README

Starting Application

To run the application as a jar file unzip the submitted zip file and find billing-0.0.1-SNAPSHOT.jar file. Run the command

```
java -jar billing-0.0.1-SNAPSHOT.jar
```

I have used java version 23. But it should run with a lower version as well.

If you want to use Docker image, I have uploaded it to skhandekar/billing repository.

To download it use

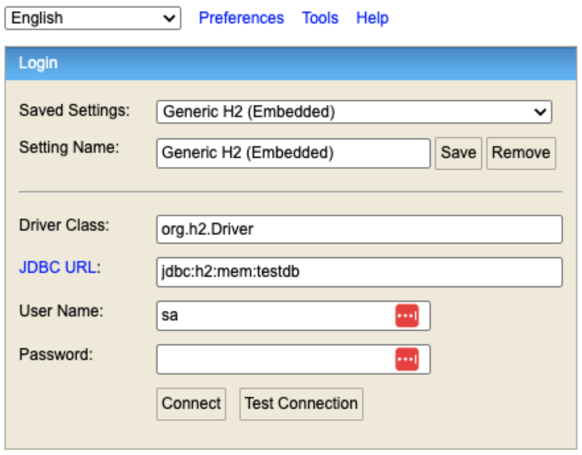
```
docker pull skhandekar/billing:latest
```

To run use use

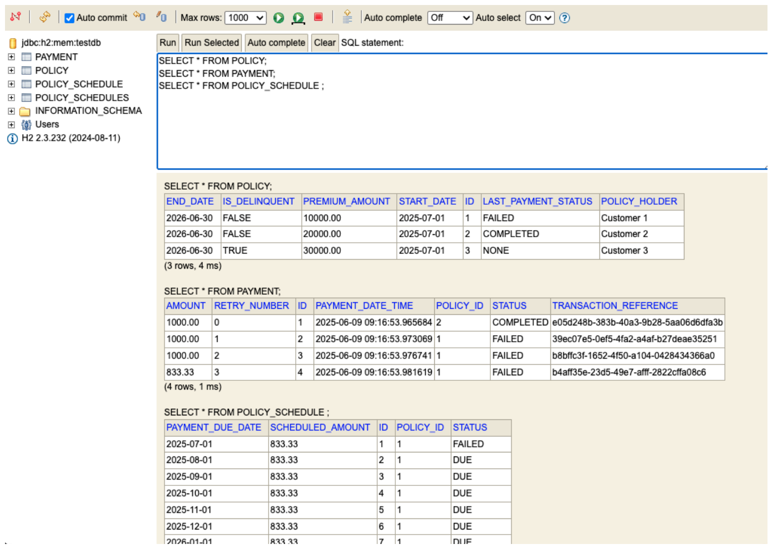
```
 docker run -p 8080:8080 skhandekar/billing:latest
```

Running application

The application starts up and runs a test suite. Code can be found in BillingApplication.java. The application uses an in memory H2 database. You can open the database console to check the schema. To open the console, navigate to http://localhost:8080/h2-console/
There is no password set.



Saved Settings: Generic H2 (Embedded)
Setting Name: Generic H2 (Embedded)
Driver Class: org.h2.Driver
JDBC URL: jdbc:h2:mem:testdb
User Name: sa
Password:


Auto commit ✓ Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:testdb
- PAYMENT
- POLICY
- POLICY_SCHEDULE
- POLICY_SCHEDULES
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

SQL statement:

```
SELECT * FROM POLICY;
SELECT * FROM PAYMENT;
SELECT * FROM POLICY_SCHEDULE ;
```

SELECT * FROM POLICY;

END_DATE	IS_DELINQUENT	PREMIUM_AMOUNT	START_DATE	ID	LAST_PAYMENT_STATUS	POLICY_HOLDER
2026-06-30	FALSE	10000.00	2025-07-01	1	FAILED	Customer 1
2026-06-30	FALSE	20000.00	2025-07-01	2	COMPLETED	Customer 2
2026-06-30	TRUE	30000.00	2025-07-01	3	NONE	Customer 3

(3 rows, 4 ms)

SELECT * FROM PAYMENT;

AMOUNT	RETRY_NUMBER	ID	PAYMENT_DATE_TIME	POLICY_ID	STATUS	TRANSACTION_REFERENCE
1000.00	0	1	2025-06-09 09:16:53.965684	2	COMPLETED	e05d248b-383b-40a3-9b28-5aa06d6dfa3b
1000.00	1	2	2025-06-09 09:16:53.973069	1	FAILED	39ec07e5-0ef5-4fa2-a4af-b27deae35251
1000.00	2	3	2025-06-09 09:16:53.976741	1	FAILED	b8bffc3f-1652-4f50-a104-0428434366a0
833.33	3	4	2025-06-09 09:16:53.981619	1	FAILED	b4aff35e-23d5-49e7-afff-2822cffa08c6

(4 rows, 1 ms)

SELECT * FROM POLICY_SCHEDULE ;

PAYMENT_DUE_DATE	SCHEDULED_AMOUNT	ID	POLICY_ID	STATUS
2025-07-01	833.33	1	1	FAILED
2025-08-01	833.33	2	1	DUE
2025-09-01	833.33	3	1	DUE
2025-10-01	833.33	4	1	DUE
2025-11-01	833.33	5	1	DUE
2025-12-01	833.33	6	1	DUE
2026-01-01	833.33	7	1	DUE

Accessing APIs

Fetch a policy
http://localhost:8080/api/policy/1

Fetch only the schedule
http://localhost:8080/api/policy/1/schedule

Fetch delinquent policy
http://localhost:8080/api/policy/delinquent

Testing of recording payments and retrying failed payments is done directly via Billing Application. The results can be verified in console output and browsing the schema.

You can also use Postman to post http requests. E.g.
POST : http://localhost:8080/api/policy/2/payment
Body

```
{
  "amount": "1000",
  "paymentMethodType" : "Credit Card"
}
```

The above request will post a successful payment for policy ID 2

Body

```
{
  "amount": "1000",
  "paymentMethodType" : "Fail"
}
```

The above request will post a failed payment for policy ID 2. When the payment type is set to Failed, the result is failed payment. Any other type is considered successful.

To retry failed payments
POST: http://localhost:8080/api/policy/retry-failed

The above request will retry failed payments. Delinquent policies are excluded from retries