**Bank Console Application**

**Overview**

This Bank Console Application is a Python-based console application that allows users to perform basic banking operations. The application supports account creation, password updates, account login, balance viewing, money transfers, and account logout. Account data is stored in an Excel file, making it easy to manage and access user information.

**Project Structure**

The project is divided into three main files:

1. **main.py**: The entry point of the application, responsible for displaying menus, accepting user inputs, and calling appropriate methods based on user actions.
2. **bank\_app.py**: Contains the **BankApp** class, which encapsulates the main functionalities of the banking system, including account management and user operations.
3. **excel\_storage.py**: Contains the **ExcelStorage** class, responsible for loading and saving account data to and from an Excel file.
4. **console\_ui.py**: Contains infos to display to users.

**Class Descriptions**

**BankApp**

The **BankApp** class is responsible for managing account data and user operations. It interacts with the **ExcelStorage** class to load and save account information.

Methods

* **\_\_init\_\_(self, excel\_filename)**: Initializes a new **BankApp** instance, loads account data from the specified Excel file, and sets the current account to **None**.
* **save\_accounts(self)**: Saves the account data to the Excel file.
* **create\_account(self, username, password)**: Creates a new account with the specified username and password if the username is not already taken. Initializes the account balance to 1000.
* **update\_password(self, old\_password, new\_password)**: Updates the password of the current account if the provided old password is correct.
* **reload\_accounts(self)**: Reloads account data from the Excel file.
* **login\_account(self, username, password)**: Logs in to the specified account if the username and password are correct.
* **view\_balance(self)**: Displays the current account balance if a user is logged in.
* **send\_money(self, to\_username, amount)**: Sends money from the current account to the specified recipient account if the current account has sufficient balance.
* **logout(self)**: Logs out from the current account.

**ExcelStorage**

The **ExcelStorage** class is responsible for loading and saving account data to and from an Excel file.

Methods

* **\_\_init\_\_(self, filename)**: Initializes a new **ExcelStorage** instance with the specified Excel file.
* **load\_accounts(self)**: Loads account data from the Excel file and returns a dictionary containing the accounts.
* **save\_accounts(self, accounts)**: Saves the account data to the Excel file.

**Console\_ui**

Methods

* **display\_menu\_logged\_out()** function displays a menu for users who are not logged in and allows them to create an account or login, or exit the application. It returns the user's choice as a string.
* **display\_menu\_logged\_in()** function displays a menu for users who are logged in and allows them to perform various operations such as updating their password, viewing their balance, sending money to other accounts, logging out or exiting the application. It returns the user's choice as a string.

These functions are called from the main.py file to display the menus to the user and get their input.

**Usage**

To run the Bank Console Application, follow these steps:

1. Ensure that you have Python 3.6 or higher installed on your system.
2. Install the required dependencies by running **pip install pandas openpyxl**.
3. Run the **main.py** file to start the application.

After starting the application, you will be presented with a menu of options. To perform an action, enter the corresponding number and provide any required information. To exit the application, select the "Exit" option from the main menu.

Please note that when updating account information directly in the Excel file, you should close the application, make changes, save the file, and then restart the application to ensure that the changes are reflected in the program.

**Limitations and Possible Improvements**

The Bank Console Application is a basic implementation of a banking system, and as such, it has several limitations. Some of the limitations include:

* Lack of security features such as two-factor authentication or encryption of user data.
* Limited error handling, which can result in unexpected crashes or errors when incorrect inputs are provided.
* The application does not support multiple users concurrently. Only one user can be logged in at a time.
* There are limited options for user account management, such as changing user information or deleting accounts.

To improve the application, some possible enhancements include:

* Adding security features such as two-factor authentication, encryption, or multi-factor authentication to ensure that user data is protected.
* Enhancing error handling to provide more informative error messages and prevent unexpected crashes.
* Supporting multiple users concurrently, allowing multiple users to log in and manage their accounts.
* Adding more options for user account management, such as changing user information, deleting accounts, or viewing transaction history.

**Conclusion**

The Bank Console Application is a simple implementation of a banking system that allows users to perform basic banking operations. It is an excellent starting point for those learning Python programming or exploring basic banking concepts. The application can be improved by adding more features and enhancing the user experience.