CPSC 304 - Group 23
Project Formal Specification
Bike Sharing System

Platform:

We will use the CS UGrad Oracle installation and provided PHP/Apache.

Functionality:

Deliverable 1:

The "Rider" User Type

Functionality	Data Involved	Output
View locations of available bikes	Latitude/longitude of all available bikes	Ideally, a map that plots the available bikes
Start bike ride	Log starting "trip" information, such as start_date, start_time, start_latitude, start_longitude, etc.	A confirmation message showing information such as the trip start time, bike ID, etc.
End bike ride	Log the ending "trip" information, such as end_date, end_time, amount_due, and charge the rider's wallet for that amount	A confirmation message showing information such as the trip end time, the amount due, and the amount left in the rider's wallet after the trip
View the designated return areas to be able to return the bike (i.e., end the ride)	Information from the Designated_Return_Area table	Ideally, a map that plots the return areas
Report maintenance issue with bike	The date, time, and a description of the issue	A confirmation that the issue was successfully submitted
Submit a complaint	Customer description, date, time, etc.	A confirmation that the complaint was successfully submitted
View their rental history	A tuple for each of the rider's previous trips from Trip table	Start and end times/dates, locations, cost per trip, etc.
View virtual wallet	walletID, e-coins balance, credit	walletID, e-coins balance, credit

	card number and expiry	card number and expiry
Add/update credit card info in virtual wallet	Input credit card number and expiry	A confirmation that the credit card info has been updated
Load money (e-coins) into virtual wallet using credit card on file	Input an amount to be charged to the credit card and added to the wallet	A confirmation that the desired amount has been added to the wallet
Update personal information (e.g., name, address, email, phone number, etc.)	Input value(s) for the information chosen to be updated	A confirmation that the personal info has been updated
Change password	Input the new password	A confirmation that the password has been changed

<u>The "Customer Service Representative" User Type</u>

Functionality	**Data Involved**	**Output**
Access riders' personal info	Info from the Rider table	The info from the Rider table
Access riders' bike rental history	A tuple for each of the riders' previous trips from Trip table	Info from the Trip table, such as start_time, start_date, end_time, end_date, etc.
Access all bike information	Info from the Bike table	The info from the Bike table
View the designated return areas to be able to return the bike (i.e., end the ride)	Information from the Designated_Return_Area table	Ideally, a map that plots the return areas
Access bike maintenance issues (but can't resolve them)	Info from the Maintenance_Issues table	Info from the Maintenance_Issues table
View all maintenance issues submitted by particular rider	Info from Maintenance_Issues table associated with that rider	All maintenance issues associated with that rider
View submitted complaints	Customer description, date, time, etc., from the	All complaints
View all complaints submitted by particular rider	Info from Complaint table associated with that rider	All complaints associated with that rider
Reset a rider's password	Click a button to reset the password	A temporary password that should be given to the user so that they can log in

The “Maintenance Technician” User Type

Functionality	**Data Involved**	**Output**
Access all bike information	Info from the Bike table	The info from the Bike table
View bike maintenance issues	Info from the Maintenance_Issues table	Info from the Maintenance_Issues table
Resolve bike maintenance issues	Technician notes and resolved date	A confirmation that the issue has been resolved
View all replacement parts in Replacement_Part	Info from Replacement_Part table	All info from Replacement_Part table
Increase/decrease the “quantity” of replacement parts when they are acquired/used	Quantity added or removed to a particular part in the Replacement_Part table	Confirmation that the quantity has been updated by returning the row for that partNo in the Replacement_Part table
Add/remove/update replacement parts	Some or all info in Replacement_Part table	Confirmation of the add/remove/update success
Add/remove bikes	Input info about new bike if a bike is being added. If being removed, should provide the bike_ID	A confirmation that the bike has been added/removed

Deliverable 2:

INSERT a new bike into the Bike table when the company acquires a new bike.
INSERT a tuple into the Trip table when a rider begins a new trip
INSERT a complaint into the Complaint table when the rider submits the complaint
INSERT a maintenance issue into the Maintenance_Issue table when the rider submits an issue
INSERT a part into the Replacement_Part table

Deliverable 3:

DELETE a bike from the Bike table when a maintenance technician decides that the bike can no longer be in service.
DELETE a rider from the Rider table when a rider wants to close his/her account.
DELETE a part from the Replacement_Part table

Deliverable 4:

UPDATE credit card number and expiry in the Rider table when the rider wants to change cards
UPDATE the end_time, end_date, etc., in the Trip table when the rider ends their bike trip
UPDATE a complaint to be resolved when a customer service representative resolves it.
UPDATE a maintenance issue to be resolved when a maintenance technician handles the issue
UPDATE part information in the Replacement_Part table

Deliverable 5:

Looking at a resolved maintenance issue for a particular bike, in order to find the bike's current location and the phone number of the technician who resolved the issue, we need to join the Maintenance_Issue, Bike, and Maintenance_Technician tables.

Looking at a complaint, in order to find the rider's name (who submitted the complaint) and the customer service representative's name, we need to join the Complaint, Rider, and Customer_Service_Rep tables..

For a particular refund, get the name and email of the rider and the name of the customer service representative who made the refund. We need to join the Rider, Refund, and Customer_Service_Rep tables.

Deliverable 6:

To find out the phone number of the rider who submitted a particular maintenance issue, we need to join the Maintenance_Issue and Rider tables.

To find the name of a rider associated with a particular trip, we need to join the Rider and Bike tables.

Deliverable 7:

To count the number of maintenance issues associated with each rider, we GROUP BY rider_ID in the Maintenance_Issue table and then COUNT the riderID occurrences

To count the number of complaints addressed by a customer service representative, we GROUP BY Customer_Service_Rep_ID in the Complaint table and then COUNT the complaint_ID occurrences.

To count the number of maintenance issues resolved by a particular maintenance technician, we GROUP BY technician_ID in the Maintenance_Issue table and then COUNT the technician_ID occurrences.

Deliverables 8 - 10:

Select all bikes that are not currently rented out (i.e., the Trip table doesn't have an end date/time for the trip associated with that bike) and give location information

Select all rental history for a particular rider

Get all unresolved complaints

Check that bike's latitude/longitude is within a Designated_Return_Area's coordinates

Find out which Maintenance_Technician resolved a particular Maintenance_Issue

Deliverable 11:

CREATE VIEW for Rider users on the Bike table. Riders should only be able to see the bike_ID, latitiude, and longitude columns.

CREATE VIEW for Customer_Service_Rep users on the Rider table. Customer_Service_Reps should be able to see everything except the credit card number and credit card expiry.

Deliverable 12:

<u>Division of Labour</u>

Task	**Relative Size of Task**	**Assigned To**
Create tables and other database objects	Small	Kevin, Daniel
Create data for the tables	Small	Jacques, Kevin
Code each set of queries and test them in SQL	Medium	Raghav, Jacques, Daniel
Embed the SQL statements in a program and code the programming logic. Use a graphical user interface.	Largest	**All** (All of your group members must take part in this talk because embedded SQL in a host language should be practiced by everyone, and is a learning goal of this course. However, it is OK if some group members do more programming than others, providing those other group members do more of the other tasks.)
Test each set of queries	Small	Daniel, Raghav
Document the project	Medium/Large	Raghav, Jacques, Daniel

Demo the application to a TA	Small	**All**