Genetic Algorithm

- **calculate_distance(latitude1, longitude1, latitude2, longitude2):** This function calculates the distance between two points on the Earth's surface using the Haversine formula. It assumes the Earth as a perfect sphere and converts the distance to kilometers.

- **time_difference(pickup_datetime1, pickup_datetime2):** This function calculates the time difference in minutes between two pickup datetimes.

- **fitness(driver, passenger):** This function calculates the fitness score for a driver based on the passenger's parameters. It considers factors such as distance, time difference, and seat availability. The fitness score is higher for drivers that better match the passenger's requirements.

- **create_individual(drivers):** This function creates a random individual (driver) from the available driver dataset.

- **create_population(drivers, population_size):** This function creates a population of individuals (drivers) by repeatedly calling the create_individual function.

- **mutate(individual, min_seats, max_seats):** This function performs mutation on an individual (driver) by randomly changing the number of available seats within a given range.

- **crossover(parent1, parent2):** This function performs crossover between two parents (drivers) by randomly selecting attributes from each parent to create a child.

- **selection(population, passenger):** This function performs selection, specifically tournament selection, to choose parents for the next generation. It compares the fitness of several drivers and selects the best ones as winners.

- **perform_assignment(passengers, drivers, population_size, mutation_rate, max_generations, filename):** This function is the main part of the algorithm. It takes in the passenger and driver datasets, as well as other parameters, and returns the assignments of drivers to passengers. It iterates through the passengers and runs a number of generations to find the best-matched driver for each passenger. The

algorithm creates an initial population of drivers, performs selection, crossover, and mutation to evolve the population. The best-matched driver is selected based on the fitness score. The assignments are stored in the assignments list.

- **Select_of_client_interface(client_id, formatted_date):** This function serves as the client interface. It takes a client ID and a formatted date as input. It prepares the passenger and driver datasets based on the provided data. It then calls the perform_assignment function to perform the driver-passenger assignment. If a suitable driver is found for a passenger, the assignment details are printed and further actions are taken.

- **GALog(APIView):** This is a view class that handles the API request. In the get method, it retrieves the current date, formats it, and retrieves the client data from the database. It then calls the Select_of_client_interface function to perform the driver-passenger assignment for each client.

By running the code, the best-matched drivers will be assigned to passengers based on their parameters, considering factors such as distance, time difference, and seat availability. The genetic algorithm optimizes the assignment process by evolving a population of drivers over multiple generations.