Screen**

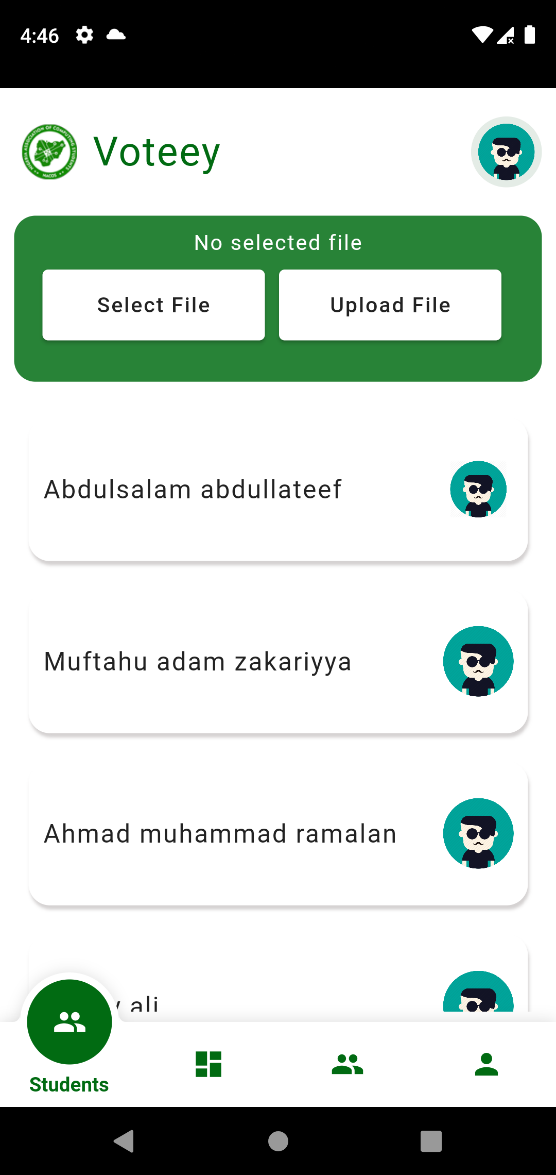
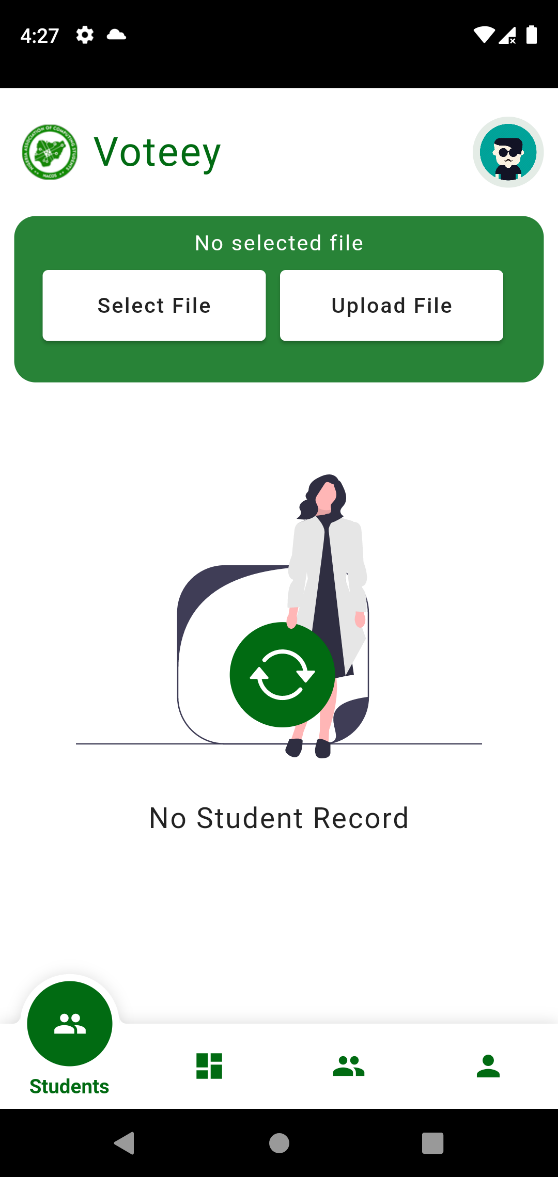
****

Fig 4.6.3: Admin Student List (empty) Fig 4.6.4: Admin Student List (not empty)

**Fig 4.6.3 Admin Student List (empty)**: The application provides a user friendly asthetic interface to show that students record was not found.

**Fig 4.6.4 Admin Student List (not empty)**: The screen show all the system registered student eligible for voting, it also provide the functionality that properly handles the uploading of the student accounts, it handles possible exception that could be encountered.

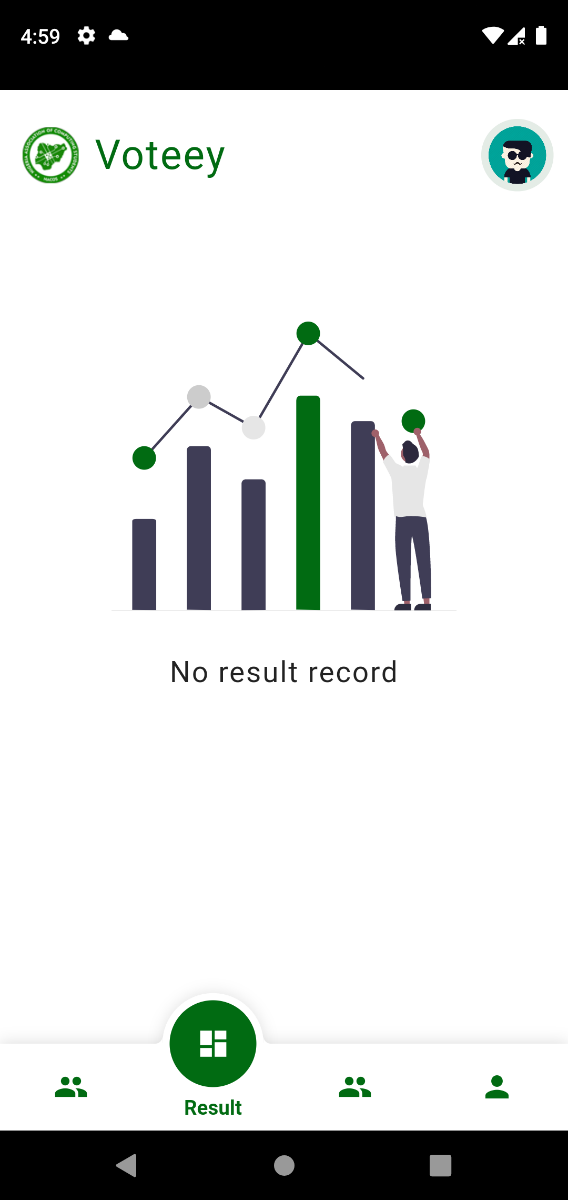
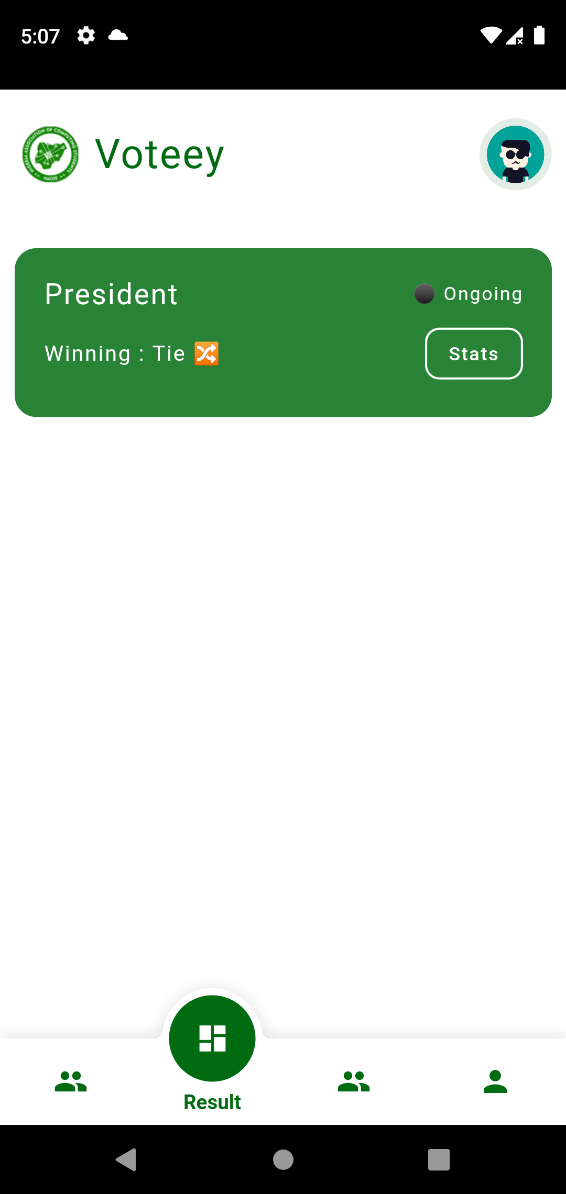


Fig 4.6.5: Result List (empty) Fig4.6.6: Result List (not empty)

**Fig 4.6.5 Result List (empty)**: The application provides a user friendly aesthetic interface to show that a voting result record was not found, it happens when the voting is not yet live.

**Fig 4.6.6 Result List (not empty)**: The screen shows all the applied positions coupled with real-time statistics of the voting, which can be viewed on the click on the “stats” button.

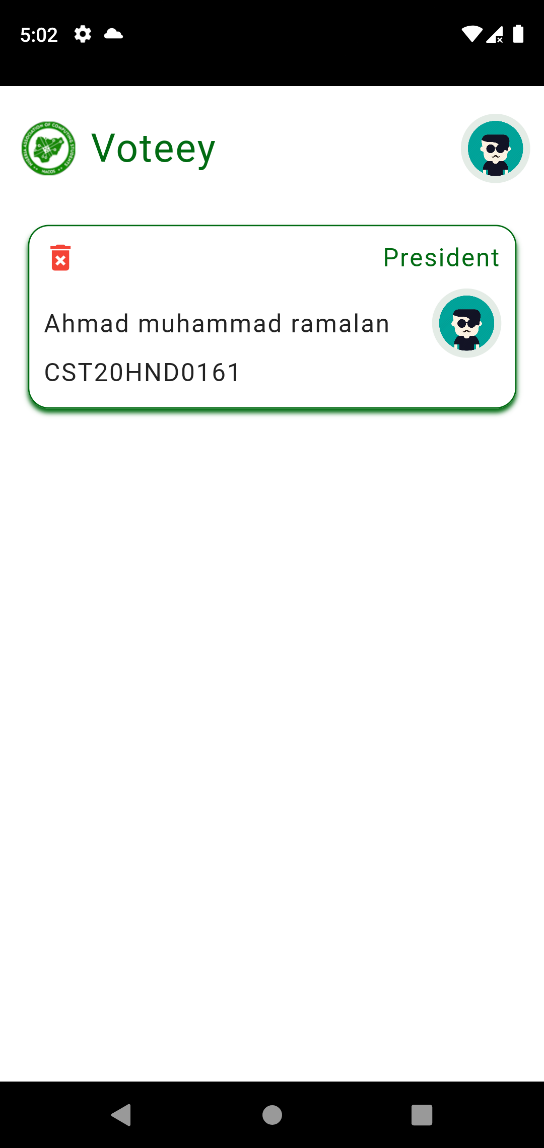
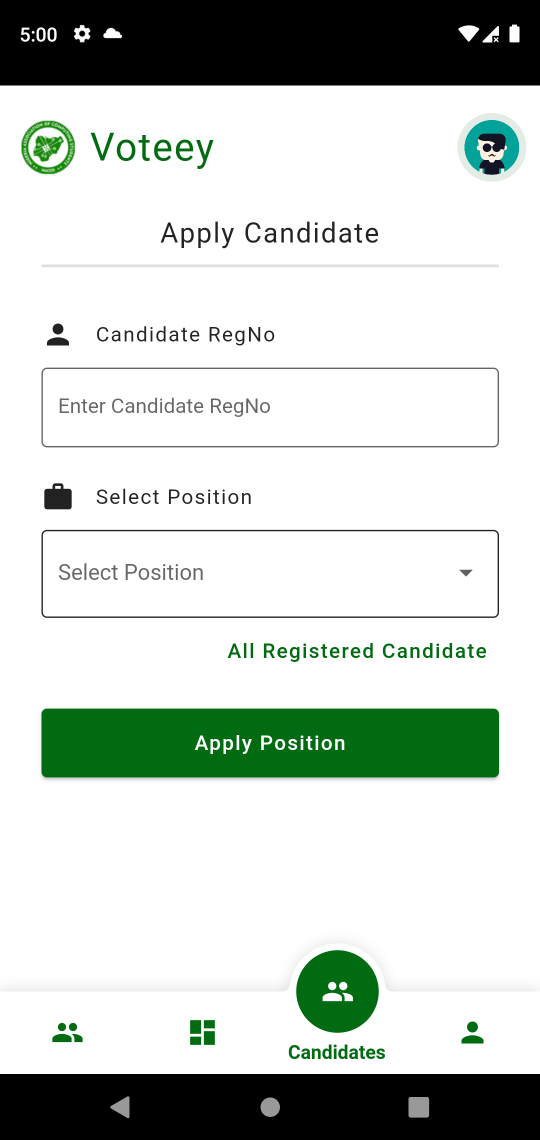


Fig 4.6.7: Create Candidate Fig 4.6.8: All Candidates

**Fig 4.6.7 Create Candidate**: The screen provides the functionality of applying candidates for a particular position, this is done by entering the candidate registration number and selecting the position then clicking on the apply position button.

**Fig 4.6.8 All Candidates**: The screen displays all the applied candidates alongside the position they applied for, the candidate can be deleted if needs be.

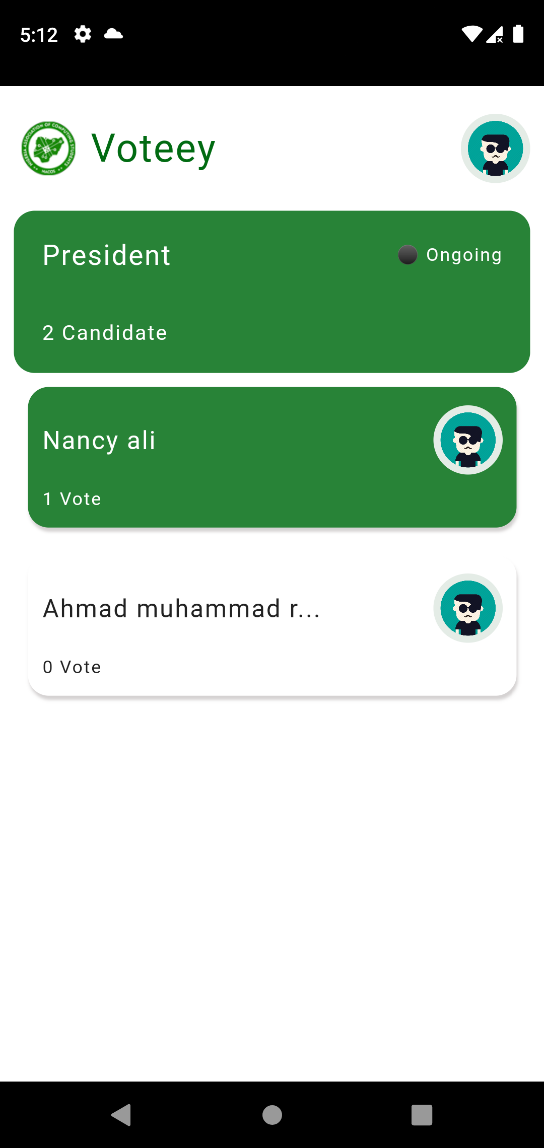
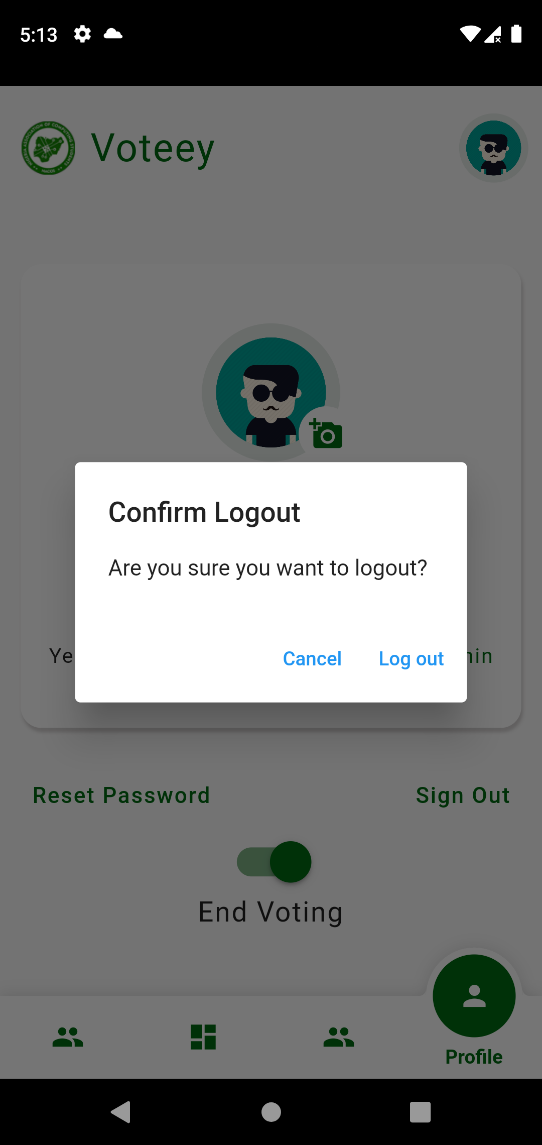


Fig 4.6.9: Result Statistics Fig 4.6.9.1: Logout Screen

**Fig 4.6.9 Result Statistics**: This screen displays the current live voting in a particular position, the voting result possesses real-time features

**Fig 4.6.9.1 Logout Option**: This option logs out the user from the application, the user has to reauthenticate again to gain access to the system

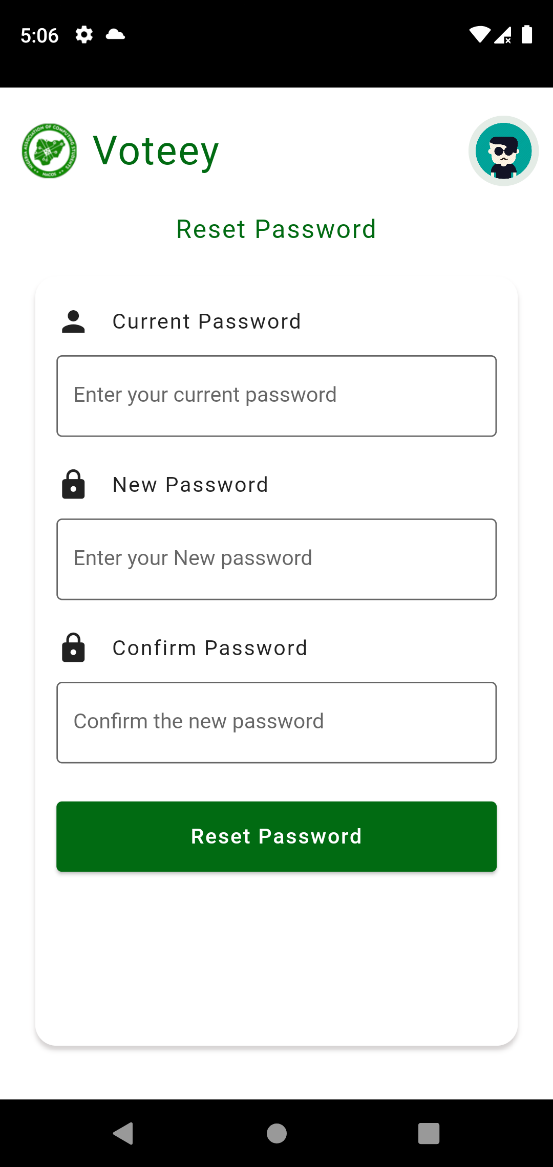
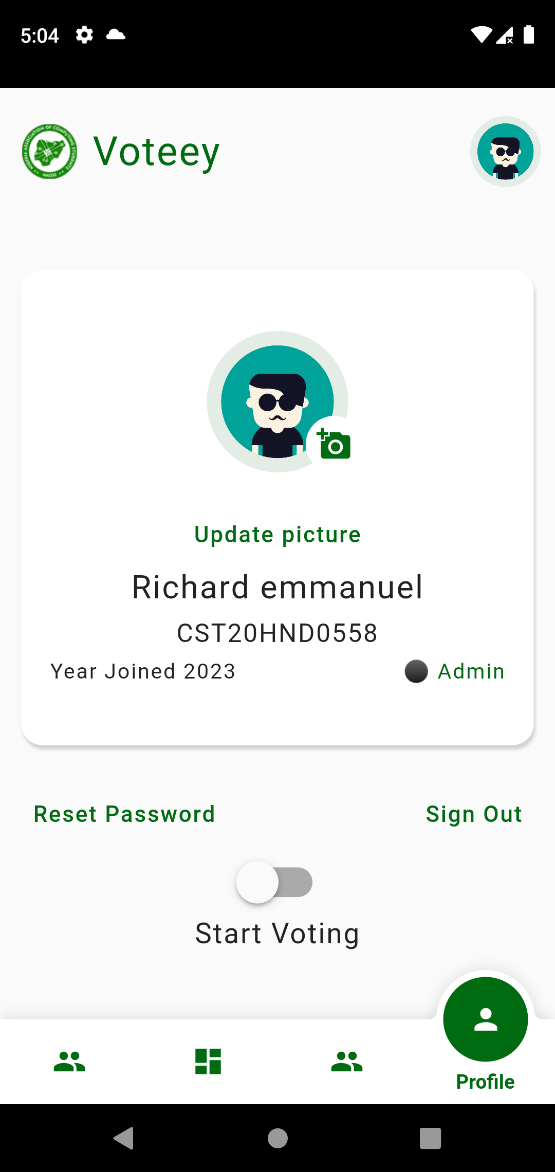


Fig 4.6.9.2: Update Profile Fig 4.6.9.3: Change Password

**Fig 4.6.9.2 Update Profile**: This screen enables users to update their account picture

**Fig 4.6.9.3 Change Password**: This screen enables users to update their account password

**Student Screens**

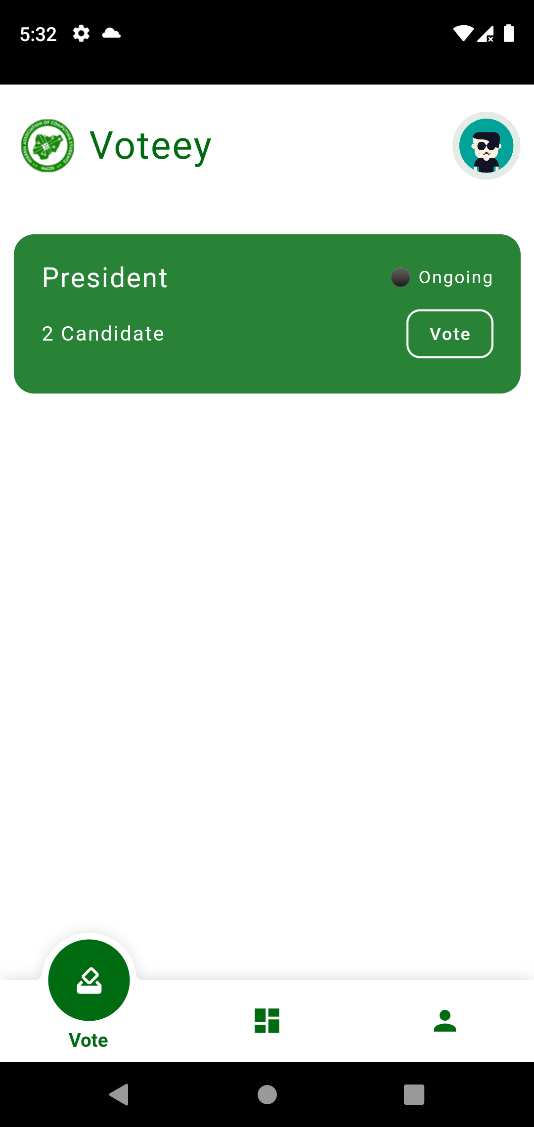
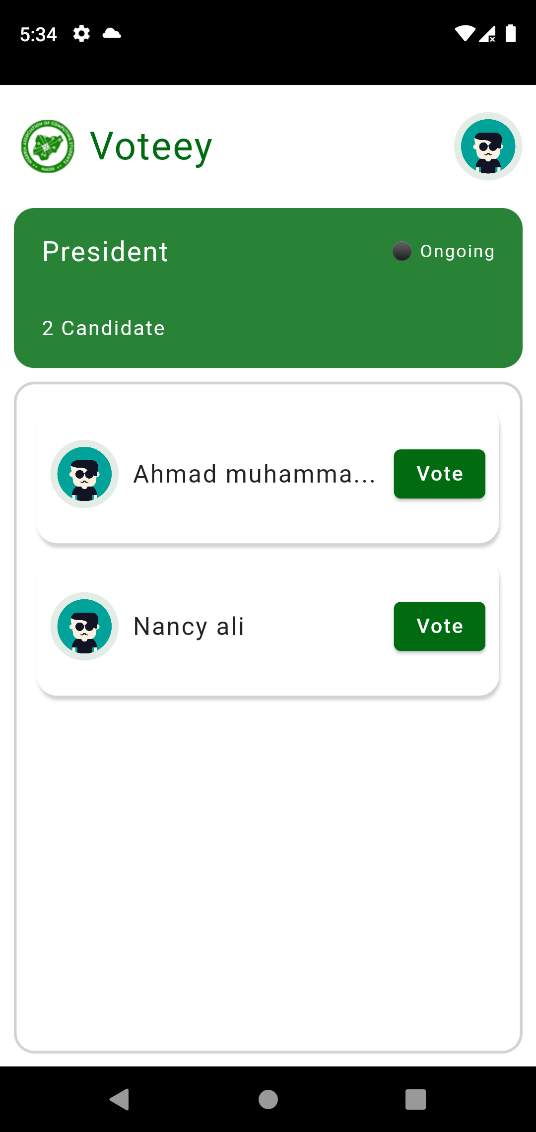


Fig 4.6.9.4: Voting List Screen Fig 4.6.9.5: Cast Vote Screen

**Fig 4.6.9.4 Voting List Screen**: This screen depicts the various position that was applied for so that student can easily find the position they want to vote for.

**Fig 4.6.9.1 Cast Vote Screen**: This screen list the various candidate that applied for the selected positions so that students can cast their votes, the system handles possible exceptions that may occur.

**CHAPTER FIVE**

**Summary Conclusion and Recommendation**

**5.1 Summary**

The voting process in Nigerian tertiary institutions, including Kaduna Polytechnic, faces significant challenges such as cumbersome procedures, low student participation, and potential vote tampering. To address these issues, a mobile-based real-time e-voting application is proposed. This application aims to streamline the voting process, increase student engagement, and ensure transparency and fairness in electing student representatives. By leveraging modern technology and employing an interactive user interface, the application seeks to overcome the limitations of traditional paper-based voting systems. Furthermore, the project emphasizes the importance of accurate student data extraction, system testing for effectiveness and efficiency, and the significance of improving the overall voting experience. By implementing the mobile-based e-voting application, the project aims to create a more inclusive, accessible, and efficient electoral system for Nigerian tertiary institutions.

**5.2 Conclusion**

In conclusion, the research project on the development of a mobile-based real-time e-voting application for Nigerian tertiary institutions has successfully addressed the limitations and challenges associated with traditional voting systems. One of the key strengths of this solution is its real-time functionality, allowing students to cast their votes and receive immediate feedback on the results. This real-time aspect of the e-voting system enhances the efficiency and effectiveness of the electoral process, eliminating the need for cumbersome manual procedures and lengthy vote-counting periods. The significance of this study lies in its potential to revolutionize the voting process by introducing real-time capabilities, ensuring a timely and accurate representation of students' choices. The implementation of such a real-time e-voting application can foster greater student engagement, increase voter turnout, and promote transparency and fairness in the electoral processes of Nigerian tertiary institutions. By embracing real-time e-voting solutions, educational institutions can advance democratic practices, enhance participation, and facilitate informed decision-making within their student communities.

**5.3 Recommendation**

Based on the research and development of the mobile-based real-time e-voting application for Nigerian tertiary institutions, the following recommendations are made:

1. Implementation of Real-Time E-Voting: Educational institutions, particularly Nigerian tertiary institutions, should consider implementing real-time e-voting systems for their student elections. The adoption of such technology will enhance the efficiency, accuracy, and transparency of the voting process, allowing students to cast their votes from anywhere and receive immediate results.
2. Security Measures: Given the sensitivity of elections, it is imperative to prioritize the security of the e-voting application. Advanced security measures, such as encryption, multi-factor authentication, and regular security audits, should be implemented to safeguard against potential cyber threats.
3. Continuous Evaluation: After implementation, regular evaluations and feedback mechanisms should be established to monitor the application's performance and identify areas for improvement. This will ensure that the real-time e-voting system remains effective and up-to-date.
4. Scaling Up: If proven successful and secure, the real-time e-voting system could be considered for adoption beyond tertiary institutions, potentially extending to local government or national elections. However, rigorous testing and collaboration with relevant authorities would be essential before such expansion.

By embracing these recommendations, Nigerian tertiary institutions can revolutionize their electoral processes and pave the way for a more democratic, inclusive, and technologically advanced voting experience for their students.

**REFERENCES**

Kshetri, N., & Voas, J. (2018). Blockchain-Enabled E-Voting. IEEE Software, 35(4), 95–99. doi:10.1109/ms.2018.2801546

Mishra, A., & Ofujeh Ahmad, A. (2019). Design and Development of Real-Time E-Voting System with High-Security Features. *International Journal of Education and Management Engineering*, *9*(3), 37–50. https://doi.org/10.5815/ijeme.2019.03.04

Okpara R., Otugeme E. and Osuagwu O. (2018). Development of a Mobile Android Voting App for Tertiary Institutions in Nigeria. *Voting Android App.*

https://www.researchgate.net/publication/344256892\_development\_of\_a\_mobile\_android\_voting\_app\_for\_tertiary\_institutions\_in\_nigeria

Sherine, A., Peter, G., Stonier, A. A., Leh Ping, D. W., Praghash, K., & Ganji, V. (2022). Development of an Efficient and Secured E-Voting Mobile Application Using Android. *Mobile Information Systems*, *2022*. https://doi.org/10.1155/2022/8705841

S.,A.,& AnilKumar, K. S. (2021). Secure mobile internet voting system using biometric authentication and wavelet based AES. Journal of Information Security and Applications, 61, 102908. doi:10.1016/j.jisa.2021.102908

Selvarani, X. I., Shruthi, M., Geethanjali, R., Syamala, R., & Pavithra, S. (2017). Secure voting

System through SMS and using smart phone application.

https://doi.org/10.1109/ICAMMAET.2017.8186724

Visalakshi, P., Pawar, R. S., & Suriya Kumar, S. (2020). QR code voting in election. *International*

*Journal of Advanced Science and Technology*, *29*(6), 2591–2596.

<https://www.mendeley.com/catalogue/8c54cb6d-c385-325f-89d8-f7c0cc34e429/>

**APPENDIX**

**Homepage.dart**

import 'package:fancy\_bottom\_navigation\_2/fancy\_bottom\_navigation.dart';

import 'package:flutter/material.dart';

import 'package:voteey/utils/constant.dart';

import 'package:voteey/views/home/student/profile.dart';

import 'package:voteey/views/home/student/resultCategory.dart';

import 'package:voteey/views/home/student/voteCategory.dart';

class HomePage extends StatefulWidget {

  const HomePage({super.key});

  @override

  State<HomePage> createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

  final scaffoldKey = GlobalKey<ScaffoldState>();

  GlobalKey bottomNavigationKey = GlobalKey();

  int currentPage = 0;

  // DatabaseService databaseService = Get.put(DatabaseService());

  @override

  Widget build(BuildContext context) {

    final size = MediaQuery.of(context).size;

    return SafeArea(

      child: Scaffold(

        key: scaffoldKey,

        backgroundColor: Constants.basicColor,

        body: IndexedStack(

          index: currentPage,

          children: [

            VoteCategory(size: size),

            const ResultCategory(),

            const ProfilePage(),

          ],

        ),

        bottomNavigationBar: FancyBottomNavigation(

          circleColor: Constants.primaryColor,

          inactiveIconColor: Constants.primaryColor,

          textColor: Constants.primaryColor,

          tabs: [

            TabData(

              iconData: Icons.how\_to\_vote\_rounded,

              title: "Vote",

            ),

            TabData(

              iconData: Icons.dashboard,

              title: "Result",

            ),

            TabData(iconData: Icons.person, title: "Profile")

          ],

          initialSelection: 0,

          key: bottomNavigationKey,

          onTabChangedListener: (position) {

            setState(() {

              currentPage = position;

            });

          },

        ),

      ),

    );

  }

}

**API**

import 'dart:io';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_storage/firebase\_storage.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:flutter/services.dart';

import 'package:voteey/components/delegatedSnackBar.dart';

import 'package:voteey/models/all\_candidate\_data.dart';

import 'package:voteey/models/candidate\_details.dart';

import 'package:voteey/models/position\_data.dart';

import 'package:voteey/models/result\_stats.dart';

import 'package:voteey/models/user\_data.dart';

import 'package:voteey/models/votingCategory.dart';

class DatabaseService extends GetxController {

  String? uid;

  DatabaseService({this.uid});

  UserData? userData;

  // collection reference

  var usersCollection = FirebaseFirestore.instance.collection("Users");

  var positionCollection = FirebaseFirestore.instance.collection("Positions");

  var votesCollection = FirebaseFirestore.instance.collection("Votes");

  var statusCollection = FirebaseFirestore.instance.collection("Status");

  var resultCollection = FirebaseFirestore.instance.collection("ShowResult");

  var candidatesCollection =

      FirebaseFirestore.instance.collection("Candidates");

  var filesCollection = FirebaseStorage.instance.ref();

  //Create user

  Future createStudentData(String name, String regNo, String type) async {

    await setImage(uid, 'Users');

    return await usersCollection.doc(uid).set(

      {

        'name': name,

        'regNo': regNo,

        'type': type,

        'created': FieldValue.serverTimestamp(),

        'last\_updated': FieldValue.serverTimestamp(),

      },

    );

  }

  //Determine userType

  Future<UserData?> getUser(String uid) async {

    // Query database to get user type

    final snapshot = await usersCollection.doc(uid).get();

    // Return user type as string

    if (snapshot.exists) {

      userData = UserData.fromJson(snapshot);

      return UserData.fromJson(snapshot);

    }

    return null;

  }

  Stream<UserData?> getUserProfile(String uid) {

    return usersCollection.doc(uid).snapshots().map((snapshot) {

      if (snapshot.exists) {

        return UserData.fromJson(snapshot);

      }

      return null;

    });

  }

  Future<bool> updateProfileTime(String uid) async {

    usersCollection.doc(uid).update({

      'last\_updated': FieldValue.serverTimestamp(),

    });

    return true;

  }

  Future<bool> updateImage(File? image, String uid, String path) async {

    filesCollection.child("$path/$uid").putFile(image!);

    return true;

  }

  Future<bool> setImage(String? uid, String path) async {

    final ByteData byteData = await rootBundle.load("assets/user.png");

    final Uint8List imageData = byteData.buffer.asUint8List();

    filesCollection.child("$path/$uid").putData(imageData);

    return true;

  }

  Future<String?> getImage(String uid, String path) async {

    try {

      final url = await filesCollection.child("$path/$uid").getDownloadURL();

      return url;

    } catch (e) {

      return null;

    }

  }

  Stream<String?> getCurrentUserImage(String uid, String path) {

    try {

      return filesCollection.child("$path/$uid").getDownloadURL().asStream();

    } catch (e) {

      return Stream.value(null);

    }

  }

  Stream<List<UserData>> getAccounts(String type) {

    return usersCollection

        .where('type', isEqualTo: type)

        .orderBy('created', descending: true)

        .snapshots()

        .map(

          (snapshot) =>

              snapshot.docs.map((doc) => UserData.fromJson(doc)).toList(),

        );

  }

  Future<List<Position>> getPositions() async {

    List<Position> positions = [];

    QuerySnapshot<Map<String, dynamic>> snapshot =

        await positionCollection.get();

    for (QueryDocumentSnapshot<Map<String, dynamic>> documentSnapshot

        in snapshot.docs) {

      Position position = Position.fromJson(documentSnapshot);

      positions.add(position);

    }

    return positions;

  }

  Future<List<AllCandidates>> getCand() async {

    List<AllCandidates> candidates = [];

    QuerySnapshot<Map<String, dynamic>> snapshot =

        await candidatesCollection.get();

    for (QueryDocumentSnapshot<Map<String, dynamic>> documentSnapshot

        in snapshot.docs) {

      AllCandidates candidate = AllCandidates.fromJson(documentSnapshot);

      candidates.add(candidate);

    }

    return candidates;

  }

  //Check if the student account exists

  Future<bool> verifyStudent(String regNo) async {

    QuerySnapshot<Map<String, dynamic>> snaps =

        await usersCollection.where('regNo', isEqualTo: regNo).get();

    if (snaps.docs.isNotEmpty) {

      return true;

    }

    return false;

  }

  //Check if the student account exists

  Future<bool> hasApplied(String regNo) async {

    final snapshot =

        await usersCollection.where('regNo', isEqualTo: regNo).get();

    if (snapshot.docs.isNotEmpty) {

      String studentId = snapshot.docs[0].id;

      final snaps = await candidatesCollection

          .where('can\_id', isEqualTo: usersCollection.doc(studentId))

          .get();

      if (snaps.docs.isNotEmpty) return true;

    }

    return false;

  }

  Future<bool> applyCandidate(String regNo, String posID) async {

    final snapshot =

        await usersCollection.where('regNo', isEqualTo: regNo).get();

    if (snapshot.docs.isNotEmpty) {

      String studentId = snapshot.docs[0].id;

      candidatesCollection.doc().set({

        'can\_id': usersCollection.doc(studentId),

        'pos\_id': positionCollection.doc(posID),

        'votes': 0,

      });

      return true;

    }

    return false;

  }

  //get the details of a particular candidate

  Future<CandidateDetail?> getCandidate(

      DocumentReference canID, DocumentReference posID) async {

    // Query database to get user

    final userSnapshot = await canID.get();

    // Query database to get position

    final positionSnapshot = await posID.get();

    if (userSnapshot.exists && positionSnapshot.exists) {

      // Query database to get image

      String? imageSnapshot = await getImage(userSnapshot.id, 'Users');

      final candidateDetail = CandidateDetail(

        id: userSnapshot.id,

        name: userSnapshot['name'],

        regNo: userSnapshot['regNo'],

        image: imageSnapshot!,

        position: positionSnapshot['title'],

      );

      return candidateDetail;

    }

    return null;

  }

  Stream<List<CandidateDetail>> getCandidates() {

    //get all candidate by position

    return candidatesCollection

        .orderBy('pos\_id', descending: true)

        .snapshots()

        .asyncMap<List<CandidateDetail?>>((snapshot) async {

      // create an empty candidate details list

      final candidateDetails = <CandidateDetail?>[];

      //loop through the gotten candidates snapshot

      for (final doc in snapshot.docs) {

        //call a helper function to get a particular candidate

        final candidate = await getCandidate(doc['can\_id'], doc['pos\_id']);

        //add the result to the candidateDetails

        candidateDetails.add(candidate);

      }

      return candidateDetails;

    }).map((list) => list.whereType<CandidateDetail>().toList());

  }

  Stream<List<VotingCategory>> groupCategories() {

    // get all positions

    return positionCollection.snapshots().asyncMap(

      (snapshot) async {

        // create an empty categories list

        List<VotingCategory> categories = [];

        // loop through the positions

        for (var positionDoc in snapshot.docs) {

          //assign the position id to a variable

          String positionId = positionDoc.id;

          //query all the candidates for that position

          QuerySnapshot candidatesSnapshot = await candidatesCollection

              .where('pos\_id', isEqualTo: positionCollection.doc(positionId))

              .get();

          //verify that the candidateSnapshot is not empty

          if (candidatesSnapshot.docs.isNotEmpty) {

            //assign the candidateDos to a variable

            List<DocumentSnapshot> candidateDocs = candidatesSnapshot.docs;

            //add the category to the created list above

            VotingCategory category = VotingCategory(

              id: positionId,

              posTitle: positionDoc['title'],

              candidateNo: candidateDocs.length,

            );

            categories.add(category);

          }

        }

        return categories;

      },

    );

  }

  Future<VotingCategory?> positionCategories(String posID) async {

    // get all the positions

    DocumentSnapshot<Map<String, dynamic>> positionSnapshot =

        await positionCollection.doc(posID).get();

    // check if the position exists

    if (positionSnapshot.exists) {

      // get all the candidates with the existing position

      QuerySnapshot<Map<String, dynamic>> candidateSnapshot =

          await candidatesCollection

              .where('pos\_id', isEqualTo: positionCollection.doc(posID))

              .get();

      //if the gotten snapshot is not empty

      if (candidateSnapshot.docs.isNotEmpty) {

        // assign the docs to a variable

            );

            maxVoteCandidates = [detail];

          }

        }

        winningCandidates.addAll(maxVoteCandidates);

      }

      return winningCandidates;

    });

  }

  Stream<List<ResultDetails>> resultStatistics(String posID) {

    return candidatesCollection

        .where('pos\_id', isEqualTo: positionCollection.doc(posID))

        .orderBy('votes', descending: true)

        .snapshots()

        .asyncMap((snapshot) async {

      final List<ResultDetails> resultDetails = [];

      for (final doc in snapshot.docs) {

        final data = doc.data();

        final userSnapshot = await data['can\_id'].get();

        final imageSnapshot = await getImage(userSnapshot.id, 'Users');

        resultDetails.add(ResultDetails(

          id: doc.id,

          name: userSnapshot['name'],

          image: imageSnapshot!,

          votes: data['votes'],

        ));

      }

        await documentRef.update({'start\_voting': status});

        return true;

      }

      return false;

    } catch (e) {

      return false;

    }

  }

  //Check for the voting status

  Stream<bool> votingStatus() {

    final Stream<bool> stream = statusCollection.snapshots().map((snapshot) {

      if (snapshot.docs.isNotEmpty) {

        final bool status = snapshot.docs.first.data()['start\_voting'] ?? false;

        return status;

      } else {

        return false;

      }

    });

    return stream;

  }

}