Sarvagya Sanjay

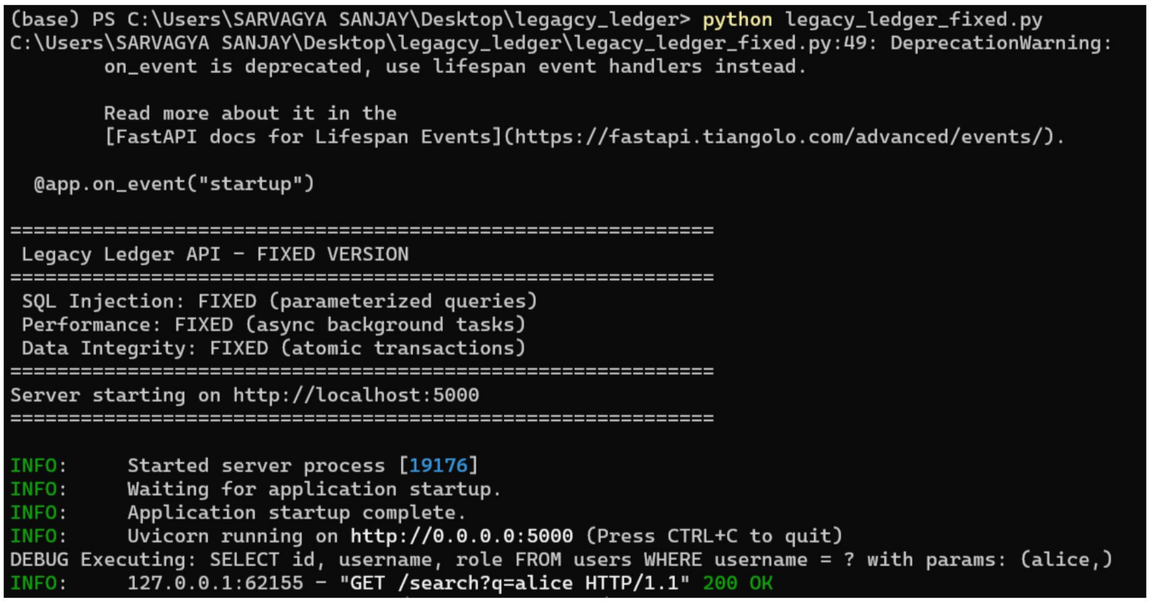
Testing Fixes

Step 1: Starting the Server

python legacy_ledger_fixed.py

Expected:

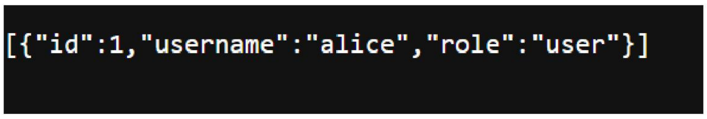
Legacy Ledger API - FIXED VERSION
Server starting on: http://localhost:5000



```
(base) PS C:\Users\SARVAGYA SANJAY\Desktop\legagcy_ledger> python legacy_ledger_fixed.py
C:\Users\SARVAGYA SANJAY\Desktop\legagcy_ledger\legacy_ledger_fixed.py:49: DeprecationWarning:
        on_event is deprecated, use lifespan event handlers instead.

        Read more about it in the
        [FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

  @app.on_event("startup")

============================================================
 Legacy Ledger API - FIXED VERSION
============================================================
 SQL Injection: FIXED (parameterized queries)
 Performance: FIXED (async background tasks)
 Data Integrity: FIXED (atomic transactions)
============================================================
Server starting on http://localhost:5000
============================================================

INFO:     Started server process [19176]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
DEBUG Executing: SELECT id, username, role FROM users WHERE username = ? with params: (alice,)
INFO:     127.0.0.1:62155 - "GET /search?q=alice HTTP/1.1" 200 OK
```

Step 2: Testing Normal Search (Browser)

Search for Alice: http://localhost:5000/search?q=alice

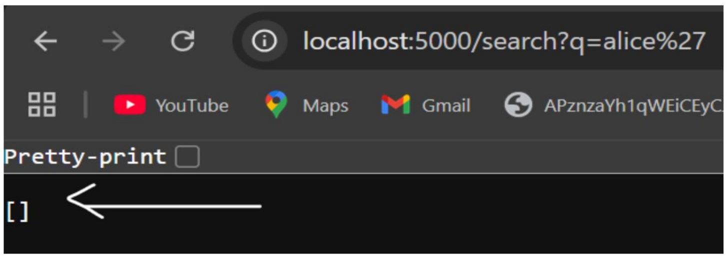


```
[{"id":1,"username":"alice","role":"user"}]
```

This shows that search works.

Step 3: Testing SQL Injection is Blocked (Browser)

In browser: *http://localhost:5000/search?q=alice'*



```
localhost:5000/search?q=alice%27
Pretty-print
[]
```

Expected: [ ] (empty - attack blocked)
Empty result. SQL injection is blocked. In the broken version, this would return ALL users.

Sarvagya Sanjay

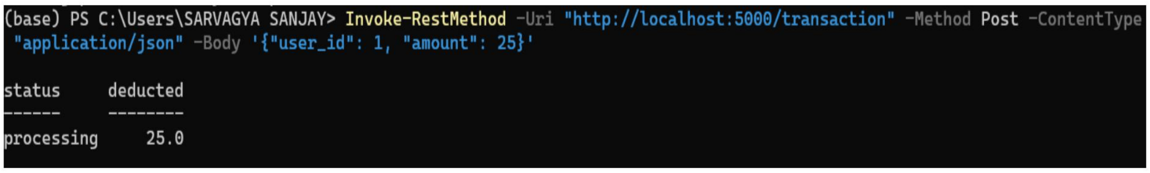
User Balance:

In browser: ***http://localhost:5000/users/1***

Output-> {"id":1,"username":"alice","balance":100.0,"role":"user"}

Step 4: Test Transactions (Using PowerShell)

Needed for POST requests.



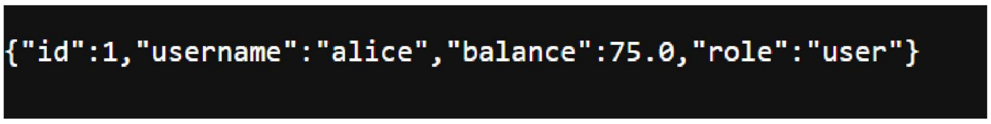
```
(base) PS C:\Users\SARVAGYA SANJAY> Invoke-RestMethod -Uri "http://localhost:5000/transaction" -Method Post -ContentType
 "application/json" -Body '{"user_id": 1, "amount": 25}'

status     deducted
------     --------
processing     25.0
```

It is observed that there is an immediate return, not post 3 seconds.

Checking if the transaction actually processed:

- Waiting for 5 seconds (the transaction processes in background)
- Browser: **http://localhost:5000/users/1**

Output:



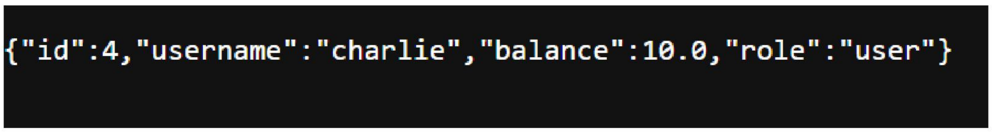
```
{"id":1,"username":"alice","balance":75.0,"role":"user"}
```

Balance went from $100 to $75. Hence, Transaction worked.

Step 5: Test Data Integrity (Insufficient Balance)

Withdrawing more than Charlie has:

1. Checking Charlie's balance first in browser: ***http://localhost:5000/users/4***
   Output: "balance":10.0
2. trying to deduct $20 (he only has $10):



```
{"id":4,"username":"charlie","balance":10.0,"role":"user"}
```

Balance unchanged. Transaction was rejected. The old code would have allowed negative balance.