NoSQLite

Introduction

1- Srishti Jain (110026562)

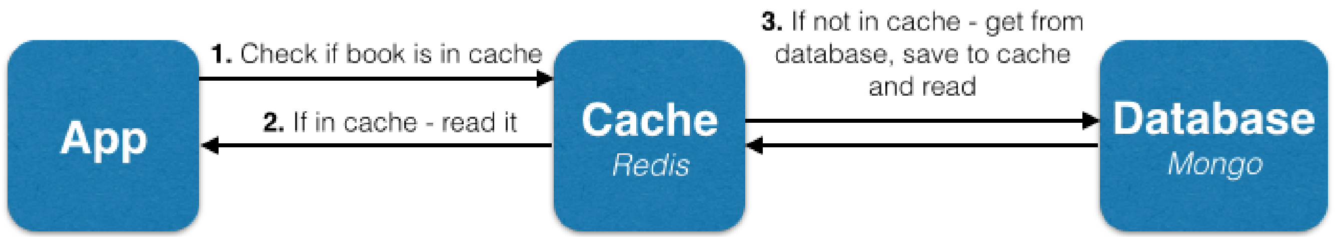
2- Siddharth Paliwal (110036256)

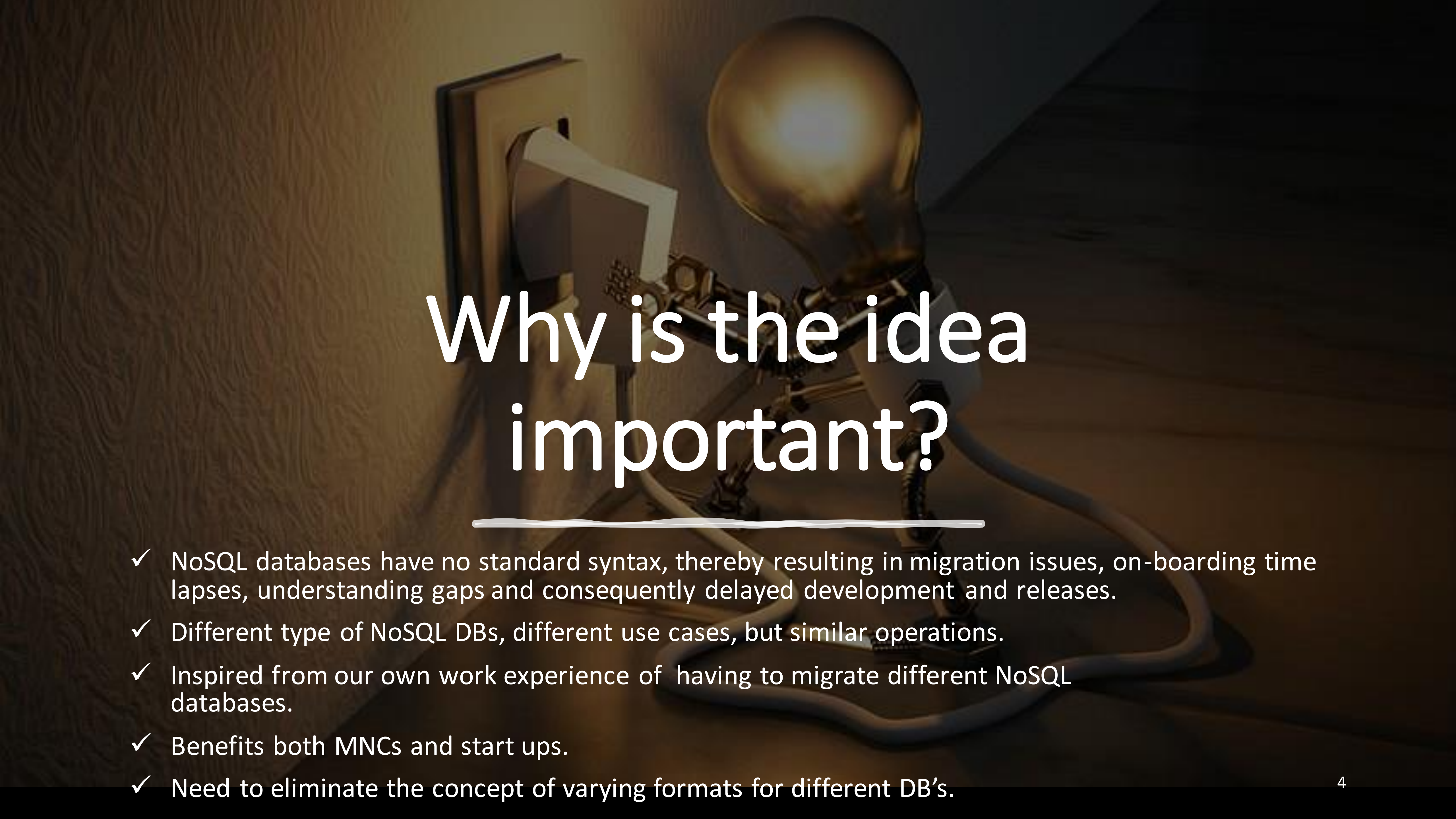
3- Margaret Arulmalar Rebeka (110026527)

4 – Jimish Shah (110014819)

Problem Statement

- ✓ The goal is to create a database access library that provides the illusion of embedding NoSQL in regular C++ code.
- ✓ Unlike SQL, there is no specific defined format for NoSQL.
- ✓ The library will provide users a common format, unified interface, and a single point of access to multiple NoSQL databases.

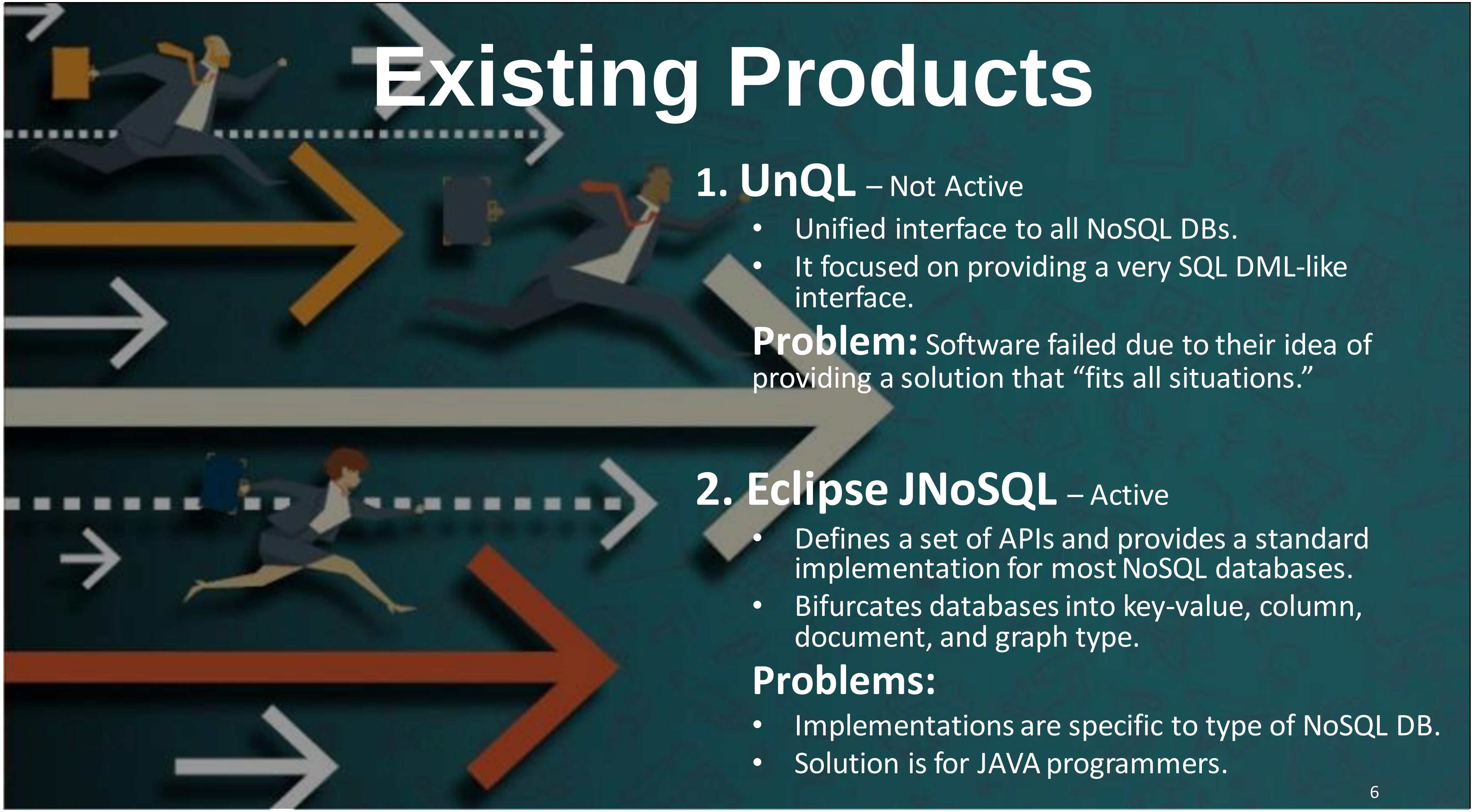

App
1. Check if book is in cache
2. If in cache - read it
Cache
Redis
3. If not in cache - get from database, save to cache and read
Database
Mongo



Why is the idea important?

- ✓ NoSQL databases have no standard syntax, thereby resulting in migration issues, on-boarding time lapses, understanding gaps and consequently delayed development and releases.
- ✓ Different type of NoSQL DBs, different use cases, but similar operations.
- ✓ Inspired from our own work experience of having to migrate different NoSQL databases.
- ✓ Benefits both MNCs and start ups.
- ✓ Need to eliminate the concept of varying formats for different DB's.

Market Analysis



Existing Products

1. **UnQL** – Not Active

- Unified interface to all NoSQL DBs.
- It focused on providing a very SQL DML-like interface.

Problem: Software failed due to their idea of providing a solution that “fits all situations.”

2. **Eclipse JNoSQL** – Active

- Defines a set of APIs and provides a standard implementation for most NoSQL databases.
- Bifurcates databases into key-value, column, document, and graph type.

Problems:

- Implementations are specific to type of NoSQL DB.
- Solution is for JAVA programmers.

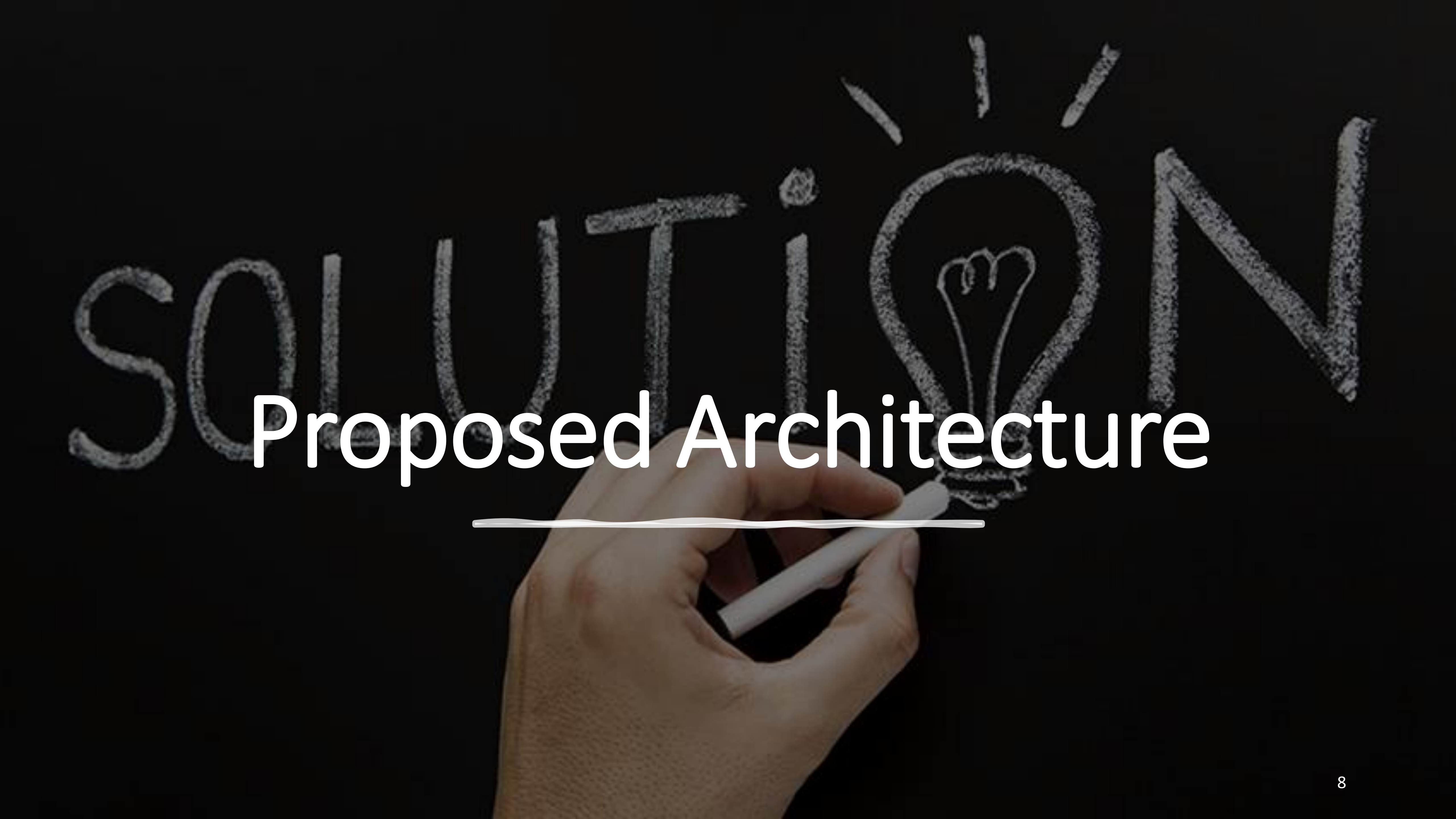
How do we stand out?

Unlike UnQL,

- Fit most of the situations while handling others smartly.
- Provide a generic format for similar commands and handle different ones efficiently as well.

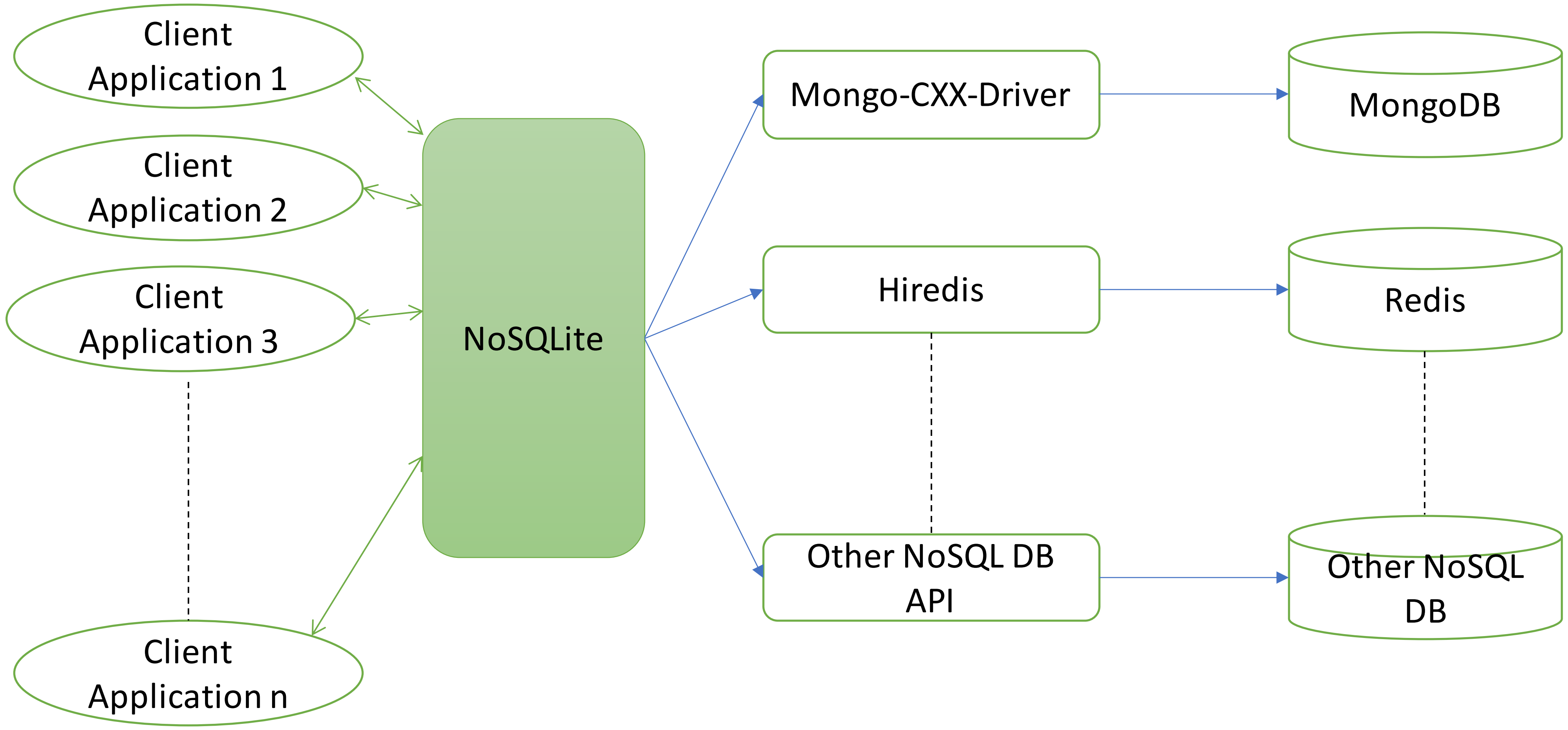
Unlike Eclipse JNoSQL,

- Library with C++ compatibility
- Provide a unified interface without differentiating the DB types.



Proposed Architecture

MODEL


Client Application 1
Client Application 2
Client Application 3
Client Application n
NoSQLite
Mongo-CXX-Driver
Hiredis
Other NoSQL DB API
MongoDB
Redis
Other NoSQL DB

Features

- Unified Interface, single library shall give access to all the supported NoSQL DBs.
- Currently support has been added for MongoDB and Redis, keeping in mind the basic use case of a cache for persistent database.
- A generic format for quicker migration and integrations.
- Standardization of syntax, to help set a defined format for developers to follow.
- Uses existing performance trained APIs to provide optimum performance
- Basic CRUD Operations implemented for MongoDB and Redis.

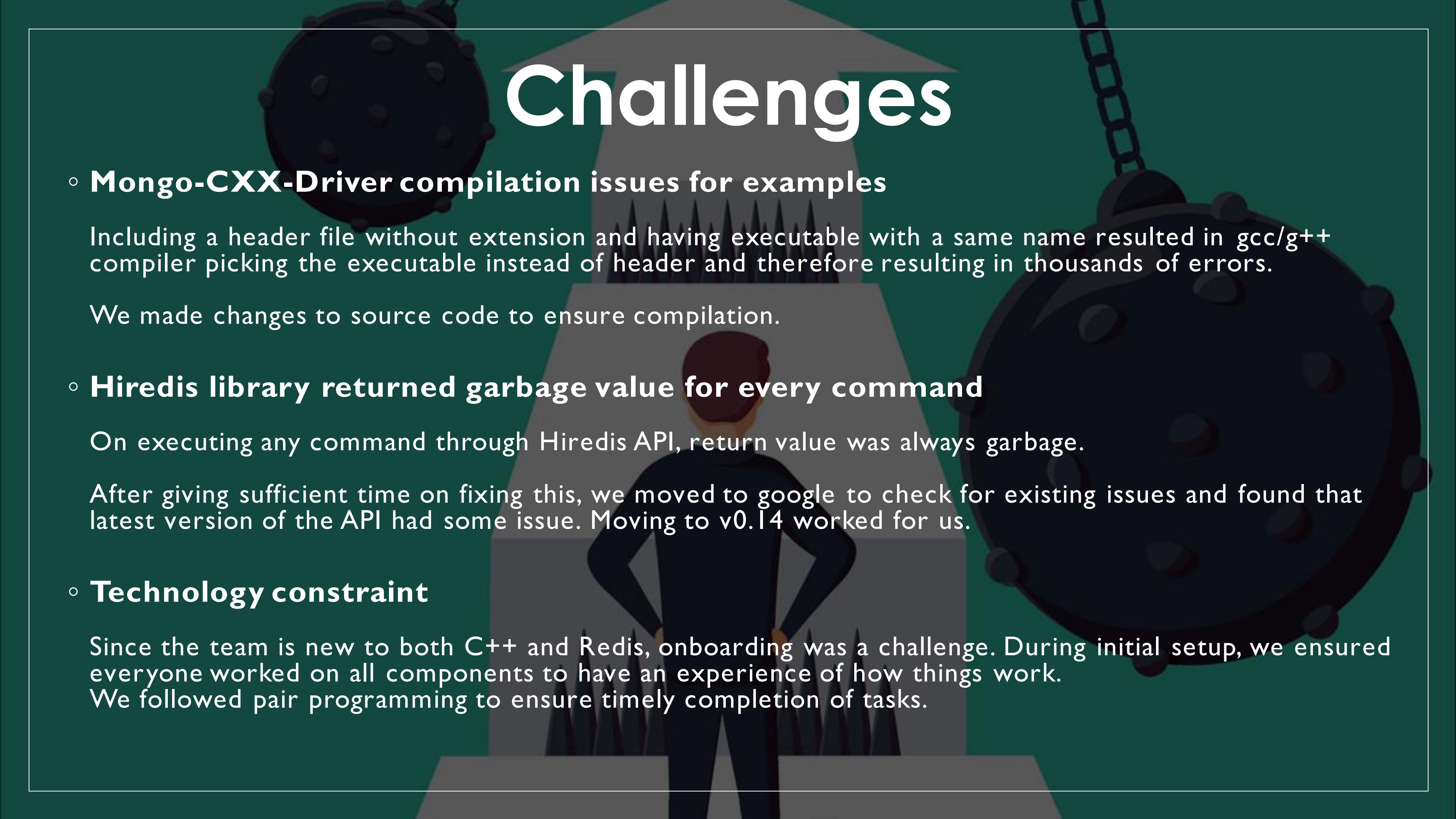
DEMO
LOADIND...

PERFORMANCE

S. No.	Operation	Database	Original API (Time Taken in Seconds)	NoSQLite (Time Taken in Seconds)
1.	SET	Redis	14.666216	15.673197
2.	HSET	Redis	14.532563	15.918098
3.	GET	Redis	14.468994	15.293276
4.	HGET	Redis	14.854378	15.580401
5.	DEL	Redis	14.561485	15.415330
6.	HDEL	Redis	14.751485	15.494107
7.	INSERT_ONE	MongoDB	123.136163	31.514089
8.	DELETE_ONE	MongoDB	108.133453	43.876428
9.	FIND_ONE	MongoDB	87.7462784	29.876479

Future Scope

- ✓ Adding support for MongoDB and Redis's other features including sub documents, aggregation and bulk operations.
- ✓ Extending API to other NoSQL databases.
- ✓ Broaden scope of API to other languages. C++ shall provide the best performance and any wrappers for varying languages written over the code shall also be performance trained.
- ✓ Adding support for a bash utility to provide users an easy access environment.



Challenges

- **Mongo-CXX-Driver compilation issues for examples**

  Including a header file without extension and having executable with a same name resulted in gcc/g++ compiler picking the executable instead of header and therefore resulting in thousands of errors.

  We made changes to source code to ensure compilation.

- **Hiredis library returned garbage value for every command**

  On executing any command through Hiredis API, return value was always garbage.

  After giving sufficient time on fixing this, we moved to google to check for existing issues and found that latest version of the API had some issue. Moving to v0.14 worked for us.

- **Technology constraint**

  Since the team is new to both C++ and Redis, onboarding was a challenge. During initial setup, we ensured everyone worked on all components to have an experience of how things work.
  We followed pair programming to ensure timely completion of tasks.

References

http://mongocxx.org/mongocxx-v3/installation/linux/

http://mongoc.org/libmongoc/current/installing.html

https://developer.mongodb.com/community/forums/t/c-and-c-driver-for-debian-ubuntu-users/2732

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-debian/

https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-redis-on-ubuntu-18-04

https://github.com/mongodb/mongo-cxx-driver

https://www.bogotobogo.com/cplusplus/cplusplus11.php

https://en.cppreference.com/w/cpp/container/vector

https://github.com/redis/hiredis

https://github.com/mongodb/mongo-cxx-driver

https://github.com/mongodb/mongo-c-driver

https://github.com/neloe/MongoDB-Cpp

https://github.com/mongodb-labs/mangrove

https://stackoverflow.com/questions/59515425/hiredis-rediscommand-returns-null-for-everything-on-raspberry-pi-4

https://en.cppreference.com/w/cpp/language/variadic_arguments

https://www.geeksforgeeks.org/variadic-function-templates-c/

https://forum.qt.io/topic/63629/bug-qtcreator-choked-with-errors-when-compiling-using-qsettings/2

Merci de
votre
attention