Car-Pooling-System (Backend API)

Brief Description :

The Car-Pooling-System is a backend API solution designed to facilitate ride-sharing or lift-sharing among users, specifically drivers and passengers.

The system aims to promote environmentally friendly travel by enabling users to share their journeys, consequently reducing carbon emissions and traffic congestion.

Users can find matches for their rides through various mediums and communicate with each other to arrange trip details.

The Car-Pooling-System is a **functions based** backend APIs that uses **JWT authentication** and supports role-based activities (**Admin, Driver, Passenger, Visitors**).

Backend Tools & Frameworks: **Python 3.12.3, Django 4.2.23, Django REST Framework 3.16.6, MySQL Workbench**. Postman is used for API testing.

This document provides detailed information about all API endpoints in the Carpool Management System, including authentication, admin controls, carpool creation, booking.

Flow Diagram of Carpool System:

```
[ Start ]
  |
  v
[ User Registration / Login ]
  - Register as Passenger or Driver
  - Login with JWT authentication
  - Forgot password & Reset password flow
  |
  v
[ Dashboard ]
```

```
|
+--> [ Passenger Actions ]
|      |
|      +--> Search Carpools
|      |      - Filter by Start/End Location, Date, Price
|      |      - Sort by Price, Departure Time, Available Seats
|      |
|      +--> View Carpool Details
|      |      - See driver info, available seats, price
|      |
|      +--> Book Carpool
|      |      - Choose seats, confirm booking
|      |
|      +--> View My Bookings
|      |      - Update / Cancel bookings
|      |      - Filter by status, date, driver
|      |
|      +--> Receive Notifications
|             - Ride reminders 40 minutes before departure
|
+--> [ Driver Actions ]
|      |
|      +--> Create Carpool
|      |      - Add route, seats, departure time, price
|      |
|      +--> Update / Delete Carpool
|      |
|      +--> View My Carpools
|      |      - See booked passengers
|      |
|      +--> Approve / Reject Bookings
|      |      - Confirm passengers for upcoming rides
|      |
|      +--> Notifications
|             - Receive booking updates
|
+--> [ Admin Actions ]
       |
       +--> View All Users
       |      - Activate / Deactivate accounts
       |
       +--> View All Activities
       |      - Track user activity logs
       |
       +--> Manage Carpools & Bookings
       |      - View all carpools, bookings
       |
       +--> Generate Reports
       |      - Revenue reports, busiest routes, cancellations
       |
       +--> Export Data
              - Excel / CSV / JSON of activities and bookings
```

Passenger Workflow (Detailed):

```
[ Login/Register ]
  |
  v
[ Dashboard ]
  |
  +--> Search Carpools
  |      |
  |      +--> View Available Carpools
  |      |
  |      +--> Apply Filters (Location, Date, Price)
  |
  +--> Book Carpool
         |
         +--> Seats Confirmation
         +--> Booking Confirmation
         |
         +--> Receive Email Notification
```

Driver Workflow (Detailed):

```
[ Login/Register ]
  |
  v
[ Dashboard ]
  |
  +--> Create Carpool
  |       - Add Car Details, Departure Time, Seats, Price
  |
  +--> View My Carpools
  |       - See passenger bookings
  |
  +--> Approve / Reject Bookings
  |       - Update booking status
  |
  +--> Update / Delete Carpools
  |
  +--> Receive Booking Notifications
```

Admin Workflow (Detailed):

```
[ Login ]
  |
  v
[ Admin Dashboard ]
  |
  +--> Manage Users
  |       - View all users
  |       - Activate / Deactivate accounts
  |
  +--> Manage Carpools & Bookings
  |       - View all carpools
  |       - View all bookings
  |
  +--> Reports
  |       - Revenue by route
  |       - Weekly / Monthly bookings
  |       - Cancellation stats
  |
  +--> Export Activity Data
          - Excel / CSV / JSON
```

Summary of Entities and Actions:

```
[ Users ]
  - Passenger
  - Driver
  - Admin
  - Actions: Register, Login, Update Profile, Delete Profile

[ Carpools ]
  - Created by Drivers
  - Viewed by Passengers
  - Actions: Create, Update, Delete, View, Search, Sort

[ Bookings ]
  - Made by Passengers
  - Approved/Rejected by Drivers
  - Actions: Create, Update, Cancel, Filter

[ Notifications ]
  - Sent to Passengers and Drivers
  - Actions: Ride reminders, Booking updates

[ Admin Reports ]
  - View All Users
  - View All Activities
  - View All Carpools and Bookings
  - Manage Search data
  - Generate Reports
```

Functions List

Total Functions: 31 divided into Authentication, User Management, Carpool Management, Booking Management, Admin Reporting, Notifications.

Each function is **explained like a docstring**, with **endpoint, method, authentication, description, and request body example**.

Custom Authentication & Permissions:

Class: CustomJWTAuthentication

Functionality: Validates JWT token and retrieves User.

Raises ***AuthenticationFailed*** for invalid or expired tokens.

Permission Classes:

- **IsAuthenticatedCustom** → Any authenticated user
-
- **IsAdminCustom** → Admin only
-
- **IsDriverCustom** → Driver only
-
- **IsPassengerCustom** → Passenger only
-
- **IsAdminOrDriverCustom** → Admin or Driver
-
- **IsAdminOrPassengerCustom** → Admin or Passenger
-
- **IsDriverOrPassengerCustom** → Driver or Passenger

★ Authentication & User Management Functions:

1. register_user()

""""
Register a new user with username, email, and password.
This will create a passenger account in the system.
""""

Endpoint: POST /register/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "username": "vills",
  "email": "vills@example.com",
  "password": "StrongPass@123",
  "first_name": "Vills",
  "last_name": "Patel",
  "profile_image": "profile.jpg"
}
```

2. login_user()

""""
Login a registered user using username and password.
Generates JWT token for session management.

"""

Endpoint: POST /login/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "username": "vills",
  "password": "StrongPass@123"
}
```

3. logout_user()

"""
Logout the currently logged-in user by invalidating the JWT token.
"""

Endpoint: POST /logout_user/

Authentication: Required

Request Body: None

4. view_profile()

"""
View the profile details of the logged-in user.
Returns first name, last name, email, username, profile image.
"""

Endpoint: GET /profile/my-profile/

Authentication: ✅ Required

Request Body: None

5. update_profile()

"""
Update the current user's profile information such as username, email, password, name, or profile image.
"""

Endpoint: PUT /profile/update-profile/

Authentication: ✅ Required

Request Body Example:

```
{
  "username": "newvills",
  "email": "newvills@example.com",
  "password": "Updated@123",
  "first_name": "Vills",
  "last_name": "Updated",
  "profile_image": "new_profile.jpg"
}
```

6. delete_profile()

"""
Delete the currently logged-in user's account permanently.
"""

Endpoint: DELETE /profile/delete-profile/

Authentication: ✅ Required

Request Body Example:

```
{
  "user_id": 12
}
```

7. forgot_password()

"""
Initiate password reset by sending OTP to registered email.
"""

Endpoint: POST /profile/forgot-password/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "email": "vills@example.com"
}
```

8. reset_password()

"""

Reset user password using OTP received via email.
""""

Endpoint: POST /profile/reset-password/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "email": "vills@example.com",
  "password": "NewPass@123",
  "OTP": "458921"
}
```

9. contact_us()

""""

Send a message to the support team from logged-in or guest user.
"""”

Endpoint: POST /profile/contact-us/

Authentication: ❌ Optional

Request Body Example:

```
{
  "email": "vills@example.com",
  "message": "I need help with my booking."
}
```

★ Carpool Management Functions

10. create_carpool()

""""

Create a new carpool as a driver.
Provide departure time, seats, price, and route details.
""""

Endpoint: POST /carpool/create/

Authentication: ✅ Required

Request Body Example:

```
{
  "car_name": "Toyota Prius",
  "departure_time": "2025-09-25T08:30:00",
  "available_seats": 3,
  "price_per_seat": 200,
  "start_location": "Mumbai",
  "end_location": "Pune"
}
```

11. update_carpool()

""""
Update existing carpool information.
Only the carpool driver can update the carpool.
""""

Endpoint: PUT /carpool/update/

Authentication: ✅ Required

Request Body Example:

```
{
  "carpool_id": 5,
  "available_seats": 2,
  "price_per_seat": 250
}
```

12. delete_carpool()

""""
Delete a carpool created by the driver.
""""

Endpoint: DELETE /carpool/delete/

Authentication: ✅ Required

Request Body Example:

```
{
  "carpool_id": 5
}
```

13. view_my_carpools()

""""

View all carpools created by the logged-in driver.

""""

Endpoint: GET /carpool/my-carpools/

Authentication: ✅ Required

Request Body: None

14. view_booked_passenger()

""""

View passengers who booked a particular carpool.

""""

Endpoint: GET /carpool/view-passengers/

Authentication: ✅ Required (Driver only)

Request Body Example:

```
{
  "carpool_id": 5
}
```

15. carpool_detail()

""""

Public endpoint to view details of upcoming carpools.
Only shows carpools with available seats.

""""

Endpoint: GET /carpool/detail/

Authentication: ❌ Not Required

Request Body: None

16. search_carpools()

""""

Search carpools by start location, end location, or date.

""""

Endpoint: POST /carpool/search-carpools/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "start_location": "Mumbai",
  "end_location": "Pune",
  "date": "2025-09-25"
}
```

17. sort_carpools_by()

""""
Sort available carpools by departure time, price, or available seats.
""""

Endpoint: POST /carpool/sort-carpools/

Authentication: ❌ Not Required

Request Body Example:

```
{
  "sort_by": "price",
  "order": "asc"
}
```

★ Booking Management Functions

18. book_carpool()

""""
Book a seat in an upcoming carpool.
Only registered users can book.
""""

Endpoint: POST /booking/create/

Authentication: ✅ Required

Request Body Example:

```
{
  "carpool_id": 5,
  "seats_booked": 1
}
```

19. my_bookings_info()

""""

View all bookings made by the logged-in user.

""""

Endpoint: GET /booking/my-bookings/

Authentication: ✅ Required

Request Body: None

20. update_my_booking()

""""

Update number of seats in an existing booking.

""""

Endpoint: PUT /booking/update/

Authentication: ✅ Required

Request Body Example:

```
{
  "booking_id": 10,
  "seats_booked": 2
}
```

21. cancel_booking()

""""

Cancel an existing booking.

""""

Endpoint: DELETE /booking/delete/

Authentication: ✅ Required

Request Body Example:

```
{
```

"booking_id": 10
}

22. filter_bookings()

"""
Filter user bookings by date, status, or driver.
"""

Endpoint: POST /booking/filter/

Authentication: ✅ Required

Request Body Example:

```
{
  "status": "confirmed",
  "date": "2025-09-25"
}
```

23. driver_view_booking_requests()

"""
Driver can view all pending booking requests for their carpools.
"""

Endpoint: GET /driver/booking-requests/

Authentication: ✅ Required (Driver only)

Request Body: None

24. driver_approve_reject_booking()

"""
Drivers can approve or reject a booking request.
"""

Endpoint: POST /driver/booking-action/

Authentication: ✅ Required (Driver only)

Request Body Example:

```
{
  "booking_id": 10,
```

"action": "approve"
}

★ Notification Function

25. ride_reminder_notifications()

"""
Send email reminders to passengers 40 minutes before the ride.
Returns "No upcoming rides" if none.
"""

Endpoint: GET /ride_reminder_notifications/

Authentication: ✅ Required (Driver only)

Request Body: None

★ Admin Functions

26. admin_view_users()

"""
List all registered users in the system.
Filters by username/email are optional.
"""

Endpoint: POST /admin/view-users/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "username": "john",      # string (optional)
  "email": "john@example.com"  # string (optional)
}
```

27. admin_active_deactive_user()

"""
Activate or deactivate a user account.
"""

Endpoint: POST /admin/user-action/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "user_id": 12,          # int
  "action": "deactivate"   # string ("activate" or "deactivate")
}
```

28. view_all_activities()

""""
Retrieve activity logs of all users in the system.
Filters optional: by user or action type.
""""

Endpoint: POST /admin/view-activities/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "user_id": null,        # int (optional)
  "action_type": "booking" # string (optional)
}
```

29. admin_view_carpools()

""""
View all carpools including upcoming and past journeys.
Filters optional: by departure/arrival location or date.
""""

Endpoint: POST /admin/carpools/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "departure_location": "Ahmedabad", # string (optional)
  "arrival_location": "Mumbai",      # string (optional)
  "date": "2025-09-25"               # string (optional)
}
```

30. admin_view_bookings()

""""
View all carpool bookings: pending, confirmed, or rejected.
Filters optional: by carpool_id, passenger_name, or status.
""""

Endpoint: POST /admin/carpool-bookings/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "carpool_id": 1,          # int (optional)
  "passenger_name": "John",  # string (optional)
  "status": "confirmed"      # string (optional: "pending", "rejected", "confirmed")
}
```

31. admin_full_report()

"""
Generate comprehensive system reports.
Includes users, carpools, bookings, cancellations, and revenue.
"""

Endpoint: POST /admin/reports/
Authentication: ✅ Required (Admin/Superuser)
Request Body:

```
{
  "report_type": "monthly",      # string ("daily", "weekly", "monthly")
  "start_date": "2025-09-01",  # string (optional)
  "end_date": "2025-09-25"    # string (optional)
}
```

★ Utilities & Helpers:

1. **activity(user, details)** – Save activity log for user.
2. **user_is_admin(user)** – Check if a user is an admin or has superuser privileges.
3. **get_tokens_for_user(user)** – Returns JWT refresh & access token with user_id, username, role.
4. **Generate_otp** - generate OTP for password between 100000, 999999 integer
5. **send_otp_email(email, otp):** Send an email with an OTP to the given email address.
6. **km_inr_format(data):** Add INR and KM units to contribution_per_km and distance_km.
7. **send_booking_email(booking, status_type):** send Email for booking confirmed / rejected / waitlisted / cancelled
8. **ride_status_function(request):** This function is used to update the ride status of all bookings based on the current time.