Secure Mobile Payment System

Instructor: **Dr. Ammar Odeh**

Prepared by: **Hazem Odeh (20190644), Mutaz Yassin (20180915), Ahmad Younis (20210698), Malik Shadid (20210520)**

Introduction

Mobiles are devices that the majority of people have at hand at all times, making it more convenient to utilize for payments than a credit card or cash. Thus, no matter if you sell your goods or services in-store or online, integrating a way to pay via mobile in addition to the traditional credit card is key for offering an easy-going experience for your clients.

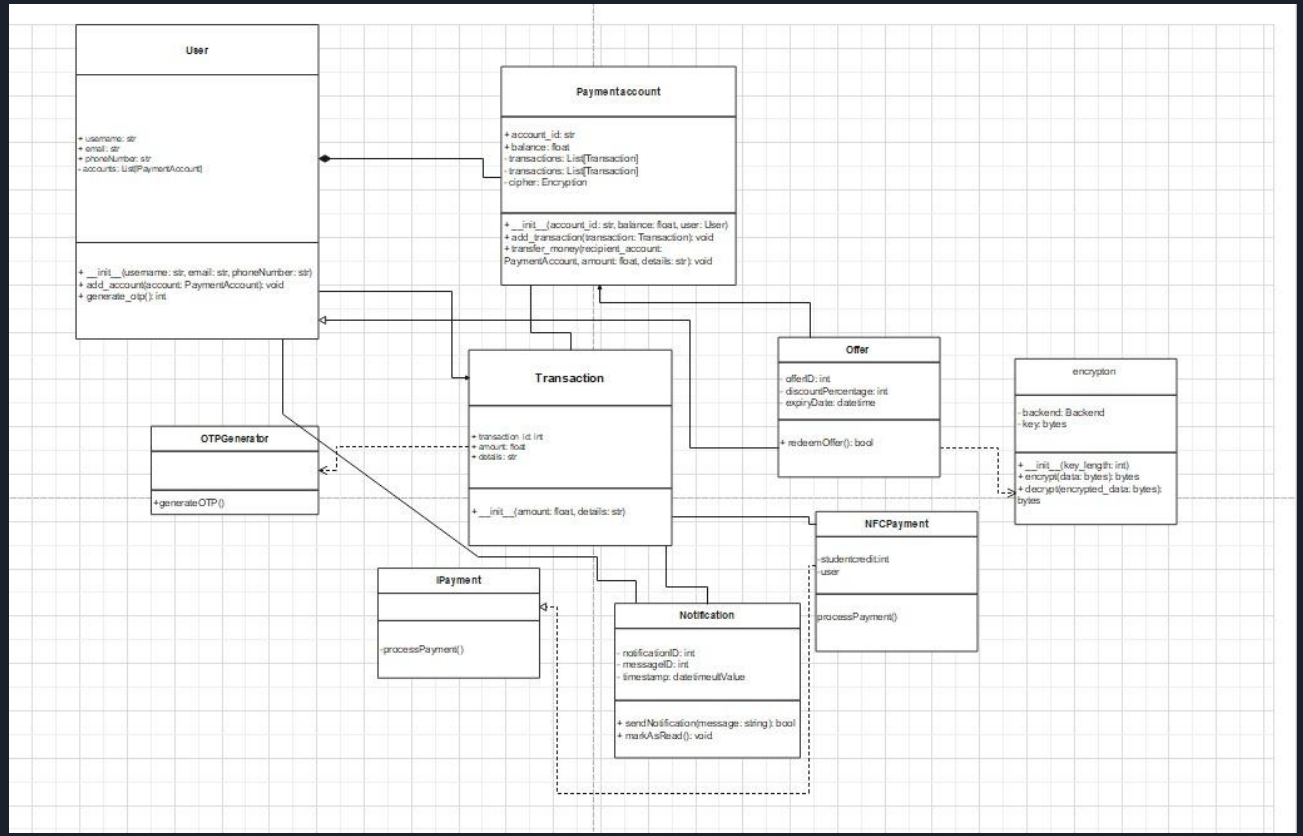
A Mobile Payment System (MPS) allows users to make purchases through their smartphones, tablets, or wearables using all major types of credit and debit cards, alongside bank account information. This is done by having card information stored on a consumer's device. A Secure Payment System (SPS) is an infrastructure integrated into any MPS known today, and ensures financial transactions are safely processed, especially those done online, and is vital for minimizing fraud risks and preventing unauthorized access.

Problem Statement

To shift from the inconvenient use of cash and cards, the implementation of secure mobile payments serves to make financial transactions safer. The most optimal mobile payment system has to have qualities such as security, ease of use, and privacy. Our mobile payment system will use authentication measures to ensure that no one else, except the user, can use the payment system.,This ensures safety even in the case of phone theft. Another aspect is making sure that the user's account information is secure. This can be done by creating a validation function that ensures that the password chosen by the user passes the constraints provided.

Finally, the user interface (UI) of the application must also be easy to use, as plenty of users might not be technologically literate. Finally, the application needs to also be catered in both English and Arabic, as the target audience for this application is Jordanians.

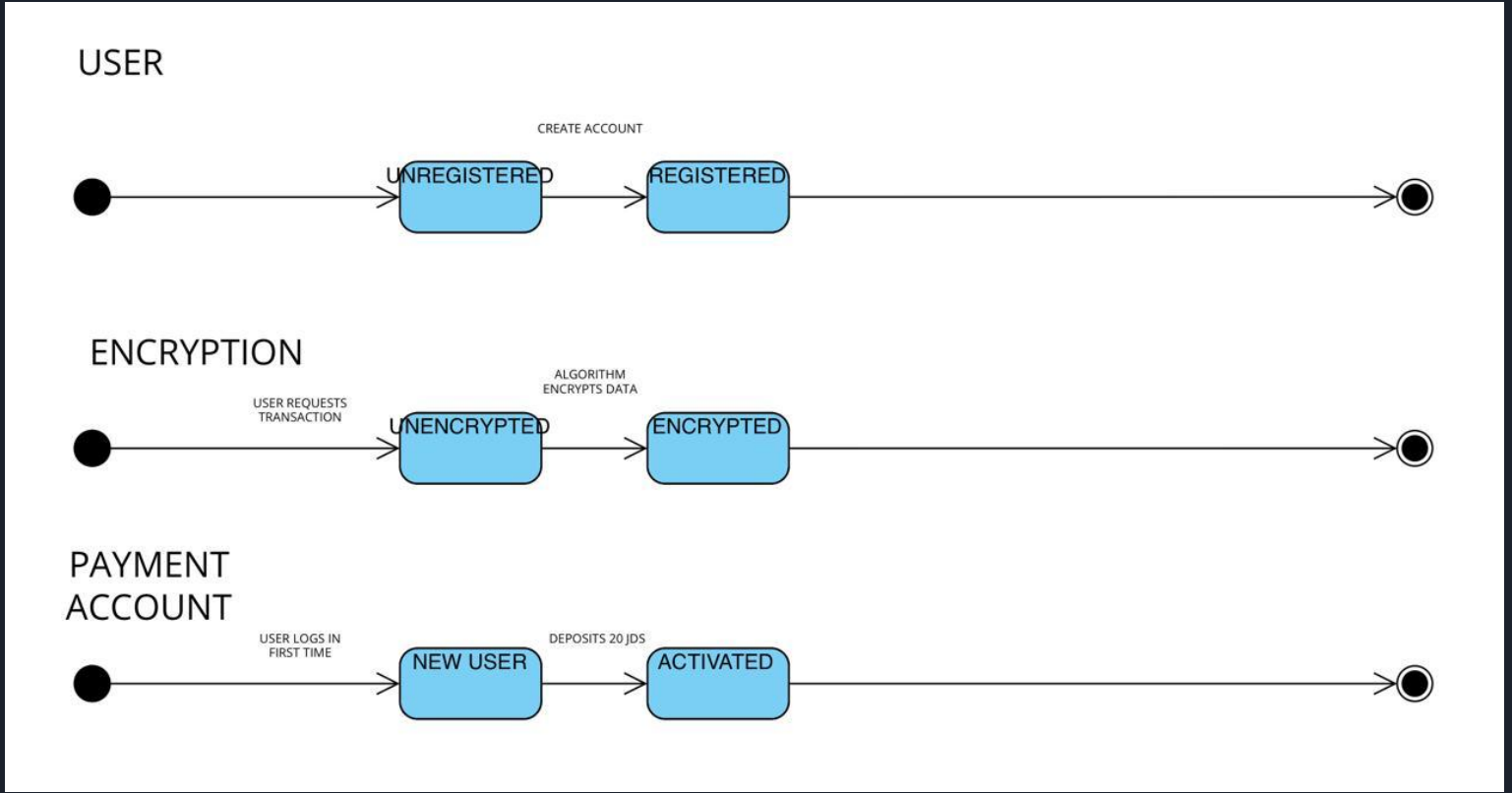
Class Diagram



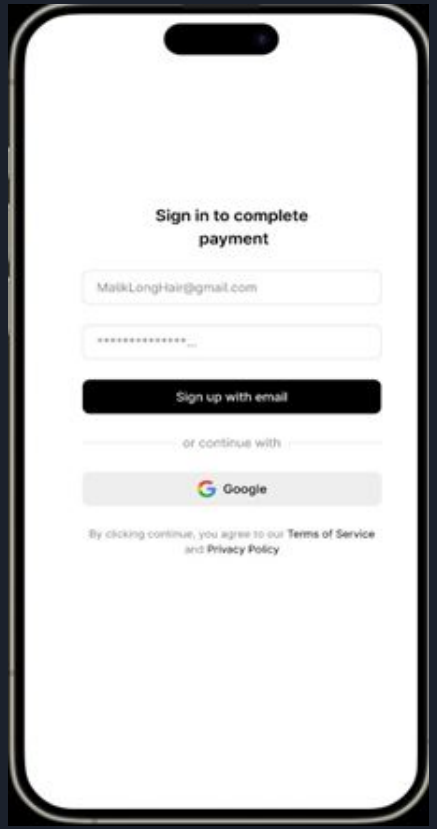
Class Diagram

The classes are users, payment accounts, transactions, offers, notifications, and encryption. Key components include the User class, which manages user details and payment accounts.the PaymentAccount class, which handles account details, transactions, and money transfers; and the Transaction class for recording transaction details. The diagram also incorporates an Offer class for managing promotional discounts, a Notification class for user alerts, and specialized classes like NFCPayment for near-field communication transactions, along with an OTPGenerator for security measures. This comprehensive system is designed to support secure and efficient financial operations through an interconnected class structure.

Sequence Diagram



Our Proposed User Interface



A short scenario:

Hazem went to shop at Mutaz HyperMarket, while checking out, the cashier told him that the visa machine wasn't working and Hazem had no cash. So they used their secure mobile payment system to transfer and receive money.

At first, Hazem has to sign in to transfer money. So he logs in using his email: **MalikLongHair@gmail.com.** After he inserts his password, the user presses the sign in with email button and goes to the next page.

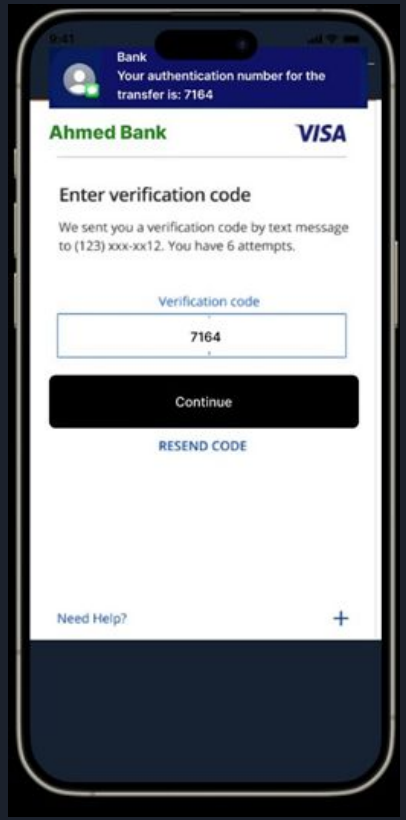
Our Proposed User Interface (cont.)

After signing in, Hazem's user name appears at the top of the page along with the bank's account number and the account balance.

He inserts the recipient's User ID which is Mutaz123 and the amount to be transferred which is 12 JOD.

Hazem then clicks on Send which takes him to the OTP's secure page.

Our Proposed User Interface (cont.)



Ahmed Bank is the bank Hazem is using to transfer his money.

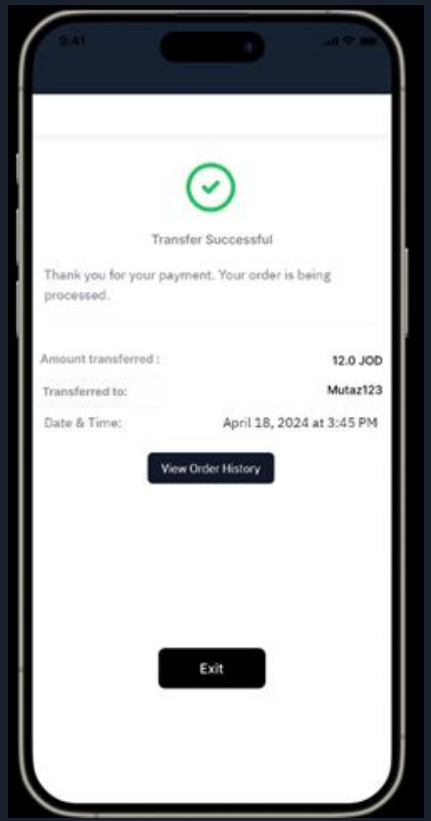
Ahmed Bank uses One Time Password which generates a random password for every transaction and is sent to the User to transfer money securely.

After inserting the OTP number, the user clicks on continue.

If the OTP inserted is valid the user is taken to the next page.

If the OTP is not valid, the user has 5 more attempts before getting blocked until an action has been taken by the bank's Customer Care.

Our Proposed User Interface (cont.)



This page is a confirmation for the user that his transaction was successful after being processed by the bank.

It acts as a receipt for both the merchant and the customer.

The amount transferred is 12 JOD.

Transferred successfully to the recipient with the User ID of Mutaz123

The user then clicks on exit.

Our Proposed User Interface (cont.)

After clicking on exit, the user goes to his messages to check his notifications to see if the bank has sent a notification regarding his transaction or not.

The bank usually sends a notification seconds after the transaction has been done, and the user receives a message that says a specific amount of money has been deducted from your account.

The recipient receives a notification as well the moment his account is credited with the money.

Our Proposed Implementation

```
def __init__(self, key_length=16):
    self.key = os.urandom(key_length)
    self.backend = default_backend()
```

Initializes an instance of the Encryption class with a randomly generated AES key of specified length.

Our Proposed Implementation

```
def encrypt(self, data):
    iv = os.urandom(16)

    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(data) + padder.finalize()
    cipher = Cipher(algorithms.AES(self.key), modes.CBC(iv), backend=self.backend)
    encryptor = cipher.encryptor()
    ct = encryptor.update(padded_data) + encryptor.finalize()
    print("Encrypted data:", iv + ct)
    return iv + ct
```

Demonstrates how to securely encrypt data using AES in CBC mode with PKCS#7 padding, including generation of a random IV (Initialization Vector).

Our Proposed Implementation

```
f decrypt(self, encrypted_data):
  iv = encrypted_data[:16]
  ct = encrypted_data[16:]
  cipher = Cipher(algorithms.AES(self.key), modes.CBC(iv), backend=self.backend)
  decryptor = cipher.decryptor()
  padded_data = decryptor.update(ct) + decryptor.finalize()
  unpadder = padding.PKCS7(128).unpadder()
  return unpadder.update(padded_data) + unpadder.finalize()
```

Outlines the process for decrypting data that was encrypted with AES in CBC mode, showing how to handle the IV and remove padding.

Our Proposed Implementation

```
class User:
    def __init__(self, username, email, phoneNumber):
        self.username = username
        self.email = email
        self.accounts = []
        self.phoneNumber = phoneNumber

    def add_account(self, account):
        self.accounts.append(account)
```

Introduces the User class that manages user details and associated financial accounts.

Our Proposed Implementation

```
def transfer_money(self, recipient_account, amount, details):
    otp = self.user.generate_otp()
    entered_otp = int(input(f"Enter OTP for {self.user.username}: "))

    if entered_otp != otp:
        print("Transaction failed: Invalid OTP")
        return
```

Showcases how the PaymentAccount class manages secure money transfers, including OTP verification and transaction logging with encryption.

Result

```
User 1 balance: 1000
User 2 balance: 500
OTP for Ahmad: 2526
Enter OTP for Ahmad: 2526
Encrypted data: b'M\x8b\xa7#\x1bH\xc8\x7f\xcb\
xbd\x98\x1c\x1d\x1f\x02\xdc\xe1\xed\x81`\xa3\x
bf\x99GZS\x96\xf2dA\xc0/?u%\xca7t\xde\xb7\xd3\
xa6\xe4$\xe3\xb8\xdac\x9d\x05\x1ch\xcf\xb2\x7f
Li(\xb8\xc9\x83\xf2\x96\xbf'
Encrypted data: b'\xf2\x903H?\xe5\xf6d\xfcS\xd
7\x80\xae\x19\x9c\xf0\xfdD\xfb"5\xcf\xbe=;\x1d
\x05oI\xcb\xeb\xedP\xe0q\xe9\x99.G\x92\x9a\xcd
H\xe4\xef\x05*G\'\x05\xf9\n\xf4\x06\xf2y\xc1\x
db0\x1ct\x1c7\x84'
Transaction successful: 200 transferred from A
ccount acc123 to Account acc456
User 1 balance: 800
User 2 balance: 700
```

Success

```
User 1 balance: 1000
User 2 balance: 500
OTP for Ahmad: 4254
Enter OTP for Ahmad: 5646
Transaction failed: Invalid OTP
User 1 balance: 1000
User 2 balance: 500
```

Fail

Compare with Prior Work

Method	Strengths	Weaknesses	Tokenization	NFC Support*	Security
Mobile P2P (PayPal, Zelle, Clip)	- Convenient. - Often integrated with social media networks for seamless transactions.	- May have transaction limits and impose fees. - Concerns about privacy and security with personal information sharing.	Yes	Sometimes (depending on platforms)	Strong

Our System:

- Offers Strong Security
- Uses the encryption algorithm AES instead of Tokenization
- Doesn't impose other fees whatsoever (i.e: Transaction fees)
- Convenient to use especially with technologically illiterate users.

Future Work

Here are some suggested improvements for our system:

- Support for more languages.
- Virtual card for online purchases.
- Points system such that the more users send & receive payments the more they are rewarded with points, which can be used to pay
- Biometric authentication
- Notifications to remind people of basic cyber security measures to be aware of.
- Constantly improving & simplifying our interface.

Conclusion

- A secure mobile payment system's main security aspects lies in the encryption, and the handling of the sensitive data.
- Our improved AES encryption algorithm allows more security, by encrypting both the sender and receiver's data
- Our application is the best secure mobile payment system, ever.