CPS 630 REPORT

Plan for Smart Services P2S Web application

Team 6

Nguyen Hong Duc - 500910998

Sourena Khanzade - 500929191

Farhan Tahmeed Sami - 500982031

Task	Assigned to
Front-end design: Logo design, layout design, context, images, icons, drag-drop design	Sourena (50%), Duc (50%)
Back-end design: Model design, Controller design	Duc (40%), Farhan (30%), Sourena (30%)
Map design	Sourena (50%), Duc (50%)
Report	Duc (34%), Farhan (33%), Sourena (33%)
Database design	Duc (50%), Farhan (50%)
Architecture design	Sourena (33%), Duc (33%), Farhan (33%)
Diagram design	Duc (33%), Farhan (34%), Sourena (33%)
Quality control	Farhan (33%), Duc (34%), Sourena (33%)
Database’s data control	Duc (25%), Farhan (50%), Sourena (25%)

OVERVIEW

Over the past few years, the living conditions of everyone have been affected heavily by the carbon emissions caused by industrial waste or transportation. Several solutions have been developed in order to tackle this problem, and they all show promising results.

The project applies the “Plan for Smart Services” (P2S) Web application methodology as a means of creating potential services to help reduce air pollution. The web application is developed under the LAMP stack (Linux, Apache, MySQL, PHP) with MVC architecture. Additionally, the web app includes open libraries such as Google Maps, SASS, JQuery to assist the system.

TECHNOLOGIES

As mentioned above, the web application will be developed under the LAMP stack with the MVC design pattern. To be more specific, these are the tools that will be used:

1. Linux OS system (As developing environment)
2. Apache (XAMPP - Web hosting tool)
3. MySQL (MySQL database)
4. PHP (As the main coding language for back-end)
5. HTML5, SASS, CSS3, JavaScript, JQuery (For UI/UX)

IMPLEMENTATION

1/ The main system

The web application will have 2 different modes: user mode and admin mode. Admin will have all the privileges to view the databases, as well as create, delete and modify records. Users will be able to see their personal information, orders, map, store, products.

To distinguish between the 2 modes, the database was set to collect user type. The “user”

type is set by default so that everyone with access to the website can make their personal account. The “admin” type, however, can only be set up by the developer/root admin.

To differentiate between the user’s type, we use the SESSION from PHP to record the ‘username’. If the username is admin, we will allow them to have access to “DB maintain” on the top navbar.

The following section will include all the implementation of SQL codes for all the transactions. (These are very generic code snippets, however, when we implement with PHP, the format are almost identical)

1. **Creating**

```
CREATE TABLE PRODUCT_REVIEW (
    REVIEW_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    R_CONTEXT VARCHAR(200),
    R_SCORE INT,
    FLOWER_ID INT,
    FOREIGN KEY (FLOWER_ID) REFERENCES FLOWER (FLOWER_ID)
);

CREATE TABLE DRIVER_REVIEW (
    REVIEW_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    R_CONTEXT VARCHAR(200),
    R_SCORE INT,
    CAR_ID INT,
    FOREIGN KEY (CAR_ID) REFERENCES CAR (CAR_ID)
);
```

```sql
CREATE TABLE CUSTOMER (
    CUSTOMER_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    CUSTOMER_NAME VARCHAR(50),
    CUSTOMER_TEL VARCHAR(20),
    CUSTOMER_EMAIL VARCHAR(50),
    CUSTOMER_ADDRESS VARCHAR(100),
    CUSTOMER_CITY_CODE VARCHAR(20),
    CUSTOMER_USERNAME VARCHAR(100),
    CUSTOMER_PASSWORD VARCHAR(100),
    CUSTOMER_BALANCE FLOAT,
    CUSTOMER_ADMIN BOOLEAN
);

CREATE TABLE CAR (
    CAR_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    CAR_MODEL VARCHAR(50),
    CAR_CODE VARCHAR(50),
    AVAILABILITY_CODE VARCHAR(50)
);

CREATE TABLE FLOWER(
    FLOWER_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    STORE_CODE VARCHAR(50),
    PRICE FLOAT
);

CREATE TABLE TRIP (
    TRIP_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    DESTINATION_CODE VARCHAR(100),
    SOURCE_CODE VARCHAR(100),
    DISTANCE FLOAT,
    CAR_ID INT,
    PRICE FLOAT,
    FOREIGN KEY (CAR_ID) REFERENCES CAR (CAR_ID)
);

CREATE TABLE CUSTOMER_ORDER (
    ORDER_ID INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    DATE_ISSUED TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    DATE_DONE DATE NOT NULL,
    TOTAL_PRICE FLOAT,
    PAYMENT_CODE VARCHAR(100),
    CUSTOMER_ID INT,
    TRIP_ID INT,
    FLOWER_ID INT,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID),
    FOREIGN KEY (TRIP_ID) REFERENCES TRIP (TRIP_ID),
    FOREIGN KEY (FLOWER_ID) REFERENCES FLOWER (FLOWER_ID)
);
```

2. Adding

```
insert into CUSTOMER (CUSTOMER_NAME, CUSTOMER_TEL, CUSTOMER_EMAIL, CUSTOMER_ADDRESS,
CUSTOMER_CITY_CODE, CUSTOMER_USERNAME, CUSTOMER_PASSWORD, CUSTOMER_BALANCE, CUSTOMER_ADMIN)
values ('Archaimbaud Fortune', '3814553378', 'afortune0@tiny.cc', '9 Sunbrook Crossing',
'7940-374', 'afortune0', MD5('m614oAR4'), 489.74, false);
```

```
insert into FLOWER (STORE_CODE, PRICE) values ('aaaaa-aaa', 0);
insert into FLOWER (STORE_CODE, PRICE) values ('52755-200', 53.21);
insert into FLOWER (STORE_CODE, PRICE) values ('59115-046', 61.51);
insert into FLOWER (STORE_CODE, PRICE) values ('51514-0251', 58.14);
insert into FLOWER (STORE_CODE, PRICE) values ('64942-1127', 49.13);
insert into FLOWER (STORE_CODE, PRICE) values ('0378-4162', 95.29);
insert into FLOWER (STORE_CODE, PRICE) values ('43547-274', 12.73);
insert into FLOWER (STORE_CODE, PRICE) values ('68828-079', 20.41);
```

```
insert into TRIP (DESTINATION_CODE, SOURCE_CODE, DISTANCE, CAR_ID, PRICE)
values ('82 Merrick Center', '15 Dakota Circle', 17, 13, 32.46);
```

```
insert into CUSTOMER_ORDER (DATE_DONE, TOTAL_PRICE, PAYMENT_CODE, CUSTOMER_ID, TRIP_ID, FLOWER_ID)
values ('2020-09-05', 86.34, '50385067829721017', 1, 24, 19);
```

```
insert into PRODUCT_REVIEW (R_CONTEXT, R_SCORE, FLOWER_ID)
values ('Cloned reciprocal solution', 4.1, 21);
```

```
insert into DRIVER_REVIEW (R_CONTEXT, R_SCORE, CAR_ID)
values ('Versatile well-modulated forecast', 2.4, 11);
```

3. **Deleting**

```
public function deleteCar($id) {
    $sql = "DELETE FROM CAR WHERE CAR_ID = $id;";
    $query = mysqli_query($this->conn, $sql) or die (mysqli_error($this->conn));
    return $query;
}
```

4. **Modifying**

```php
public function updateById($id, $model, $code, $avail) {
    $sql = "UPDATE CAR
    SET CAR_MODEL = '$model', CAR_CODE= '$code', AVAILABILITY_CODE='$avail'
    WHERE CAR_ID = '$id';";
    echo $sql;
    $query = mysqli_query($this->conn, $sql) or die (mysqli_error($this->conn));
    return $query;
}
```

5. **Searching**

```sql
SELECT * FROM CUSTOMER_ORDER WHERE ORDER_ID LIKE '%123%';
```

```php
public function search($search, $id){
  $sql = "SELECT * FROM CUSTOMER_ORDER INNER JOIN FLOWER on CUSTOMER_ORDER.FLOWER_ID = FLOWER.FLOWER_ID WHERE CUSTOMER_ID=$id AND ORDER_ID=$search;";
  $query = mysqli_query($this->conn, $sql) or die(mysqli_error($this->conn));
  return $query;
}
```

2/ The front-end

Each page will contain certain information, functionality that assists the user. All pages will navigate through a **Navbar component**. The website will contain the following pages:

- Home page
- About us
- Contact us
- Sign up
- Reviews
- Shopping cart
- Type of Service

The Home page where all the information about the web is displayed. This will give a quick overview of the page along with its features. On this page, other content will be included such as the “About Us” page - a brief bio description of Team 6 members as well as their contribution to the site, and the form “Contact us”.

The Signup page will lead the user to a form where they could fill up their information for signing up for an account. After they have successfully created an account, they can log in to the system using the Login form.

The Shopping Cart is where the user can see their current orders. This page will

summarize all the prices along with all items the customer has ordered. By default, this page will show an empty cart. However, once the user has selected items on the "**Type of service**" page, the interface will be changed accordingly to the orders.

The Type of Service will toggle users between services: Delivering service or driving service. The user can use drag and drop to add items to their cart. Once the user decides to process a payment, they will be redirected to the order page or their "Cart".

The Review page: Users can read over the reviews for a product or a driver, however, they will not be able to make any review/ rating unless they have previously ordered/got delivered from that specific driver or bought the specific flower type. They will also not be able to use the review page if they have not logged in.

3/ The back-end

There are 2 main components that made up the back-end, which are the relational database system and the MVC back-end.

For the relational database system, we have decided to create the following tables:

Table	Description	Attributes	# of Records
Car	Information about cars	ID, model, code, availability code	15
Customer	Information about customers, including user name and hashed password	ID, name, telephone, email, address, city code, username, password, balance, admin (To check for users)	10
Customer Order	Information about customer orders	ID, date issued, date done, total price, payment code, customer ID, trip ID, flower ID	100
Flower	Information about the store and price of the product	ID, Store code, price	30
Trip	Information about the trip	ID, destination code, source code, distance, car id, price	50

Product review	All the reviews on products	ID, context, score, flower ID	30
Driver review	All the reviews on driver	ID, context, score, car ID	30

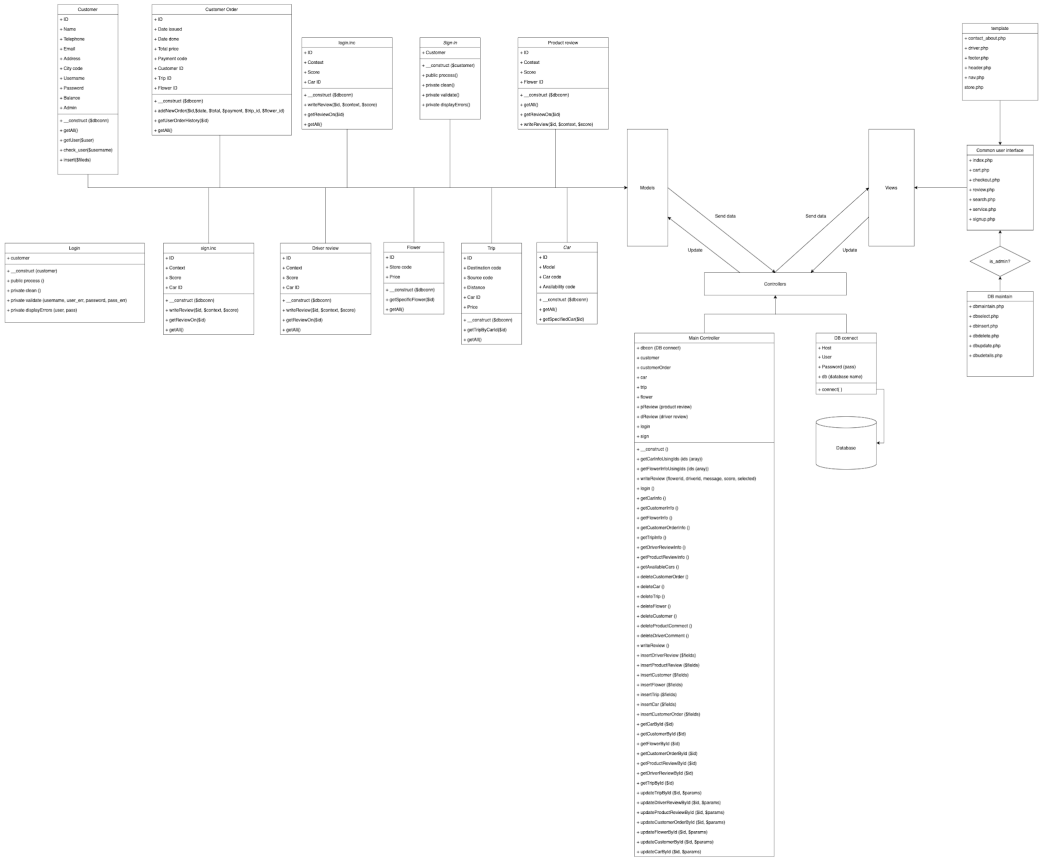
There will be separated scripts to generate the tables according to the design board above, most of the data are consistent, however, the customer order total price is inconsistent with the add up of trip price and the product type, since they were generated for demo purpose. However, once a new user is created, the new records will be much more consistent comparing to existed ones. Since all of our team members are developing the web application on the Linux platform, we have made a file named “generate.sh” in order to create and populate data to the database.

Additionally, customer passwords will be hashed once they are recorded in the database. Once the user decided to log in to the system, the database will decipher the hashed code and check for matching credentials. If the record exists, the system will populate accordingly to the type of the account (Since there will be a dashboard system for admin).

As for the back-end system, it was designed and implemented by the MVC pattern and object-oriented architecture. There will be 7 different classes with different methods to implement to the web application. These 7 classes match all the tables from the database and will be used as our **Model.** These models will be called in the main **Controller**, which will respond accordingly based on the parameters. Afterwards, the front-end (**View**) will be updated whenever it makes a request to the back-end side.

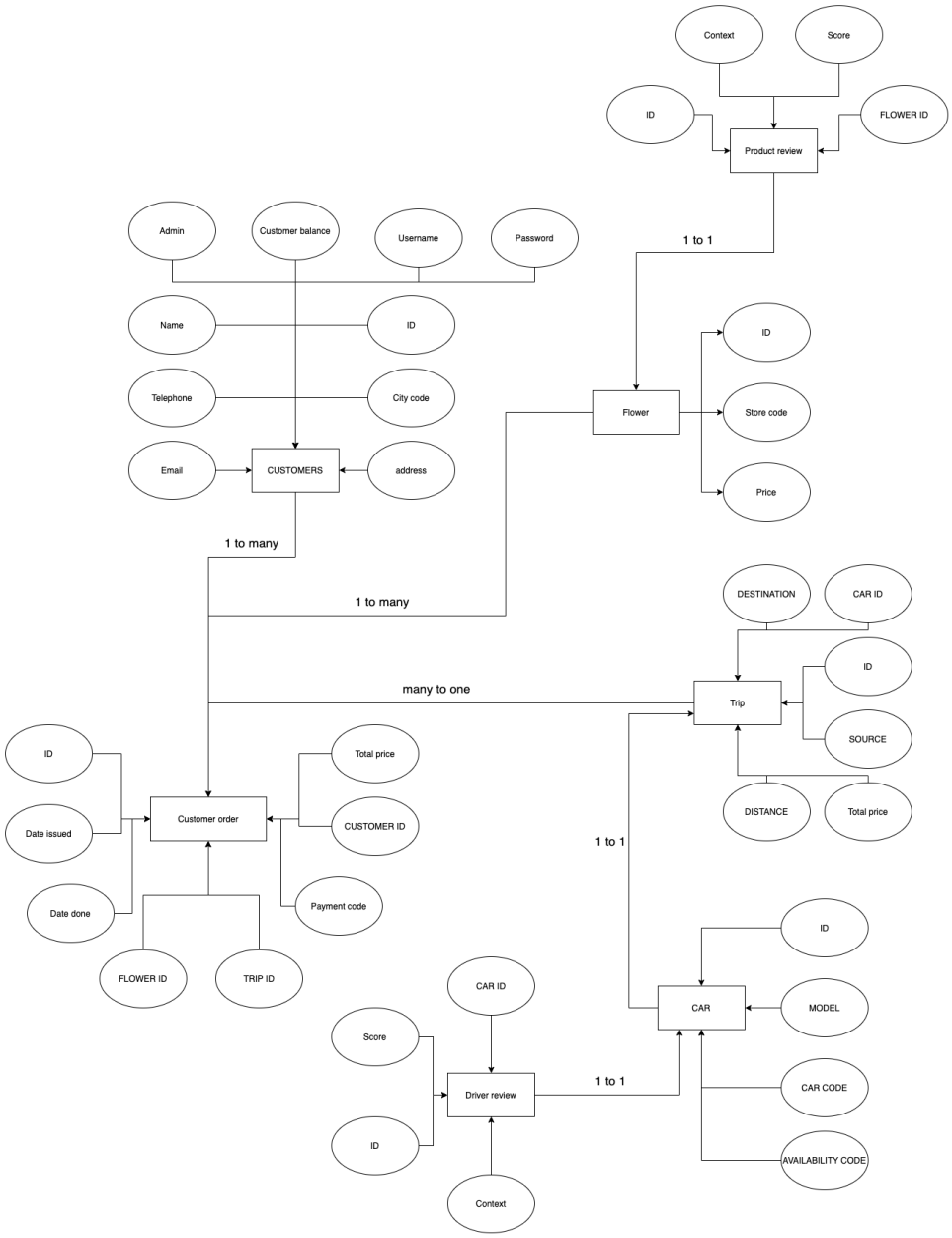
DESIGN DIAGRAM

The following diagrams are the overall design for our system



DATABASE DIAGRAM

The following diagrams are the overall database design for our system


Context
Score
ID
FLOWER ID
Product review
1 to 1
Admin
Customer balance
Username
Password
Name
ID
Telephone
City code
Email
CUSTOMERS
address
Flower
ID
Store code
Price
1 to many
1 to many
DESTINATION
CAR ID
ID
many to one
Trip
SOURCE
DISTANCE
Total price
ID
Date issued
Date done
Customer order
Total price
CUSTOMER ID
Payment code
FLOWER ID
TRIP ID
1 to 1
ID
CAR
MODEL
CAR ID
Score
ID
Driver review
1 to 1
CAR CODE
AVAILABILITY CODE
Context