SnapNStore

A Lightweight Key-Value Database with Snapshot & Restore

- Surabhi
- Kirti
- Bhavya
- Riya (Team Leader)
- Priyamvada
- Gayatri

Problem Statement

Key Value Database with CRUD Operations, Snapshot and Restore Support

Goals

- **Design a Lightweight Key-Value Store**
  → Create a simple, efficient in-memory key-value database to support rapid data access and manipulation
- **Implement core DB operations (CRUD)**
  → Efficiently handle create, read, update, and delete actions.
- **Build snapshot & restore system**
  → Allow saving and reverting to previous data states.
- **Maintain Data History**
  → Record historical changes to allow version tracking and understand key evolution over time.
- **Build Developer-Friendly Interfaces**
  → Provide both a Command-Line Interface (CLI) and a Web Dashboard for interacting with the database.
- **Connect with Modern Backend Architecture**
  → Use FastAPI for backend operations, ensuring scalability, speed, and integration with the frontend.

FLOW

JSON-based Storage (store.json, snapshots.json, history.json)

↓

CLI Interface using Python & Click

↓

Backend API using FastAPI

↓

Frontend Web Page (React + Vite + Lucid-react)

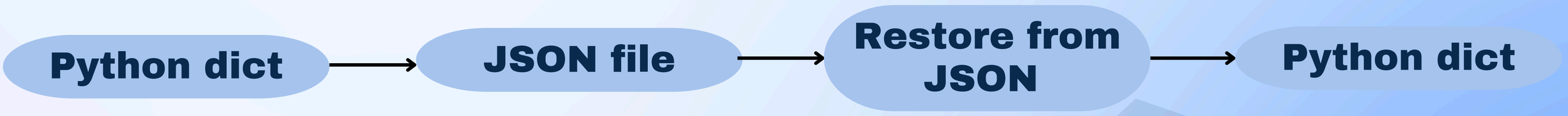
↓

Full Integration:
Web UI ↔ FastAPI ↔ JSON DB

Why JSON?

- **Natural Key-Value Format**
- **Schema-free & Dynamic**
- **Easy Snapshot & Restore**
- **Inspired by NoSQL (Redis, MongoDB)**
- **High Performance in Memory**

∴JSON helps us build a lightweight, flexible, and scalable key-value database that meets all our project goals.



ACTIONS

- Set Key
- Get Key
- View Data
- Delete Key
- Search Key

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py set SN01 Alice
Set SN01 = Alice
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py set SN02 Bob
Set SN02 = Bob
```

```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py get SN01
SN01 = Alice
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py view
Current Store:
{'SN01': 'Alice', 'SN02': 'Bob'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py delete SN01
Deleted SN01
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py exists SN01
SN01 exists: False
```

ACTIONS

Take_Snaps

View_Snaps

Restore

```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py view
Current Store:
{'SN02': 'Bob'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py takesnap snap1
Snapshot 'snap1' created.
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py viewsnap snap1
{'SN02': 'Bob'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py set SN03 Ram
Set SN03 = Ram
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py takesnap snap2
Snapshot 'snap2' created.
```

```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py restore snap1
Restored snapshot 'snap1'.
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py view
Current Store:
{'SN02': 'Bob'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py viewsnap snap2
{'SN02': 'Bob', 'SN03': 'Ram'}
```

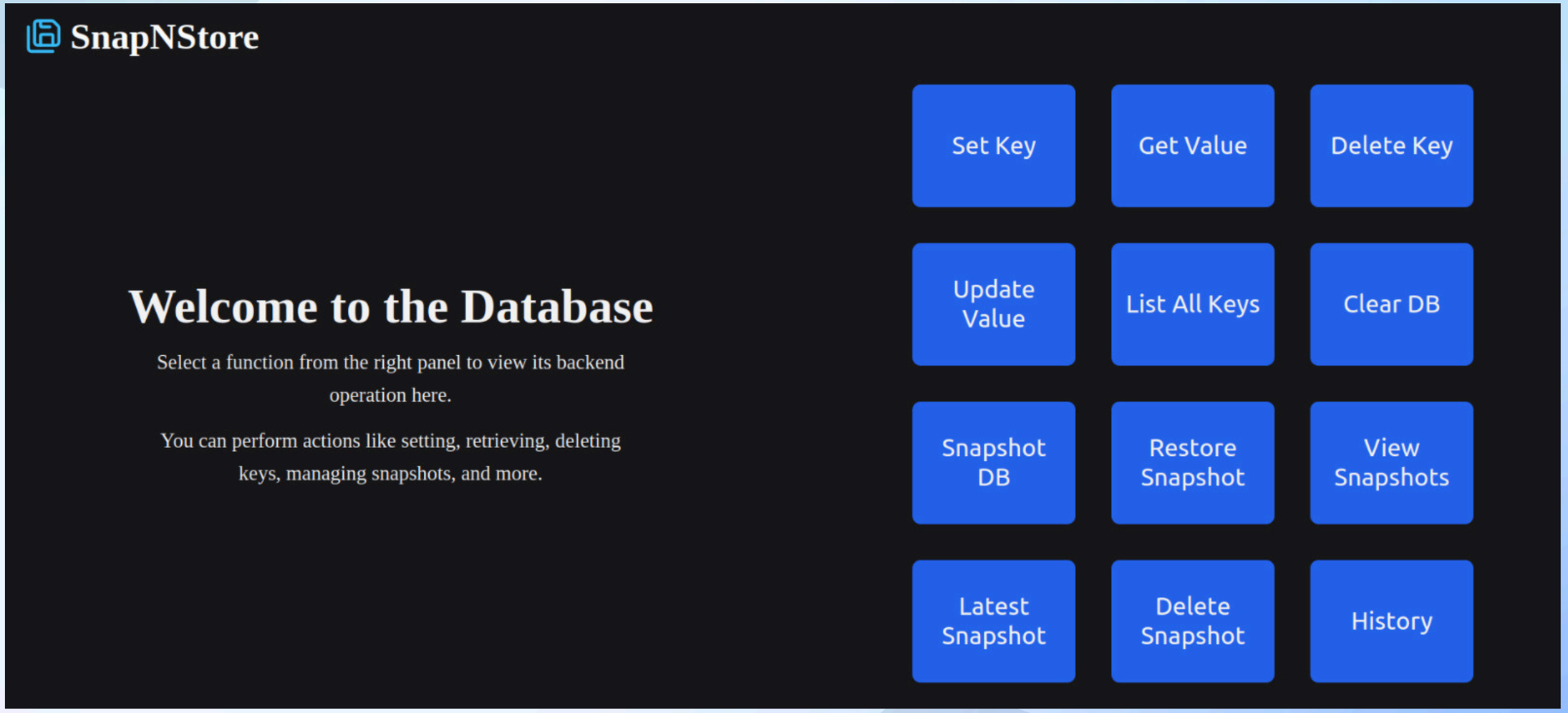
ACTIONS

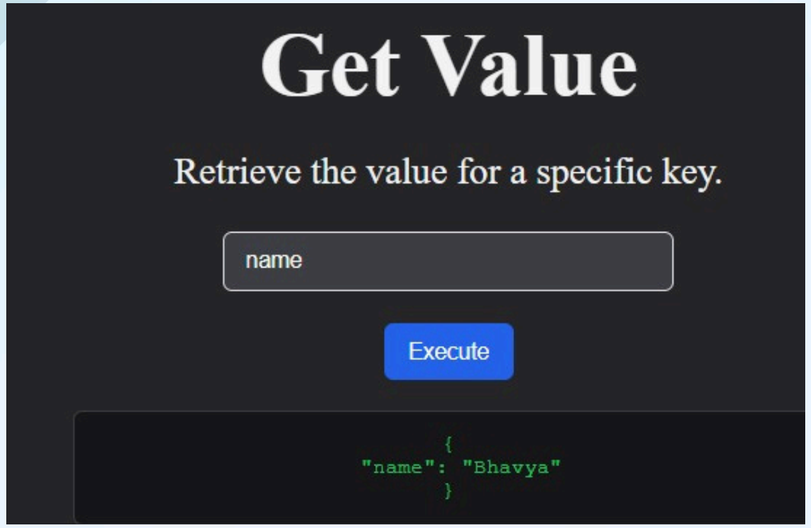
- All_Snap
- Latest_Snap
- Delete_Snaps
- History
- Clear

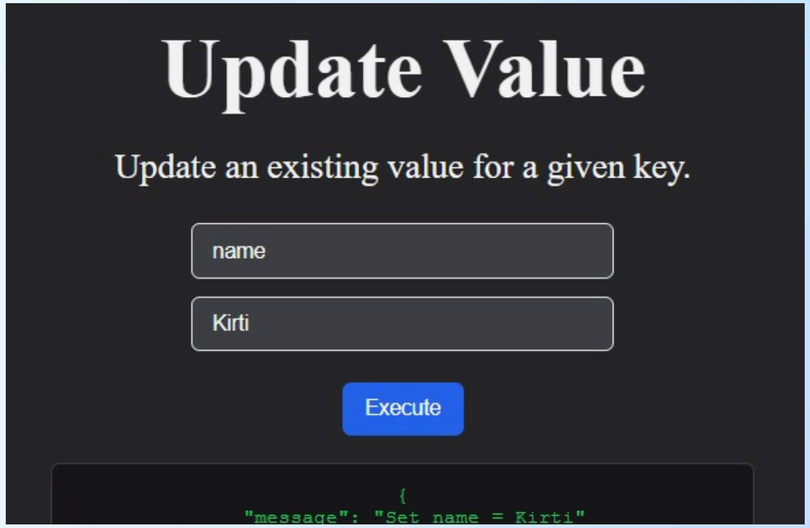
```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py allsnap
{'snap1': {'SN02': 'Bob'}, 'snap2': {'SN02': 'Bob', 'SN03': 'Ram'}}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py latestsnap
{'SN02': 'Bob', 'SN03': 'Ram'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py deletesnap snap2
Deleted snapshot 'snap2'.
```

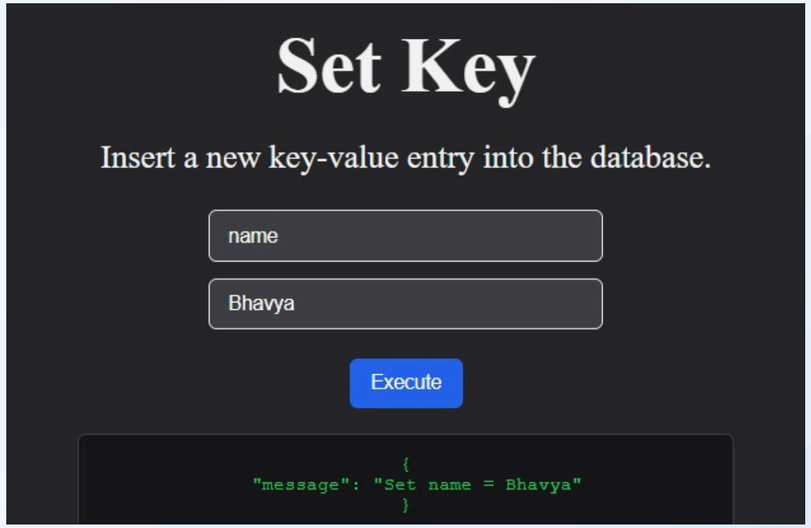
```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py history
{'2025-04-10 19:37:40.062904': 'SET SN01 = Alice', '2025-04-10 19:38:38.208641': 'SET SN01 = Alice', '2025-04-10 19:38:48.358345': 'SET SN02 = Bob', '2025-04-10 19:39:48.602165': 'GET SN01', '2025-04-10 19:41:57.068566': 'VIEW STORE', '2025-04-10 19:42:12.881007': 'DELETE SN01', '2025-04-10 19:42:25.499219': 'CHECK EXISTS SN01', '2025-04-10 19:42:49.133162': 'VIEW STORE', '2025-04-10 19:43:38.135146': 'SNAPSHOT snap1', '2025-04-10 19:44:26.935888': 'VIEW SNAPSHOT snap1', '2025-04-10 19:45:15.486535': 'SET SN03 = Ram', '2025-04-10 19:45:49.434358': 'SNAPSHOT snap2', '2025-04-10 19:46:27.006504': 'RESTORE SNAPSHOT snap1', '2025-04-10 19:47:26.943888': 'VIEW STORE', '2025-04-10 19:48:09.858522': 'VIEW SNAPSHOT snap2', '2025-04-10 19:49:19.799233': 'VIEW SNAPSHOT allsnap', '2025-04-10 19:49:48.546826': 'VIEW ALL SNAPSHOTS', '2025-04-10 19:50:44.906437': 'VIEW LATEST SNAPSHOT', '2025-04-10 19:51:08.535093': 'DELETE SNAPSHOT snap2'}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database>
```

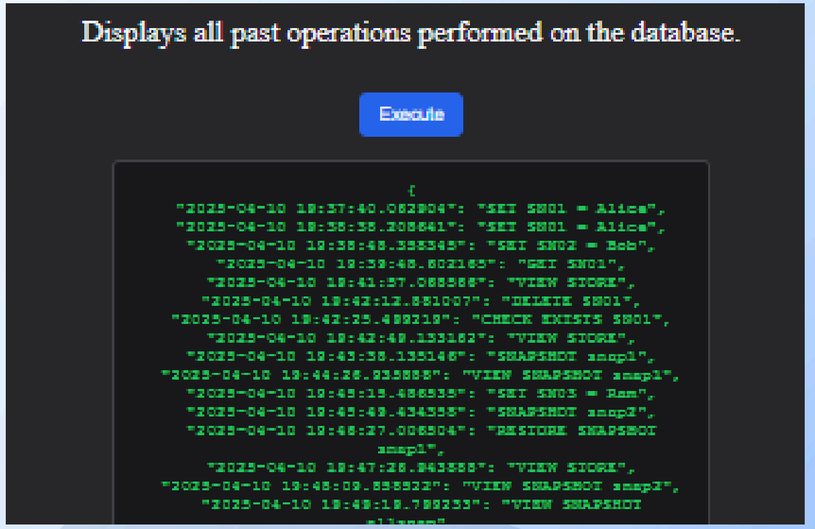
```
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py clear
Database cleared.
PS C:\Users\HP\Documents\GitHub\Key-Value-Database> python backend/cli/main.py view
Current Store:
{}
PS C:\Users\HP\Documents\GitHub\Key-Value-Database>
```


SnapNStore
Welcome to the Database
Select a function from the right panel to view its backend operation here.
You can perform actions like setting, retrieving, deleting keys, managing snapshots, and more.
Set Key
Get Value
Delete Key
Update Value
List All Keys
Clear DB
Snapshot DB
Restore Snapshot
View Snapshots
Latest Snapshot
Delete Snapshot
History


Get Value
Retrieve the value for a specific key.
name
Execute
{
"name": "Bhavya"
}


Update Value
Update an existing value for a given key.
name
Kirti
Execute
{
"message": "Set name = Kirti"


Set Key
Insert a new key-value entry into the database.
name
Bhavya
Execute
{
"message": "Set name = Bhavya"
}


Displays all past operations performed on the database.
Execute
{
"2025-04-10 18:37:40.062804": "SET SN01 = Alice",
"2025-04-10 18:38:38.208641": "SET SN01 = Alice",
"2025-04-10 18:38:48.333345": "SET SN02 = Bob",
"2025-04-10 18:39:48.602165": "SET SN01",
"2025-04-10 18:41:57.088568": "VIEW STORE",
"2025-04-10 18:42:12.881007": "DELETE SN01",
"2025-04-10 18:42:25.499218": "CHECK EXISTS SN01",
"2025-04-10 18:43:48.133162": "VIEW STORE",
"2025-04-10 18:43:58.135148": "SNAPSHOT snap1",
"2025-04-10 18:44:26.833888": "VIEW SNAPSHOT snap1",
"2025-04-10 18:45:18.486535": "SET SN03 = Ram",
"2025-04-10 18:45:49.434355": "SNAPSHOT snap2",
"2025-04-10 18:46:27.006504": "RESTORE SNAPSHOT
snap1",
"2025-04-10 18:47:26.843555": "VIEW STORE",
"2025-04-10 18:48:09.855522": "VIEW SNAPSHOT snap2",
"2025-04-10 18:48:18.799253": "VIEW SNAPSHOT

View Snapshots

Lists all saved database snapshots.

Execute


```
{
  "1": {
"name": "Kirti",
  "age": "20"
  },
  "2": {
```


Snapshot DB

Creates a snapshot of the current database state.

1

Execute


```
{
"message": "Snapshot '1' taken."
}
```


Latest Snapshot

Displays the most recent snapshot of the database.

Execute


```
{
"name": "Kirti",
  "age": "20",
"branch": "cse"
}
```


List All Keys

Displays all keys currently stored in the database.

Execute


```
{
"name": "Kirti",
  "age": "20"
}
```

Clear DB

Removes all key-value pairs from the database.

Execute


```
{
"message": "Database cleared successfully."
}
```


Restore Snapshot

Restores the database from a previous snapshot.

1

Execute


```
{
"message": "Snapshot '1' restored."
}
```


List All Keys

Displays all keys currently stored in the database.

Execute


```
{
"name": "Kirti",
"age": "20"
}
```


Delete Snapshot

Delete a snapshot by its name.

1

Execute


```
{
"message": "Deleted snapshot '1'"
}
```

Backend Logic

Modules & Frameworks

- Json, Os, Datetime
- Sys, Click
- FastAPI (Backend Web Server)

Files Used

- snapshots.json (snapshots)
- store.json (data file)
- history.json (actions log)

Frontend

Tools & Frameworks

- Vite (Build Tool)
- React (Frontend Library)
- Lucid-react (Icons)
- JavaScript (Language)

Real-Life Applications

- **User Sessions Management (Authentication Systems):**

  Minimal version of how web servers store and validate session tokens using a key-value store. Our system can simulate login/logout and restore past session states using snapshots.

- **IoT Device Configurations (Smart Homes, Vehicles):**

  Our system can simulate a simplified backend for smart homes or connected cars, storing and restoring the latest configurations of each device.

- **Minimal Student Attendance & ID Mapping System:**

  We mimicked a lightweight attendance system that can capture multiple attendance snapshots, useful for recovery or audits.

Future Enhancement : Cloud Storage Integration

🔍 Why This Matters:

- **Scalability:**
  Easily handle large numbers of snapshots beyond local limits.
- **Remote Restore:**
  Enable recovery from any machine or location.
- **Data Safety:**
  Use cloud-native encryption and redundancy for added protection.
- **Seamless Backup:**
  Automate and manage versions for audit and rollback.

∴ Moving snapshot storage to the cloud will prepare our system for multi-user, distributed access and real-world reliability.

Conclusions

- Built a lightweight, modular Key-Value DB with CLI + Web UI + REST APIs
- Strict unique key-value mapping — ideal for sessions, tokens, configs
- Snapshot system — save, restore, and track DB states seamlessly

- FastAPI + Uvicorn-powered backend — blazing fast, async-ready APIs
- User-friendly frontend — intuitive interface for non-technical users
- Clear separation of concerns — scalable CLI, API, DB layers
- Conclusion: A creative, user-focused real-time KV system bridging simplicity and power — ready for real-world use

Thank You!