TWO-FACTOR HUMAN AUTHENTICATION

By

LIANG XIAN LIANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2018

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

_____

_____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_____ _____

(Author's signature) (Supervisor's signature)

Address:

_____

_____ _____

_____ Supervisor's name

Date: _____ **Date**: _____

TWO-FACTOR HUMAN AUTHENTICATION

By

LIANG XIAN LIANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled "Two-Factor Human Authentication" is my own work, except where indicated by referencing. The report has not been accepted for any degree and is not being submitted concurrently in candidature for my degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOLEDGEMENTS

I would like to express my gratitude to my final year project supervisor Dr Vasaki a/p Ponnusamy for her guidance and supervision throughout this project. She has guided me and given me a lot of advice from a security perspective point of view in the project development.

Finally, I would like to thank my parents who played a major part in my life for supporting me to pursue my studies.

ABSTRACT

A two-factor authentication system which verifies user using credentials from 3 factors. These 3 credentials are a password “Something you know”, a private key assigned to user in his mobile application “Something you have”, and fingerprint/s “Something you are”. People have already been using credentials from these 3 categories to proof their identity even before the days of the Internet. This application system combines these credentials from all the 3 categories in order to provide a secured way of authentication, while maintaining the simplicity for user to use. This system implements time-based one-time password, also known as the TOTP algorithm which is capable of constantly generating a unique string of code within a specified time step. The TOTP algorithm is implemented on both the mobile application (prover) and web application server (verifier). Each user will be assigned an unique key, and thus, capable of generating a unique TOTP within the time-step. The TOTP generated by the mobile application will be send to the web application server as two-factor authentication (2FA) via HTTPS connection. Besides, this system implements fingerprint authentication to secure the generation of the TOTP and instantly send the generated TOTP to the server on fingerprint authenticaton success. Thus, making it a 3 factors authentication system that can be done in 2 steps.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2FA	Two -Factor Authentication
2SV	Two-Step Verification
AAA	Authentication, Authorization and Accounting
CHAP	Challenge-handshake authentication protocol
EAP	Extensible Authentication Protocol
KDC	Key Distribution Centre
OTP	One-Time Password
IDE	Integrated Development Environment
APK	Android application package
ADB	Android Debug Bridge
SDK	Software Development Kit
JWT	JSON Web Token
TOTP	Time-based One-Time Password
TEE	Trusted Execution Environment
NTP	Network Time Protocol
HTTPS	Hypertext Transfer Protocol Secure
SQL	Structured Query Language
JPQL	Java Persistence Query Language
XSS	Cross-Site Scripting
MITM	Man-in-the-Middle Attack
SHA-256	Secure Hash Algorithm (Digest Size: 256)
AES	Advanced Encryption Standard

Chapter 1: Introduction

1.1 Authentication Background

Information asset is a piece of information owned by an individual or an organization which has a monetary value (Todorov, 2007, p.1). Authentication, authorization and accounting known as "AAA" is an architecture that has been in use for asset protection since before the days of the internet (Convery, 2007). These three processes combined can provide effective information assets management and security. In this project, we are focusing on authentication. Authentication is the process of identifying an individual to ensure that the individual is who he claimed to be. To ascertain that users are who they say they are, the operating system or the application requiring authentication requires users to provide evidence to prove themselves which is known as user credentials (Todorov, 2007, p.18).

Authentication credentials can be "something you know" like a password, "something you have" like an identity card, or "something you are" like a fingerprint. Today, information assets are stored all over the Internet and hence, Information Technology security plays a major role to protect the confidentiality, integrity and availability of these assets. Many network authentication protocols have been introduced to meet different requirements by the industry such as AAA architecture protocols, Challenge-handshake authentication protocol (CHAP), Extensible Authentication Protocol (EAP) and Kerberos.

Kerberos is a computer network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography ("Kerberos: The Network Authentication Protocol", 2015). In this protocol, authentication is granted through tickets generated. The client is required to get a ticket from the Key Distribution Centre (KDC) through authentication server and stores it locally so that they can present it to a file server to access data from the web application server.

Nowadays, a single-step authentication can be easily compromised, therefore a two-step verification scheme comes in. Two-step verification requires two credentials from the same category "Something you know", "Something you have", or "Something you are". Google, Sony PlayStation and Apple are implementing two-step verification (2SV) to add in extra security for their users. For example, Gmail requires its user to provide one-time password (OTP) sent to phone

after providing the user account password. OTP sent to the phone may appear to be “something you have”, but from a security perspective it is considered as “something you know” because the key to the authentication is not the device itself, but it is the information stored on the device (Henry, 2016). Figure 1.1 below shows a flow diagram to demonstrate the difference between traditional authentication, two-factor authentication, two-step verification.

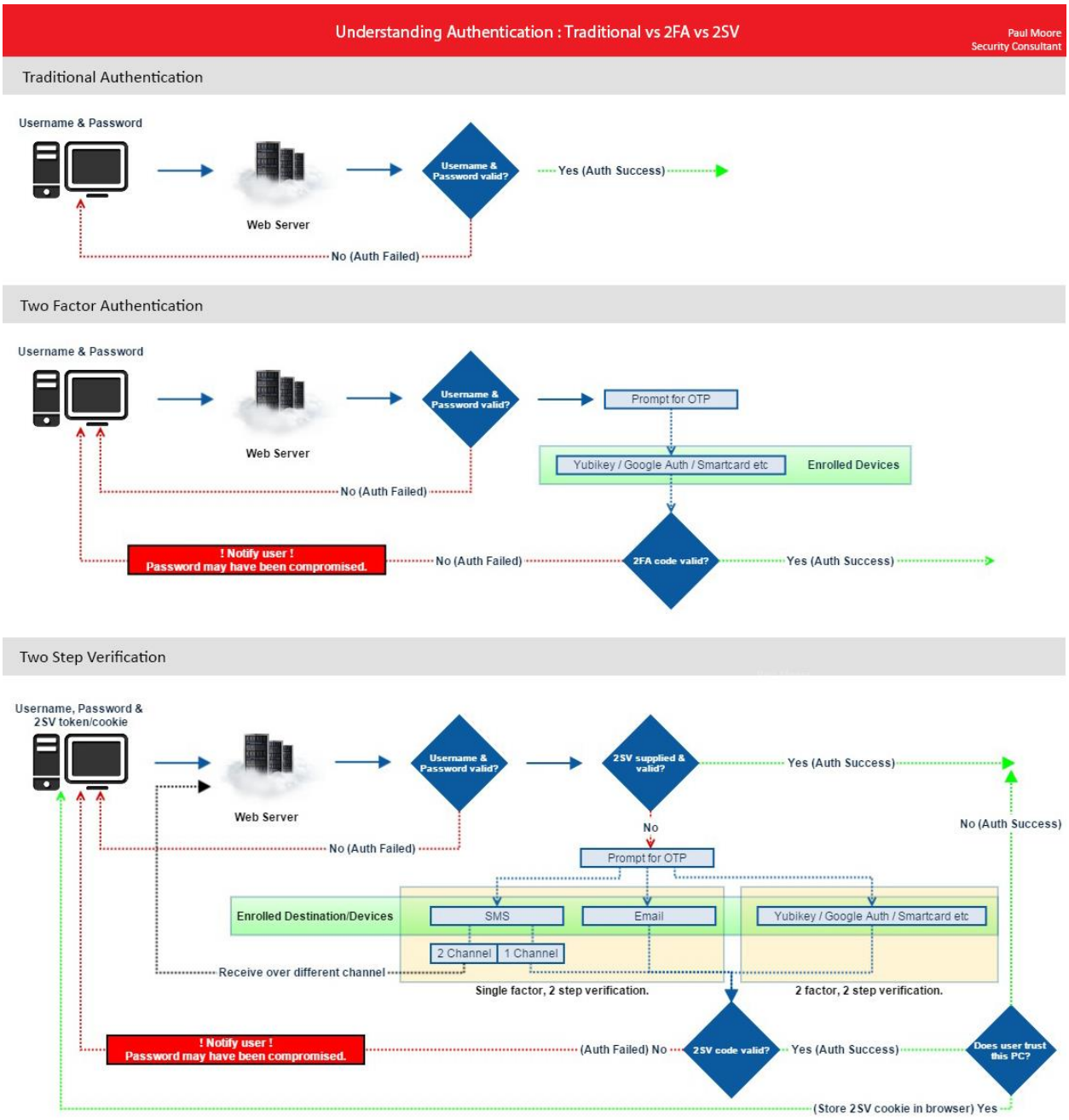


Figure 1.1: Understanding Authentication: Traditional vs 2FA vs 2SV (Moore, 2014)

1.2 Motivation and Problem Statement

Authentication using only “Something you know” is very vulnerable to social engineering. Social engineering in the context of information security is a technique that cybercriminals use to trick victims to divulge their confidential information.

For example, shoulder surfing is a type of social engineering that does not require technical skills, it is performable by anyone. Phishing is another type social engineering technique which tricks users usually through phishing mails or chats. It is hard to notice a phishing site once you get in there as it is almost identical to a legitimate one.

Basic authentication protocols using only passwords are also vulnerable to many other threats such as eavesdropping, keylogging, man-in-the-middle attacks, replay attack and dictionary attacks.

In multi-step authentication, more steps mean more security but it could also mean poor user experience. It could be monotonous and annoying to the users as they need to go through extra steps and effort in order to login. Let’s take two-step verification as an example. In the first step, the user is required to enter his password, then in the second step, the user is required to enter the one-time password (OTP). Although it is for security purpose but sometimes users might find it annoying especially when they are performing time-critical transaction.

1.3 Project Scope

This project focuses on delivering an improved version of a two-factor authentication system with minimal user effort. This system consists of an Android application that can perform 2FA by generating a unique authentication code, and an authentication server and also a simple website to perform some sensitive user actions such as user login. This project also involves a new algorithm design that can generate a time-sensitive unique code for security purpose.

1.4 Project Objective

The main objective of this project is to come out with a secured yet effortless two-factor authentication system.

This project mainly focuses on improving security of user login authentication for online banking. Although this project allows users to perform 2FA using their fingerprint, all kinds of fingerprint verification will not be covered in the project because Android phone stores the fingerprint template data in a secured storage and are not accessible other than the Android OS itself. However, Android 6.0 API offers new APIs for developers to authenticate users by using their fingerprint scans on supported devices, so we will make use of these APIs to provide a third layer security in this project.

This project aims to improve the ease of use of two-factor verification to increase users experience while implementing an extra layer of security. Instead of retyping the generated code, users are able to verify themselves on the apps designed with a touch of a button. For extra security purposes, each code can only be used once and also expires every 30 second.

1.5 Impact, Significance and Contribution

This project is applicable for all online authentication system especially online banking sites. By implementing this authentication system, many general cyber-attacks such as social engineering, brute-force and keyloggers can be prevented.

This project increases authentication security with sacrificing minimal user experience. Unlike the other existing system which requires a lot of retyping of passcode such as OTP, this project allows user to perform 2FA with a touch of a finger.

Chapter 2: Literature Review

Two-factor authentication, also known as 2FA is a method of authentication using two credentials from at least two of the following categories:

- Knowledge factors ("Something you know") such as a password,
- Possession factors ("Something you have") such as a device,
- Inherence factors ("Something you are") such as a fingerprint.

A typical 2FA uses the combination of "Something you know" and "Something you have" or "Something you know" and "Something you are". Two-factor authentication is widely implemented in current practice to resolve the problems of traditional single factor authentication and two-step authentication. This is because two-factor authentication requires two different factors of credentials and therefore, it can protect an account even when the password is compromised as a physical or biometric component is required along with the password in order to log in successfully.

There are limitations in two-factor authentication. One of the most common limitation is the factors can get lost. Here are some examples:

- You can forget your password
- Your 2FA registered device can be stolen
- A second-degree burn can deform the pattern of your fingerprint

In this paper, four examples of the existing authentication system are chosen to be reviewed.

2.1 Steam Guard Mobile Authenticator

Steam Mobile App developed by Valve Corporation has a feature called Steam Guard Mobile Authentication. Steam Guard Mobile Authentication implements two-factor authentication with a combination of "Something you know" and "Something you have". Only one authenticator can be activated on one account at a time ("Steam Guard Mobile Authenticator", Valve Corporation). When a steam user has this feature enabled on their phone, this user is required to enter the generated code after he has entered his username and password. Hence, the first factor of authentication will be "Something you know" which is the password while the second factor will be "Something you have" which is the phone which generates the Steam Guard code. The authenticator generates a unique code which expires in 30 seconds. A new code will be generated after the old one expires and a code can be used only once ("Steam Guard Mobile Authenticator", Valve Corporation). The screen capture is shown below in figure 2.2.1.

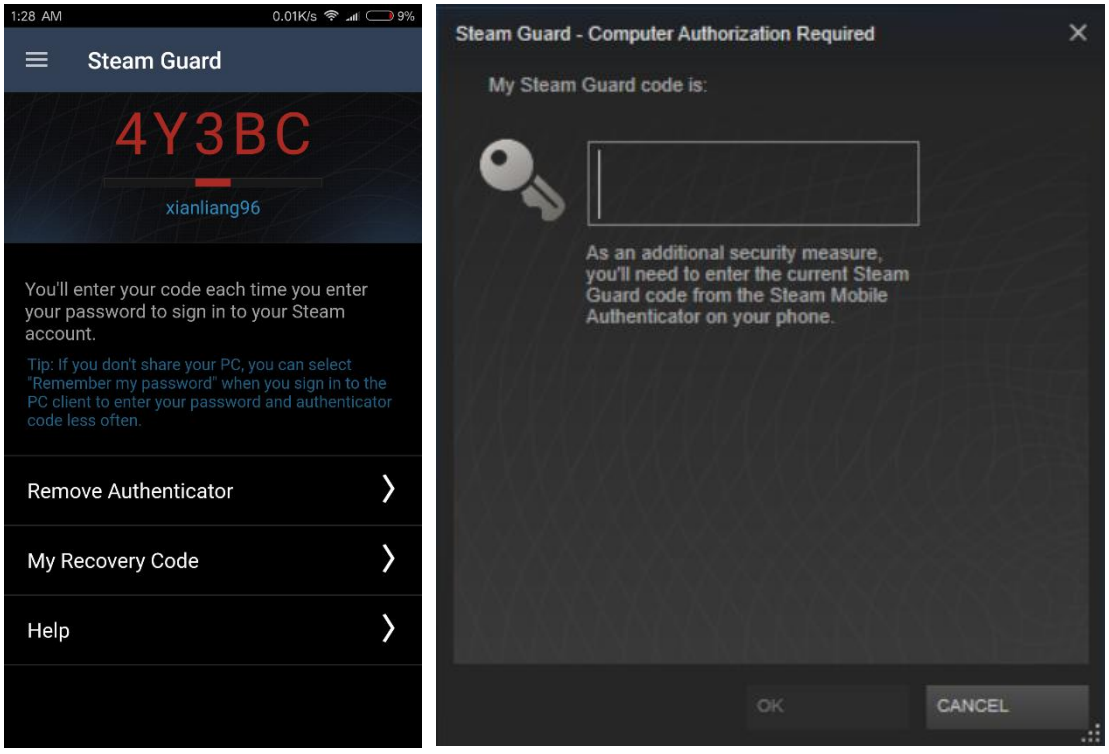


Figure 2.2.1: (left) Screen capture of Steam Guard Mobile Authenticator generates a code. (right) Screen capture of computer authorization request for Steam Guard code. (Valve Corporation, 2017)

Strength

- The secret key for the code generation is stored in the phone itself and uses date and time variables to generate the code. Hence, Steam Guard Authentication code can be generated without Internet connection.
- Steam Guard Mobile Authenticator Code has a short life span of 30 seconds.
- Valve has implemented safeguards against brute force. An account will be temporary blocked for an hour after 20 consecutive login failures, a screen capture is shown in figure 2.2.2 below. Every code generated is 5 characters long numerical and uppercase alphabetical characters which means that there are 36^5 different combinations and it expires in 30 seconds. A hacker can easily brute-force a 36^5 possible combinations code within 30 seconds with the help of high-performance computer but with the Steam's safeguards enabled, it is nearly impossible to do so. Based on probability, we can calculate that a hacker has only a 0.00000033 percent chance to crack every single code.

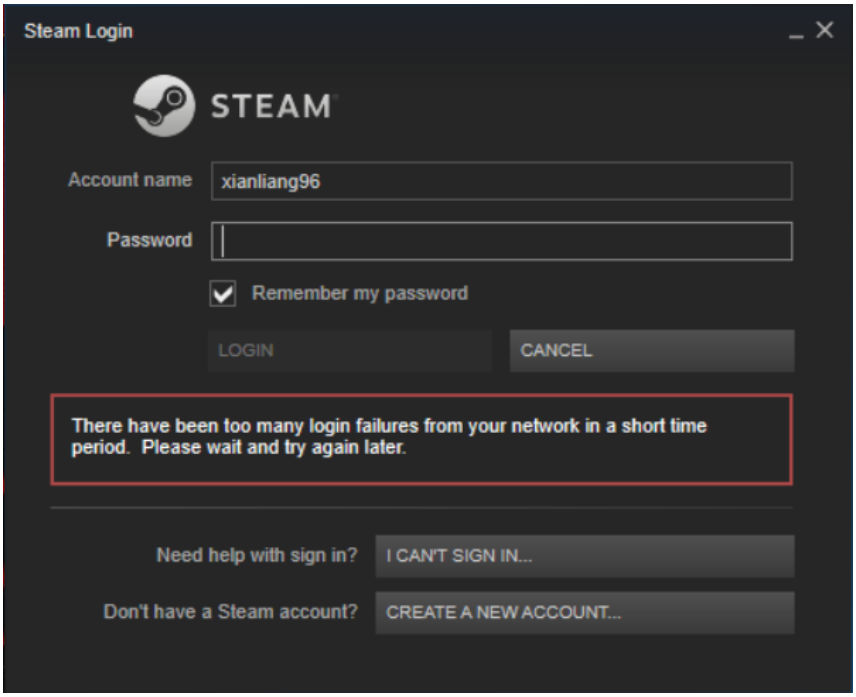


Figure 2.2.2: Screen capture temporary blocked account after too many login failures. (Valve Corporation, 2017)

Weakness/limitations

- Users are required to type in the Steam Guard Code generated in the Mobile Authenticator which is a tedious process.
- It requires the clocks of the mobile device where the Steam Guard Mobile Authenticator is installed and the authentication server set to approximately the same time.

Solutions to the weakness/limitation

- Make the app to be able to send the Steam Guard Mobile Authenticator Code to the server without needing the user to retype in the code in the Steam Client.

2.2 Steam Guard Email Code

Before Steam introduced the Steam Guard Mobile Authenticator, the Steam Guard email is used to authenticated unrecognized device by providing special access code sent to the registered email in order to verify it's owner. This approach is still being used as an alternative to Steam Guard Mobile Authenticator. This code will be sent to the registered email of the user account and the user are required to retrieve this code from his mail box and provide this code as 2FA.

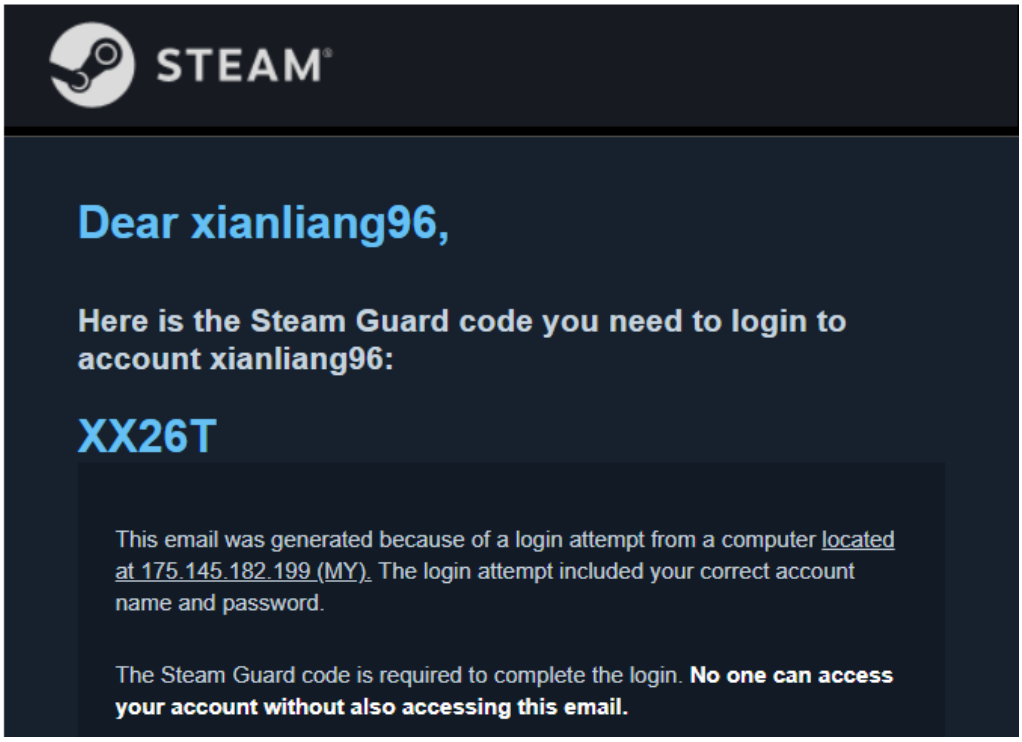


Figure 2.2: Screen capture of Steam Guard Email Code. (Valve Corporation, 2018)

Strength

- Unlike other approaches which requires attaching to a device, this approach does not require user to have any extra device.
- Besides, this approach is the simplest to implement without developing any extra software or hardware. Thus, a lot of development time and cost can be saved.

Weakness/limitations

- User has to first login to the registered email account in order to retrieve the Steam Guard Email Code. Then, they are required to provide the code retrieved by typing it in manually or copy and paste. This process is so long and tedious giving the user a lot of hassle.
- Although an email account seemed to be "Something you have", it is actually protected by only "Something you know". Therefore, you can access to an email account by knowing its email address and password without actually possessing anything.
- According to a new survey of 1,000 respondents by Keeper Security, more than 80 percent of folks ages 18 and up reuse the same password across multiple accounts (Keeper, 2017), which means users are very likely to use the same password for their email account. Therefore, hacker might have access to the victim's registered email if he acquires the victim's account password at the first place.

Solution to the weakness/limitations

- As a solution, credentials from other factor has to be used, e.g.: Inherence factors, Possession factors.

2.3 Public Bank eCommerce Purchase One-Time Password (OTP)

Public Bank eCommerce Purchase One-Time Password (OTP) allows user to perform secure payments to online merchants through Public Bank eCommerce, upon making an online purchase through PB Credit/Debit Card, user will receive a verification SMS to his registered phone number containing the OTP (Public Bank Berhad, 2018). The user then are required to provide this code in order to complete the online payment he made.

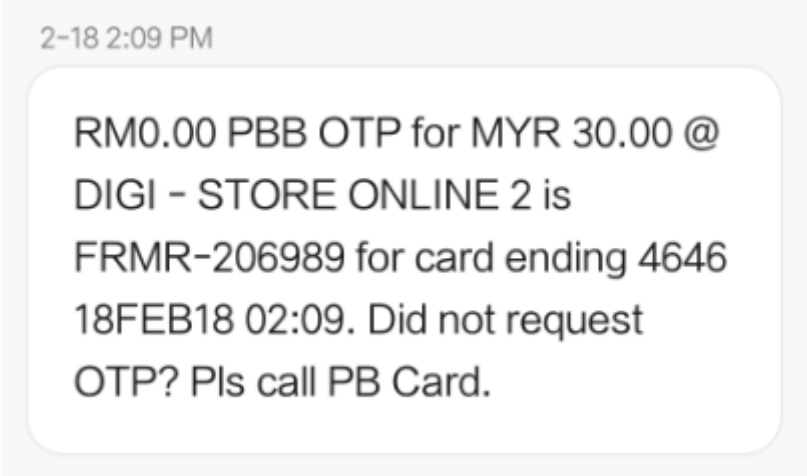


Figure 2.3: Screen capture of Public Bank eCommerce Purchase One-Time Password (Public Bank Berhad, 2018)

Strength

- SMS is secured because the messages are encrypted by network providers
- SIM Card is secured because it contains two secret codes which are IMSI (International Mobile Subscriber Identity) and Ki (Authentication Key), making it unique and difficult to clone without actually acquiring the physical SIM Card.
- This approach can also save a lot development time and cost as it does not require any extra software or hardware to be developed since SMS service can be outsourced.

Weakness/limitations

- Difficult to look for the OTP in the message of the SMS
- Heavily rely on mobile network carrier, user will not be able to receive the OTP if the mobile network carrier is having an outage or service down
- Security level is also depended on the mobile network carrier
- User in certain area with poor network coverage will not be able to receive the OTP as well.
- Users are required to type in the OTP they received which is a tedious process
- The SMS might take some time to arrive. In the worst-case scenario, the 2FA might have been timed out before the SMS arrives.

Solution to the weakness/limitations

- The text message can be formatted to increase the readability of the message so the OTP can be found easily within the text message. However, this approach still heavily relies on 3rd party mobile network carrier has to sacrifice user living in area with poor mobile network coverage.

2.4 HSBC Security Device

HSBC Security Device is able to generate a dynamic, time-sensitive Security Code. The Security Code changes constantly in a fixed duration. Every time user logon to HSBC'cs Personal/Business Internet Banking service, they are required to provide this Security Code as 2FA.

Figure 2.4: HSBC Security Device (HSBC, 2018)

Strength

- Each generated security code has a short life span.
- Easy to use, able to generate the security code at a touch of a button.

Weakness/limitations

- Users are required to type in the security code generated by the Security Device which is a slow and tedious process.
- Users are required to carry the device to every place where they want to use the HSBC Internet Banking service.

Proposed solution(s) to the weakness/limitation

- Develop it to a mobile application since nowadays most people carry their mobile app to anywhere

2.5 Literature Review Summary Table

#	System name	Category	Strength	Weakness	Proposed solution
1	Steam Guard Mobile Authenticator	Software based token	• Generate constantly changing verification code • Safe guard against brute force • Work offline	• Tedious retyping • Client time must be in sync with server time	Send the TOTP across network without the need to retype
2	Steam Guard Email Code	Email based token	• Not attached to any device • Save development time and cost	• Tedious retyping • Not secured	Use credentials from other factors
3	Public Bank eCommerce Purchase One-Time Password (OTP)	SMS based token	• Secured as they are encrypted by network providers • Save development time and cost	• Rely on mobile network carrier • Difficult to find OTP • Tedious retyping	Format the SMS content to increase readability
4	HSBC Security Device	Hardware based token	• Generate constantly changing verification code • Easy to be use	• Tedious retyping • Need to be carried to everywhere	Develop it to a mobile application

Table 2.5: Literature Review Summary Table

Chapter 3: System Design

3.1 Design Specifications

In this project, we are developing an improved version of two-factor authentication system which can be grouped into three main parts:

- A mobile application
- A web application server
- A simple website

Android is selected as the development platform for the mobile application because of its low barrier of entry which meets the budget of this project. This project targets Android 6.0 Marshmallow (API level 23) to implement fingerprint authentication which is provided in the new APIs release. This project is developed with Android Studio, the official IDE for Google's Android operating system. During the development phase, the app is tested on Samsung Galaxy S6 SM-G920F via Android Debug Bridge (adb). At the end of this project, we will generate a self-signed apk of this project for distribution and installation to other supported phones.

Apache TomEE web application server is chosen as the back-end of this project. Apache TomEE supports many Java Enterprise Edition (Java EE) technologies which are extremely useful for this project. The Java EE technologies that we applied are Java Servlets, Java Server Pages (JSP), Java Persistence API (JPA), Java Contexts and Dependency Injection (CDI), and Enterprise JavaBeans. On the other hand, this project will be using MySQL as its database.

A simple website is created for demonstration purposes. This website is used to perform some user actions which will trigger 2FA request to the registered mobile device. This website is created using HTML, CSS, JavaScript, jQuery and Ajax.

3.2 Fingerprint Authentication Module

Most of Android device that come with Android 6.0 Marshmallow or above have fingerprint sensors embedded on it which we wanted to integrate in this project.

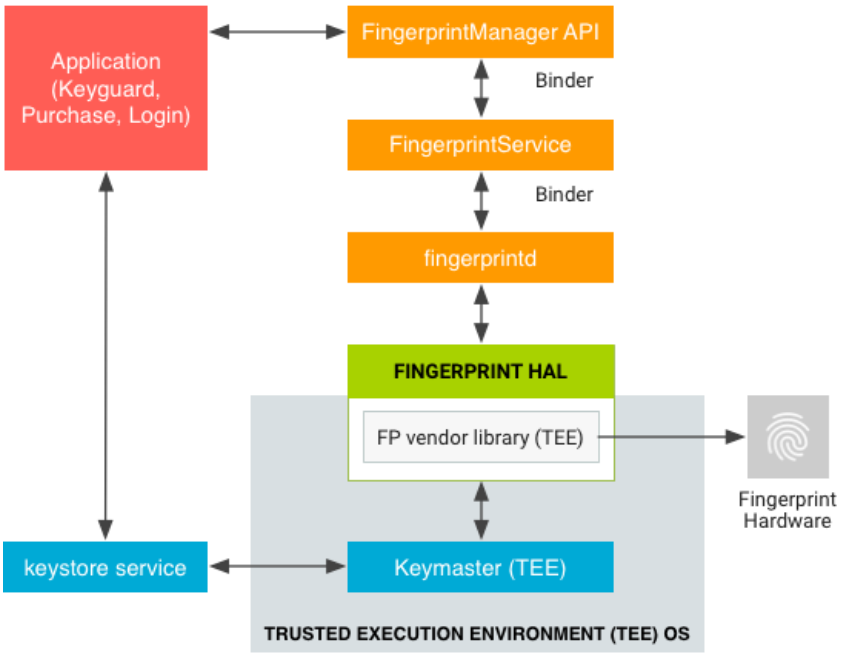


Diagram 3.2.1: Fingerprint Hardware Abstraction Layer Diagram (Android, 2018)

Android uses the Fingerprint Hardware Abstraction Layer (HAL) to connect to a vendor-specific library and fingerprint hardware as shown in the diagram above. Unfortunately, according to android documentation (Android, 2018):

"A vendor-specific HAL implementation must use the communication protocol required by a TEE. Thus, raw images and processed fingerprint features must not be passed in untrusted memory. All such biometric data needs to be secured within sensor hardware or trusted memory. (Memory inside the TEE is considered trusted; memory outside the TEE is considered untrusted.)"

This means that we are not able to acquire raw images from the fingerprint scanner to authenticate user with their fingerprints in our web application server. Fortunately, Android 6.0 offers fingerprint authentication APIs for app developers, we are still able to use these APIs provided to authenticate user as the device owner without accessing the raw fingerprint images.

3.3 TOTP: Time-Based One-Time Password Algorithm

As we mentioned earlier, fingerprint data cannot leave the Trusted Execution Environment (TEE), therefore, users are required to provide other credentials to the server to perform 2FA. This is the part where the TOTP algorithm documented in RFC-6238 plays an essential role to this 2FA system.

Basically, TOTP is defined as TOTP = HOTP(K, T), where K represents the shared secret key and T represents the number of time steps X between the initial counter time T0 and the current Unix time. In other words, T = (Current Unix Time - T0) / X.

In this project development, we have taken the following requirements into account:

1. We sync our server (verifier) clock with Network Time Protocol (NTP) server since Android device (prover) system clock are synced to NTP by default. This enables them to derive the current Unix time.
2. On user registration, the server (verifier) randomly generates a unique secret key and assign it to the user.
3. When user login to the mobile app, his mobile device (prover) will receive his unique secret key from the server (verifier) through a HTTPS connection.
4. This secret key received in his device is encrypted and safely stored in the Android keystore to prevent unauthorized access and usage.
5. We implemented HOTP which is documented in RFC4226 as the key building block in this algorithm.

We set the server (prover) and the mobile app (verifier) to have the same time-step value X of 30 seconds so that the shared secret (TOTP) generated by both of them have the exact same lifetime.

3.4 HAuth: Mobile Application

In this project, we combine fingerprint authentication together with TOTP in our mobile application. More specifically, this application requires user fingerprint authentication to generate a TOTP to be sent to the server as 2FA.

In order to implement fingerprint authentication in our app, we must firstly add USE_FINGERPRINT permission in to our app's manifest file. To make this module even simpler, a FingerprintHandler class extending FingerprintManager.AuthenticationCallback was created to listen to user fingerprint authentication. This class will delegate back to the main thread onAuthenticationSucceeded() or onAuthenticationFailed().

Every time we call the FingerprintManager.authenticate() method, we can parse along a CryptoObject. KeyGenerator was used to generate a symmetric key with the following specification - AES/CBC/PKCS7Padding in order to encrypt the CryptoObject. In the KeyGenParameterSpec, we also specify that setUserAuthenticationRequired(true) so that this key is authorized to be used only if the user has been authenticated. The main purpose that we parse in a CryptoObject into the FingerprintManager.authenticate() method, is that to provide an extra security measurement since it invalidates itself every time user gets authenticated which also mean that the same key cannot to perform two consecutives authentication. In order to do so, a new key is required to be generated and bind to the CryptoObject.

Once the user authenticated by scanning his fingerprint, this app will retrieve and decrypt the user's secret key from the keystore. This secret key is then used to generate a TOTP to be sent to the sever as 2FA. Our web application server will then verify and authenticates the user if the TOTP he provided is valid.

3.5 Web application server

Apache TomEE was chosen as our back-end. The Java EE technologies that we implemented are Java Servlets, Java Server Pages (JSP), Java Persistence API (JPA), Java Contexts and Dependency Injection (CDI), and Enterprise JavaBeans.

Among these Java EE technologies, the one that we would like to highlight is the Java Persistence API (JPA), it is an ORM solution that is a part of the Java EE framework. We use it to helps us manage relational data in our server. JPA allows the use of native SQL and defines its own query language, JPQL (Java Persistence Query Language). Although it is not bulletproof to SQL/JPQL injection, by practicing some simple security measurements, we can easily mitigate this attack.

On the other hand, we configured our server to support SSL. Using Java Keytool, we are able to generate a PKCS12 file which contains a private key and a X.509 certificate. We also manage to get this certificate signed by a Trusted Certificate Authority (CA). Then, we configure our server's SSL/TLS Connector to use this SSL certificate. By enabling SSL, we can ensure privacy and data integrity when communicating with our clients.

For our clients' session management, we chose JSON Web Tokens (JWT). JWT enables the implementation of token-based/stateless authentication. In stateless authentication, the server-side does not need maintain the state of a user, which can reduce the work load of our server. JWT can be stored in client's localStorage, sessionStorage or Cookie Storage. Abbott, T (2016) mentioned, "Cookies, when used with the HttpOnly cookie flag, are not accessible through JavaScript, and are immune to XSS. You can also set the Secure cookie flag to guarantee the cookie is only sent over HTTPS." Based on the statement above, we come to a decision to store JWT in our client's cookies. In our web application server deployment descriptor, web.xml, we also set the HttpOnly cookie flag and the Secure cookie flag to true. By implementing HttpOnly and Secure flagged cookie together with SSL, we are able to prevent session hijacking like XSS and MITM.

3.6 JSON Web Tokens

A successful login by providing username and password will gives user a token with level 1 access. In order to receive a level 2 token, user has to perform 2FA on his registered mobile app or manually key in the TOTP generated by the mobile app.

Every time user makes a request to a secured web servlet in our web application server, the JWT stored in the cookie will be appended in the request header. The filters in our web application server will verify the token's validity and level of access. Requests with invalid JWTs or insufficient level of access will be declined by our server and redirected to our login page.

The following table shows an example of encoded level-1 JWT and level-2 JWT strings and their respective decoded header and payload:

JWT with Level-1 Access		**JWT with Level -2 Access**	
JWT String		**JWT String**	
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsaWFuZyIsInJvbGUiOiJ1c2VyIiwiY2xpZW50Ijoid2ViIiwibGV2ZWwiOjEsImV4cCI6MTU2NTU1NjU4NywiaWF0IjoxNTM0NDUyNTg3fQ.8x4dp6d2o5c89L7hz81ClHWKrWBrIZ9DtwtrcpFkkIM		eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsaWFuZyIsInJvbGUiOiJ1c2VyIiwiY2xpZW50Ijoid2ViIiwibGV2ZWwiOjIsImV4cCI6MTU2NTU1NjYyMSwiaWF0IjoxNTM0NDUyNjIxfQ.PbVQA5P48y9mcRyihtzopdk-h6UZ7814mgF1AzXnZJE	
Header	**Payload**	**Header**	**Payload**

{
"typ": "JWT",
"alg": "HS256"
}
	{
	"sub": "liang",
	"role": "user",
	"client": "web",
	"level": 1,
	"exp": 1534456193,
	"iat": 1534452587,
	"jti": "9bf703de-ebbf-49d5-89be-098e8e9e2a8c"
	}
		{
		"typ": "JWT",
		"alg": "HS256"
		}
			{
			"sub": "liang",
			"role": "user",
			"client": "web",
			"level": 2,
			"exp": 1534456227,
			"iat": 1534452621,
			"jti": "4c01e7f9-9d1b-4a3c-b51d-c9d340080b74"
			}

Table 3.6.1: JWT Web Level-1 vs Level-2

Why do we need JWT with different access level? Every time a user performs a successful login, he will automatically make a request listening to the server via a long-lived connection, HTTP long polling so that we provide user a real time response together with authentication when user completes his 2FA on his mobile device. We also opt user to make a request sending a TOTP generated from the mobile app on his phone that he manually typed in case he does have a working internet connection on his mobile phone. However, these requests are not safe because one can bypass the login stage (1st factor) directly to waiting for 2FA stage (2nd fator) using them. Therefore, we need to make sure that people that make these requests were already logged in and had provided their username and password previously. Hence, as a solution to this and the answer to the question above, we'll give users a level-1 token when they provide the correct 1st factor credentials and we'll require them to pass along this token when making the request to receive the level-2 token. In other words, we can also say that user with level-1 token proofs that he is one-factor authenticated and user with level-2 token proofs that he is two-factor authenticated.

Nevertheless, we also used a similar token for our mobile app. Only that, we changed the value for our client key from "web" to "mobile". Here's an example of an encoded mobile app JWT string together with its decoded header and payload:

JWT for Mobile App

JWT String	
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsaWFuZyIsInJvbGUiOiJ1c2VyIiwiY2xpZW50Ijoid2ViIiwibGV2ZWwiOjEsImV4cCI6MTU2NTU1NjU4NywiaWF0IjoxNTM0NDUyNTg3fQ.8x4dp6d2o5c89L7hz81ClHWKrWBrIZ9DtwtrcpFkkIM	
Header	**Payload**

{
"typ": "JWT",
"alg": "HS256"
}
	{
	"sub": "liang",
	"role": "user",
	"client": "web",
	"level": 1,
	"exp": 1534456193,
	"iat": 1534452587,
	"jti": "9bf703de-ebbf-49d5-89be-098e8e9e2a8c"
	}

Table 3.6.2: JWT Mobile

3.7 MySQL Database

Highlights:

- Password are hashed in SHA-256 to ensure that in case if our database is compromised, we do not expose our user's password.
- To prevent brute force attacks, the number of consecutives 2FA failure attempts is recorded. User account will be suspended when the number of consecutives 2FA failure reaches 5.
- An IPv4 — location lookup table which was retrieved from www.ip2location.com is being used to notify user if there are any suspicious user activity at unusual location.

3.7.1 Entity Relationship Diagram (ERD)

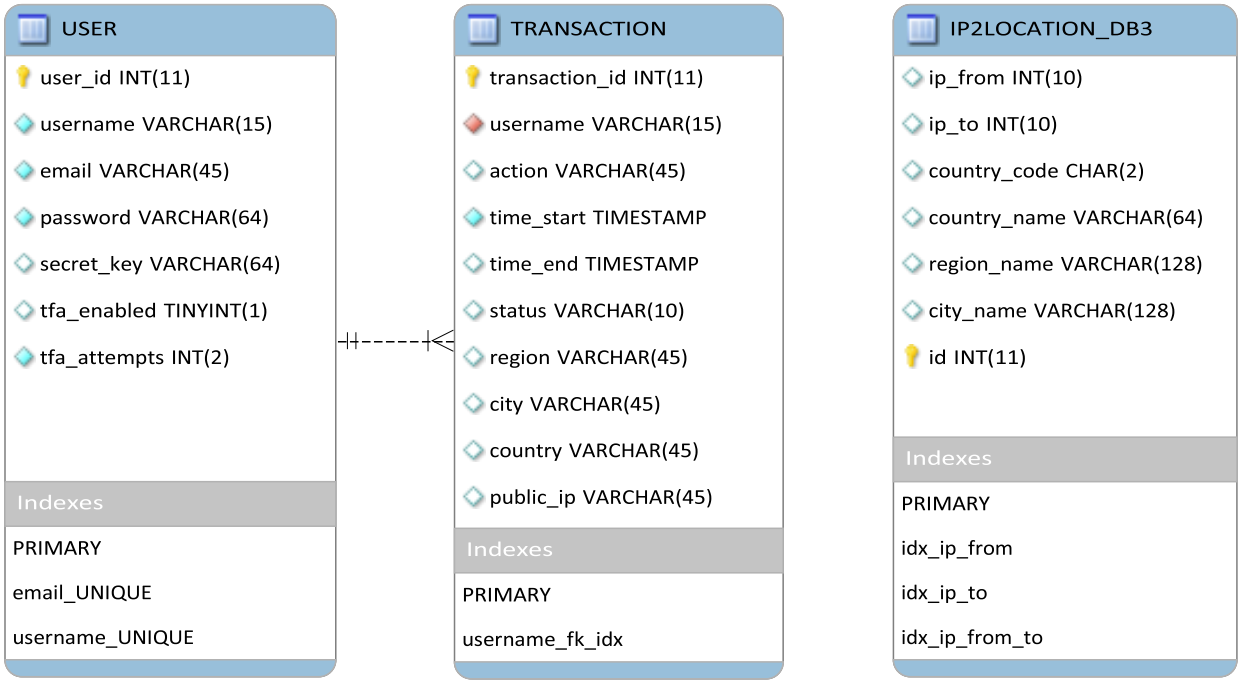


Diagram 3.7.1 Entity Relationship Diagram (ERD)

3.7.2 Data Dictionary

Legend:

(1) Notation for Data Type: X represents for VARCHAR
(2) Primary Key, Foreign Key, Foreign Key (Cascade) etc., are shown at the end of each table in the following sections

1) USER

This table stores the user account information

Column	Data Type	Description/Remark
USER_ID	INT(11)	Automatically incremented user ID.
USERNAME	X(15)	Username.
EMAIL	X(45)	Email address.
PASSWORD	X(64)	SHA-256 hashed password.
SECRET_KEY	N(64)	Unique secret key assigned to user.
TFA_ENABLED	TINYINT(1)	Status for whether the 2-factor authentication is enabled.
TFA_ATTEMPTS	INT(2)	Number of consecutives unsuccessful 2-factor authentication attempts.

Key	Column	Table
Primary	**USER_ID**	
Foreign		
Index	**EMAIL, USERNAME**	

Table 3.7.2.1: Data Dictionary of User Table

2) TRANSACTION

This table stores the information activity that user performed.

Column	Data Type	Description/Remark
TRANSACTION_ID	INT(11)	Automatically incremented transaction ID.
USERNAME	X(15)	Username of the user who made the transaction.
ACTION	X(45)	Action performed by user.
TIME_START	TIMESTAMP	The time user performed the action.
TIME_START	TIMESTAMP	The time user completes the action by 2FA.
STATUS	X(10)	The status of the action.
CITY	X(45)	The city where user perform the action.
REGION	X(45)	The region where user perform the action.
COUNTRY	X(45)	The country where user perform the action.
PUBLIC_IP	X(45)	Public IP address of user.

Key	Column	Table
Primary	**USER_ID**	
Foreign	**USERNAME**	**USER**
Index	**USERNAME**	

Table 3.7.2.2: Data Dictionary of Transaction Table

3) IP2LOCATION_DB3

This table stores IPv4 to location lookup table retrieved from www.ip2location.com (Version: August 2018)

Column	Data Type	Description/Remark
ID	INT(11)	Netblock ID
IP_FROM	INT(10)	First IP address in netblock.
IP_TO	INT(10)	Last IP address in netblock.
COUNTRY_CODE	CHAR(2)	Two-character country code based on ISO 3166.
COUNTRY_NAME	X(64)	Country name based on ISO 3166.
REGION_NAME	X(128)	Region or state name.
CITY_NAME	X(128)	City name.

Key	Column	Table
Primary	ID	
Foreign		
Index	**IP_FROM, IP_TO**	

Table 3.7.2.3: Data Dictionary of Ip2Location Table

3.8 System Design / Overview

In this section, an overview of the system design will be described. The topics covered are system flow diagram, activity flow event and UML diagrams such as activity diagram, class diagram as well as use-case diagram.

3.8.1 System Flow Diagram

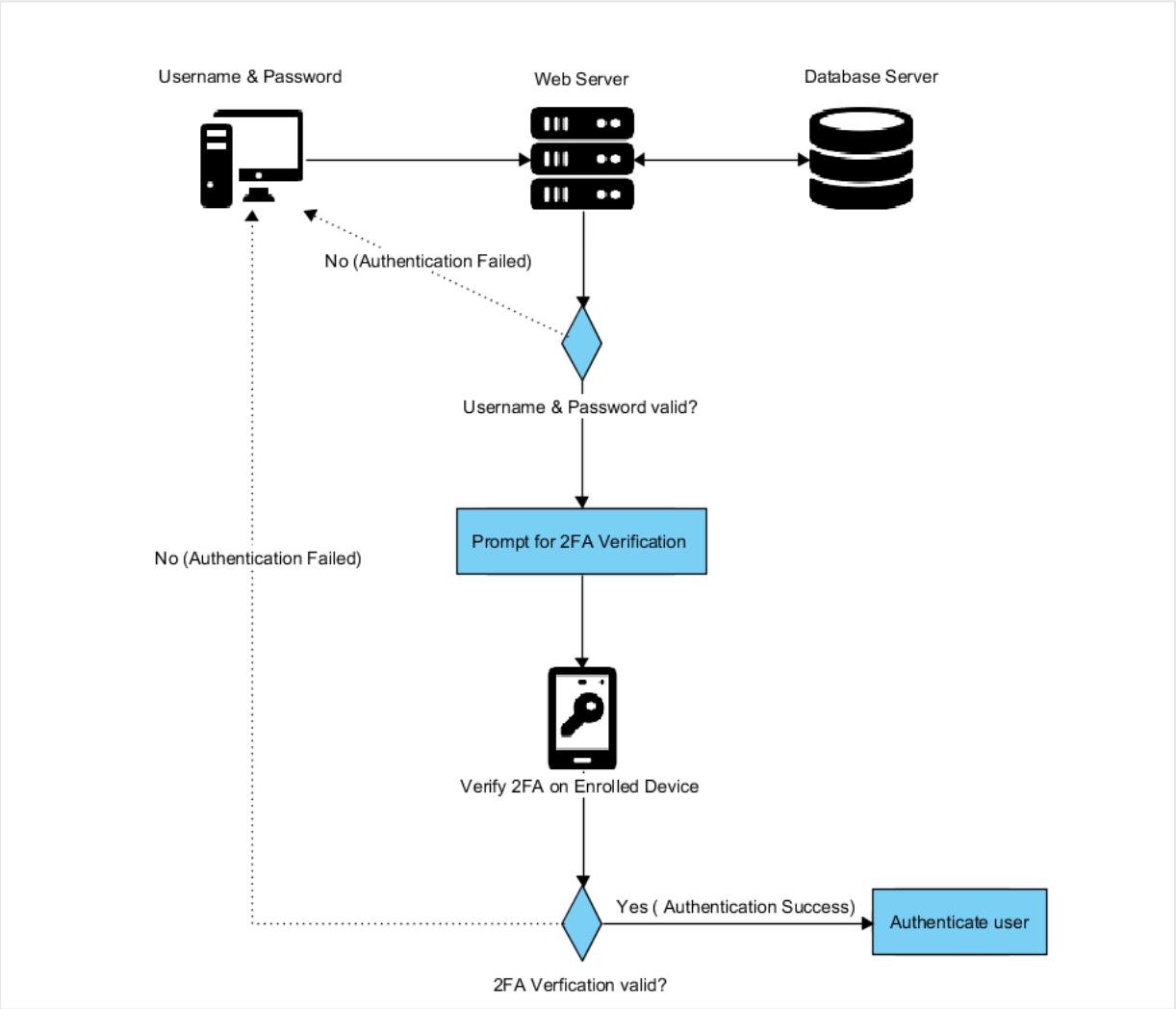


Diagram 3.8.1: System Flow Diagram

3.8.2 Activity Flow Event

1. User sends his username and password as parameters to the web application server in an HTTP POST request.
2. Web application server validates the request by comparing the parameters with the database server
3. The web application server logs this record in the database.
4. Web application server sends user a level-1 JWT in the response and prompts user for 2FA in his web browser.
5. The user registered dynamic mobile app retrieves the 2FA request together with the action description.
6. User places his fingerprint on the fingerprint scanner of his mobile device.
7. The app generates a TOTP using a private key assigned only to the user, time and date from the server.
8. The app sends this TOTP to the web application server.
9. The web application server receives and validates this token by comparing it with a server generated String using the same algorithm and same parameters.
10. The web application server verifies the user and send the user a level-2 JWT to his browser local storage.

Alternatives:

3a. Validation fails. The server rejects the login request.

7a. Fingerprint authentication fails. The app prompt for user's fingerprint again

10a. Validation fails. The server rejects the login request, disable the account, and notify user that his username and password has been compromised

10b. JWT validation fails. The server prompt user to login again.

3.8.3 UML Activity Flow Diagram

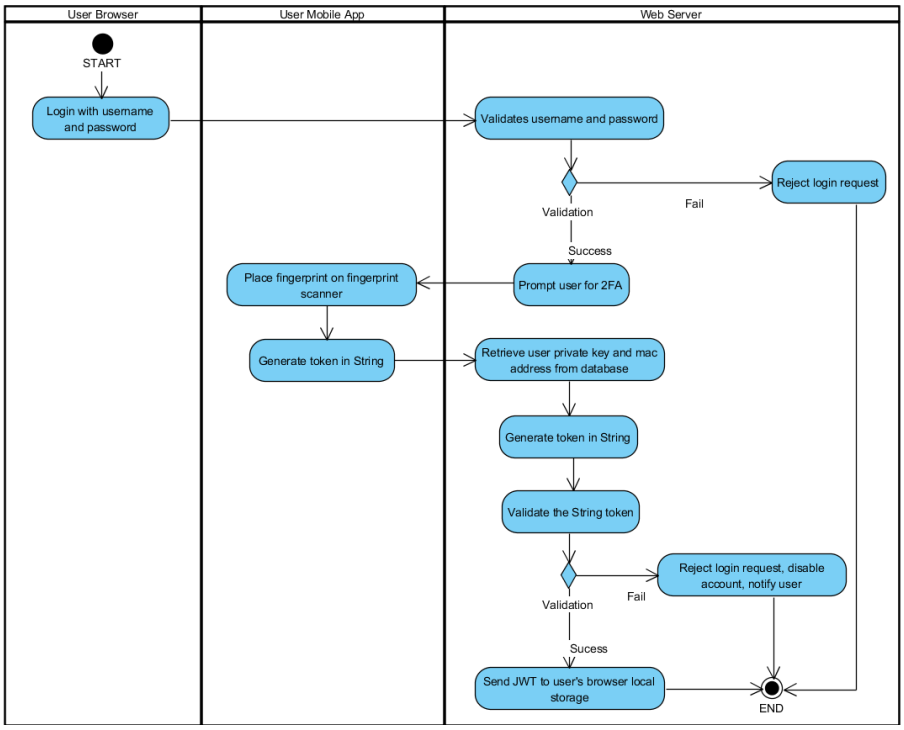


Diagram 3.8.3: Activity Flow Diagram

3.8.4 Web Application Server Architecture Diagram

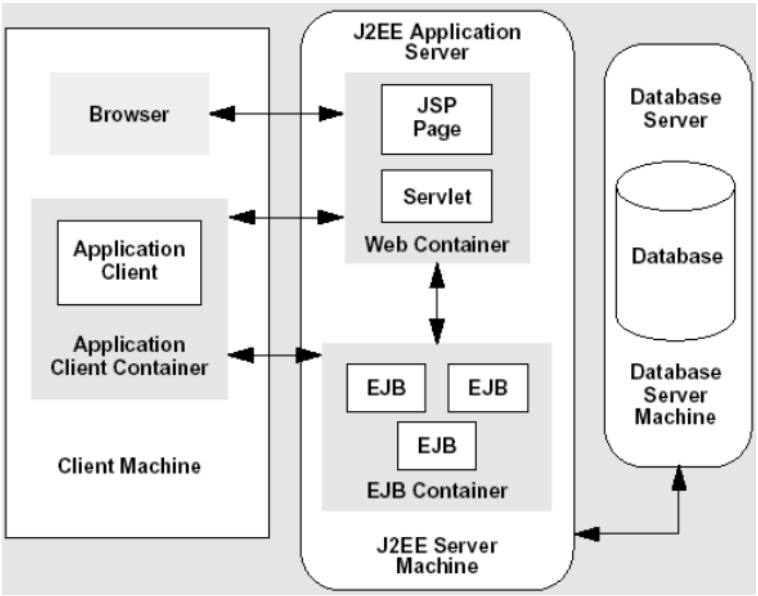


Diagram 3.8.4: Web Application Server Architecture Diagram

3.8.5 Mobile App UML Class Diagram

Diagram 3.8.5: Mobile App UML Class Diagram

3.8.6 UML Use-Case Diagram

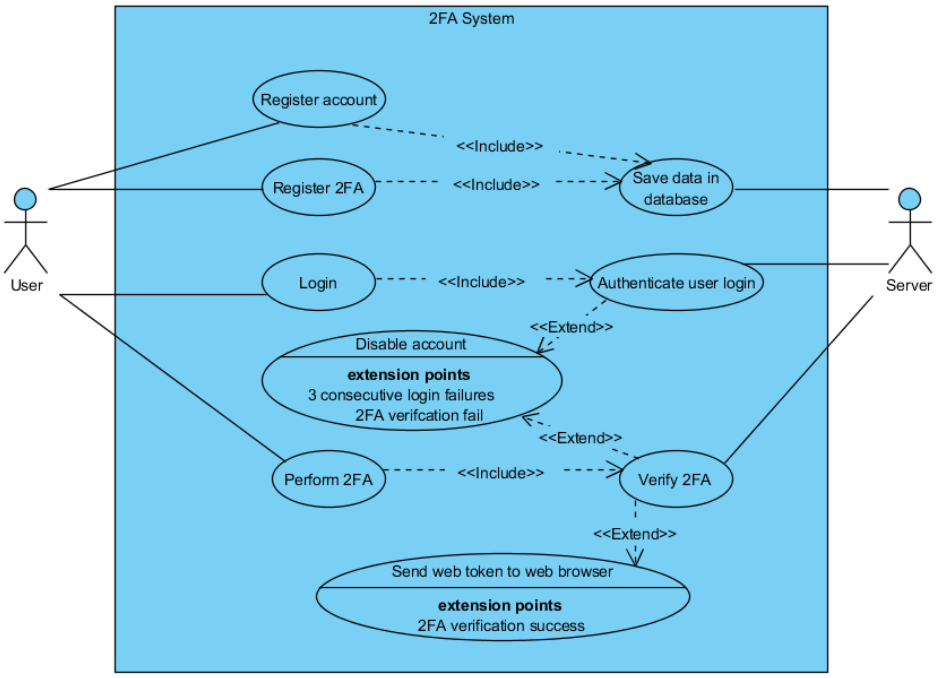


Diagram 3.8.6: UML Use-Case Diagram

3.9 Development Methodology and Planning

3.9.1 Development Methodology

Prototyping methodology is selected to develop this project. Using this approach, analysis, design and implementation phases can be performed concurrently. A system prototype with least number of features will be worked on as soon as the basic of analysis and design are performed. A 2nd prototype will be re-analyzed, re-design & re-implemented until refinement occurs. These phases are repeated until the prototype reaches a stable state.

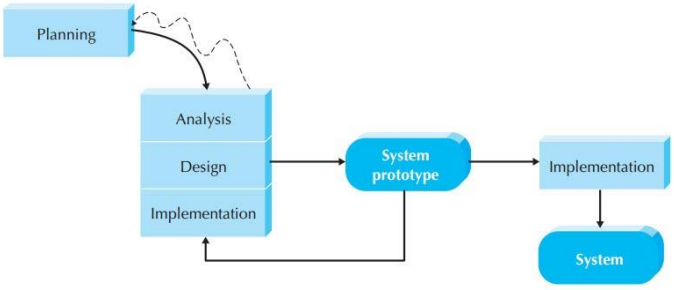


Diagram 3.9.1: Prototyping-based Methodology

3.9.2 Timeline

Project Schedule

Task Name	Start Date	End Date	Duration
PLANNING	1/15/2018	1/21/2018	6
Research Project Background	1/15/2018	1/21/2018	6
Develop Work Plan	1/18/2018	1/21/2018	3
ANALYSIS	1/21/2018	2/14/2018	24
Review Existing System	1/21/2018	2/1/2018	11
Literature Review	1/28/2018	2/10/2018	13
Investigate Requirements	2/4/2018	2/14/2018	10
Document System Proposal	2/7/2018	2/14/2018	7
DESIGN	2/14/2018	2/26/2018	12
System Architectural Design	2/14/2018	3/17/2018	31
Database Design	2/17/2018	6/20/2018	123
Interface Design	2/20/2018	6/20/2018	120
Program Design	2/23/2018	6/20/2018	117
IMPLEMENTATION	2/26/2018	8/20/2018	175
Prototype Development	2/26/2018	8/20/2018	175
Prototype Testing	3/23/2018	8/18/2018	148
PRESENTATION	6/20/2018	8/29/2018	70
Document System Report	6/20/2018	8/20/2018	61
Presentation Preparation	8/20/2018	8/29/2018	9

Table 3.9.2: Project Schedule

Gantt Chart

From the Gantt Chart below, it shows that the Design phase and the Implementation phase are happened parallelly since prototyping methodology is being used in the project development.

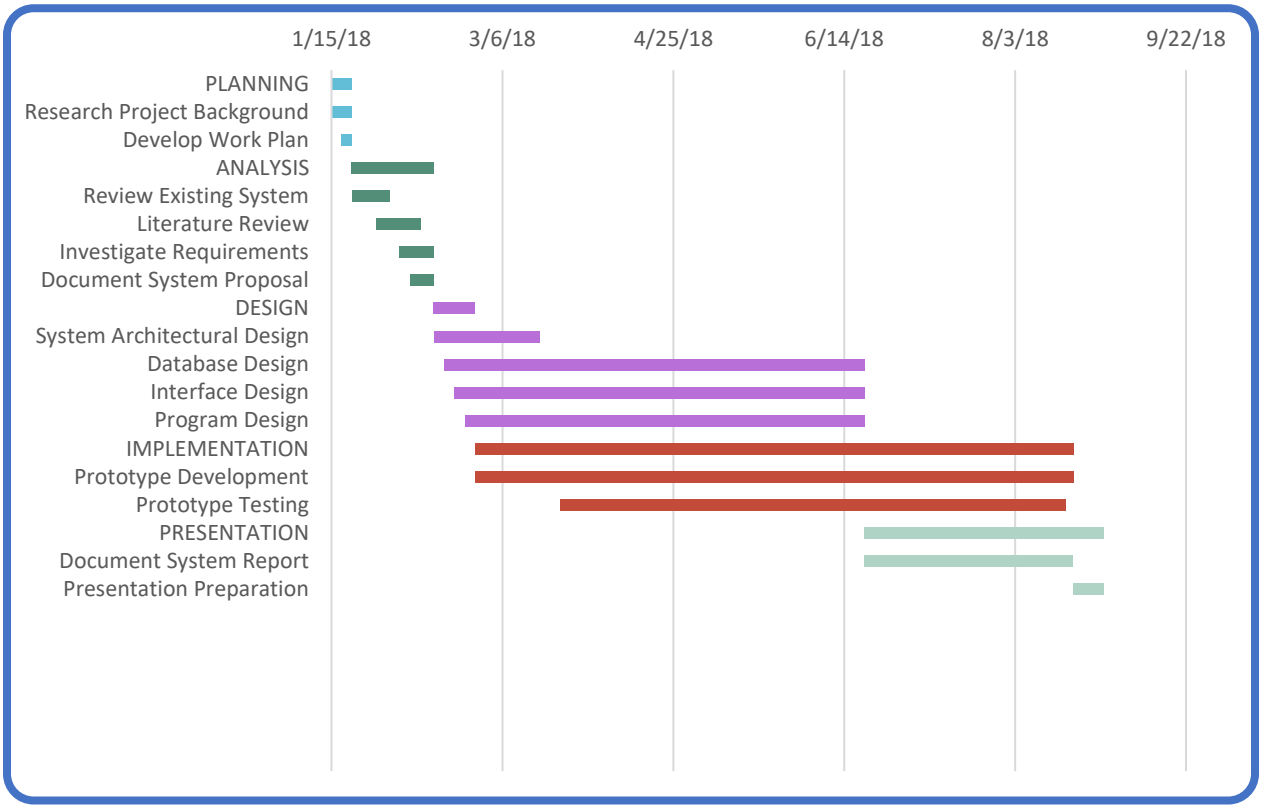
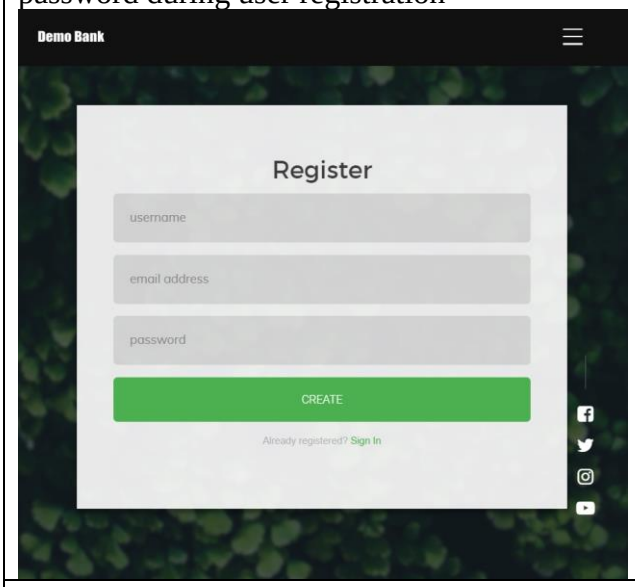
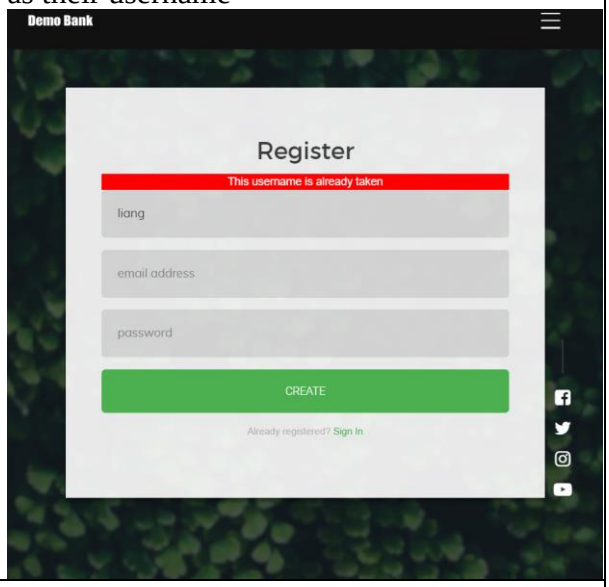
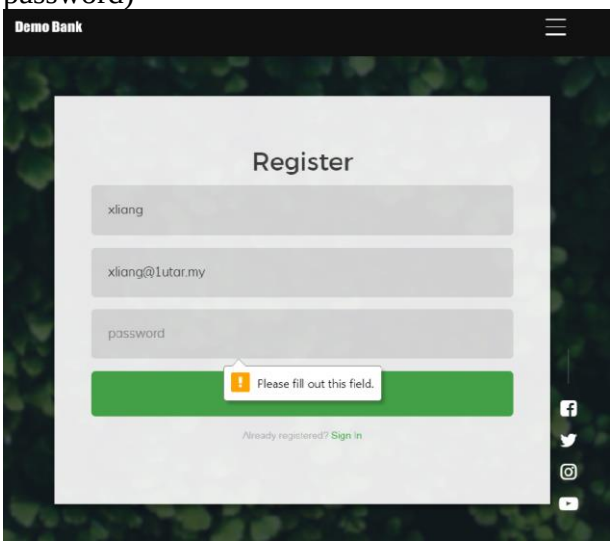
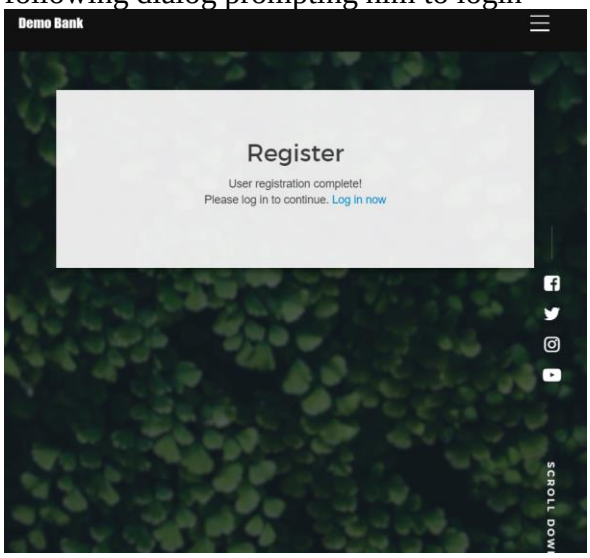


Diagram 3.9.2: Gantt Chart

Chapter 4: Implementation and Testing

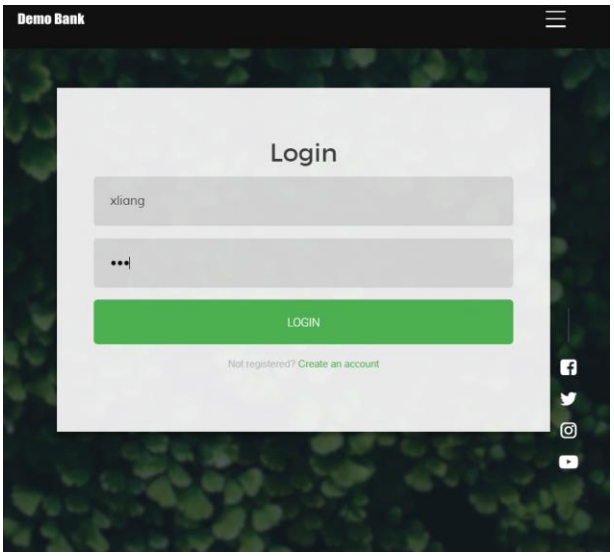
4.1 Implementation

<table>
<tr><th colspan="2">Screen Output</th></tr>
<tr><td>User has to enter for his username, email and password during user registration
 
</td><td>User are not allowed to use existing username as their username
 
</td></tr>
<tr><td>User are also required to use a password with at least 3 characters (Note that in production, we should require user to use an even stronger password)
 
</td><td>If the users input passes all of the form validation criteria, the registration success. On registration successful, user will see the following dialog prompting him to login
 
</td></tr>
</table>

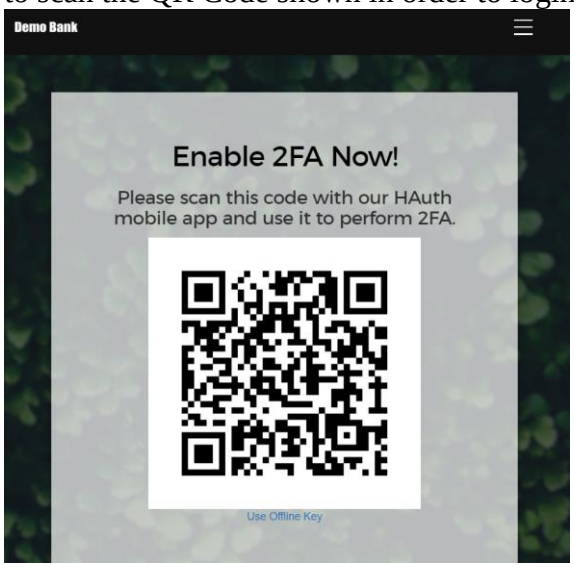
In our database, we can now see that the information of the new user is inserted. As a security measurement, the password is SHA-256 hashed. Also, a unique secret key will be assigned to the user. This unique secret key is used to generate TOTP.

user_id	username	email	password	secret_key	tfa_enabled	tfa_attempts
10	liang	xl@xl.xl	a4e5c395e62e5c55b91c774fc4046711f45d...	31323334353637383930313233...	1	0
12	balotelli	b45@balo.telli	a4e5c395e62e5c55b91c774fc4046711f45d...	c06270328d683971836e76a237a...	1	5
13	desmand	tm@tm.tm	0d40d6b8710d58471e9b67a5cf18cf283574...	415eb61beadad64877603190c05...	1	1
14	xliang	xliang@1utar.my	a4e5c395e62e5c55b91c774fc4046711f45d...	b83e7070b5583ddcfefbd3a68697...	0	0

User is then able to login into their account using his username and password.



On first time login, user will see a dialog asking him to enable 2FA. User are required to scan the QR Code shown in order to login.



The QR Code shown above can be decoded into the following string:

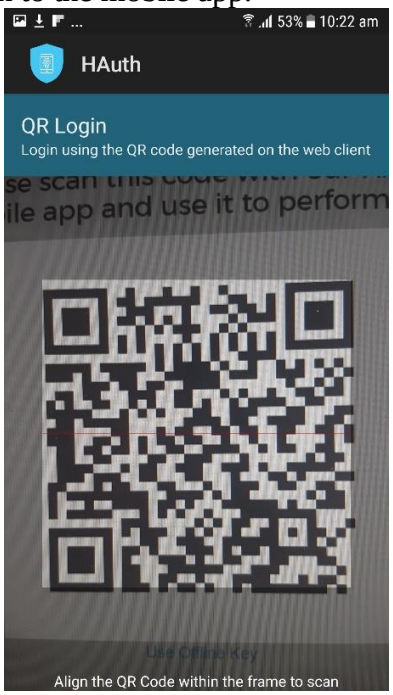
- aLJqc8Ddk6xWKT98orrCtmgWyC3jxgEvHge61eVFA7MhwqAgUR4uwKia1eSz74+Q

This string is an AES encrypted JSON Object which can be further decrypted into:

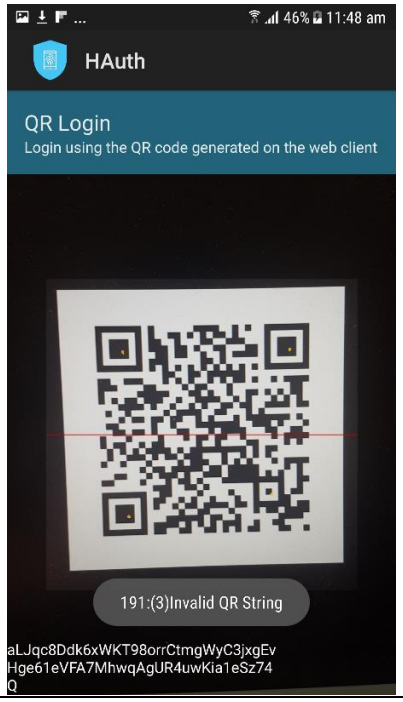
- {"username":"liang","totp":"73392361"}

This QR Code is real-time refreshed in every 30 seconds via WebSocket to match user's current TOTP.

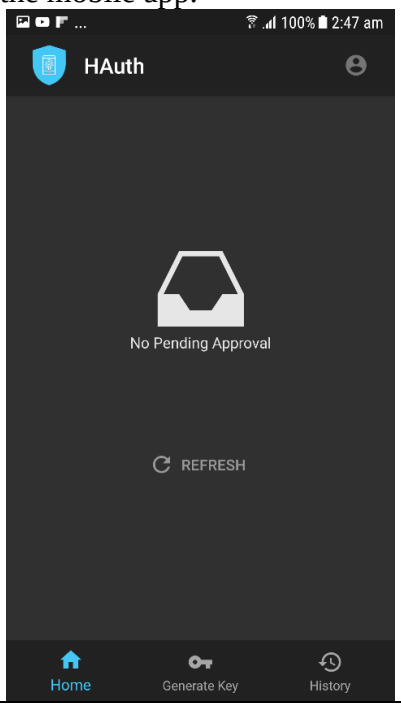
When the mobile app scans QR code, it will automatically send the AES encrypted string to the server. The server will then decrypt and validate this string. If the it is valid, user will be logged in to the mobile app.



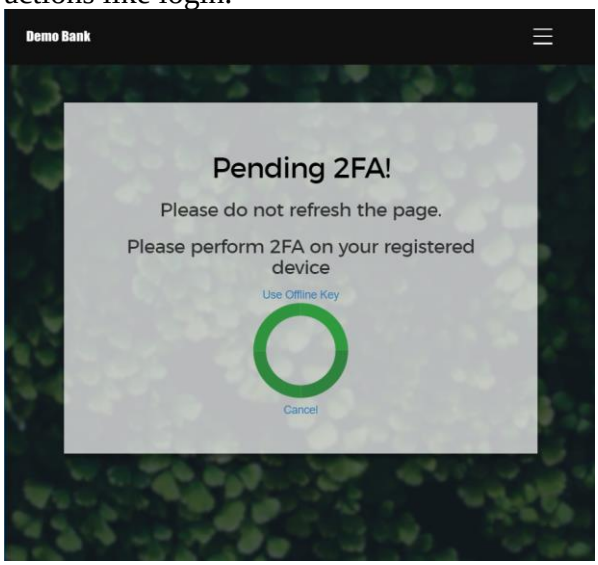
Else if the validation fails, user will receive an error message.



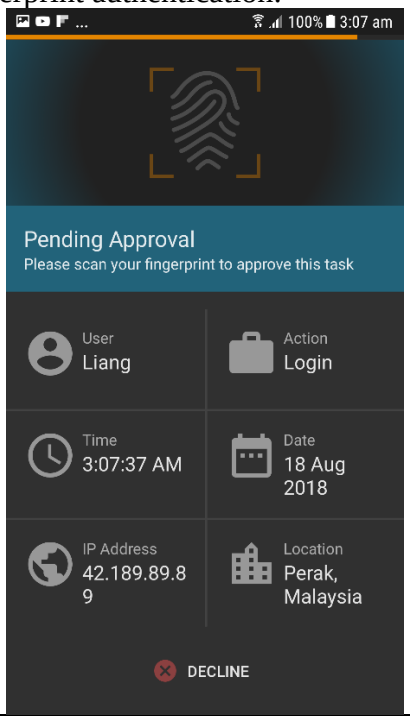
Once login successful, user will reach this home page of the mobile app.



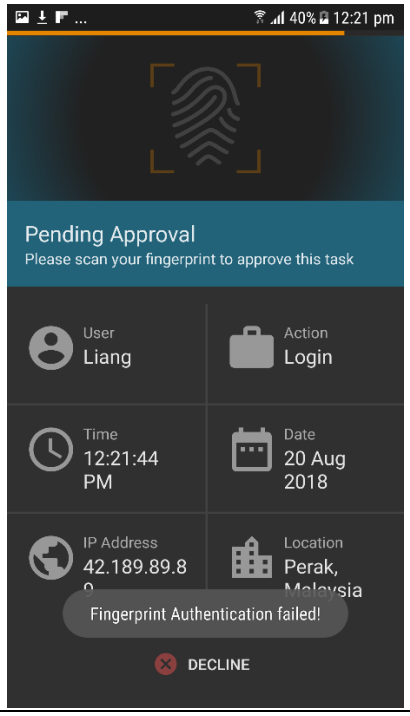
From now onwards, user's account is bound to the mobile device which act as an 2FA authenticator. Thus, user is required to perform 2FA using that particular device for actions like login.



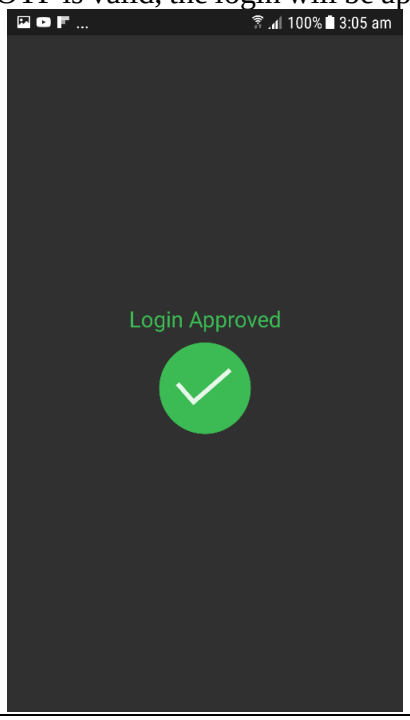
Evertime, 2FA is required, this app will automatically receive the request and prompt for fingerprint authentication.



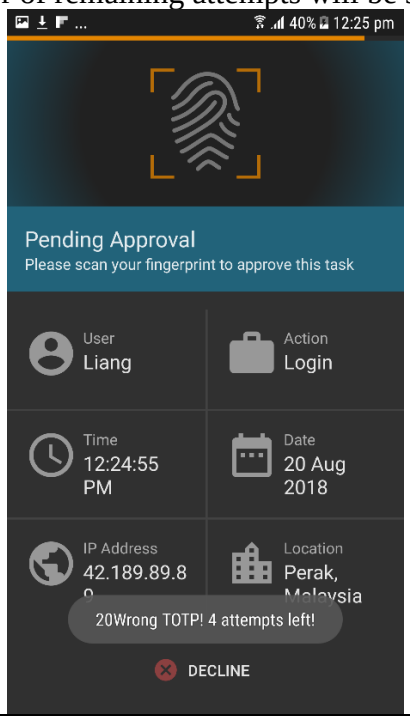
Else if fingerprint authentication failed, an error message will be shown.

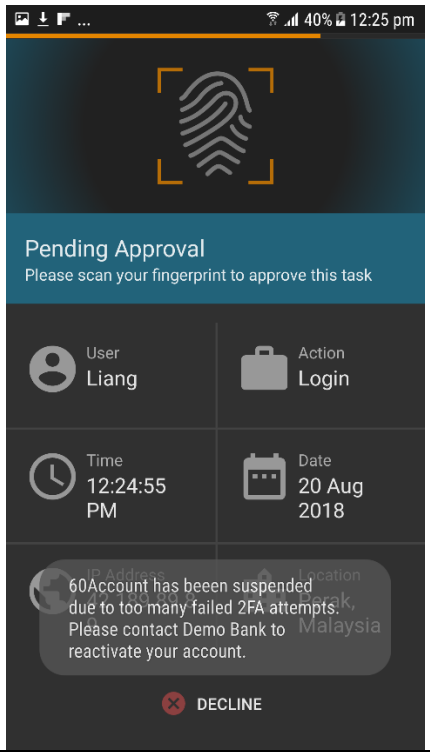
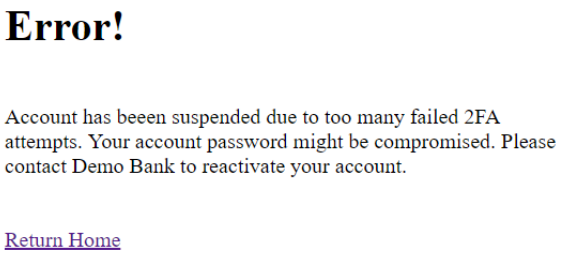
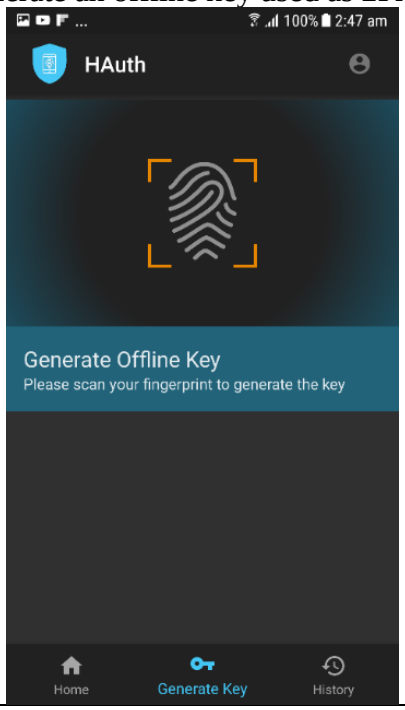
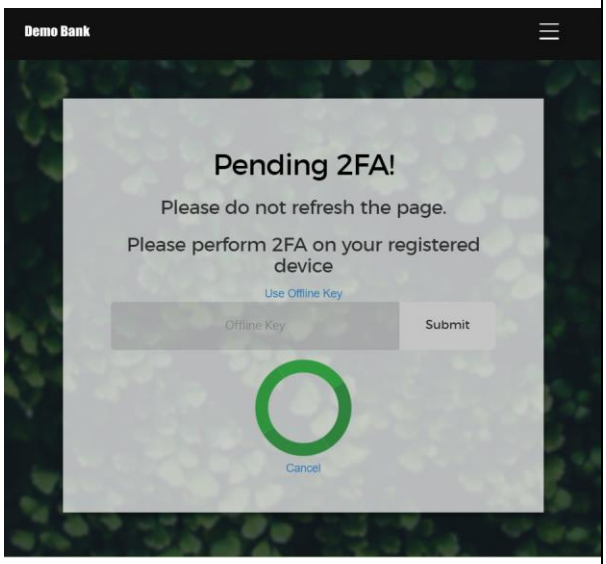


If the fingerprint is authenticated, this mobile app will generate a TOTP send it to the server. If the TOTP is valid, the login will be approved.

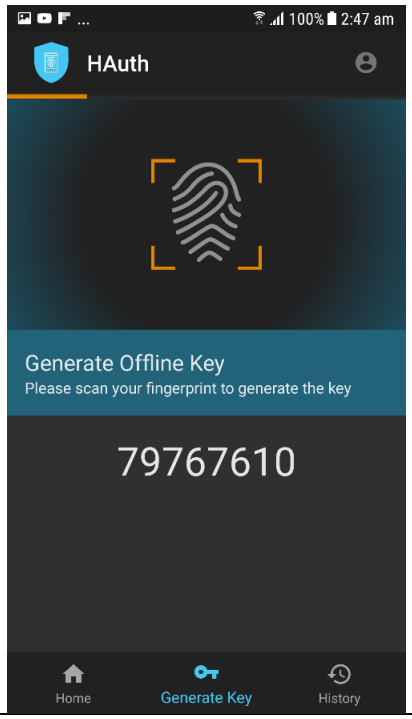


Else if the TOTP is not valid, the server will reject the request. An error message with number of remaining attempts will be shown.

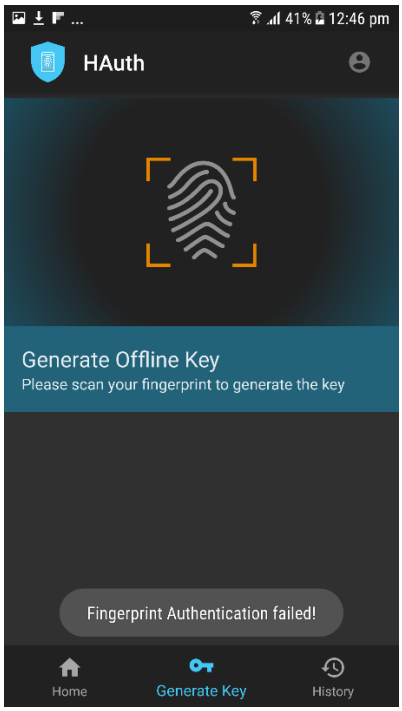


<table>
<tr>
<td>If user gets rejected by server for 5 consecutive times, the account will be suspeneded.
 
</td>
<td>While in the web browser, user will receive the an error message as the following.
 
</td>
</tr>
<tr>
<td>On the other hand, in the mobile app, user can also generate an offline key used as 2FA.
 
</td>
<td>In the web browser, an input box for offline key will appear when user clicked on the “Use Offline Key” button.
 
</td>
</tr>
</table>

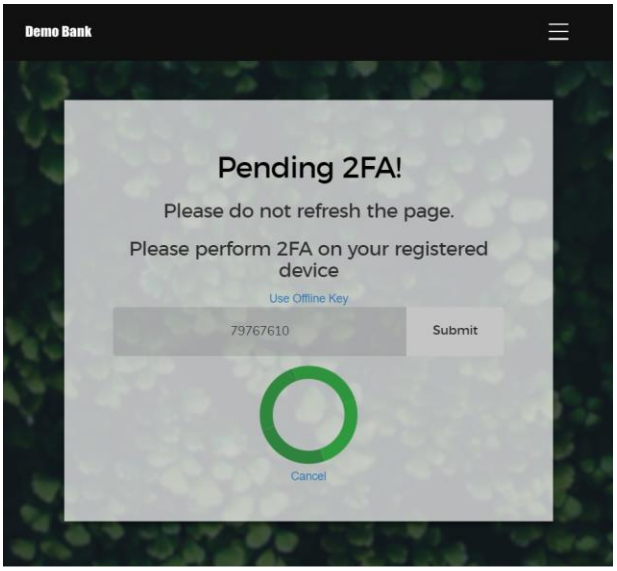
Nevertheless, in order to generate an offline key, fingerprint authentication is required as well. The TOTP will appear once user fingerprint scan is authenticated.



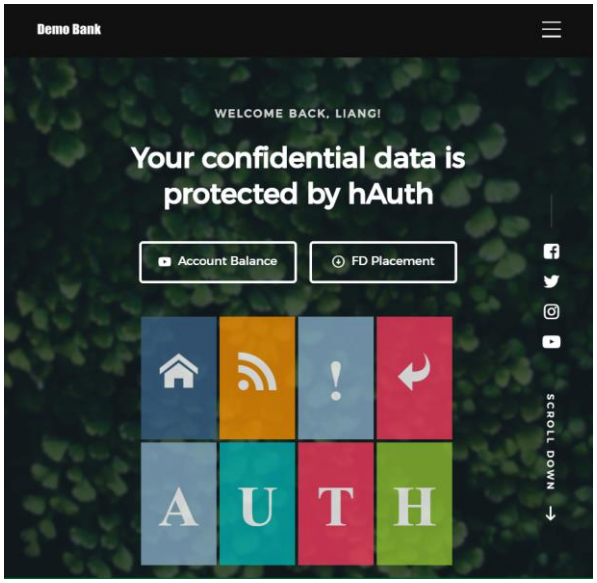
Like previously, user will also receive an error message if the fingerprint authentication fails.



User can then manually key in the generated offline key into the input box and click submit.



Once server approves the 2FA, user get logged in and redirected to his profile page in the web browser.



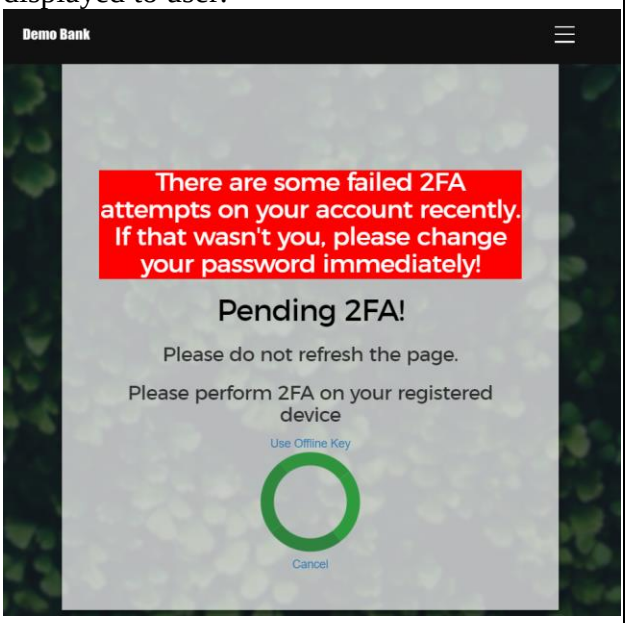
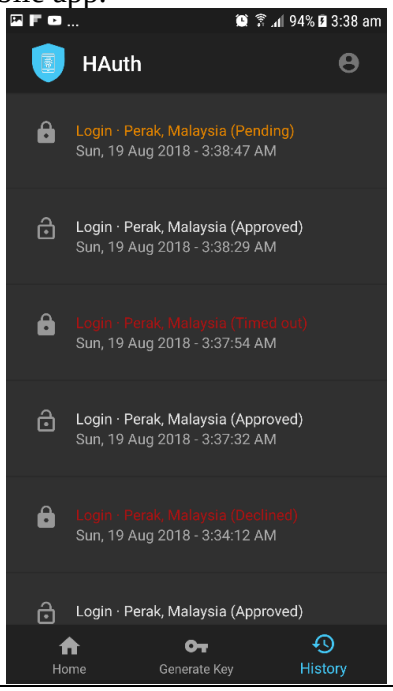
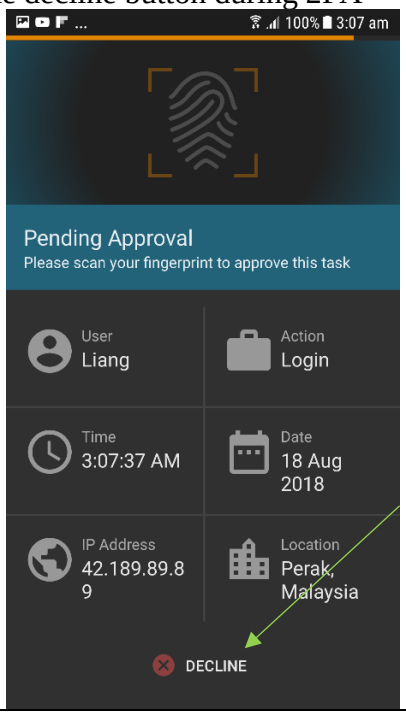
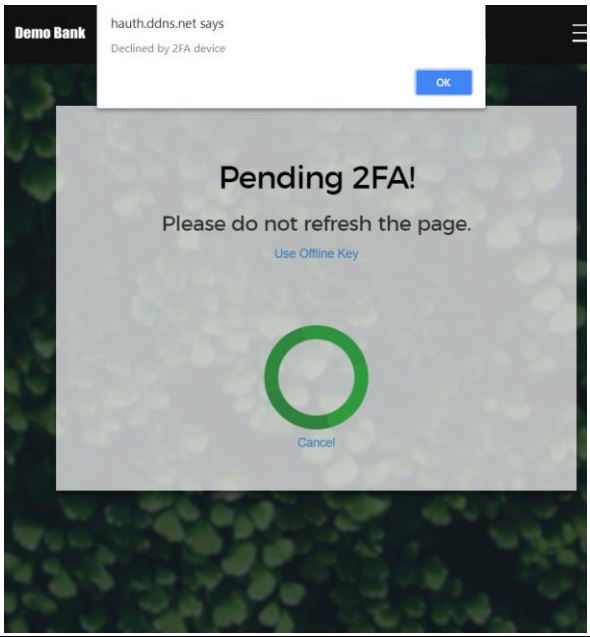
<table>
<tr>
<td>If there are recent failed 2FA attempts in user's account and the account has not been suspended yet. A warning message wll be displayed to user.
 
</td>
<td>User can also check for the recent activity in his account using the "History" feature of our mobile app.
 
</td>
</tr>
<tr>
<td>Finally, user can also decline a login request using the decline button during 2FA
 
</td>
<td>An alert box with message will be shown when the login request is declined
 
</td>
</tr>
</table>

Table 4.1: Screen Output Table

4.2 Unit Testing

The unit test results are recorded in this section.

Unit Testing 1: User registration in Web Browser

No	Description	Predictions	Result
1	User enters a valid username and password	Registration success and prompt a success message to user	True
2	User enters a user name less than 5 characters	Registration fail and prompt an error message with description to the user	True
3	User enters an existed username	Registration fail and prompt an error message with description to the user	True
4	Users leave the entire field blank	Registration fail and prompt an error message with description to the user	True
5	Users enters a short password less than 3 characters	Registration fail and prompt an error message with description to the user	True

Unit Testing 2: User login in Web Browser

No	Description	Predictions	Result
1	User enters a valid username and password.	Gives user a Level-1 JWT and display pending 2FA.	True
2	User enters invalid username and password.	Prompt a message showing that login failed.	True
3	Users leave the entire field blank.	Prompt a message showing that login failed	True

Unit Testing 3: User browser calls AsyncLogin Servlet (A Servlet that is used for listening 2FA result, callbacks and receive Level-2 JWT)

No	Description	Predictions	Result
1	User has a valid Level-1 JWT	Success and response user with "Pending 2FA" page.	True
2	User has an expired JWT	Prompt a message asking user to re-login and redirect him to login page	True
3	User has an invalid JWT	Prompt a message asking user to re-login and redirect him to login page	True
4	User already has a valid Level-2 JWT	Redirect user to his profile	True

Unit Testing 4: User browser request to access his user profile page

No	Description	Predictions	Result
1	User has a valid Level-2 JWT	Respond user with the profile page	True
2	User has an expired JWT	Prompt a message asking user to re-login	True
3	User has an invalid JWT	Prompt a message asking user to re-login	True
4	User only has a valid Level-1 JWT	Redirect user to the "Pending 2FA" page	True

Unit Testing 5: User login in Mobile App (QR Code)

No	Description	Predictions	Result
1	Scanning a valid QR Code	Login success	True
2	Scanning an invalid QR Code	Prompt an error message showing that login failed	True
3	Login to an account that is already 2FA enabled	Prompt an error message showing that login failed.	True
4	Login in to an account that does not exist	Prompt an error message showing that login failed	True

Unit Testing 6: Perform 2FA in Mobile App

No	Description	Predictions	Result
1	Fingerprint authentication succeed and send a valid TOTP	2FA success and display user a success dialog	True
2	Fingerprint authentication failed	Prompt an error message showing that fingerprint authentication failed	True
3	Fingerprint authentication succeed and send an invalid TOTP	Prompt an error message to user showing that TOTP is not valid and his remaining number of attempts.	True

4.3 Implementation Issues and Challenges

The main challenge in this project is the 2FA algorithm to generate a unique passcode that expires every 30 seconds if we set the time step to 30 seconds. Assume that if every passcode generation is happened on 0 to 29 seconds and 30 to 59 seconds of every minute, if the passcode on the client side is at the 29th seconds, due to some latency, by the time the passcode reaches the server it is already at the 30th seconds. In this case, server will reject the authentication. In order to solve this issue, some methods are taken into consideration, such as making timeout at the 29th and 59th seconds. There also might be clock drift the mobile device and our server, so our server should be enhanced in order to handle this kind of situation.

4.4 Comparison of Time Required by Existing System

A test is conducted to compare the time taken for a successful 2FA attempt. In order to produce a fair result, the 2FA registered phone is locked with no running apps in background at the beginning of each attempts.

2FA System	1st Attempt	2nd Attempt	3rd Attempt	Mean
Two-Factor Human Authentication	7.8s	7.8s	6.0s	7.2s
Steam Guard Mobile Authenticator	22.8s	22.5s	26.4s	23.9s
Public Bank eCommerce OTP	51.0s	34.5s	25.7s	37.1s

Table 4.4: Comparison of Time Required by Existing System

From the result, we can see that our system is able to achieve a consistent result of within 10 seconds, while the other two competitors requires twice as much the time that our system is able to complete a 2FA. Besides that, the Public Bank eCommerce OTP that uses SMS produced inconsistent result.

Chapter 5: Conclusion

Using only something you know like a password is very vulnerable to common hacking techniques especially social engineering. However, multi-factor authentication processes are often monotonous. Even until today, there are still many people that prefer not to enable 2FA on their account to save themselves some hassle.

2FA are widely used by the industry, even the big technology companies like Google and Facebook. However, these 2FA that were introduced cannot get away with requiring user many steps to setup and also to use it. Besides that, the most common 2FA that is being used in the industry currently is OTP (One-time Password). OTP cannot avoid user for typing it out again especially when user is browsing in desktop client and the OTP is sent to his mobile device. This is where our system has the edges over these traditional 2FA systems. Our system is extremely easy to setup, with just one scan of a QR Code, and the time required to perform a 2FA is extremely quick as well, with just one touch of a finger. In additional to that, our system also uses a combination 3 factors of credentials while the traditional 2FA systems only uses 2 factors.

This two-factor authentication system is a user-friendly authentication system which can be a new alternative for online banking authentication system. This system authenticates you by using your password (Something you know), a private key assigned to your mobile app (Something you have) and your fingerprint (Something you are). These credentials are from 3 different categories but it takes user only 2 steps to authenticate themselves. This system has the capability of providing a secured authentication system while minimizing the hassle of using 2FA.

Bibliography References

Todorov, D. (2007). *Mechanics of user identification and authentication*. Boca Raton: Auerbach Publications, pp.1, 18.

Convery, S. (2007). *Network Authentication, Authorization, and Accounting: Part One - The Internet Protocol Journal - Volume 10, No. 1*. [online] Cisco. Available at: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-35/101-aaa-part1.html [Accessed 12 Aug. 2017].

Web.mit.edu. (2015). *Kerberos: The Network Authentication Protocol*. [online] Available at: https://web.mit.edu/kerberos/ [Accessed 13 Aug. 2017].

Henry, A. (2016). *The Difference Between Two-Factor and Two-Step Authentication*. [online] Lifehacker.com. Available at: http://lifehacker.com/the-difference-between-two-factor-and-two-step-authenti-1787159870 [Accessed 13 Aug. 2017]. Moore, P. (2014). *The difference between two-factor and two-step authentication*. [online] Paul Moore. Available at: https://paul.reviews/the-difference-between-two-factor-and-two-step-authentication/ [Accessed 14 Aug. 2017].

HSBC, 2017. *HSBC Mobile Banking.* Version 1.5.18.0 [Mobile app]. Google Play Store.

Connect-content.us.hsbc.com. (2016). *Apple Touch ID now available on the HSBCnet Mobile app*. [online] Available at: http://connect-content.us.hsbc.com/hsbc_pcm/onetime/2017/January/17_mobile_touch_id.html [Accessed 15 Aug. 2017].

Burke, D. (2017). Android: celebrating a big milestone together with you. [Blog] *The Keyword*. Available at: https://www.blog.google/products/android/2bn-milestone/ [Accessed 15 Aug. 2017].

Valve Corporation, 2017. *Steam.* Version 2.3.1. [Mobile app]. Google Play Store.

Support.steampowered.com. (2017). *Steam Guard Mobile Authenticator - Managing Your Account Features - Knowledge Base - Steam Support*. [online] Available at: https://support.steampowered.com/kb_article.php?ref=8625-wrah-9030 [Accessed 15 Aug. 2017].

Pbebank.com. (2017). *Personal Banking*. [online] Available at: https://www.pbebank.com/ [Accessed 15 Aug. 2017].

Pbebank.com. (2017). *Personal Banking FAQs*. [online] Available at: https://www.pbebank.com/Personal-Banking/FAQs.aspx [Accessed 4 Aug. 2018].

www.sun.com. (2005) Architecting and Designing J2EE Applications. [online] Available at: http://java.boot.by/scea5-guide/ch02s02.html. [Accessed 4 Aug. 2018].

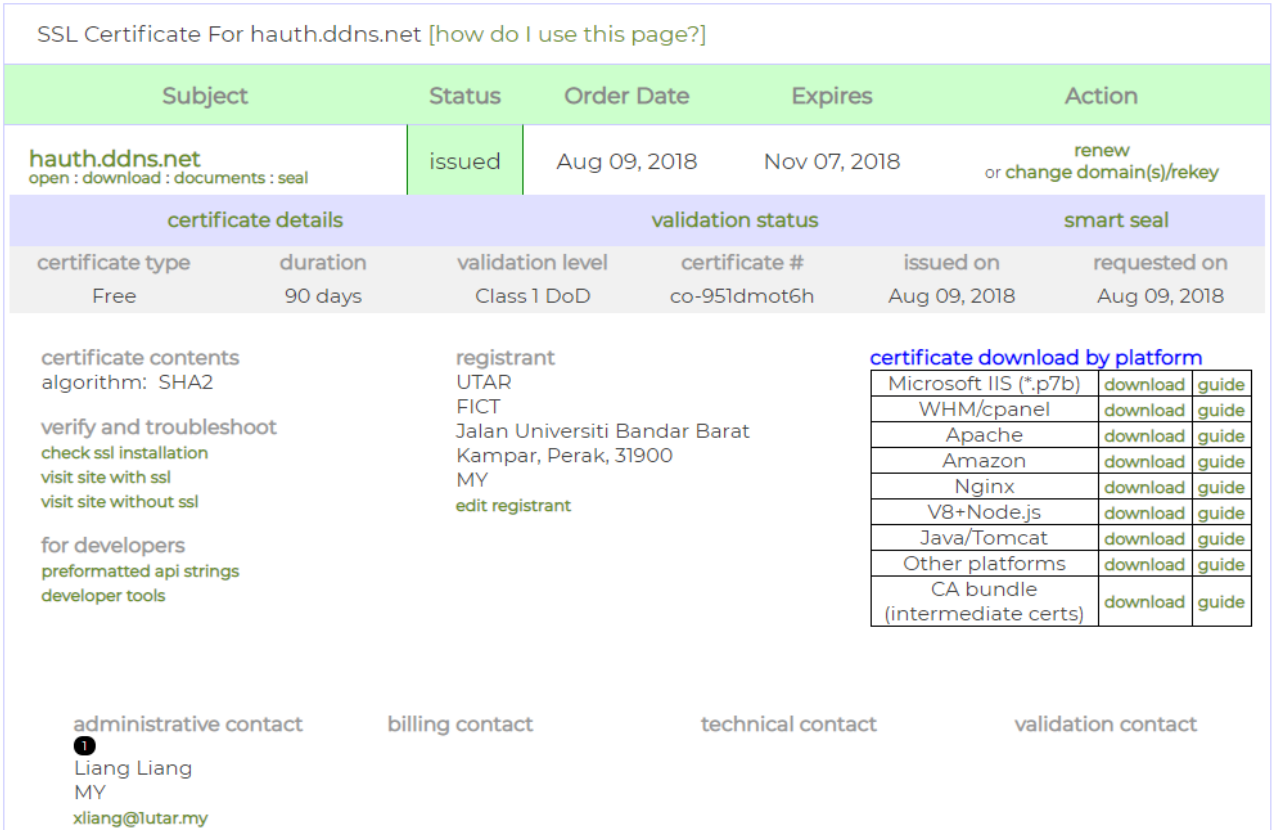
Garrod, C. (2006). *System Analysis and Design*. 3rd ed. John Wiley & Sons, Inc., p.46.

developer.android.com. (2018). *Android 6.0 APIs.* [online] Available at: https://developer.android.com/about/versions/marshmallow/android-6.0#fingerprint-authentication [Accessed 4 Aug. 2018].

M'Raihi, D. (2011). TOTP: Time-Based One-Time Password Algorithm. [online] Available at: https://tools.ietf.org/html/rfc6238 [Accessed 4 Aug. 2018].

Abbott, T. (2016). *Where to Store your JWTs – Cookies vs HTML5 Web Storage.* [online] Available at: https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage [Accessed 4 Aug. 2018].

APPENDIX A: SSL Certificate Signing by Certificate Authority



SSL Certificate For hauth.ddns.net [how do I use this page?]

Subject	Status	Order Date	Expires	Action
hauth.ddns.net open : download : documents : seal	issued	Aug 09, 2018	Nov 07, 2018	renew or change domain(s)/rekey

certificate details		validation status		smart seal	
certificate type	duration	validation level	certificate #	issued on	requested on
Free	90 days	Class 1 DoD	co-951dmot6h	Aug 09, 2018	Aug 09, 2018

certificate contents
algorithm: SHA2

verify and troubleshoot
check ssl installation
visit site with ssl
visit site without ssl

for developers
preformatted api strings
developer tools

registrant
UTAR
FICT
Jalan Universiti Bandar Barat
Kampar, Perak, 31900
MY
edit registrant

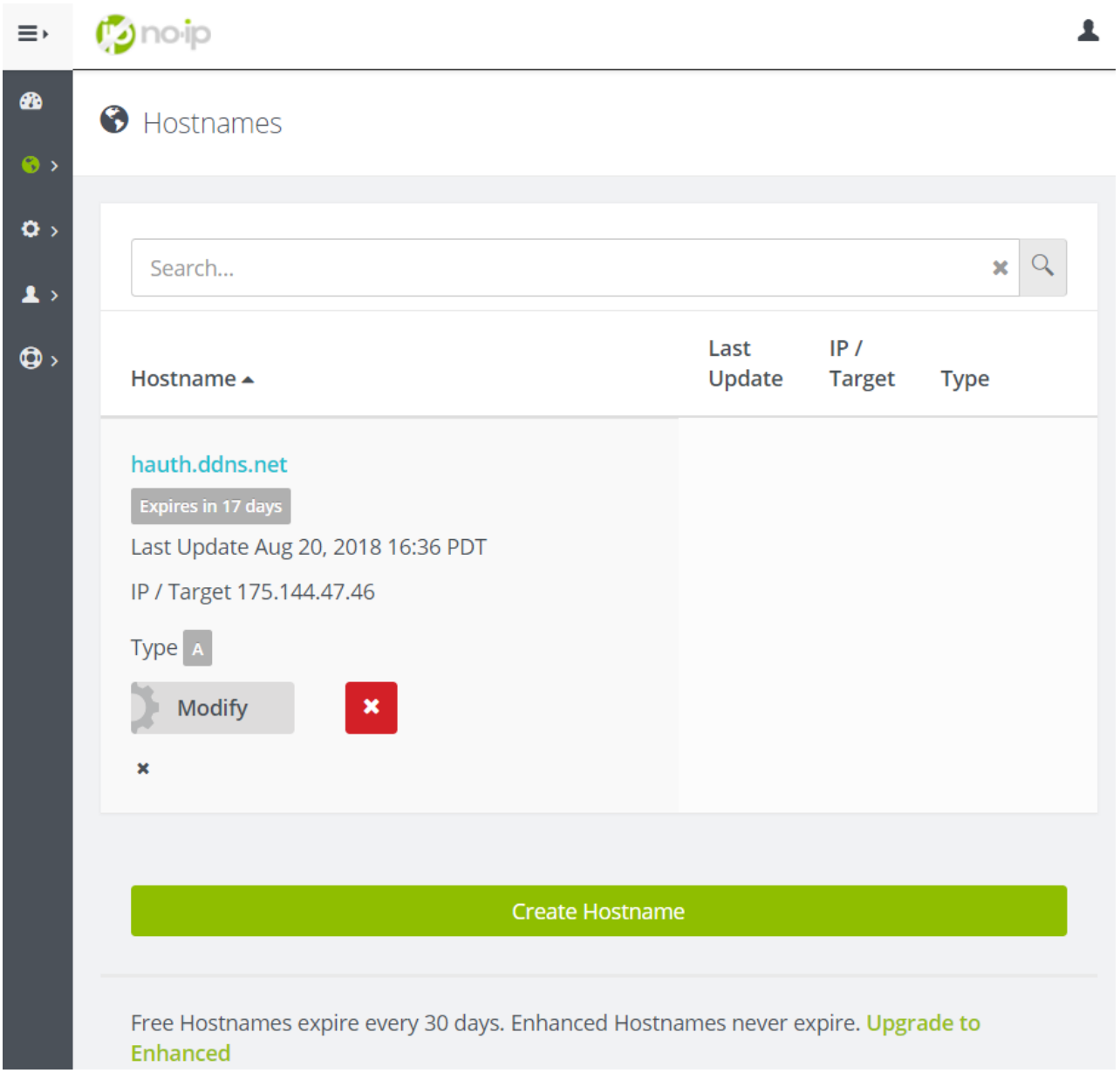
certificate download by platform

Microsoft IIS (*.p7b)	download	guide
WHM/cpanel	download	guide
Apache	download	guide
Amazon	download	guide
Nginx	download	guide
V8+Node.js	download	guide
Java/Tomcat	download	guide
Other platforms	download	guide
CA bundle (intermediate certs)	download	guide

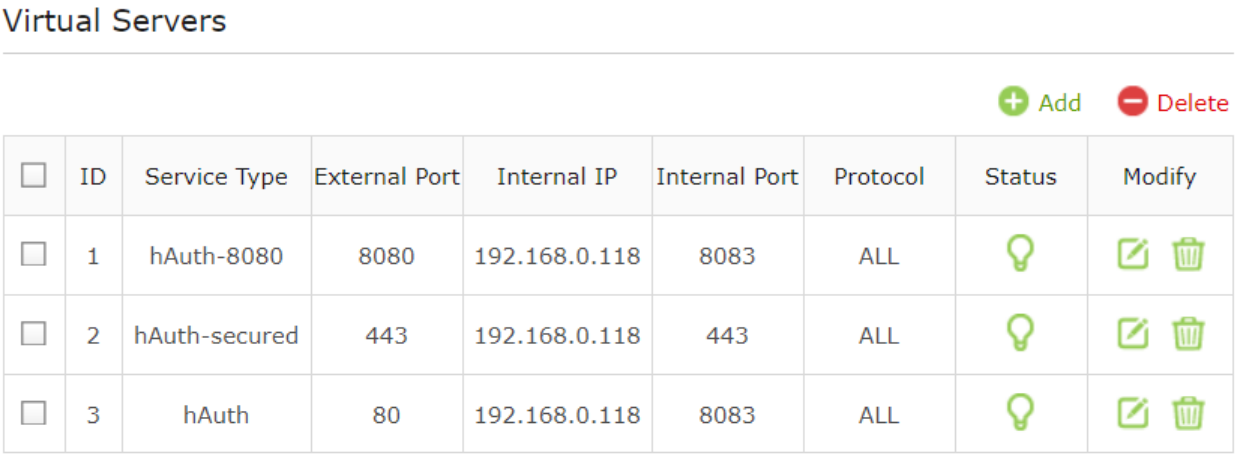
administrative contact | billing contact | technical contact | validation contact

Liang Liang
MY
xliang@1utar.my

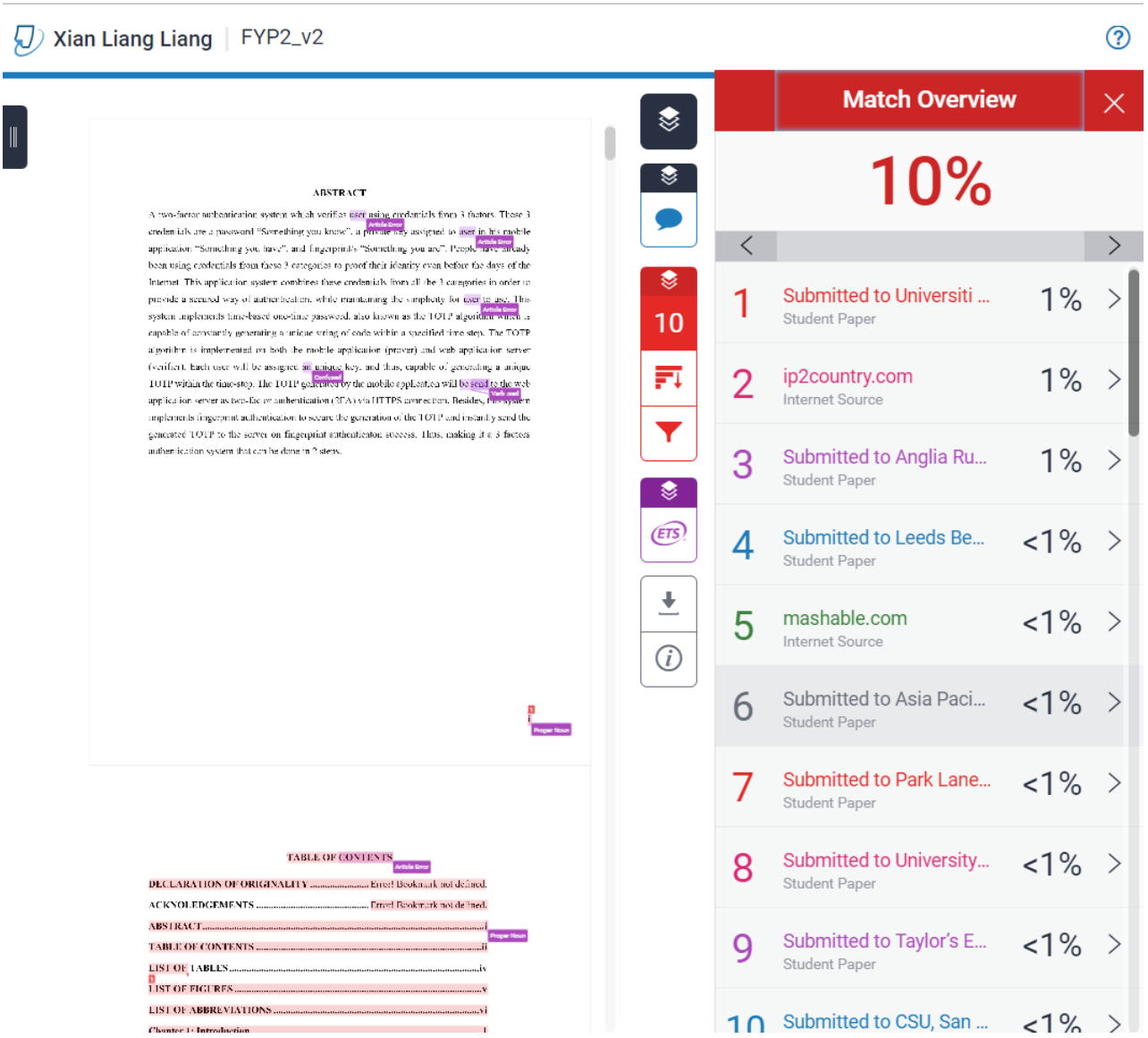
APPENDIX B: DDNS Configuration



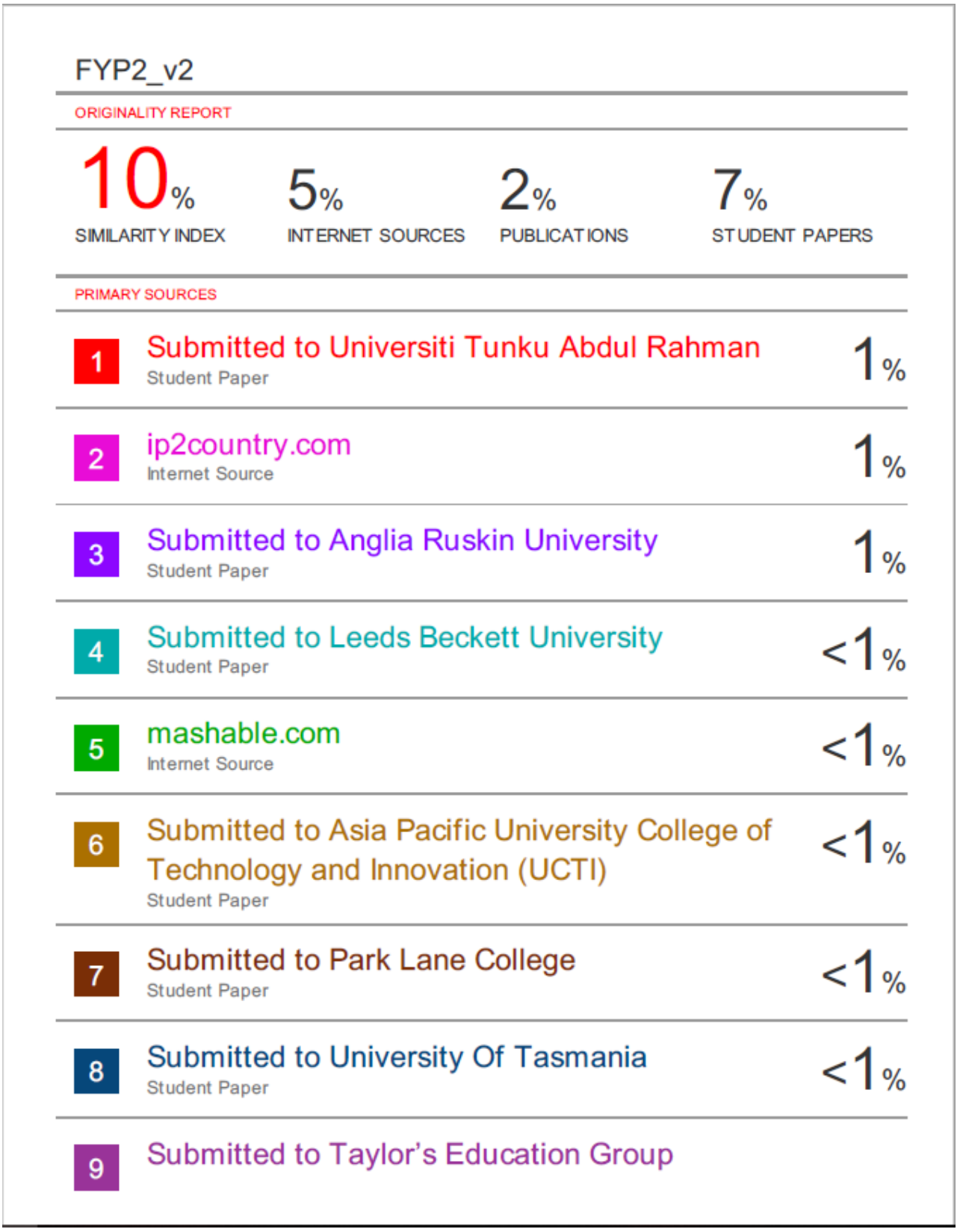
APPENDIX C: Router Port Forwarding Configuration



APPENDIX D: Summary of Plagiarism (Turnitin)



APPENDIX D: Originality Report (Turnitin)



FYP2_v2

ORIGINALITY REPORT

10%	5%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
2	ip2country.com Internet Source	1%
3	Submitted to Anglia Ruskin University Student Paper	1%
4	Submitted to Leeds Beckett University Student Paper	<1%
5	mashable.com Internet Source	<1%
6	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
7	Submitted to Park Lane College Student Paper	<1%
8	Submitted to University Of Tasmania Student Paper	<1%
9	Submitted to Taylor's Education Group	

UNIVERSITY TUNKU ABDUL RAHMAN
Two-Factor Human Authentication

Liang Xian Liang
Supervisor: Dr Vasaki a/p Ponnusamy

PROJECT OVERVIEW

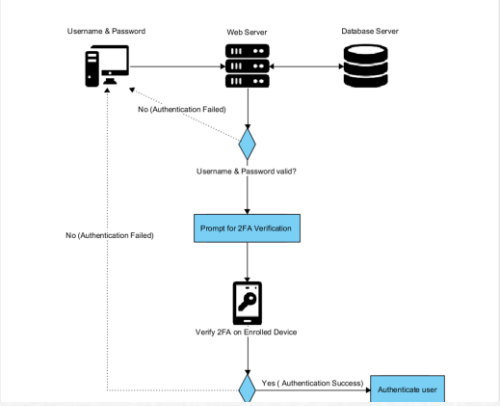
Two-Factor Human Authentication is able the verify and authenticate user using credentials from 3 different factors, which are a password, a unique security key, and biometric fingerprint/s.

PROJECT OBJECTIVES

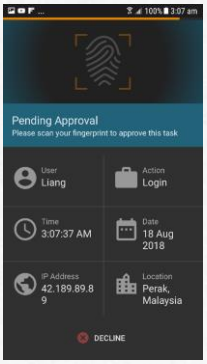
The main objective of this project is to come out with a secured yet effortless two-factor authentication system.

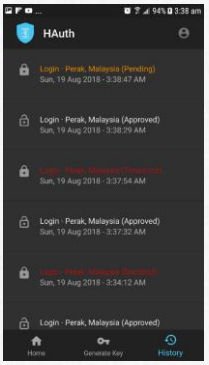
This project aims to reduce the time needed to perform 2FA and increase users experience at the same time

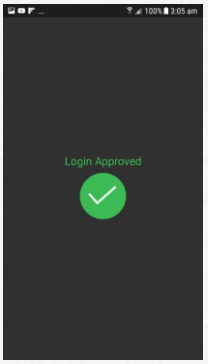
SYSTEM FLOW



SCREEN OUTPUT







FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 2**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Continue with FYP1's work.

2. WORK TO BE DONE

Modify FYP1's report structure into FYP2's report format.

3. PROBLEMS ENCOUNTERED

N/A

4. SELF EVALUATION OF THE PROGRESS

N/A

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Updated FYP report.

2. WORK TO BE DONE

Modify system according feedback from moderator during FYP1 presentation

3. PROBLEMS ENCOUNTERED

Creating an SSL certificate to be signed by CA

4. SELF EVALUATION OF THE PROGRESS

Able to solve the problem using resources available

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Received a signed SSL certificate.

2. WORK TO BE DONE

Enhance the FYP1 system by adding history function

3. PROBLEMS ENCOUNTERED

N/A

4. SELF EVALUATION OF THE PROGRESS

Able to complete the task before the expected time

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 8**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Completed system's new history function.

2. WORK TO BE DONE

Convert user public IP to Location in the history function

3. PROBLEMS ENCOUNTERED

Finding a free IP to Location lookup table

4. SELF EVALUATION OF THE PROGRESS

N/A

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 10**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Added IP to Location lookup table in the system's database

2. WORK TO BE DONE

Research on JWT to manage user's session

3. PROBLEMS ENCOUNTERED

Report chapters arrangement

4. SELF EVALUATION OF THE PROGRESS

Able to find related information on the internet and do self-learning

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Liang Xian Liang 1403375	
Supervisor: Dr Vasaki a/p Ponnusamy	
Project Title: Two-Factor Human Authentication	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

A working system

2. WORK TO BE DONE

Perform testing on the system

3. PROBLEMS ENCOUNTERED

Documentation of testing in the report

4. SELF EVALUATION OF THE PROGRESS

Take initiative to consult supervisor for help

_____ _____

Supervisor's signature Student's signature

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: _____ % **Similarity by source** Internet Sources: _____ % Publications: _____ % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: _____	
Parameters of originality required and limits approved by UTAR are as follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

_____
Signature of Supervisor

Name: _____

Date: _____

_____
Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (PERAK CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. _____ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. _____ (Signature of Supervisor) Date: