Infosys Springboard Virtual Internship 6.0 Completion Report

Batch Number: 11 PyKV – A Scalable In-Memory Key-Value Store with Persistence

Start date : 29-12-2025

Names:

1.	DIVYAPPRIYA S
2.	MANASHI DEBNATH
3.	HARI PRIYAN V
4.	DEVANSH TANTWAY
5.	SHAIK MAHABOOB BASHA

Internship Duration: 8 Weeks

1.Project Title

PyKV – A Scalable In-Memory Key-Value Store with Persistence

2. Project Objective

PyKV is a scalable in-memory key-value store developed to demonstrate high-performance backend system design combined with reliable file-based persistence. The objective of the project is to design and implement a system that enables fast data access using in-memory storage while ensuring durability through automatic JSON-based persistence. The project exposes RESTful APIs using FastAPI and integrates a Streamlit-based frontend dashboard to provide an interactive interface for managing key-value operations. The system supports core CRUD operations including insertion, retrieval, deletion, and full data visualization.The primary goal of PyKV is to simulate a lightweight enterprise storage engine that reflects real-world backend architectural practices, modular design, persistence mechanisms, and frontend-backend integration.

3. Project description in detail

PyKV is a backend-focused scalable storage engine that combines the performance of in-memory data storage with the reliability of persistent file-based storage. The system uses a Python dictionary as the core in-memory storage engine, enabling constant-time data access and modification.

To ensure durability, the project implements a persistence layer that automatically saves all key-value pairs to a JSON file (data/store.json). On application restart, the stored data is reloaded into memory, ensuring no data loss.

The backend is implemented using FastAPI, which exposes REST endpoints for performing operations such as:

- PUT – Insert or update key-value pairs
- GET – Retrieve value using a key
- DELETE – Remove key-value pair
- ALL – Retrieve all stored entries

The frontend dashboard is developed using Streamlit, allowing users to interact with the backend system through a clean and responsive interface. The dashboard communicates with the backend via HTTP requests and dynamically updates the data display.

The system architecture is modular and divided into:

- Core Engine Layer
- Persistence Layer
- API Layer
- Frontend Dashboard Layer

The project demonstrates practical implementation of backend system architecture, persistence management, REST API integration, and interactive dashboard development.

4. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Requirement analysis and project planning	Requirements finalized and system architecture defined

Week 2	Core engine development	Implemented in-memory key-value storage logic
Week 3	Persistence layer implementation	Integrated JSON-based file persistence mechanism
Week 4	REST API development using FastAPI	Developed and tested CRUD (PUT, GET, DELETE) endpoints
Week 5	Frontend dashboard development	Designed and implemented Streamlit-based dashboard interface
Week 6	System integration	Successfully connected frontend dashboard with backend REST APIs
Week 7	Testing and debugging	Identified and resolved synchronization issues, API validation errors, and integration bugs
Week 8	Documentation and final review	Prepared internship completion report and finalized project presentation

5a. Key Milestones

Milestone	**Description**	**Date Achieved**
Project Kickoff	Defined objectives, architecture, and team roles	30.12.2025
Core Engine Completion	In-memory store implemented successfully	05.01.2026
API Integration	FastAPI endpoints fully operational	07.01.2026
Dashboard Integration	Streamlit dashboard connected to backend	10.02.2026
Final Submission	Complete system tested and documented	13.02.2026

5b. Project execution details

• Requirement Analysis – Identified system requirements, performance expectations, and architecture design.

• System Design – Designed modular architecture dividing engine, persistence, API, and frontend layers.

• Core Engine Development – Implemented Python dictionary-based storage engine for fast data operations.

• Persistence Implementation – Developed JSON-based automatic saving and loading mechanism.

• API Development – Built RESTful endpoints using FastAPI framework.

• Frontend Development – Developed interactive dashboard using Streamlit for real-time system interaction.

• Integration – Linked backend API with Streamlit frontend using HTTP requests.

• Testing – Conducted CRUD operation testing, error validation, and consistency verification.

• Debugging – Resolved API method mismatches, request validation errors, and persistence synchronization issues.

• Documentation – Prepared detailed internship completion report and project explanation materials.

6.Snapshots / Screenshots

Step 1: Cloning the Repository to Local System

The project source code was maintained in a centralized GitHub repository. To begin development and execution, the repository was cloned to the local development environment.

Command Used:

git clone https://github.com/TEAM-C-Infosys-Springboard/Project1

Navigate into the project directory:

cd Project1

Outcome:

- Complete project source code downloaded locally
- Folder structure including api/, engine/, data/, and frontend/ directories initialized

Step 2: Virtual Environment Setup

To ensure dependency isolation and avoid conflicts, a Python virtual environment was created.

- **Create Virtual Environment:**

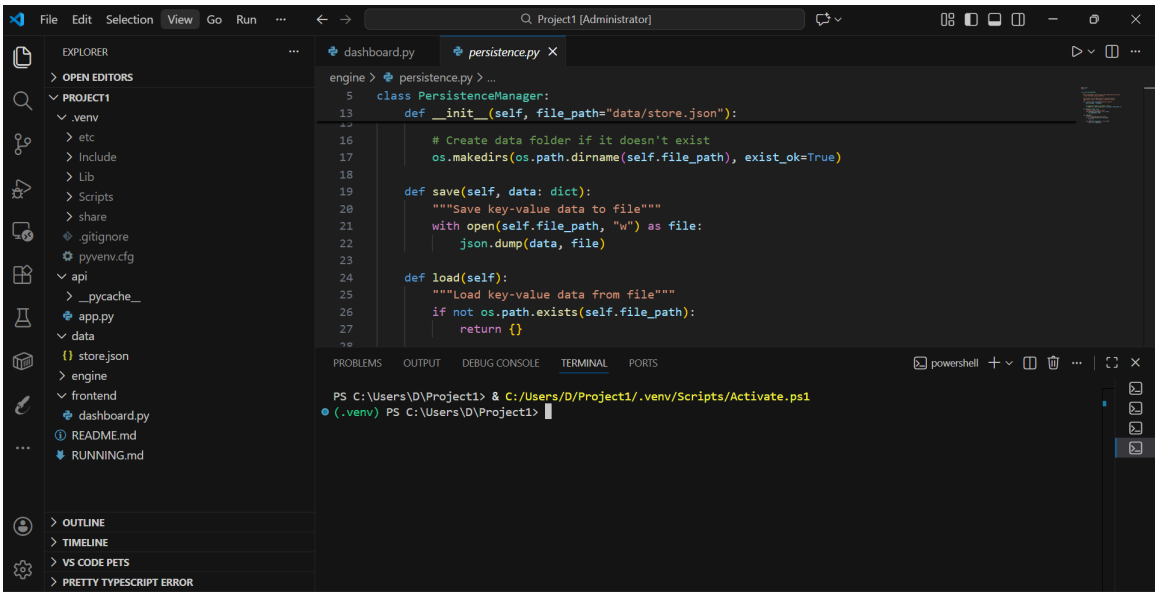
python -m venv venv

- **Activate Virtual Environment (Windows):**

venv\Scripts\activate

Outcome:

- Dedicated Python environment created
- Project dependencies isolated



Step 3: Installing Required Dependencies

All required libraries for backend and frontend execution were installed.

Command Used:

pip install fastapi uvicorn streamlit requests pandas altair

Libraries Installed:

- FastAPI (Backend API framework)
- Uvicorn (ASGI server)
- Streamlit (Frontend dashboard)
- Requests (HTTP communication)
- Pandas (Data manipulation)
- Altair (Data visualization)

Outcome:

- All dependencies successfully installed
- Environment ready for execution

Step 4: Backend API Execution

The backend server was launched using Uvicorn to expose REST endpoints.

Command Used:

python -m uvicorn api.app:app --reload

Backend Runs On:

http://127.0.0.1:8000

Verification:

- Interactive API documentation accessible at:

http://127.0.0.1:8000/docs

Outcome:

- API endpoints initialized successfully
- Live reload enabled for development
- HTTP 200 responses confirmed in terminal logs

Step 5: Streamlit Frontend Dashboard Execution

The frontend dashboard was developed using Streamlit and launched separately.

Command Used:

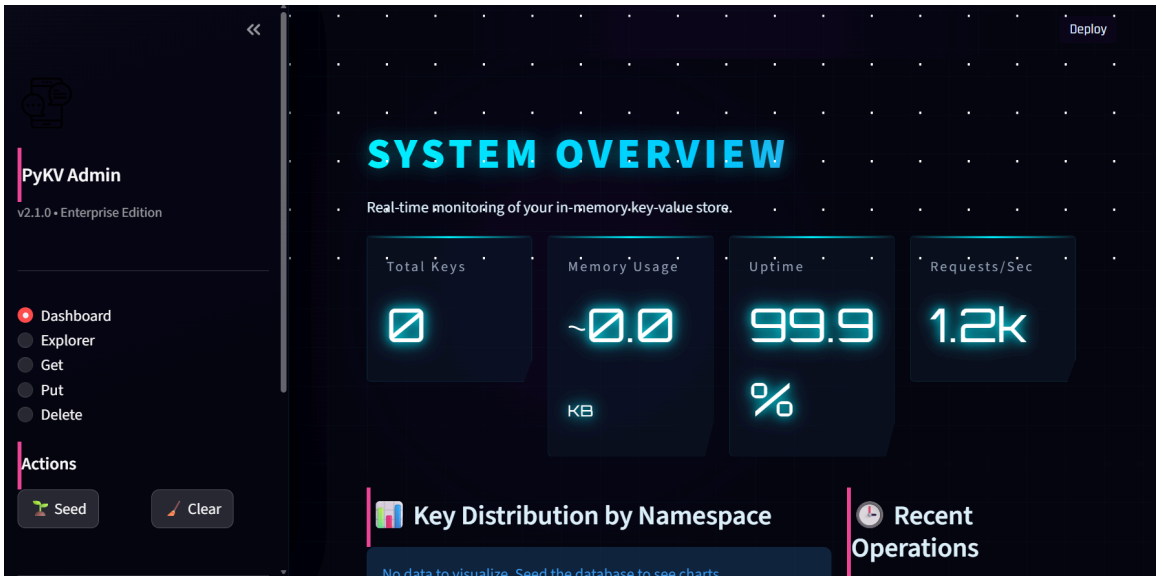
python -m streamlit run frontend/dashboard.py

Frontend Runs On:

http://localhost:8501

Outcome:

- Enterprise-style Admin Dashboard loaded
- Navigation sidebar enabled
- Backend API successfully integrated



Step 6: System Overview Verification

The dashboard displays real-time system metrics including:

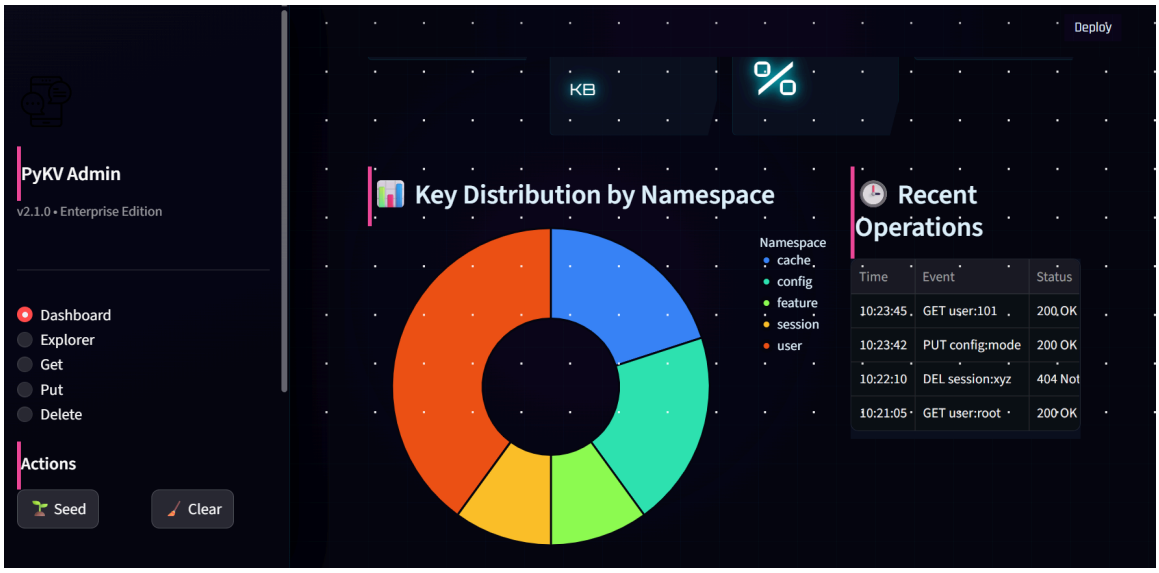
- Total Keys

- Memory Usage
- Uptime
- Requests Per Second

These values are dynamically retrieved from backend endpoints.

Outcome:

- Real-time monitoring validated
- Dashboard UI confirmed functional



Step 7: PUT Operation Testing (Store Value)

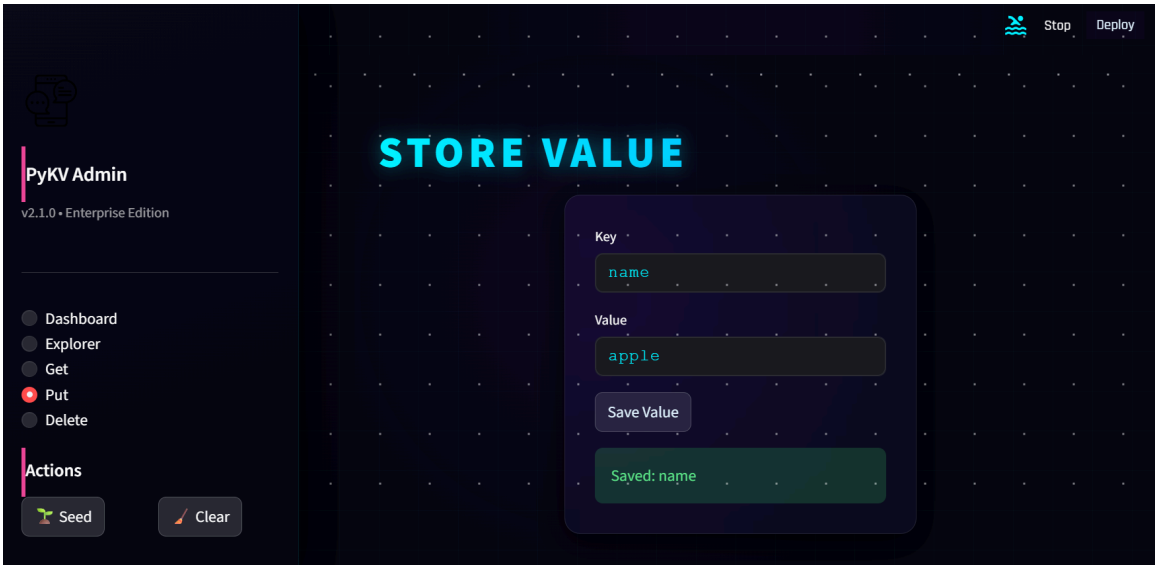
A key-value pair was inserted using the frontend interface.

Workflow:

1. Enter Key and Value
2. Click "Save Value"
3. Streamlit sends POST request to /put
4. Backend stores data in memory
5. Persistence layer updates store.json

Outcome:

- Data successfully stored
- Confirmation message displayed
- JSON file updated



Step 8: GET Operation Testing (Fetch Value)

Stored values were retrieved using the GET module.

Workflow:

1. Enter Key
2. Click "Fetch Value"
3. Streamlit sends GET request /get/{key}
4. Backend returns stored value

Error Handling:

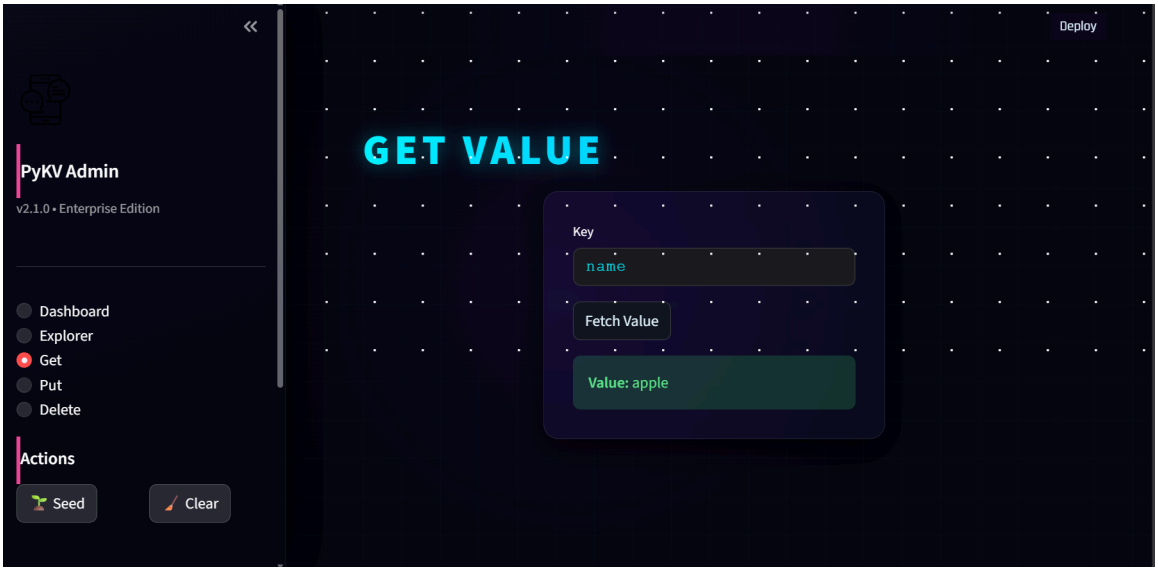
If key does not exist:

Key not found

Outcome:

- Value retrieved successfully
- Proper validation for missing keys

-> fetch: successful



Step 9: DELETE Operation Testing

The DELETE module was used to remove stored keys.

Workflow:

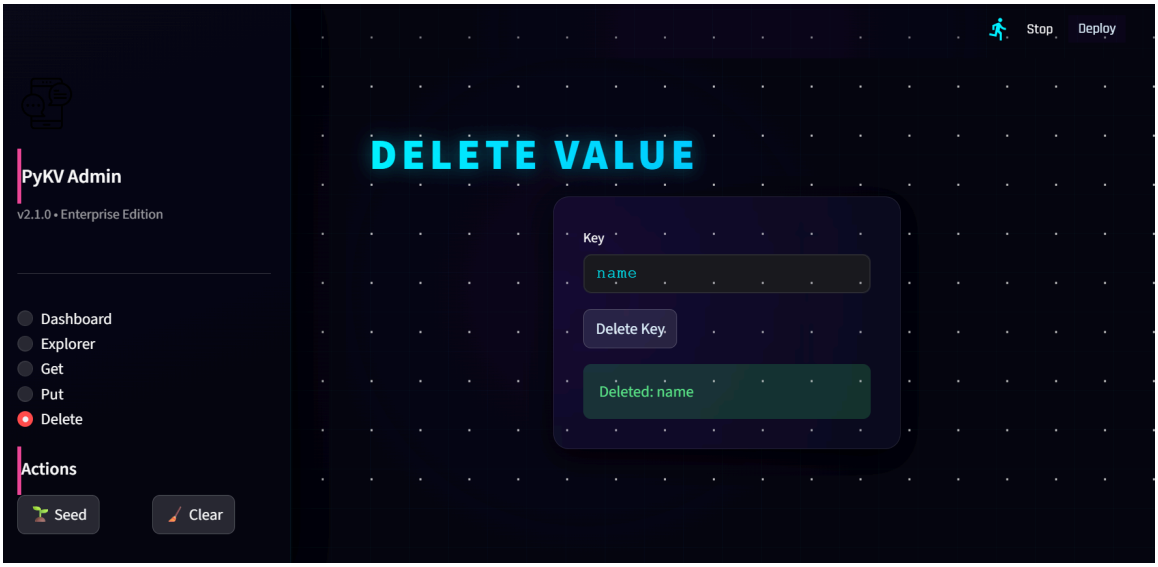
1. Enter Key
2. Click "Delete Key"
3. Streamlit sends DELETE request /delete/{key}
4. Backend removes key
5. JSON file updated

Validation:

- Subsequent GET confirms deletion

Outcome:

- Key removed successfully
- Persistence validated



Step 10: Data Explorer & Visualization Validation

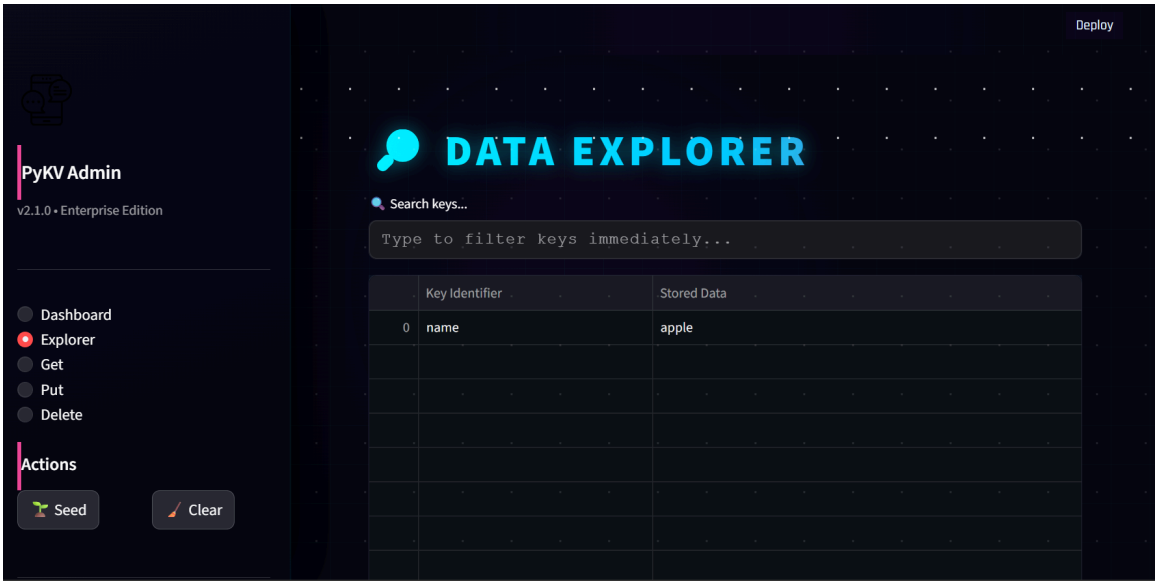
The Data Explorer module was tested.

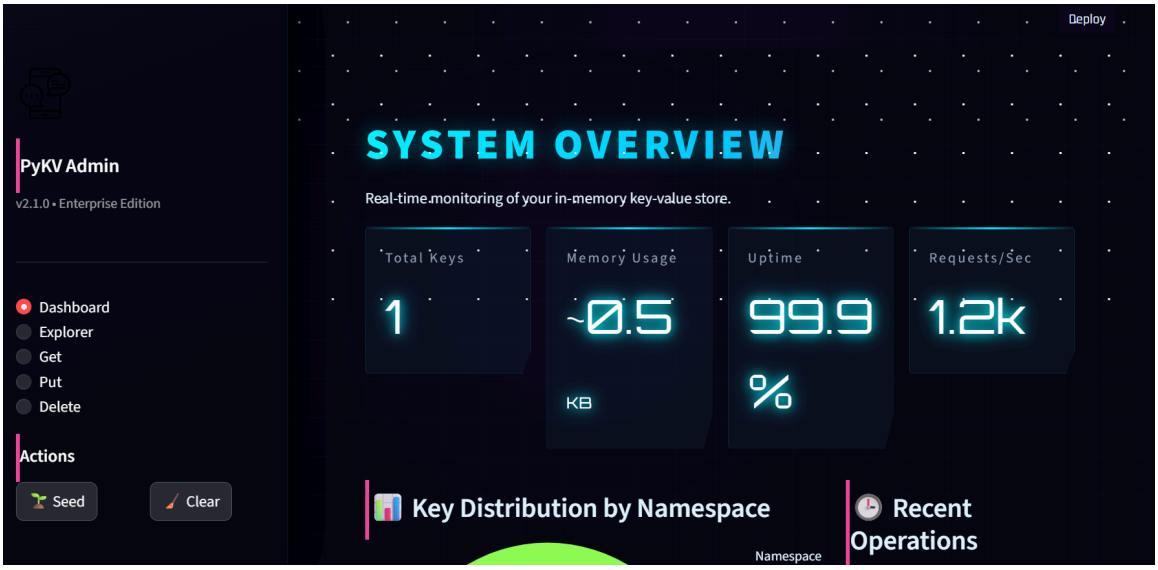
Features Validated:

- Key search functionality
- Namespace-based categorization
- Real-time donut chart visualization
- Recent operations monitoring

Outcome:

- Charts dynamically updated
- API integration verified
- Full-stack functionality confirmed





7.Challenges Faced

- Managing Git merge conflicts during collaborative development
- Handling request validation errors in FastAPI
- Synchronizing frontend and backend logic
- Debugging HTTP method mismatches
- Ensuring data consistency between memory and persistent storage
- Managing state updates after deletion operations
- Handling error responses for missing keys
- Time management across backend, frontend, and documentation phases

8. Learnings & Skills Acquired

• Backend System Architecture Design

Through the development of PyKV, a strong understanding of backend architectural principles was developed. The system was structured using a layered modular approach separating the core storage engine, persistence mechanism, API layer, and frontend integration. This separation of concerns improved maintainability, scalability, and clarity of implementation. The experience strengthened knowledge of designing backend systems that are extensible and aligned with enterprise development standards.

• REST API Development using FastAPI

The project involved designing and implementing RESTful APIs using the FastAPI framework. Endpoints were created to handle PUT, GET, DELETE, and data retrieval operations. The implementation included request validation, structured response handling, and HTTP status code management. The use of FastAPI enabled rapid API development while maintaining high performance and clean documentation.

• File-Based Persistence Mechanisms

A persistence layer was implemented to ensure that in-memory data remains durable across application restarts. The system uses JSON-based file storage to automatically save and reload data. This provided hands-on experience in implementing data durability mechanisms and understanding the importance of maintaining consistency between volatile memory and persistent storage.

• In-Memory Data Management

The core storage engine was built using Python's dictionary data structure to enable constant-time (O(1)) data access. This helped in understanding how in-memory systems operate and why they offer high-speed performance compared to disk-based systems. The implementation strengthened understanding of efficient data storage techniques and runtime complexity considerations.

• Frontend Development using Streamlit

The frontend dashboard was developed using Streamlit, allowing real-time interaction with backend APIs. The experience included designing a responsive UI, handling user input, and dynamically updating displayed data. Streamlit enabled rapid dashboard development while reinforcing concepts of interactive data-driven interfaces.

• API Integration and HTTP Communication

The project involved integrating the Streamlit frontend with the FastAPI backend using HTTP requests. This provided practical exposure to client-server communication, REST-based interactions, and handling JSON responses. It enhanced understanding of how frontend applications consume backend services in real-world systems.

• JSON Data Handling

JSON was used as the primary data format for storing persistent data and exchanging API responses. The project strengthened understanding of serialization, deserialization, and structured data formatting. It also improved knowledge of managing structured data within web-based applications.

• Error Handling and Validation

The system incorporated validation checks for missing keys, invalid requests, incorrect HTTP methods, and empty inputs. FastAPI's validation mechanisms were utilized to enforce structured request handling. Debugging and resolving these issues enhanced the ability to design robust and fault-tolerant systems.

• Git and GitHub Version Control

The project was managed using Git for version control and GitHub for repository hosting. Activities included branching, committing, resolving merge conflicts, handling push/pull errors, and collaborative development practices. This improved understanding of professional code management workflows.

• Debugging and System Testing

Extensive testing was performed on CRUD operations, persistence consistency, and API integration. Several runtime and logical errors were identified and resolved, including method mismatches, state inconsistencies, and persistence synchronization issues. This enhanced analytical problem-solving and debugging skills.

• Modular Software Design

The project was structured into independent modules such as engine, persistence, API, and frontend. This modular approach improved readability, maintainability, and scalability of the codebase. The experience reinforced the importance of designing loosely coupled and well-organized systems.

• Full-Stack Integration

Although primarily backend-focused, the project achieved complete integration between backend logic and frontend interface. The implementation demonstrated how storage engines, REST APIs, and interactive dashboards work together in a unified system. This provided exposure to full-stack system design principles.

9. Testimonials from team

Team Member 1: Divyappriya S
"This project enhanced my understanding of backend architecture and storage engine implementation."

Team Member 2: Manashi Debnath
"I gained practical experience in implementing persistence mechanisms and ensuring system reliability."

Team Member 3: Hari Priyan V
"The integration of REST APIs with a frontend dashboard improved my understanding of full-stack development."

Team Member 4: Devansh Tantway
"Working on PyKV strengthened my debugging skills and knowledge of modular system design."

Team Member 5: Shaik Mahaboob Basha
"This project helped me understand real-world backend engineering concepts and scalable design practices."

10. Conclusion

The PyKV project successfully demonstrates the implementation of a scalable in-memory key-value store integrated with file-based persistence and a REST API interface. By combining high-speed memory storage with durable JSON persistence, the system achieves both performance and reliability.

The integration of FastAPI and Streamlit showcases full-stack system design capabilities. The project reflects enterprise-style modular architecture and provides a strong foundation for future enhancements such as distributed storage, replication, and eviction policies.

Overall, PyKV effectively meets its objectives and represents a practical application of backend engineering principles

11. Acknowledgements

I would like to express my sincere gratitude to **Infosys Springboard** for providing me the opportunity to work on the *PyKV – Scalable In-Memory Key-Value Store with Persistence* project and gain valuable hands-on experience in backend system design and full-stack integration.

I am especially thankful to my mentor for their continuous guidance, technical support, and constructive feedback throughout the internship period. Their mentorship played a significant role in enhancing my understanding of system architecture, REST API development, and persistence mechanisms. The insights and direction provided greatly contributed to the successful completion of this project.

I would also like to extend my appreciation to my team members for their cooperation, collaboration, and collective efforts in executing the project. Their support and coordination helped ensure smooth progress and effective implementation of all modules.

Finally, I sincerely thank everyone who contributed directly or indirectly to my learning journey and the successful completion of this internship project. The experience has significantly strengthened my technical, analytical, and problem-solving skills.