University of British Columbia, Department of Computer Science

CPSC 304

Project Formal Specification

Date: 2018-09-28

Group Member:

NAME:	STUDENT ID:	CS ID:	TUTORIAL SECTION:	EMAIL:
KYO TANG	35163104	u0q0b	T1D	kyo.hideki.tang@gmail.com
CHRISTPHER YAO	20924163	d1e1b	T1D	yaowongzhou@gmail.com
RUI ZHANG	28834166	n3v0b	T1D	zhangrui_ca@sina.com
YANYUN BU	42828146	b6h1b	T1C	iamclaudebu@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Platform

For our project, we decide to use CS UGrad Oracle installation and provided PHP as our development platform.

Functionality

There are two classes of users in our application: Developer and Test Driver. The mainly difference between these two class is that Test Driver can modify the setting and data in our application on the testing car and Develop can modify the setting and data and extract the record on the computer terminal.

The functionalities for our users will be followed by one or more query item:

1. **Add Account:**
   User: Developer
   Query: INSERT

2. **Remove Account:**
   User: Developer
   Query: DELETE

3. **Set Account as Test Driver:**
   User: Developer
   Query: INSERT

4. **Add Car with Device:**
   User: Developer
   Query: INSERT

5. **Remove Car with Device:**
   User: Developer
   Query: DELETE

6. **Change Car's Device:**
   User: Developer
   Query: UPDATE

7. **Add Self-Driving Software:**
   User: Developer
   Query: INSERT

8. **Install Self-Driving Software for a Car:**

User: Developer
Query: INSERT

9. **Add Path**
   User: Developer and Test Driver
   Query: INSERT

10. **Add Path Condition:**
    User: Developer and Test Driver
    Query: INSERT

11. **Create a new test:**
    User: Developer and Test Driver
    Query: INSERT

12. **Start a test:**
    User: Developer and Test Driver
    Query: UPDATE

13. **Edit a test:**
    User: Developer and Test Driver
    Query: UPDATE

14. **Find a list of test with a specific Path Condition:**
    User: Developer
    Query: SELECT, JOIN, CREATE VIEW

15. **Find a list of test with a specific self-driving software:**
    User: Developer
    Query: SELECT, JOIN, CREATE VIEW

16. **Find a list of test with a specific self-driving Device:**
    User: Developer
    Query: SELECT, JOIN, CREATE VIEW

17. **Find the number of test for a self-driving software**
    User: Developer
    Query: GROUP BY

18. **Find a list of self-driving software that has minimal test:**
    User: Developer
    Query: GROUP BY

Data

As the schemas of functionality show above, our application will contain Account Information, Test Driver Information, Self-Driving Device, Car, Self-Driving Software, Self-Driving Test, Path Condition, Path, and List of Software in Car as our data. On the other hand, the Geometry list for Path and the self-driving software console log will be automatically generated from our application and stored in our database.

Because the high professionality of our database, it is different to collect the real data in the remain time of this term. Thus, our group will fabricate the sample data reasonably for our application.

1. Developer (accountNo:integer, password:string, name:string)
2. TestDriver (accountNo:integer, driverid:string, phoneNo:integer)
3. Car (carid:integer, cartype:string, deviceid:integer)
4. SelfDrivingDevice (deviceid:integer, deviceVersion:string, carid:integer)
5. SelfDrivingSoftware (versionid:integer, updatetime:string, comment:string)
6. ListSoftwareInCar (carid:integer, versionid:integer)
7. SelfDrivingSoftwareRecord (swrecordid:integer, consolelog:string, versionid:integer)
8. SelfDrivingTest (recordid:integer, status:string, carid:integer, versionid:integer, swrecordid:integer, pathid:integer, driverid:integer, fromdatetime:string, todatetime:string)
9. Path (pathid:integer, city:string, location:string, startpoint:string, endpoint: string, pathcondid:integer)
10. PathCondition (pathcondid:integer, roadtype:string, weather:string, climate:string, dayornight:string)
11. Geometry (geoid:integer, lat:integer, lon:integer, pathid:integer)

Devision of Labour

Our Group decides to split our work into x parts and each member in our group may need to take one or more task:

- The design of application (contains the creating of table and database object): Yanyun Bu
- The creating of data and the test of Data (contains the test for each set of queries after programming): Christpher Yao
- The backend programming (contains the coding of the queries and the embedding SQL statement): Kyo Tang, Rui Zhang
- The frontend programming (contains the implement of user interface and the formatting of output): Yanyun Bu, Christpher Yao

- Communicating and scheduling: Yanyun Bu
- Project Document: Yanyun Bu

Our group expects that group member only work on their responsible task. For the task for two member, members will need to collaborate with each others. The devision of Labour may change in the future.