EVAT Project: MongoDB Database Architecture Document

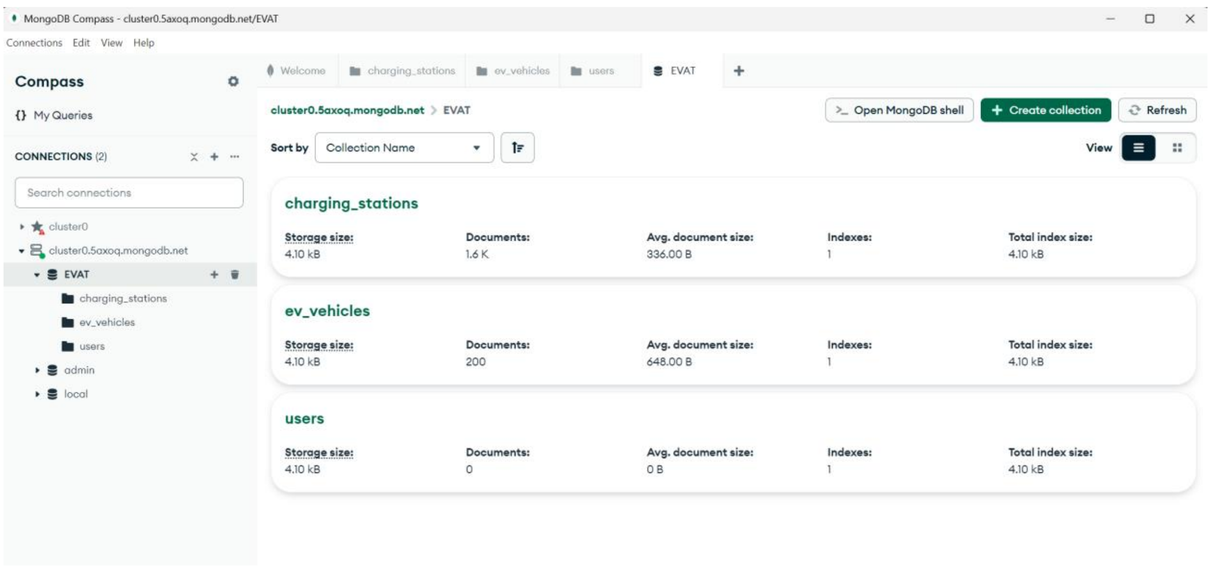
1. Overview

The EVAT project utilizes MongoDB Atlas as its primary database to store and manage data for various features, including EV station finder, trip planner, and user management. This document provides a detailed explanation of the database architecture, including the collections, fields, data types, and usage guidelines.

2. Database Name

Database Name: EVAT

This database contains all collections related to the EVAT project, including data on charging stations, electric vehicles, user profiles, and real-time user data.



Collection	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
charging_stations	4.10 kB	1.6 K	336.00 B	1	4.10 kB
ev_vehicles	4.10 kB	200	648.00 B	1	4.10 kB
users	4.10 kB	0	0 B	1	4.10 kB

3. Collections and Their Structures

The EVAT database consists of three main collections:

1. charging_stations
2. ev_vehicles
3. users

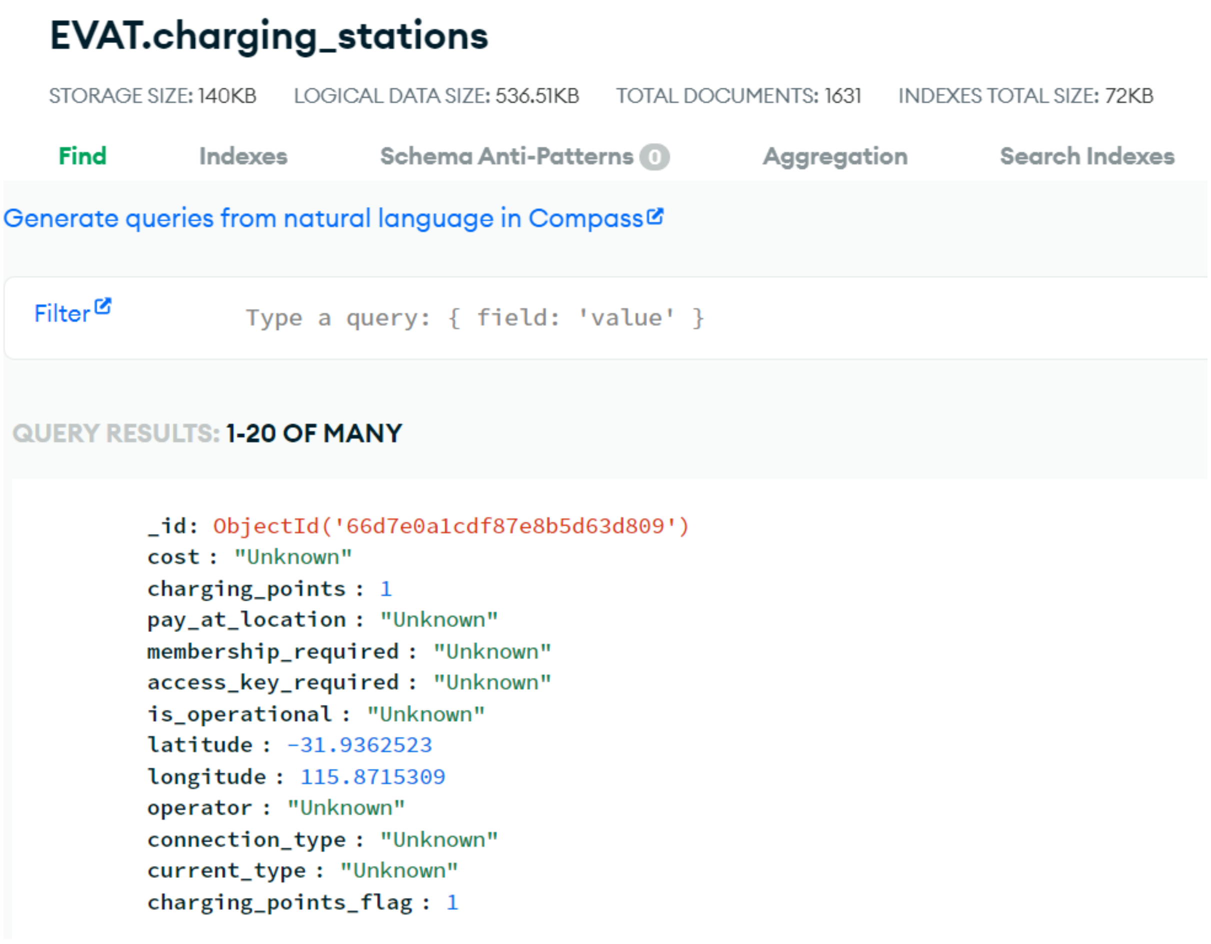
Each collection is designed to store specific types of data. Below, we outline the structure, fields, and data types for each collection.

3.1 Collection: charging_stations

This collection stores data about electric vehicle (EV) charging stations.

Fields:

Field Name	**Data Type**	**Description**
cost	Mixed	Cost of using the charging station (e.g., "Free", "$0.25/kWh").
charging_points	Integer	Number of charging points available at the station.
pay_at_location	String	Indicates if payment is required at the location ("Yes", "No").
membership_required	String	Indicates if membership is required to use the station ("Yes", "No").
access_key_required	String	Indicates if an access key is required to use the station ("Yes", "No").
is_operational	String	Indicates if the station is operational ("Operational", "Out of Service").
latitude	Double	Latitude of the charging station’s location.
longitude	Double	Longitude of the charging station’s location.
operator	String	Name of the company or entity operating the charging station.
connection_type	String	Type of connection available at the station (e.g., "Type 2", "CHAdeMO").
power_output	Double	Maximum power output of the charging station in kilowatts (kW).
current_type	String	Type of current used (e.g., "AC", "DC").
charging_points_flag	Integer	Flag indicating specific conditions or features of charging points (e.g., 0 = standard).



```
_id: ObjectId('66d7e0a1cdf87e8b5d63d809')
cost : "Unknown"
charging_points : 1
pay_at_location : "Unknown"
membership_required : "Unknown"
access_key_required : "Unknown"
is_operational : "Unknown"
latitude : -31.9362523
longitude : 115.8715309
operator : "Unknown"
connection_type : "Unknown"
current_type : "Unknown"
charging_points_flag : 1
```

Usage:

This collection can be queried to find charging stations based on location, availability, cost, and other attributes. It is primarily used by the EV station finder feature in the app.

3.2 Collection: ev_vehicles

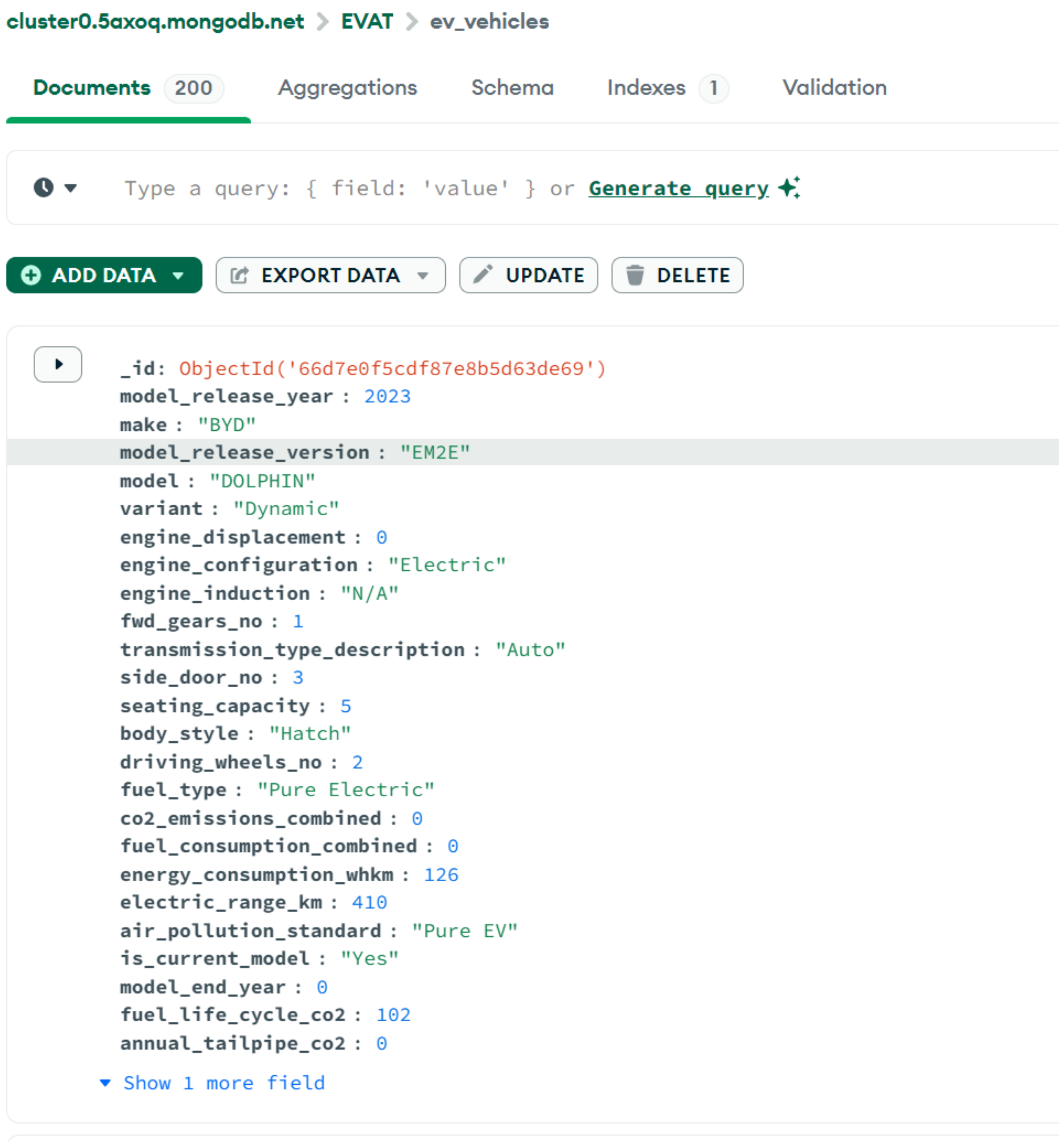
This collection stores data about different electric vehicles, including their specifications and performance metrics.

Fields:

Field Name	Data Type	Description
model_release_year	Integer	Year the vehicle model was released.
make	String	Manufacturer or brand of the vehicle (e.g., "Tesla").
model_release_version	String	Release version of the vehicle model.
model	String	Model name of the vehicle (e.g., "Model 3").
variant	String	Variant or trim level of the vehicle (e.g., "Standard Range Plus").

Field Name	Data Type	Description
engine_displacement	String	Engine displacement information (not typically applicable to electric vehicles).
engine_configuration	String	Engine configuration (e.g., "Single Motor").
engine_induction	String	Engine induction type (not typically applicable to electric vehicles).
fwd_gears_no	Integer	Number of forward gears (e.g., 1 for most electric vehicles).
transmission_type_description	String	Description of the transmission type (e.g., "Automatic").
side_door_no	Integer	Number of side doors on the vehicle.
seating_capacity	Integer	Number of seats in the vehicle.
body_style	String	Body style of the vehicle (e.g., "Sedan", "SUV").
driving_wheels_no	Integer	Number of driving wheels (typically 4 for most vehicles).
fuel_type	String	Type of fuel used (e.g., "Electric").
co2_emissions_combined	Double	Combined CO2 emissions in grams per kilometer (usually 0 for electric vehicles).
co2_emissions_urban	Double	CO2 emissions in urban conditions in grams per kilometer.
co2_emissions_extra_urban	Double	CO2 emissions in extra-urban conditions in grams per kilometer.
fuel_consumption_combined	Double	Combined fuel consumption in liters per 100 kilometers.
fuel_consumption_urban	Double	Fuel consumption in urban conditions in liters per 100 kilometers.
fuel_consumption_extra_urban	Double	Fuel consumption in extra-urban conditions in liters per 100 kilometers.
energy_consumption_whkm	Double	Energy consumption in watt-hours per kilometer.
electric_range_km	Double	Electric range in kilometers on a full charge.
air_pollution_standard	String	Emission standard met by the vehicle.
stationary_noise_data	Double	Noise level when stationary in decibels.
test_speed	Double	Test speed for emissions and consumption data in kilometers per hour.
is_current_model	Boolean	Indicates if this is the current model (true or false).
model_end_year	Integer	The last year this model was produced.
fuel_life_cycle_co2	Double	Life cycle CO2 emissions for the fuel type in grams per kilometer.
annual_tailpipe_co2	Double	Annual tailpipe CO2 emissions in grams.

Field Name	Data Type	Description
annual_fuel_cost	Double	Estimated annual fuel cost in local currency.


cluster0.5axoq.mongodb.net > EVAT > ev_vehicles
Documents 200
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE

```
_id: ObjectId('66d7e0f5cdf87e8b5d63de69')
model_release_year : 2023
make : "BYD"
model_release_version : "EM2E"
model : "DOLPHIN"
variant : "Dynamic"
engine_displacement : 0
engine_configuration : "Electric"
engine_induction : "N/A"
fwd_gears_no : 1
transmission_type_description : "Auto"
side_door_no : 3
seating_capacity : 5
body_style : "Hatch"
driving_wheels_no : 2
fuel_type : "Pure Electric"
co2_emissions_combined : 0
fuel_consumption_combined : 0
energy_consumption_whkm : 126
electric_range_km : 410
air_pollution_standard : "Pure EV"
is_current_model : "Yes"
model_end_year : 0
fuel_life_cycle_co2 : 102
annual_tailpipe_co2 : 0
▾ Show 1 more field
```

Usage:

This collection is used to retrieve vehicle information for users when they select their vehicle or when the app provides recommendations based on vehicle specifications.

3.3 Collection: users

This collection stores user profiles and preferences.

Fields:

Field Name	Data Type	Description
username	String	Unique identifier for the user.
password	String	Hashed password for secure authentication.
make	String	Make of the user’s vehicle (e.g., "Tesla").
model	String	Model of the user’s vehicle (e.g., "Model 3").
variant	String	Variant of the user’s vehicle (e.g., "Standard Range Plus").
connection_type	String	Type of socket used by the user’s vehicle (e.g., "Type 2").

Usage:

The `users` collection is used to store persistent user data, such as login credentials and vehicle details. It also temporarily stores dynamic data like current range, battery percentage, location, and destination for real-time functionalities such as trip planning and EV station finding.

4. How to Use the Database

4.1 Accessing the Database

To access the `EVAT` database, team members need to have the appropriate permissions and use the MongoDB connection string provided by the database administrator.

Example Connection String:

```
mongodb+srv://<username>:<password>@cluster0.mongodb.net/EVAT?retryWrites=true&w=majority
```

username: EVAT
password: EVAT123

4.2 Common Queries and Operations

1. **Find Charging Stations Near a Location:**

```
db.charging_stations.find({
  "latitude": { $gte: minLat, $lte: maxLat },
  "longitude": { $gte: minLng, $lte: maxLng },
  "is_operational": "Operational"
});
```

2. **Get Vehicle Details for a Specific Model:**

```
db.ev_vehicles.find({ "make": "Tesla", "model": "Model 3" });
```

3. **Authenticate a User:**

```
db.users.find({ "username": "test_user", "password": "hashed_password" });
```

5. Integration with the App:

- **Backend Setup:**
  - Ensure the backend server (API) has the necessary environment variables for MongoDB connection strings and credentials.
  - Use appropriate MongoDB drivers for your backend language (e.g., `pymongo` for Python, `mongodb` for Node.js).
- **User Authentication:**
  - Implement secure user authentication using hashed passwords (e.g., bcrypt).
  - Ensure secure storage and handling of sensitive information.
- **Data Access:**
  - The app team will use API endpoints to interact with MongoDB. Make sure the API provides necessary endpoints for CRUD operations on each collection (`charging_stations`, `ev_vehicles`, `users`).

6. Conclusion

This document provides a comprehensive overview of the MongoDB database architecture for the EVAT project. The database is designed to handle a variety of data types, ranging from static vehicle information to dynamic real-time user data. Proper management and usage of this database will ensure robust and efficient functionality of the EVAT app.