Bank Account Withdrawal - Implementation Overview

Software Engineer: Jeshen Appanna

Approach Outline

I implemented the solution in Python because it is my primary language. I used Kafka as the event-driven tool since I work with it regularly, and it can run locally. The core business functionality—handling withdrawals and event notifications—remains unchanged.

Key improvements:

- Modularized design
- Event-driven architecture with Kafka
- Transactional integrity using SQLAlchemy
- Logging for tracking transactions
- Kafka publishing with retries
- Portability with SQLite

Directory Structure

The project is structured to ensure clear separation of concerns:

```
bank-account-withdrawal/
├── src/
│   ├── config.py              # Configuration settings and logging
│   ├── controller.py          # Handles withdrawal logic
│   ├── events.py              # Defines withdrawal event structure
│   ├── main.py                # CLI interface for withdrawals
│   ├── models.py              # Database models with SQLAlchemy
│   ├── producer.py            # Kafka event producer with retries
├── requirements.txt           # Dependencies for the project
├── Makefile                   # Commands to set up and run the project
├── activate_venv.sh           # Script to activate the virtual
environment
```

Implementation Choices

1. Database and Transaction Handling

- Used SQLAlchemy ORM for transactions.
- `with_for_update()` prevents race conditions.
- Rollback mechanism ensures consistency.

2. Event Notification System (Kafka)

- Kafka was chosen because it's the event-driven tool I work with.
- Runs locally, simplifying development.
- Uses exponential backoff and retries for reliability.

3. Logging

- Configured structured logging globally.
- Logs track transaction outcomes and Kafka publishing.

4. Code Structure

- Used Enums (`WithdrawalStatus`) for status tracking.
- Organized components into separate modules.

Library Usage

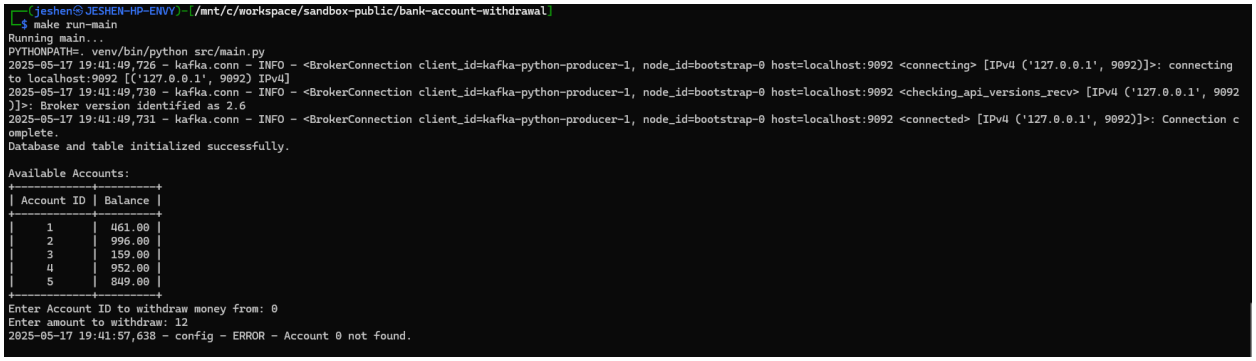
Library	Purpose
`sqlalchemy`	ORM for database transactions
`kafka-python`	Kafka producer for event notifications
`pretty-table`	Formats account balances for CLI display

Clarifications

- `sqlalchemy.dialects.sqlite.insert`: Inserts account data while ignoring duplicates (`OR IGNORE`).
- `KafkaProducer.flush()`: Ensures event delivery before continuing execution.

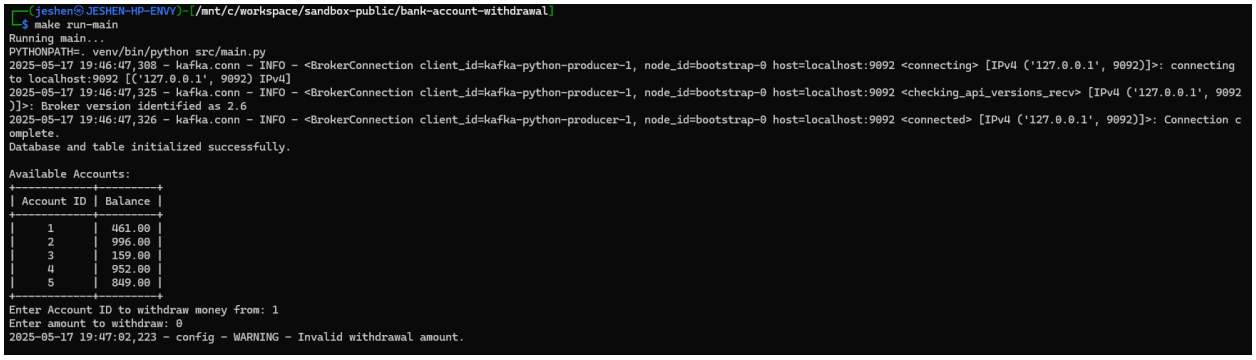
Testing

Test-1: Try withdrawing money from an account that does not exist



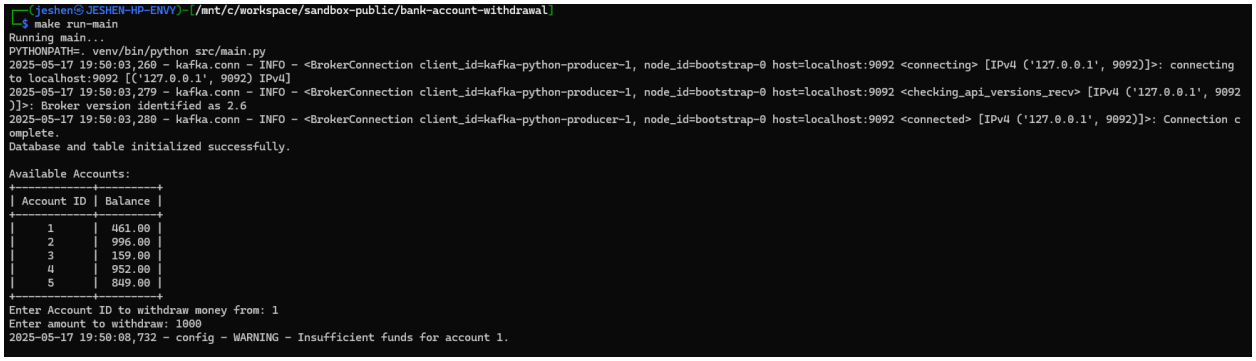
Status: PASS

Test-2: Try withdrawing an invalid amount from an existing account



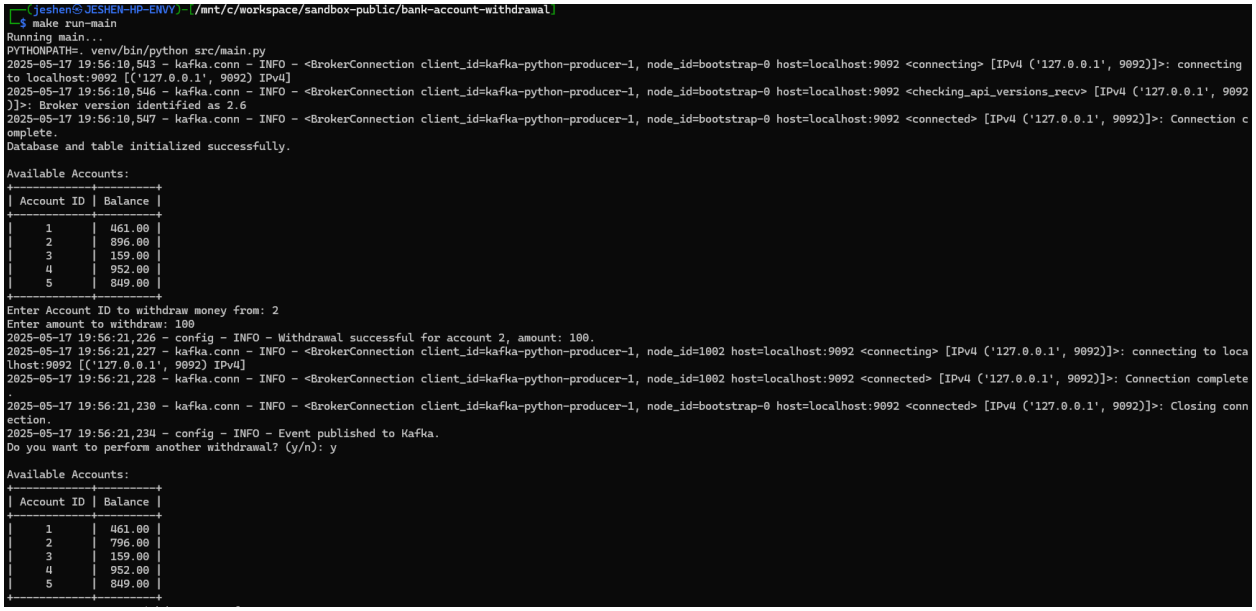
Status: PASS

Test-3: Try withdrawing a large amount from an existing account



Status: PASS

Test-4: Try withdrawing a valid amount from an existing account



Status: PASS

- Bank Account Balance Updated
- Kafka Event Published