Sudan University of Science and Technology

College of post graduates studies

Self-protection of Mobile Application Data using Reflection (E-vote Application as a Case Study)

الحماية الذاتية لبيانات تطبيقات الهاتف المحمول باستخدام الانعكاس

(تطبيق التصويت الإلكتروني كدراسة حالة)

A Thesis Submitted in Partial Fulfillment of the Requirements of M.Sc. in Information Technology

Prepared by:

Saria Mohi Aldein Mohammed

Supervisor:

Dr. Nadir Kamal salih

Feb.2020

الآية

قال تعالى :

(وَيَسْأَلُونَكَ عَنِ الرُّوحِ ۖ قُلِ الرُّوحُ مِنْ أَمْرِ رَبِّي وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا)

سورة الإسراء الآيه(85)

DEDICATION

I dedicate my work for my beloved parents (Hind Osman Mohammed Ahmad and MohiAldein Mohammed), my dear brothers and sister

Special dedication to my grandfather (Osman Mohammed Ahmed) who always give me encouragement in my life, my study and to finish my Project.

To my Supervisor

Dr. Nadir Kamal Salih

To all SUST’s lecturers

To all my classmate

To Dr. Mohammed Abdurrahman Mohammed Salih, To Eng. Mohammed Elrasheid in DAMS.

And all my friends out here

Thank You for everything

THANK YOU SO MUCH

ACKNOWLEDGEMENT

Thanks first to Allah Almighty for his countless grace, which enabled me to carry out this study and is interested in strength and patience.
My special thanks, my great gratitude and my deep gratitude to my supervisor, Dr. Nader Kamal Salih, who made these study possible thanks to his valuable guidance, effort and patience.
Many thanks to the friend and colleague Halima Mustafa Al-Basher for her help in providing me with instructions and notes that motivated me a lot in completing this work.
Finally, I would like to express my big thanks to all the people who helped me with this work.

ABSTRACT

Android considered is the most common operating system for mobile phones. As it an open source operating system which led to produce of many applications which resulted to occurrence of a lot of problems facing applications, such as security problems, which is the most affecting problems on application’s data.

This research provides a mechanism to protect application’s data, which is self-protection using reflection that means the application is able to examine and protect itself without external interferences. This mechanism has been applied to the E-Vote applications a model for an application that contents sensitive data. This application implemented on android studio, using java programming language coding, MySQL is used for database and PHP to create server pages.

As a result of implemented this mechanism it is succeeded to protected application's data from violation and theft, and protect the server from overloaded of inside application, finally solved some types of bugs such as input data, validation form, password and buttons.

المستخلص

يعتبر الأندرويد نظام التشغيل الأكثر شيوعاً للهواتف المحمولة, نظراً لأنه نظام تشغيل مفتوح المصدر ادى إلي انتاج الكثير من التطبيقات مما أدى الي حدوث الكثير من المشاكل التي تواجه التطبيقات, مثل مشاكل الأمان والتي تعد من أكثر المشاكل التي تؤثر علي بيانات التطبيقات.

يوفرهذا البحث آلية لحماية بيانات التطبيق والتي هي حماية ذاتية باستخدام الانعكاس مما يعني أن التطبيق قادرعلى فحص وحماية نفسه دون تدخل خارجي. تم تطبيق هذه الآلية على تطبيق التصويت الإلكتروني كنموذج لتطبيق يحتوي على بيانات حساسة . تم عمل هذا التطبيق علي أندرويد استديو بإستخدام لغة جافا في البرمجة واستخدمت MySQL لإنشاء قاعدة البيانات وPHP لعمل صفحات المخدم.

نتيجة لتطبيق هذه الآلية، نجحت في حماية بيانات التطبيق من الانتهاك والسرقة، وحماية الخادم المثقل من داخل التطبيق، وأخيراً حل بعض مشاكل أنواع الأخطاء مثل الإدخال، كلمة السر ومشاكل الأزرار.

Table of Contents

List of Table

List of Figure

CHAPTER I

INTRODUCTION

1.1. Introduction

This chapter briefly describes the self-protection using reflection design to achieve mobile applications self-protection. It consists of five sections: the first section describes the background of the project. The second section describes the problem statement and motivation of the project. The third section describes the objectives for the project. The fourth section describes the scope for the project. Finally, in section five thesis organizations is described.

Compared to traditional mobile phones, which mainly provide mobile telephony functions, Smart phones are general-purpose handheld computing and communications devices that support multimedia communications and applications for entertainment and work (He et al., 2015). One of the most attractive features of smart phones is the availability of a large number of apps for users to download and install (Ametller et al., 2004b). Due to this quantum jump leap in functionality, threat of upgrading traditional mobile phones to smart phones is tremendous (Davi et al., 2010).That means hackers can easily distribute malware to smart-phones, launching various attacks. This issue should be addressed by both preventive approaches and effective detection techniques. (Lane et al., 2010).Android Market has fewer retractions when it comes to registering as a developer. This is Android's strategy to encourage future app developers, but this also makes it is easier for cybercriminals to register as developers to upload their malicious apps or their Trojanized counterparts.

Android is a modern and popular software platform for smart- phones(Lane et al., 2010). Among its predominant features is an advanced security model which is based on application-oriented mandatory access control and sandboxing (Davi et al., 2010). This allows developers and users to restrict the execution of an application to the privileges it has (mandatorily) assigned at installation time. The exploitation of vulnerabilities in program code is hence believed to be conned within the privilege boundaries of an application's sandbox (Jin et al., 2014). So the presence of bugs and security holes is statistically numerous and variety Necessitates necessity combine protection mechanisms inside the mobile system to become applications self-protection.

Android is an open, security focused, mobile platform that is programmed with Java(Anderson et al., 2010).

Mobile application's is very important for the production companies so they take many Techniques to protect it. The one of ways to protect the mobile applications is self-protection using reflection.

The term Self-protection is one of autonomic system that aims at enabling computing infrastructures to perform administration tasks without human intervention that means ability of an autonomic system to secure itself against attacks to detect illegal activities and to trigger counter measures in order to stop them(Burns, 2008).

Reflection is the ability of a program to examine and modify the structure and behavior of an object at runtime.

1.2. Problem Statement

The most mobile applications may contain security vulnerabilities that can be a source of danger to users and their privacy.

The most security problems faced mobile application data problems such as capture data, retransmission data, and change and problem cause bugs like validation form, input data, and password.

Research question 1:

Which method can be developed for self-protection mobile application data? Start to apply the concept of the autonomic system that let the system work without intervention of the user. It has given by implemented and designed method that using the reflection concept.

Research question 2:

How can use reflection design to develop the application to protect them self? In the reflection, design the methods to do the prevention process and determine the data which need to save it by putting constraint in all steps to sure from it.

In this research tacked the E-vote application as a model to implements self-protection for application data using reflection concept

1.3. Objectives

Programming applications for smart phones has become a common area and new in the world of programming and has become a platform for many developers to spread their creativity and enable them to work applications that help in a lot of business, whether in our daily lives or practical, and it became very necessary to protect applications from attackers who use illegal methods to sabotage them, Our main goal is to create an application that protects itself without user interaction or intervention and achieves:

1. To combine protection mechanisms within the mobile system to become self-protection applications.
2. To protect data from detection and theft.
3. To protect applications from attackers who exploit vulnerabilities in applications.
4. To enhanced robot tools that are used to give robot permission.
5. To improve tools to increase self-security in mobile applications.

1.4. Scope

This autonomic system use self-protection to solve the problem by using reflection which is algorithm can help the application to examine itself.

1.5. Thesis organization

This is about reflection for self-protection mobile application data and consists of five chapters as follows:

Chapter One: Introduction

In this chapter, provide background information about the application which normally includes problem statement, objectives and scope.

Chapter Two: Literature Review

Consists of case study, literature reviews and previous research evaluation in mobile data application protection.

Chapter Three: Methodology

Guideline for solving a problem, with specific components of CBR such as phases, tasks, methods, techniques and tools using in the project.

Chapter Four: Design, Implementation, Analysis and Results

The implementation of the research project is presented; interfaces design, present the result of the project and discuss the result of the project.

Chapter Five: Conclusion and Recommendation

The conclusion and suggests some recommendation in order improve the project in future. This chapter summarizes the overall project.

References

CHAPTER II

Literature Review

2.1. Introduction

This chapter describes the review on android mobile application, in this chapter, two sections are comprised: The first section briefly explain the background of android mobile application security, E-vote as a model and the second section describes the review on previous works.

2.2. Mobile Application Security

Modernizing traditional mobile phones to Smartphone’s is a great event, and accordingly the rapid growth of the global smart phone market in the coming years has increased rapidly through the increased use of smart phone business.

One of the distinguishing features of Smartphone’s is that they allow users to install and run third-party application software, which is usually referred to as applications. It greatly broadens the limits of Smartphone’s functions, thus enriching the user experience(He et al., 2015).

These apps are officially distributed across online stores that are offered as app markets - Apple App Store for IOS and Google Play Store from Android. These markets provide a convenient place for app developers to install their apps and for users to explore and download new apps. This has led to a high rate of application development in recent years. Like other electronic systems, Smartphone’s are also vulnerable to malware, which is malware designed to run on the affected system without the owners' knowledge. While users are keen to download apps from the app market, this provides hackers with a convenient way to infect Smartphone’s with malware. For example, they will repackage common games with malware and distribute them in the app market(Jin et al., 2014).

A recent survey indicated that 267,259 applications were found infected with software, of which 254,158 were found on Android. He also suggested that the number of malware applications have increased by 614 percent since 2012. There is also a variety of other ways of malware to infect targets (La Polla et al., 2012). Some malicious software, such as macros, is hidden from files. Some of them are installed by some known vulnerabilities in the network device or the mobile platform. Some are installed in victims' Smartphone’s when they click on an MMS message, open the email attachment.

However, malware can cause serious information security and data privacy problems, with severe consequences for users and even organizations(Sui and Guo, 2012).

2.2.1. Android Application Development and Security

Android is a mobile operating system based on Java, and it is a freely available system powered by Google.

Android mobile OS provides a flexible environment for Android Mobile ApplicationDevelopments the developers can not only make use of Android Java Libraries but it is also possible to use normal Java IDEs. The software developers at Mobile Development in India have expertise in developing applications based on Android Java Libraries and other important tools (Holla and Katti, 2012).

While mobile application programming languages are based on traditional programming languages, there are differences in the methods that a mobile application is development, tested, deployed, and secured(Liu et al., 2014).

2.2.2. Mobile Application Design Bugs

A bug is a difference between the real and expected results, which can find on a daily basis, can be divided into three groups:

1. App-specific bugs. They are related to business logic of the app. They might be pretty hard to detect so deep app knowledge may really help you. It is also very important to write down test cases for such type of bugs.
2. Platform-specific bugs. Each mobile platform (Android, iOS) has its own bugs connected to the way the operating system works.
3. Specific bugs related to the basic elements of the app architecture(Argersinger, 1985).

The most common types of mobile testing bugs which may occur are:

a. **Input Validation: Bugs related to valid/invalid inputs:**
   Displaying the appropriate type of keyboard for the user is easy to use and reduces errors .For example, for a phone number text field, we'd like to show a user a numeric keyboard and for an email text field a keyboard having the "@" character.
b. **Login/Logout:** Bugs related to Login/Logout process.
c. **Password Validation**: Bugs related to weak password validation and should also pay attention to which is a password input type. It's just not safe to use a text field for a password. Reading a password over

someone’s shoulder is definitely much easier than reading their keystrokes

d. **Cookies:** Bugs related to insecure cookies validation.

e. **Crash after tapping on button**: This one is like a “time-bomb” hidden in your app. usually, this applies to buttons that are “hidden” deep inside the application (Ex. inside settings) which are relatively easy to overlook. Clicking on such a button makes application crash.

Unfortunately, 100% bug-free apps don’t exist. All can did is try to minimize the amount of mobile app bugs(Pathak et al., 2011).

2.3. E-voting application Model

In this research, the concept of self-protection using reflection has been applied to the application of electronic voting as it is an application that contains sensitive data vulnerable to attack by hackers(Schaupp and Carter, 2005).

2.3.1. Elections:

Elections are the formal process of selecting a person to take a formal position by voting.

2.3.1.1. Elements of the electoral process

1. The electoral district: A group of individuals who are entitled to elect a representative in the legislature.
2. The candidate: The person who nominates himself to represent individuals (voters) in the electoral district.
3. The individual / citizen (voter): they are the ones who are entitled to vote and before conducting any electoral process. Determine the conditions that must be met by people in order for them to vote, such as the voter being a citizen of the same nationality as the country, adult and sometimes other conditions are established such as the voter He was never convicted of a crime.
4. The electoral campaign: It is a process that the candidate undertakes to introduce himself and his goals.
5. Election (voting): an organized process in which individuals choose their representatives.

2.3.2. Vote

Voting for elections is a legitimate right of every citizen, as voting is the method used by a group of people to make a specific decision or to express their opinion. In the electoral process, voting is when individuals choose one of the candidates to represent them in the elected bodies.

Voting is an effective step in an election, as it contains sensitive data that must be secured and protected to ensure its transparency and credibility.

2.3.2.1. Vote methods

1. Paper: It is the most common voting method in which the voter refers to the candidate he prefers. Ballot messages can be used so that there is a room with boxes for each party or candidate and the messages are placed in the specified ballot box.
2. Automatic voting: This is done by voting machines, where the voter manually enters the number of the candidate who wishes to vote, and when his image appears on the screen, the voter confirms his choice.
3. Via the Internet: Voting is done through websites.
4. Postal voting: The ballot paper is sent by post.
5. Open Voting: It is done in public and often by show of hands.

2.3.3. Election fraud

It is also called electoral fraud, and it is an illegal interference in the election process to change votes in favor of a candidate or steal them from a candidate, which affects the results after the counting process(Lehoucq, 2003).

2.3.4. E-voting

Information and communication technology is the main driving force that influences the growth of the global economy, civil societies and governments in general and the lives of people in particular. And that its entry into the political field is considered the most influencing the life of society. As it entered the electoral process in all its stages to ensure efficiency, accuracy and speed in its achievement.
Free and fair elections are the main pillar of building a democratic system in any country(Lankina and Skovoroda, 2017).

Electronic voting is a tool that increases the efficiency of the electoral process and increases confidence in its management if it is applied properly so that it can increase the security of polling and the speed of announcing the results.

2.3.4.1. Benefits of electronic voting:

It has more advantages than traditional methods of voting:

1. . The lowest cost.
2. . Faster scheduling of results.
3. Improved access to results with greater accuracy.
4. Less dangerous than human and mechanical errors.
5. Increasing citizen participation in the political process.

2.3.4.2. Challenges facing e-voting

1. . The ability to vote on behalf or by proxy, such as a family vote, where the voting process is subject to coercion and pressure on the part of the head of the household by his ownership of the electronic cards.
2. The possibility of the electronic system being exposed to piracy from abroad, which leads to data leakage and tampering with it, which corrupts the integrity of the electoral process.
3. . Errors occurred during program execution.(Lankina and Skovoroda, 2017)

2.4. Previous works

2.4.1. Android Permissions

Adrienne Porter Felt et al.(Felt et al., 2011) They built Stowaway, a tool that detects excessive privilege in bundled Android applications and used automated self-testing tools on the Android API in order to build the permission map needed to detect permissions and was tested on more than one application. So they checked for excessive permissions.

Long Lu et al. (Lu et al., 2012) they study a general category of vulnerabilities found in Android apps, namely the component hijacking vulnerabilities. They used CHEX that is a static analysis method to automatically vet Android apps and detects possible hijack-enabling flows by conducting low-overhead reach abilitytest on customized system dependence graphs. To tackle analysis challenges imposed by Android’s special programming paradigm

D. Bucerzan, C. et al(Bucerzan et al., 2013). The Android SmartSteg app is able to quickly hide and encrypt files using digital images of MB dimensions as a cover. data masking information is integrated with random function and symmetric key encryption to transfer digital information, securely between Android smart phones.

Lucas Davy et al.(Davi et al., 2010)it turned out that an escalation attack is possible. They show that a real application exploited at run time or a malicious application can escalate the granted permissions. Our results immediately indicate that the Android security model cannot handle the pass-through attack and that the Android protection box model fails as a last resort against malware and advanced uptime.

2.4.2. Protection data:

Golam, et al(Babil et al., 2013)they used the popular TaintDroid system is used to track sensitive information on Android mobile phones. The data is tracked during its publication through variables, exchanged messages and files, by marking them with contaminated tags, such as device identifiers or user contact details, then warnings are issued when this information is misused, they provide a set of attacks to track robot-based damage.

The technologies presented are applied in the Android app, ScrubDroid. They have successfully tested their app with TaintDroid apps for Android versions 2.3 to 4.1.1, although dye tracking may be a valuable tool for developers, but it will not effectively protect sensitive data from the black box of the zealous attacker that applies any of the anti-tracking methods the pollutants provided.

Olivier Mehaniet al.(Sarwar et al., 2013)they using taint analysis for tracking the privacy information on Android-based mobile conclude that, although taint tracking may be a valuable tool for software developers, it will not effectively protect sensitive data from the black-box code of a Motivated attacker applying any of the presented anti-taint tracking methods.

Hao Penget al.(Peng et al., 2012) they used the idea of risk classification and improved communication process for android applications.

Thomas Bl¨asing et al.(Bläsing et al., 2010)they suggest an Android Sandbox (AASandbox) app that can perform static and dynamic analysis on Android programs to automatically detect suspicious apps. Static analysis scans the program for harmful patterns without installing it.

Dynamic analysis performs the application in a completely isolated environment, that is, the protection box, which intervenes and records a low level Interactions with the system for further analysis. Both the security box and detection algorithms can be deployed in the cloud, providing quick and distributed detection of suspicious software in a mobile app store similar to Google's Android Market. In addition, AASandbox may be used to improve the efficiency of classic antivirus applications available for Android.

FumihiroKanei et al.(Kanei et al., 2017)They proposed a method of strong self-protection against evasion attacks. The suggested method automatically creates the ability to rediscover discovery in applications. The detection code randomly divides into several blocks, which are inserted directly into the zip of applications. The results of the evaluation indicate that the degree of durability, which is calculated based on false positives from the attackers' point of view, is 3.5 times higher than in the current method.

The suggested method can also easily protect apps because they only require their secondary code.

j Ametller et al(Ametller et al., 2004a).they present a new solution for the implementation of flexible protection mechanisms in the context of mobile agent systems, where security problems are currently a major issue. In their scheme, agents protect their code and data by carrying their own protection mechanisms. This approach improves traditional solutions, where protection was managed by the platform.

The implementation is far from trivial. They have implemented this scheme in the JADE framework, using Java. Any application using mobile agents can incorporate these mechanisms to implement agent protection with a minimum impact in the existing code base.

By using cryptographic inscription and decryption methods the Mobile agent protection can be achieved. Following table (2.1) shows the analysis of previous work

Table 2.1 previous works summary

Study	Techniques	Result	Open issue
[7]	Stowaway	This tools can detects over privilege in compiled Android applications they used automated testing tools on the Android API in order to build the permission map that is necessary for detecting over privilege	Fail due to in sufficient API documentation.
[8]	CHEX Component Hijacking Examiner.	To automatically vet Android apps and detects possible hijack-enabling flows by conducting low-over head reach ability testson customized system dependence graphs	The fact that CHEX only checks data-flows to detect vulnerabilities may cause false negatives. Rare vulnerable components may exist that enable hijacking attacks without explicit data-flows
[9]	SmartSteg	This mechanism able to quickly hide and encrypt files using digital images of MB dimensions as a cover. LSB data masking information is integrated with random function and symmetric key encryption to transfer digital information, securely between Android smart phones.	Now not permit secret communication between computes and smart phones
[10]			

[11]	Taint Droid	It using for tracking the privacy information on Android-based mobile conclude that, although taint tracking may be a valuable tool for software developers	It will not effectively protect sensitive data from the black-box code of a motivated attacker applying any of the presented anti-taint tracking methods.
[12]	Taint Droid	It successfully tested their app with the Taint Droid implementations for Android OS versions 2.3 to 4.1.1, both using the emulator and with real devices. And they evaluate the success rate and time to complete of the presented attacks.	It will not effectively protect sensitive data from the black-box code of a Motivated attacker applying any of the presented anti-taint tracking methods.
[14]	AASandbox	It that can perform static and dynamic analysis on Android programs to automatically detect suspicious apps and may be used to improve the efficiency of classic antivirus applications available for Android.	-

2.5. Summary

The number of mobile applications is constantly increasing because Android is open source and therefore will be a very large attack on them, and therefore programmers should use available technologies that ensure them confidentiality of the application and the confidentiality of its data.

This concept is very useful especially for novice programmers to help them implement the confidentiality of their work so that they are less vulnerable to attack and this technique is simple as it does not work to encrypt data but remain the same without encryption and therefore do not need algorithms to decode and decrypt. The methodology for implementation was discussed in the next chapter.

CHAPTERIII

METHODOLOGY

3.1. Introduction

This chapter discusses the methodology of our system, Self-Protection for Mobile Application data by Using Reflection Design.

3.2. Autonomic System

Autonomic Computing presents a new paradigm where computing systems manage themselves, guided by high-level objectives to enabling computing infrastructures to perform administration tasks without human intervention (Dawson, 2009).

3.3. Self-Protection

The term self-protection is one of the autonomic systems that aim to enable computer infrastructures to perform administrative tasks without human intervention. This means that an independent system can secure itself against attacks to detect illegal activities and launch counter measures to stop them(Malek et al., 2019).

Mobile Application Security Test is a process to review application properties and vulnerability code. It is a combination of static analysis, code review and penetration testing in the reflection model, one of the components can think about and modify itself, in terms of different situations, so the concept of reflection will be used to achieve self-protection

3.4. The Reflection

Compositional adaptation enables software to modify its structure and behavior dynamically in response to changes nits' execution environment(McKinley et al., 2004).

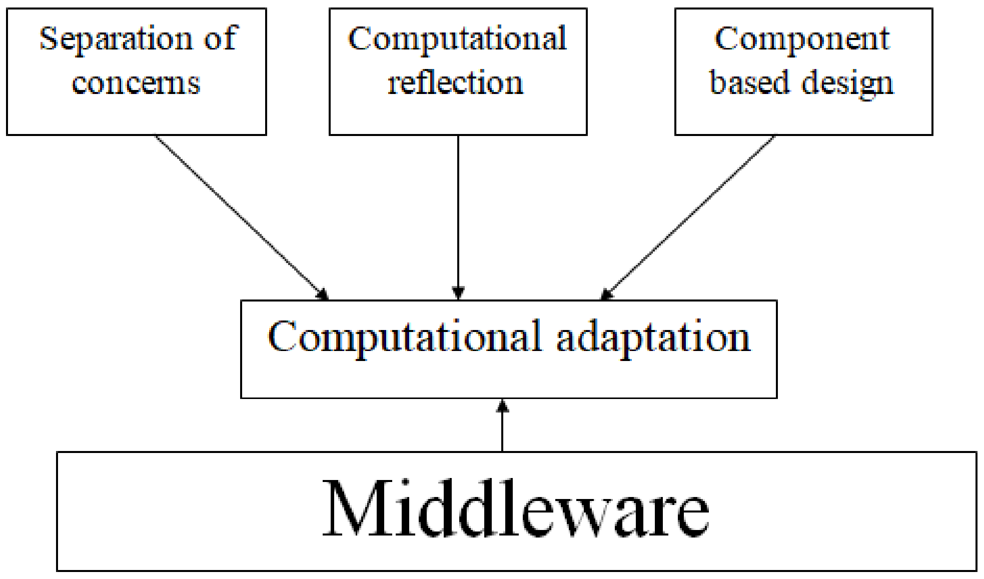


Figure 3.1 Main Technologies Supporting Composition a adaptation

3.4.1. Overview of Reflection

The reflection mechanisms of a programming language provide a running program with the ability to examine itself and its environment to perform self-examination, a program needs an accessible representation of itself; this level of indirection is facilitated through metadata and is fundamental to a reflective system(Pontes et al., 2019).

The two main aspects of self- manipulation are introspection and intercession, which are the abilities of a program to observe and modify (respectively) its own state and behavior. Both aspects require a mechanism for encoding execution state as data. In Java this is realized with so-called Meta objects, which provide access to the representation of Java classes and are available in the java.lang.reflect package(Li et al., 2019).

The java reflection allows the program system to examine or modify the behavior of classes, interfaces, methods, and fields at run time, allowing the program to adapt to dynamically changing runtime environments(Malabarba et al., 2000).

3.4.2. Computational Reflection

Computational reflection refers to a program's ability to reason about, and possibly alter; its own Behavior. Reflection enables a system to reveal selected details of its implementation without compromising portability

3.4.3. Levels of Reflection

a. Introspection.
    i. No alteration, read only to let an application observe its own behavior

b. Behavioral Reflection.
    i. Limited Alteration possible, such as method invocation.
    ii. In general: behavior of operations is changeable.

c. Structural Reflection.
    i. The ability to change data structures used by the runtime system, to let a system or application acts on these observations and modify its own behavior. Such as Class for example.

3.4.4. Reflection Types

a. Structural reflection

Addresses issues related to class hierarchy, object interconnection, and data types. As an example, a met level object can examine a base-level object to determine what methods are available for invocation.

b. Behavioral reflection

Focus on the application's computational semantics. For instance, a distributed application can use behavioral reflection to select and load a communication protocol well suited to current network conditions.

3.4.5. Reflection Process:

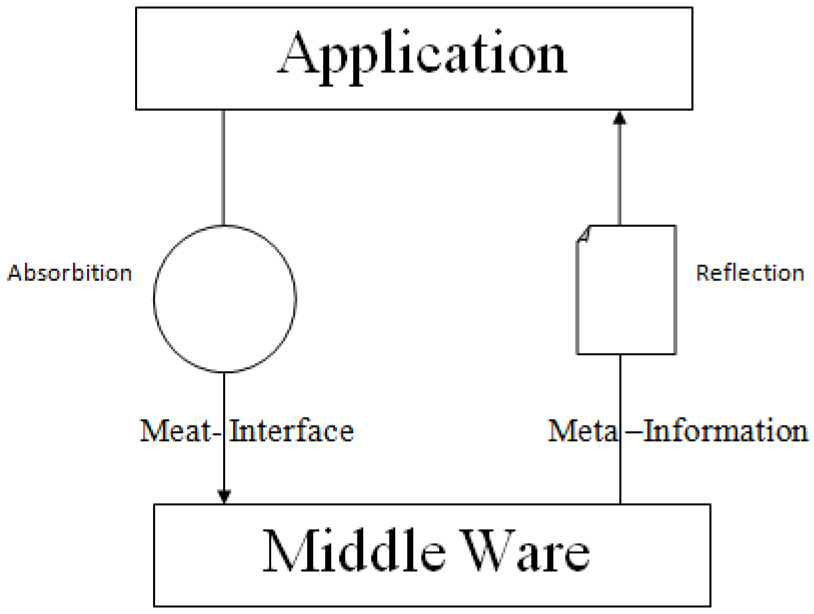


Figure 3.2 Reflection process

Absorption is the process where some aspects of the system are altered or overridden

Reification The key to the approaches to make some aspects of the internal representation of the middleware explicit, and hence accessible from the application, through a process.

The behavior of the middleware with respect to a specific application is described as a set of associations between:

1. the services that the middleware customizes ,
2. the policies that can be applied to deliver the services, and
3. the context configurations that must hold in order for a policy to be applied

3.5. Role andResponsibility ofReflective Process

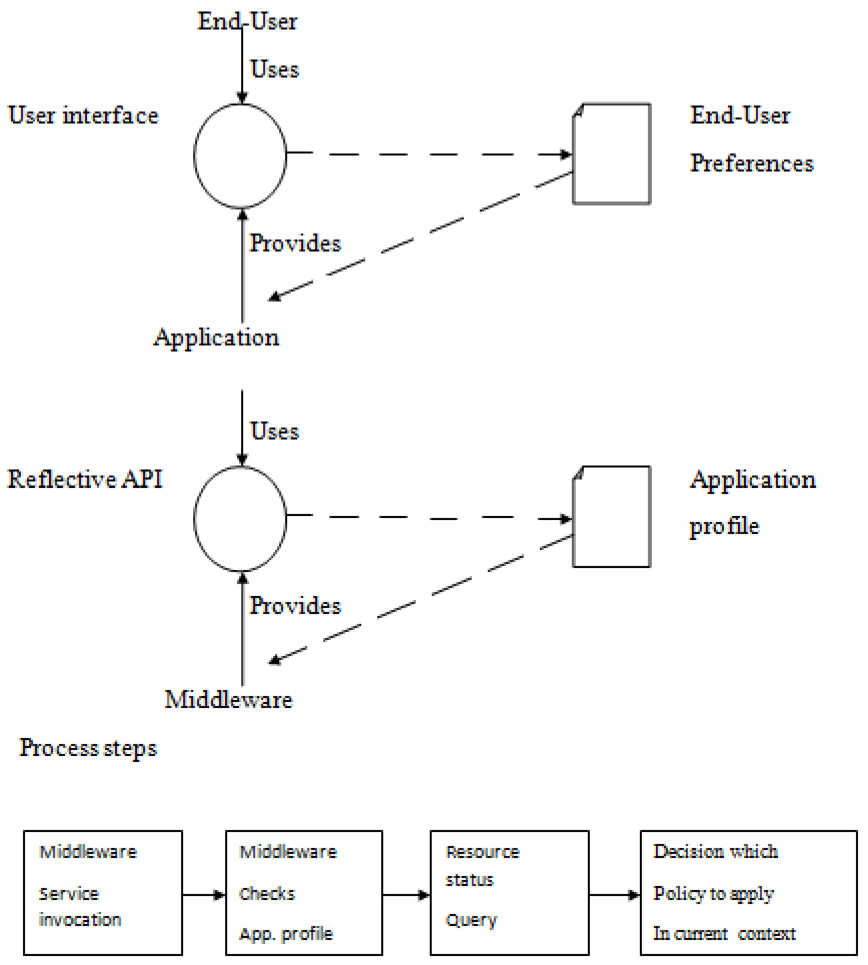


Figure 3.3 Reflection design

3.6. Reflective Designs

Enhance object-oriented design and programming by techniques for runtime system adaptation.However,they are two key notions: Design elements and Design operators.

3.6.1. Design elements:

Use the term design element to denote representations of design decisions in programs.

In fact, the design required elements were amenable to reflection such that design decisions can be observed and modified at runtime.

3.6.2. Design operators:

When compared to basic techniques such as the use of a Meta object protocol, the use of design elements makes runtime system adaptations more disciplined and more manageable. To this end, the abstractions that capture common design elements in a reusable manner were provided. Applications of such abstractions perform system adaptations at a design level hence; which called design operators(Li et al., 2019).

3.7. Summary

The reflection on the protection of application data using some of the functions designed to allow the process of protection and wishes to achieve a good impact in the process of self-protection of application data, the implementation of the application discussed in next chapter.

CHAPTER IV

DESIGN, IMPLEMENTATION OF SELF-PROTECTION E-VOTE APPLICATION AND RESULTS

4.1. Introduction:

This chapter discussed about how the implementation stage has been done in developing the E-voting application, and discussed the results. The development was involved the Graphical User Interface (GUI) design, data base, and the coding development for entire application. The method used in developed the application and database was also discussed here.

E-voting application has been developed using Android Studio. The source code of the application used the Java programming language and the database was created using MySQL (using PHP language to create server pages).

Below figure (4.1) shows the implemented framework of the E-voting application using reflection:

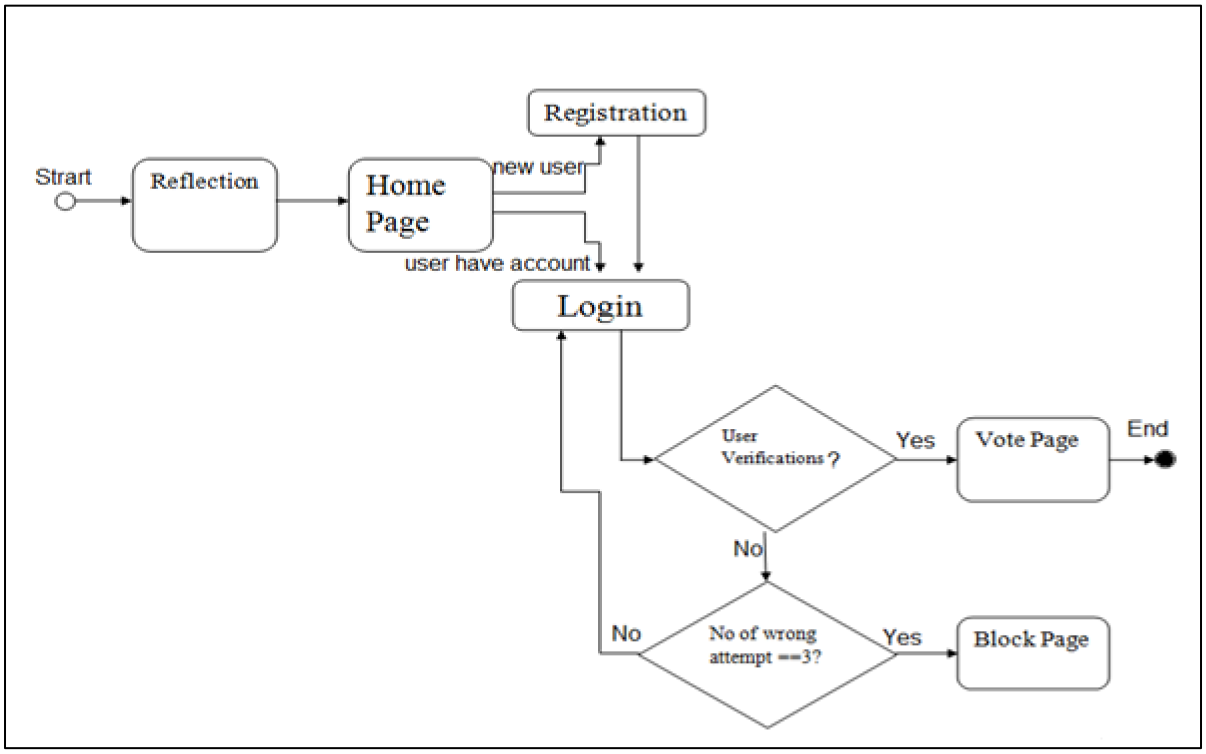


Figure 4.1 E-Vote Application framework using Reflection

4.2. Development Environment:

This application was developed in Android Studio using the Java programming language. Window 8 was used as an operating system with an Intel (R) processor and 4 GB of RAM to develop the application environment. This application relies on the civil registry databases in the registration process where the national number from the user was validated. The table (4.1) as the below shows the environmental needs for the application development.

Table 4.1 Environmental needs for the application development

Type	Tool	Platform
Programming Platform	Android Studio	Windows
Programming Language	Java	Windows
Operating System	-	Windows 8
Hardware	-	Asus Laptop
Processor	Core 3	Intel (Side) Core i3,4030U with 1.9GHz
RAM	-	4GB
Data base	XAMP	MySQL

4.3. Interface design:

The interface allows the user to interact with the application. The application of electronic voting here consists of four interfaces, the first interface is registration, the second interface is the login, the third is the vote, and the last interface appears in the case of the user who was blocked.The figure (4.2) below shows the E-voting model:

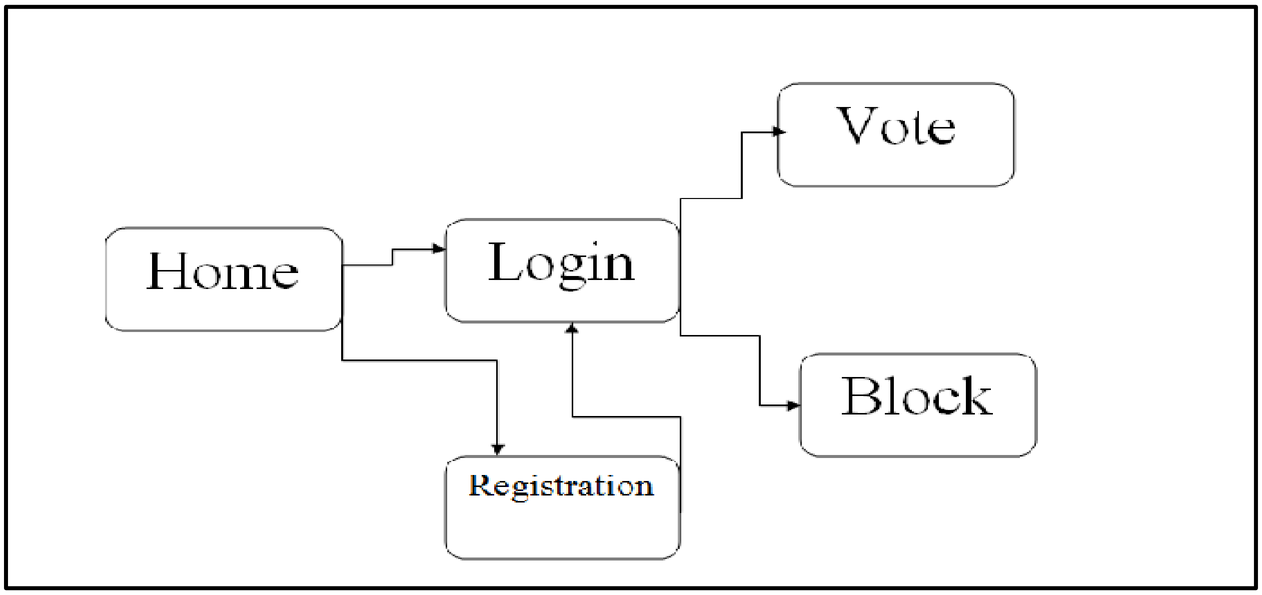


Figure 4.2 E-Vote interfaces model

4.3.1. Registration:

It is the first screen of the application which appears to the user when it is opened, and by which the new users were registered to the application. The screen contains three fields; the first field for the national number, which consists of 11 digits. The entrance number must be already founded in the civil registry database. The second field for the verification code which is a number consists of 4 digits where it is sent to the user, and the last field for the password numbers which contain 8 digits.

The figure (4.3) below illustrates the design of this screen:

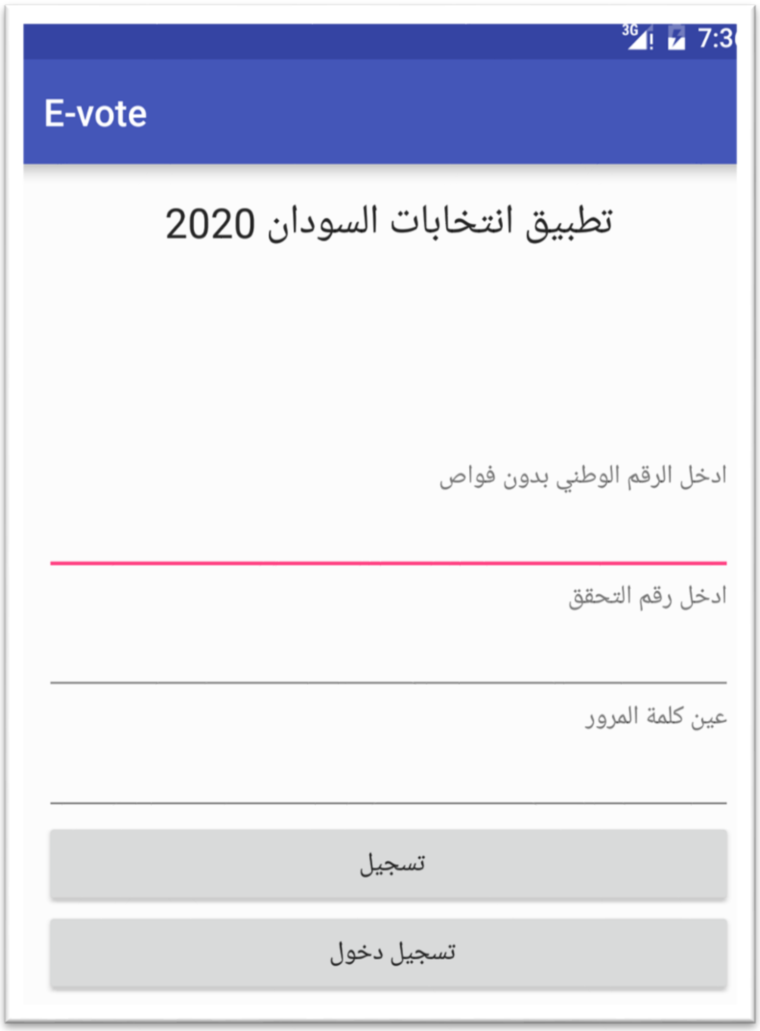
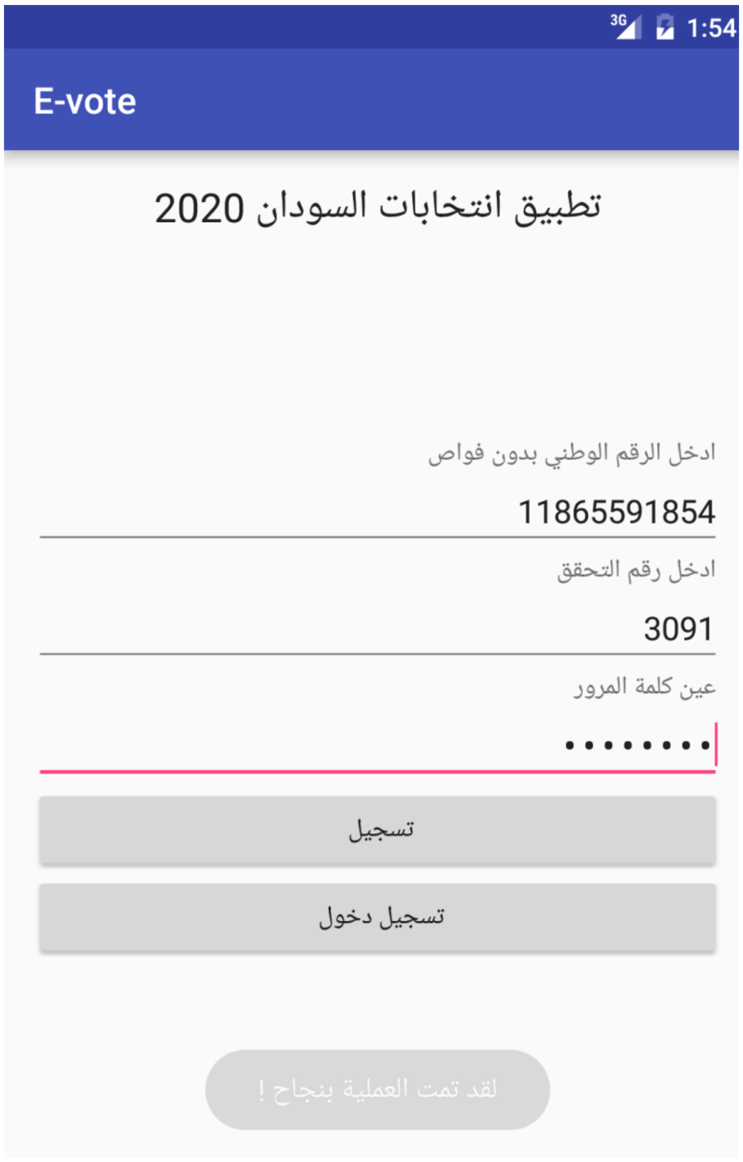


Figure 4.3 Registration Screen

If the users want to register in E-vote application must have verified national number that already found in the civil registry database, and then insert their password and then verification code which was sent automatically to their phone number. The figure (4.4) as below shows that:


3G 1:54

E-vote

تطبيق انتخابات السودان 2020

ادخل الرقم الوطني بدون فواص

11865591854

ادخل رقم التحقق

3091

عين كلمة المرور

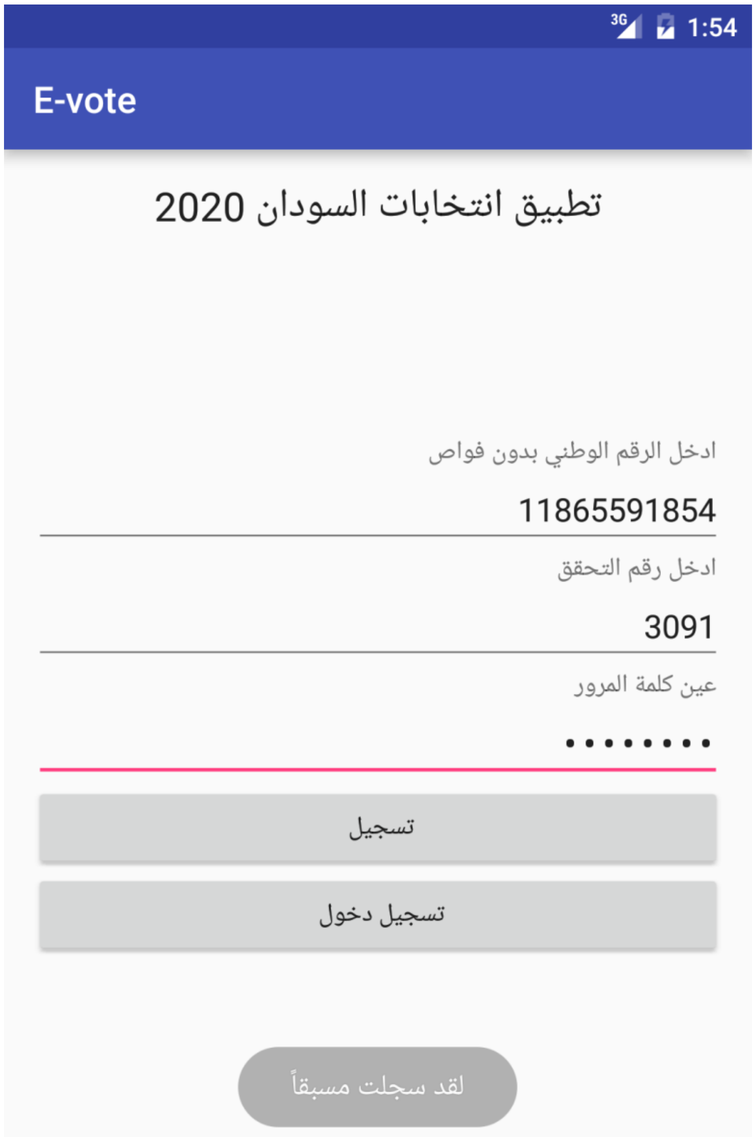
•••••••••

لقد تمت العملية بنجاح !

Figure4.4 RegistrationsSuccessfully

If the users were already registered in this application, they cannot use their national number to try this process, as shown in the following figure where the same national number used to re-sign in again. A message was appeared which indicate that they have already registered.

The figure (4.5) below shows the message:



تطبيق انتخابات السودان 2020

ادخل الرقم الوطني بدون فواص

11865591854

ادخل رقم التحقق

3091

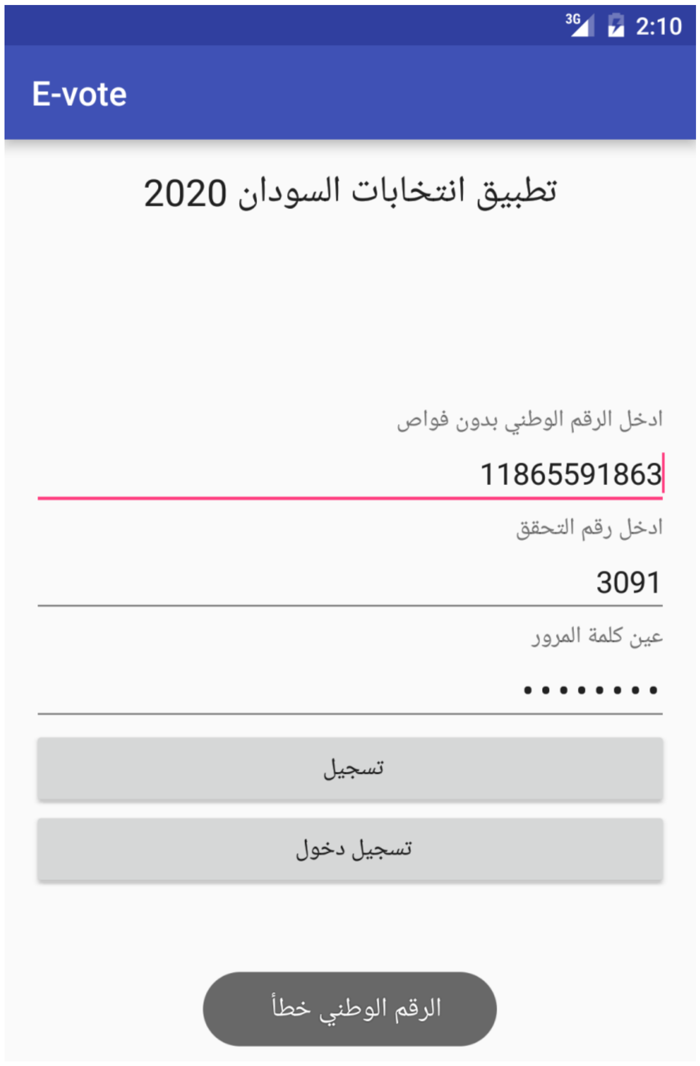
عين كلمة المرور

•••••••••

لقد سجلت مسبقاً

Figure 4.5 Already Registered

When wrong national number was used (not in the civil registry database) a message was appeared as shown in figure (4.6)



تطبيق انتخابات السودان 2020

ادخل الرقم الوطني بدون فواص

11865591863

ادخل رقم التحقق

3091

عين كلمة المرور

•••••••••

الرقم الوطني خطأ

Figure 4.6 Register Using Error National Numbers

4.3.2. **Login**:

It's the second screen appears when the user was already registered, to see this screen the users must click "sign-in”. moreover, the content of this screen are: the national number that is pre-registered, Password: Created.

The figure (4.7) below illustrates the design of this screen:

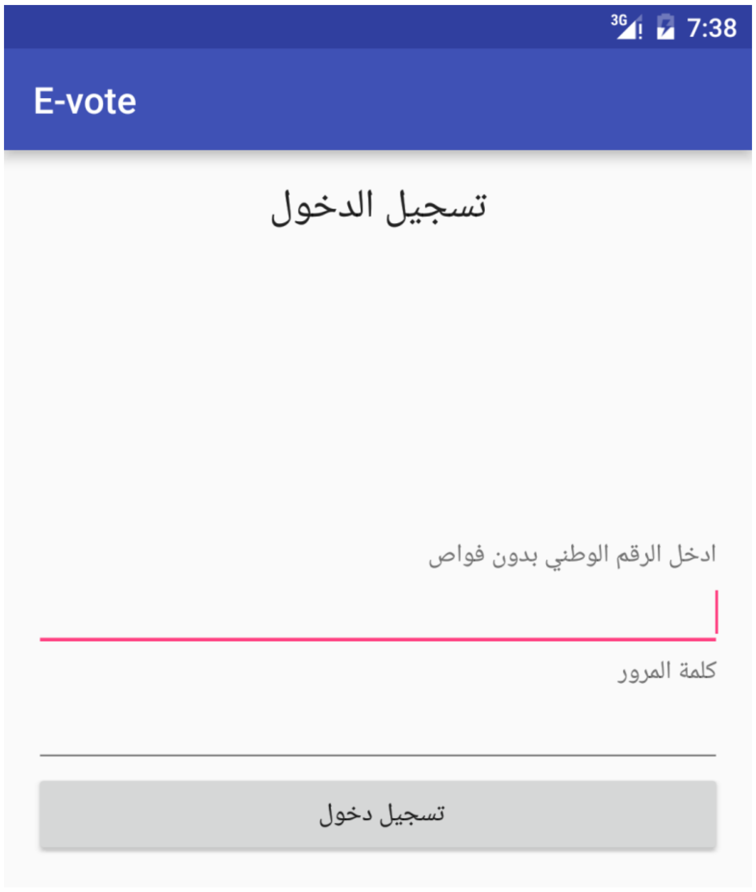
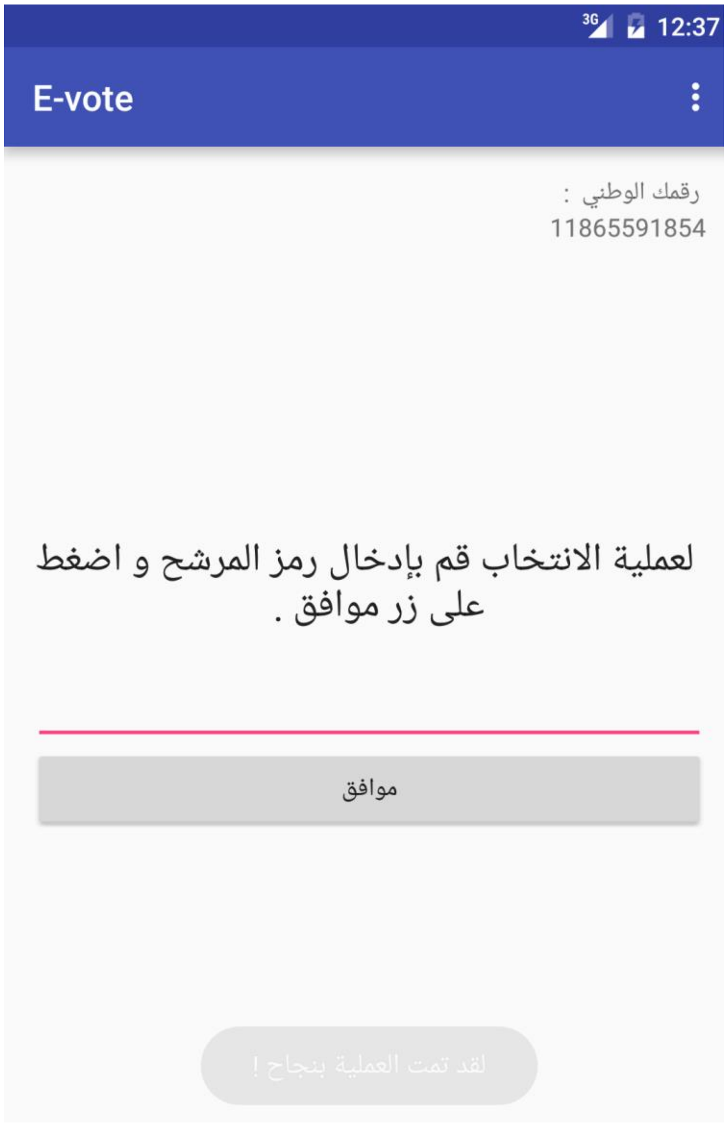


Figure 4.7 Login Screen

For a successful login, users use their national number at which the registration process was completed, because if they used a valid national number, but it was not registered in the app, the login process is considered to be a failure.

The following figure (4.8) illustrates the completion of the registration process using a valid national number Complete the registration process and open the profile page.



رقمك الوطني :
11865591854

لعملية الانتخاب قم بإدخال رمز المرشح و اضغط على زر موافق .

موافق

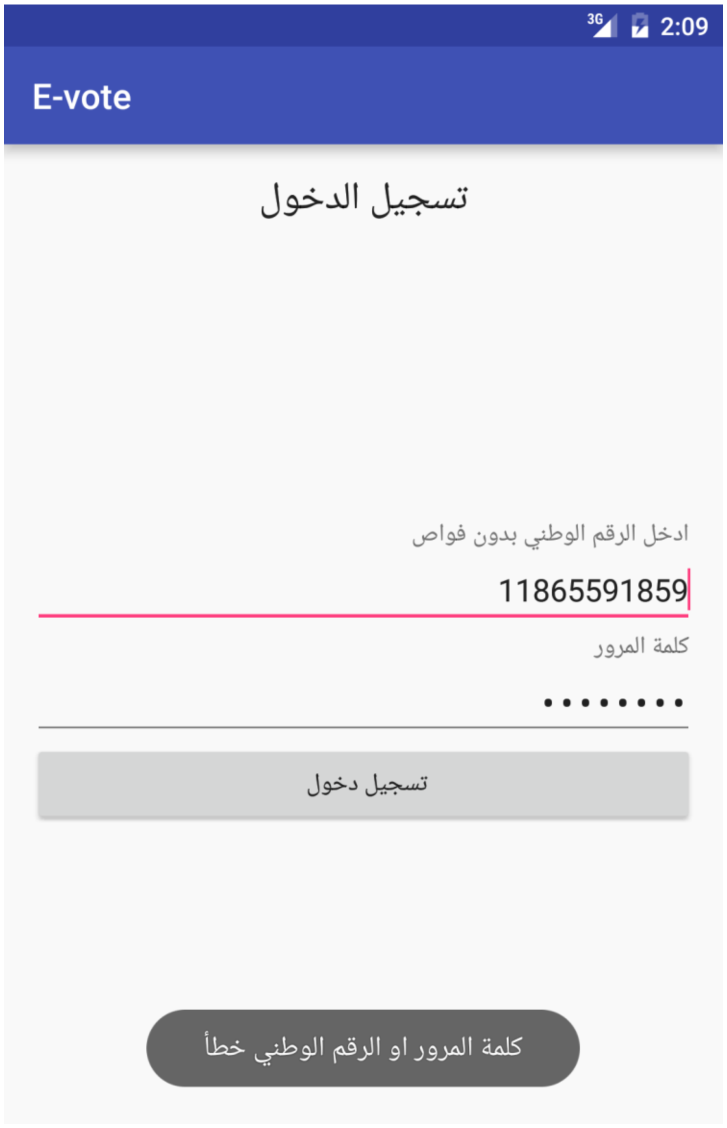
لقد تمت العملية بنجاح !

Figure 4.8 Login Successful Process

If users use Error national number that means not valid to allow login to the application.

The following figure (4.9) shows the message appear in this case

.



كلمة المرور او الرقم الوطني خطأ

Figure 4.9Login Using Error National Number

If users use the correct national number to sign in and enter their password incorrectly, a message will appear indicating this.

The following figure (4.10) shows the message

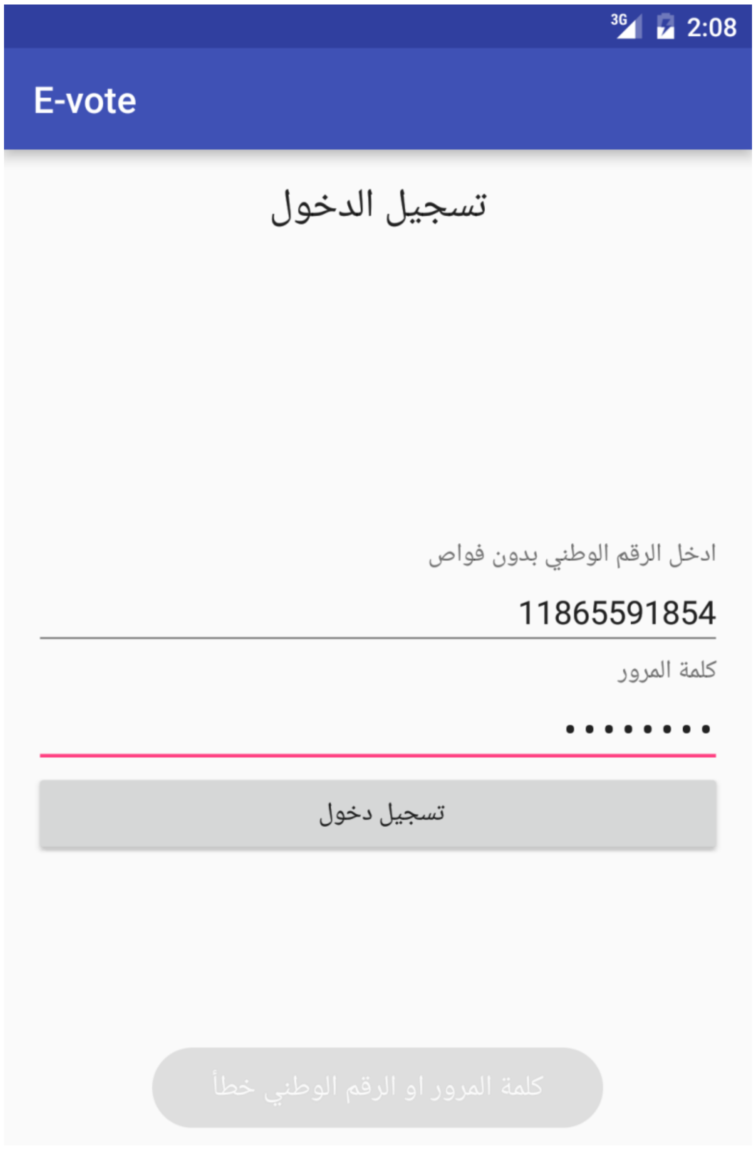


Figure 4.10 Login Using Error Password

If users use an incorrect national number, by adding or minimize the specified length (11 digits) in the field designated for entering the national number; a phrase will appear stating that (11 digits) must be entered, the following figure (4.11) explains what is mentioned.

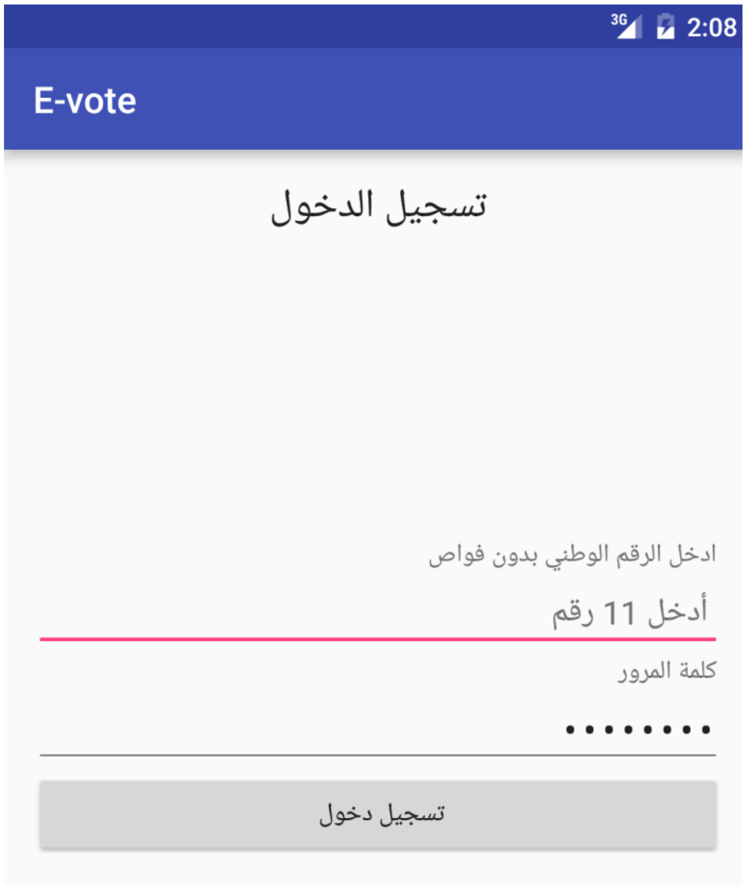


Figure 4.11Login using Long National Number

In the case of using a valid national number, but an extra password has been entered, meaning that it is more than (8 digits), it is considered incorrect in the field designated for entering the password, explaining that it must be entered (8 digits).

The following figure (4.12) shows what is mentioned:

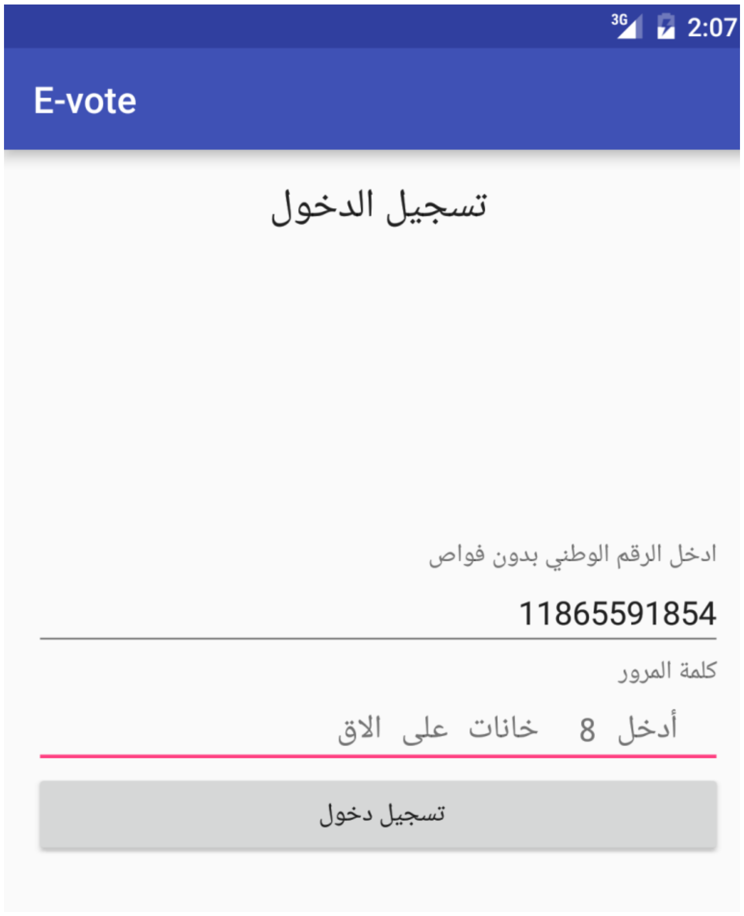
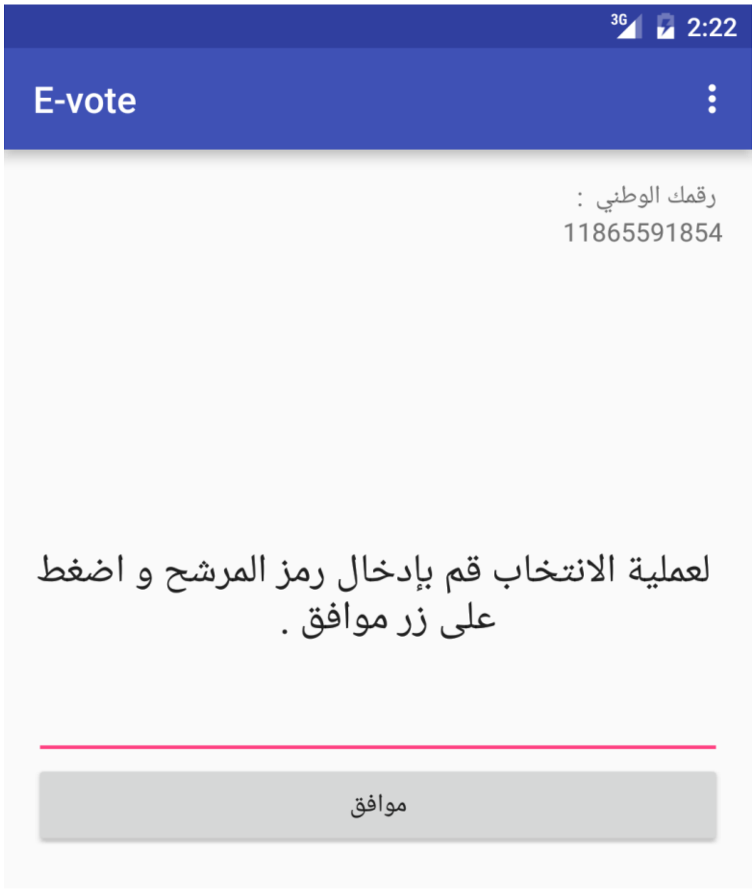


Figure 4.12 Login Using Long Password

4.3.3. The voting page

This screen appears after the user logs successfully in to the application where the user votes for an election candidate by entering the candidate code already found in the database.

The figure (4.13) below illustrates the design of this screen:



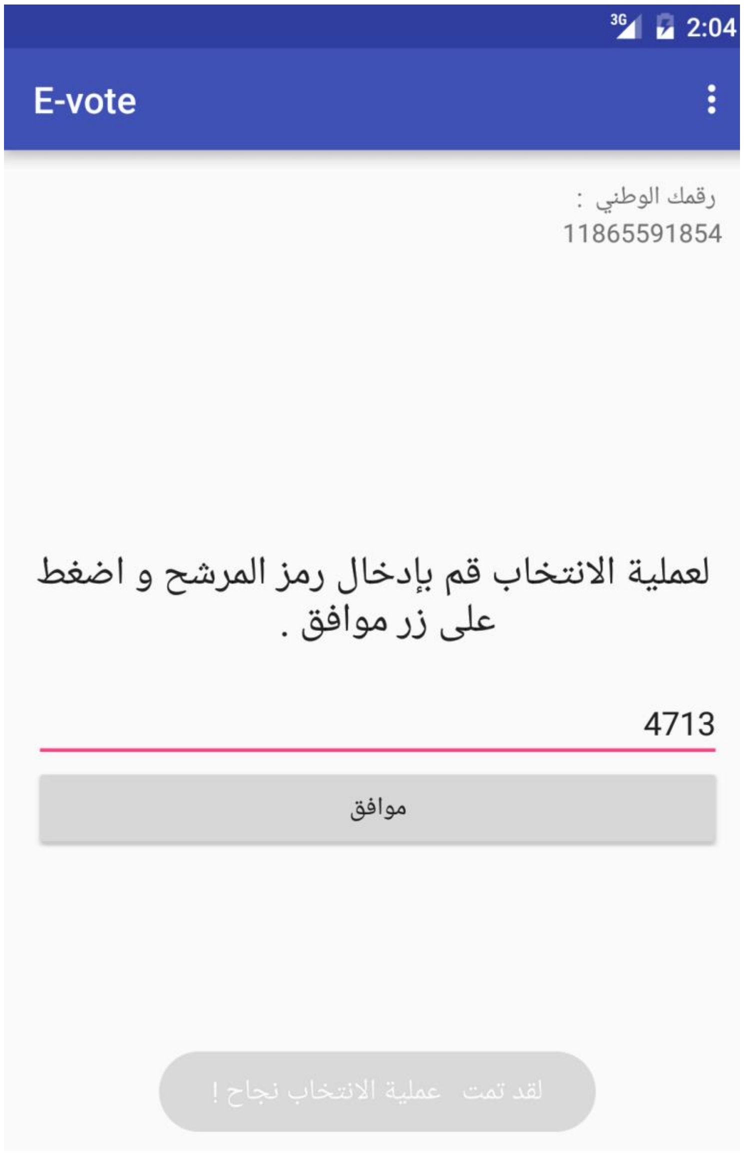
رقمك الوطني :
11865591854

لعملية الانتخاب قم بإدخال رمز المرشح و اضغط على زر موافق .

Figure 4.13 profile Screen

In the voting process, the user must enter a candidate code present in the database and the code is made of four digits.

The following figure (4.14) shows what is mentioned:



رقمك الوطني :
11865591854

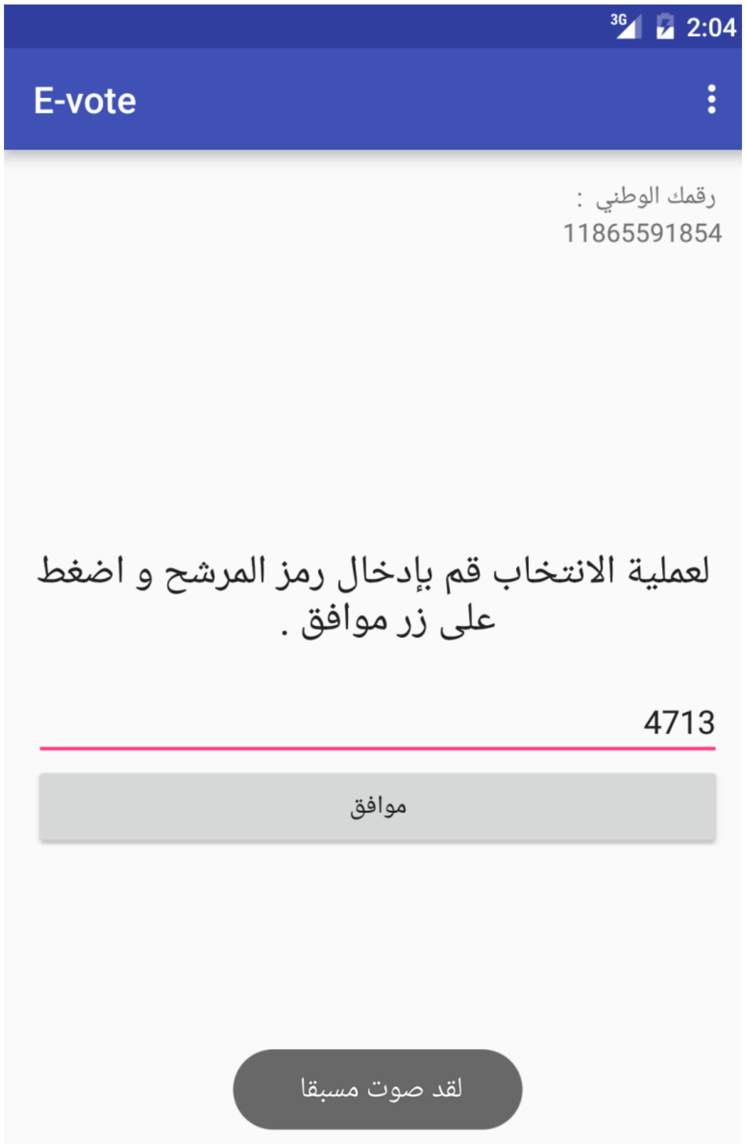
لعملية الانتخاب قم بإدخال رمز المرشح و اضغط على زر موافق .

4713

لقد تمت عملية الانتخاب نجاح !

Figure 4.14E-VotesSuccessfully

The national number is not allowed to vote more than once. The following figure (4.15) shows the message that appears in this case:



لعملية الانتخاب قم بإدخال رمز المرشح و اضغط على زر موافق .

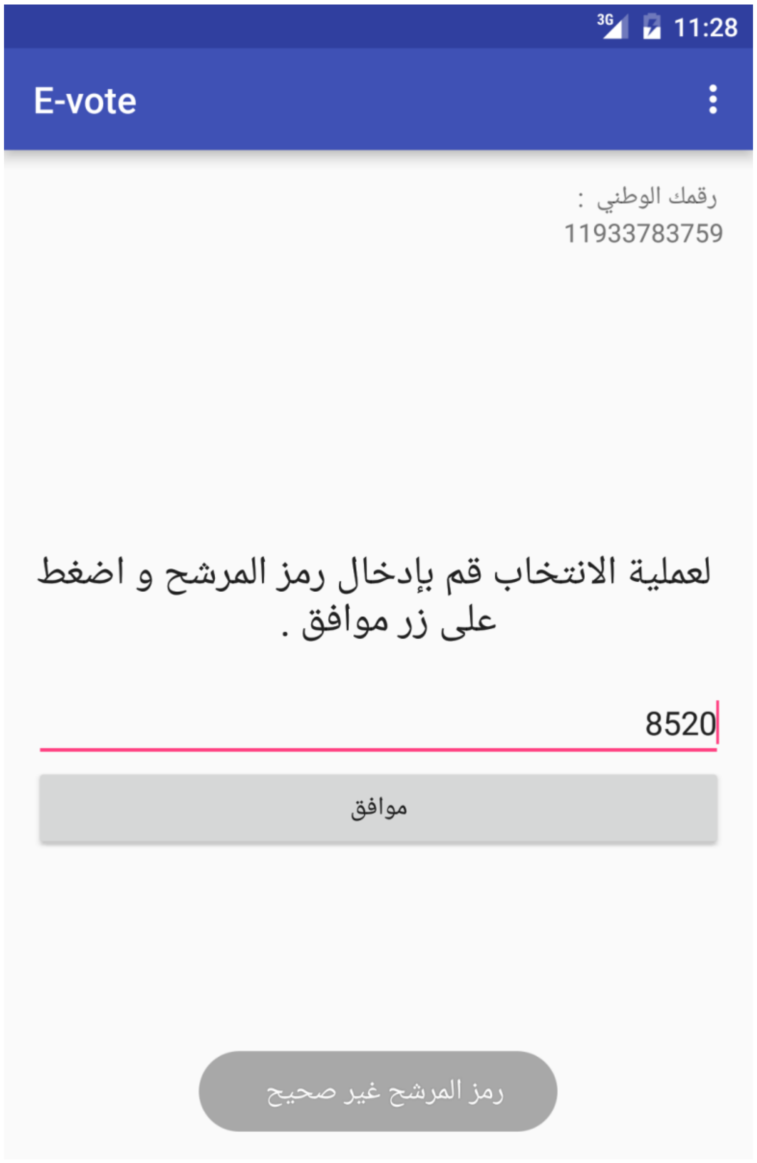
4713

لقد صوت مسبقا

Figure 4.15Already Voted

When users use the wrong candidate code (not found in the application's database),

The following figure (4.16) shows the message that appears in this case:



رقمك الوطني :
11933783759

لعملية الانتخاب قم بإدخال رمز المرشح و اضغط على زر موافق .

8520

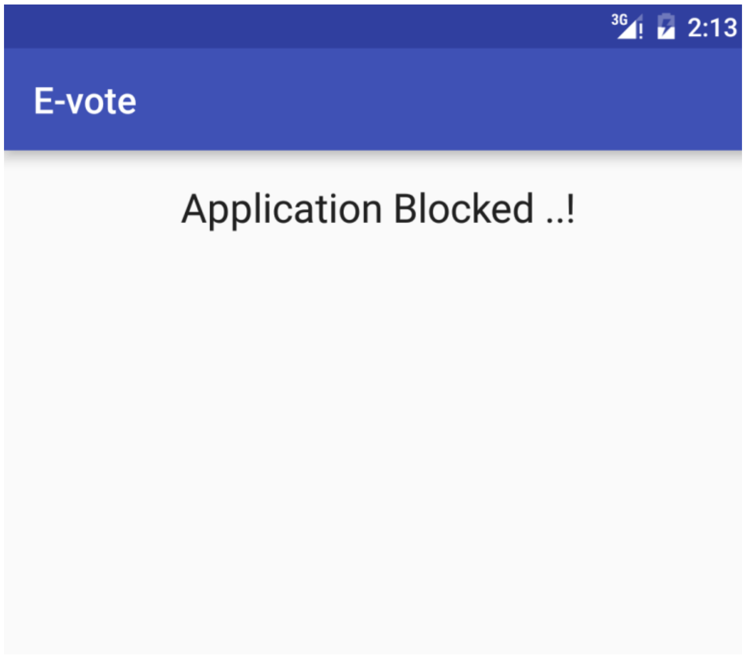
رمز المرشح غير صحيح

Figure 4.16 Vote Using Error Candidate Code

4.3.4. Block page:

This page appears in the wrong input state more than three times by the user, be it a national number or a password.

The figure (4.17) below illustrates the design of this screen:



Application Blocked ..!

Figure 4.17 Block Screen

4.4. Database design:

The application's database named (e-vote-db) and consists of five tables as follows:

The first table is the national numbers table (N-ids) and contains all the national numbers issued to Sudanese citizens only, and this table consists of the following fields as shown in below (4.2):

Table 4.2 National-Numbers Table

#	Name	Type	Collection	Attribute	Null	Default	Comments	Extra
.1	N_id	varchar(11)	Utf8_bin		No	None		
.2	name	varchar(255)	Utf8_bin		No	None		
.3	phone	varchar(17)	Utf8_bin		No	None		
.4	gander	int(11)			No	None		
.5	address	varchar(255)	Utf8_bin		No	None		
.6	b_date	Date			No	None		

The second table is the registration table for the new users of the application and depends in the registration process on the table of national numbers and therefore only allows registration users who have national numbers from Sudan and the table consists of the following fields as shown in below (4.3):

Table 4.3 Registration Table

#	Name	Type	Collection	Attribute	Null	Default	Comments	Extra
1.	N_id	varchar(11)	Utf8_bin		No	None		
2.	r-date	timestamp	Utf8_bin		No	Current_timestamp()		
3.	User_code	varchar(11)	Utf8_bin		No	None		
4.	Password	varchar(255)			No	None		

The third table is the list of candidates for the election process who are elected from the general Sudanese people and are required not to be among those registered for the voting process in order to avoid the vote of the candidate for himself, and the table consists of the following fields as shown in below (4.4):

Table 4.4 Candidates Table

#	Name	Type	Collection	Attribute	Null	Default	Comments	Extra
1.	n_id	varchar(11)	Utf8_bin		No	None		
2.	c_code	varchar(4)	Utf8_bin		No	None		
3.	discription	Text	Utf8_bin		No	None		

The fourth table is the schedule of the electoral process, provided that the elected user of the application and the candidate who is in the table of candidates, each user one opportunity to vote, selects only one candidate and vote for it, thus ensuring that the national number (the user) to vote only once. The fields of this table include a field from the registration table and the table of candidates in addition to its own fields as shown in the below (4.5):

Table 4.5 E-Vote-Operations Table

#	Name	Type	Collection	Attribute	Null	Default	Comments	Extra
1.	Id	int(11)			No	None		AUTO_INCREMENT
2.	c_code	varchar(4)	Utf8_bin		No	None		
3.	r_n_id	Varchar(11)	Utf8_bin		No	None		
4.	V_code	timestamp			No	Current_timestamp()		

The last table is the error logon records table for users where their data is saved when they login to the application and used in the process of blocking after several wrong attempts up to three times and then blocking the MAC address of the device used and the IP address used in those attempts, the following Table (4.6) shows the fields of this table:

Table 4.6 Error Login Record Table

#	Name	Type	Collection	Attribute	Null	Default	Comments	Extra
1.	id	int(11)			No	None		AUTO_INCREMENT
2.	try_date	timestamp			No	Current_timestamp()		
3.	n_id	Varchar(11)	Utf8_bin		No	None		
4.	User_ip	Varchar(255)	Utf8_bin					

4.5. Connect with PHPMyAdmin:

Most Android applications need data that is often stored in a database where the application (client) does not communicate with the database directly except through the server, which in turn deals with the database.

4.5.1. Hash Map

Hash Map is a type of Collection that stores data in a pair such that each element has a key associated with it. The pair of key and value is often known as Entry and these entries can have only unique keys.

Hash Map is a class that implements Map Interface and Extends Abstract Map class which provides the basic structural implementation of Map Interface which minimizes the efforts that are required to implement the Map interface directly in Hash Map Class.

4.5.2. REST API (Volley: 1.1.1)

A representational state transfer (REST) API is a way to provide compatibility between computer systems on the Internet. The concept was first outlined in a dissertation by Roy Fielding in 2000.

Volley is a library that makes networking for Android apps easier and most importantly, faster. Volley Library was announced by Ficus Kirkpatrick at Google I/O '13. It was first used by the Play Store team in Play Store Application and then they released it as an open source library. Although it is part of the Android Open Source Project (AOSP), Google announced in January 2017 that Volley will move to a standalone library.

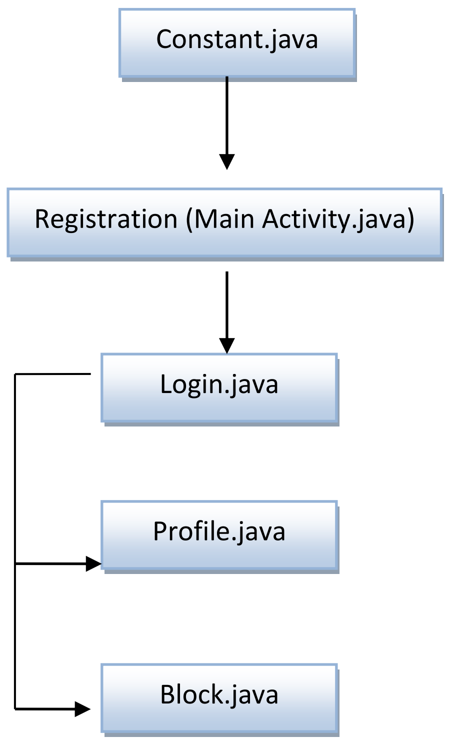
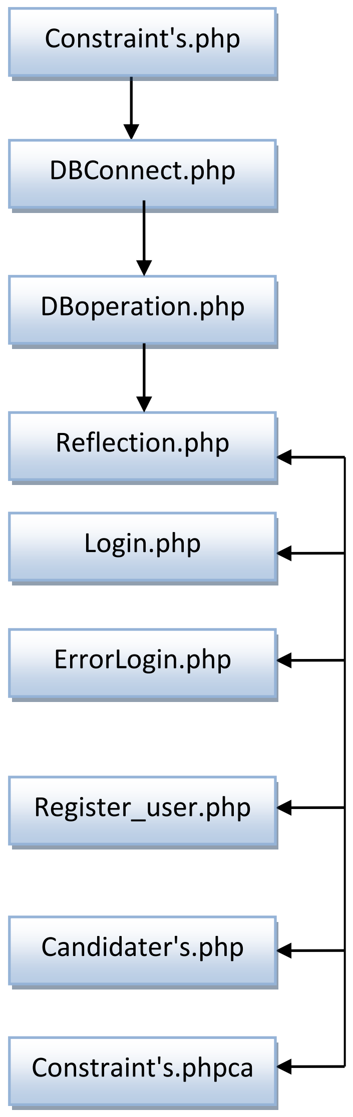
4.5.2.1. Why Volley?

- Volley can pretty much do everything with that has to do with Networking in Android.
- Volley automatically schedules all network requests such as fetching responses for image from web.
- Volley provides transparent disk and memory caching.
- Volley provides powerful cancellation request API for canceling a single request or you can set blocks of requests to cancel.
- Volley provides powerful customization abilities.
- Volley provides debugging and tracing tools.

4.6. Hierarchy files

The following table (4.7) shows the hierarchy of the files on the application (.java) and on the server (. PHP)

Table 4.7 Describing Hierarchy Files

Application's Files	Server Files
 Constant.java Registration (Main Activity.java) Login.java Profile.java Block.java	 Constraint's.php DBConnect.php DBoperation.php Reflection.php Login.php ErrorLogin.php Register_user.php Candidater's.php Constraint's.phpca

4.7. Application's files

In this section the classes' functions that created in the application were discussed with details.

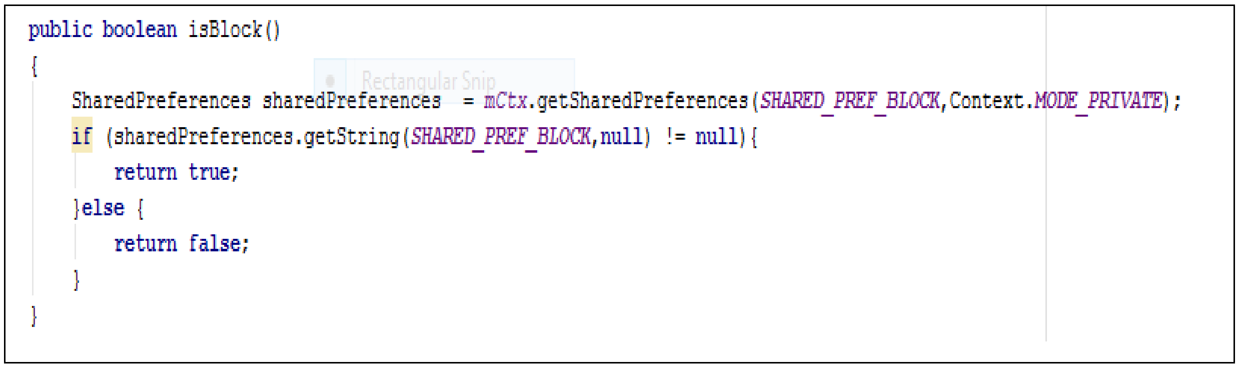
4.7.1. Sheared preference class functions:

This class is managing a set of files that contain specific information to be shared anywhere in the application, and login information and preferences were usually saved in that files. A set of functions were created to suit the application. The functions were discussed below:

4.7.1.1. IsBlockFunction

This function returns a Boolean value (true / false). It's required in a Login method (later on). This function is invoked with the beginning of the program for the first time. Which means it was invoked the**onCreate** method in **mainActivity**. It tested the user's status if it was prohibited from login. The following table (4.8) shows the function code:

Table 4.8isBlockFunction

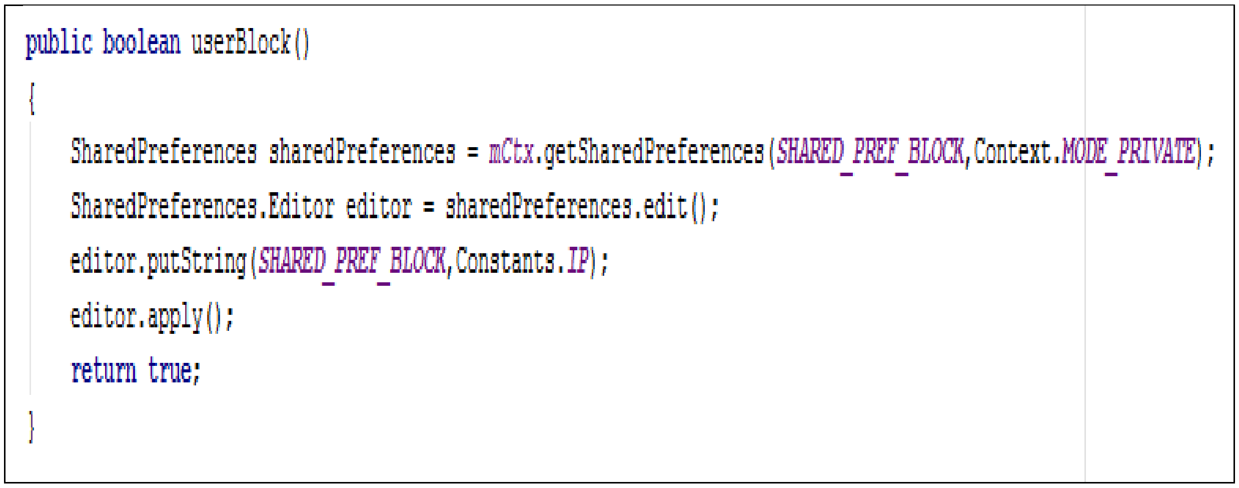


```
public boolean isBlock()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    if (sharedPreferences.getString(SHARED_PREF_BLOCK,null) != null){
        return true;
    }else {
        return false;
    }
}
```

4.7.1.2. UserBlockFunction

This function prevents the IP address in case of wrong login more than 3 times by used the (stop) function in the login file and keeping the Internet address in (context) to prevent it from entering again; the following table (4.9) shows the function code:

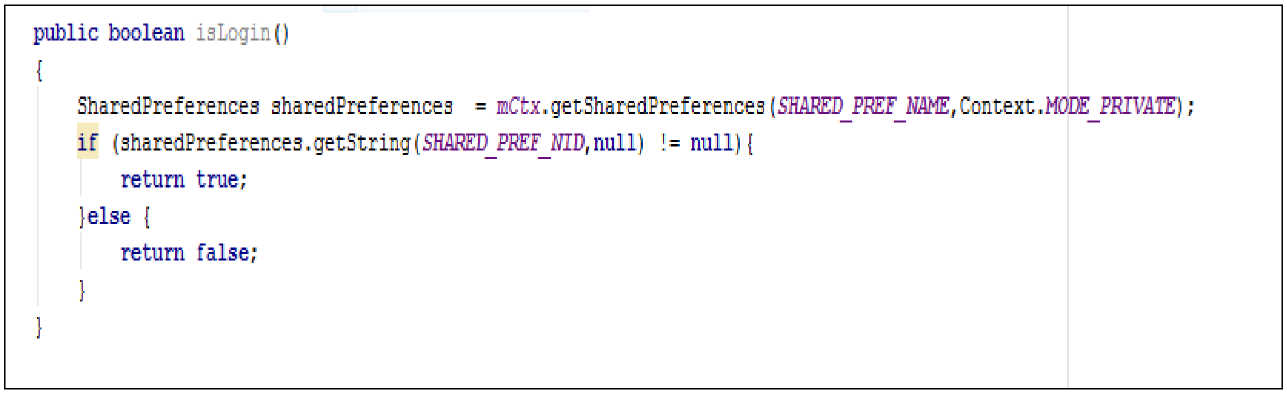
Table 4.9UserBlockFunction


```
public boolean userBlock()
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_BLOCK,Constants.IP);
    editor.apply();
    return true;
}
```

4.7.1.3. IsLoginFunction

Is a function that tests the status of the user currently logged in or not, returns the value (true / false) and checks the sharing preferences files with private mode, the following table (4.10) displays the function code:

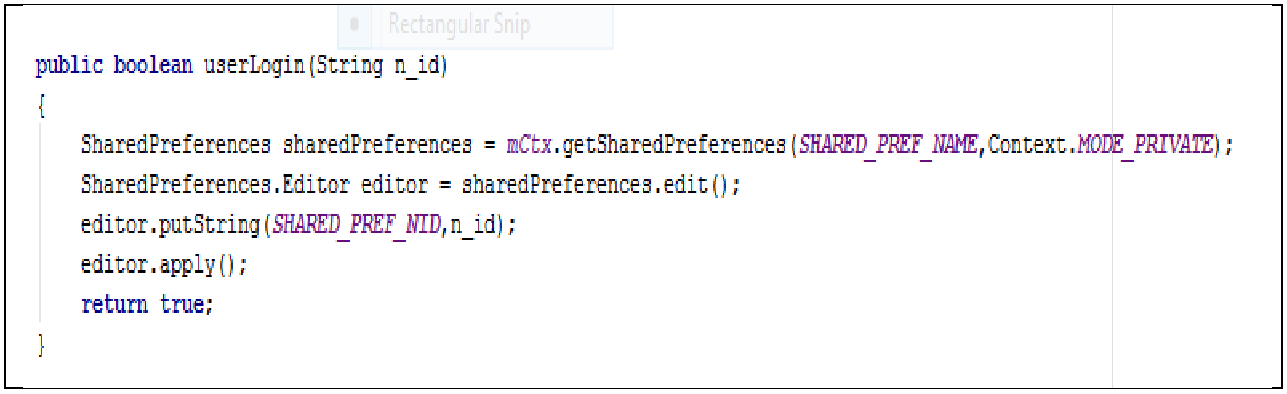
Table 4.10isLoginFunction


```
public boolean isLogin()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
    if (sharedPreferences.getString(SHARED_PREF_NID,null) != null){
        return true;
    }else {
        return false;
    }
}
```

4.7.1.4. UserLoginFunction

After the actual registration process, (enter the national number and password to activate login) this function was used to save user information in a shared preferences. This information was used to know the user have a login or not. The following table (4.11) displays the function code:

Table 4.11UserLoginFunction

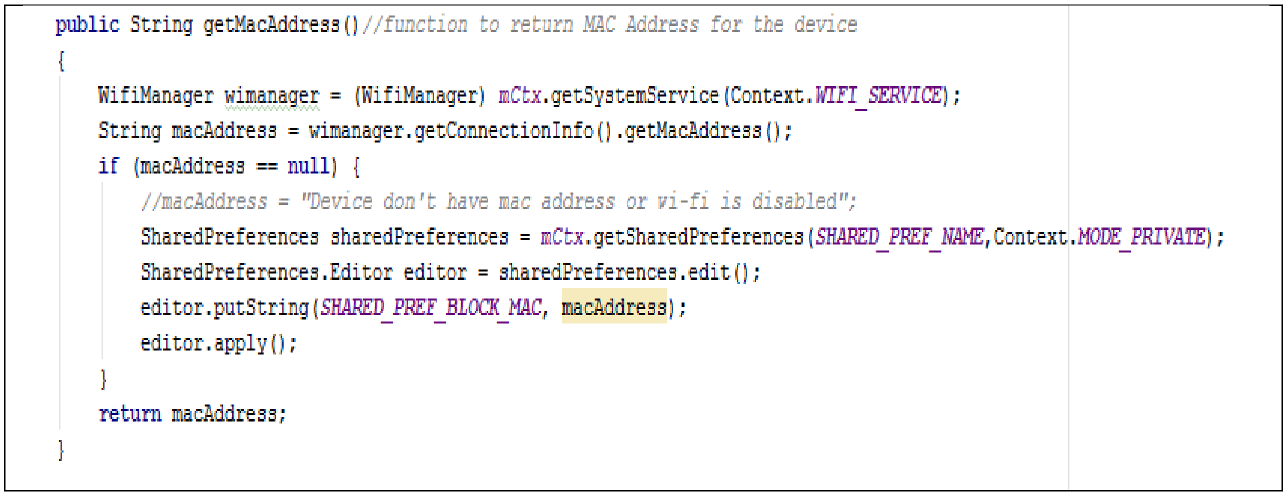

Rectangular Snip

```
public boolean userLogin(String n_id)
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_NID,n_id);
    editor.apply();
    return true;
}
```

4.7.1.5. GetMacAddressFunction

This function returns the address of the device that was logged in, and saved in the shared preference file to validate if is this MAC Address is blocked or not in Block MacFunction. The following table (4.12) that display the function code:

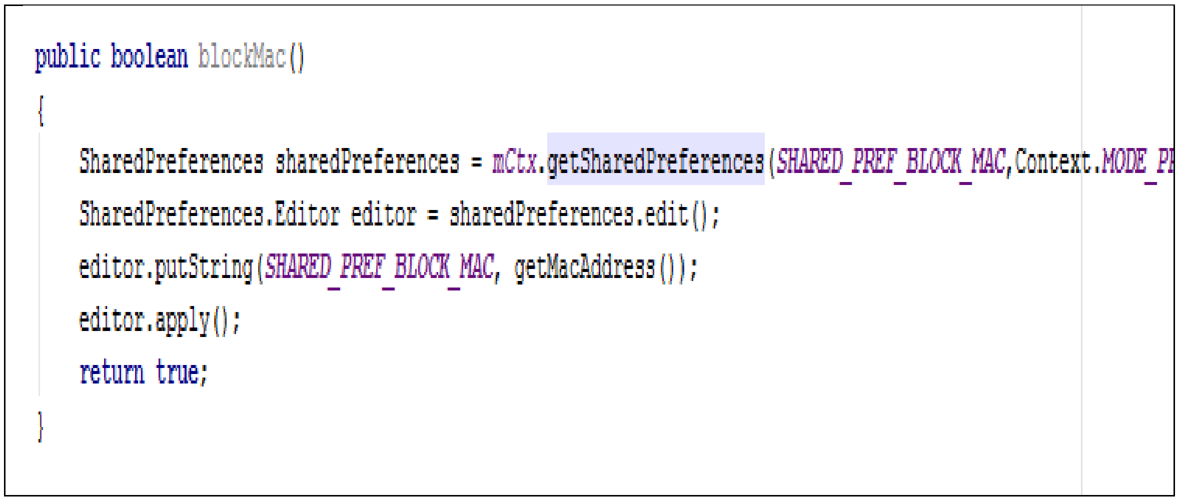
Table 4.12GetMacAddressFunction


```
public String getMacAddress()//function to return MAC Address for the device
{
    WifiManager wimanager = (WifiManager) mCtx.getSystemService(Context.WIFI_SERVICE);
    String macAddress = wimanager.getConnectionInfo().getMacAddress();
    if (macAddress == null) {
        //macAddress = "Device don't have mac address or wi-fi is disabled";
        SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString(SHARED_PREF_BLOCK_MAC, macAddress);
        editor.apply();
    }
    return macAddress;
}
```

4.7.1.6. BlockMacAddressFunction

The function prevents MAC address in case of wrong login more than 3 times using the stop function in the log file and keeping the MAC address in (context) to prevent it from entering again, the following table (4.13) shows the function code:

Table4.13 BlockMacAddress Function

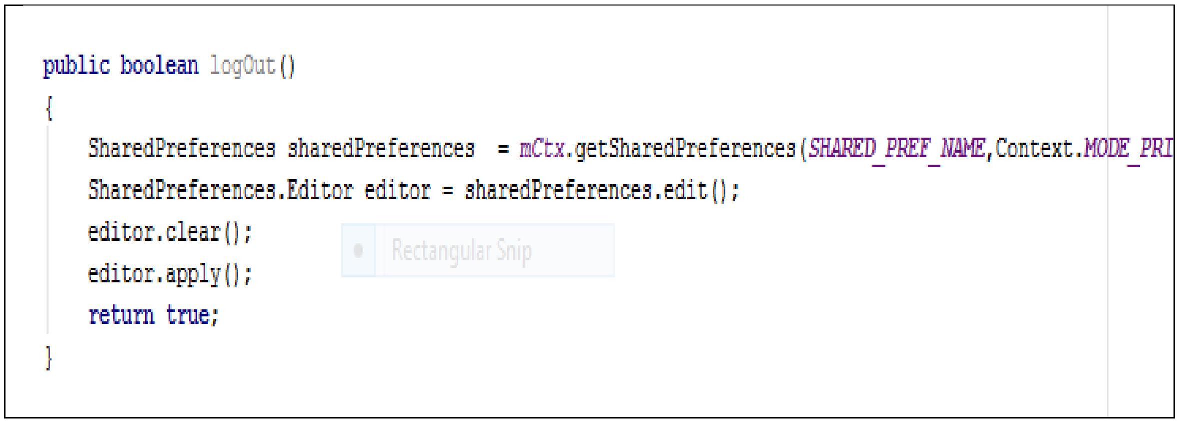


```
public boolean blockMac()
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_BLOCK_MAC,Context.MODE_PR
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_BLOCK_MAC, getMacAddress());
    editor.apply();
    return true;
}
```

4.7.1.7. Logout Function

Clear all (contexts) related to login file from the sheared preferences, the following table (4.14) shows the function code:

Table 4.14 Logout Function

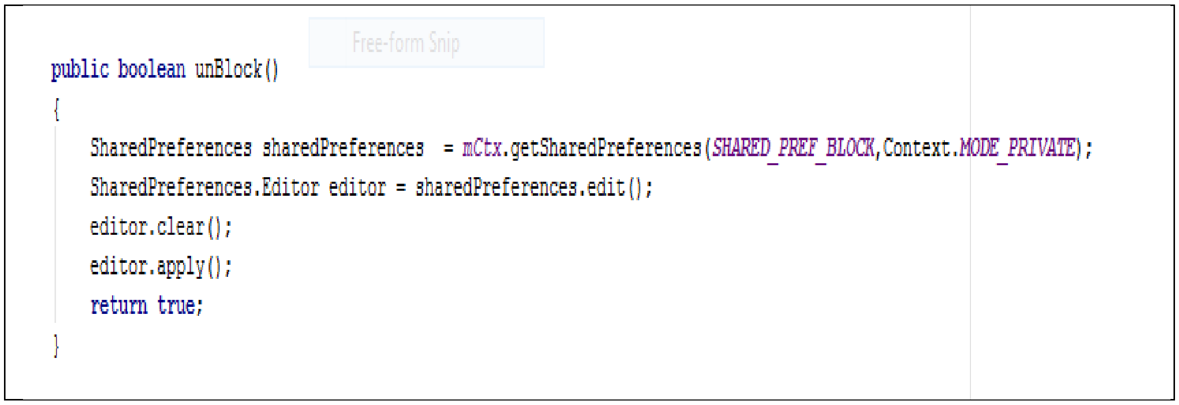


```
public boolean logOut()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRI
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();
    return true;
}
```

4.7.1.8. Unblock Function

Clear all data related to the IP Address and MAC Address from sheared preferences file, the following table (4.15) shows the function code:

Table 4.15 Unblock Function



```
public boolean unBlock()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();
    return true;
}
```

4.7.2. Login class Functions:

4.7.2.1. Login Function:

Using the JSON object class already in a group to send the logon data to the server and bypassing the protected Hash Map to all exceptions when PARAMS was sent by Http Request (POST), actually on the server, the PARAMS was expected by Http Request specified. PHP sends a response in the array (json_encode). This array has two keys one is for errors (True/false) and the other for data (Message) that needed to use it in the app.

In case there were no errors in http Request (false value) the following steps were done:

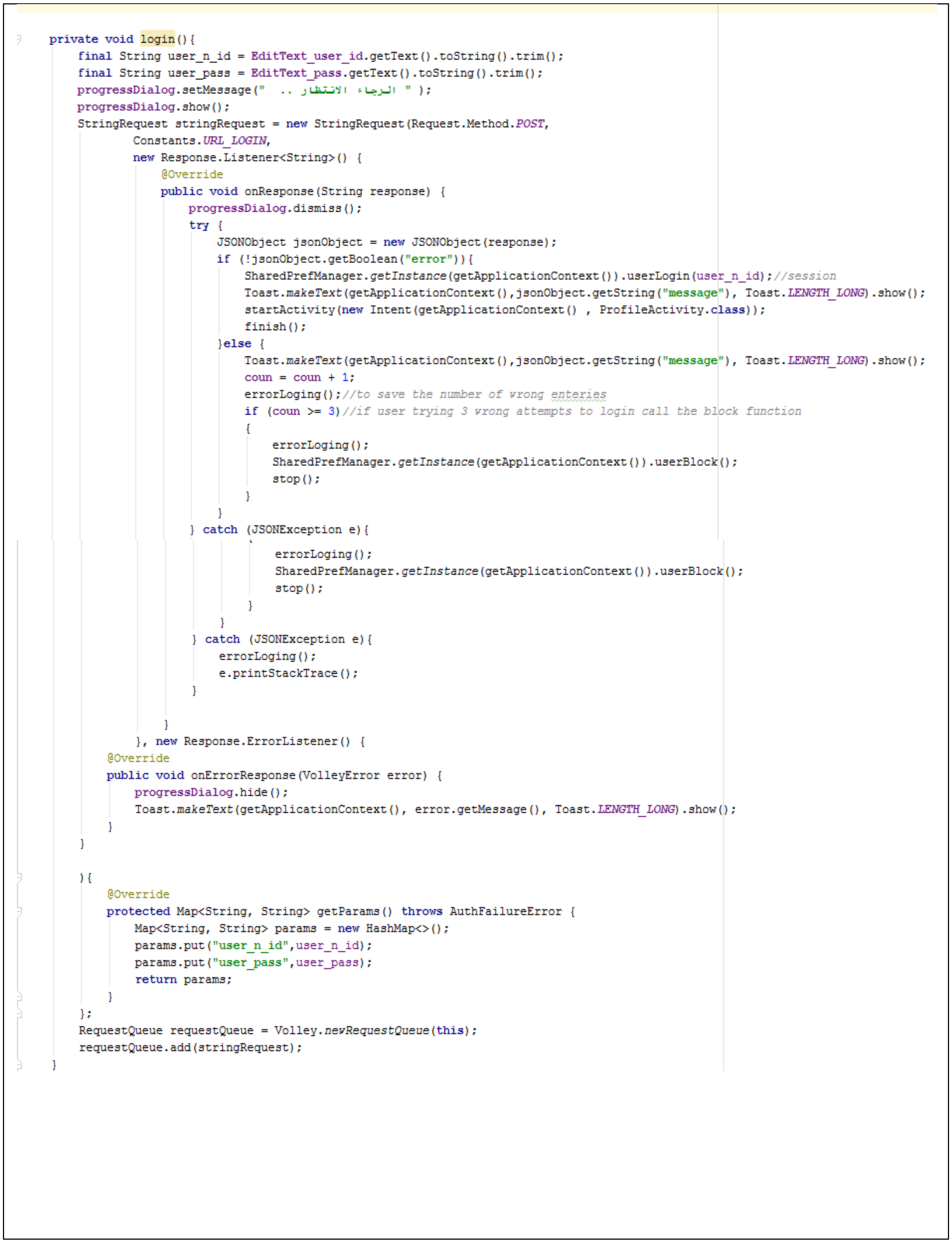
1. The login data was saved in a sheared preferences file
2. IP and MAC Address were got.
3. Profile activity was started.

Otherwise:

1. One value will be added in error counter.
2. When the errors counter equal 3, errorLogin() function will be invoked.
3. Block activity Stop () functionwillstart.

Login function was used in the case of the function isLogin() returned false value then the user creates a new login so that a new session will be done to save the user login, In all this steps were used message key to display (Toast) to explain user login status, The following table (4.16) shows the function code:

Table 4.16 Login Function

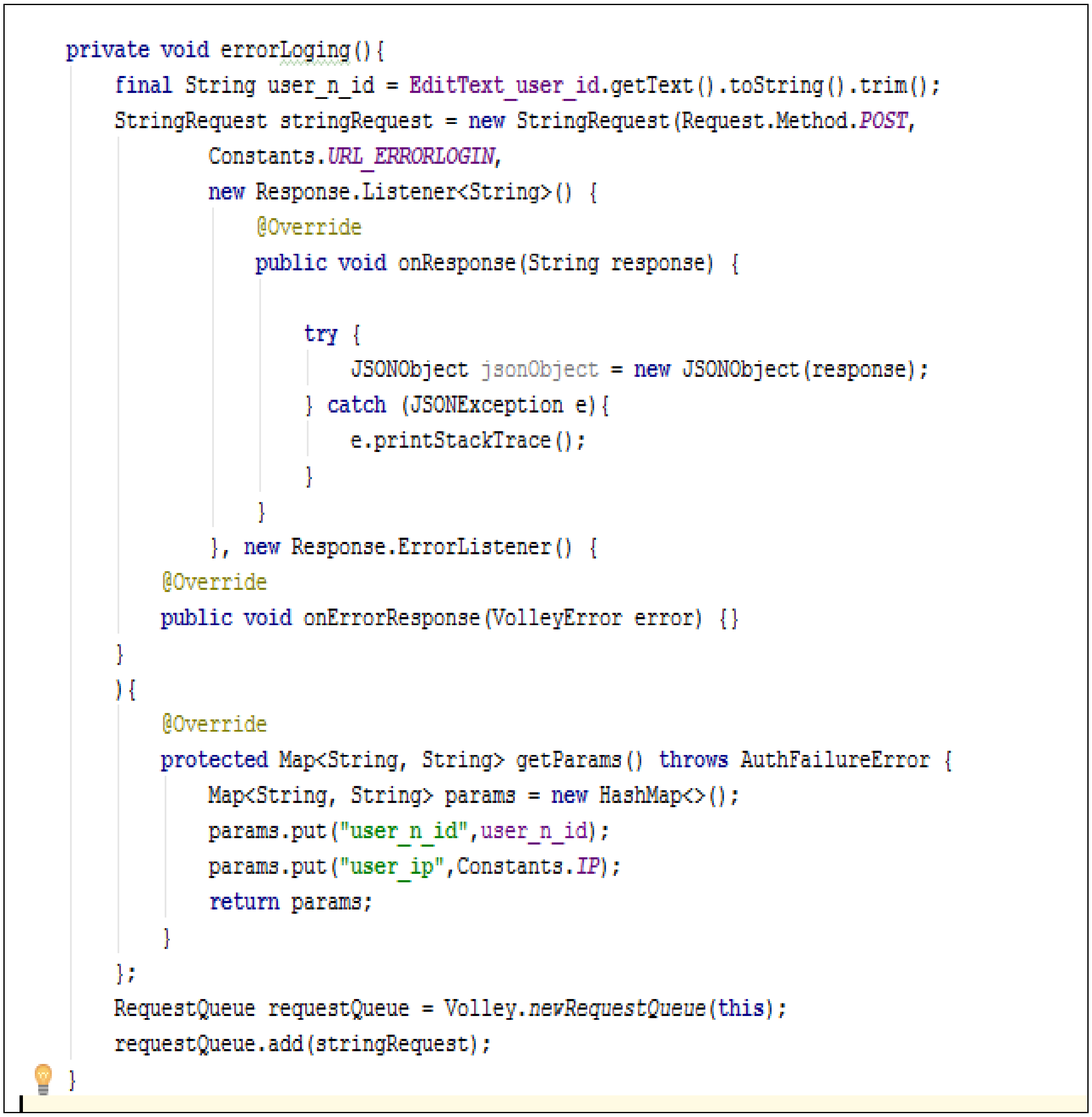


```java
private void login(){
    final String user_n_id = EditText_user_id.getText().toString().trim();
    final String user_pass = EditText_pass.getText().toString().trim();
    progressDialog.setMessage(" الرجاء الانتظار .. " );
    progressDialog.show();
    StringRequest stringRequest = new StringRequest(Request.Method.POST,
            Constants.URL_LOGIN,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    progressDialog.dismiss();
                    try {
                        JSONObject jsonObject = new JSONObject(response);
                        if (!jsonObject.getBoolean("error")){
                            SharedPrefManager.getInstance(getApplicationContext()).userLogin(user_n_id);//session
                            Toast.makeText(getApplicationContext(),jsonObject.getString("message"), Toast.LENGTH_LONG).show();
                            startActivity(new Intent(getApplicationContext() , ProfileActivity.class));
                            finish();
                        }else {
                            Toast.makeText(getApplicationContext(),jsonObject.getString("message"), Toast.LENGTH_LONG).show();
                            coun = coun + 1;
                            errorLoging();//to save the number of wrong enteries
                            if (coun >= 3)//if user trying 3 wrong attempts to login call the block function
                            {
                                errorLoging();
                                SharedPrefManager.getInstance(getApplicationContext()).userBlock();
                                stop();
                            }
                        }
                    } catch (JSONException e){

                                errorLoging();
                                SharedPrefManager.getInstance(getApplicationContext()).userBlock();
                                stop();
                            }
                        }
                    } catch (JSONException e){
                        errorLoging();
                        e.printStackTrace();
                    }

                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            progressDialog.hide();
            Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
        }
    }

    ){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("user_n_id",user_n_id);
            params.put("user_pass",user_pass);
            return params;
        }
    };
    RequestQueue requestQueue = Volley.newRequestQueue(this);
    requestQueue.add(stringRequest);
}
```

4.7.2.2. ErrorLoginFunction:

Save all information about User login data if present in the database because when N-Id is not found in the civil registry database it is ignored. The following table (4.17) shows the function code:

Table 4.17ErrorLogin Function

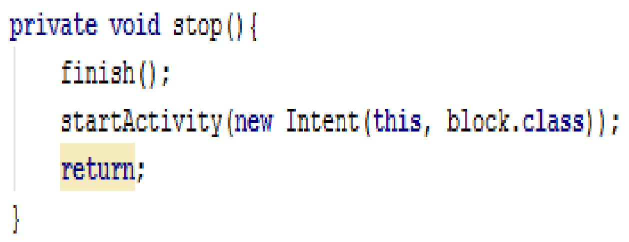


```
private void errorLoging(){
    final String user_n_id = EditText_user_id.getText().toString().trim();
    StringRequest stringRequest = new StringRequest(Request.Method.POST,
            Constants.URL_ERRORLOGIN,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

                    try {
                        JSONObject jsonObject = new JSONObject(response);
                    } catch (JSONException e){
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {}
    }
    ){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("user_n_id",user_n_id);
            params.put("user_ip",Constants.IP);
            return params;
        }
    };
    RequestQueue requestQueue = Volley.newRequestQueue(this);
    requestQueue.add(stringRequest);
}
```

4.7.2.3. Stop Function

The task of this function is to finish the current activity and start block activity, the following table (4.18) shows the function code:

Table 4.18 Stop Function



```
private void stop(){
    finish();
    startActivity(new Intent(this, block.class));
    return;
}
```

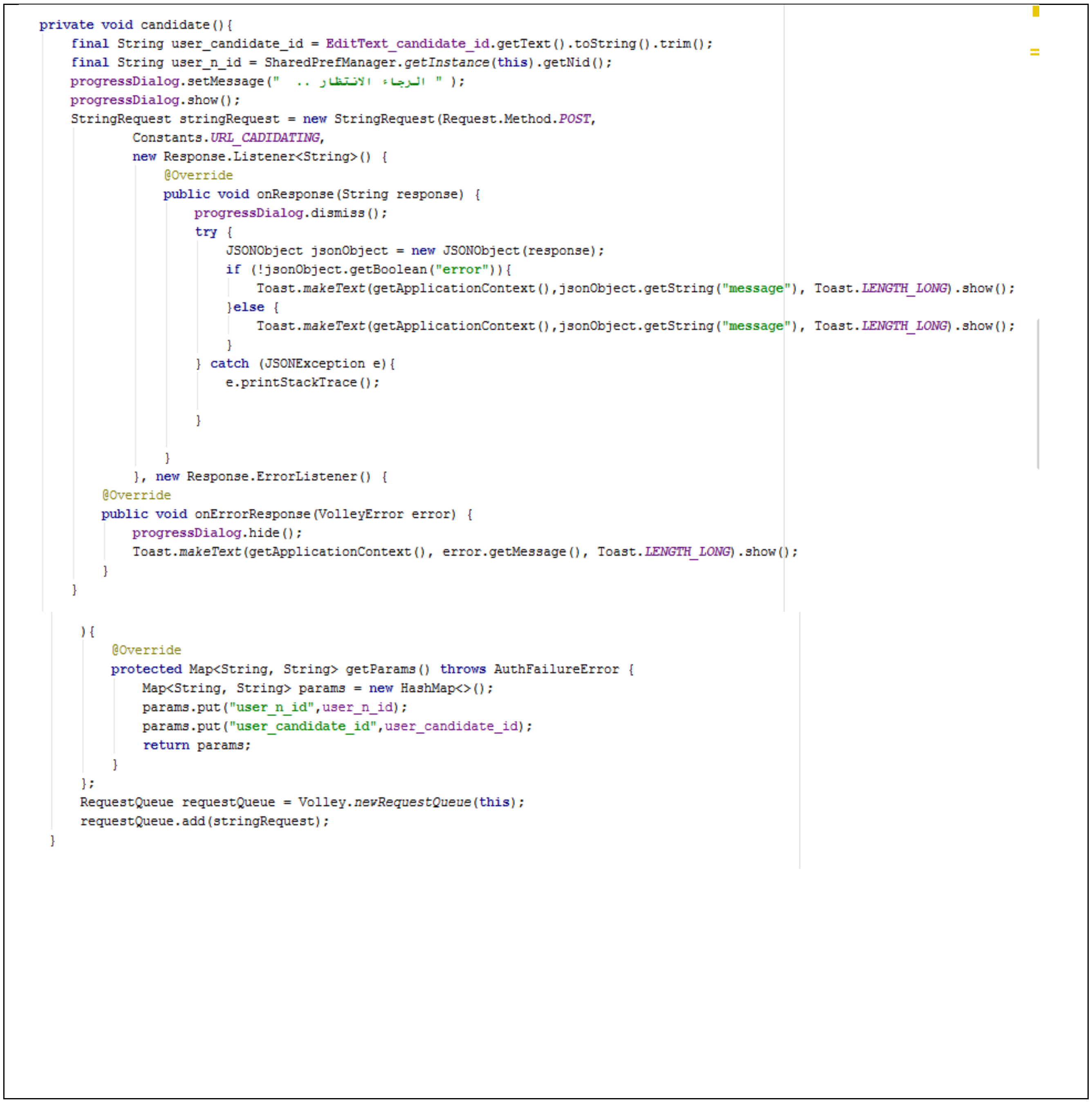
4.7.3. Profile class functions

4.7.3.1. Candidate Function:

It is the only function in the application that is validated by a function in the server; where the national number in share preference file is taken after a successful login process. And send it to the server to complete the voting process.

Voting operations are validated to ensure that the voting process is not from a pre-voting national number, or voting was done for someone who is not a candidate or a non-existent (the voting function was explained in the server functions section).The following table (4.19) shows the function code:

Table 4.19 Candidate Function



```java
private void candidate(){
    final String user_candidate_id = EditText_candidate_id.getText().toString().trim();
    final String user_n_id = SharedPrefManager.getInstance(this).getNid();
    progressDialog.setMessage("  .. الرجاء الانتظار " );
    progressDialog.show();
    StringRequest stringRequest = new StringRequest(Request.Method.POST,
            Constants.URL_CADIDATING,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    progressDialog.dismiss();
                    try {
                        JSONObject jsonObject = new JSONObject(response);
                        if (!jsonObject.getBoolean("error")){
                            Toast.makeText(getApplicationContext(),jsonObject.getString("message"), Toast.LENGTH_LONG).show();
                        }else {
                            Toast.makeText(getApplicationContext(),jsonObject.getString("message"), Toast.LENGTH_LONG).show();
                        }
                    } catch (JSONException e){
                        e.printStackTrace();

                    }

                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            progressDialog.hide();
            Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
        }
    }

     ){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("user_n_id",user_n_id);
            params.put("user_candidate_id",user_candidate_id);
            return params;
        }
     };
     RequestQueue requestQueue = Volley.newRequestQueue(this);
     requestQueue.add(stringRequest);
 }
```

4.7.4. Block Class function:

In this class checked if the user was block or not by using isBlock function from share preferences class. The flowing figure (4.18) shows the flowchart for Block process:

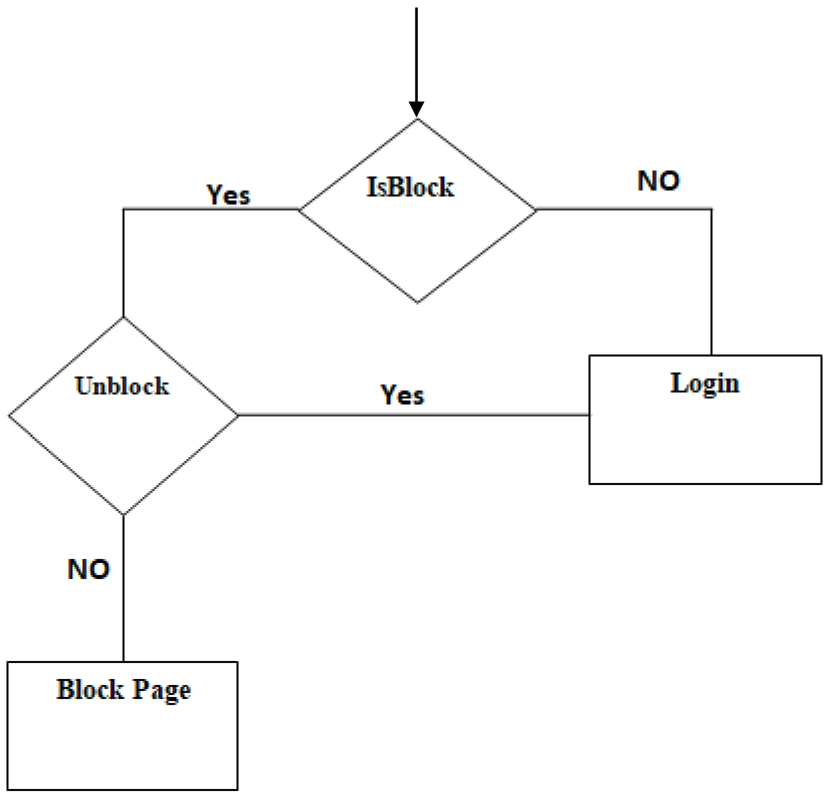


Figure 4.18Block Process

4.8. Server Files

This section discussed PHP files that exist in the server; in the below show the details of it.

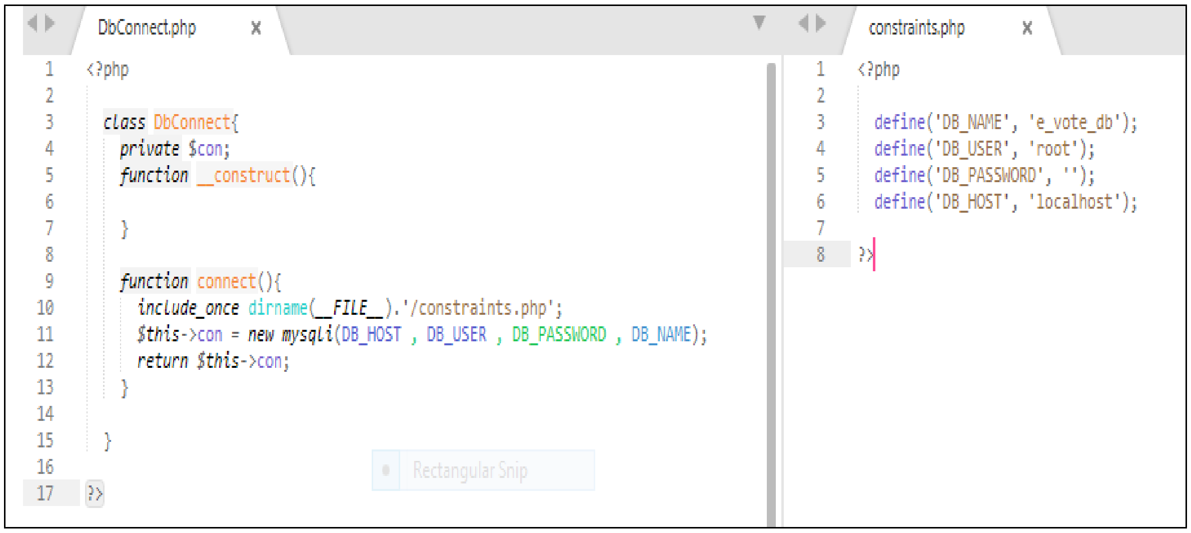
4.8.1. PHP classes

In this class was described the PHP connectivity with MySQL database using MySQLi function, used OOP concept to invoked the method in the required places, this class used several constants were registered in constraints.php file

4.8.1.1. DB Connnect.php

To connected with MySQL there must be: server host, user name, database name and database password that were described in the table (4.20) below

Table 4.20BD Connect



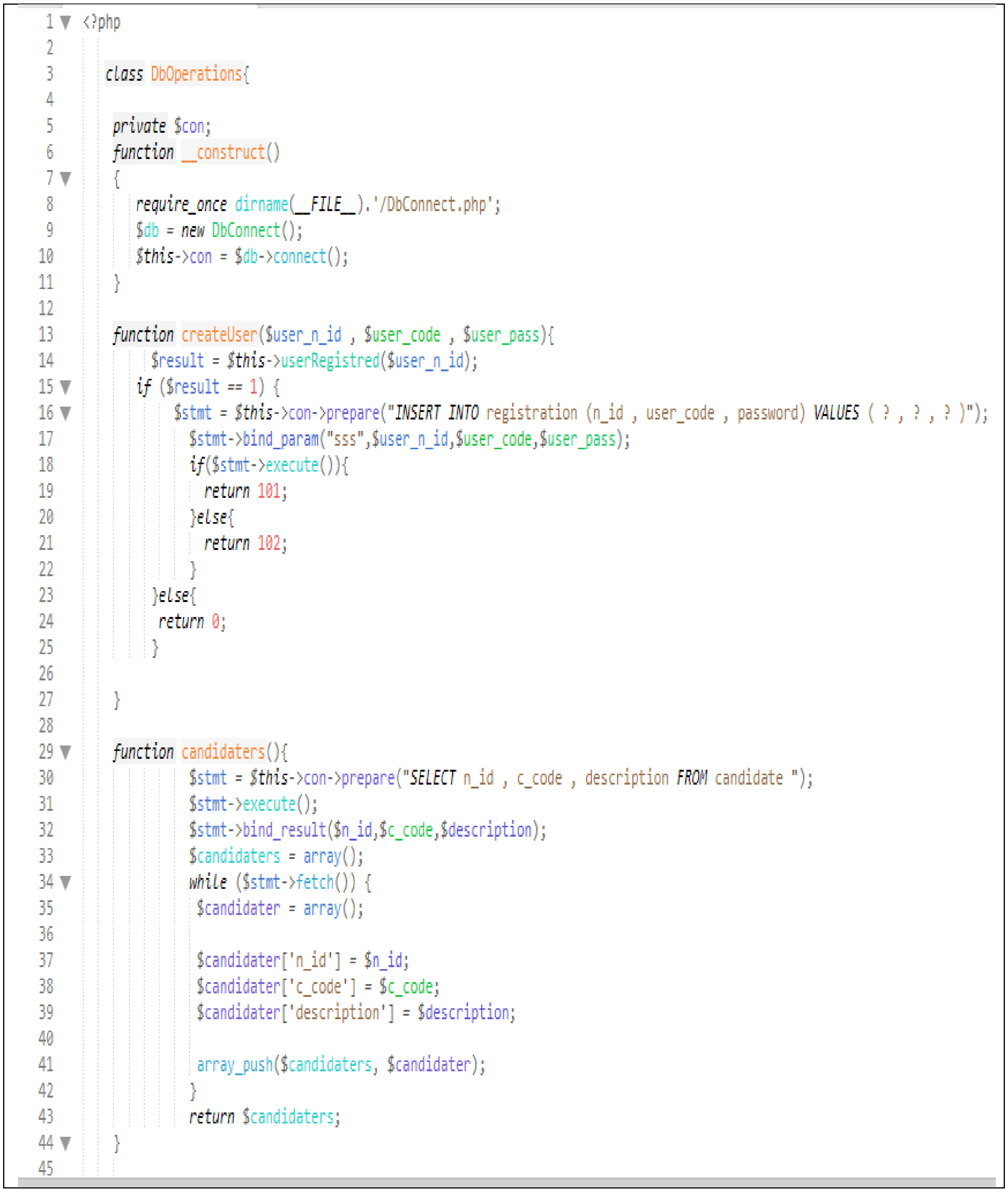
```php
<?php

 class DbConnect{
   private $con;
   function __construct(){

   }

   function connect(){
     include_once dirname(__FILE__).'/constraints.php';
     $this->con = new mysqli(DB_HOST , DB_USER , DB_PASSWORD , DB_NAME);
     return $this->con;
   }

 }

?>
```

```php
<?php

 define('DB_NAME', 'e_vote_db');
 define('DB_USER', 'root');
 define('DB_PASSWORD', '');
 define('DB_HOST', 'localhost');

?>
```

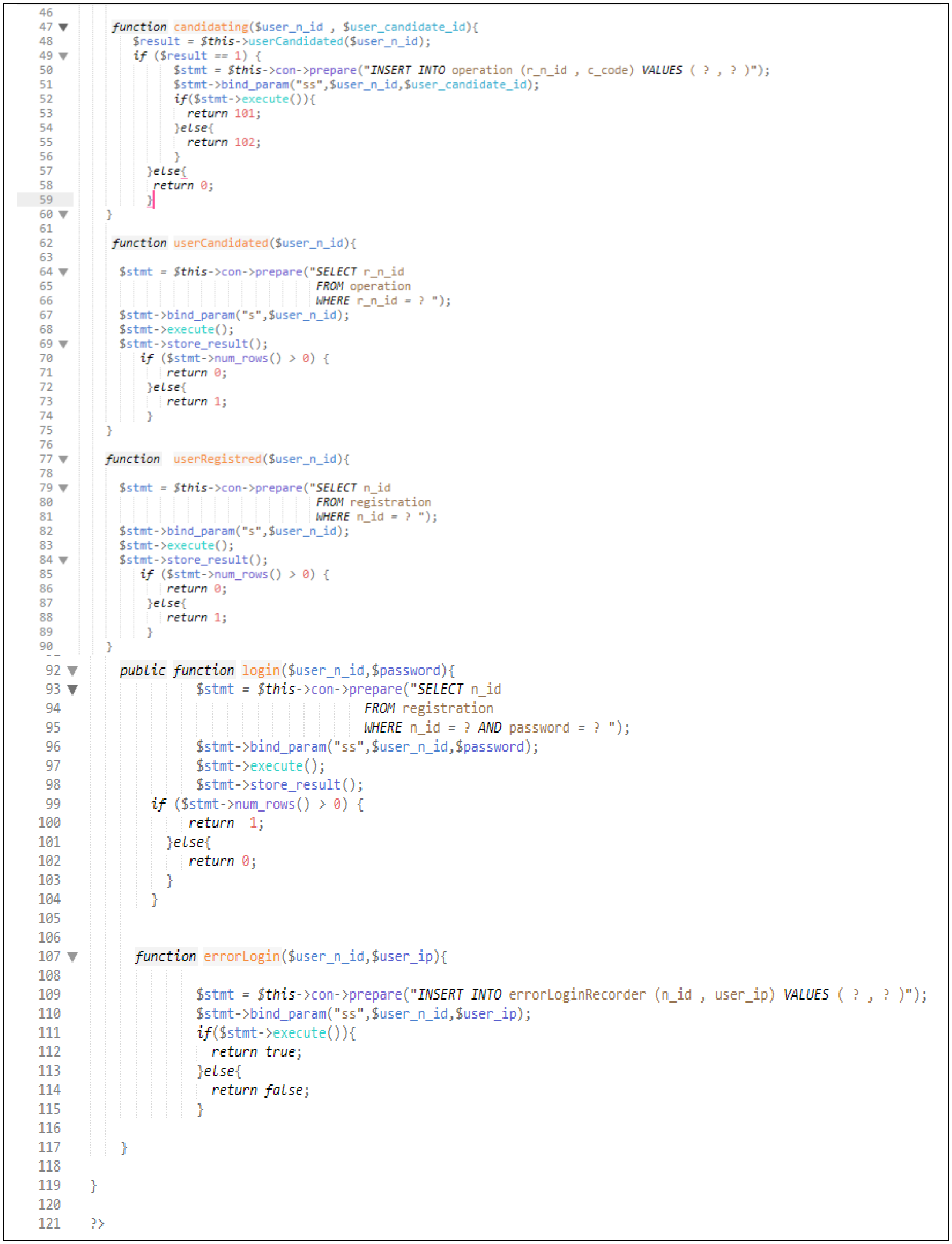
4.8.1.2. DB Operations.php

In this class, there are all the main functions that needed to completed the processes of registration, logging in, voting, etc., and the functions called in this class were called in the places that needed with used the OOP concept, or where every process that taken place in the application required entering or viewing data, another file was created for it with an extension PHP, which received and sent data via JSON used The Super Global Post. The following table (4.21) shows the functions in the DBOperations.php

Table (4.21) DBOperation.php Functions



```php
<?php

class DbOperations{

 private $con;
 function __construct()
 {
    require_once dirname(__FILE__).'/DbConnect.php';
    $db = new DbConnect();
    $this->con = $db->connect();
 }

 function createUser($user_n_id , $user_code , $user_pass){
      $result = $this->userRegistred($user_n_id);
    if ($result == 1) {
         $stmt = $this->con->prepare("INSERT INTO registration (n_id , user_code , password) VALUES ( ? , ? , ? )");
           $stmt->bind_param("sss",$user_n_id,$user_code,$user_pass);
           if($stmt->execute()){
             return 101;
           }else{
             return 102;
           }
      }else{
       return 0;
      }

 }

 function candidaters(){
           $stmt = $this->con->prepare("SELECT n_id , c_code , description FROM candidate ");
           $stmt->execute();
           $stmt->bind_result($n_id,$c_code,$description);
           $candidaters = array();
           while ($stmt->fetch()) {
            $candidater = array();

            $candidater['n_id'] = $n_id;
            $candidater['c_code'] = $c_code;
            $candidater['description'] = $description;

            array_push($candidaters, $candidater);
           }
           return $candidaters;
 }

```



```
    function candidating($user_n_id , $user_candidate_id){
      $result = $this->userCandidated($user_n_id);
      if ($result == 1) {
            $stmt = $this->con->prepare("INSERT INTO operation (r_n_id , c_code) VALUES ( ? , ? )");
            $stmt->bind_param("ss",$user_n_id,$user_candidate_id);
            if($stmt->execute()){
              return 101;
            }else{
              return 102;
            }
        }else{
         return 0;
        }
   }

    function userCandidated($user_n_id){

     $stmt = $this->con->prepare("SELECT r_n_id
                                  FROM operation
                                  WHERE r_n_id = ? ");
     $stmt->bind_param("s",$user_n_id);
     $stmt->execute();
     $stmt->store_result();
        if ($stmt->num_rows() > 0) {
           return 0;
        }else{
           return 1;
        }
   }

   function  userRegistred($user_n_id){

     $stmt = $this->con->prepare("SELECT n_id
                                  FROM registration
                                  WHERE n_id = ? ");
     $stmt->bind_param("s",$user_n_id);
     $stmt->execute();
     $stmt->store_result();
        if ($stmt->num_rows() > 0) {
           return 0;
        }else{
           return 1;
        }
   }

    public function login($user_n_id,$password){
                $stmt = $this->con->prepare("SELECT n_id
                                             FROM registration
                                             WHERE n_id = ? AND password = ? ");
                $stmt->bind_param("ss",$user_n_id,$password);
                $stmt->execute();
                $stmt->store_result();
        if ($stmt->num_rows() > 0) {
            return  1;
          }else{
            return 0;
          }
        }


      function errorLogin($user_n_id,$user_ip){

                $stmt = $this->con->prepare("INSERT INTO errorLoginRecorder (n_id , user_ip) VALUES ( ? , ? )");
                $stmt->bind_param("ss",$user_n_id,$user_ip);
                if($stmt->execute()){
                  return true;
                }else{
                  return false;
                }

    }

}

?>
```

4.8.2. Reflection Class

The process of invoking server's functions is controlled Reflection class.

The following figure (4.19) shows the implementation of reflection class process in the application.

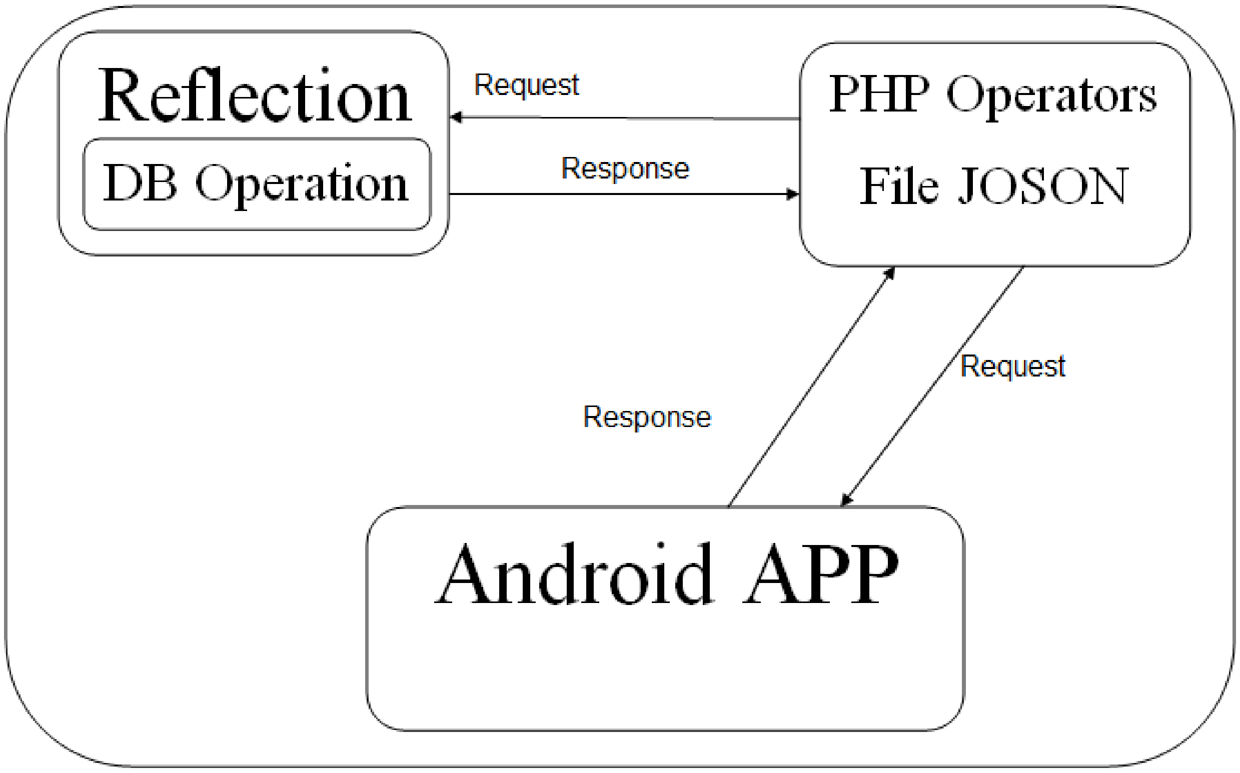
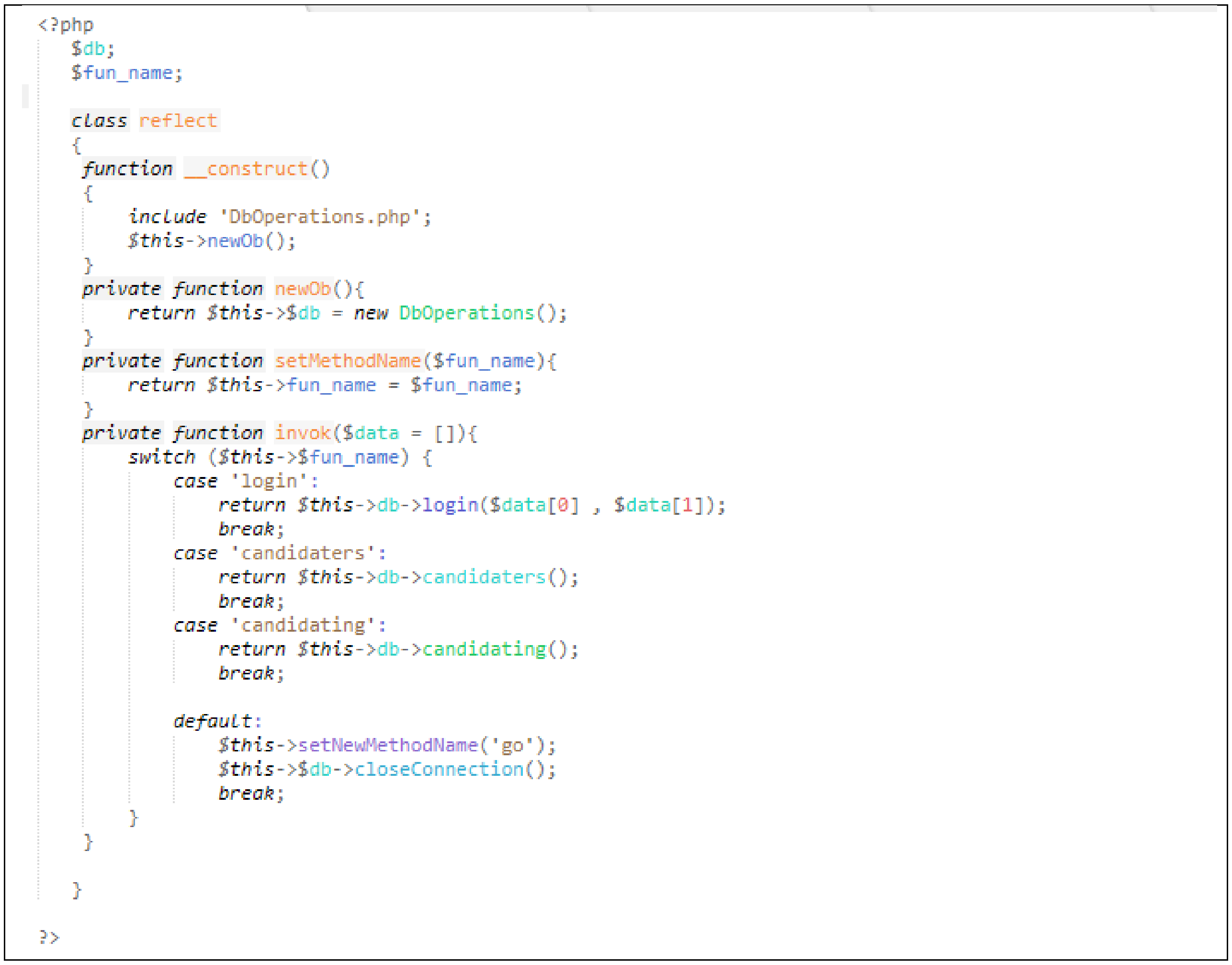


Figure 4.19 Reflection class

In the table (4.22) as below shows the reflection class code.

Table 4.22Reflection Code



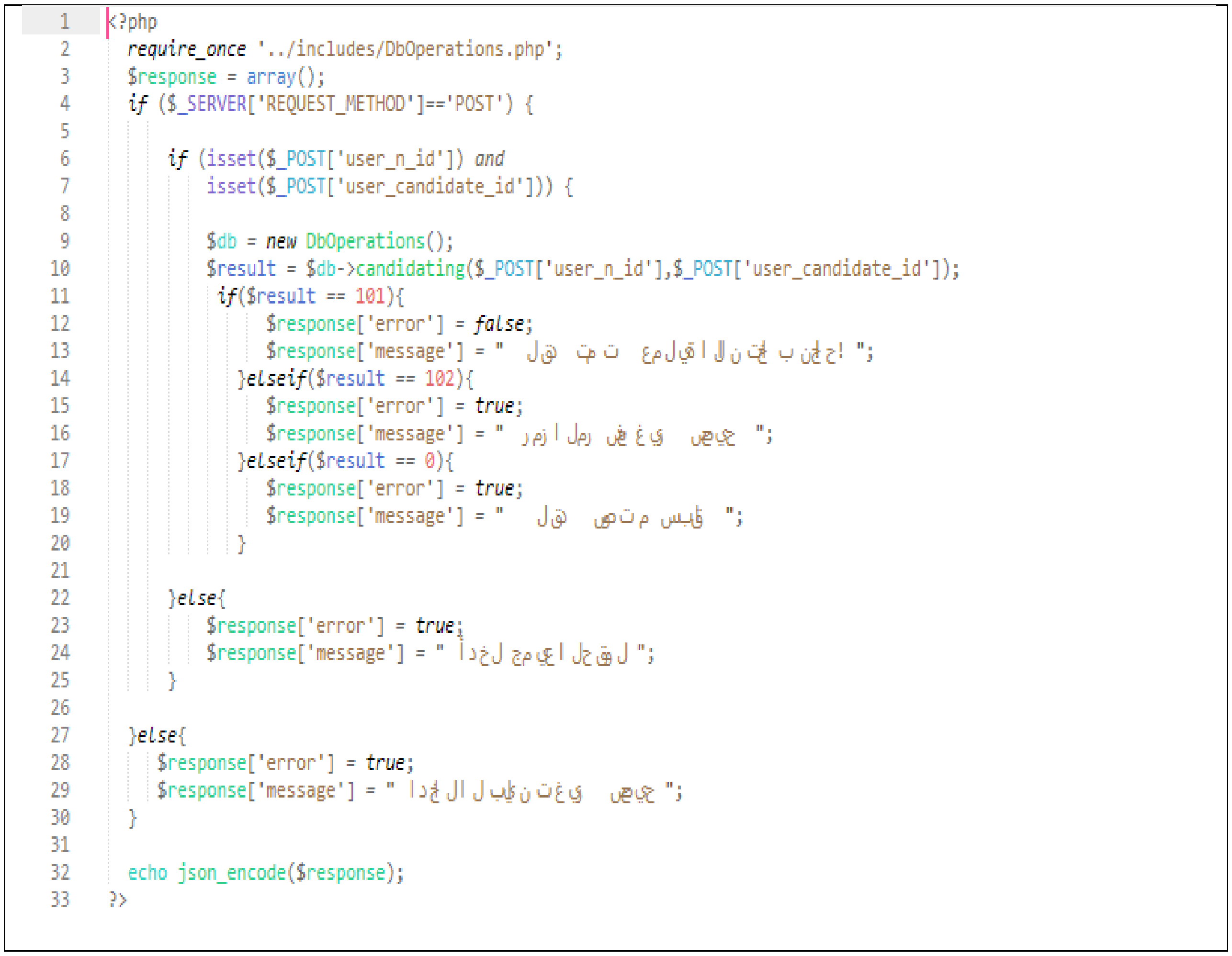
```
<?php
   $db;
   $fun_name;

   class reflect
   {
    function __construct()
    {
        include 'DbOperations.php';
        $this->newOb();
    }
    private function newOb(){
        return $this->$db = new DbOperations();
    }
    private function setMethodName($fun_name){
        return $this->fun_name = $fun_name;
    }
    private function invok($data = []){
        switch ($this->$fun_name) {
            case 'login':
                return $this->db->login($data[0] , $data[1]);
                break;
            case 'candidaters':
                return $this->db->candidaters();
                break;
            case 'candidating':
                return $this->db->candidating();
                break;

            default:
                $this->setNewMethodName('go');
                $this->$db->closeConnection();
                break;
        }
    }

   }

?>
```

4.8.3. PHP Operation Files (JOSON encode)

Each process in the application has a file in the server, as mentioned previously, it sends data in the form of a JSON array that is dealt with in the application by hash map String, as is the case with the voting file, where it receives the national number and the candidate's code and completes the voting process and explains the stages with instant messages sent through the JSON array

These voting files invoke all the functions it needs to ensure no repeat vote for the same number or imaginary vote. The following table (4.23) shows the server voting file

Table 4.23 CandidatingFile



```php
<?php
  require_once '../includes/DbOperations.php';
  $response = array();
  if ($_SERVER['REQUEST_METHOD']=='POST') {

      if (isset($_POST['user_n_id']) and
          isset($_POST['user_candidate_id'])) {

          $db = new DbOperations();
          $result = $db->candidating($_POST['user_n_id'],$_POST['user_candidate_id']);
           if($result == 101){
                $response['error'] = false;
                $response['message'] = " لقد تمت عملية الانتخاب بنجاح! ";
             }elseif($result == 102){
                $response['error'] = true;
                $response['message'] = " الرمز غير صحيح ";
             }elseif($result == 0){
                $response['error'] = true;
                $response['message'] = " لقد صوت مسبقا ";
             }

      }else{
          $response['error'] = true;
          $response['message'] = " ادخل جميع الحقول ";
      }

  }else{
     $response['error'] = true;
     $response['message'] = " ادخال الطلب غير صحيح ";
  }

  echo json_encode($response);
?>
```

4.9. Discussion:

Reflection design is optimal with a mobile application to implement security at the application level, especially in electronic voting, which requires a high degree of integrity and transparency during the voting process due to the sensitivity of the data used to ensure the correct results are obtained.

4.9.1. E-voting:

This application uses the concept of reflection starting from running operations on the application so that the procedures in the application have been taken from the end user, validated and approved or not on the server where the data is synchronized with the civil registry data to ensure the validity of the vote.

In this way, the operations are divided into two parts, the first part is at the application level and does not contain any sensitive information other than keeping the national vote number, which is the result of a valid login process concurrent with the civil registry data, which is after the registration process, which means that it must be registered In the email voting program first via the application, this ensures that the server is not over right with unauthorized data, and therefore we guarantee that there is no re-transfer of data back to the server (restart after transmission) and that there is no concern about SQL injection .As for the issue of sending and receiving data from the server to the server, the Super Global Post guarantees the confidentiality of data in the web or HTTPs request, an application that is usually in a secure format, especially after addition character "S", which means that it is secure from beside re-send data request.

The reflection helped to protect against data capture during its transmission to the server due to the inability to change data except by logging out of the application and trying to login again, which required valid data again.

4.10. Summery

The success of this work will allow the developing of the security of mobile applications in the Android system; alsothis work allows application developers to use the reflection concept in securing applications as it provides protection for the server through the application itself.

CHAPTER V

CONCLUSION AND RECOMMENDATION

5.1. Conclusion

The purpose of using reflection is to make the application protect its data without interference from any external factor. Concludefrom that it’s protecting our application's data from detection and theft by the attackers with using reflection design. As a result of using reflection in the application, data security was achieved and some types of bug’s problem were solved.

1. There are certain ways to call functions via the Reflection Class.
2. The connection to the database will be closed for the current session, which requires opening the application again and making a connection to the database again, and thus we have implied that in every attempt to change the function name or reuse it in inappropriate location, the session is completely canceled, all these operations are done within the application Or rather, inside the server, the application reconnects again when the application is opened normally.
3. The application is also protected from data retransmissions.

5.2. Contributions

As a result of using reflection in the E-Vote application contributed on:

1. Increase self-security in mobile applications.
2. As a result of using reflection in the application has achieved results to provide data security
3. Combine protection mechanisms within the mobile system which facilitate self-protection for applications.
4. Protect server from overload which happen form attacker more over request (Reply message) replay after send.
5. Create method use to prevent grant robot permission.
6. Solve some types of bug problems that occur to applications.

5.3. Recommendations

From the results of this study, it’s recommended that:

1. Farther study and researches must be done in self-protect error injection in mobile application using reflection.
2. Use the reflection to make certain self-protect to decrease some type of bugs**.**

References

1. HE, D., CHAN, S. & GUIZANI, M. 2015. Mobile application security: malware threats and defenses. *IEEE Wireless Communications,* 22**,** 138-144.
2. AMETLLER, J., ROBLES, S. & ORTEGA-RUIZ, J. A. An implementation of self-protected mobile agents. Proceedings. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004., 2004a. IEEE, 544-549.
3. DAVI, L., DMITRIENKO, A., SADEGHI, A.-R. & WINANDY, M. Privilege escalation attacks on android. international conference on Information security, 2010. Springer, 346-360.
4. LANE, N. D., MILUZZO, E., LU, H., PEEBLES, D., CHOUDHURY, T. & CAMPBELL, A. T. 2010. A survey of mobile phone sensing. *IEEE Communications magazine,* 48**,** 140-150.
5. JIN, X., HU, X., YING, K., DU, W., YIN, H. & PERI, G. N. Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014. 66-77.
6. ANDERSON, J., BONNEAU, J. & STAJANO, F. Inglorious Installers: Security in the Application Marketplace. WEIS, 2010. Citeseer.
7. BURNS, J. 2008. Developing secure mobile applications for android.
8. LA POLLA, M., MARTINELLI, F. & SGANDURRA, D. 2012. A survey on security for mobile devices. *IEEE communications surveys & tutorials,* 15**,** 446-471.
9. SUI, A.-F. & GUO, T. A behavior analysis based mobile malware defense system. 2012 6th International Conference on Signal Processing and Communication Systems, 2012. IEEE, 1-6.
10. HOLLA, S. & KATTI, M. M. 2012. Android based mobile application development and its security. *International Journal of Computer Trends and Technology,* 3**,** 486-490.
11. LIU, Y., XU, C. & CHEUNG, S.-C. Characterizing and detecting performance bugs for smartphone applications. Proceedings of the 36th international conference on software engineering, 2014. 1013-1024.
12. ARGERSINGER, P. H. 1985. New perspectives on election fraud in the gilded age. *Political Science Quarterly,* 100**,** 669-687.
13. PATHAK, A., HU, Y. C. & ZHANG, M. Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices. Proceedings of the 10th ACM Workshop on Hot Topics in Networks, 2011. 1-6.
14. SCHAUPP, L. C. & CARTER, L. 2005. E-voting: from apathy to adoption. *Journal of Enterprise Information Management*.
15. LEHOUCQ, F. 2003. Electoral fraud: Causes, types, and consequences. *Annual review of political science,* 6**,** 233-256.
16. LANKINA, T. & SKOVORODA, R. 2017. Regional protest and electoral fraud: evidence from analysis of new data on Russian protest. *East European Politics,* 33**,** 253-274.
17. FELT, A. P., CHIN, E., HANNA, S., SONG, D. & WAGNER, D. Android permissions demystified. Proceedings of the 18th ACM conference on Computer and communications security, 2011. 627-638.
18. LU, L., LI, Z., WU, Z., LEE, W. & JIANG, G. Chex: statically vetting android apps for component hijacking vulnerabilities. Proceedings of the 2012 ACM conference on Computer and communications security, 2012. 229-240.
19. BUCERZAN, D., RATIU, C. & MANOLESCU, M.-J. 2013. SmartSteg: A New Android Based Steganography Application. *Int. J. Comput. Commun. Control,* 8**,** 681-688.
20. BABIL, G. S., MEHANI, O., BORELI, R. & KAAFAR, M.-A. On the effectiveness of dynamic taint analysis for protecting against private information leaks on Android-based devices. 2013 International Conference on Security and Cryptography (SECRYPT), 2013. IEEE, 1-8.

21. SARWAR, G., MEHANI, O., BORELI, R. & KAAFAR, M. A. On the Effectiveness of Dynamic Taint Analysis for Protecting against Private Information Leaks on Android-based Devices. SECRYPT, 2013.
22. PENG, H., GATES, C., SARMA, B., LI, N., QI, Y., POTHARAJU, R., NITA-ROTARU, C. & MOLLOY, I. Using probabilistic generative models for ranking risks of android apps. Proceedings of the 2012 ACM conference on Computer and communications security, 2012. 241-252.
23. BLÄSING, T., BATYUK, L., SCHMIDT, A.-D., CAMTEPE, S. A. & ALBAYRAK, S. An android application sandbox system for suspicious software detection. 2010 5th International Conference on Malicious and Unwanted Software, 2010. IEEE, 55-62.
24. KANEI, F., TAKATA, Y., AKIYAMA, M., YAGI, T. & YADA, T. 2017. Poster: Protecting Android Apps from Repackaging by Self-Protection Code.
25. AMETLLER, J., ROBLES, S. & ORTEGA-RUIZ, J. A. Self-protected mobile agents. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1, 2004b. 362-367.
26. DAWSON, D. 2009. *Facilitating autonomic computing using reflection.*
27. MALEK, S., BAGHERI, H., GARCIA, J. & SADEGHI, A. 2019. Security and software engineering. *Handbook of Software Engineering.* Springer.
28. PONTES, F., GHEYI, R., SOUTO, S., GARCIA, A. & RIBEIRO, M. Java reflection API: revealing the dark side of the mirror. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019. 636-646.
29. LI, Y., TAN, T. & XUE, J. 2019. Understanding and analyzing java reflection. *ACM Transactions on Software Engineering and Methodology (TOSEM),* 28**,** 1-50.
30. MALABARBA, S., PANDEY, R., GRAGG, J., BARR, E. & BARNES, J. F. Runtime support for type-safe dynamic Java classes. European Conference on Object-Oriented Programming, 2000. Springer, 337-361.