Plus Points in Implementation (Overall Evaluation Criteria)

1. Authentication:
- Implement robust user authentication protocols to ensure secure access.

2. **Cost Estimation - Time and Space**:
- Conduct a thorough analysis of time and space complexity in the system.
- Utilize efficient algorithms and data structures to optimize both time and space requirements.

3. **Handling System Failure Cases**:
- Implement fault-tolerant mechanisms to address system failures.
- Employ backup and recovery strategies for data integrity.
- Develop comprehensive error recovery procedures to minimize downtime.

4. **Object-Oriented Programming Language (OOPS):**
- Choose a robust OOPS language for structured and modular code.
- Leverage OOPS principles such as encapsulation, inheritance, and polymorphism for maintainability and extensibility.

5. **Trade-offs in the System:**
- Clearly define and document trade-offs made during system design.
- Evaluate and communicate the rationale behind architectural and design decisions.
- Consider trade-offs in terms of performance, scalability, and maintainability.

6. **System Monitoring:**
- Implement comprehensive monitoring tools to track system performance.
- Utilize real-time dashboards and logging mechanisms to promptly identify and address issues.

7. **Caching**:
- Integrate caching mechanisms to enhance system response times.
- Utilize caching for frequently accessed data to reduce database load.
- Implement cache eviction policies for optimal resource utilization.

8. **Error and Exception Handling**:
- Develop a robust error and exception handling framework.
- Provide meaningful error messages for effective debugging.
- Regularly review and update error-handling strategies based on system usage patterns.

Instructions:

1. Read and Understand the Problem Statement:
- Carefully read the problem statement provided. Understand the requirements, inputs, expected outputs, and any constraints mentioned.

2. **Choose a Programming Language:**
- Select a programming language you are comfortable with and that is suitable for solving the problem described in the case study.

3. Design Your Solution:
- Plan the overall structure of your solution. Consider the algorithms, data structures, and any potential optimizations needed.

4. Write the Code:
- Implement your solution in code. Follow best practices for coding standards, such as meaningful variable names, proper indentation, and comments where necessary.
- Break down the problem into smaller functions or modules to improve code readability and maintainability.

5. Test Your Code:
- Test your code thoroughly with different sets of input data, including edge cases and boundary conditions.
- Ensure that your code produces the expected outputs for all test cases.

7. Document Your Code :
- Consider adding documentation or comments to explain the logic and purpose of your code, especially for complex parts or algorithms.

8. Submit Your Solution:
- Once you're satisfied with your code and it meets all the requirements, submit your solution on GitHub and share the GitHub link.

9. Demonstration:
- Include a demonstration video showcasing key features of the ride-sharing platform.
- Alternatively, use screenshots to visually highlight the user interface and functionality.

Carpooling System: A Smart and Privacy-Focused Ride-Sharing Solution

A Carpooling System enables users to share rides efficiently, reducing travel costs, congestion, and carbon footprints. This system connects riders and drivers securely while ensuring privacy and convenience. Below is a detailed breakdown of the key functionalities:

I. Pool Creation & Joining

Creating a Ride Pool (For Drivers)

Drivers can create a ride pool by entering the following details:
✅ Pickup & Drop Locations – Exact or approximate locations for both starting and ending points.
✅ Departure Time & Date – When the ride will begin.
✅ Available Seats – Maximum passengers allowed.
✅ Vehicle Details – Car model, license plate, and other relevant info.
✅ Preferences & Rules – Music, smoking, pet-friendly, etc.

🔹 **Process**:

1. The driver posts a new ride with all necessary details.
2. The ride becomes visible to nearby users looking for a ride.
3. Interested riders can send a request to join.
4. The driver can approve or reject requests based on their preferences.

Joining a Ride Pool (For Riders)

Passengers looking for a ride can:
✅ Search for available ride pools based on their destination and departure time.
✅ Filter rides based on preferences (e.g., female-only rides, no smoking, etc.).

✅ Request to join a ride and wait for approval from the driver.
✅ Once approved, receive ride details, including pickup location and estimated time.

🔹 **Benefits**:

- Reduces individual travel costs.
- Helps minimize traffic congestion.
- Encourages community-based carpooling.

II. Intelligent Ride Matching

To ensure efficiency, the system automatically matches riders with available carpools based on:

✅ Proximity – Users are matched with ride pools starting near their location.
✅ Route Similarity – The system finds rides that follow a similar path to the rider's destination.
✅ Timing & Availability – Matches only those rides that fit the user's schedule.
✅ Preferences & Restrictions – Takes into account user-specific preferences like gender, smoking, pets, and other ride policies.

🔹 **Process**:

1. The system fetches all available rides.
2. It compares the ride's start, destination, and route with the rider's request.
3. Rides are ranked based on best match scores and displayed to the user.
4. The rider selects a suitable ride and sends a join request.

🔹 **Key Advantages:**

- Optimized Ride Sharing: Minimal detours for drivers while maximizing seat occupancy.
- Efficient Filtering: Riders see only relevant options.
- Flexible Matching: Users can choose based on convenience and preferences.

III. Route Matching Percentage

To help users make better decisions, the system calculates the Route Match Percentage, indicating how closely a ride matches their travel path.

✅ **How it Works:**

- The system analyzes the route planned by the driver.
- It then compares this route with the rider's preferred route.
- A percentage score (e.g., 85%) is displayed to show the match level.
- Higher percentage rides are prioritized in search results.

🔹 **Example**:

- 100% Match: If a driver's route perfectly aligns with the rider's start and destination points.
- 75% Match: If the ride covers most of the rider's route but may require a short detour.
- 50% Match: If the ride covers half of the route but requires additional transportation.

🔹 **Benefits**:

- Helps users quickly identify the best-matching ride.
- Reduces unnecessary diversions and travel delays.
- Improves ride efficiency by optimizing driver-passenger compatibility.

IV. Privacy Protection

Since ride-sharing involves interacting with strangers, privacy is a top priority. The system ensures user safety by masking personal information.

✅ **Phone Number Privacy:**

- Users can call or message each other via an in-app VoIP system, keeping phone numbers hidden.
- Calls are routed through masked numbers (e.g., a temporary number).
- The system automatically disconnects after the ride is completed to prevent misuse.

✅ **Profile Security:**

- Users can choose to hide their full name, showing only their first name or initials.
- Profile pictures can be blurred until a ride is confirmed.
- Only necessary details (such as car model and departure time) are visible to matched riders.

✅ **SOS & Emergency Features:**

- Live Location Sharing – Users can share their ride status with family members.
- Panic Button – In case of emergencies, users can alert local authorities.

🔹 **Advantages**:

- Ensures anonymity and protects users from unsolicited contact.
- Prevents data misuse and unwanted follow-ups.
- Increases trust and security among participants.