

Python/Django Developer Test

Objective

Create a RESTful API using Django REST Framework for managing ride information.

Core Requirements

1. Utilize Django REST Framework
 - Create necessary models for `Ride`, `User`, and `RideEvent`.
 - Implement serializers for the models to enable JSON serialization/deserialization.
 - Use Viewsets for managing CRUD operations.
2. Authentication
 - Implement authentication to ensure only users with the role 'admin' can access the API.
3. Ride List API
 - Implement an API which returns a list of Rides
 - Each Ride should include related RideEvents and Users (`id_rider`, `id_driver`)
 - Support pagination
 - Support filtering Ride status, Rider email
 - Support for sorting by `pickup_time` and distance to pickup given a GPS position as input in the same API. Must implement both sorting options as efficiently as possible with the assumption that the ride table will be very large. The sorting implementation should still support pagination.
4. Performance
 - Since we anticipate that the `RideEvent` table will be very large, the Ride List API should return an additional field on `Ride` named '`todays_ride_events`' which only retrieves `RideEvents` that occurred in the last 24 hours. For performance reasons, we should never retrieve the full list of `RideEvents` in the SQL queries that Django generates. You should also use advanced Django features to retrieve this data in a way that minimizes the number of SQL queries generated.
 - The number of database queries for the Ride List API should be minimized. Retrieving the ride list with the related driver, rider, and `RideEvents` can be achieved with 2 queries (3 if you include the query required to get the total count used in Pagination).
5. Table Definitions:

Ride Table

Field	Data Type	Description
<hr/>		

id_ride	INT	Primary key
status	VARCHAR	Ride status (e.g., 'en-route', 'pickup', 'dropoff')
id_rider	INT	Foreign key referencing User(id_user)
id_driver	INT	Foreign key referencing User(id_user)
pickup_latitude	FLOAT	Latitude of pickup location
pickup_longitude	FLOAT	Longitude of pickup location
dropoff_latitude	FLOAT	Latitude of dropoff location
dropoff_longitude	FLOAT	Longitude of dropoff location
pickup_time	DATETIME	Pickup time

User Table

Field	Data Type	Description
id_user	INT	Primary key
role	VARCHAR	User role ('admin' or other roles)
first_name	VARCHAR	User's first name
last_name	VARCHAR	User's last name
email	VARCHAR	User's email address
phone_number	VARCHAR	User's phone number

Ride_Event Table

Field	Data Type	Description
id_ride_event	INT	Primary key

id_ride	INT	Foreign key referencing Ride(id_ride)
description	VARCHAR	Description of the ride event
created_at	DATETIME	Timestamp of when the event occurred

Submission

- Django project source code hosted on a version control repository (e.g., GitHub).
- Ensure that your commit history is clean and commits are meaningful, showing the progression of the project.
- Make sure that the README is comprehensive and helps any developer set up the project without any hassles and also include any additional notes or comments regarding design decisions or challenges faced during implementation.

Evaluation Criteria

- Functionality: Does the application perform all the defined requirements?
- Code Quality: Is the code modular, readable, and maintainable?
- Error Handling: Check how errors and edge cases are handled within the application.
- Performance: Is the API optimized to use the minimal number of SQL queries sent to the database?

Bonus - SQL

- For reporting purposes, we need a raw SQL statement (include this in the README), which returns the count of Trips that took more than 1 hour from Pickup to Dropoff and we want this by Month and Driver.
- Below is a sample output of the report:

Sample Report

Month	Driver	Count of Trips > 1 hr
2024-01	Chris H	4
2024-01	Howard Y	5

2024-01	Randy W	2
2024-02	Chris H	7
2024-02	Howard Y	5
2024-03	Chris H	2
2024-03	Howard Y	2
2024-03	Randy W	11
2024-04	Howard Y	7
2024-04	Randy W	3

- When a driver picks up a rider a Ride_Event will be created with the description 'Status changed to pickup' and similarly when a rider is dropped off a Ride_Event with the description 'Status changed to dropoff'. So to calculate the duration of the Trip, you will need to find the pickup and dropoff RideEvents for each ride and calculate the time difference. (Note: You do not need to implement the code that populates the Ride_Event table, and you can assume it's already populated to create the SQL)
- Ideally, in the real world, we would probably store the distance on the Ride table itself, but for the purposes of this assessment, let's assume we cannot change the Ride table structure.