**Security:**

1.The risk of compromised passwords.

Solution: Password is hashed by using the provided User model in Django.

2.User try to create duplicated username.

Solution: Specific check procedure was implemented in the phase of registration in case of duplicated user name.

3. Unlogged users try to visit data.

Solution: Login check was implemented to ensure that only logged in users can visit the data. (like searching rides, edit rides, etc.)

4.Race condition for rides. For example, two drivers try to confirm the same ride.

Solution: When processing the confirm request from the driver, the status of the ride was checked again to ensure the ride is open. However, race condition is still existed. Transactions and lock could be implemented in the future to solve this.

5.Handle invalid user input. Sometimes user may enter invalid/empty inputs. This brings uncertainties to the server.

Solution: Firstly, empty input is not acceptable in the phase of HTML page. When user try to submit a from with empty fields, the from can not be successfully submitted, and warning message will be prompted. Secondly, invalid inputs (expected numbers, but characters) are also controlled on the HTML page or checked when user want to submit the form.

**Resilience:**

1. Server may fail due to unexpected errors.

Solution: No preparations for this situation at this moment. Server should be restarted by hands.

2. Database may fail.

Solution. No solution at this moment.

**Scalability:**

1. Add new features into the server.

Solution: MTV design model is used when design the server. The software has low coupling. When adding new features into the server, few modifications are required for other parts.