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 for Project submissions:

Document Format:
- Combine textual explanations, screenshots, and code snippets for clarity.
- Organize information in a structured manner, following a logical flow.

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

Case study: Intelligent Floor Plan Management System for a Seamless Workspace Experience

Admin's Floor Plan Management (Onboarding/Modifying the floor plans):

Objective: Develop features for administrators to upload floor plans, addressing potential conflicts during simultaneous updates.

Tasks:

- Develop a conflict resolution mechanism for simultaneous seat/room information uploads.
- Implement a version control system to track changes and merge floor plans seamlessly.
- Resolve conflicts intelligently, considering factors such as priority, timestamp, or user roles.

Offline Mechanism for Admins (For the UI person):

Objective: Implement an offline mechanism for admins to update the floor plan in scenarios of internet connectivity loss or server downtime.

Tasks:

- Develop a local storage mechanism for admins to make changes offline.
- Implement synchronization to update the server when the internet or server connection is re-established.
- Ensure data integrity and consistency during offline and online transitions.

Meeting Room Optimization (Meeting Room Suggestions and Booking):

Objective: Enhance the system to optimize meeting room bookings, suggesting the best meeting room based on capacity and availability.

Tasks:

- Develop a meeting room booking system considering the number of participants and other requirements.
- Implement a recommendation system to suggest meeting rooms based on capacity and proximity.
- Ensure dynamic updates to meeting room suggestions as bookings occur and capacities change.
- Show the preferred meeting room based on the last booking weightage.