**Danger Log for hw1**

1. **Security**
2. Users cannot read others’ information.

**Solution:** We should access data through sessions that could be isolated from each other, instead of directly searching by users’ id.

1. We cannot expose our source code in the production environment, which may leak the absolute path of the web application and the path of the Python interpreter. Besides, the debug page contains an interactive shell for Python that can execute arbitrary Python code.

**Solution:** To avoid this security danger, we set the DEBUG to false to close the debug page in the production environment.

1. Sometimes we need to perform a series of operations on the database. If one of the operations fails, then the other operations must be rolled back to the previous state of the operation. For example, when a user register to be a driver, but his/her vehicle plate info doesn’t save to the database successfully, then we need to roll back to the status where the user hasn’t registered to be a driver.

**Solution:** Convert data operations of functions in the views into transactions.

1. Potential CSRF attack: it is an implicit authentication mechanism originating from the web. Although the authentication mechanism of the Web can guarantee that a request comes from a user's browser, it cannot guarantee that the request is approved by the user.

**Solution:** Anti CSRF token. When Django responds to a request from a client for the first time, it will randomly generate a token on the server-side and put this token in a cookie. Then every POST request will bring this token, so as to avoid being attacked by CSRF.

1. **Create a user account:**
   1. Users may input an invalid format of email address, and we need to verify it.
   2. An email address cannot be registered again.
2. **Register as a driver:**
   1. A used vehicle plate cannot be registered again.
   2. The number of passengers cannot be smaller than 1.
3. **Request a ride:**
   1. The arrival time must be later than the present time.