Study Case

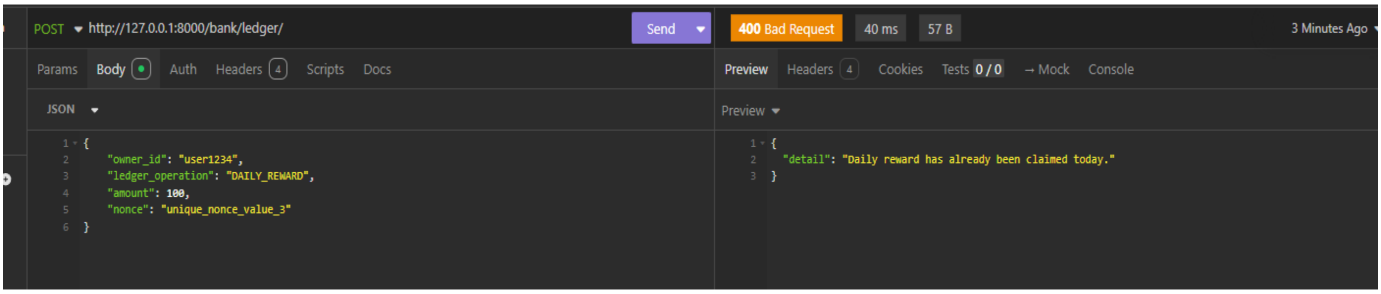
Over the past week, I have been developing a FastAPI application within a monorepo structure, integrating it with a PostgreSQL database. Initially, I utilized SQLAlchemy to define the database schema. Subsequently, I recognized the importance of incorporating Alembic for managing database migrations, which I tried to do. Throughout this process, I employed Uvicorn to run the FastAPI application and used Insomnia to test the endpoints. Despite the initial challenges, I have successfully achieved the desired outcomes.

The project works with following commands:

venv\Scripts\Activate

uvicorn bank_app.main:app –reload

I checked my logic and if it worked with Insomnia by checking every possible outcome I wrote.



POST http://127.0.0.1:8000/bank/ledger/

```
{
    "owner_id": "user1234",
    "ledger_operation": "DAILY_REWARD",
    "amount": 100,
    "nonce": "unique_nonce_value_3"
}
```

400 Bad Request 40 ms 57 B

```
{
  "detail": "Daily reward has already been claimed today."
}
```

I added BaseLedgerOperation to my apps AdditionalOperations and it worked fine

```
from core.base_ledger_operation import BaseLedgerOperation
from enum import Enum

class AdditionalOperations(str, Enum):
    WITHDRAW_CREDIT = "WITHDRAW_CREDIT"

# Combine enums
combined_members = {**BaseLedgerOperation.__members__,
**AdditionalOperations.__members__}
BankOperation = Enum('BankOperation', combined_members)
```

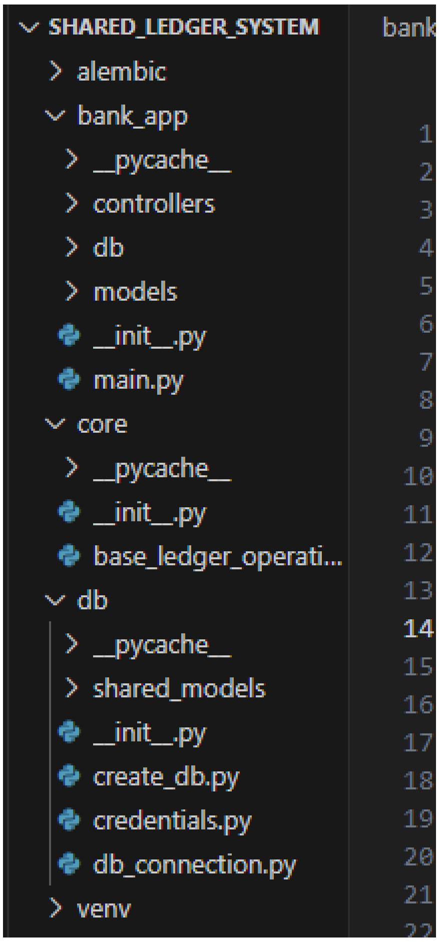
I created the database table with the following code.

```
from db_connection import Base, engine
from shared_models.Ledger import Ledger


Base.metadata.create_all(engine)
```

I wrote the url of my PostgreSQL database in alembic.ini and db\credentials.

```
#Class because additional properties can be defined
class Credentials:
    _DATABASE_URL =
"postgresql://postgres:Uygr12_!@localhost/monorepo_ledger_db"

    #read only
    @property
    def DATABASE_URL(self) -> str:
       return self._DATABASE_URL
```

My structure was this:



```
SHARED_LEDGER_SYSTEM
  alembic
  bank_app
    __pycache__
    controllers
    db
    models
    __init__.py
    main.py
  core
    __pycache__
    __init__.py
    base_ledger_operati...
  db
    __pycache__
    shared_models
    __init__.py
    create_db.py
    credentials.py
    db_connection.py
  venv
```

I only wrote controller and db layer, because I do not think a service layer was needed for this project. In db repository I stored, create_db (created my table with create_db before I knew that I needed to use Alembic for table creation), database connection, and database credentials there.

I stored models that only my app needed in models folder under bank_app.