Description: Implement a ride-sharing application with the below-expected features.

Features:

The application allows users to share rides on a route.

Users can either offer a shared ride (Driver) or consume a shared ride (Passenger).

Users can search and select one from multiple available rides on a route with the same source and destination.

Requirements:

Application should allow user onboarding.

add\_user(user\_detail)

Add basic user details

add\_vehicle(vehicle\_detail)

Add the user’s vehicle(s) details

User should be able to offer a shared ride on a route with details.

offer\_ride(ride\_detail)

Ride will have details like vehicle, origin, destination, available seats. (A ride will have no intermediate stops.)

Users can select a ride from multiple offered rides using a selection strategy. (A user can only request a ride (only for 1 or 2 people))

select\_ride(source, destination, seats, selection\_strategy)

Preferred Vehicle (Activa/Polo/XUV)

Most Vacant.

System should be able to end the ride. User can only offer a ride for a given vehicle, once there are no active offered rides for that vehicle.

end\_ride(ride\_details)

Find total rides offered/taken by all users.

print\_ride\_stats()

Bonus Question:

If the user’s origin/destinations are not available directly but it’s possible via multiple rides, then the application should output multiple rides. (Example: for input: Bangalore to Mumbai, the output can be Bangalore to Goa and Goa to Mumbai)

Other Notes:

Write a driver class for demo purposes. Which will execute all the commands in one place in the code and test cases.

Do not use any database or NoSQL store, use in-memory data-structure for now.

Do not create any UI for the application.

Please prioritize code compilation, execution, and completion.

Work on the expected output first and then add good-to-have features of your own.

Expectations:

Make sure that you have a working and demonstrable code.

Make sure that the code is functionally correct.

Use of proper abstraction, modeling, separation of concerns is required.

Code should be modular, readable and unit-testable.

Code should easily accommodate new requirements with minimal changes.

Proper exception handling is required.

Sample Test Cases:

Onboard 5 users

add\_user(“Rohan, M, 36”); add\_vehicle(“Rohan, Swift, KA-01-12345)

add\_user(“Shashank, M, 29”); add\_vehicle(“Shashank, Baleno, TS-05-62395)

add\_user(“Nandini, F, 29)

add\_user(“Shipra, F, 27”) ; add\_vehicle(“Shipra”, Polo, KA-05-41491); add\_vehicle(“Shipra, Activa KA-12-12332”)

add\_user(“Gaurav, M, 29)

add\_user(“Rahul, M, 35); add\_vehicle(“Rahul”, “XUV”, KA-05-1234);

Offer 4 rides by 3 users

offer\_ride(“Rohan, Origin=Hyderabad, Available Seats=1, Vehicle=Swift, KA-01-12345, Destination= Bangalore”)

offer\_ride(“Shipra, Origin=Bangalore, Available Seats=1, Vehicle=Activa KA-12-12332, Destination=Mysore”)

offer\_ride(“Shipra, Origin=Bangalore, Available Seats=2, Vehicle=Polo, KA-05-41491, Destination=Mysore”)

offer\_ride(“Shashank, Origin=Hyderabad, Available Seats=2, Vehicle=Baleno, TS-05-62395, Destination=Bangalore”)

offer\_ride(“Rahul, Origin=Hyderabad, Available Seats=5, Vehicle=XUV, KA-05-1234, Destination=Bangalore”)

offer\_ride(“Rohan, Origin=Bangalore, Available Seats=1, Vehicle=Swift, KA-01-12345, Destination=Pune”)

This call should fail, since a ride has already been offered by this user for this vehicle.

Find rides for 4 users

select\_ride(“Nandini, Origin=Bangalore, Destination=Mysore, Seats=1, Most Vacant”)

2(c) is the desired output.

select\_ride(“Gaurav, Origin=Bangalore, Destination=Mysore, Seats=1, Preferred Vehicle=Activa”)

2(b) is the desired output

select\_ride(“Shashank, Origin=Mumbai, Destination=Bangalore, Seats=1, Most Vacant”)

No rides found

select\_ride(“Rohan, Origin=Hyderabad, Destination=Bangalore, Seats=1, Preferred Vehicle=Baleno”)

2(d) is the desired output

select\_ride(“Shashank, Origin=Hyderabad, Destination=Bangalore, Seats=1,Preferred Vehicle=Polo”)

No rides found

End Rides

end\_ride(2-a)

end\_ride(2-b)

end\_ride(2-c)

end\_ride(2-d)

Find total rides by user: Rides Taken: Rides that have were taken and have been marked as “ended”

Rides Offered: Rides that were offered and have been marked as “ended”.

Note: Even if the offered ride was not taken by any user, it counts as an offered ride.

print\_ride\_stats()

Nandini: 1 Taken, 0 Offered

Rohan: 1 Taken, 1 Offered

Shashank: 0 Taken, 1 Offered

Gaurav: 1 Taken, 0 Offered

Rahul: 0 Taken, 0 Offered

Shipra: 0 Taken, 2 Offered