KINGDOM OF SAUDI ARABIA
Ministry of Education
Taibah University
Applied college
Cyber Security program

المملكة العربية السعودية
وزارة التعليم
الكلية التطبيقية
(برنامج الأمن السيبراني)

Secure Authentication System based on Multi-Factor Authentication

Graduation Project

By

Rayan Abdulrahim Al-Sahli 4300147

Abdulrahman Aqeel Al-Mutairi 4301286

A project submitted in partial fulfilment of the requirements for the degree of diploma in Cyber Security.

Supervised by

Dr. Khalid Nasr

First Semester-Academic Year 1445 (2023/2024)

Acknowledgement

We are honored to express my sincere gratitude and appreciation to Taibah University and Dr. Khalid Nasr for their invaluable support, guidance, and assistance in the successful completion of our graduation project. Their contributions have played a vital role in our achievement, and we are truly grateful for their unwavering environment and the necessary resources to pursue our studies. The university's dedication to academic excellence and commitment to empowering students have been instrumental in shaping our educational journey. We are grateful for the opportunities and privileges we have received as students of Taibah University.

Abstract

Authentication is a critical process for establishing trust and verifying the identities of entities involved in a connection. While passwords are commonly used for authentication, they are susceptible to cracking and other forms of attack. To enhance security, it is important to introduce complexity into the authentication process. Multi-factor authentication (MFA) is an effective approach that involves using multiple factors for authentication. MFA typically consists of three authentication factors, the knowledge factor, ownership factor, and biometric factor. The knowledge factor involves something the user knows, such as a password, PIN, or answers to security questions. It serves as the first line of defence. The Ownership factor relies on something the user possesses, like a physical token, smart card, or mobile device. It generates a One-Time Password (OTP) or receives push notifications for authentication. The biometric factor is based on something inherent to the user, such as biometric identifiers like fingerprints, retina scans, or voice recognition.

Our project concerned with implementing a two-factor authentication system that considers the OTP (One Time Password) as an additional factor for authentication, after the password comparison, to enhance the security and address the vulnerabilities associated with relying solely on passwords. From the previous work, we decided to focus on implementing Time-based One-Time Passwords (TOTP) which is an authentication method in which unique codes are generated every 30 to 60 seconds based on an algorithm. TOTP is the most secure and commonly used mechanism. OTP adds an extra layer of protection by generating unique OTPs that are valid only for a short period of time. This makes it more difficult for attackers to intercept and misuse the OTP.

Our proposed system consists of three main modules; the first module is the sign up module where the user registers his main information and password regarding predefined conditions. The second module is the login module which is used for password comparison. The third module is responsible for generating the TOTP. There is an additional module for generating QR code for synchronizing the generated OTP between the application server and authenticator (on a mobile device).

After successful registration (sign up), the QR code is generated regarding the generated OTP in our system. The user can use an authenticator application on his mobile device, such as Google Authenticator, for scanning the QR code and obtaining the synchronized OTP code. For the authentication process, if the user enters the valid password for login, the system will forward him to the OTP verification page to enter the valid OTP code displayed on the authenticator application. The results demonstrate the effectiveness of our proposed system.

Keywords: password, OTP, QR code, multi-factor authentication.

Table of Contents

Acknowledgement ii

Abstract..... iii

Table of contents……..……iv

List of Figures.....vi

List of Tables..... ix

List of Abbreviations. x

1 Chapter 1: Introduction..... 1

1.1 Introduction 1

1.2 Problem Definition 1

1.3 Project Objectives..... 2

1.4 Project Scope 2

1.5 Project Timeline 2

1.6 Document Organization..... 3

2 Chapter 2: Literature Review..... 4

2.1 Introduction 4

2.2 Background..... 4

2.3 Related work. 5

2.4 Summary..... 15

3 Chapter 3: System Analysis..... 16

3.1 Introduction 16

3.2 Requirements..... 16

3.2.1 Functional Requirements. 16

3.2.2 Non-Functional Requirements..... 16

3.3 System Analysis 17

3.3.1 Use Cases Diagram. 17

3.3.2 Flow Chart. 22

3.4 Developmental Methodology 22

3.5 Summary. 24

4 Chapter 4: System Design 25

4.1 Introduction 25
4.2 Architectural design 25
4.3 Object Oriented Design 26
4.3.1 Class Diagram 26
4.3.2 Sequence Diagram 26
4.3.3 Activity Diagram 27
5 Chapter 5: System Implementation 28
5.1 Introduction 28
5.2 Tools and Languages. 28
5.3 Main and Most Important Codes 29
5.3.1 Code for signup 29
5.3.2 Database 32
5.3.3 Code for login 33
5.3.4 Code for OTP generation 34
5.3.4 Code for Generating and displaying the QR code 35
5.3.4 Code for verification 35
6 Chapter 6: Evaluation Testing 38
6.1 Introduction 38
6.2 System Testing. 38
6.2.1 Testing Strategy 38
6.2.2 Testing Items 38
6.2.3 Testing Approach 39
6.2.3.1 Unit Testing 39
6.2.3.2 Integration Testing 44
6.2.3.3 Performance Testing 44
6.3 Summary 57
7 Chapter 7: Conclusion and Future Work 58
7.1 Conclusion 58
7.2 Goals Achieved 58
8 References 59

List of Figures

Figure 2-1 Three steps of the OTP generation] ……..…….……6

Figure 2-2 Static extraction…….……..7

Figure 2-3 Dynamic extraction……..7

Figure 2-4 Improved dynamic extraction.……8

Figure 2-5 The framework for the proposed technique ……10

Figure 2-6 2FA Based on Blockchain Framework ……..……..11

Figure 2-7 Sequence Diagram of Login request and OTP generation Verification …….……13

Figure 2-8 OTP generators ……...……14

Figure 2-9 A hight-level of a web 2FA sequence ……...…….15

Figure 3-1 Use Case Diagram…….…….……..17

Figure 3-2 Flow chart …….……...……23

Figure 3-3 Agile model ……..……..…..……...……24

Figure 4-1 architecture design……...25

Figure 4-2 Class diagram …….……26

Figure 4-3 Sequence Diagram ……..……27

Figure 4-4 Active Diagram……27

Figure 5-1 code for Sign up ……..……30

Figure 5-2 code for Sign up page……..30

Figure 5-3 Sign up page……31

Figure 5-4 Register in the database……...……31

Figure 5-5 MySQL database……..32

Figure 5-6 code for login…..…….……33

Figure 5-7 login page……...…….33

Figure 5-8 code for generate OTP……34

Figure 5-9 for generate OTP……..35

Figure 5-10 code for generate QR code……..……..35

Figure 5-11 code for display QR code……..……36

Figure 5-21 Displayed QR code……..36

Figure 5-13 code for Verification page……...……..……37

Figure 5-14 Verification page……..…….37

Figure 6-1 Testing Strategy……..…….39

Figure 6-2 Login page……...……..……..40

Figure 6-3 Sign up page……...……..41

Figure 6-4 QR code Generation for scanning ….……...…….……41

Figure 6-5 OTP code on google Authenticator…….……42

Figure 6-6 login…….42

Figure 6-7 OTP verification ……...…….43

Figure 6-8 Welcome page…….44

Figure 6-9 QR code generated after Sign Up……45

Figure 6-10 Login……..45

Figure 6-11 OTP Verification regarding Google Authenticator …….…….46

Figure 6-12 OTP on Google Authenticator……...…….46

Figure 6-13 Verification page…….……..47

Figure 6-14 Test Case Number 1……..…….……...……50

Figure 6-15 Test Case Number 2……..…….…….……..51

Figure 6-16 Test Case Number 3……..……51

Figure 6-17 Test Case Number 4……..……52

Figure 6-18 Test Case Number 5……..……52

Figure 6-19 Test Case Number 6……..……53

Figure 6-20 Test Case Number 7……..……...…….53

Figure 6-21 Test Case Number 8……..……...……..54

Figure 6-22 Test Case Number 9……..…….……...…54

Figure 6-23 Test Case Number 10……..……..…….55

Figure 6-24 Test Case Number 11……..…….…….55

Figure 6-25 Test Case Number 12……..…….…….56

Figure 6-26 Test Case Number 13……..…….…….56

Figure 6-27 Test Case Number 14……..…….…….57

List of Tables

Table 3-1 Sign up……..……..……18

Table 3-2 Generate OTP..……...……..……18

Table 3-3 Generate QR code……..…19

Table 3-4 Login…….……...……..…….19

Table 3-5 Verify username and password…….……20

Table 3-6 display login error…….……20

Table 3-7 access using OTP……...……...……21

Table 3-8 Verify OTP……21

Table 3-9 display incorrect OTP……...22

Table 5-1 Technology summary……..……..28

Table 5-2 External Libraries…….……29

Table 6-1 Run Program……..……...40

Table 6-2 Sign-Up…….40

Table 6-3 Scan QR code……41

Table 6-4 Login……...……..42

Table 6-5 OTP Verification…….……..43

Table 6-6 Test cases ……...……..47

List of Abbreviations

One Time Password (OTP)

Time-Based One Time Password (TOTP)

Multi-factor authentication (MFA)

man in the middle (MITM)

Personal Identification Number (PIN)

Telecommunication Technology Association (TTA)

Fast Identity Online (FIDO)

Mobile Station Equipment Identity (IMEI)

Graphical User Interface (GUI).

Chapter 1: Introduction

1.1 Introduction

Authentication is a technique and technology employed to establish a connection between two entities. This connection relies on trust and certainty that the involved parties are indeed the authentic ones in forging the association. The widely employed method of authentication revolves around the utilization of passwords, which are employed across a range of services including social networks, bank accounts, websites, and various applications. However, passwords can be susceptible to cracking through various means, and simpler methods can be targeted without the need for extensive computations. The most effective approach to mitigate password attacks is to augment security measures and introduce complexity into the authentication process. multi-factor authentication (MFA) plays a vital role in this context.

1.2 Problem Definition

One problem addressed by multi-factor authentication is the inherent vulnerability of relying solely on passwords for user authentication. Passwords can be easily stolen, guessed, or cracked, putting sensitive information and systems at risk. Attackers can employ various methods, such as phishing, keylogging, or social engineering, to obtain passwords and gain unauthorized access to accounts or systems. This problem can be solved using various authentication mechanisms in layering configuration which is typically called multi-factor authentication (MFA).

Authentication mechanisms use any of three qualities to confirm a user's identity [PPM15]:

– Something the user knows. This is something the user knows, such as a password, PIN, or answers to security questions. It is the most common form of authentication and is used as the first line of defense.

– Something the user has. This is something the user possesses, such as a physical token, smartcard, or mobile device. It generates a One-Time Password (OTP) or receives a push notification for authentication purposes.

– Something the user is. These authenticators, called biometrics, are based on a physical characteristic of the user, such as a fingerprint, face recognition, or retina scan, or voice recognition. Biometric authentication methods are becoming increasingly

popular as they provide a unique and personal identifier.

Two or more mechanisms can be combined to ensure strong security against unauthorized access attempts. For example, a bank card and a PIN combine something the user has (the card) with something the user knows (the PIN). The most common form of strong authentication mechanisms is using the password in conjunction with OTP, and thus our project concerned with this combination.

1.3 Project Objectives

- Enhance Security
- Prevent Unauthorized Access
- Mitigate Password-related Risks.
- Safeguard Sensitive Transactions
- Comply with Regulatory and Compliance Standard
- Improve User Experience.

1.4 Project Scope

This project aims to create a user interface that contains registration and login. After successful registration, an OTP will be generated by displaying it in the QR code, which will be scanned using Google authenticator.

By implementing OTPs, users can significantly enhance the security of their accounts and reduce the risk of unauthorized access, even if their passwords are compromised or intercepted by attackers.

1.5 Project Timeline

Month	August	September				October				November				December	
Week	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Discussion and selecting the topic															
Chapter1 (introduction)															
Chapter4 (Literature Review)															
Chapter3 (systemanalysis)															
Chapter 4 (SystemDesign)															
Chapter 5 (implementation)															

Chapter 6 (Evaluating Testing)															
Chapter 7 (conclusion and future work)															
Reviewing the report and Preparing presentation															
Meeting and dscussion															

1.6 Document Organization

Chapter 1: This chapter introduces the problematic and motivation of our work. It presents an overview of authentication mechanisms and the problem associated with sing a single factor authentication specially the password that may be easily compromised or stolen, and also our project objectives for developing a proactive countermeasure, using two factor authentication that involves the password and OTP, to combat any unauthorized access attempts.

Chapter 2: This chapter contextualizes the project by reviewing existing works relevant to the topic and discusses related research studies and their results, highlighting similarities, differences, and gaps in the literature. It provides background information on the materials and subject matter of the project.

Chapter 3: This chapter elicits requirements for the proposed system, including functional requirements (desired system behavior) and non-functional requirements (constraints and qualities of the system). Our proposed system is summarized and presented by the use-case diagram.

Chapter 4: This chapter presents the architectural design of our proposed system, including the overall structure and organization of the system. It also discusses object-oriented design, including static and dynamic models, including data modeling and user interface design.

Chapter 5: This chapter clarifies in detail the steps of our proposed system implementation and verification. It describes the tools and programming languages used in the development process, and maps the design specifications to the actual implementation.

Chapter 6: In this chapter, experimental evaluation tests were conducted to prove the concept. It presents the results and discusses them regarding the project objectives. The results proved the effectiveness of our proposed and implemented system.

Chapter 7: this chapter presents our conclusion and future work.

Chapter 2: Literature Review

2.1 Introduction

The literature review centers on two main facets: the system perspective and the methodological standpoint. Regarding the system aspect, it delves into the presently utilized tools, highlighting their limitations. On the methodological front, it explores alternative approaches proposed by various researchers.

2.2 Background

Authentication is crucial when individuals access systems to confirm their identity and safeguard their private data [GMA21]. Consequently, various authentication techniques, incorporating elements such as knowledge, ownership, and biometrics, have been proposed.

Authentication based on knowledge allows access to a device through a single authentication process [MZ22]. In the realm of internet banking systems, the predominant method for user authentication and access control relies on knowledge due to its attributes and familiarity to both service providers and users. Nonetheless, knowledge-based authentication is susceptible to various types of attacks, including brute-force attacks, rainbow table attacks, dictionary attacks, social engineering, phishing, frameworks for man-in-the-middle attacks (MITMF), password-based attacks, session hijacking, and malware. Moreover, relying solely on password-based authentication has been acknowledged as insufficient for providing adequate security, given the multitude of security threats. User login assessments have revealed that a substantial 86% of user-selected passwords are insecure. To address the shortcomings associated with knowledge-based authentication, ownership-based authentication emerged as a viable solution for enhancing the security, authentication, and verification of online transactions for both devices and applications.

Ownership-based authentication, which often involves the use of security hardware tokens, is frequently employed as a means of two-factor authentication [PB19]. However, it can also serve as the primary method of authentication. There are compelling reasons not to rely solely on ownership-based authentication in this role. One of these reasons is that even a weak password, when combined with ownership- based authentication, still provides some level of security by increasing the difficulty for potential attackers. Additionally, if the security hardware token lacks protection in the form of a PIN or an equivalent security measure, it becomes vulnerable to physical theft, enabling an attacker to easily impersonate users who are in close physical proximity by

stealing and utilizing the token.

Biometrics rely on the measurement of distinct physiological or behavioral attributes, such as fingerprints or voice patterns, for authentication purposes. Among these, fingerprints stand out as the most used biometric for authentication [PB19]. The primary advantage of biometrics lies in their ability to genuinely authenticate a human user, as they are inherent and readily available. However, they also present several challenges. One key challenge is that measurements of the same biometric trait are never identical, and the result of a comparison is not a simple binary yes/no decision. Consequently, there is a need for an acceptance (or rejection) threshold, which introduces the possibility of false positives and false negatives. Biometrics may not be accessible to everyone due to permanent or temporary absence or damage to body parts, and environmental factors like weather can lead to false rejections. Moreover, biometric data often lacks the necessary entropy for cryptographic purposes, and it is generally public knowledge, as people leave their fingerprints on various surfaces. This makes it relatively easy and inexpensive to create fake biometric samples, and liveliness detection mechanisms can often be bypassed. To address many of these challenges, a common approach involves using biometrics locally to protect a robust cryptographic key, effectively combining the strengths of both methods while mitigating their respective weaknesses. However, it's important to note that there is a finite number of unique "passwords" that can be derived from limited body parts (e.g., fingers), and these biometric credentials cannot be easily revoked once compromised.

2.3 Related work

Kim et al. [HJCO20] have categorized three algorithms for extracting the ultimate data from a pseudo-random bit sequence. They have identified a vulnerability during this extraction process, leading to a heightened occurrence of specific numbers being extracted, as outlined in their research.

In the employed process, generating a One-Time Password (OTP) involves three key steps: Association information generation, generation algorithm, and extraction algorithm. Remarkably, this process closely resembles the procedure used for generating pseudo-random numbers in cryptography [HJCO20]. Figure 2-1 shows similar parts of the two processes in the same color.

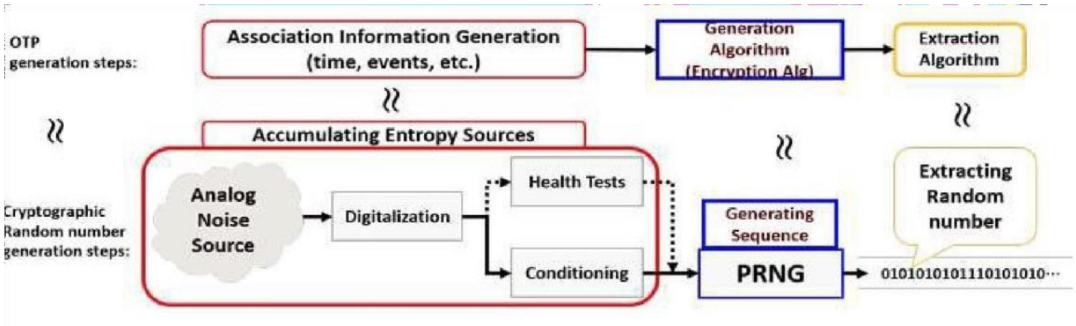


Figure 2-1 Three steps of the OTP generation

Association Information Generation is the process of creating data that serves as a source of randomness. This data can be obtained from various sources, including time, event information, and similar inputs. It may exist in its raw form, depending on how it's collected, or it may undergo additional processing to become a random number through methods like hashing and encryption. In essence, "association information" can be likened to a noise source or an entropy source, which is essential for generating cryptographic pseudorandom numbers. It plays a crucial role in ensuring the unpredictability and security of One-Time Password (OTP) systems. This process corresponds to the red part of Figure 2-1.

The "Generation Algorithm" is a cryptographic process employed to create a Onetime Password (OTP). It operates by encrypting the association information, resulting in the production of a 20-byte bit string. The Korean OTP standard specifies the cryptographic algorithm utilized in this step. Because the generation algorithm is deterministic, it consistently produces the same ciphertext when given the same association information. It's essential for the encryption algorithm employed in the generation algorithm to offer a security strength of at least 112 bits to ensure robust security. It's worth noting that different OTP values can be generated from the same bit string, depending on the extraction algorithms applied. This process corresponds to the blue part of Figure 2-1.

This is an algorithm for extracting 3-byte data from the cipher string for use as OTP. This can be divided into static and dynamic algorithms (RFC4226 , an OTP-related standard, specifies only dynamic algorithms, and the Telecommunication Technology Association (TTA)’s standard specifies static, dynamic, and improved dynamic algorithms.). This algorithm uses ‘extraction information’ to specify the location to extract. It also uses the value as the ‘extraction data’ at the location specified by the extraction information. The extraction information may use a value of a specific area of the cipher text or a predefined value. Then, the extracted 3-byte hexadecimal value is

encoded in decimal so that humans can recognize it. At this time, the extracted 3-byte range is [0, 16777215] (i.e., [0, 2^24 - 1]), but the top two digits have little change. Therefore, the two digits are discarded and only the lower six digits are used (Only 0 ~ 1 for 10^7 , only 0 ~ 6 for 10^6) ; so the final range is [0, 999999]. This process corresponds to the yellow part of Figure 2-1. The static algorithm extracts data by specifying in advance the extraction information (the offset of the cipher text) to be used. An example is shown in Figure 2-2: Data is extracted by specifying the extraction information as the first starting point of the cipher text. It extracts 3-byte contiguous data from the extraction information. If the result of the extracted data is 0x1A2B3C, the number is 1,715,004 when expressed as a decimal number, and the final output OTP is derived as 715,004 corresponding to the lower 6 digits of the decimal number.

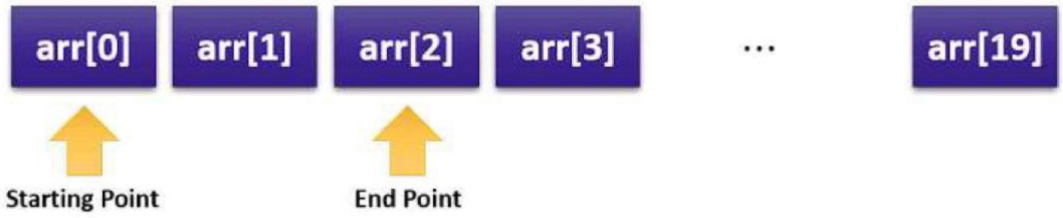


Figure 2-2 Static extraction

The dynamic algorithm extracts data by obtaining extraction information from the cipher text. About this example is shown in Figure 2-3: The value of the lower 4-bit of the lowest byte (i.e., 19th) in the cipher text is used as extraction information. The extracted data is contiguous 3-byte with an index of the extraction information as a starting point. If the result of the extracted data is0xE7F809, that's decimal is 15,202,313, the final output OTP is derived as 202,313 corresponding to the lower 6 digits.

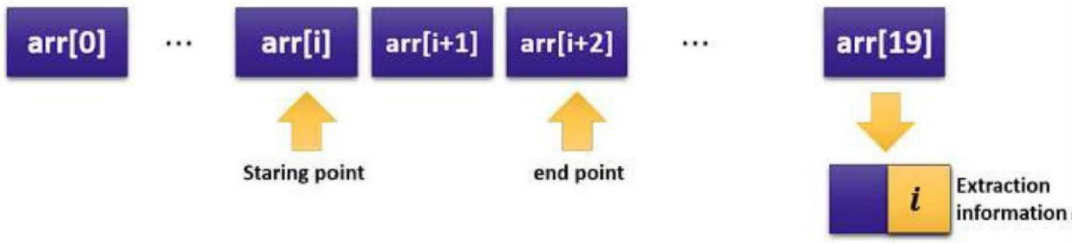


Figure 2-3 Dynamic extraction

He improved dynamic algorithm is used when generating OTP using a chip used as a smart card or USIM. Figure 2-4 shows an improved dynamic algorithm; it extracts three bytes fromtwo or more pieces of extraction information in the cipher text.

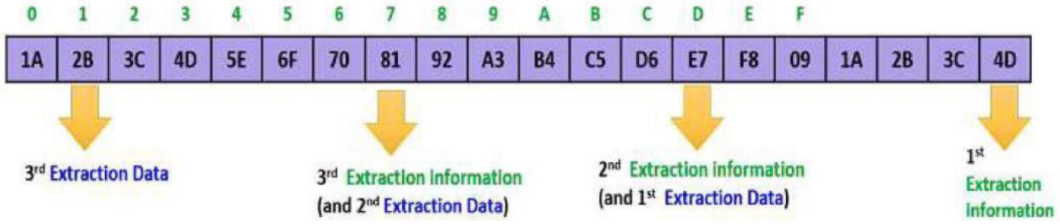


Figure 2-4 Improved dynamic extraction

Extraction information of this algorithm is used to designate specific areas of cipher text. If the value of the lower 4-bit of the lowest byte is used as the first extraction information (namely, LSB 4-bit0x0D, the lowest byte 0x4D of the cipher text), then the data 0xE7 at that location (0x0D) is used as the first extraction data. And the second extraction information is 0x07, the value of the lower 4-bit of the first extraction data. The same method is repeated to select 3-byte extracted data. Final OTP is 171,883which is the lower 6 digits of 15,171,883 that converts the extracted data 0xE7812B to decimal.

G. Ali et al. [GMA21] introduced a robust and efficient multi-factor authentication algorithm designed specifically for FinTech applications. Their approach combines three factors: a Personal Identification Number (PIN), a one-time password (OTP), and biometric fingerprint recognition. This combination enhances the security of FinTech authentication. Additionally, they employ a biometric fingerprint along with quick response (QR) codes to verify mobile money withdrawals. The security of the PIN and OTP components is ensured through the use of the secure hashing algorithm SHA-256. For biometric fingerprint recognition, they utilize the Fast Identity Online (FIDO) framework, incorporating the RSA standard for public key cryptography. Furthermore, Fernet encryption is applied to secure QR codes and database records. The research includes the development of applications that were rigorously tested, and a comprehensive security analysis was conducted. The results demonstrate that the proposed algorithm is not only secure but also efficient and highly effective against various threat models. It provides a secure and efficient authentication process while ensuring data confidentiality, integrity, nonrepudiation, user anonymity, and privacy. The performance analysis indicates that their algorithm achieves superior overall performance when compared to existing FinTech systems.

In their work, M. Arif et al. [MZ22] introduced a design for wallet applications that includes an attached security element. This design primarily focuses on device identity as a fundamental component of the user authentication mechanism, encompassing four

essential categories of authentication: password-based authentication, one-time password (OTP) verification, fingerprint recognition, and international mobile device authentication. A noteworthy aspect of this design is the utilization of the International Mobile Station Equipment Identity (IMEI) of the user's device. This approach restricts the use of an e-wallet to one specific device at any given time, aligning it with the characteristics of a physical wallet. The proposed authentication design consists of two distinct stages: the registration stage and the authentication stage. They implemented this method using Android Studio, Firebase for real-time database management, and PayPal integration. To assess the effectiveness of their design, comprehensive evaluations were conducted through functional requirement testing. The results from these tests indicate that the proposed method successfully meets the functionality requirements. Additionally, the design ensures that a single account cannot be used on two different devices, enhancing security by preventing potential attacks. Their proposed method has demonstrated higher security standards compared to existing authentication methods.

The registration and authentication phases are the two steps in the authentication design. Before using Zamwallet, the user must enter their information through a process called registration. A technique known as step authentication is used to verify information. In this registration stage, the user must enter his credentials, such as password, fingerprint, and one-time password (OTP), into the mobile device, where the information will be transferred to the database. User information will be stored in the Firebase server through real-time databases. The user is required to enter their registered credential information, such as password, fingerprint and one time password (OTP), into the mobile device throughout this authentication phase, where the server will compare the registration and authentication value. The proposed conceptual design for authentication is shown in Figure 2-5

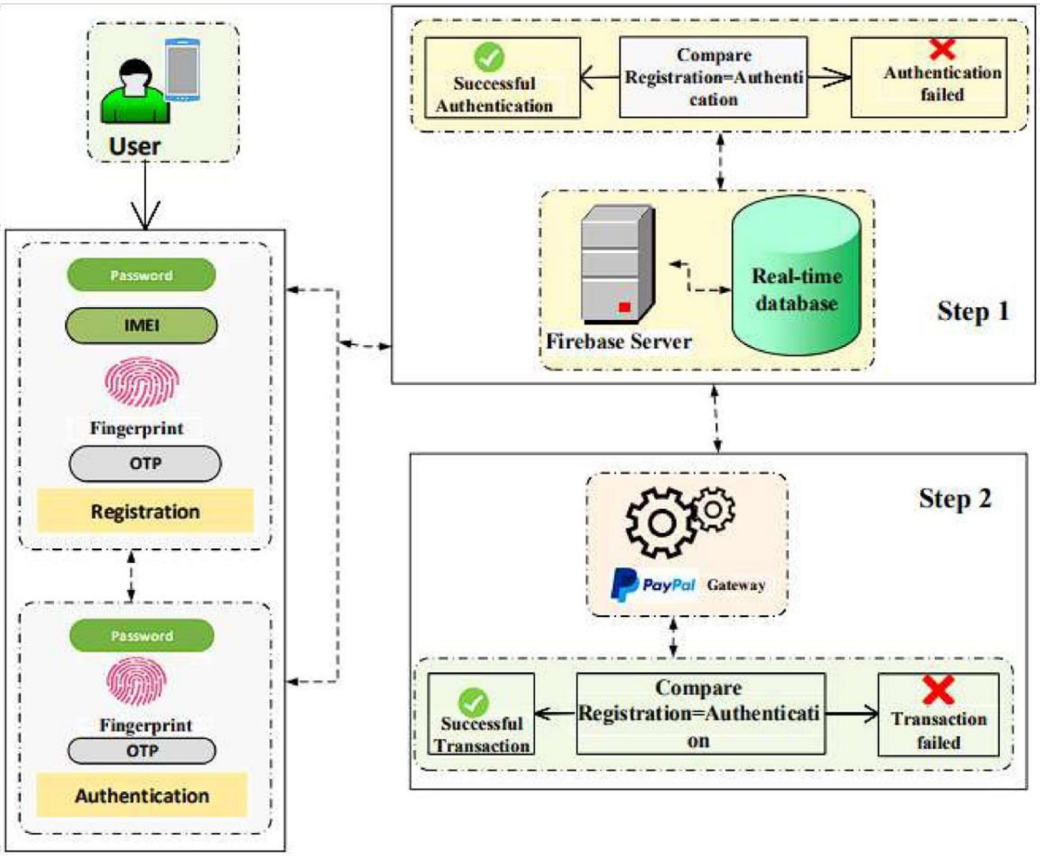


Figure 2-5 The framework for the proposed technique

The document titled "Two Factor Authentication Framework Using OTP-SMS Based on Blockchain" discusses a proposed framework for enhancing the security of two-factor authentication (2FA) using OTP-SMS (One Time Password sent via SMS) by incorporating blockchain technology. The authors highlight the vulnerabilities of traditional OTP-SMS methods and the need for a more secure approach. they proposed framework utilizes blockchain technology to generate and transmit encrypted, OTPs through smart contracts. This approach aims to prevent attacks such as Man in the Middle (MITM) and third- party attack. The authors compare the framework with other blockchain-based approaches for securing OTP-SMS and claim that their framework offers better security, lower computation time, and complexity.

The document also discusses the concept of 2FA and the different categories of authentication methods, including something you are (biometrics), something you know (passwords), and something you have (e.g., mobile phone). It highlights the

advantages of 2FA and the need to make the authentication process more complex to prevent passwordattacks.

The paper mentions various attacks that can compromise the 2FA process, such as MITM attacks, where an attacker intercepts the communication between a user and a website/application to steal personal information. It also references studies that have identified vulnerabilities in mobile 2FA schemes used by service providers like Dropbox, Google Authenticator, and Twitter. Overall, the document presents a proposed framework using blockchain technology to enhance the security of OTP-SMS-based2FA, addressing vulnerabilities and providing a more secure authentication process.

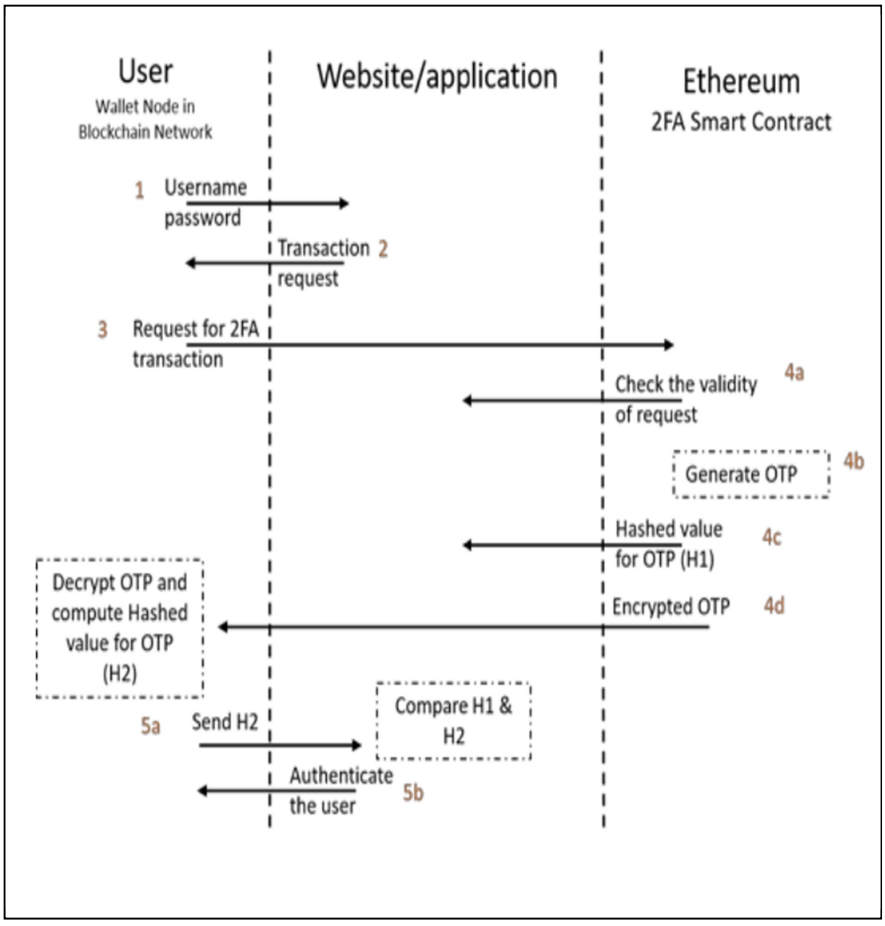


Figure 2-6 2FA Based on Blockchain Framework

The report titled "Two-Factor Authentication" is a technical report written by Jose Costafrom the University of Westminster. The report discusses the importance of two-factor authentication (TFA) as a security measure in various industries and presents

the author's aim to develop a secure TFA platform that can be easily integrated into existing infrastructures. the report begins with an introduction to TFA and its three categories of factors: knowledge factors (e.g., passwords), possessive factors (e.g., smart cards), and inherence factors (e.g., biometrics). It explains how TFA adds an extra layer of security by requiring users to provide additional authentication elements along with their passwords. the author states the purpose of the project, which is to provide a secure TFA platform that is cost-effective and easy to implement for businesses. They emphasize the importance of complying with data protection legislation and ensuring the security of users' personal data. the aims and objectives of the project are outlined, including the development of a lightweight and simple application, the provision of services through an API, and the creation of an iOS app for users to receive dynamic one-time passwords (OTPs).

The document identifies the stakeholders of the project, which include business owners who want to offer TFA to their users and the users themselves. It also acknowledges the presence of malicious users and the need to address potential threats.

Overall, the document provides an overview of the importance of two-factor authentication, the author's aims to develop a secure TFA platform, and a brief comparison with existing implementations.

The authors proposed a two-factor authentication protocol which consists of a biometric formulation known as Bio Hash. This combines a user-specific fingerprint Bi with a tokenized random number T, which in turn produces a set of n binary bit strings B = {b1, ..., bn}. This method makes it hard for an adversary ADV to get hold of the required authentication elements (B, T) to even make the final product needed for successful authentication.

Biometrics would not be a viable option to use in system due to the high implementation cost both on Their side and on the client's side.

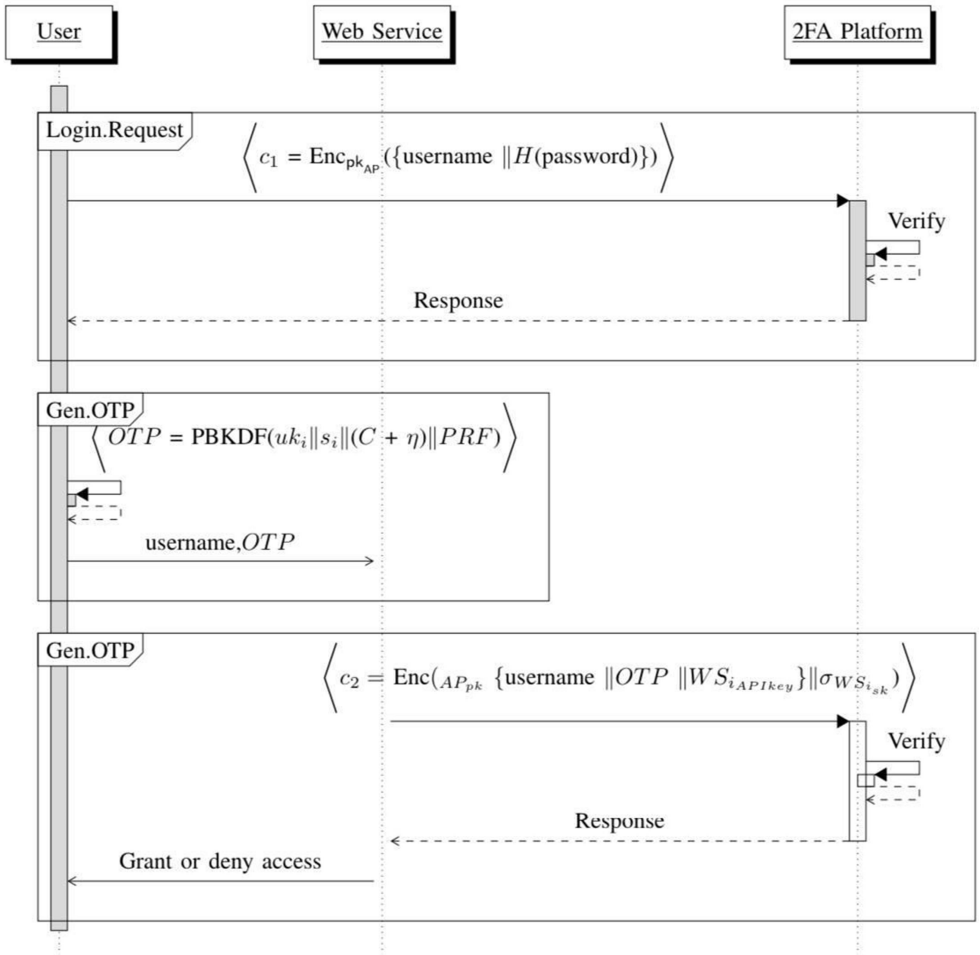


Figure 2-7 Sequence Diagram of Login request and OTP generation Verification

"Two-Factor Authentication Made Easy" is a conference paper from June 2015 authored by Alex Q. Chen and Weihan Goh. The paper addresses the challenge of authentication on the web and its impact on user experience, particularly for older and visually impaired users. It proposes a technique to simplify two-factor authentication (2FA) by using wearables as the authentication device and enabling seamless and automatic transfer of authentication information between the device and the web application. The authors conducted preliminary tests with older users and found that their approach was less stressful on users' memory and easier to use. The paper also discusses the weaknesses of existing online security techniques and the barriers faced by elderly and impaired users. It emphasizes the need for intuitive and accessible security systems for these user groups.

The document provides an overview of the authentication models, including what the user knows, possesses, and inherently is, and highlights the adoption of 2FA by web service providers. It introduces the concept of wearables as OTP generators and suggests automating the 2FA process through secure pairing between wearables and mobile devices. The authors conducted user evaluations to compare traditional 2FA methods (SMS-based OTP and token devices) with their proposed approach using a smartwatch as a wearable device. The evaluation involved participants accessing sensitive data on the web using a mobile phone. The document provides a brief description of the user evaluation setup but does not include the detailed results or conclusions of the study.

In all investigations, phantom mobile web applications simulate a 2FA-capable online banking site. Each of them has similar login screens to request a user identification and PIN. Once the entire authentication process is completed successfully, the participant is shown a menu screen displaying common online banking transactions. Their difference, however, is in the second-factor authentication: for the token and SMS investigations, participants are required to enter the OTP as shown on the token and simulated SMS, respectively, into the OTP forms, while for the watch investigation.

(a) OTP token device.

(b) Wearable OTP device.

Figure 2-8 OTP generators

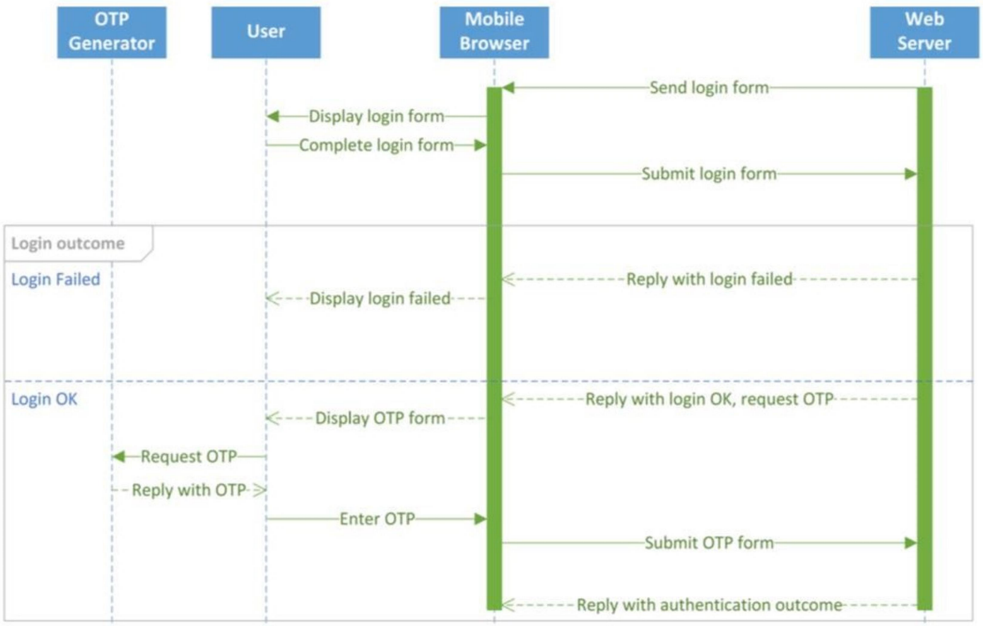


Figure 2-9 A hight-level of a web 2FA sequence

2.4 Summary

The literature review provides a comprehensive overview of authentication methods with a focus on knowledge-based and ownership-based authentication and biometrics. It discusses the importance of authentication in confirming individuals' identities and protecting their private data.

Chapter 3: System Analysis

3.1 Introduction

In this chapter, we conduct a thorough examination of the proposed system, covering a detailed assessment of its feasibility, an exploration of its functional and nonfunctional prerequisites, an overview of the overarching system structure, and an outline of the development approach to be employed for project completion.

3.2 Requirements

This section provides an overview of the project's deliverables' requirements, which can be classified into two distinct categories based on their nature: functional requirements and non-functional requirements.

3.2.1 Functional Requirements

- **User Registration**: Users should be able to create new accounts by providing essential information such as username, password, and email address.
- **Login**: Users should be able to log in using their registered credentials (e.g., username and password)
- **Password Policies**: Enforce password policies, including password complexity requirements (e.g., minimum length, special characters)
- **One-Time Password (OTP) Generation**: Implement the generation of OTPs, which are temporary codes, typically numeric, that users can use for authentication.
- **OTP Delivery Methods**: Provide OTP delivery using authenticator application and QR code associated with the OTP generator for maintaining the synchronization.
- **Validate**: Ensure OTPs are valid fora limited time window to enhance security.

3.2.2 Non-Functional Requirements

- **Usability**: The user interfaces for registration, login, and account management should be intuitive and user-friendly.
- **Resource Efficiency**: Optimize resource usage (CPU, memory, storage) to minimize costs and reduce the system's environmental impact.
- **Maintainability**: Design the system in a modular and maintainable way, allowing for updates and enhancements with minimal disruption.
- **Load Testing**: Conduct load testing to identify performance bottlenecks and optimize system performance.

3.3 System Analysis

This section focuses on the analysis phase of the proposed system. It describes the process modeling and system user interaction.

3.3.1 Use Case Diagram

Figure 3-2 presents the use case diagram for the proposed system. This diagram offers a broad overview of the authentication system's functionality.

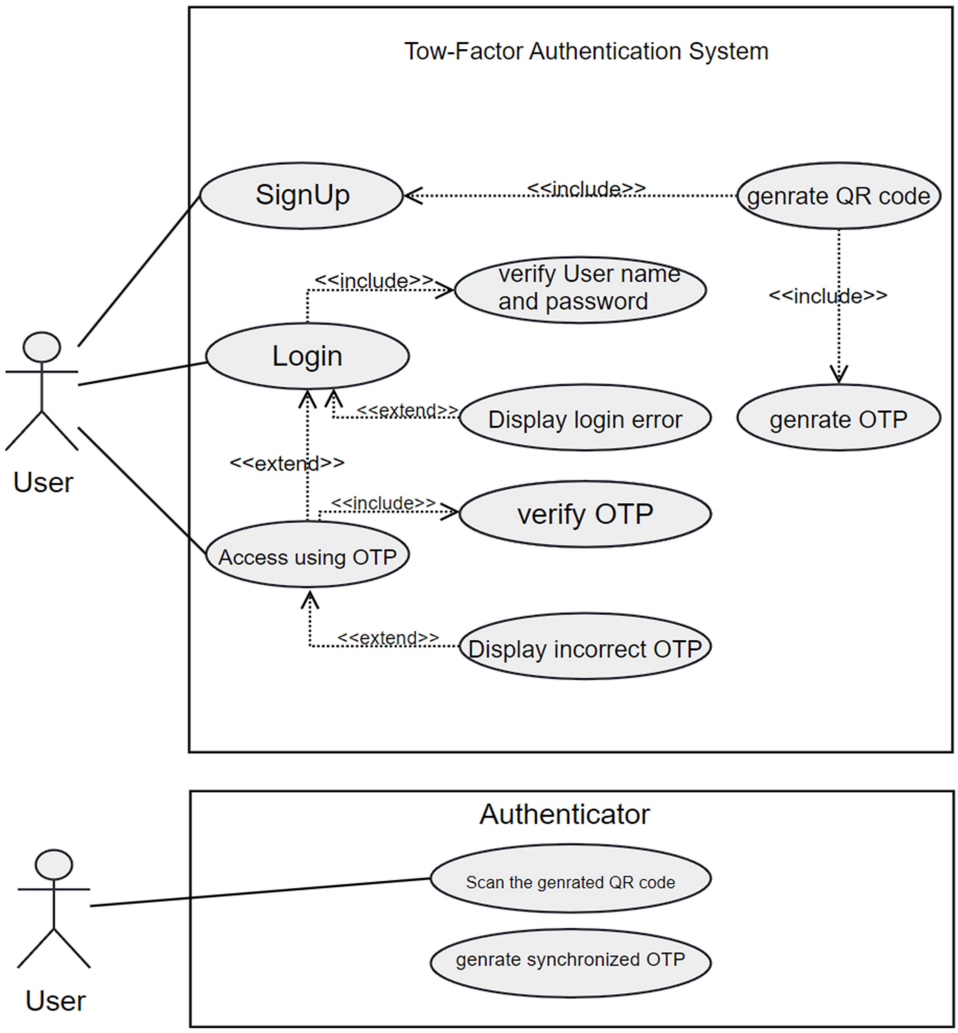


Figure 3-1 Use Case Diagram

The following tables, explains the use cases in detail, each includes actors, description, precondition, and event flow.

Table 3-1 Sign up

Use case number	UC: 1
Use case name	Sign Up
Actors	User
Description	This use case involves the user creating a new account by providing the necessary information.
Pre-condition	The user must have access to the registration page or interface.
Scenario flows	1. User accesses the registration page or interface. 2. System presents a form for the user to enter the required details (username, password, email, etc.) 3. User enters the required information. 4. System validates the entered information. 5. If the validation is successful, the system creates a new user account and stores the account details. 6. User account is registered successfully. 7. QR code is now displayed to user. 8. Scan QR code By using Google Authenticator
Post condition	The user account is successfully registered in the system.

Table 3-2 Generate OTP

Use case number	UC: 2
Use case name	Generate OTP
Actors	User, System
Description	This use case involves the user generating a one-time password (OTP) for authentication or verification purposes.
Pre-condition	The user requests to generate an OTP.
Scenario flows	1.User should sign up to generate OTP. 2.System generates a unique one-time password and associates it with the user's account. 3.System sends the generated OTP to the user.
Post condition	The system generates and sends an OTP to the user for further authentication or verification.

Table 3-3 Generate QR code.

Use case number	UC: 3
Use case name	Generate QR code
Actors	system
Description	Generate QR code containing OTP to user after successful Sign up
Pre-condition	The user must Sign up to display the QR code
Scenario flows	1.The user should scan the generated QR code by using Google authenticator. 2.Now user have a complete synchronize.
Post condition	The user scanned the generated QR code and no need to QR anymore.

Table 3-4 Login

Use case number	UC: 4
Use case name	Login
Actors	User, system
Description	This use case involves the user logging into their account using their credentials.
Pre-condition	The user must have a registered account.
Scenario flows	1.User accesses the login page or interface. 2.System presents a form for the user to enter their credentials (username, password, etc.).

Table 3-5 Verify Username and password

Use case number	UC: 5
Use case name	Verify Username and Password
Actors	User, System
Description	This use case involves the user verifying their entered username and password combination.
Pre-condition	The user must provide their username and password.
Scenario flows	1.User provides their username and password. 2System validates the provided username and password against the stored user account information. 3.System confirms the correctness of the username and password combination. 4.The system returns a verification result indicating the validity of the username and password.
Post condition	The system verifies the correctness of the provided username and password combination.

Table 3-6 Display login error

Use case number	UC: 6
Use case name	Display login error
Actors	User, System
Description	This use case involves displaying an error message when the user fails to log in.
Pre-condition	The user's login attempt fails due to incorrect credentials.
Scenario flows	1. System detects that the login attempt was unsuccessful. 2. System displays an appropriate error message indicating the reason for the failure.
Post condition	An error message is displayed to the user explaining the login failure.

Table 3-7 Access Using OTP

Use case number	UC: 7
Use case name	Access Using OTP
Actors	User, System
Description	This use case involves accessing a secure feature or action using the generated OTP.
Pre-condition	The user has a valid OTP.
Scenario flows	1. User enters the OTP showing in google authenticator. 2. User submits the OTP for verification. 3. System validates the entered OTP. 4. System grants access to the requested secure feature or action if the OTP is valid.
Post condition	The user gains access to the secure feature or action.

Table 3-8 Verify OTP

Use case number	UC: 8
Use case name	Verify OTP
Actors	System
Description	This use case involves verifying the entered OTP for authentication purposes.
Pre-condition	The user has entered an OTP.
Scenario flows	1. System compares the entered OTP with the generated OTP. 2. System determines whether the entered OTP is valid or not.
Post condition	The entered OTP is successfully verified.

Table 3-9 Display incorrect OTP

Use case number	UC: 9
Use case name	Display Incorrect OTP
Actors	User, System
Description	This use case involves displaying an error message when the entered OTP is incorrect.
Pre-condition	The user's entered OTP does not match the generated OTP.
Scenario flows	1. System detects that the entered OTP is incorrect. 2. System displays an appropriate error message indicating the incorrect OTP.
Post condition	An error message is displayed to the user indicating the incorrect OTP.

3.3.2 Flow Chart

Figure 3-2 presents a flow chart depicting the structure of the proposed application program. This visual representation is highly valuable in the process of designing the application program.

3.4 Developmental Methodology

System development encompasses various phases, including planning, development, and maintenance, which are crucial for project success. Here is an extended version of the previous text that includes the maintenance phase:

System development can be approached through various methods, some of which offer more advantages than others. Common system development models include the Waterfall model, Spiral model, and Incremental development. For our application, we have opted for the Agile model, a contemporary software development approach that offers several benefits over the traditional Waterfall model.

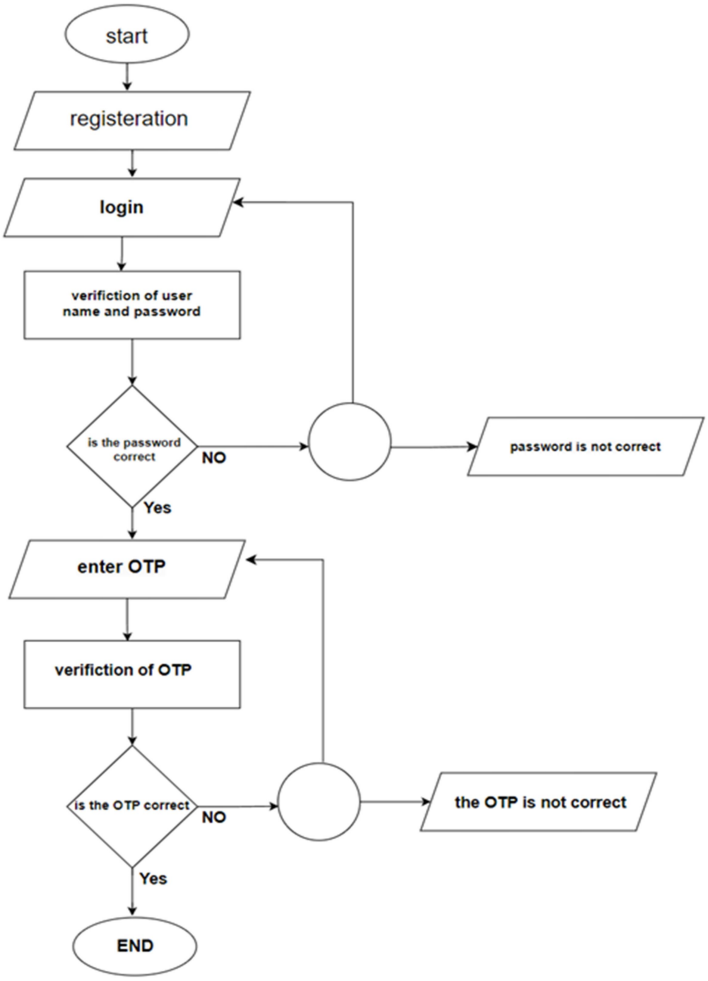


Figure 3-2 Flow chart

The Agile model is characterized by iterative cycles, as illustrated in Figure 3-4. This model enables developers to break down the software application's development into discrete components. Within each cycle, a specific application component is worked on. Given the project's scope, each Agile model cycle comprises five essential steps:

- **Planning**: In the initial planning phase, we define the project scope, objectives, and requirements. This is a crucial step to ensure that the project is well-defined and aligns with the intended goals.
- **Analysis**: During this phase, we gather information and analyze components by identifying the actors and their respective functions.
- **Design**: In this stage, we create the design for the component that is to be

developed.

- **Implementation**: This step involves the actual implementation of the component under development.
- **Evaluation Testing:** Here, we conduct thorough testing of the component being developed.

Following the completion of a cycle, we proceed to the maintenance phase, where we ensure that the system operates smoothly and remains up to date with changing requirements and technology. The maintenance phase involves routine updates, bug fixes, and improvements to enhance the system's performance and longevity. It also includes user support and addressing any issues that may arise after deployment.

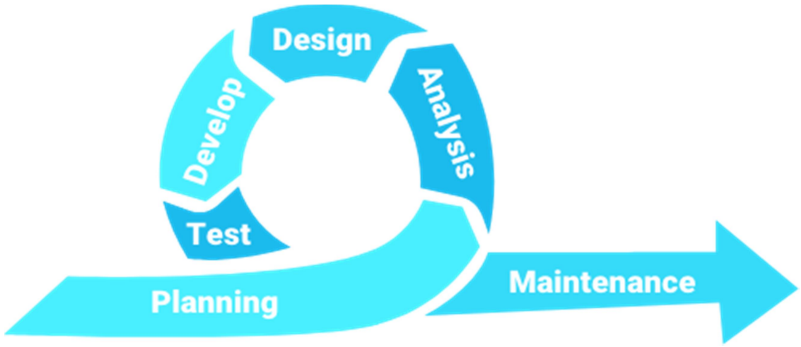


Figure 3-3 Agile model

3.5 Summary

This chapter focuses on the system analysis and the associated functional and nonfunctional requirements and use cases. Functional requirements include user registration, login, password policies, OTP generation and delivery methods, OTP validation, and rate limiting. Non-functional requirements encompass usability, resource efficiency,..etc. The chosen developmental methodology for the project is the Agile model which is considered a flexible and iterative approach that facilities the SDLC. It allows for gathering feedback, early delivery of working software, and addressing the most important aspects of the project.

Chapter 4: System Design

4.1 Introduction

In this chapter, we introduce the system design, considering the key aspects of our proposedsystem. Additionally, we provide an in-depth discussion of the entire architecture of the proposed system, which will be implemented in the subsequent phase.

4.2 Architectural design

The system architecture of our project (multi-factor authentication) involves components such as the user interface, backend server, user management, TOTP generation, authenticator app, time synchronization, TOTP verification, multi-factor integration, and security measures. The user interface allows users to initiate the authentication process, while the backend server handles requests and coordinates TOTP generation and verification. User management stores account information, and TOTP generation generates passwords based on shared secret keys. The authenticator app generates TOTPs on the user's device. Time synchronization ensures accurate TOTP generation, and TOTP verification compares user-provided TOTPs with expected values

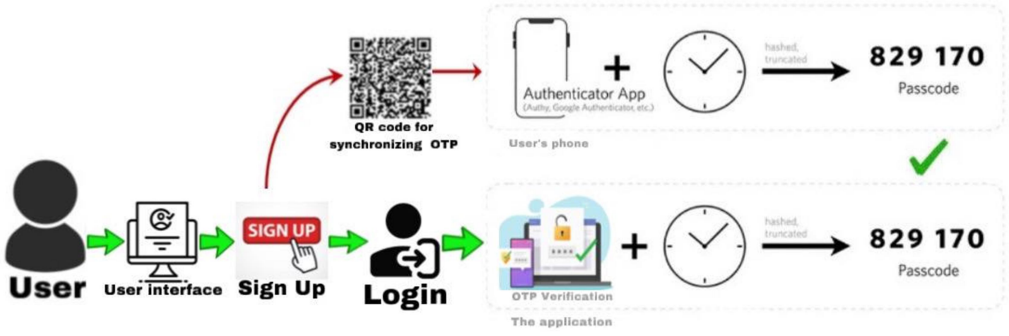


Figure 4-1 architecture design

4.3 Object Oriented Design

4.3.1 Class Diagram

In the class diagram, we've organized and grouped the various elements of the proposed system, such as (users, registration, login, OTP, and OTP generation) based on their specific functions and characteristics. We've used Python classes to represent each entity and definetheir functionalities. Figure 4-2 below provides a visual representation of the essential classes that have been identified and are scheduled for development in this project.

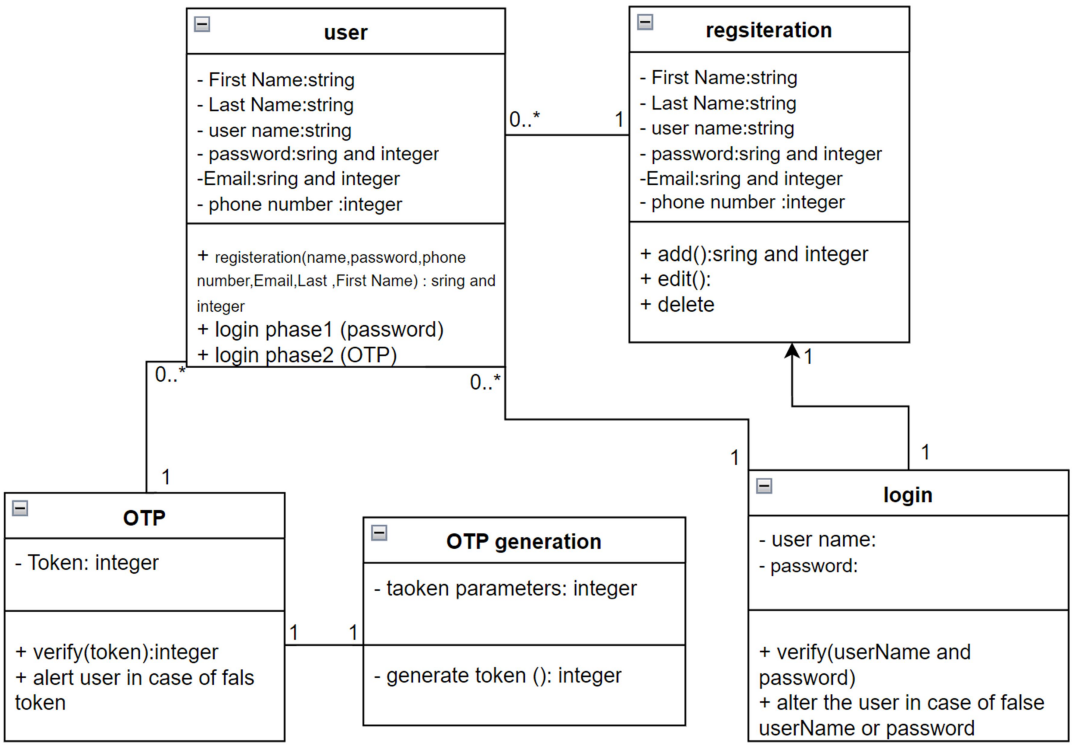


Figure 4-2 Class Diagram

4.3.2 Sequence Diagrams

are a type of interaction diagram that provides a comprehensive depiction of the step-by-step execution of operations. These diagrams depict the interactions between objects within a collaborative context. Figure 4-3 presents the sequence diagrams for the proposed system.These

diagrams provide a time-focused representation of the application flow, .showcasing the sequence of interactions as they occur.

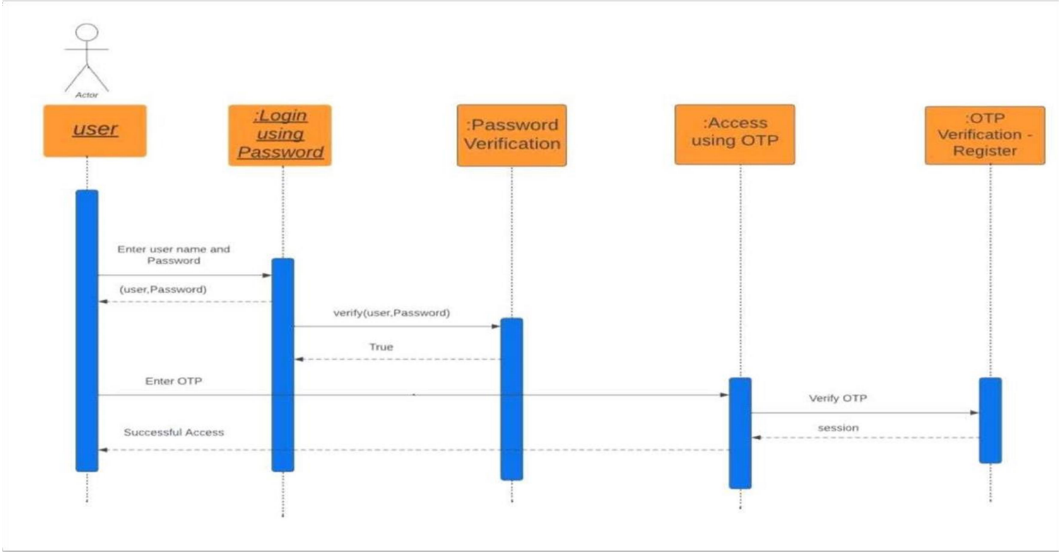


Figure 4-3 Sequence diagram

4.3.3 Activity Diagram

Figure 4-4 illustrates the activity diagram of the activity diagram for the proposed application.

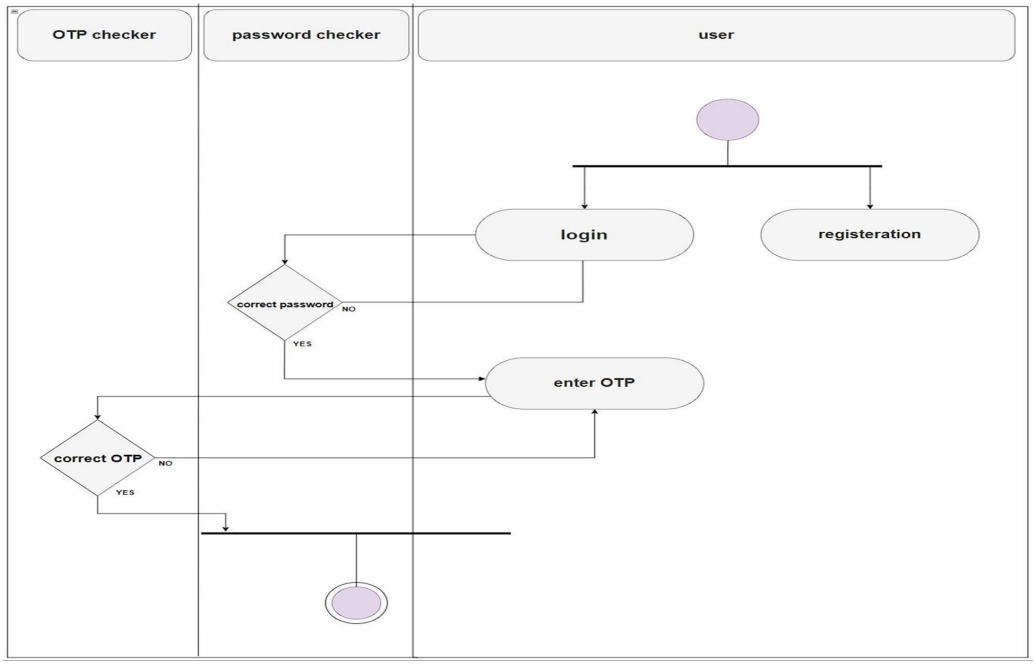


Figure 4-4 Activity diagram

Chapter 5: System Implementation

5.1 Introduction

During this stage, the system specifications are transformed into a functional and dependable solution. This is when the actual coding of the system takes place. The duration and effort required for this phase are significantly influenced by the preceding phases, as both the analysis and design stages serve as the foundation for implementation. In this same phase, initial testing, such as unit testing, is conducted by the developer to confirm if the system's requirements at the unit level align with the development. Typically, it's challenging to accurately predict the duration of this phase because practical challenges may arise when translating the conceptual design into an executable one.

5.2 Tools and Languages

This section presents tools and language used during the system implementation.

Table 5-1 Technology summary

Technology construct	Usage
java	To develop authentication system
NetBeans	An integrated development environment (IDE) for Java
MySQL	To establish a connection to a MySQL database from Java

5.2.1 External Libraries

The following table 5.2 summarizes four main external libraries that were used to develop the authentication system.

Table 5-2 External Libraries

Library	Description
mysql-connector-j-8.1.0	It is the official JDBC driver for MySQL. MySQL Connector/J 8.0 and higher is compatible with all MySQL versions
commons-codec-1.16.0	it's a small library, which includes encoders and decoders for common encoding algorithms
core-java6-3.2.0	This library provides the fundamental building blocks and tools for developing Java applications.
javase-java6-3.2.0	The core image decoding library, and test code. javase, JavaSE-specific client code. android, Android client Barcode Scanner Barcode Scanner.

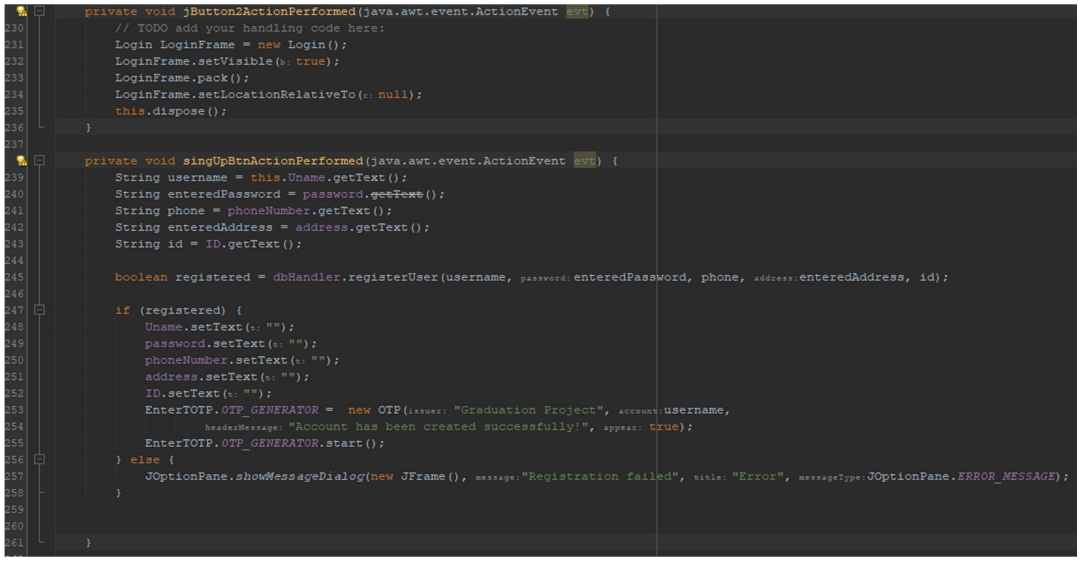
5.3 Main and Most Important Codes

5.3.1 Code for signup

This code snippet shows the user interactions that handle login and user registration in a GUI application in figure 5-1,2. It creates a new login window when a button is clicked and registers the user by storing their information in a database in figure 5-3. And We have established validate a Password Figure 5-3 because Validating passwords provides several benefits for both security in software applications such as:

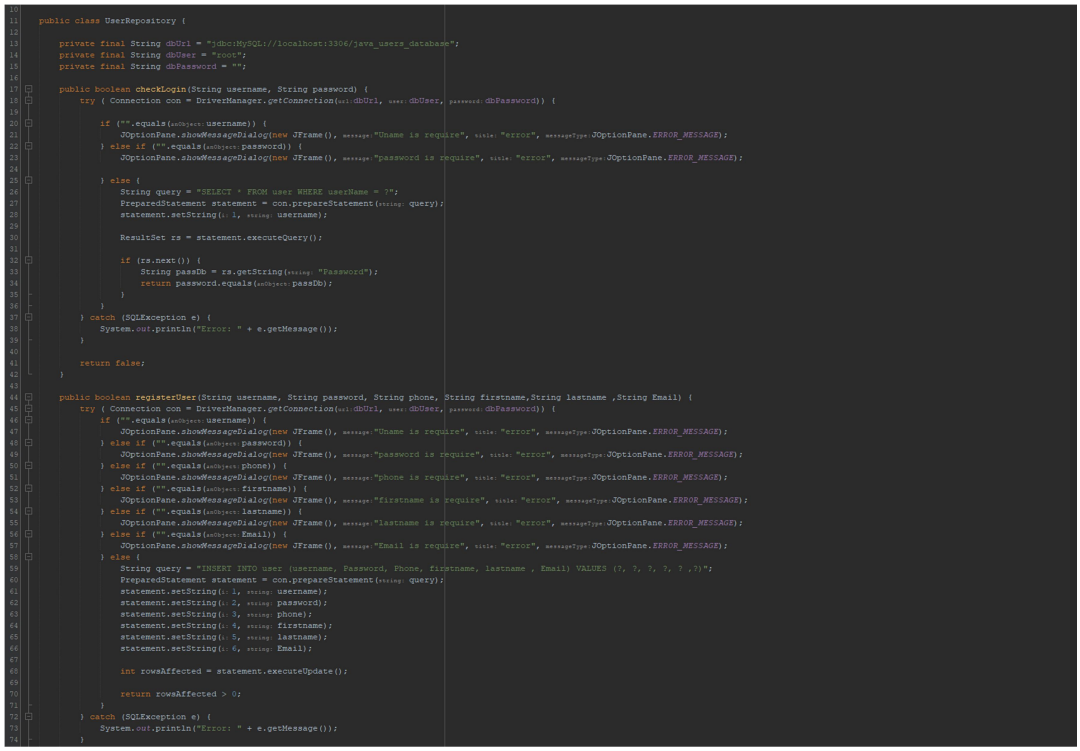
-Prevent Weak Passwords: Validation helps enforce certain criteria, such as minimum length, inclusion of special characters, numbers, and a mix of upper and lower-case letters. This reduces the likelihood of users choosing weak and easily guessable passwords.

-Protection Against Brute Force Attacks: By requiring a minimum length and complexity, the system becomes more resistant to brute force attacks, where attackers systematically try all possible passwords.



```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Login LoginFrame = new Login();
        LoginFrame.setVisible(b: true);
        LoginFrame.pack();
        LoginFrame.setLocationRelativeTo(c: null);
        this.dispose();
    }

    private void singUpBtnActionPerformed(java.awt.event.ActionEvent evt) {
        String username = this.Uname.getText();
        String enteredPassword = password.getText();
        String phone = phoneNumber.getText();
        String enteredAddress = address.getText();
        String id = ID.getText();

        boolean registered = dbHandler.registerUser(username, password: enteredPassword, phone, address:enteredAddress, id);

        if (registered) {
            Uname.setText(s: "");
            password.setText(s: "");
            phoneNumber.setText(s: "");
            address.setText(s: "");
            ID.setText(s: "");
            EnterTOTP.OTP_GENERATOR =  new OTP(issuer: "Graduation Project", account:username,
                    headerMessage: "Account has been created successfully!", appear: true);
            EnterTOTP.OTP_GENERATOR.start();
        } else {
            JOptionPane.showMessageDialog(new JFrame(), message:"Registration failed", title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
        }


    }
```

Figure 5-1 code for sign up



```
public class UserRepository {

    private final String dbUrl = "jdbc:MySQL://localhost:3306/java_users_database";
    private final String dbUser = "root";
    private final String dbPassword = "";

    public boolean checkLogin(String username, String password) {
        try ( Connection con = DriverManager.getConnection(url:dbUrl, user:dbUser, password:dbPassword)) {

            if ("".equals(anObject:username)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"Uname is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject:password)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"password is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);

            } else {
                String query = "SELECT * FROM user WHERE userName = ?";
                PreparedStatement statement = con.prepareStatement(string: query);
                statement.setString(i: 1, string: username);

                ResultSet rs = statement.executeQuery();

                if (rs.next()) {
                    String passDb = rs.getString(string: "Password");
                    return password.equals(anObject:passDb);
                }
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }

        return false;
    }

    public boolean registerUser(String username, String password, String phone, String firstname,String lastname ,String Email) {
        try ( Connection con = DriverManager.getConnection(url:dbUrl, user:dbUser, password:dbPassword)) {
            if ("".equals(anObject: username)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"Uname is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject:password)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"password is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject: phone)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"phone is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject: firstname)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"firstname is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject: lastname)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"lastname is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else if ("".equals(anObject: Email)) {
                JOptionPane.showMessageDialog(new JFrame(), message:"Email is require", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
            } else {
                String query = "INSERT INTO user (username, Password, Phone, firstname, lastname , Email) VALUES (?, ?, ?, ?, ? ,?)";
                PreparedStatement statement = con.prepareStatement(string: query);
                statement.setString(i: 1, string: username);
                statement.setString(i: 2, string: password);
                statement.setString(i: 3, string: phone);
                statement.setString(i: 4, string: firstname);
                statement.setString(i: 5, string: lastname);
                statement.setString(i: 6, string: Email);

                int rowsAffected = statement.executeUpdate();

                return rowsAffected > 0;
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
```

Figure 5-2 code for Sign up page.

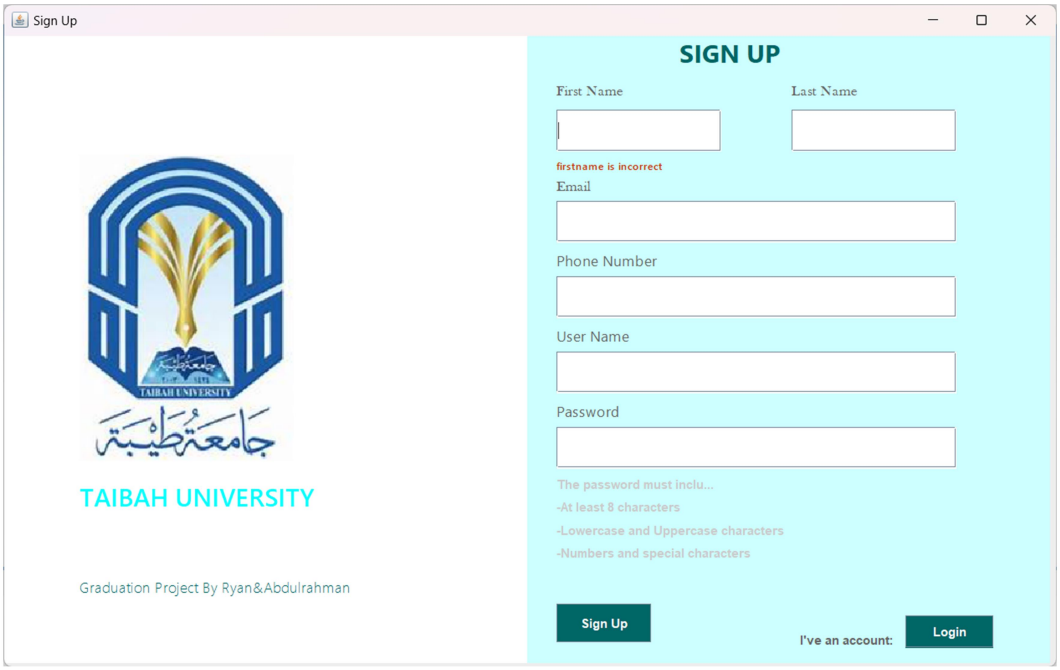


Figure 5-3 Sign up page.

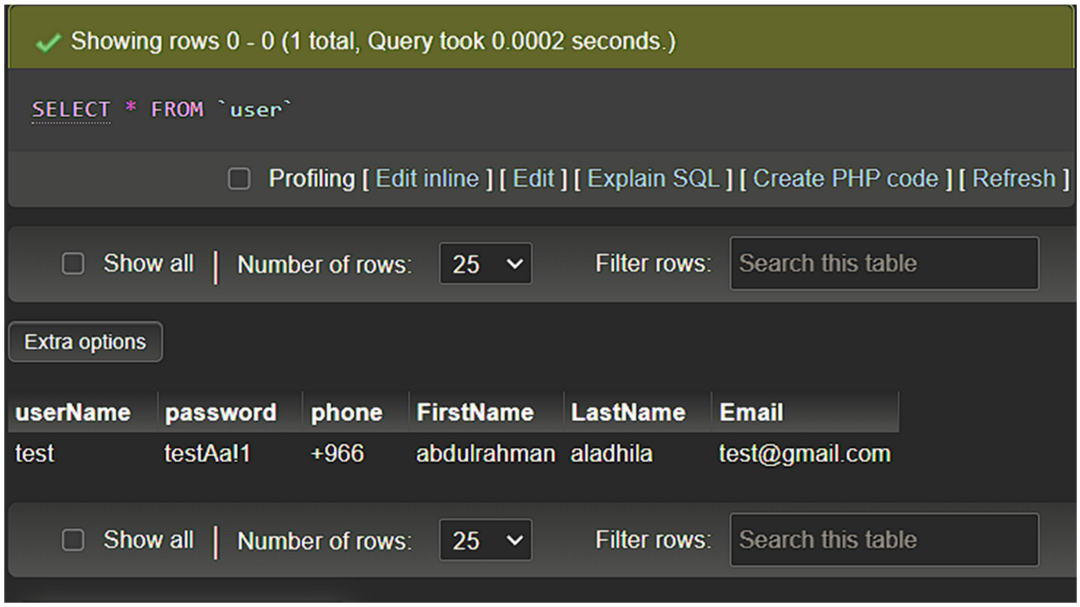


Figure 5-4 Register in the database.

5.3.2 Database

First, we created a database in MySQL. because it is an open-source database, meaning it is freely available for use and modification Figure 5-4. This reduces software costs and allows for customization to meet specific needs. And is known for its simplicity and ease of use. and can handle large amounts of data and supports scalability by allowing users to efficiently manage and organize databases as they grow. It's suitable for both small-scale projects and large enterprise applications. MySQL includes robust security features to protect data integrity and privacy. It supports user authentication, access controls, and encryption, helping developers implement secure database applications. Integration Capabilities MySQL integrates well with various programming languages and development frameworks. It provides connectors and APIs for languages such as Java, Python, PHP, and more, facilitating seamless integration with different applications. Overall, MySQL is a versatile and powerful database management system that is widely used in web development, enterprise applications, and various other domains due to its reliability, performance, and extensive feature set.

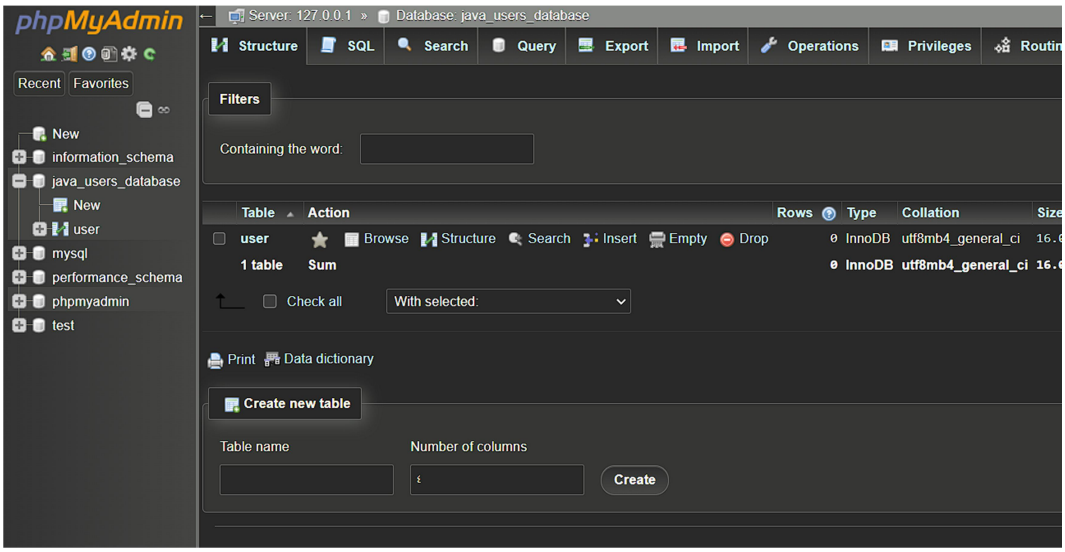


Figure 5-5 MySQL database

5.3.3 Code for login

This code sets up a basic login screen with functionality for user login and sign-up. It interacts with a user database (presumably implemented in the User Repository class) to authenticate users and transition to the next step after successful login.

```
public class Login extends javax.swing.JFrame {

    private final UserRepository dbHandler = new UserRepository();

    public Login() {
        initComponents();
        if(EnterTOTP.OTP_GENERATOR == null){
                EnterTOTP.OTP_GENERATOR =  new OTP(issuer: "Graduation Project", account:Uname.getText(),
                        headerMessage: "Account has been created successfully!", appear: false);
                EnterTOTP.OTP_GENERATOR.start();
        }
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        SignUp SignUpFrame = new SignUp();
        SignUpFrame.setVisible(b: true);
        SignUpFrame.pack();
        SignUpFrame.setLocationRelativeTo(c: null);
        this.dispose();
    }

    private void LoginBtnActionPerformed(java.awt.event.ActionEvent evt) {
        String enteredUsername = Uname.getText();
        String enteredPassword = password.getText();

        boolean loginSuccessful = dbHandler.checkLogin(username: enteredUsername, password: enteredPassword);

        if (loginSuccessful) {
            EnterTOTP enterTOTPFrame = new EnterTOTP(account:enteredUsername);
            this.dispose();
            enterTOTPFrame.setVisible(b: true);
            enterTOTPFrame.pack();
            enterTOTPFrame.setLocationRelativeTo(c: null);
        } else {
            JOptionPane.showMessageDialog(new JFrame(), message:"Incorrect username or password", title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
        }
        password.setText(s: "");

    }
```

Figure 5-6 code for login.

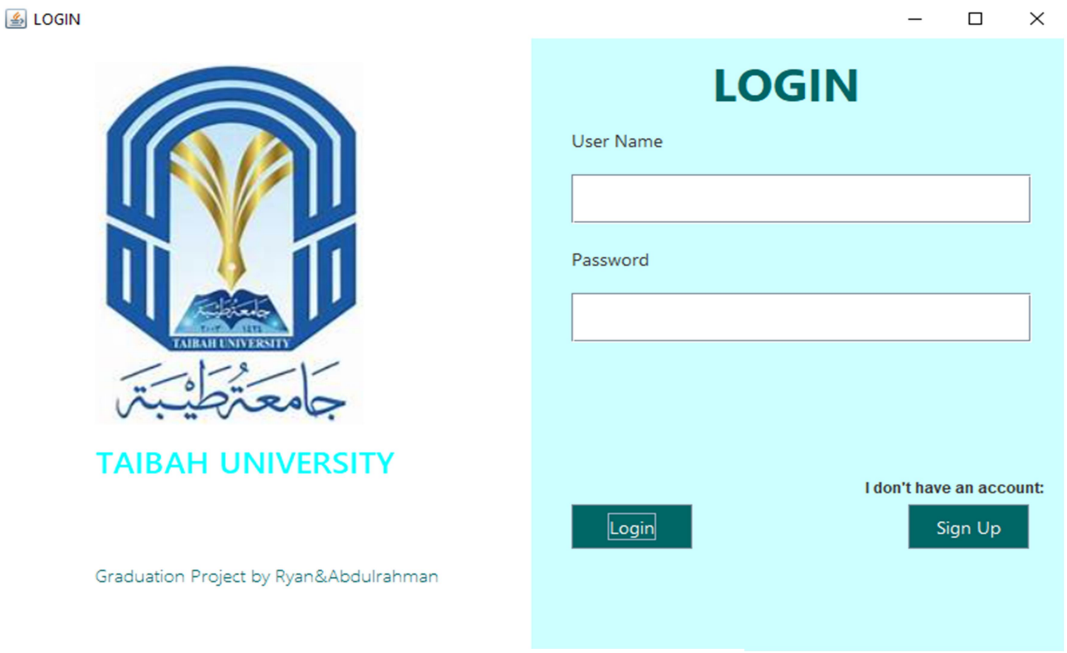


Figure 5-7 login page

5.3.4 Code for OTP generation

The code generates an OTP token by calculating a counter value based on a timestamp, generating a hash using the HMAC-SHA1 algorithm with a shared secret key, extracting part of the hash, and converting it into a digital OTP token. The code also includes methods to generate an OTP authentication URL, generate a QR code image for the URL, and display the QR code and OTP code in the graphical user interface (GUI).

```
public String generateTOTP(String issuer, String account, long timestamp) throws NoSuchAlgorithmException, InvalidKeyException {
    long counter = timestamp / TIME_STEP;

    byte[] key = new Base32().decode(pArray:secretKey);
    byte[] msg = ByteBuffer.allocate(capacity: 8).putLong(value: counter).array();

    Mac hmac = Mac.getInstance(algorithm:"HmacSHA1");
    SecretKeySpec secretKey = new SecretKeySpec(key, algorithm:"HmacSHA1");
    hmac.init(key:secretKey);

    byte[] hash = hmac.doFinal(input: msg);
    int offset = hash[hash.length - 1] & 0xf;
    int binary
            = ((hash[offset] & 0x7f) << 24)
            | ((hash[offset + 1] & 0xff) << 16)
            | ((hash[offset + 2] & 0xff) << 8)
            | (hash[offset + 3] & 0xff);
    int otp = binary % (int) Math.pow(a: 10, b: TOTP_LENGTH);

    String result = Integer.toString(i: otp);
    while (result.length() < TOTP_LENGTH) {
        result = "0" + result;
    }
    return result;
}
```

```
public class OTP extends Thread {

    private String issuer;
    private String account = "";
    private String totp;
    private JFrame frame;
    private JLabel imageLabel = new JLabel();
    private JLabel codeLabel = new JLabel();
    private String headerMessage = "";
    private boolean appear;
    private boolean stop = false;

    private static final int TOTP_LENGTH = 6;
    private static final int TIME_STEP = 60; // 60 seconds
    private static final int SECRET_KEY_LENGTH = 20; // Adjust as needed
    private static final String secretKey = ("4HJMXVZMCLTAPXP7NCTJ4XP5VLXQDEOM");

    public OTP(String issuer) {
        this.issuer = issuer;
    }

    @Override
    public void run() {
        long currentTime, start;
        while (!stop) {
            try {
                    currentTime = System.currentTimeMillis() / 1000;
                    start = currentTime;

                    totp = generateTOTP(issuer, account, timestamp:currentTime);
                    if (appear) {
                        this.codeLabel.setText(text: totp);
                    }

                    currentTime = System.currentTimeMillis() / 1000;
                    String otpAuthURL = generateOTPAuthURL(issuer, account, currentTime - start);
```

Figure 5-8 code for generate OTP.

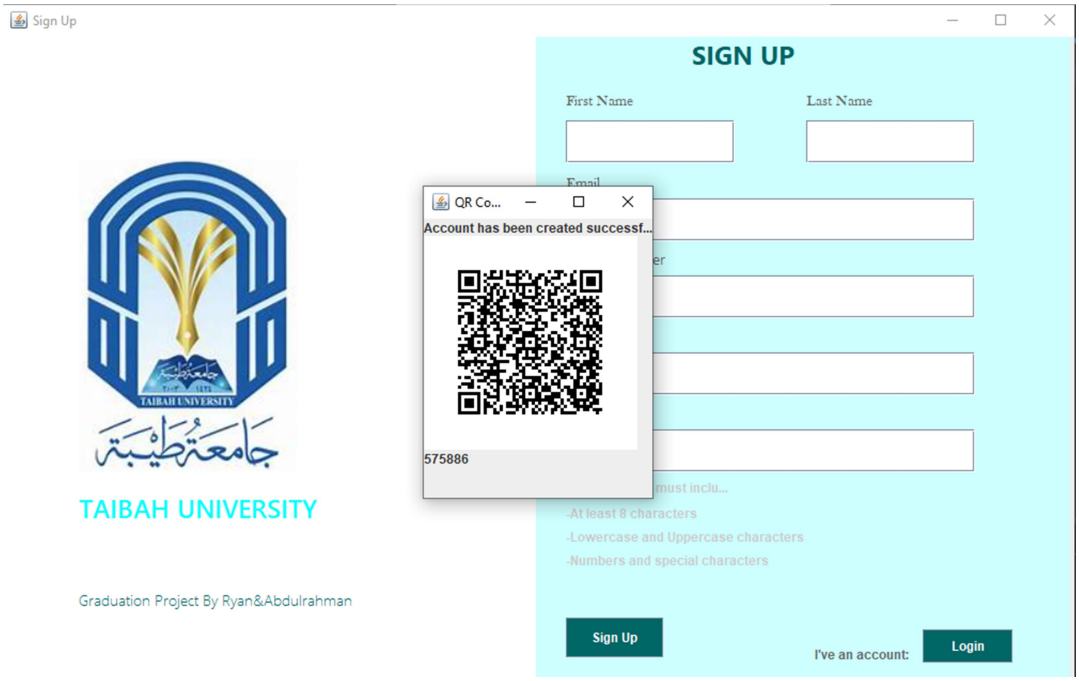


Figure 5-9 for generate OTP.

5.3.5 Code for Generating and displaying the QR code

This piece of code provides a way to generate and display OTP codes using QR code for easy scanning. The OTP category can be used in an application or system that requires OTP authentication. The QR code appears after registration is completed and is scanned using Google Authenticator to maintain synchronization.

```
public String generateOTPAuthURL(String issuer, String account, long timestamp) {
    return "otpauth://totp/" + issuer + ":" + account + "?secret=" + secretKey
            + "&issuer=" + issuer + "&algorithm=SHA1&digits=" + TOTP_LENGTH
            + "&period=" + TIME_STEP + "&timestamp=" + timestamp;
}

public BufferedImage generateQRCode(String otpAuthURL, int width, int height) {
    try {
        Map<com.google.zxing.EncodeHintType, Object> hints = new HashMap<>();
        hints.put(key: com.google.zxing.EncodeHintType.CHARACTER_SET, value: "UTF-8");
        BitMatrix matrix = new MultiFormatWriter().encode(contents: otpAuthURL, format: BarcodeFormat.QR_CODE, width, height, hints);

        return MatrixToImageWriter.toBufferedImage(matrix);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

public String decodeQRCodeFromImage(BufferedImage image) {
    try {
        LuminanceSource source = new BufferedImageLuminanceSource(image);
        BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));

        Map<DecodeHintType, Object> hints = new EnumMap<>(keyType: DecodeHintType.class);
        hints.put(key: DecodeHintType.CHARACTER_SET, value: "UTF-8");

        Reader reader = new MultiFormatReader();
        Result result = reader.decode(bb: bitmap, map: hints);

        return result.getText();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Figure 5-10 code for generate QR code.

```
    public JFrame displayQRCode(String message) {
        JFrame frame = new JFrame(title: "QR Code");
        frame.setDefaultCloseOperation(operation:JFrame.DISPOSE_ON_CLOSE);

        // Create a JPanel to hold the image and code
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(target:panel, axis: BoxLayout.Y_AXIS));
        codeLabel.setHorizontalAlignment(alignment:SwingConstants.CENTER); // Center the text

        panel.add(new JLabel(text: message));
        panel.add(comp: imageLabel);
        panel.add(comp: codeLabel);

        // Add the panel to the frame
        frame.getContentPane().add(comp: panel);
        frame.setSize(width: 230, height:300);

        frame.setLocationRelativeTo(c: null);
        frame.setVisible(b: true);
        return frame;
    }

}
```

Figure 5-11 code for display QR code

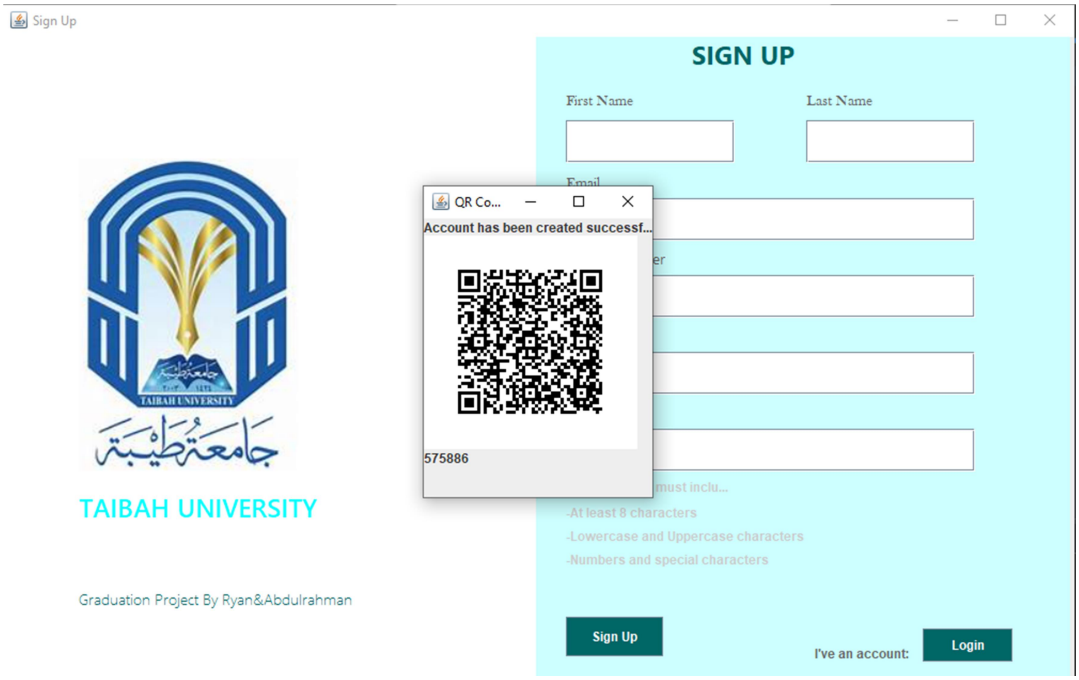


Figure 5-12 Displayed QR code

5.3.6 Code for verification

this code sets up a basic OTP entry form with a login button, verifies the entered OTP against a generated OTP, and performs certain actions based on the verification result.

```
public class EnterTOTP extends javax.swing.JFrame {

    public static OTP OTP_GENERATOR;
    /**
     * Creates new form enterTOTP
     */
    public EnterTOTP(String account) {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    Generated Code

    private void loginActionPerformed(java.awt.event.ActionEvent evt) {
          if(OTP_GENERATOR != null && OTP.getText().equals(anObject: this.OTP_GENERATOR.getTotp())){
              this.dispose();
              // Go to the login page
              MainFrame mainFrame = new MainFrame();
              mainFrame.setVisible(b: true);
              mainFrame.pack();
              mainFrame.setLocationRelativeTo(c: null);
          } else{
               JOptionPane.showMessageDialog(new JFrame(), message:"Incorrect OTP", title: "error", messageType:JOptionPane.ERROR_MESSAGE);
          }
    }
```

Figure 5-13 code for Verification page

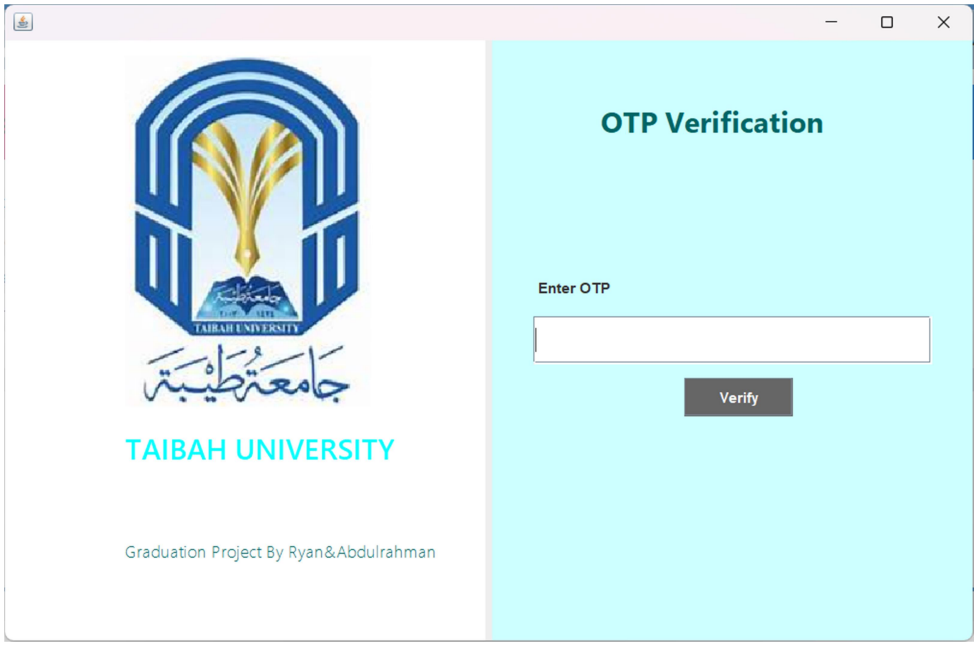


Figure 5-14 Verification page

Chapter 6: Evaluation Testing

6.1 Introduction

The testing process is part of the software development cycle. It involves gathering information about the quality of a system's operation and the implementation of its various features. Evaluation testing is a process that aims to evaluate the quality of a system's operation. It involves gathering information about its requirements and characteristics. For testing of our project, we have followed three levels of testing: tnit testing, integration testing, and system testing.

6.2 system Testing

6.2.1 Testing Strategy

The following are all the types of testing which are going to be illustrated more in the following sections:

- Unit Testing
- Integration Testing
- Performance Testing

6.2.2 Test Items

This section is going to demonstrate the user classes and who are going to be served by the proposed model eventually.

6.2.2.1 User Procedures

There are 7 main components that were mentioned previously. The combination are as the following.

- Graphical User Interface (GUI)
- Database
- Registration
- OTP&QR generation
- Login
- OTP entry
- OTP verification

6.2.2.2 Operator Procedures

The requirements which should be available in order for the testing to take a place that leads

to testing the model in an accurate way are as the following.

- Computer Device
- NetBeans to Run Project
- google Authenticator to scan the Generated QR Code
- My SQL database

6.2.3 Testing Approach

This section describes the approaches for testing which are going to be illustrated in sufficient detail to explain the major tasks, and types of testing to be performed. Also, explaining the methods, and the purpose of using these methods.

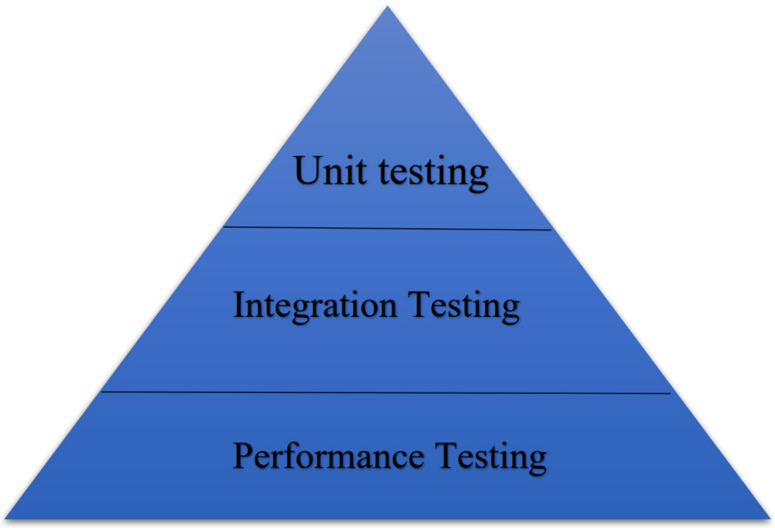


Figure 6-1 Testing Strategy

6.2.3.1 Unit Testing

Individual units or components of software are tested in unit testing, which is a sort of software testing. The goal is to ensure that each unit of software code works as intended. Developers perform unit testing throughout the development (coding) phase of an application. Unit tests are used to isolate a part of code and ensure that it is correct. A singular function, method, procedure, module, or object might be considered a unit. The following shows the unit testing which has been performed on the project.

6.2.3.1.1 Run Program

Table 6-1 Run Program

Step	Action	UI Unit Response
1	Run Program	Login page will displayed

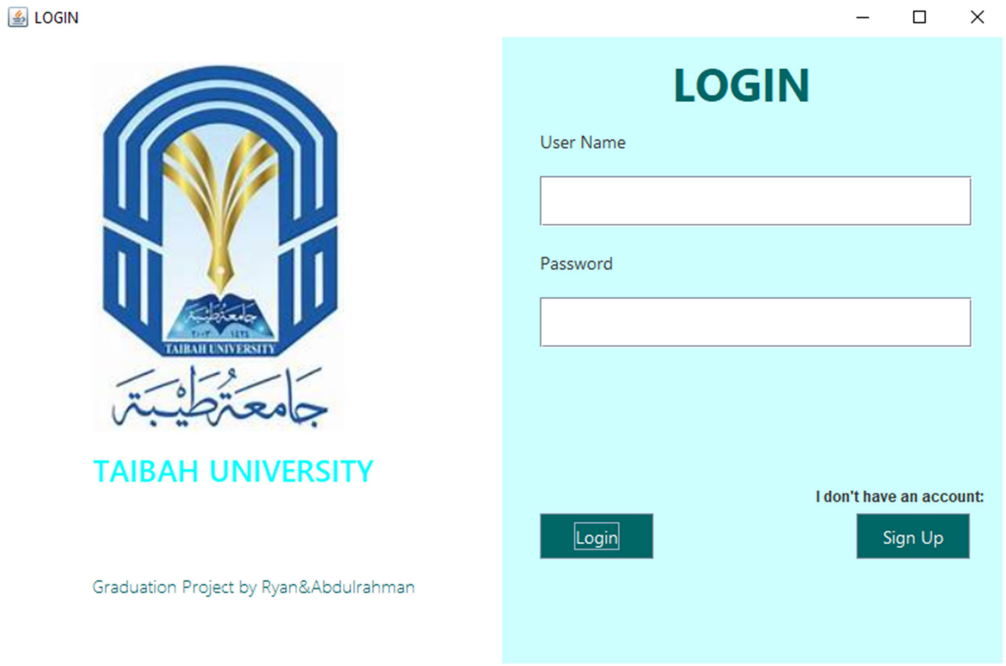


Figure 6-2 Login page

6.2.3.1.2 Sign Up

Table 6-2 Sign up

Step	Action	UI Unit Response
1	Click on Sign up	UI will move to Registration page
2	After Entered Registration Information Click on Sign Up	A QR code will appear

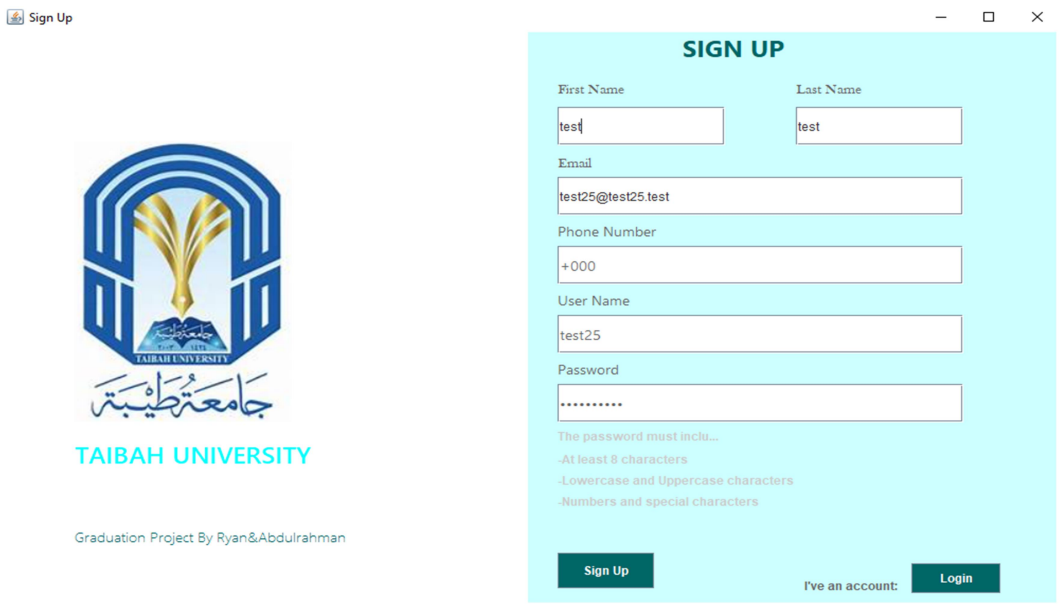


Figure 6-3 Sign up page.

6.2.3.1.3 QR code Generation

Step	Action	UI Unit Response
1	Scan A generated QR By using Google Authenticator	The Generated QR code still disable and changing every 60 second

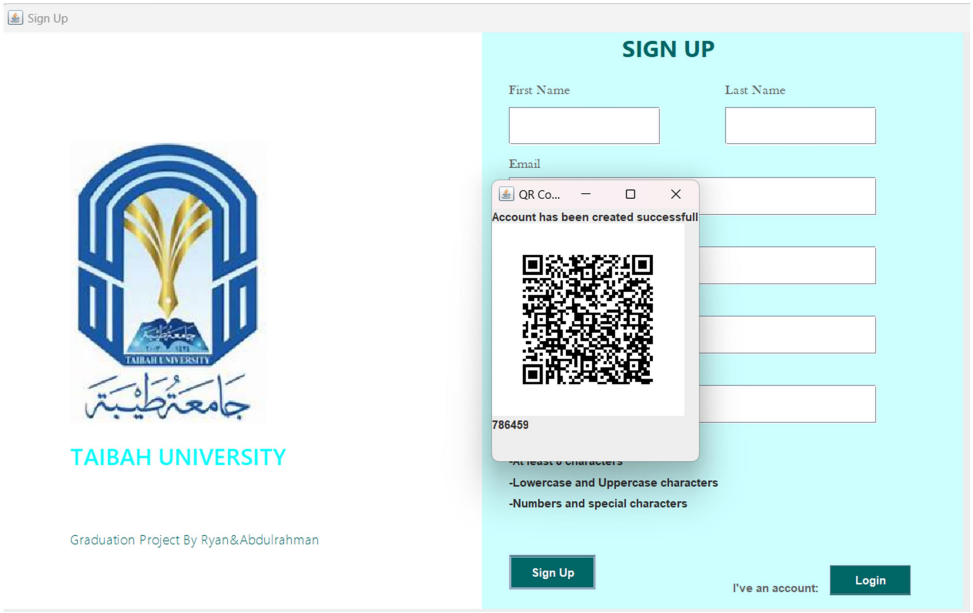


Figure 6-4 QR code Generation for Scanning

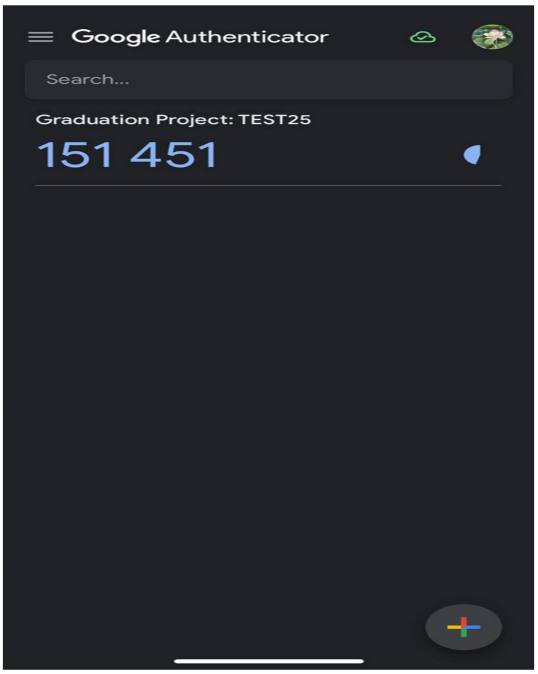


Figure 6-5 OTP code on google Authenticator.

6.2.3.1.4 Login

Table 6-4

Step	Action	UI Unit Response
1	Enter Username and Password	IF username and password correct UI will move to OTP entry page

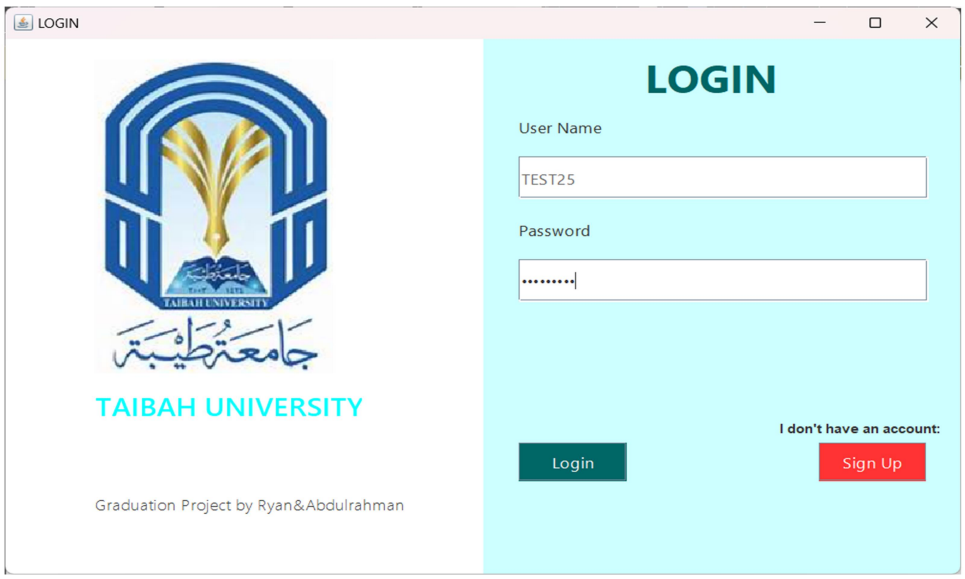


Figure 6-6 Login.

6.2.3.1.4 OTP verification

Table 6-5 OTP verification

Step	Action	UI Unit Response
1	Enter The verification code displayed in Google Authenticator	If the OTP is correct, UI will move to welcome page

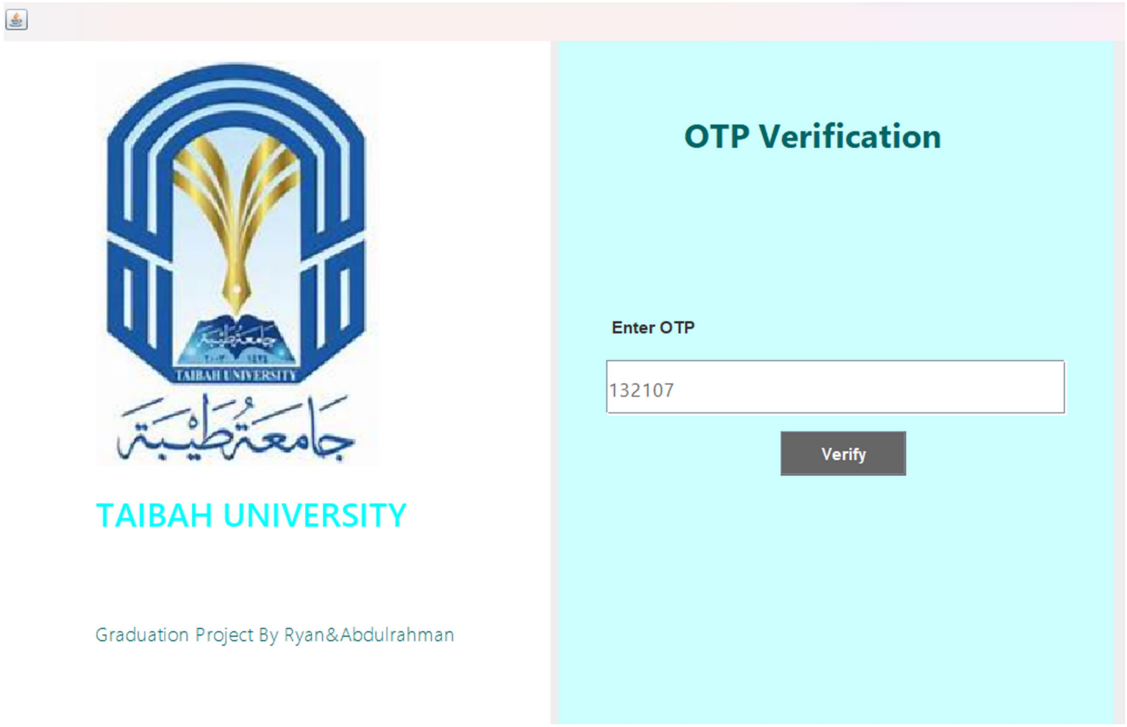


Figure 6-7: OTP verification

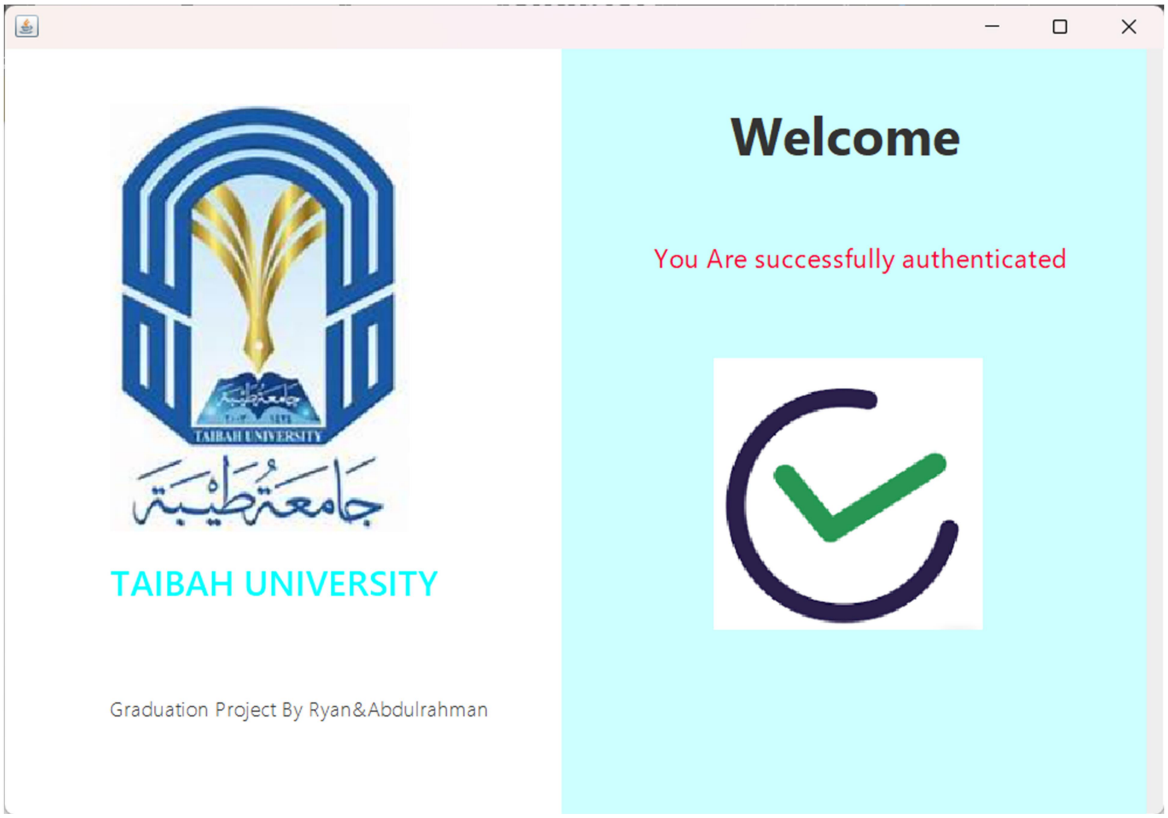


Figure 6-8 Welcome page

6.2.3.2 Integration testing

This type of testing is usually performed on a software project where the various components of the software are integrated. The goal of this type of testing is to expose any defects that might occur when these components are combined. The integration test checks whether a component passes correct values to other components. It also tests compatibility and interaction between components. Integration testing focuses on that the model works correctly as designed. The following figures show sample of integration testing.

6.2.4.1 Sign up & QR generation with login.

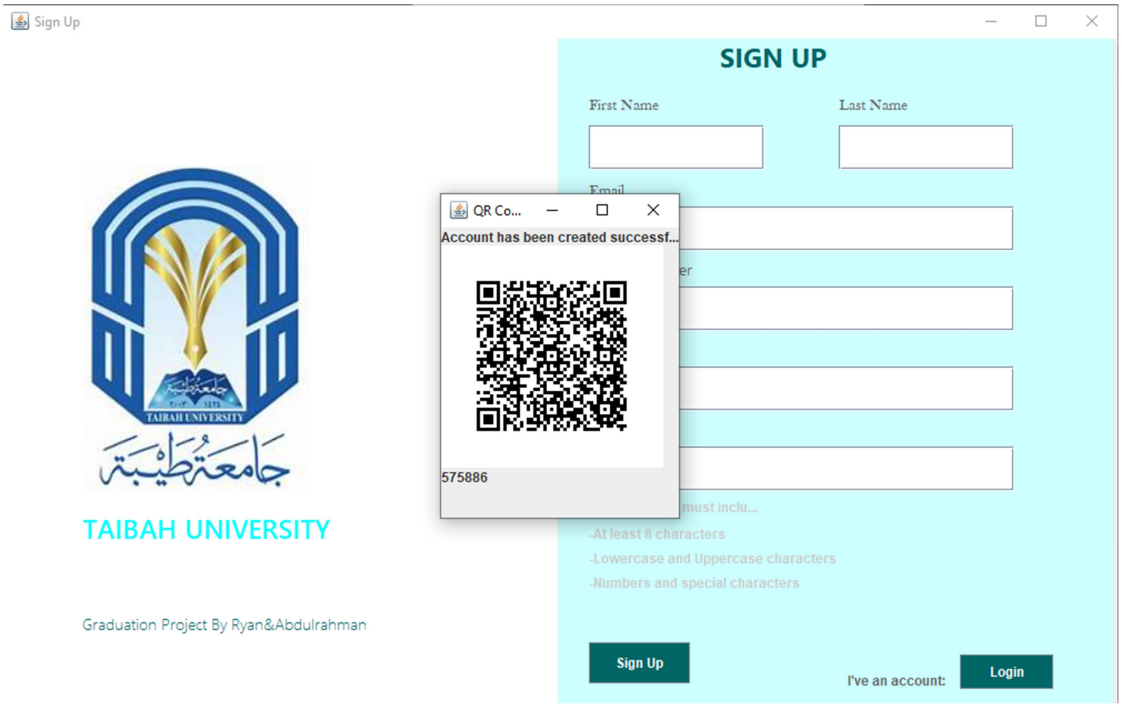


Figure 6-9 QR code generated after Sign Up

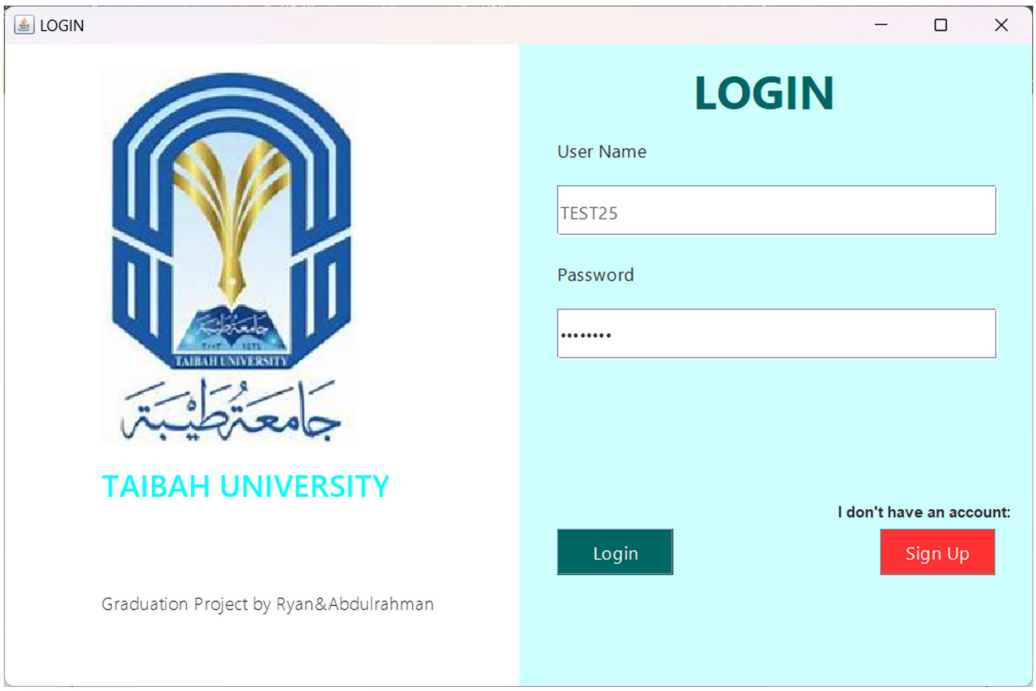


Figure 6-10 Login

6.2.4.2 OTP Verification regarding the OTP code shown in Authenticator app.

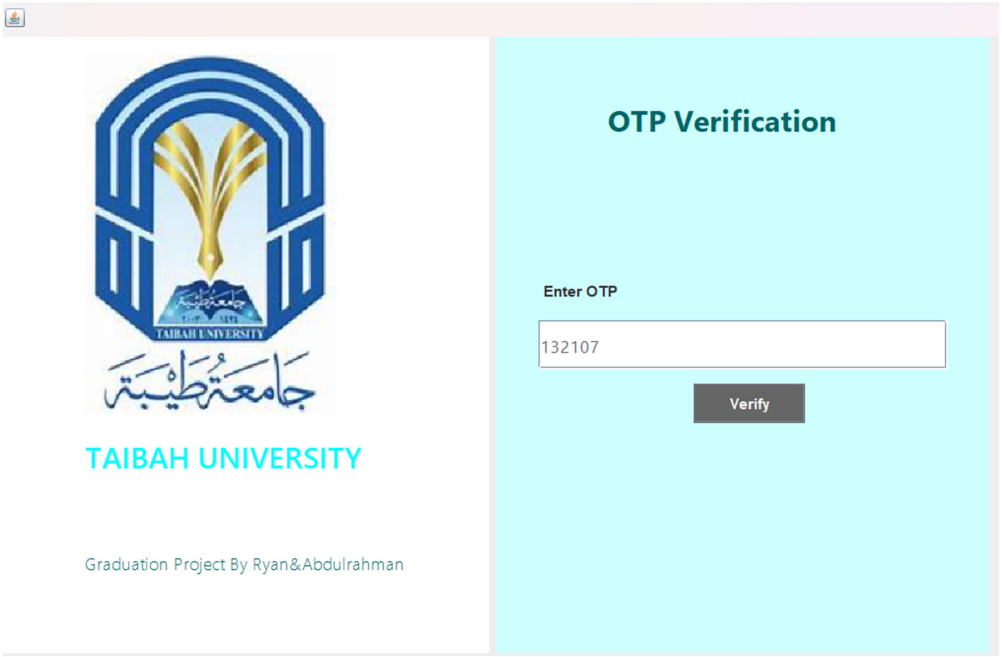


Figure 6-11: OTP Verification regarding Google Authenticator

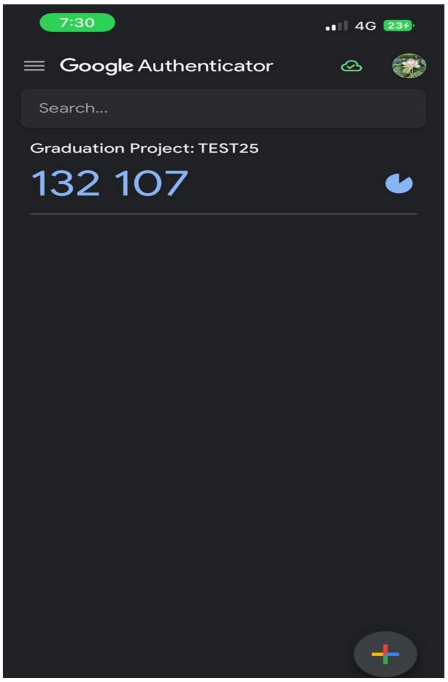


Figure 6-12 OTP on Google Authenticator

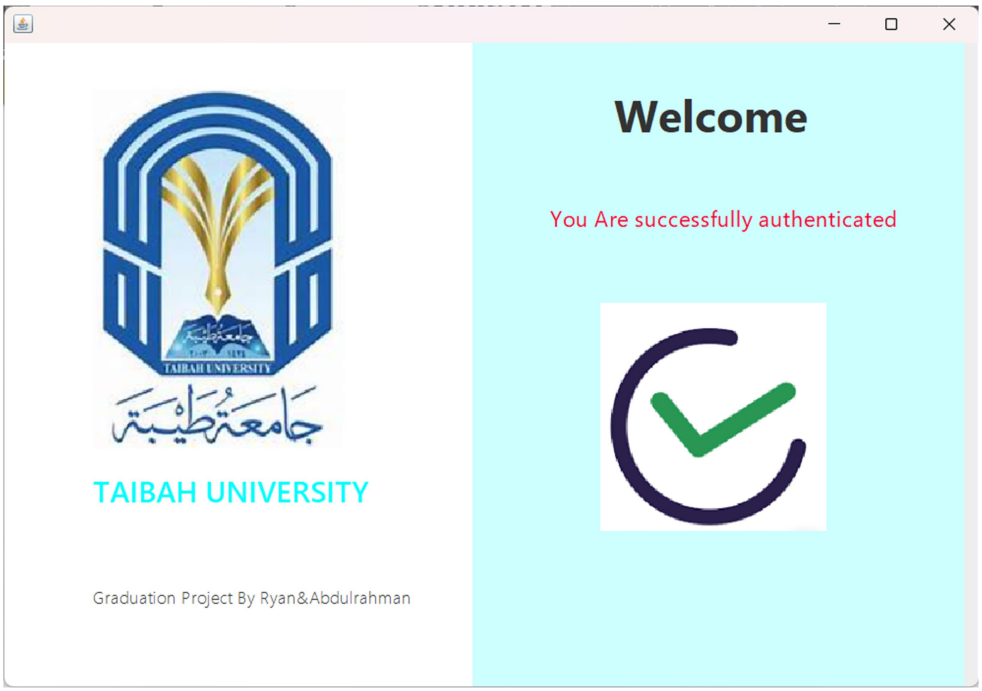


Figure 6-13 Verification page

6.2.3.3 Performance Testing

The expectation is that the response time of the detection action is going to be acceptable. Also, the response time of the model that handles the matching with dataset is going to be reasonable.

Table 6.6: Test Cases

No	Input	Expected Output			Real Output			
		Sign-Up	Sign-In	OTP Verification	Sign-Up	Sign-In	OTP Verification	Result
1	Name: Ryan Alharbi Email: RyanAlharbi25@gmail.com Phone Number: +966550596855 Username: Ryan Password: R11m22h33@	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass

2	Name: Abdulrahman Almutairi Email: AbdulrahmanA12@gmail.com Phone Number: +966550596855 Username: Abdulrahman Password: 18Bac87#	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
3	Name: Ali Max Email: aAliRR15@hotmail.com Phone Number: +966550596855 Username: Ali Password: pqrt1122&	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
4	Name: Salman KKd Email: Salmanui10@hotmail.com Phone Number: +966550596855 Username: Salman99 Password: Mub190c@	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
5	Name: Omar glm Email: Omarca1144@gmail.com Phone Number: +966550596855 Username: 1356122 Password: @aami1295	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
6	Name: Naif nmk Email: naifalHu9@hotmail.com Phone Number: +966550596855 Username: Naif1865 Password: 04b7A3@1	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
7	Name: Khalid kdj	Successfu	Successfu	Successful	Successfu	Successfu	Successful	Pass

	Email: khali10@gmail.com Phone Number: +966550596855 Username: Khalid21 Password: Bu11741$	l Sign-Up	l Sign-In	Verification	l Sign-Up	l Sign-In	Verification	
8	Name: Ahmed sda Email: Acbvdggn18@Outlook.sa Phone Number: +966550596855 Username: Ahmed Password:1023KJs@	Successful Sign-Up	Successful Sign-In	Successful Verification	Successful Sign-Up	Successful Sign-In	Successful Verification	Pass
9	Name: Naser bcv Email: Mfjsla1144@outlook.com Phone Number: +966550596855 Username: Naser Password: 123456789	Incorrect Password	Incorrect Username or Password	-	Incorrect Password	Incorrect Username or Password	-	Pass
10	Name: Hasan Hny Email: Hjmr10372@gmail.com Phone Number: +966550596855 Username: Hasan Password: asdzxcgh	Incorrect Password	Incorrect Username or Password	-	Incorrect Password	Incorrect Username or Password	-	Pass
11	Name: Abdulah Jlt Email: Abdulah9293@gmail.com Phone Number: +966550596855 Username: Abdulah12@ Password: Fhb145lm#	Incorrect Username	Incorrect Username or Password	-	Incorrect Username	Incorrect Username or Password	-	Pass
12	Name: Zyad zzv Email: ZyadAl1298 Phone Number: +966550596855	Incorrect Email	Incorrect Username or Password	-	Incorrect Email	Incorrect Username or Password	-	Pass

	Username: Zyad Password: 11354Ab@							
13	Name: Nawaf Ma Email: JK12233Mi@outlook.com Phone Number: +966550596855 Username: Nawaf Password: BNk124o#	Incorrect Name	Incorrect Username or Password	-	Incorrect Name	Incorrect Username or Password	-	Pass
14	Name: Yasir blg Email: JCKlm54@gmail.com Phone Number: 1937328753 Username: Yasir Password: Kmz197@2	Incorrect Phone number	Incorrect Username or Password	-	Incorrect Phone number	Incorrect Username or Password	-	Pass

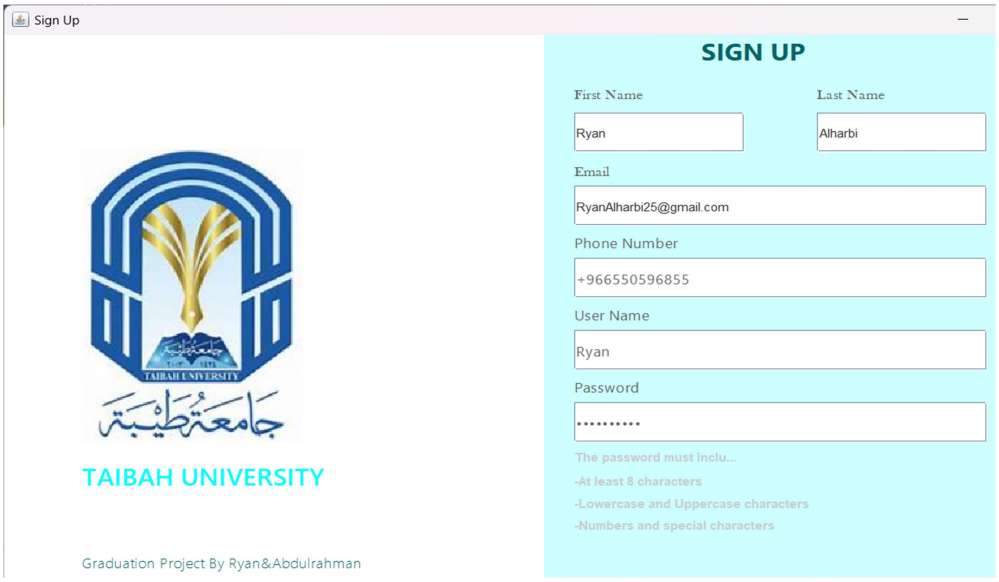


Figure 6-14: Test Case Number 1

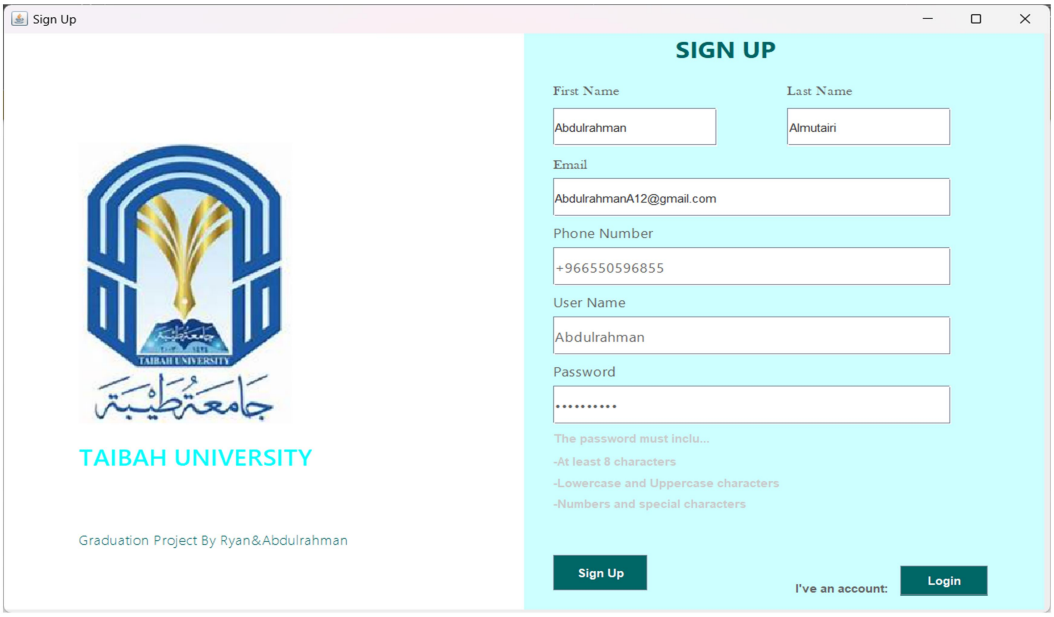


Figure 6-15: Test Case Number 2

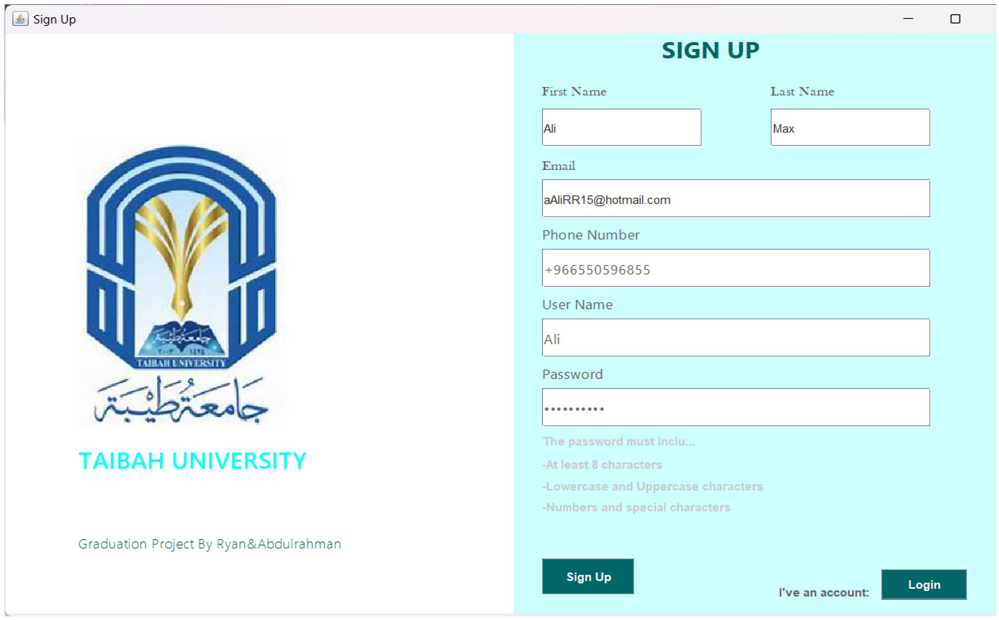


Figure 6-16: Test Case Number 3

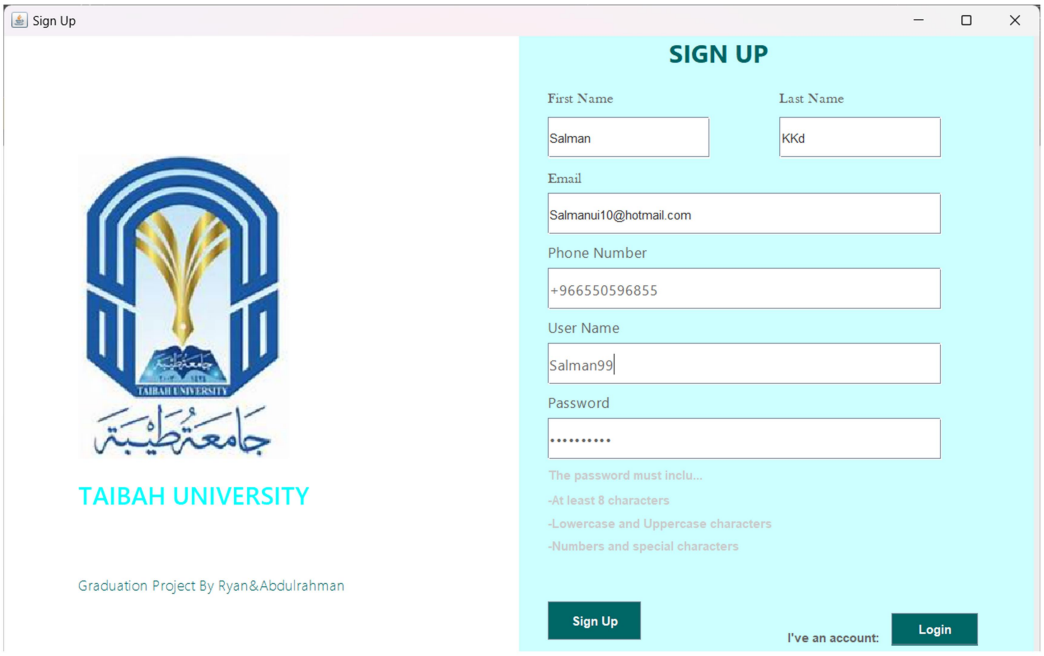


Figure 6-17: Test Case Number 4

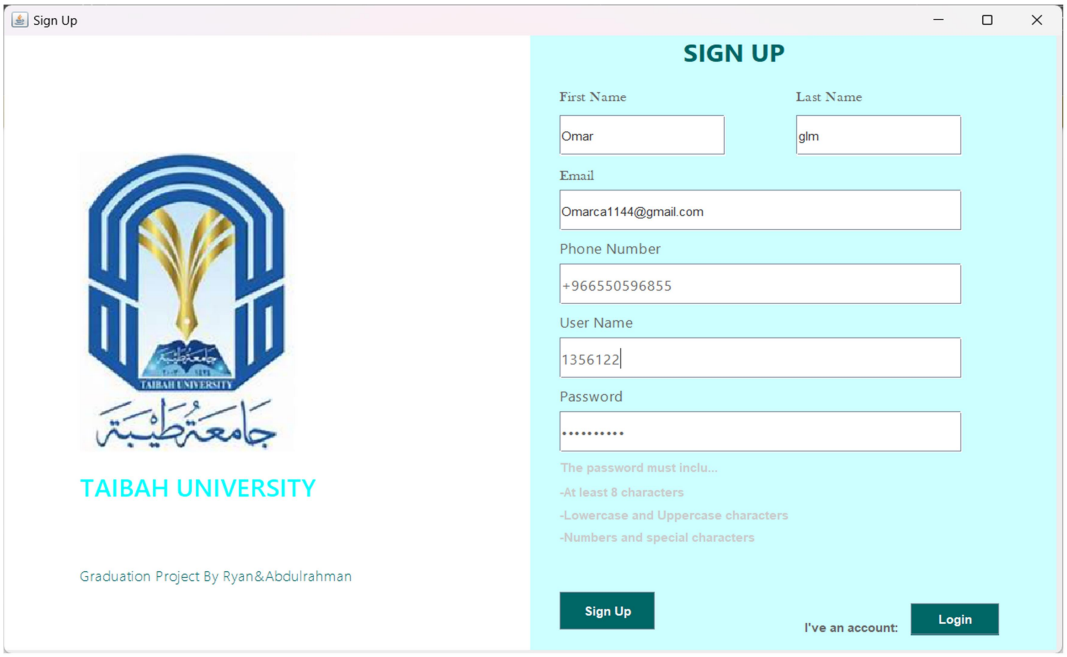


Figure 6-18: Test Case Number 5

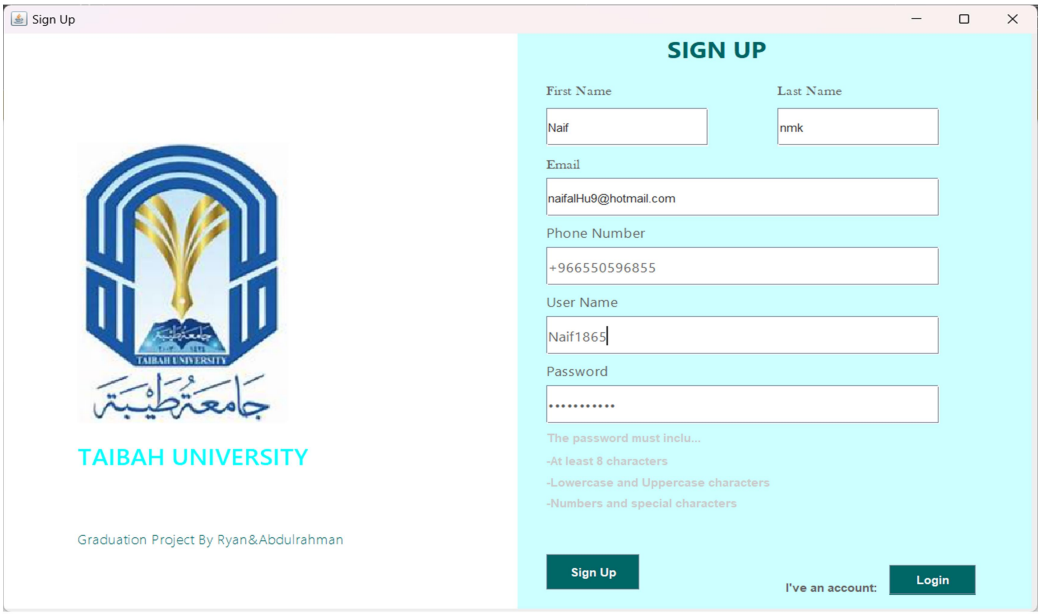


Figure 6-19: Test Case Number 6

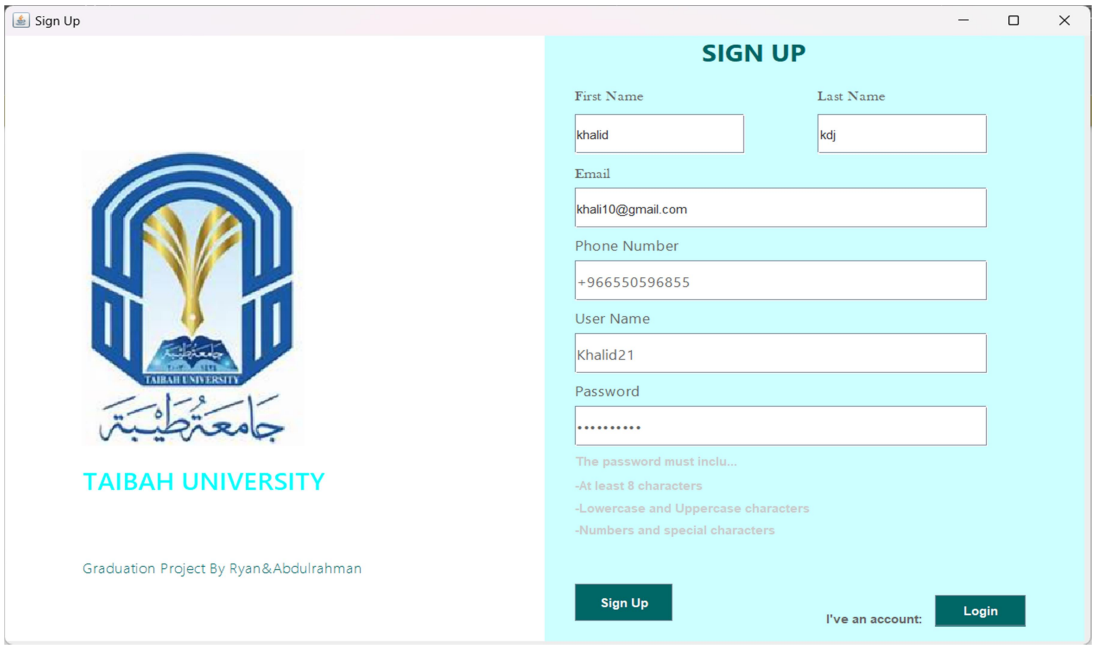


Figure 6-20: Test Case Number 7

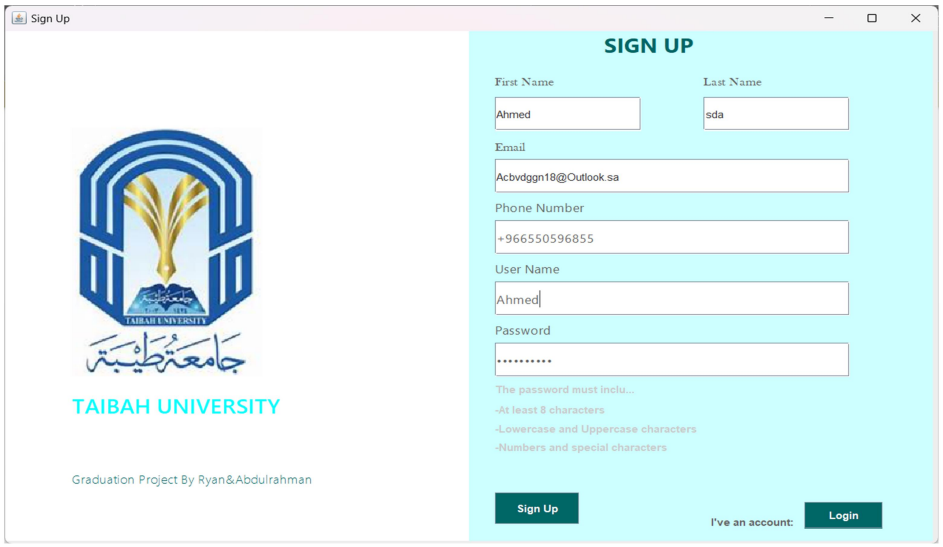


Figure 6-21 : Test Case Number 8

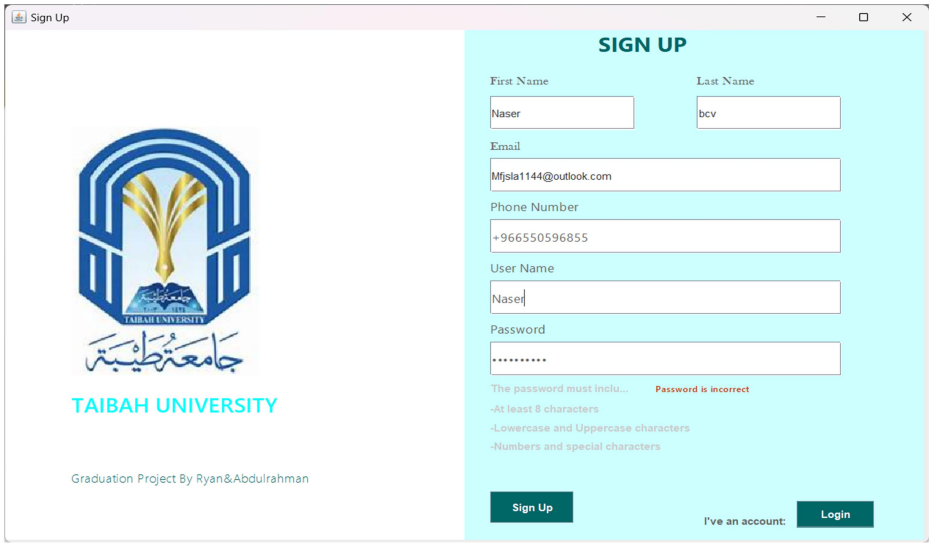


Figure 6-22 : Test Case Number 9

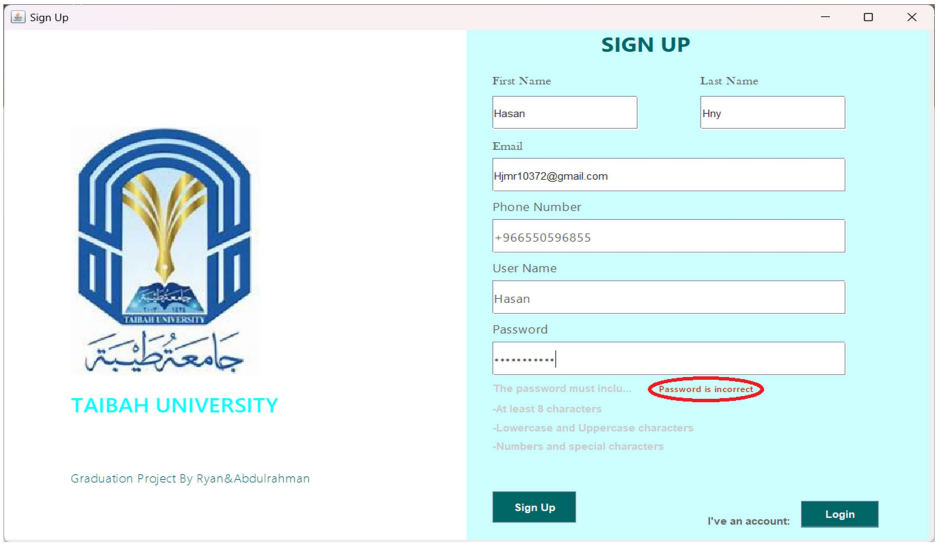


Figure 6-23 Test Case Number 10

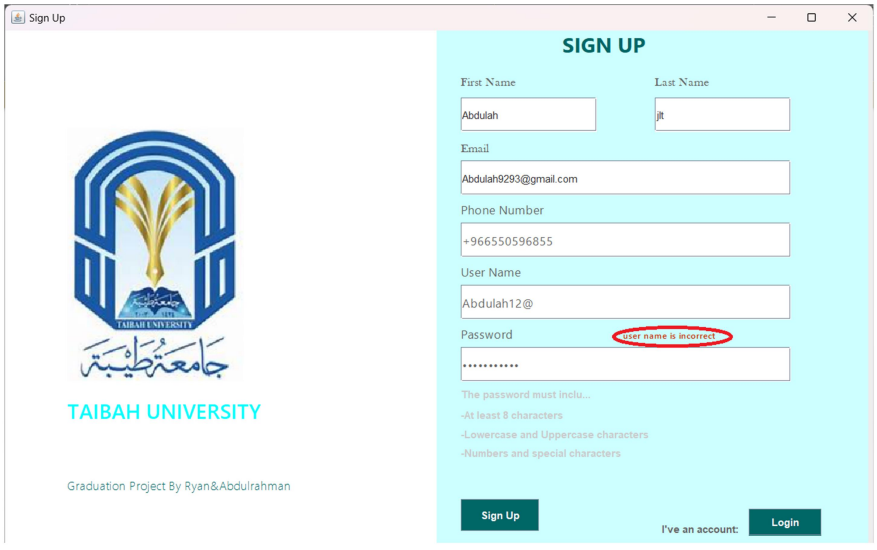


Figure 6-24 Test Case Number 11

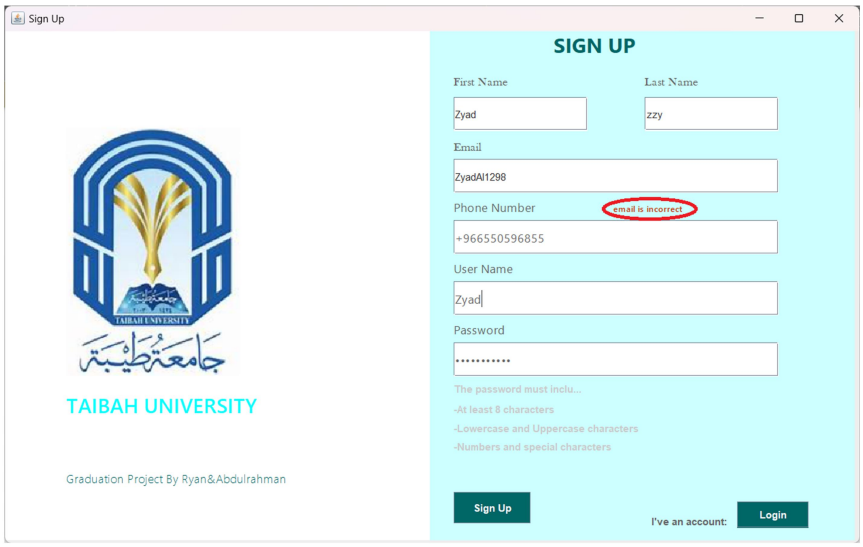


Figure 6-25 Test Case Number 12

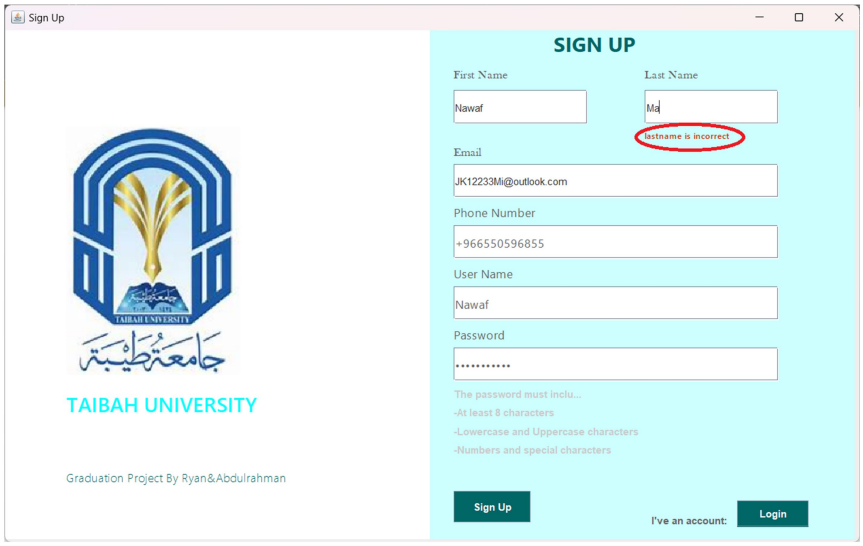


Figure 6-26 Test Case Number 13

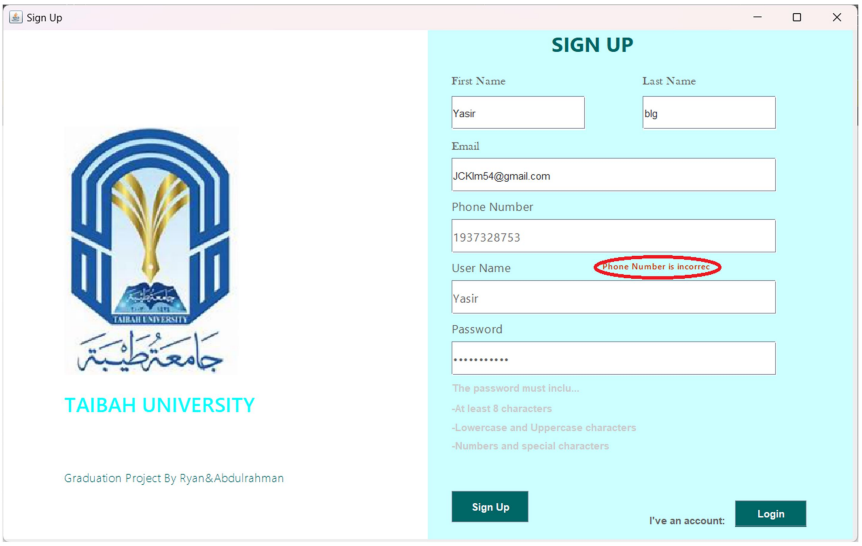


Figure 6-27 Test Case Number 14

6.3 Summary

In this chapter, the system is tested through three approaches of testing; unit testing, integration testing, and performance testing. According to the testing results, the model performed as expected.

Chapter 7: Conclusion and Future Work

7.1 Conclusion

For manipulating the vulnerabilities associated with using the password as a single-factor authentication method, our designed and implemented system considered two-factor authentication using the password comparison and time-based one-time password (TOTP) which is considered the most secure method to thwart unauthorized access, where the code is changed every 60 seconds.

The developed system consists of three main modules, sign up module, login, and TOTP generation module, in addition to auxiliary module for generating QR code for synchronizing the generated OTP between the application server and authenticator (on a mobile device). After successful registration, the QR code is generated regarding the generated OTP and the user can scan it using an authenticator application, such as Google Authenticator, for obtaining the synchronized OTP code. If the user enters the valid password for login, the system will forward him to the OTP verification page to enter the valid OTP code displayed on the authenticator application. Results of the conducted evaluation testing demonstrated the effectiveness of the developed system.

7.2 Future Work

In future, we may consider additional features to the system, by considering a third factor of authentication, biometric features, such as fingerprint, face recognition, hand geometry to reinforce the authentication mechanism.

References

[**GMA21**] G,ALI and M,Ally and A,Elikana. A Secure and Efficient Multi-Factor Authentication Algorithm for Mobile Money Applications. Journal: Future Internet, 2021. https://www.mdpi.com/1999-5903/13/12/299.

[**MZ22**] M,ARIF. and Z,SHUKUR. Device Identity-Based User Authentication on Electronic Payment System for Secure E-Wallet Apps. Electronics 2022

[**BP2019**] Be. Pavel Brousek. Multi-Factor Authentication in Large Scale. Brno, Spring 2019

[**HJCO20**] H,Kim and J,Han and C,Park and O,Yi Analysis of Vulnerabilities That Can Occur When Generating One-Time Password. Applied Sciences, 2020

[**QG15**] Alex Q. Chen and Weihan Goh. Two Factor Authentication made Easy. 2015

[**J17**] Costa , J.. 2FA2P2 : A Two Factor Authentication Scheme . Technical Report, University of Westminster. 2017.

[**K19**] Eman.K Focused on Two Factor Authentication Framework Using OTP – SMS Based on Blockchain. 2019

[**PPM15**] C. Pfleeger, S. Pfleeger. and J. Margulies. Security in Computing, Fifth Edition, 2015.