UNIVERSITY OF BUEA

****

FACULTY OF ENGINEERING AND TECHNOLOGY

****

DEPARTMENT OF COMPUTER ENGINEERING

****

CEF 440: MOBILE PROGRAMMING

LEVEL 400: 2022/2023 ACADEMIC YEAR

Project Title:

DESIGN AND IMPLEMENTATION OF A PASSENGER POSITIONING SYSTEM

TASKS 4:

DATABASE DESIGN AND IMPLEMENTATION

Supervisor:

DR NKEMENI VALERY

Group 8 Members:

NGI KEVIN AYUK	FE20A076
NDEM LARRY NCHENY	FE20A071
TASHA OLIVIA	FE20A0112
TAGHA WILFRED	FE20A0122
A. CLINTON MBURLI	FE20A010

Contents

INTRODUCTION

The advent of technology has transformed the way we travel, and the demand for efficient and reliable transportation solutions continues to grow. In this era of digital connectivity, a robust and scalable passenger positioning system has become indispensable for modern transportation networks. This system enables passengers to book or request rides seamlessly while providing drivers with the necessary tools to accept and fulfill those requests promptly.

The purpose of this report is to present the design and implementation of a comprehensive database for our passenger positioning system FindMoto. The database plays a pivotal role in storing and managing the vast amount of data generated by the system, ensuring its reliability, availability, and performance. By leveraging cutting-edge technologies and best practices in database design, we aim to create a highly efficient and scalable solution that meets the evolving needs of both passengers and drivers.

Key Objectives:

1. Designing an intuitive and user-friendly database schema to store passenger and driver information, ride details, and system configurations.
2. Implementing a robust data storage and retrieval mechanism to ensure fast and accurate processing of ride requests, driver availability, and booking information.
3. Ensuring data consistency, integrity, and security through appropriate data modeling, normalization techniques, and access controls.
4. Optimizing the database performance to handle concurrent requests efficiently, reducing latency, and ensuring a seamless user experience.
5. Integrating the database with the passenger positioning system's frontend and backend components to facilitate real-time data updates and synchronization.
6. Employing appropriate backup and recovery mechanisms to safeguard critical data and minimize the risk of data loss or system downtime.
7. Conducting thorough testing and evaluation of the database system to validate its functionality, reliability, and scalability.

By successfully implementing an efficient and reliable database system, we can empower passengers to easily book or request rides, and drivers to accept and fulfill those requests in a timely manner. This will ultimately enhance the overall efficiency and convenience of the transportation network, leading to improved customer satisfaction and increased operational efficiency for service providers.

In the following sections of this report, we will delve into the details of our database design, implementation approach, and the challenges encountered during the development process.

Additionally, we will discuss the testing methodologies employed and present the results and insights gained. Finally, we will conclude with a summary of the achieved outcomes and recommendations for future enhancements and expansions of the passenger positioning system,FindMoto's database.

DATABASE

Definition:

A database is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, management, and manipulation of information. It is designed to store and manage large volumes of data, providing a centralized and controlled environment for data storage and access.

Databases are essential for storing and organizing structured information in a manner that facilitates easy searching, querying, and analysis. They are widely used in various domains such as business, finance, healthcare, research, and many others.

A database typically consists of tables, which are composed of rows and columns. Each table represents a specific entity or concept, and the rows and columns contain the actual data related to that entity. The rows, also known as records or tuples, represent individual instances or entries, while the columns, also called fields or attributes, define the specific characteristics or properties of the data.

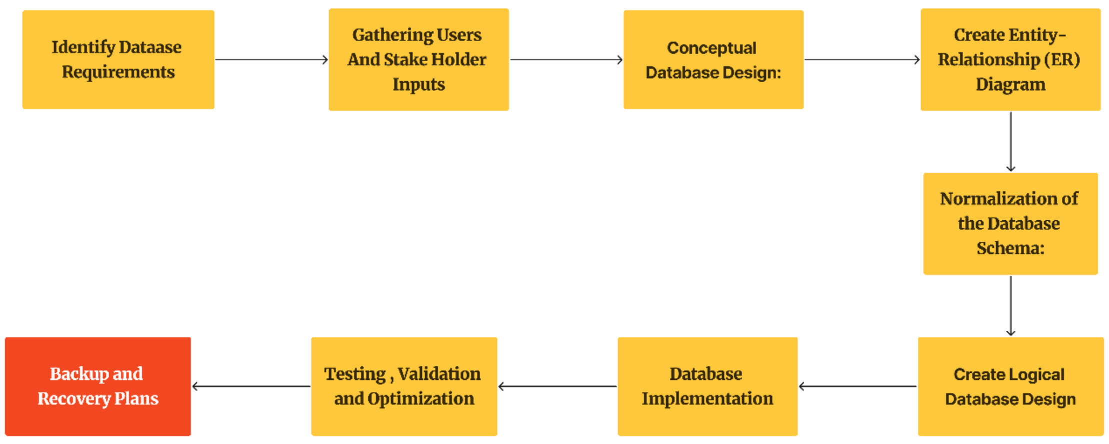
The structure and organization of a database are defined by a database schema, which specifies the tables, their fields, and the relationships between them. The schema provides a blueprint for how the data is organized and helps ensure data integrity and consistency.

To interact with a database, users can perform various operations such as inserting, updating, and deleting data, as well as retrieving and querying information using specific programming languages or query languages such as SQL (Structured Query Language).

Databases can be classified into different types based on their architecture and functionality. Some common types include relational databases, which use tables and enforce relationships between them using keys; NoSQL databases, which are designed to handle unstructured or semi-structured data; and object-oriented databases, which store data as objects with properties and behaviors.

Overall, databases play a crucial role in modern information systems, providing a structured and efficient way to store, manage, and access data for a wide range of applications and use cases.

Database Design And Implementation Process:



Identify Dataase Requirements
Gathering Users And Stake Holder Inputs
Conceptual Database Design:
Create Entity-Relationship (ER) Diagram
Normalization of the Database Schema:
Create Logical Database Design
Database Implementation
Testing , Validation and Optimization
Backup and Recovery Plans

Figure 2: Database Design and Implementation Process Flow

Types Of Data To Be Stored In Our Systems Database(FindMoto)
NoSQL database on Firebase for the FindMoto app:

1. User Data:

- Driver Information: Details such as driver ID and Liscense, name, contact information, vehicle information (e.g., car model, license plate), driver rating, and any other relevant details.

- Passenger Information: Details such as passenger ID, name, contact information(Phone and Email), and any other relevant details.

2. Ride Data:

- Ride Information: Data related to individual rides, including ride ID, driver ID, passenger ID, pickup location, drop-off location, ride status (e.g., ongoing, completed, canceled), fare details (suggested price), and timestamps for various ride events (e.g., pickup time, drop-off time).

- Route Information: This may include the route taken during the ride, including the coordinates or addresses of waypoints or checkpoints.

3. Geolocation Data:

- Driver Location: Real-time or periodic updates of driver location, allowing passengers to see nearby available drivers.
- Passenger Location: Passenger's current location or pickup location details, used for matching with nearby available drivers.

4. User Authentication and Authorization Data:

- User Credentials: Data such as usernames, passwords, or authentication tokens required for user login and session management.

5. User Feedback and Ratings:

- Driver Ratings: Feedback and ratings provided by passengers for drivers.
- Ride Ratings: Feedback and ratings provided by passengers for each ride experience.

DATABASE MANAGEMENT SYSTEMS (DBMS)

Generally, there are two different types of databases based on the relationship of main objects of the database. These are relational and non-relational database. Relational databases can be referred to as SQL databases while non-relational databases referred to as NoSQL databases .

NON-RELATIONAL DATABASE

Non-relational databases are different from relational databases in that they store their data in a non-tabular form. Data of this format is stored using 4 main formats given below:

- Key-value storage: It stores every single item as a key or attribute holding its value together.
- Document-oriented Database: It stores data as JSON-like files. It helps developers in storing data by using the same document-model format as used in the application code.
- **Graph database: I**t is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.
- **Wide-column stores:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

Differences Between SQL and NoSQL

DATA INTEGRITY

SQL databases must comply with ACID rules (Atomicity, Consistency, Isolation, and Durability) to resolving transactions. Hence ensuring data integrity. On the other hand, NoSQL do not follow these rules strictly. Rather, they follow the CAP (Consistency, Availability, Partition tolerance) theorem. As a result, NoSQL provide varying levels of data integrity guarantees. The ACID and CAP rules are explained below as follows.

- Atomicity means all transactions must succeed or fail completely as a unit. A transaction cannot be partially complete, even in the case of system failure.
- Consistency means that all nodes in the network see the same data at the same time. This rules validates and prevents corruption.
- Isolation prevents concurrent transactions from affecting each other. Transactions must result in the same final state as if they were run sequentially, even if they were run in parallel.
- Durability makes transactions final. Even system failure cannot undo the effects of a successful transaction.
- **Availability** is a guarantee that every request receives a response about whether it was successful or failed. However, it does not guarantee that a read request returns the most recent write. The more number of users a system can cater to better is the availability.
- **Partition Tolerance** is a guarantee that the system continues to operate despite arbitrary message loss or failure of part of the system. In other words, even if there is a network outage in the data center and some of the computers are unreachable, still the system continues to perform.

Scalability

Most SQL databases can be scaled vertically by increasing the processing power of existing hardware. NoSQL databases use a master-replica architecture which scales better horizontally, with additional servers or nodes, but also have the ability to be scaled vertically.

Support and communities

SQL databases represent massive communities, stable codebases, and proven standards. Multitudes of examples are posted online and experts are available to support those new to working with relational data. NoSQL technologies are being adopted quickly, but communities remain smaller and more fractured.

Structure

SQL database schema always represent relational, tabular data, with rules about consistency and integrity. They contain tables with columns (attributes) and rows (records), and keys have constrained logical relationships. NoSQL databases are not of this format but rather store data either as Key-value, Document or graph.

Schema

Schema helps determine how tables are configured, and data is stored. SQL databases follow a rigid structure that ensures optimized storage and data integrity but limits flexibility. NoSQL database uses a dynamic schema. Hence, it doesn't follow a predefined data structure.

WHY CHOOSE NOSQL

Based on these differences it is tempting to choose a relational database. But looking at the nature of the data we are dealing with, We choose NoSQL. The reasons for this selection are given below.

- Flexible Data Model: NoSQL databases are designed to handle unstructured and semi-structured data. Drivers have no relation with passengers. Hence it was needless to choose a relational database.
- Horizontal Scalability: NoSQL databases are often designed to scale horizontally across commodity hardware, which allows for high availability and performance. This is important for real-time data applications where data is generated at a high rate and needs to be processed quickly.
- Low Latency: NoSQL databases are often designed to provide low latency access to data, which is important for real-time data applications where data needs to be

processed and analyzed in near real-time. This is important since we want passengers to easily get taxis suggested to them.

◆

DIFFERENT TYPES OF NOSQL DATABASES

CASSANDRA

MONGODB

FIREBASE

REDIS

ORACLE NOSQL

COUCHBASE

Of all NoSQL databases listed we choose to use firebase for the reasons listed below.

WHY CHOOSE FIREBASE

- ◆ Firebase is explicitly designed for mobile application development, and its entire user interface and on-boarding flow are built around that use case. As such, Firebase has more features for mobile application. On the other hand, MongoDB Atlas is built for general-purpose data development, and has many more buttons and knobs to tweak performance, integration with third-party business intelligence tools, and advanced features for large data installations like Atlas Data Lake.
- ◆ Firebase offers built-in authentication services, which allow for secure access control to the database. It also provides SSL encryption and secure connections to ensure that data is transmitted securely. MongoDB, on the other hand, requires developers to implement their own security measures, although it does provide support for SSL encryption and secure connections.
- ◆ Firebase performs well in handling **real-time data** for mobile applications. With Firebase's r**eal-time database**, you can get low latency in offline mode. MongoDB provides superior performance for web-based applications. It also offers the strong querying capability to

handle voluminous data both on-premises and on the cloud. It is designed to provide high performance for high-traffic web applications.

CONS OF USING FIREBASE

Although firebase is high has a performance, it is unreliable since it uses a cloud data storage.

DATABASE DESIGN AND IMPLEMENTATION FOR FindMoto.

DATABASE SCHEMA

A Database Schema defines how data is organised in a database. This includes logical constraints such as table names, fields, data types and the relationships between the entities. It represents the skeleton structure that represents the logical view of the entire database. There are two types;

Physical Database Schema

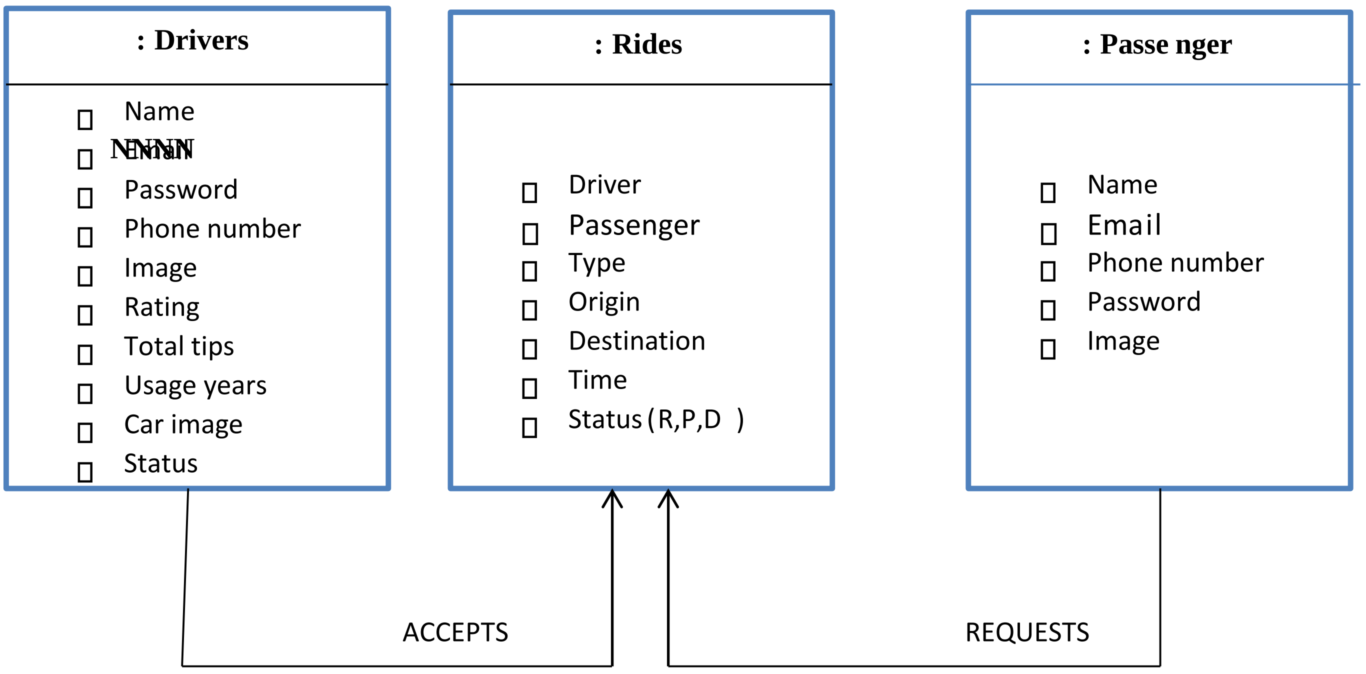
This shows pertains to the actual storage of data and its form of storage like files, indices etc. it defines how the data will be stored in a secondary storage.

Logical Database Schema

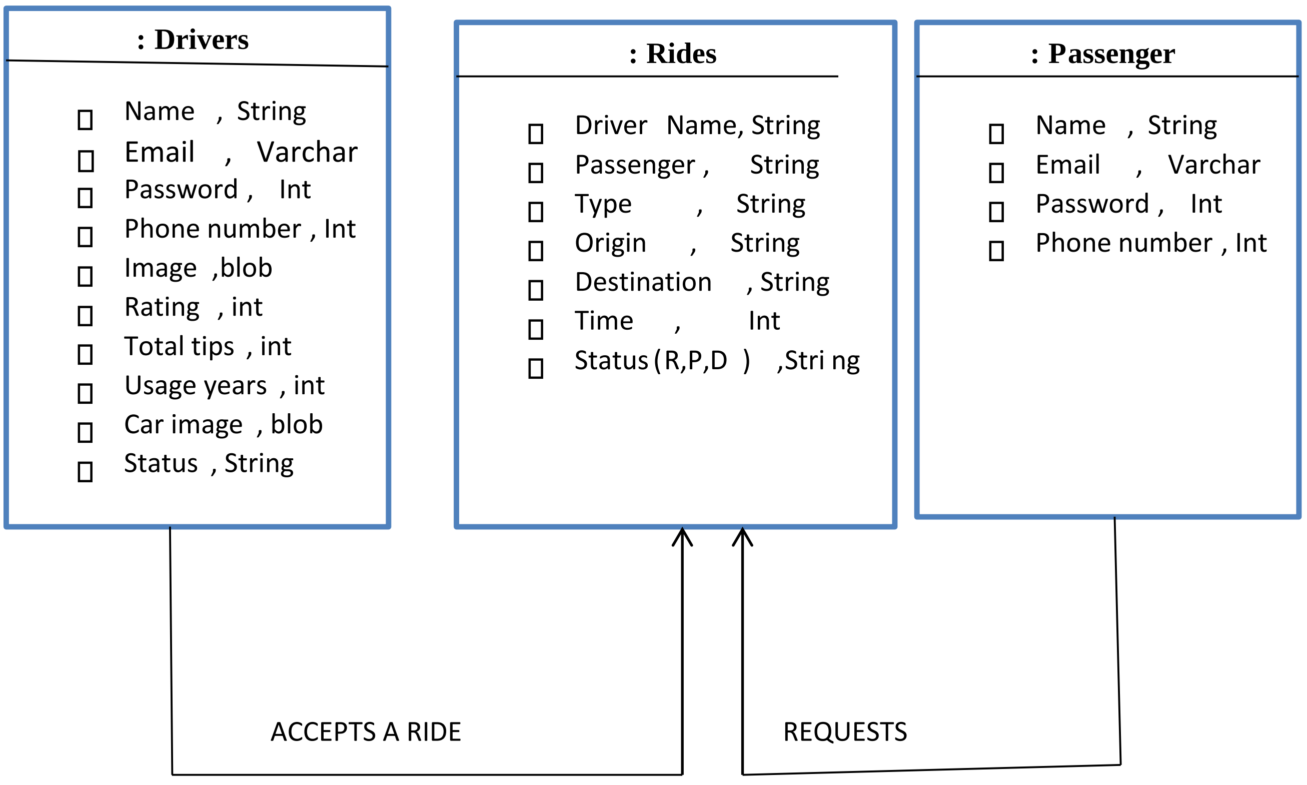
It defines all the logical constraints that need to be applied on the data stored. But basically, the two schema can be used interchangeably.

For our system FindMoto, we will be illustrating these two schema based on some of the actors whose data will be collected and stored in our database**.**

PHYSICAL SCHEMA



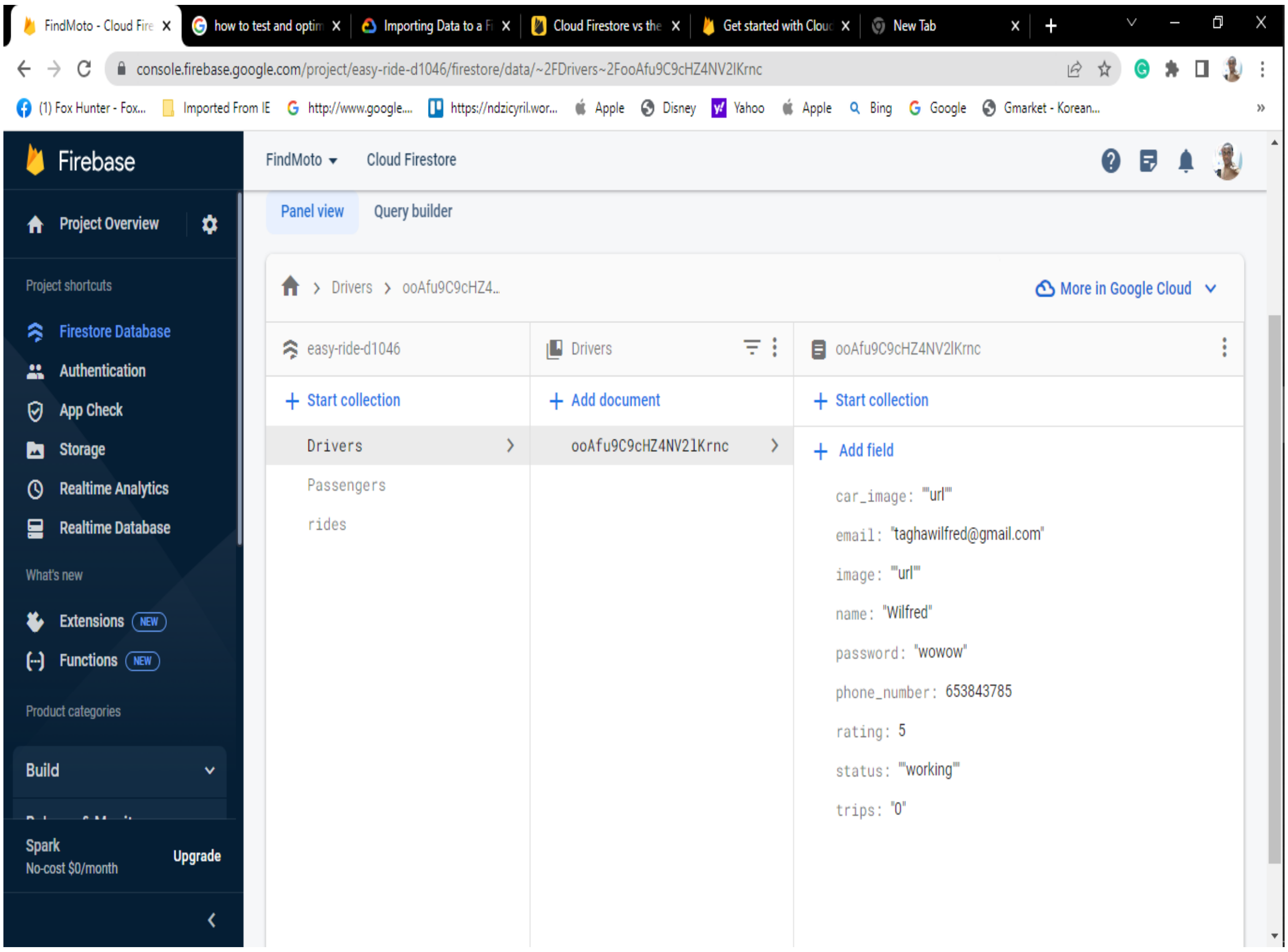
LOGICAL SCHEMA



This Schema gave us a blueprint of the various tables of the database and their attributes.

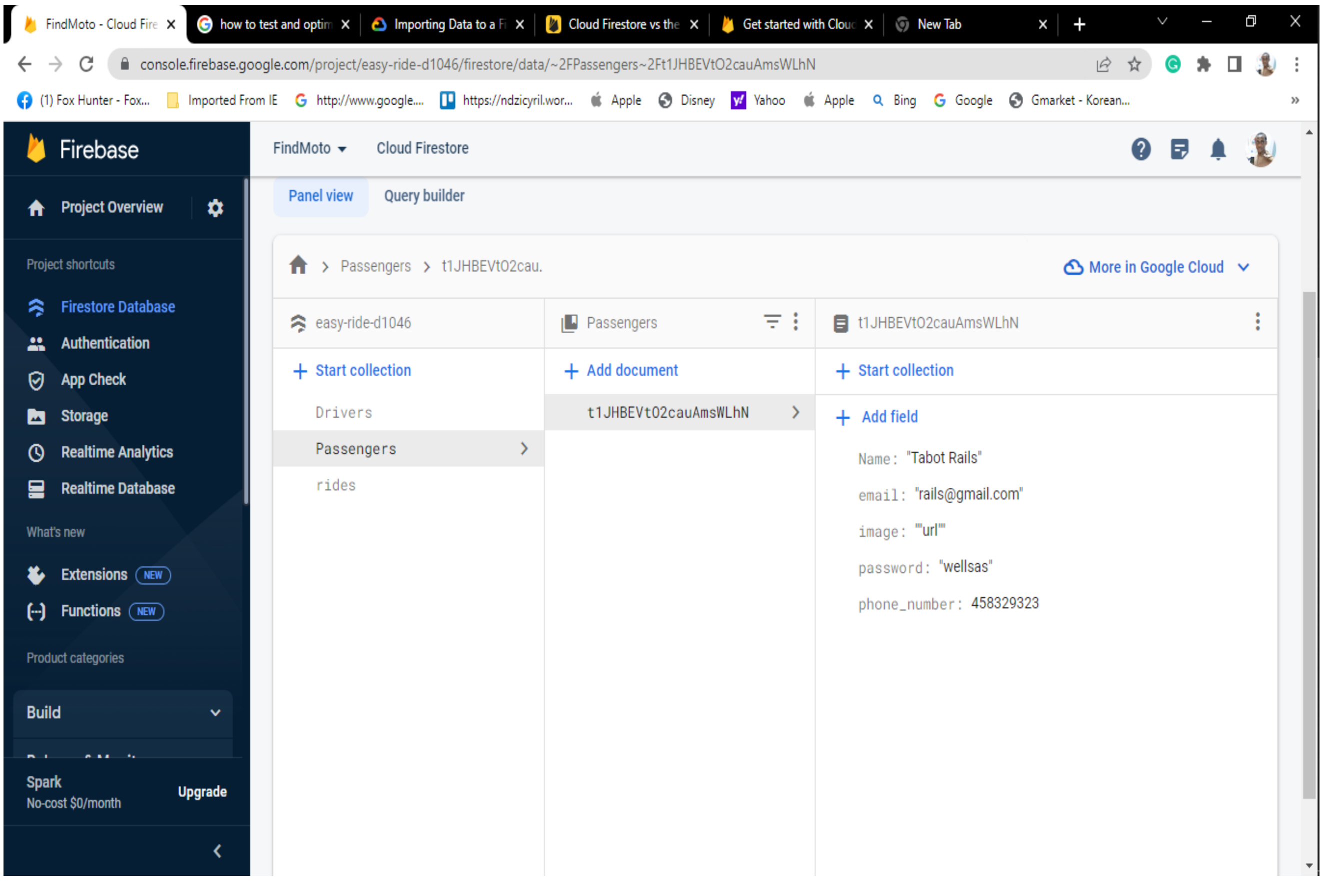
IMPLEMENTATION OF THE DATABASE

Firebase has a very easy way of implimenting its data base. We just have to create the database, create collections and push doccumnets to each collection.
A collection is just like a dable while a doccumnet is like a record. Here are the pictures of the database implimentation.



```
car_image: ""url""
email: "taghawilfred@gmail.com"
image: ""url""
name: "Wilfred"
password: "wowow"
phone_number: 653843785
rating: 5
status: ""working""
trips: "0"
```

Createion of the drivers Collection



console.firebase.google.com/project/easy-ride-d1046/firestore/data/~2FPassengers~2Ft1JHBEVtO2cauAmsWLhN

Passengers > t1JHBEVtO2cau.

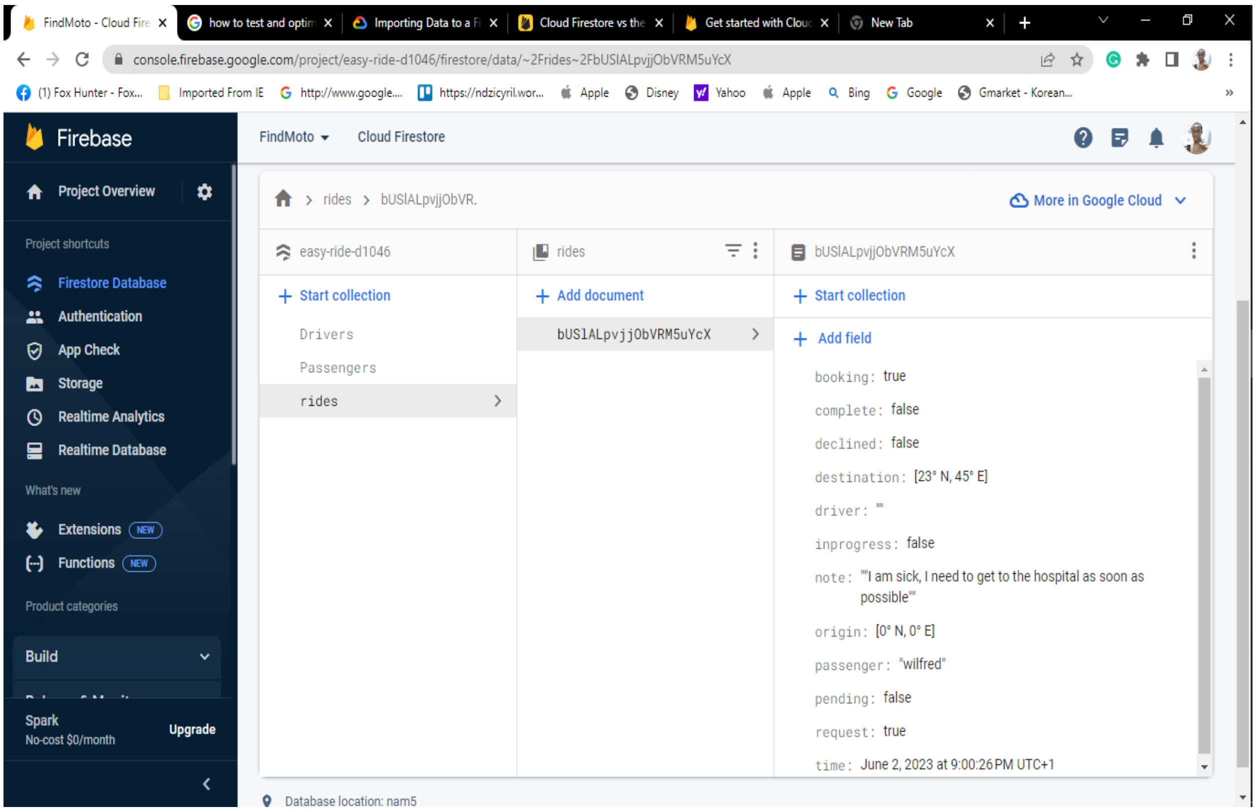
easy-ride-d1046: Drivers, Passengers, rides

Passengers: t1JHBEVtO2cauAmsWLhN

t1JHBEVtO2cauAmsWLhN

Name:	"Tabot Rails"
email:	"rails@gmail.com"
image:	""url""
password:	"wellsas"
phone_number:	458329323

Creation of the passenger Collection



console.firebase.google.com/project/easy-ride-d1046/firestore/data/~2Frides~2FbUSlALpvjjObVRM5uYcX

rides > bUSlALpvjjObVR.

easy-ride-d1046: Drivers, Passengers, rides

rides: bUSlALpvjjObVRM5uYcX

bUSlALpvjjObVRM5uYcX

booking:	true
complete:	false
declined:	false
destination:	[23° N, 45° E]
driver:	""
inprogress:	false
note:	""I am sick, I need to get to the hospital as soon as possible""
origin:	[0° N, 0° E]
passenger:	"wilfred"
pending:	false
request:	true
time:	June 2, 2023 at 9:00:26 PM UTC+1

Database location: nam5

Creation of the rides collection

Testing and Optimization

The database schema went through a series of testings which led to its optimization to this present state. Initially the database had five different collection. Querying the data base was quite difficult and considering the NoSQL structure of our data base, relationships were not an option. It had to be restructued into three major collections. After this seperation, some fields pertaining to a collection for a doccument were ambigus and made it hard to parse in data. They were fields such as status, and Type which were futher brocken into different fields as booking and request for type and inprogress, pending, complete and declined for status.

REFERENCES

https://www.javatpoint.com/types-of-databases

https://www.mongodb.com/databases/non-relational

https://www.talend.com/resources/sql-vs-nosql/#:~:text=SQL%20is%20the%20programming%20language,generally%20do%20not%20use%20SQL.

https://successive.cloud/sql-nosql-databases-key-differences-use-cases/

https://www.geeksforgeeks.org/difference-between-sql-and-nosql/

https://cloudxlab.com/assessment/displayslide/345/nosql-cap-theorem#:~:text=NoSQL%20can%20not%20provide%20consistency,%2C%20Availability%2C%20and%20Partition%20Tolerance.

https://www.mongodb.com/firebase-vs-mongodb#:~:text=Both%20Firebase%20and%20MongoDB%20are,and%20high%2Dperformance%20use%20cases.