# ATM Interface

Project Report

By:

**Onkar Uday Deokate**

**219303061**

Contact Information:

[officialonkardeokate@gmail.com](mailto:officialonkardeokate@gmail.com)

[**https://www.linkedin.com/in/onkardeokate/**](https://www.linkedin.com/in/onkardeokate/)

**9420920656**

A close up of a logo

Description automatically generated

2021-2025

Department of Computer and Communication Engineering

School of Computing and Intelligent Systems

Manipal University Jaipur

VPO. Dehmi Kalan, Jaipur, Rajasthan, India – 303007

## Table of Contents

1. Introduction
2. System Requirements
3. Components
   1. Account Class
   2. ATM Class
   3. ATMInterface Class
4. Usage Instructions
5. Code Examples
6. Error Handling
7. Additional Notes

## 1. Introduction

This documentation provides an overview of the ATM Interface Code implemented in Python. The system features a graphical user interface (GUI) built using the `tkinter` library, allowing users to perform ATM transactions such as deposits, withdrawals, transfers, and balance checks. The application also supports account management and transaction history tracking.

## 2. System Requirements

1. Python: Ensure Python 3.6 or higher is installed.
2. tkinter: The GUI library is included with Python standard libraries.
3. JSON: Used for data storage; no additional installations required.
4. Operating System: Compatible with Windows, macOS, and Linux.

## 3. Components

### Account Class

Purpose

The `Account` class represents an individual bank account. It manages account details, transactions, and balance.

**Attributes**

* + - `account\_number`: A unique identifier for the account (string).
    - `holder\_name`: Name of the account holder (string).
    - `balance`: Current balance in the account (float).
    - `transaction\_history`: List of transactions performed on the account (list of strings).

**Methods**

* + - `\_\_init\_\_(self, account\_number, holder\_name, balance=0)`
* Initializes an `Account` instance with the specified account number, holder's name, and an optional initial balance.
  + - `deposit(self, amount)`
* Deposits the specified amount into the account. Returns `True` if successful, `False` otherwise.
  + - `withdraw(self, amount)`
* Withdraws the specified amount from the account if sufficient funds are available. Returns `True` if successful, `False` otherwise.
  + - `transfer(self, amount, target\_account)`
* Transfers the specified amount to another `Account` instance. Returns `True` if successful, `False` otherwise.
  + - `get\_balance(self)`
* Returns the current balance of the account.
  + - `get\_transaction\_history(self)
* Returns the transaction history of the account.

**Example**

account = Account("12345", "John Doe", 1000)

account.deposit(200)

account.withdraw(50)

print(account.get\_balance()) # Outputs: 1150

### ATM Class

**Purpose**

The `ATM` class manages the overall ATM system, including account operations and data persistence.

**Attributes**

- `accounts`: Dictionary where keys are account numbers and values are `Account` objects.

**Methods**

- `\_\_init\_\_(self)`

Initializes the ATM system and loads account data from the `accounts.json` file.

- `load\_accounts(self)`

Loads account data from a JSON file. Returns a dictionary of accounts.

- `save\_accounts(self)`

Saves the current state of accounts to the `accounts.json` file.

- `authenticate\_user(self, account\_number)`

Authenticates a user by checking if the account number exists and is valid.

- `deposit(self, account\_number, amount)`

Handles deposit transactions for the specified account.

- `withdraw(self, account\_number, amount)`

Handles withdrawal transactions for the specified account.

- `transfer(self, from\_account\_number, to\_account\_number, amount)`

Handles fund transfers between accounts.

- `get\_balance(self, account\_number)`

Retrieves the balance of the specified account.

- `get\_transaction\_history(self, account\_number)`

Retrieves the transaction history of the specified account.

**Example**

atm = ATM()

atm.deposit("12345", 100)

print(atm.get\_balance("12345")) # Outputs: 1100

### ATMInterface Class

**Purpose**

The `ATMInterface` class provides a user-friendly graphical interface for interacting with the ATM system using `tkinter`.

**Methods**

- `\_\_init\_\_(self, atm)`

Initializes the GUI and sets up the main window for the ATM system.

- `create\_login\_screen(self)`

Displays the login screen for user authentication.

- `create\_main\_menu(self)`

Displays the main menu with options for transactions.

- `create\_deposit\_screen(self)`

Displays the deposit screen to input deposit amounts.

- `create\_withdraw\_screen(self)`

Displays the withdrawal screen to input withdrawal amounts.

- `create\_transfer\_screen(self)`

Displays the transfer screen to input transfer details.

- `create\_balance\_screen(self)`

Displays the screen showing the current account balance.

- `create\_transaction\_history\_screen(self)`

Displays the transaction history screen for the account.

**Event Handlers**

- `handle\_login(self, account\_number)`

Processes user login requests and transitions to the main menu.

- `handle\_deposit(self, amount)`

Processes deposit requests and updates the account balance.

- `handle\_withdraw(self, amount)`

Processes withdrawal requests and updates the account balance.

- `handle\_transfer(self, to\_account\_number, amount)`

Processes transfer requests and updates balances accordingly.

- `handle\_check\_balance(self)`

Displays the current balance of the account.

- `handle\_view\_transaction\_history(self)`

Displays the transaction history of the account.

**Example**

if \_\_name\_\_ == "\_\_main\_\_":

atm = ATM() # Initialize the ATM system

atm\_interface = ATMInterface(atm)

## 4. Usage Instructions

1. Ensure Prerequisites

- Verify that Python 3.6+ is installed on your system.

- Confirm that `tkinter` is available (usually included with Python).

2. Running the Application

- Save the code in a file named `atm\_interface.py`.

- Open a terminal or command prompt.

- Navigate to the directory containing the script.

- Run the script using the command: `python atm\_interface.py`.

3. Interacting with the Application

- Follow the on-screen instructions to log in or create an account.

- Use the provided buttons to perform transactions or check account details.

**Code Example**

# Example of creating an account and performing a deposit

atm = ATM()

atm.create\_account("12345", "John Doe")

atm.deposit("12345", 100)

print(atm.get\_balance("12345")) # Should print 100