Infosys Springboard Virtual Internship 6.0 Completion Report

# Team Details: Team – B

Venkatesh K

Sai Teja

Srija Budata

Harini S

Siddhesh Pitale

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

**Start date:** 29-12-2025

**Names:**

|  |  |
| --- | --- |
| 1 | Venkatesh K |
| 2 | Sai Teja |
| 3 | Srija Budata |
| 4 | Harini S |
| 5 | Siddhesh Pitale |

**Internship Duration**: 8 Weeks

# 1. Project Title

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

# 2. Project Objective

A Scalable In-Memory Key-Value Store with Persistence is designed to provide fast and efficient data storage and retrieval using Python. The system stores data in memory to ensure high performance while maintaining persistence through JSON files so that information remains safe even after restarting the application. It supports basic operations such as creating, reading, updating, and deleting key-value pairs through RESTful APIs, enabling smooth communication between the frontend and backend. The application follows a modular structure, making it easy to maintain, extend, and scale in the future, while also strengthening practical knowledge in backend development and data management. It ensures organized data handling through a clear separation of core logic and storage mechanisms. The system is structured to support future improvements such as concurrency and advanced data operations. Proper validation techniques are applied to maintain accuracy and consistency of stored information. The design approach enhances understanding of real-time data processing and efficient resource utilization. It also encourages clean coding practices and structured application development. Overall, the system represents a practical implementation of a reliable and scalable data management solution.

# 3. Project description in detail

A Scalable In-Memory Key-Value Store with Persistence is a software application developed using Python to manage data efficiently through key-value pairs. The system stores data directly in memory, which allows faster access and quick processing compared to traditional disk-based storage systems. Each piece of data is associated with a unique key, enabling simple and structured storage and retrieval. This approach makes the system lightweight, efficient, and suitable for applications that require quick data access.

The architecture of the system is designed with a clear separation between storage logic, API handling, and user interface components. The backend manages the in-memory data structure and handles operations such as insertion, retrieval, modification, and deletion of data. RESTful APIs are implemented to enable smooth communication between the frontend and backend. This structured design improves maintainability, readability, and scalability of the application.

To ensure data safety, the system implements persistence using JSON files. Even if the application is stopped or restarted, the stored data can be reloaded from the JSON file without loss. This feature combines the speed of in - memory storage with the reliability of disk-based backup. Proper file handling techniques are used to maintain data consistency and prevent corruption during read and write operations.

The system also includes validation and error-handling mechanisms to ensure stable performance. It checks for duplicate keys, handles invalid inputs, and provides appropriate responses for different types of requests. These measures improve reliability and user experience. The modular structure further allows easy integration of additional features such as concurrency support, authentication, or advanced querying in the future.

Overall, this application demonstrates practical implementation of scalable system design and efficient data management principles. It enhances understanding of backend development, API integration, file handling, and structured programming concepts. The project reflects real-world software development practices and provides a strong foundation for building more advanced distributed or database-driven systems in the future.

**4. Timeline Overview**

|  |  |  |
| --- | --- | --- |
| **Week** | **Activities Planned** | **Activities Completed** |
| Week 1 | Requirement gathering, project scope definition, and architecture planning | Requirements finalized, scope documented, modular architecture designed |
| Week 2 | Research on key-value storage mechanisms, TTL handling, and persistence techniques | In-memory storage approach selected, JSON persistence and WAL strategy finalized |
| Week 3 | Design of data storage structure and file organization | JSON structure created, WAL log format defined, folder structure organized |
| Week 4 | Development of core store module (CRUD operations) | Create, Read, Update, Delete operations implemented and tested |
| Week 5 | |  | | --- | |  |   Implementation of persistence and recovery mechanisms   |  | | --- | |  | | Data successfully written to JSON file, WAL logging and crash recovery implemented |
| Week 6 | Frontend development (dashboard, forms, UI design) | Responsive interface developed using HTML, CSS, and JavaScript |
| Week 7 | System integration (frontend, backend, persistence modules) | Complete integration achieved, real-time key-value operations enabled |
| Week 8 | |  | | --- | |  | | Testing, debugging, documentation, and final project deployment | | | Functional testing completed, error fixed, project deployed successfully |

# 5a. Key Milestones

|  |  |  |
| --- | --- | --- |
| Milestone | Description | Date Achieved |
| Project Kickoff | Project scope, objectives, and core features (CRUD operations, TTL, persistence) were finalized. Roles and responsibilities were assigned to team members. | 29 – Dec - 2025 |
| Prototype/First Draft | Basic key-value storage system was developed with PUT, GET, and DELETE operations. Initial UI was designed. | 09 – Jan -2026 |
| Mid-Term Review | Implemented TTL functionality and automatic key expiration logic. Reviewed system performance and refined backend structure. | 23- Jan - 2026 |
| Final Submission | Completed data persistence feature and integrated frontend with backend. All operations were tested successfully. | 6 - Feb - 2026 |
| Presentation | Prepared and delivered the project presentation showcasing features and outcomes. | 16- Feb - 2026 |

# 5b. Project execution details

**1. Requirements Analysis -** Functional and non-functional requirements of the key-value store were clearly identified. Features such as CRUD operations, data persistence, TTL support, and recovery were defined.

**2. Technology Selection -** Python and Flask were chosen for backend development due to simplicity and efficiency.HTML, CSS, JavaScript, and JSON were selected for frontend design and data storage.

**3. System Design -** A modular architecture was designed to separate frontend, backend, and persistence layers. Clear interaction flow was defined between user interface, server logic, and data storage.

**4. Database Structure Design** - In-memory storage was implemented for fast data access. Data persistence was maintained using JSON files and Write-Ahead Logging (WAL).

**5. Module Development -** Individual modules such as store management, persistence, and recovery were developed. Each module was assigned specific responsibilities to maintain structured implementation.

**6. API Design and Implementation -** RESTful APIs were designed for communication between frontend and backend. Endpoints were created for create, read, update, and delete operations.

**7. Frontend Development -** A responsive and user-friendly interface was designed using HTML, CSS, and JavaScript. Interactive forms and dashboard pages enable smooth user interaction.

**8. Backend Development -** Flask routes were implemented to handle client requests. Business logic manages key-value operations and data processing.

**9. Testing and Integration -** Individual modules were tested to verify correct functionality. Complete integration testing ensured smooth interaction among all components.

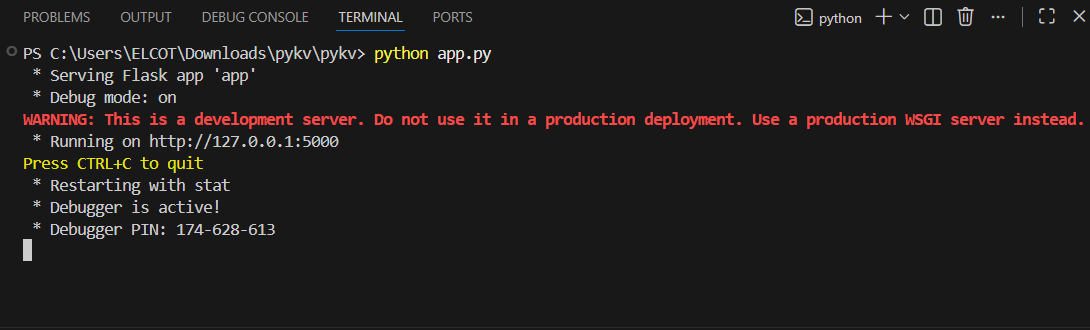
**10. Deployment and Documentation -** Application was deployed locally through the Flask server. Technical documentation was prepared describing setup, execution, and working process.

# 6. Snapshots / Screenshots

**Step 1**: Go to GitHub and clone the Repository in your system then open in VS Code to do the execution, here is the GitHub repo link:

<https://github.com/Siddheshpitale/Team-B-.git>

**Step 2 :** Start the with python app.py



**Step 3:** Here is the Landing page

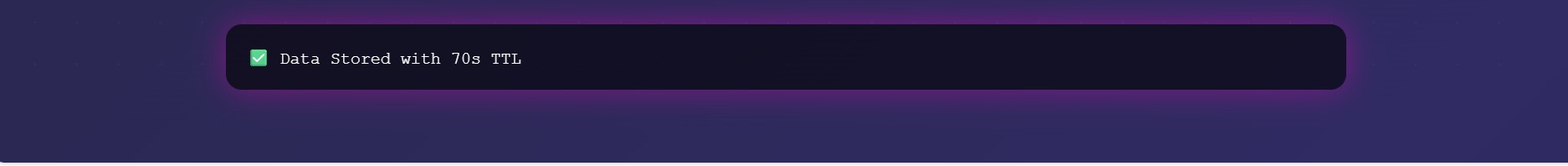


**Step 4:** After clicking on Enter Project it shows all the operations

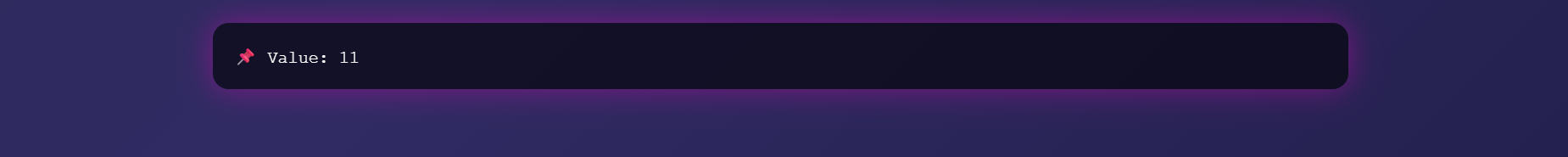
**Step 4 :** After clicking Enter project it shows all the operations



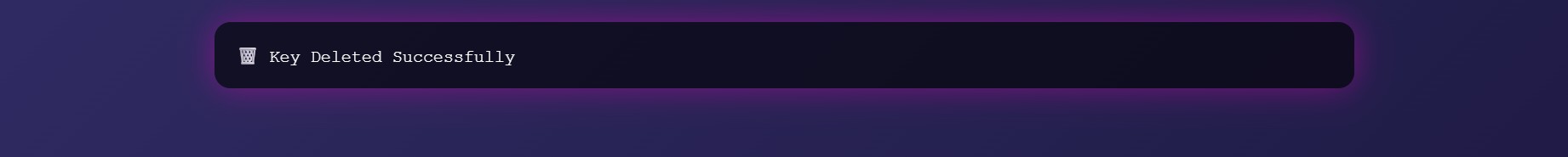
**Step 5:**  The user enter key and its value and click **PUT** operation, it saves the entered key and value in the database with an optional TTL for automatic expiry



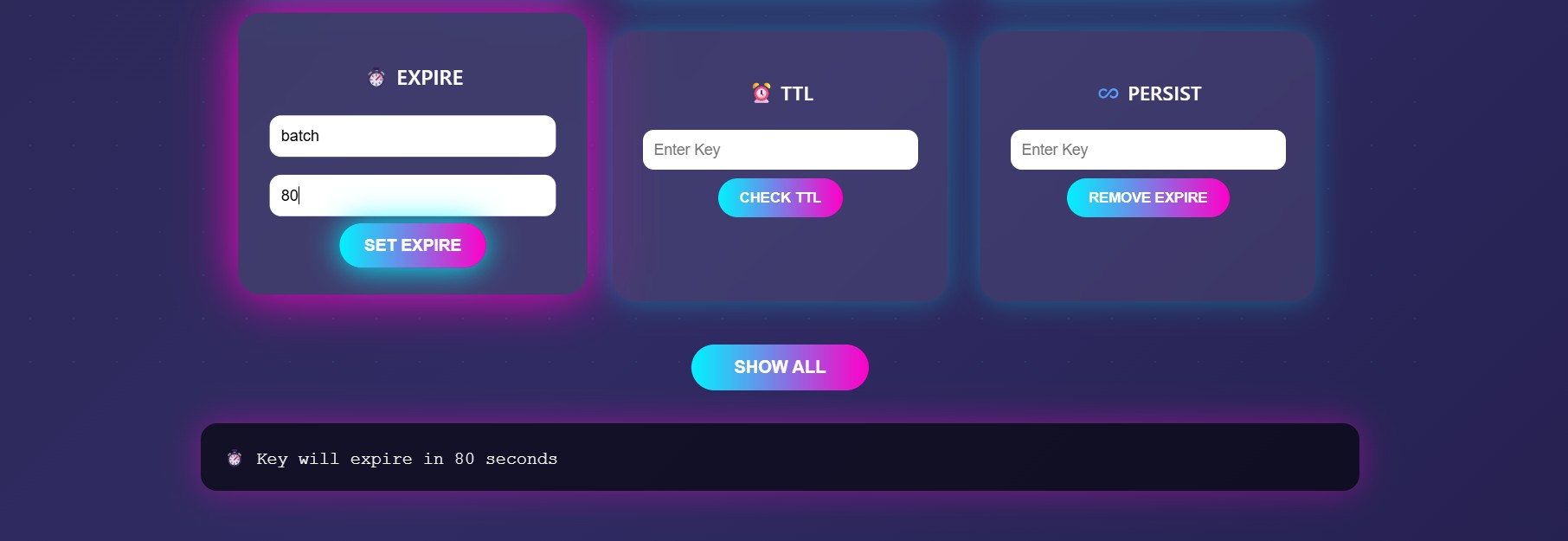
**Step 6:** The user enters the key and click **GET**, it get and show the stored value of that key.



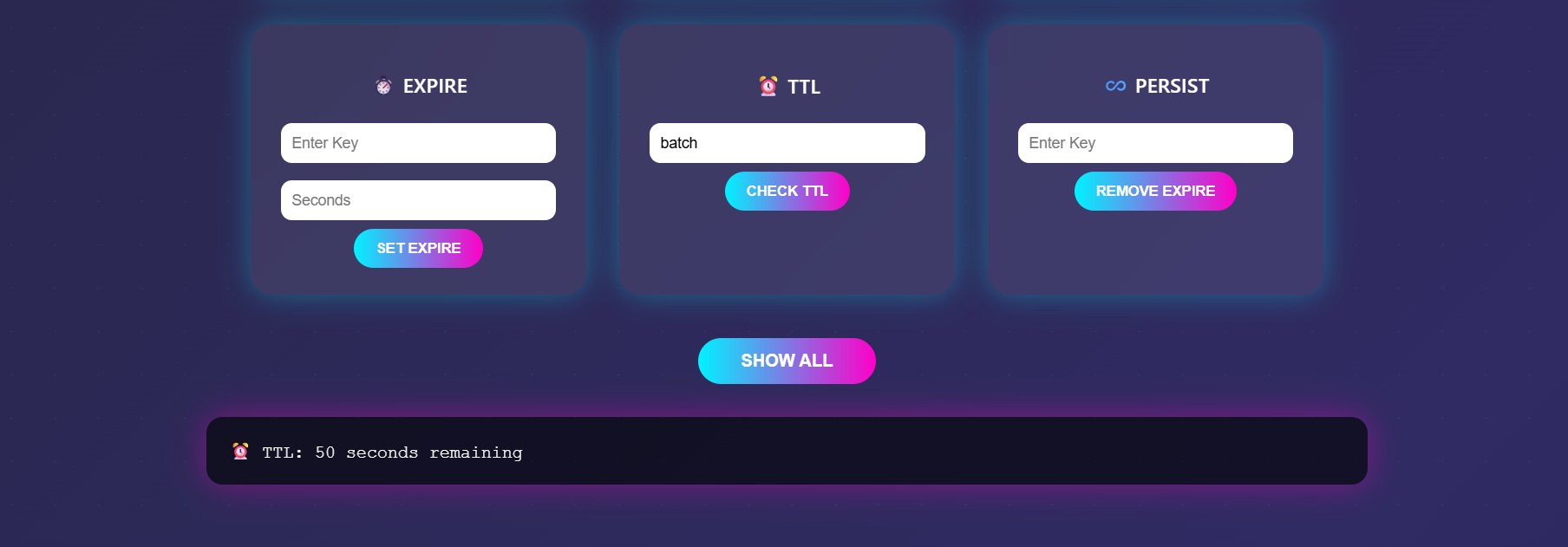
**Step 7:** The user enter the key then click **DELETE** , and the key’s value permanently removes the data from the database



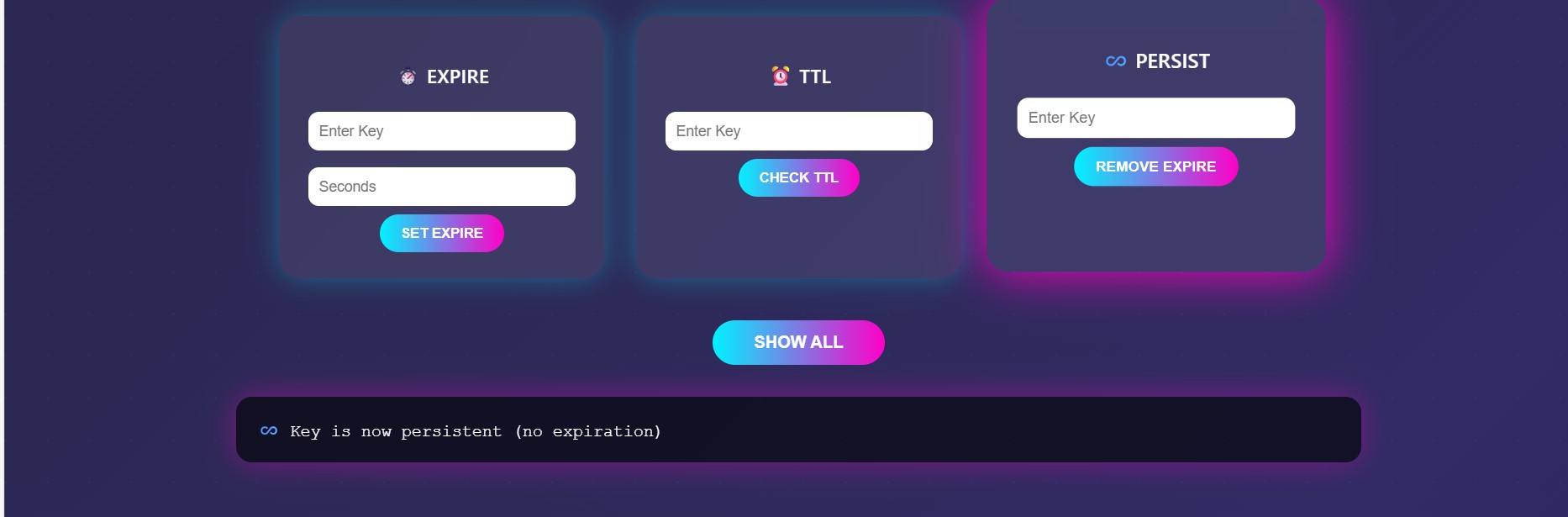
**Step 8 :** The user enters the key, set time in seconds, and click **SET EXPIRE** the key’s value automatically deletes the key after the time



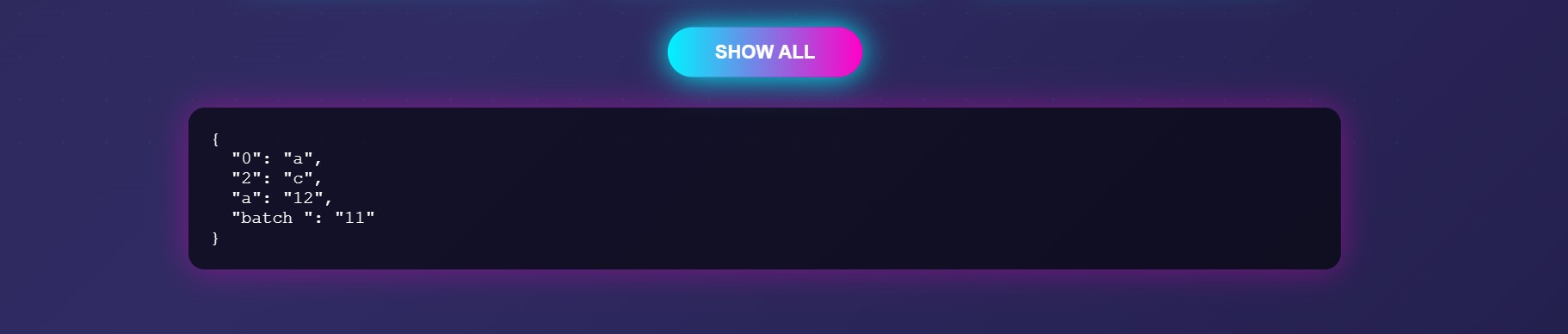
**Step 9:** The user enters the key and click **CHECK TTL** it shows how much time is left before expiry



**Step 10:** The user enters the key and click **REMOVE** **EXPIRE, it** cancels the expiry and keeps the key permanently

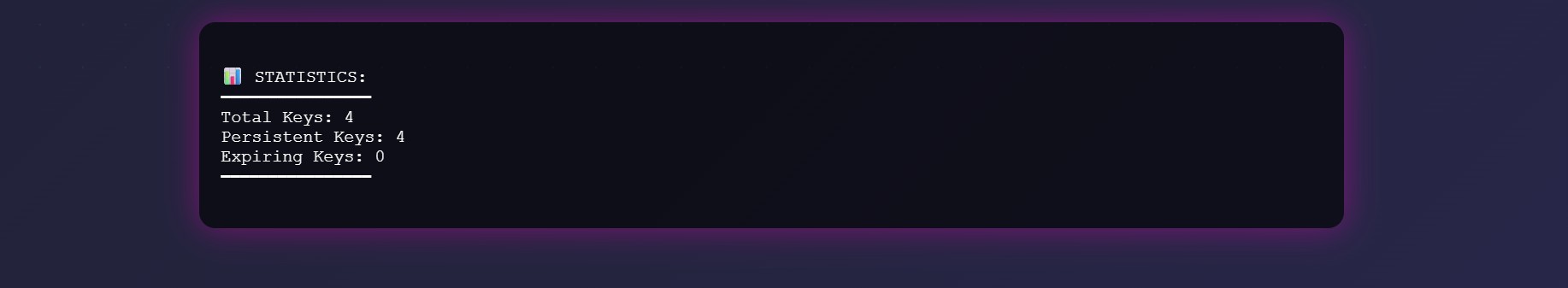


**Step 11**: The user click **SHOW ALL**, it shows what are the key values are stored



**Step 12**: The user click **SHOW STATISTICS**, it shows total stored keys, number of permanent keys, and the number of expiring keys





# Challenges Faced

* **Managing File Storage** – Handling data storage using JSON files sometimes caused unexpected errors, especially during frequent updates.
* **API Design Decisions –** Defining clear and efficient API endpoints for different operations required careful planning and revisions.
* **Implementing Write-Ahead Logging –** Designing the logging mechanism correctly before updating the main file was more complex than expected.
* **Handling Crash Recovery –** Ensuring that data was restored properly after sudden interruptions required careful testing and corrections.
* **Debugging Errors –** Identifying and fixing errors across different modules took significant time and patience.
* **Frontend–Backend Integration –** Ensuring seamless communication between the user interface and backend APIs required careful configuration of routes, request handling, and response formatting.
* **Performance Issues –** Improving response time while handling multiple operations needed optimization efforts.
* **Edge Case Handling –** Managing cases like invalid inputs, missing keys, or expired data required additional validation logic.
* **Time Management –** Balancing development, debugging, documentation, and testing within the given timeline was challenging.
* **Testing and Refinement** – Repeated testing and improvements were necessary to ensure stability and reliability.

# 8. Learnings & Skills Acquired

* **Full Stack Development Exposure –** Gained hands-on experience in integrating frontend interface with backend logic and data storage.
* **Backend Logic Development –** Implemented core operations like PUT, GET, DELETE, EXPIRE, TTL, and PERSIST.
* **Data Structure Implementation** – Improved understanding of dictionaries for key-value storage.
* **CRUD Operations Knowledge –** Gained practical experience in create, read, update, and delete functionalities.
* **Time-Based Expiry Logic (TTL) –** Learned to implement automatic key expiration using time calculations.
* **Problem-Solving Ability –** Enhanced logical thinking by handling edge cases like expired or missing keys.
* **Debugging Skills –** Improved ability to identify and fix runtime and logical errors.
* **Frontend-Backend Integration –** Learned how to connect user interface with backend processing.
* **System Design Concepts –** Understood how real-time key-value storage systems work internally.
* **Collaboration & Teamwork** – Strengthened coordination through Agile methodology, weekly sprints, and feedback sessions.

# 9. Testimonials from team

**Team Member 1:**

“Through this project, I gained practical exposure to handling data storage and retrieval operations. It strengthened my confidence in backend development and improved my coding efficiency.”

**Team Member 2:**

" This project improved my teamwork and coordination skills. Collaborating with team members helped me learn new approaches and understand system integration more clearly."

**Team Member 3:**

" Contributing to this project helped me understand how structured data handling works in modern applications. It also improved my confidence in debugging and optimizing system performance."

**Team Member 4:**

" Developing this application enhanced my analytical thinking and strengthened my backend development skills. I gained valuable experience in implementing real-time operations efficiently."

**Team Member 5:**

“This project helped me gain practical exposure to building a real-time key-value storage system. Implementing different operations improved my understanding of backend logic and system functionality.”

**10. Conclusion**

A Scalable In-Memory Key-Value Store with persistence project was successfully developed with all core operations implemented. The system efficiently handles data storage, retrieval, deletion, and expiry management. The TTL feature enables automatic key expiration after a specified time.

This project enhanced understanding of backend development and data structures.It improved problem-solving skills and real-time system logic implementation. Overall, the project provided valuable practical learning experience during the internship.

**11. Acknowledgements**

Our team expresses sincere appreciation to **Infosys** for providing the opportunity

to work on the **PyKV :A Scalable In-Memory Key-Value Store with Persistence** as part of the internship program. This experience greatly enhanced our knowledge of data storage, full stack development and their practical applications.

We are deeply grateful to our mentor, **Sangeetha**, for continuous guidance, constructive feedback, and invaluable support. Her mentorship played a vital role

in helping us overcome challenges and grow both technically and professionally.

We also extend our appreciation to our team members for their collaboration,

commitment, and consistent efforts, which contributed significantly to the

successful execution of the project. Their teamwork made the internship

experience, productive and enriching.

Finally, heartfelt thanks to everyone who contributed, directly or indirectly, to our learning journey and the successful completion of this project. Their support has

been truly invaluable.