

# Project Proposal: PSDB - Python Sharded Database

Team Number: 31

Purva Dobariya (2024202001)

Malay Damani (2024201081)

## Project Description

PSDB demonstrates a distributed Non-relational database by using Python and gRPC (for communication between nodes). This project is built from scratch which is inspired by modern document-oriented databases. This project handles basic CRUD operations which is doing sharding for horizontal scaling and provides high availability through replication. PSDB acts as a backend-only system which includes a simple CLI for user interaction and testing purposes. By building this project, we explore and implement distributed database principles within the timeline and also leveraging python's accessibility and gRPC's RPC framework.

## Problem Description

Scalable and fault-tolerant databases are needed to handle big-data demands in modern applications which ensure uninterrupted service. Currently available relational-databases can be complex, making it valuable to create a simple, open-source system for our requirement. PSDB focuses the challenge for designing a distributed database via sharding and availability by using python and gRPC (for communication).

## Objectives

- Create relational-database with key-value pair and document-style storage.
- Implement sharding to distribute data across multiple nodes.
- Guaranteed high availability by using replication and failover mechanisms.
- Use Python for core logic and gRPC for inter-node communication.
- Complete a working prototype within 1 month. (with a team of 2 developers)

## Features

1. Core Database Functionality
  - Provides basic CRUD operations (Create, Read, Update, Delete).
  - Document-style storage using JSON-like structures.
  - Simple, easy to use and intuitive CLI, for better user-interaction with the database (e.g., insert, find, update, delete).

## 2. Sharding

- Partition data across multiple nodes using a shard key (e.g., hash-based partitioning).
- Implement basic shard management to redistribute data so that nodes can be added or removed.
- Execute routing logic to direct queries. (for appropriate shard)

## 3. High Availability

- Use data replication across a primary-secondary model (at least 2 replicas per shard).
- If the primary node fails, the secondary node must automatically handle failovers.
- Implement eventual consistency via writes across replicas.

## 4. Distributed Architecture

- Implement gRPC-based communication between nodes. (for replication and query routing).
- Configuration file for defining cluster nodes and their roles (e.g., shard, replica).

## 5. Persistence

- Use basic on-disk storage using files (e.g., JSON files per shard) for data persistence.
- Implement a recovery mechanism to reload data after a node restart.

## Technical Stack

- Language: Python 3.11 (for rapid development and simplicity).
- Communication: gRPC (for efficient, cross-platform RPC).
- Storage: File-based persistence (JSON files for simplicity).
- Libraries: `grpcio`, `protobuf` (for gRPC), `hashlib` (for sharding), `json` (for data storage).

## Project Scope

- Core CRUD operations
- Sharding
- Replication
- CLI for testing
- Persistence

### Timeline (1 Month, 2 Team Members)

Week	Tasks	Outcomes
Week 1	<ul style="list-style-type: none"><li>- Project setup (environment, gRPC boilerplate).</li><li>- Design data model and sharding logic.</li><li>- Define protobuf messages for gRPC.</li></ul>	<ul style="list-style-type: none"><li>- Initial codebase with gRPC setup.</li><li>- Architecture design doc.</li></ul>
Week 2	<ul style="list-style-type: none"><li>- Implement CRUD operations.</li><li>- Integrate a sharding mechanism (hash-based).</li><li>- Develop CLI for testing CRUD.</li></ul>	<ul style="list-style-type: none"><li>- Single-node database with CLI.</li><li>- Sharding prototype.</li></ul>
Week 3	<ul style="list-style-type: none"><li>- Implement replication and failover logic.</li><li>- Build gRPC for inter-node sync.</li><li>- Implement persistence (file-based).</li></ul>	<ul style="list-style-type: none"><li>- Multi-node cluster with replication.</li><li>-Functional persistence.</li></ul>
Week 4	<ul style="list-style-type: none"><li>- Testing and bug fixing.</li><li>- Optimize sharding and replication.</li><li>- Finalize docs and demo prep.</li></ul>	<ul style="list-style-type: none"><li>- Working prototype.</li><li>- Test cases and README.</li><li>- Final Report</li><li>- Documentation</li><li>- Project Demo</li></ul>