# **Software Projects (55-407815)**

# **Stage 2**

|  |  |
| --- | --- |
| **Project Title** | Transaction Authorisation Code (TAC) for Internet Banking Authentication and Fund Transfer using Vigenère Cipher Encryption |
| **Client** | Mohd Zairul Mazwan Jilani |
| **Deliverable** | A proof of concept of a web application |

# **Project Description**

Transaction Authorisation Code (TAC) is commonly used in banking to complete financial transactions securely over the Internet. This method is known as two-factor authentication (2-FA). The use of TAC is also extended to other online processes, such as login, as part of a security measure. TACs are usually sent to the user’s mobile phone or email, who performs transactions online. A TAC may consist of alphabets and numerical values of 5 characters in length.

This project requires the developer to develop a proof of concept of a web application that enables users to log in to an Internet banking platform and securely perform fund transfers to recipients. The system to be developed is an Internet banking application for a consumer bank, namely “**MZMazwan Bank**.”

# **Business Requirements and the Scope**

The MZMazwan Bank would like to implement a security measure for their login process to the Internet banking system and fund transfer transactions using TAC. All unique TACs will be generated by the system and sent to the user’s email address (Bank’s customer). The system will use the user’s email address, which is **securely saved** in the system database.

The main requirements and the scope are as follows:

# **Login**

The TAC process will be implemented to validate genuine logins to the system. However, not all logins will require a TAC. There are conditions:

* Users who have not logged in to the system for a long interval, i.e., more than 30 days, **must** be verified with a TAC.
* Users logged in to the system frequently, more than twice within 5 minutes, **must** be verified with a TAC.

# **Fund Transfer**

All fund transfers of **more than £1000** will require a TAC to complete the transactions. The implementation of TAC for this phase will only apply to internal fund transfers (sending money between Bank account holders of MZMazwan Bank).

# **TAC Format**

A TAC will consist of five (5) characters from the following list combinations:

* Three (3) upper case alphabets. The first three characters.
* Two (2) numeric values. The last two characters.

*Example:*

ABC56

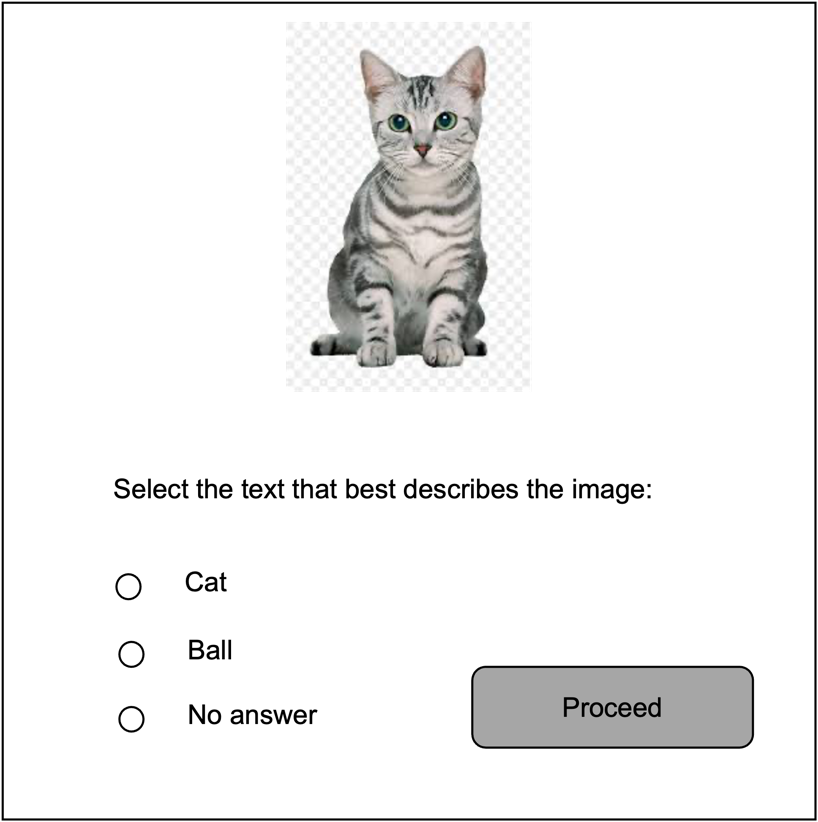
The code should be generated randomly by the system. The code will be encrypted (encoded) before sending it to the user’s email. The encryption method is elaborated in section 2.4.1.

# **Encryption Method**

All TACs will be encrypted using the Vigenère Cipher method before sending them to the user’s email. The method algorithm is explained in this [reference](https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html). Upon receiving the TAC (encoded) through email, the user will need to key in the TAC and click “Proceed”. The system will then decode the encoded TAC using the same keyword so that it matches the original value of the TAC.

# **Encryption Process**

The Vigenère Cipher method needs a keyword to encode a TAC. The system should randomly provide a keyword through an image (object). The user would need to select the text that best describes the object from the image. *Example:*



From the example above, the system must validate the answer from the user. Only a valid answer will be processed if the user selects “*Ball*” or “*No answer*”, an appropriate error message should be prompted (**maximum 2 chances**). If the user selects “Cat” (used as the keyword), then the following processes will be executed.

Upon clicking the proceed button (with a correct answer), the system will:

1. generate a TAC.
2. encode the TAC using the keyword. In the example above is “cat”.
3. send the encoded TAC to the user’s email address.
4. The user will be prompted to check their email, which is registered with the bank, to retrieve the TAC for the transaction.
5. The user will need to input the TAC they receive and click “Proceed”.
6. The system will decode the TAC using the same keyword (“cat”) and validate it.
7. The login/fund transfer should be flagged as successful upon process 6 (successful validation).

A TAC is only valid for a single transaction within 5 minutes. If the user inputs the TAC beyond this period, the TAC will be flagged as stale, and the user should redo the login/fund transfer process.

Incorrect input of a TAC will result in **two attempts** for the user to input the same TAC. Otherwise, the system will cancel the transaction/login and return to the user’s homepage. To log in, return to the bank’s landing page.

# **Admin User**

Apart from the customer users, the admin user is also required for this project. The admin users can log in to the system and do the following tasks.

# **Block Customer Login Account**

The admin user should be able to search customer accounts (Internet Banking) and block them from accessing the system. Upon blocking the account, an email should be sent to their email address informing them the Internet banking account has been blocked due to suspicious transactions.

# **Set System Parameters**

The admin user should be able to set the following system parameters:

1. *Fund transfer limit for TAC*. The default maximum limit for fund transfers without a TAC verification is £1000. However, the system admin should be able to increase and decrease the limit.
2. *Login interval time for TAC*. The default inactivity login interval, which requires a TAC verification, is 30 days. The system admin should be able to amend this value according to central bank regulations.

# **Technical Requirements**

The system should be a proof of concept for a web application. A local host is acceptable for the project’s deliverables. However, using an email API for Google emails is required for TACs. Using any web application framework is not a requirement, and procedural programming of PHP or **equivalents** is sufficient to progress with this project.

A simple database design and some fabricated datasets are required for this project. The project's focal objective is not the database design but the application of Vigenere Cipher encryption for the TAC. However, this project seeks a good database design that supports the concept.

# **Other Generic Requirements**

This project is an enhancement to the existing functionalities of the Internet banking platform, i.e., login and fund transfer. Therefore, the proof of concept for this project would necessitate having common Internet banking features such as account holders, account balance, user status and many more. Using sample data for account holders, accounts, and internal users (admin) is part of the project package. The requirements for this section should be gathered from the client during the lecture/tutorial sessions.

# **Optional Requirement**

Sending TACs to WhatsApp (business account) instead of email is preferable. However, this requirement is not mandatory for the proof-of-concept version.

# **Useful Source Code**

<https://replit.com/@zairulmazwan/Vigener-Cipher-Encryption-and-Decryption>

<https://replit.com/@zairulmazwan/Date-and-Interval>

<https://github.com/zairulmazwan/emailPHP.git>

<https://replit.com/@zairulmazwan/Questions-for-Key>

<https://replit.com/@zairulmazwan/Random>

# **References**

# The Vigenère Cipher Encryption and Decryption, Available at:  <https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html>

PHP script to send messages with WhatsApp Business API, Available at:

<https://medium.com/@256cub/php-script-to-send-messages-with-whatsapp-business-api-a732d5206a0d>

How to Send a Message by WhatsApp API using PHP easily, Available at:

<https://blog.ultramsg.com/send-whatsapp-message-by-whatsapp-api-using-php/>

Programmable Messaging for WhatsApp and PHP Quickstart, Available at:

<https://www.twilio.com/docs/whatsapp/quickstart/php>