# Feature Overview of our Distributed Key-Value Database

Sukhomay, Soumojit, Raj

March 12, 2025

## Feature List and Explanations

### Commands

- **SIGNUP (username, pwd)**: Creates a new user account with the specified username and password.
- **LOGIN (username, pwd)**: Authenticates an existing user with the provided credentials.
- **CREATE (table_name) (model)**: Creates a new table with the given name and optionally specifies additional features.
- **GET (key)**: Retrieves the value associated with the specified key from the current table.
- **PUT (key, value)**: Inserts or updates the value for the specified key in the current table.
- **DELETE (table_name)**: Removes the specified table from the system.
- **LIST (lower_bound, upper_bound)**: Retrieves a list of key-value pairs within the specified key range. Default values (e.g., smallest key, largest key) may be used if no explicit bounds are provided.

### Client Interface

Terminal based interface. Simple query language to make operations.

### Consistency Model

User will be provided with 2 options:

- **Strong Consistency** The availability of the database, under this option, will be compromised.
- **Eventual Consistency**: High availability but some recent writes may not get reflected.

### Provisioned Throughput

- The system takes a specific number of Read Capacity Units (RCUs) and Write Capacity Units (WCUs) during table creation.
- If time permits, we want to change these capacity values dynamically to provide flexibility.

- Requests exceeding the provisioned throughput are rejected (throttling)
- **Short Temporal Spikes**: Bursting will be implemented to absorb short spikes less than approximately 300 times the requested RCUs.
- **Long Spikes and Hot Partitions**: Adaptive capacity to handle non-uniform access pattern and the problem of hot partitions.

## Durability and Corrections

- **Hardware Failures**:
  - Use of Write-Ahead Logging (WAL) to ensure that updates are recorded before being applied.
  - Backups and restores for recovery in case of severe failures.
  - We assume the presence of Stable storage.
- **Silent Data Errors**:
  - Continuous verification within a replication group using checksums to prevent silent data errors.

## Atomicity

- **Atomic by Default**.
- **Non-Atomic Mode**: Will inform the user how many operations could not succeed with error message [only if time permits].

## Security and Isolation

- **Security**: [only if time permits].
- **Isolation**: [only if time permits].

## Query optimizer (if time permits)

We want to bundle operations at the request manager and process them through a simple query optimizer before actual execution in the replication groups.

## Secondary indices (if time permits)

As in NoSQL databases, the user can provide tuples with multiple keys (i.e each tuple is of the form $< Primarykey >< Secondarykey1 >< Secondarykey2 > ... < value >$)

The primary key will be used for partitioning the table into replication groups. But, the user can also set some key as Secondary index for efficient LIST operation via that key.