**Project Documentation: Python Banking System**

**Table of Contents**

1. **Introduction**
   * Purpose and Scope
   * Features Overview
2. **System Overview**
   * Architecture Overview
   * User Roles and Interactions
3. **Setup and Dependencies**
   * Python Version
   * Required Libraries
4. **File Structure**
   * Directory Overview
   * File Organization
5. **Functionality Description**
   * Admin Functions
     + Create Staff Account
     + Edit Staff Details
     + List Staff
   * Customer Functions
     + Create Customer Account
     + Edit Customer Details
     + Deposit and Withdraw
     + Generate Customer Statement
   * General Functions
     + File Handling
     + Error Handling
6. **Code Samples**
   * Highlighted Code Snippets
7. **Usage Instructions**
   * How to Run the Program
   * Step-by-Step Guide for Admin and Customers
8. **Testing**
   * Testing Procedures
   * Known Issues
9. **Conclusion**
   * Achievements and Lessons Learned
   * Future Improvements
10. **References**

**1. Introduction**

The Python Banking System is a command-line application designed to simulate basic banking operations. It provides functionalities for both administrators and customers, enabling account management, transaction handling, and statement generation.

**2. System Overview**

**Architecture Overview**

The system utilizes a modular approach with Python’s file handling capabilities. It separates functionalities into admin and customer roles, ensuring security and ease of management.

**User Roles and Interactions**

* **Admin:** Manages staff accounts, customer accounts, and oversees system operations.
* **Customer:** Accesses personal accounts, performs transactions, and manages account details.

**3. Setup and Dependencies**

The project requires Python 3.x and uses standard libraries such as os and datetime for file handling and date manipulation.

**4. File Structure**

**Directory Overview**

The project directory is structured as follows:

* **main.py**: Entry point for the application.
* **Customers/**: Directory storing customer data and transactions.
* **Staff/**: Directory for staff information and accounts.
* **super.txt**: Configuration file for admin credentials.

**5. Functionality Description**

**Admin Functions**

**Create Staff Account**

Admins can create new staff accounts dynamically, assigning unique IDs and managing staff details such as name, address, contact, and citizenship.

Example:

python

Copy code

def create\_staff\_account():

# Code snippet for creating a new staff account

**Edit Staff Details**

Allows admins to modify staff information such as address, contact number, and password.

Example:

python

Copy code

def edit\_staff\_details(staff\_id):

# Code snippet for editing staff details

**List Staff**

Displays a list of all staff members currently registered in the system.

Example:

python

Copy code

def list\_all\_staff():

# Code snippet for listing all staff members

**Customer Functions**

**Create Customer Account**

Enables the creation of new customer accounts, distinguishing between current and savings account types.

Example:

python

Copy code

def create\_customer\_account(account\_type):

# Code snippet for creating a new customer account

**Deposit and Withdraw**

Customers can deposit and withdraw funds from their accounts, with real-time balance updates and transaction logging.

Example:

python

Copy code

def withdraw(customer\_id, amount):

# Code snippet for processing a withdrawal

**Generate Customer Statement**

Generates a transaction statement for a specified date range, providing an overview of account activities.

Example:

python

Copy code

def generate\_statement(customer\_id, start\_date, end\_date):

# Code snippet for generating a customer statement

**General Functions**

**File Handling**

Includes functions for reading from and writing to text files to manage account and transaction data.

Example:

python

Copy code

def read\_customer\_data(customer\_id):

# Code snippet for reading customer data from a file

**Error Handling**

Ensures robustness by implementing error handling for file operations, user inputs, and authentication.

Example:

python

Copy code

try:

# Code snippet with error handling

except FileNotFoundError as e:

print(f"Error: {e}")

**6. Code Samples**

**Highlighted Code Snippets**

python

Copy code

# Example of a key function or method

**7. Usage Instructions**

**How to Run the Program**

To start the Python Banking System:

1. Open a terminal or command prompt.
2. Navigate to the project directory.
3. Run python main.py to launch the application.

**Step-by-Step Guide for Admin and Customers**

* **Admin:**
  + Login using admin credentials.
  + Choose from various admin functions (create staff account, edit details, etc.).
  + Follow prompts to perform desired actions.
* **Customer:**
  + Login using customer account number and password.
  + Select from available options (deposit, withdraw, check balance, etc.).
  + Enter necessary details as prompted to complete transactions.