

Bank Service Management

Fast API + MongoDB Based
Bank Service Management
System



01 Introduction

- The Bank Service System manages customers, accounts, transactions, and other banking services.
- CRUD operation for customers and accounts
- Designed for developers, testers, and bank staff



02 System Overview

Modules:

- Customer Management
- Account Management
- Transactions
- Balance Inquiry
- Authentication & Authorization
- Reports & Audit Logs



03 Features

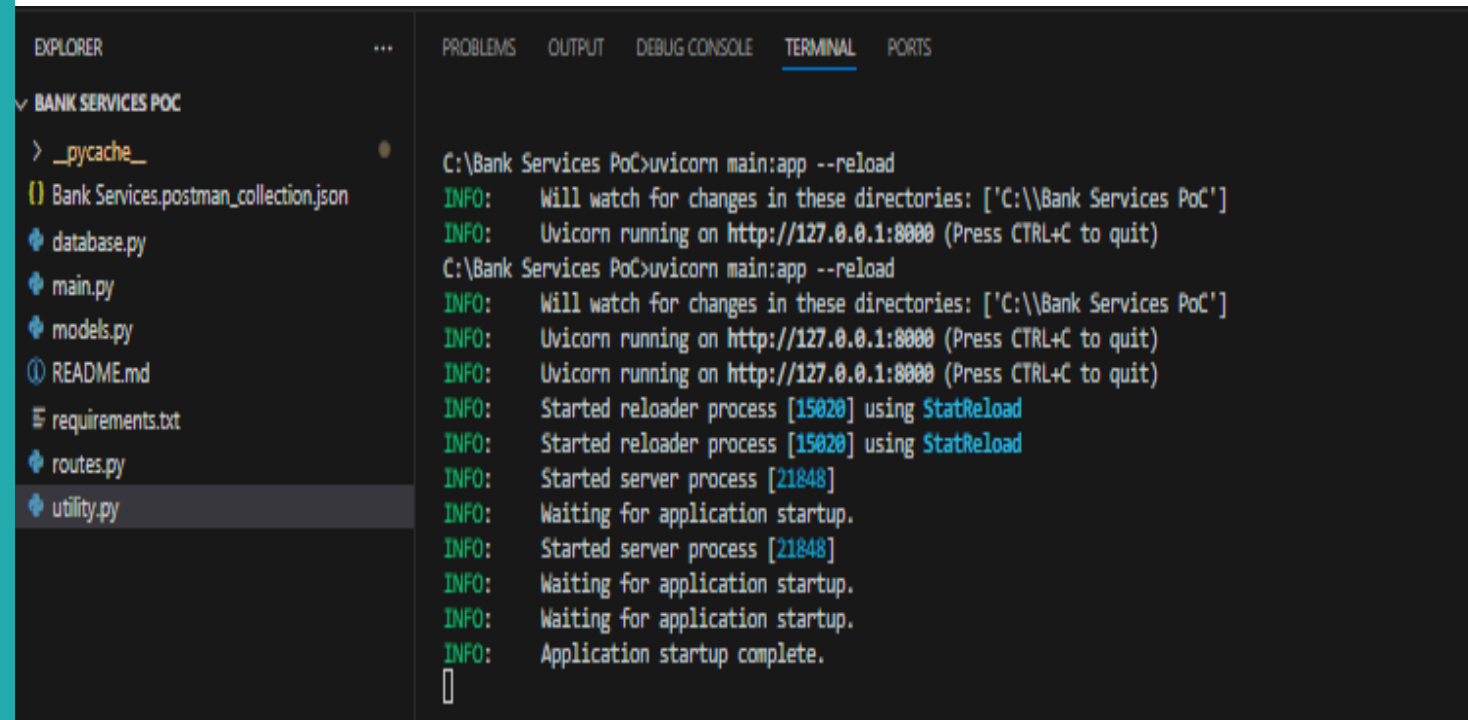
- Customer CRUD
- Account Open/PIN Change/KYC Update
- Transactions: Deposit, Withdraw, Transfer
- Balance Inquiry
- Loans apply / Loans repay



Bank Service Management

- ❖ Database.py
- ❖ Bank Service Postman_collection.json
- ❖ Main.py
- ❖ Readme.md
- ❖ Requirement.txt
- ❖ Router.py
- ❖ Utility.py

04 Project Structure

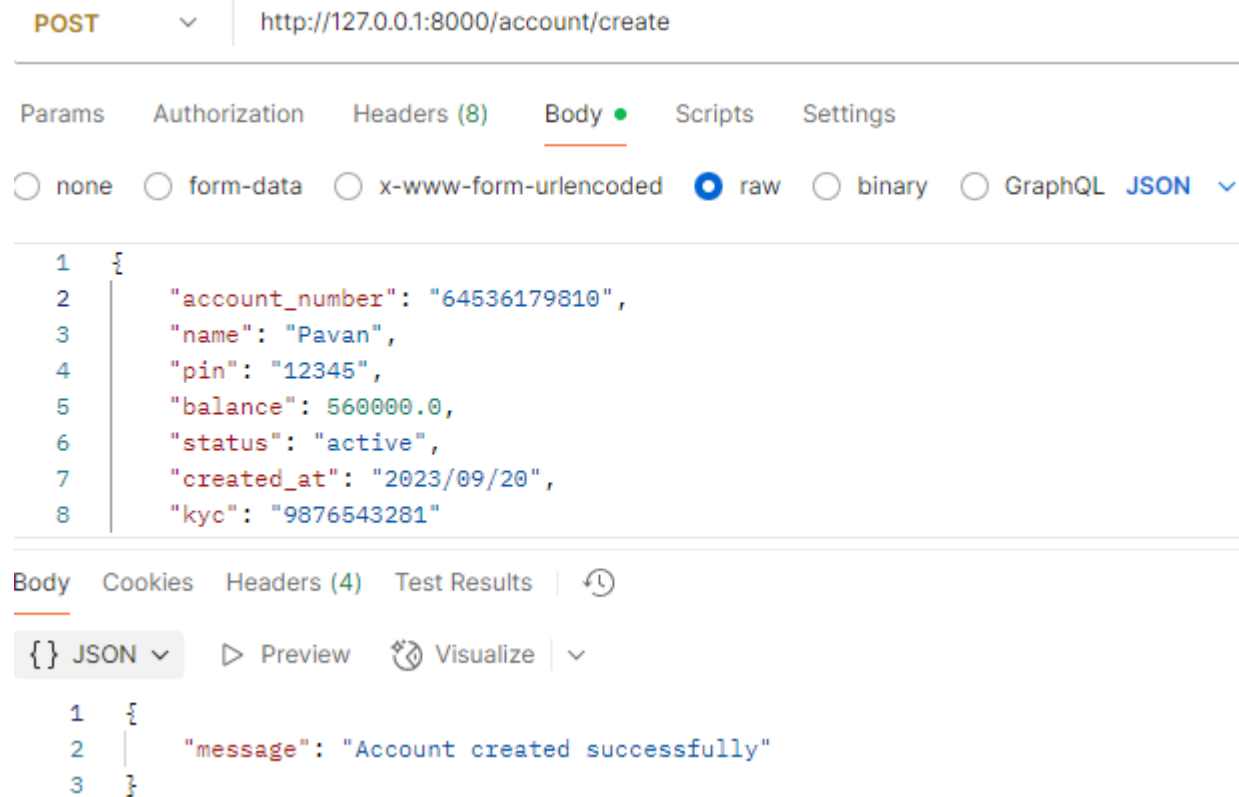


```
EXPLORER
BANK SERVICES POC
  _pycache_
  Bank Services.postman_collection.json
  database.py
  main.py
  models.py
  README.md
  requirements.txt
  routes.py
  utility.py

TERMINAL
C:\Bank Services PoC>uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\Bank Services PoC']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
C:\Bank Services PoC>uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\Bank Services PoC']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [15020] using StatReload
INFO: Started reloader process [15020] using StatReload
INFO: Started server process [21848]
INFO: Waiting for application startup.
INFO: Started server process [21848]
INFO: Waiting for application startup.
INFO: Waiting for application startup.
INFO: Application startup complete.
```



05 Postman Testing



Step1 : paste the URI & set to **Post**

Step2 : Body > raw > give the create the data as for program

Step3 : Send the request after we get Test Results (**200 OK**)
{**"message"** : **"Account created Successfully"** }

Step4 : save the results in collections



Security

- JWT Authention
- Role-based Authorization
- Data Encryption



06 MongoDB Data Storage

DB = Bank Service

Collection :

- Account collection
- Loans collection
- Transcations collection

After testing APIs in Postman, we can authomatically see the data uploaded into MongoDB Collections

Error Handling

- 400 Bad Request
- 401 Unauthorized
- 404 Not Found
- 500 Server Error



Conclusion

The **Bank Service API with MongoDB** provides a complete and modular banking solution, supporting **account management, transactions, and loan services**.

- **Account Services** – Create accounts, update KYC, change PINs, and securely check balances.
- **Transaction Services** – Deposit, withdraw, and transfer money with **real-time transaction logging**.
- **Loan Services** – Apply for loans, track outstanding dues, and make repayments.
- **Transaction History** – Every operation (deposit, withdrawal, transfer, repayment) is stored in MongoDB for **audit and compliance**.

Using **FastAPI** ensures high performance and clean API design, while **MongoDB** provides scalability and flexibility for handling large volumes of customer and transaction data.





**Thank
You**