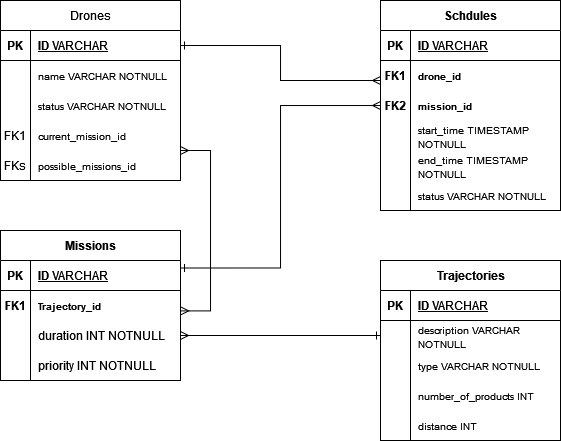
**WDS Home Assignment**

**Database URL:**  
mongodb+srv://root:secret123@homeassignment.miynwen.mongodb.net/?retryWrites=true&w=majority

**Assumptions:**

* There is an Id generated by mongo, and a custom id for each object in the database
* I created the database structure by analyzing and matching the format of the CSV files.
* End time of a schedule is the start time + the duration of the mission

**Database Design:**

****

**Explanation:**

**Drones Table:**

The **Id** column is the primary key.

**current\_mission\_id** is a foreign key referencing the Missions table.

**possible\_missions\_ids** is an array of VARCHARS representing possible mission a drone can handle.

**Trajectories Table:**

The **Id** column is the primary key.

**description** and **type** are VARCHAR fields describing the trajectory.

**number\_of\_products** is an integer representing the number of products to handle during the trajectory.

**Missions Table:**

The **Id** column is the primary key.

**trajectory\_id** is a foreign key referencing the Trajectories table.

**Schedules Table:**

The **Id** column is the primary key.

**drone\_id** is a foreign key referencing the Drones table.

**mission\_id** is a foreign key referencing the Missions table.

**start\_time** and **end\_time** are timestamps representing the scheduled time range for a mission.

**status** is a VARCHAR field indicating the status of the scheduled mission.

**Indexes:**

**Drones Table:**

**Index on the id field**: This index will speed up lookups based on the unique identifier for each drone.

Index on the status field: This index can be beneficial for filtering drones based on their availability status.

**Trajectories Table:**

Index on the id field: Like the Drones table, an index on the unique identifier will optimize lookup operations.

**Missions Table:**

Index on the id field: For efficient lookups based on the unique identifier of each mission.

**Index on the trajectory\_id** field: If there are queries that involve filtering based on the trajectory, an index on this field can be beneficial.

**Schedules Table:**

Index on the id field: For quick lookups based on the unique identifier of each schedule.

**Compound index on (drone\_id, start\_time, end\_time)**: This index can significantly speed up queries involving scheduling logic, especially for preventing double booking and checking drone availability during a specific time frame.

Note: Due to the constraints I had by the free tier of MongoDB, which limits the number of atlas searches that can be created to three, I implemented only the most important searches: Compound index on (drone\_id, start\_time, end\_time), trajectory\_id in "missions" collection and Id in "drones" collection.

**AWS Integration:**

**Ip:** http://13.60.42.212(/docs for openApi)

EC2 allows to easily scale computing resources up or down based on demand. I can dynamically add or remove EC2 instances to handle increased or decreased workloads. This scalability ensures that my application can handle varying levels of traffic without experiencing performance degradation or downtime.  
EC2 also provides Auto Scaling capabilities, allowing me to automatically adjust the number of EC2 instances in response to changes in demand. This ensures that the application always has the appropriate number of resources available to handle incoming requests, even during peak traffic periods.   
In the field of availability, by deploying my application across multiple EC2 instances in different availability zones, you can achieve high availability. If one EC2 instance experiences issues, traffic can be routed to healthy instances, minimizing downtime, and ensuring continuous operation.

**Handling Edge Cases**

There were a couple of edge cases I noticed while developing this api service:

Input Validations

* Duplicate ids are not allowed and handled by the service.
* When adding possible mission that a drone can take, the service checks if there is such a kind of mission.
* When scheduling a mission, all inputs are checked:
  + Id validations
  + Missions input validations
  + Drone availability.
* For missions' validation, I check for valid numbers for the duration and priority parameters.
* For trajectories, I check for a valid duration and number of products.

Double Booking

The scheduling service checks for any conflicts before confirming a new schedule. If a drone is already booked for the specified time, the request is rejected. In addition, drones can't start a mission with the same description at the same time.

Unavailable Drones

Before scheduling, drones are checked for their availability and operational status. If a drone is unavailable, the system will not allow it to be scheduled for a mission. In addition, drones can be scheduled only for missions that are in their list of possible missions.

Completed Mission

When mission is completed, drones are marked as available, their current mission is set to None, and the mission is scheduled as "completed".