

Driveways

Mohit Sonwane, Dat Le, Michael Moore

Team 1

Table of Contents

Section 1. Project Requirements	2
1.1. Project Description	2
1.2. Functional Requirements	3
Section 2. Project Design	6
2.1. The ERD Diagram	6
2.2. Entity sets and relationships	7
2.3. ERD Schemas	10
2.4. MySQL Workbench contents	11
Section 3. Project Implementation	25
3.1. Real-time running of functional requirements	25
3.1.1. User Login / Sign-up	25
3.1.2. Account creation	27
3.1.3. Display parking information	30
3.1.4. Add a listing, vehicle information and rent a listing	32
3.2. How to set up and run the system	34
3.2.1. (In MySQL Workbench) Create a connection named “driveway”	34
3.2.2. (In MySQL Workbench) Import the DrivewayDump.sql database from the zip file	34
3.2.3. (In Eclipse) Open project from File System → choose the folder contained in the zip file named “CS157A-team1”	34
3.2.4. (In Eclipse) Correct the username and password to your own credentials in every file in order for the SQL connection to work	35
3.2.5. (In Eclipse) Run the driveways.jsp with TomCat server	37
3.2.6. Follow described steps in Section 3.1. to use the web application	37
Section 4. Personal Experience	38
4.1. Dat Le	38
4.2. Michael Moore	38
4.3. Mohit Sonwane	39

Section 1. Project Requirements

1.1. Project Description

This project is a database application that is based around the idea of renting out user-owned driveways / parking spots. The project's main goal is to provide hosts with renters who are interested in parking at driveways on a daily basis. Hosts will have the ability to list their driveways for rent and modify their listings at any time. Renters will have the ability to search for the nearest location to park their cars. The project is trying to solve the problem of overcrowded parking lots in the Bay Area. Renters will have the opportunity to park in a neighborhood near their work-places. This could promote the use of public transport and create a more sustainable environment for working commuters. The stakeholders for this project would be the hosts and renters who are interested in either renting out their driveways or looking for a convenient parking spot with a price that is less than traditional parking lots. Through this project, we aim to provide a service where people can find a parking spot at their convenience. The application domain will be a business domain as we would generate profit through being a middle-man to help match the needs of both parties in the service.

1.2. Functional Requirements

This application provides functionality for different types of users, including hosts, renters, and admins. Hosts and renters will be the primary user. The primary user will be able to create an account and be able to list a driveway rental or rent a parking space from the rentees. The behaviors will change depending on the selection of roles when the user interacts with the application. Renters will have to input their basic information, car identification, and payment methods. The application will provide them with the list of closest parking spots when they search based on their location. The hosts will have the option to list their driveways on the application. The application will ask for the space description, address to approve a listing. They also have the authority to modify and update their listings at any time. Admins have access to remove, edit and add any listing at any time. This allows admins to maintain and test the application.

User Login / Sign-up

- The system will prompt a “guest” user to either login or signup, and will prompt the user with a form to do so.
- The first time users should sign-up inorder to access the application.
- A “Signed-in” user will be able to search and reserve listings, or list their own listing on the system.

Account creation

- There will be user accounts and admin accounts. Admin accounts will have additional features on top of the standard user account.
- There will be a form provided to the user by the system to input user details such as name, email, payment information, and car details.

Display Parking Information

- The user will be able to search an area for available parking spots (limited to the Bay Area in this project). They will be able to see price data, size of parking spots, available dates/times, and location.
- The system will display the parking spots on a map (Google Maps API), and the user will be able to click each spot to reveal a detailed view of that specific spot.

Host Parking Spot

- A user account will be able to list a parking spot that they own for “rent”.
- The user will need to provide details of the parking spot such as size of vehicle permitted, dates/times available.

Search for Parking Spot

- Users will be able to enter an address (within a valid area) and the system will display all the nearest parking spots. Also the available parking spots are up on the map GUI (See: “Display Parking Information”).

Update / Edit Listing

- Admin will be able to update and edit the current rental postings (add/remove place for rent, and modify)

User classification (host / guest)

- Non-admin users will be separated into hosts and renters based on their selection when accessing the web application.

Vehicle Identification

- Users will be able to provide car identification (ID) such as VIN, Color, Make, Model and license plate number.
- Users can add multiple vehicles into their accounts.

Parking Duration

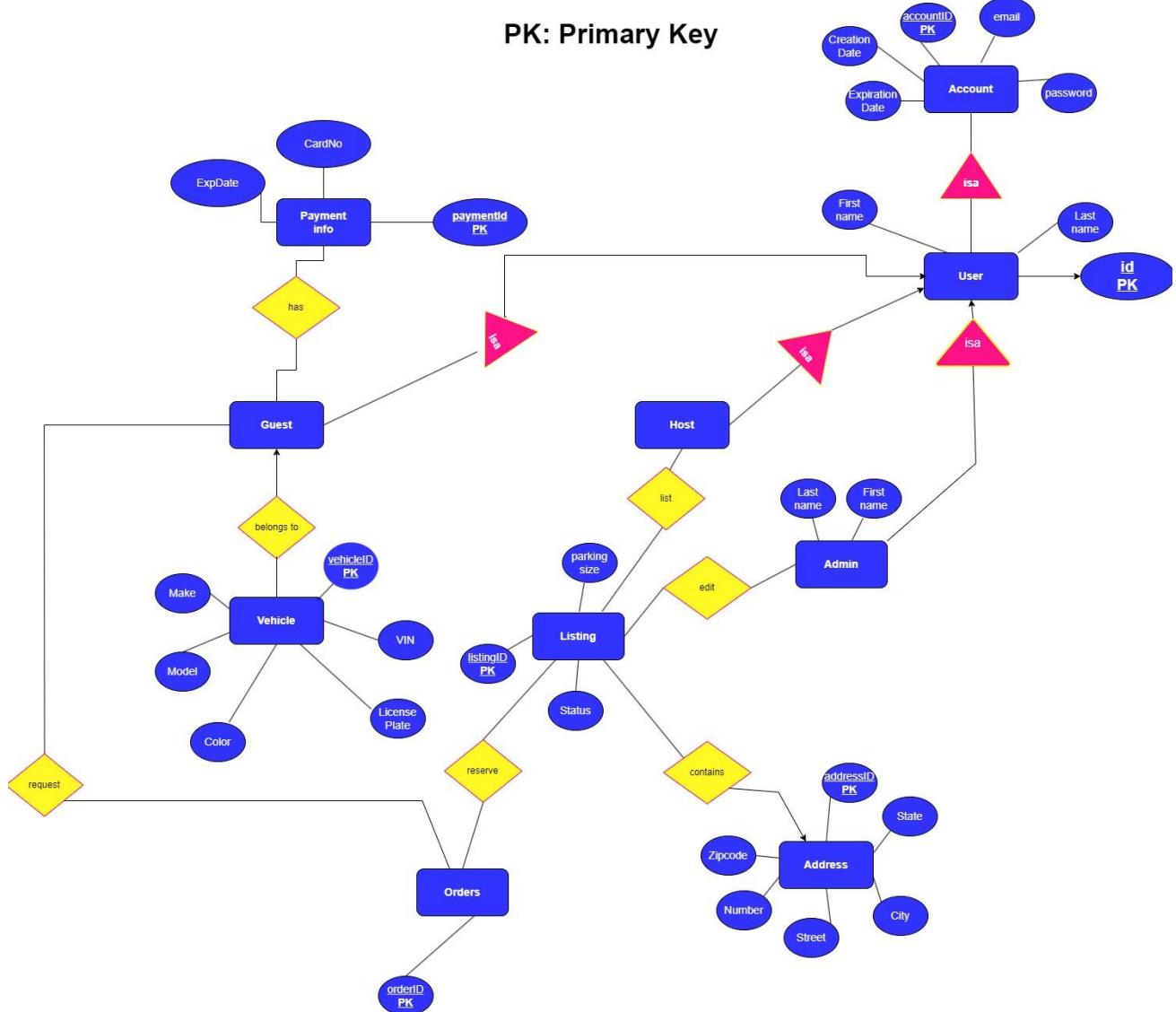
- The User's while booking a parking spot can enter the date and time from the time of drop-off to pick-up.

Payments

- The user can enter their card details in order to book the parking on our website.

Section 2. Project Design

2.1. The ERD Diagram



2.2. Entity sets and relationships

PK = Primary Key

FK = Foreign Key

ERD = Entity Relationship Diagram

1. Account Entity Set

The Account Entity set is the root entity set in our ERD. The PK is accountId.

This table is responsible for creating(Sign-Up) and managing(Login) accounts in our application.

Account Entity Relationship:

- The Account handles the User and its PK is accountId. The User is a subclass of account and inherits its PK.

2. User Entity Set

The User entity set is the parent of Guest, Host, Admin Entity Set. The PK is inherited from the Account called Id.

This table manages different types of User's in our application.

User Entity Relationship

- The User handles Guest and its PK is accountId. The Guest is a subclass of User and inherits its PK.

- The User handles Host and its PK is accountId. The Host is a subclass of User and inherits its PK.
- The User handles Admin and its PK is accountId. The Admin is a subclass of User and inherits its PK.

3. Guest Entity Set

The Guest Entity set checks the vehicle that belongs to the Guest, and requests orders for the Guest.

The PK is accountId. The FK is vehicleId.

Guest Entity Relationship

- The Vehicle has a many-to-one relationship with the Guests. Each vehicle belongs to a specific Guest. The PK of Vehicle is vehicleId.
- The Guest has many-to-many relationships with orders. A guest can make multiple orders to book a parking space. The PK of Orders is orderId. The FK is listingId & vehicleId.
- The Guest is a subclass of User.

4. Host Entity Set

The Host lists the parking spots in the listing. The PK is accountId. The FK is listingId.

Host Entity Relationship

- The Host has many-to-many relationships with Listing. The ListingId is the PK. A host can host multiple parking spaces to the Users.
- The Host is a subclass of User.

5. Admin Entity Set

The Admin edits the list. The PK is accountId.

Admin Entity Relationship

- The Admin has many-to-many relationships with Listing. There can be multiple Listings modified or accessed by more than one user.
- The Admin is a subclass of User.

6. Vehicle Entity Set

The Vehicle belongs to a specific Guest. The PK is vehicleId.

Vehicle Entity Relationship

- The vehicle has a many-to-one relationship with the Guest. Each vehicle belongs to a specific guest. The PK of Guest is accountId.

7. Listing Entity Set

Listing is the most important entity as it has relationships with many entity sets. Listing entity set manages available parking space and its fares for both the host and user. The PK is listingId. The FK is addressId.

Listing Entity Relationship

- The listing is listed by a host that has a many-to-many relationship. A host can host multiple parking spaces in listings. The PK of the host is accountId.

- The listing can be edited by Admin which has many-to-many relationships.
An admin can modify numerous listings. The PK of Admin is accountId.
- The listing contains the address of a one-to-many relationship. Thus a listing ID corresponds to at most one address. The PK of Address is addressId
- The listing reserves orders has many-to-many relationships. The PK of orders is orderId.

8. Order Entity Set

The Order Entity Set manages the parking request of the Guest User, and reserves the order of listing for Host users. The PK is orderId. The FK is listingId and vehicleId.

Order Entity Set Relationship

- The Orders requested by Guest is a many-to-many relationship. A guest can request multiple orders for parking cars. The PK of Guest is accountId.
- The order reserved by listing has a many-to-many relationship. A listing can be ordered by different users at different times. The PK of Listing is listingId.

9. Address Entity Set

The Address Entity set manages the address of the specific listing. The PK is addressId.

Address Entity Set Relationship

- The address is contained by Listing is a one-to-many relationship.
Every listing where a car will be parked will have only a single address.

10. Payment Info Entity Set

The Payment Info Details manages the payment details for guests. The PK is paymentId.

Payment Info Entity Set Relationship

- Every guest has payment information. It has many-to-many relation with guests.

2.3. ERD Schemas

Account(**accountId**, email, password, creationDate, expirationDate)

user(**id**, first_name, last_name, **accountID**)

guest(**guestId**, **vehicleId**)

PaymentInfo(cardNo, ExpDate, **paymentId**);

host(**hostId**, **listingId**)

admin(**adminId**, lastName, firstName)

Vehicle(**vehicleId**, Make, Model, Color, VIN, license_plate)

Listing(**ListingId**, parking_size, status)

order(**orderId**, **listingId**)

Address(**addressId**, Country, State, City, ZipCode, street, number)

belongTo(**vehicleId**, **id**)

edit(**id**, **listingId**)

list(**id**, **listingId**)

requests(guestId, orderId)

reserves(orderId, listingId)

contains(listingId, addressId)

has(paymentId, id)

2.4. MySQL Workbench contents

Account:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "Driveways".
- Tables:** The "Account" table is selected.
- Object Info:** Shows the table definition:

	accountid	email	password	creationDate	expirationDate
int PK	accountid	varchar(45)	password	creationDate	datetime
	email	varchar(45)			datetime
- Result Grid:** Displays 10 rows of account data:

accountid	email	password	creationDate	expirationDate
1	michael@driveways.com	5f4dcc3b5aa765d61d8327deb...	2022-07-09 17:23:12	NULL
2	mohit@driveways.com	5f4dcc3b5aa765d61d8327deb...	2022-07-09 17:23:35	NULL
3	dat@driveways.com	5f4dcc3b5aa765d61d8327deb...	2022-07-09 17:23:52	NULL
4	mike@gmail.com	5f4dcc3b5aa765d61d8327deb...	2022-07-09 17:24:10	NULL
5	Jane@hotmail.com	48cccc9ab2ad18832233e8...	2022-07-09 17:24:47	NULL
6	Ann1993@hotmail.com	21744541587a40702377ca435...	2022-07-09 17:25:18	NULL
7	baseball123@yahoo.com	d3fe7fb93d2119902ab4c55e6...	2022-07-09 17:25:37	NULL
8	1337coder@msn.com	808b57c2867ba6129c1a045ca...	2022-07-09 17:26:01	NULL
9	jimmy@gmail.com	b7a052618919e26ab74f4bc83...	2022-07-09 17:26:25	NULL
10	realperson@gmail.com	6701a210751b76568650b050b...	2022-07-09 17:26:59	NULL
NULL	NULL	NULL	NULL	NULL

User:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the schema structure under 'SCHEMAS' with 'User' selected. The central pane shows the results of the query `SELECT * FROM Driveways.User;`. The results grid contains 10 rows of data:

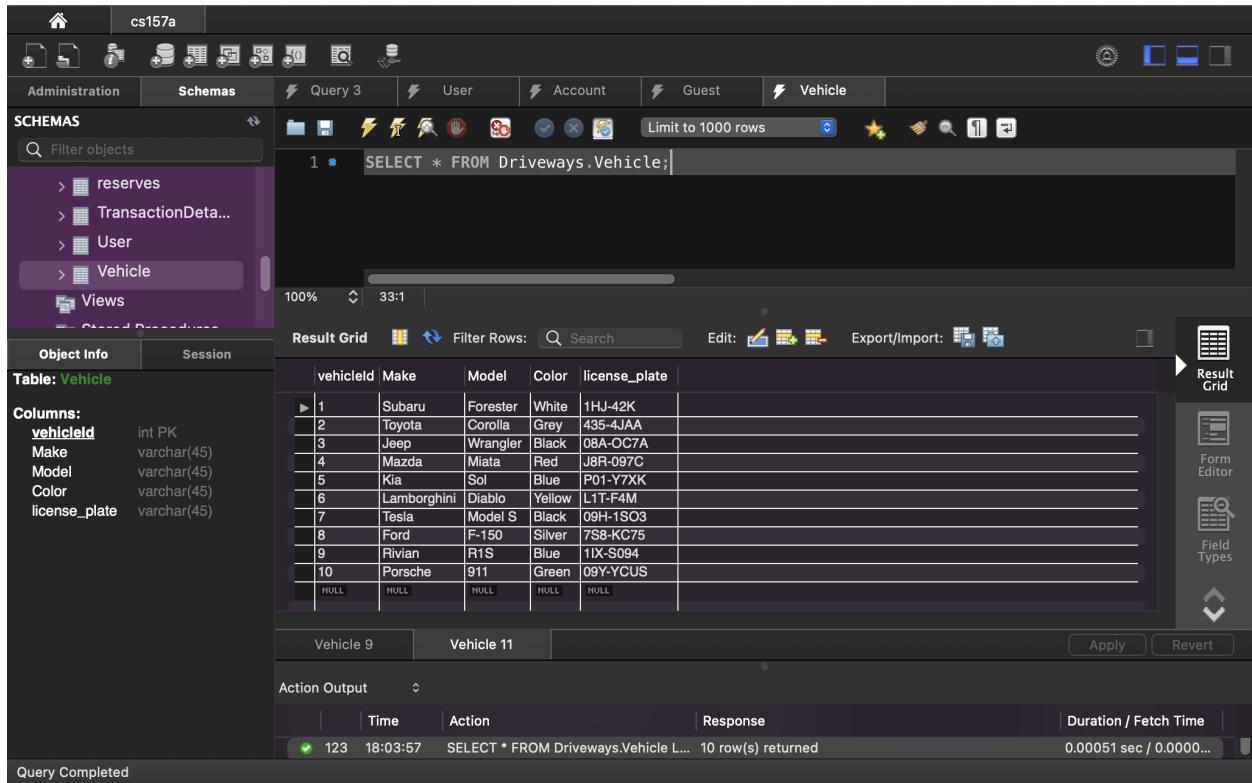
userId	first_name	last_name
1	Michael	Moore
2	Mohit	Sonwane
3	Dat	Le
4	Mike	Wu
5	Jane	Dough
6	Ann	Denton
7	John	Texas
8	Thomas	Anderson
9	Jim	McAfee
10	May	Bee

The bottom pane shows the 'Action Output' with one entry:

Time	Action	Response	Duration / Fetch Time
17:43:14	SELECT * FROM Driveways.User LIM...	10 row(s) returned	0.00048 sec / 0.000...

A message 'Query Completed' is displayed at the bottom.

Vehicle:



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The "Vehicle" schema is selected in the left sidebar.
- Query Editor:** The query `SELECT * FROM Driveways.Vehicle;` is run, and the results are displayed in the main pane.
- Result Grid:** The results are presented in a grid format with the following columns: vehicleId, Make, Model, Color, and license_plate.
- Data:** The grid contains 10 rows of data.
- Action Output:** A log entry shows the query was executed at 18:03:57 and returned 10 row(s) in 0.00051 sec / 0.000... ms.

	vehicleId	Make	Model	Color	license_plate
1	Subaru	Forester	White	1HJ-42K	
2	Toyota	Corolla	Grey	435-4JAA	
3	Jeep	Wrangler	Black	08A-OC7A	
4	Mazda	Miata	Red	J8R-097C	
5	Kia	Sol	Blue	P01-Y7XK	
6	Lamborghini	Diablo	Yellow	L1T-F4M	
7	Tesla	Model S	Black	09H-1SO3	
8	Ford	F-150	Silver	7S8-KC75	
9	Rivian	R1S	Blue	1IX-S094	
10	Porsche	911	Green	09Y-YCUS	
	NULL	NULL	NULL	NULL	

Order:

The screenshot shows a database management interface for the 'Driveways' schema. The left sidebar lists schemas: Guest, Host, Listing, Order (selected), and PaymentInformation. The main area displays the results of a SELECT query on the 'Order' table. The table has three columns: orderId, vehicleId, and listingId. The data is as follows:

orderId	vehicleId	listingId
1	2	5
2	3	6
3	1	4
4	1	7
5	8	2
6	4	1
7	9	10
8	5	3
9	6	8
10	7	9
NULL	NULL	NULL

Below the table, an 'Action Output' section shows a single row inserted into the 'Order' table at 18:44:56, with a duration of 0.0013 sec.

Time	Action	Response	Duration / Fetch Time
18:44:56	INSERT INTO Driveways.Order VALUES...	1 row(s) affected	0.0013 sec

Query Completed

Contains:

The screenshot shows the MySQL Workbench interface with the database 'cs157a' selected. In the 'Schemas' tree, the 'contains' table is highlighted. The 'Object Info' panel displays the table structure: 'Table: contains' with columns 'orderid' and 'transactionid', both defined as int PK. The 'Result Grid' shows the following data:

	orderid	transactionid
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
	NUL	NUL

The 'Action Output' pane shows the query: 'SELECT * FROM Driveways.contains;' with a duration of '0.00052 sec / 0.0000...'. The status bar at the bottom indicates 'Query Completed'.

requests:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "driveways".
- Tables:** The "requests" table is selected.
- Query Editor:** The query `SELECT * FROM Driveways.requests;` is run, and the result set is displayed.
- Result Grid:** The data is shown in a grid format:

guestId	orderId
4	1
6	2
8	3
7	4
4	5
4	6
8	7
7	8
6	9
5	10
NULL	NULL

- Action Output:** The log shows the execution details:

Time	Action	Response	Duration / Fetch Time
18:50:26	SELECT * FROM Driveways.requests...	10 row(s) returned	0.00040 sec / 0.000...
- Status:** The message "Query Completed" is displayed at the bottom.

Reserves:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "driveways".
- Tables:** The "reserves" table is selected.
- Query Editor:** The query `SELECT * FROM Driveways.reserves;` is run, and the result set is displayed.
- Result Grid:** The data is shown in a grid format:

orderId	listingId
1	8
2	4
3	9
4	1
5	2
6	3
7	9
8	10
9	5
10	7
HULL	HULL

- Action Output:** The output shows the query was executed at 18:52:16, returned 10 row(s), and took 0.00036 sec / 0.000...
- Status:** Query Completed.

Admin:

The screenshot shows a database management interface with the following details:

- Schemas:** The current schema is "Driveways".
- Tables:** The "Tables" section lists "Account", "Address", "Admin", and "contains".
- Table: Admin** (selected):
 - Columns:** "adminId" (int PK) and "Type" (varchar(45)).
 - Result Grid:** Displays the following data:

adminId	Type
1	Full
2	Full
3	Full
NULL	NULL
 - Action Output:** Shows the query "SELECT * FROM Driveways.Admin" was executed at 18:54:39, returning 3 rows in 0.00047 sec.

Guest:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the schema structure under 'Schemas' for 'Driveways'. The 'Tables' section is expanded, showing 'Account', 'Address', 'Admin', and 'contains'. The 'Admin' table is selected and highlighted in purple. The central workspace shows a query editor with the following SQL statement:

```
1 • SELECT * FROM Driveways.Guest;
```

The result grid displays the data from the 'Admin' table:

guestId	vehicleId
4	1
5	2
6	3
7	4
8	5
NULL	NULL
NULL	NULL
NULL	NULL

The bottom status bar indicates 'Query Completed'.

Host:

The screenshot shows the MySQL Workbench interface with the database 'cs157a' selected. In the Schemas tree, the 'Admin' table under the 'Driveways' schema is selected. A query window displays the result of the SQL command: 'SELECT * FROM Driveways.Host;'. The result grid shows the following data:

accountid	listingId
9	1
10	2
11	3
12	4
13	5
NULL	NULL
NULL	NULL
NULL	NULL

The Action Output panel shows the query was executed at 18:54:39 and returned 3 rows. The duration was 0.00047 sec / 0.0000... .

Has:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the schema tree under 'Schemas' with 'AvailabilityDetails', 'contains', 'Guest', 'has' (which is selected), 'Host', and 'Listing'. The central pane shows a query editor with the following content:

```
1 • SELECT * FROM Driveways.has;
```

The result grid displays the following data:

listingId	availabilityId
1	1
2	2
3	3
4	4
5	5
6	6
5	7
8	8
1	9
4	10
HULL	HULL

Below the result grid, there is a status bar with the message "has 11". The bottom pane shows the "Action Output" tab with the following details:

Time	Action	Response	Duration / Fetch Time
19:04:16	SELECT * FROM Driveways.has LIMIT...	10 row(s) returned	0.00043 sec / 0.000...

The status bar at the bottom also shows "Query Completed".

Payment Information:

The screenshot shows the MySQL Workbench interface with the database 'cs157a' selected. In the Schemas list, 'PaymentInformation' is highlighted. A query window displays the result of the SQL command: 'SELECT * FROM Driveways.PaymentInformation;'. The results grid shows 13 rows of payment information for users with IDs 4 through 16. The columns are 'userid', 'CardNumber', 'ExpDate', and 'depositInfo'. The 'depositInfo' column consistently contains the value 'Bank Account Information'. Below the grid, the status bar shows 'PaymentInformation 11' and 'Duration / Fetch Time' of '0.00053 sec / 0.0000...'. On the right side, there are various tool icons for Result Grid, Form Editor, Field Types, and Query Stats.

userid	CardNumber	ExpDate	depositInfo
4	132045024	NULL	Bank Account Information
5	327208572	NULL	Bank Account
6	439539436	NULL	Bank Account Info.
7	207261273	NULL	Bank Account Info.
8	891100594	NULL	Bank Account Info.
9	838207995	NULL	Bank Account Info.
10	528968566	NULL	Bank Account Info.
11	948707708	NULL	Bank Account Info.
12	576294234	NULL	Bank Account Info.
13	547966870	NULL	Bank Account Info.
14	NULL	NULL	NULL
15	NULL	NULL	NULL
16	NULL	NULL	NULL

Action Output

Time	Action	Response	Duration / Fetch Time
17 18:39:01	SELECT * FROM Driveways.Payment...	8 row(s) returned	0.00053 sec / 0.0000...
18 18:39:10	INSERT INTO PaymentInformation V...	1 row(s) affected	0.0020 sec

Address:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the schema structure under 'Schemas' for the 'Driveways' database. The 'Tables' section is expanded, showing 'Account', 'Address', 'contains', and 'Guest'. The 'Object Info' tab is selected for the 'Address' table, showing its columns: addressId (int PK), Country (varchar(45)), State (varchar(45)), City (varchar(45)), Zipcode (int), Street (varchar(45)), and Number (int). The main pane shows the results of the query: SELECT * FROM Driveways.Address;. The results grid contains 10 rows of address data from San Jose, California, including entries for Redwood City, Oakland, and Fremont. Below the grid, the 'Action Output' section shows the query execution details: 144 rows, 18:15:17 time, and 0.00063 sec / 0.000 duration. The status bar at the bottom indicates 'Query Completed'.

addressId	Country	State	City	Zipcode	Street	Number
1	United State of America	California	San Jose	95126	Lincoln Ave	207
2	United State of America	California	San Jose	95126	San Carlos St	968
3	United State of America	California	San Jose	95126	San Salvador St	1864
4	United State of America	California	Redwood City	95114	Fresno Ave	62
5	United State of America	California	Oakland	94501	E 17th St	226
6	United State of America	California	Oakland	94501	21st Ave	15
7	United State of America	California	San Francisco	94110	Market St	200
8	United State of America	California	San Francisco	94110	Guerrero St	622
9	United State of America	California	San Mateo	94401	Franklin St	12
10	United State of America	California	Fremont	94560	Biacow Rd	264
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Action Output

Time	Action	Response	Duration / Fetch Time
144 18:15:17	SELECT * FROM Driveways.Address...	10 row(s) returned	0.00063 sec / 0.000...

Query Completed

Listing:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Guest, Host, Listing (selected), Order, PaymentInformation, Requests.
- Query Editor:** SELECT * FROM Driveways.Listing;
- Result Grid:** Displays 10 rows of data from the Listing table.
- Table Definition:** Listing (listingId int PK, Status varchar(45), parking_size varchar(45)).
- Action Output:** Shows the execution details: 227 rows affected, 18:31:52, SELECT * FROM Driveways.Listing Li... 10 row(s) returned, Duration / Fetch Time: 0.00052 sec / 0.0000... .
- Status:** Query Completed.

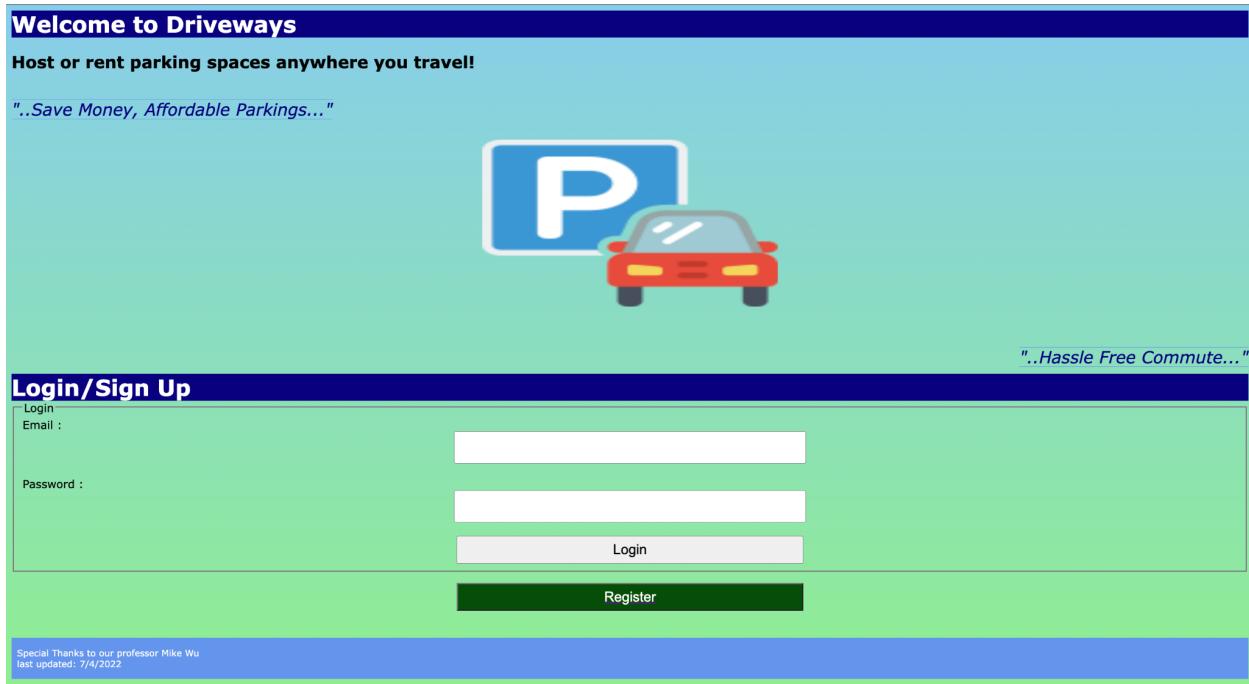
listingId	Status	parking_size
1	Available	Large
2	Occupied	Medium
3	Available	Medium
4	Available	Small
5	Available	Small
6	Inactive	Small
7	Inactive	Large
8	Available	Large
9	Occupied	Medium
10	Occupied	Small
NONE	NONE	NONE

Section 3. Project Implementation

3.1. Real-time running of functional requirements

3.1.1. User Login / Sign-up

The following picture is the main page of the web application:



The following picture shows the registration page if the user has not registered yet:

Driveways Signup Screen

Name:

Last name:

Email:

Password:

Special Thanks to our professor Mike Wu
last updated: 7/4/2022

The following snapshot shows how a user successfully logged into the dashboard:

Driveways Add Vehicle Add listing Add payment method My Account Sign out

Customer Details
Welcome, Gon
Email: gon@gmail.com

Host a Parking



Host Parking

Rent a Parking Spot

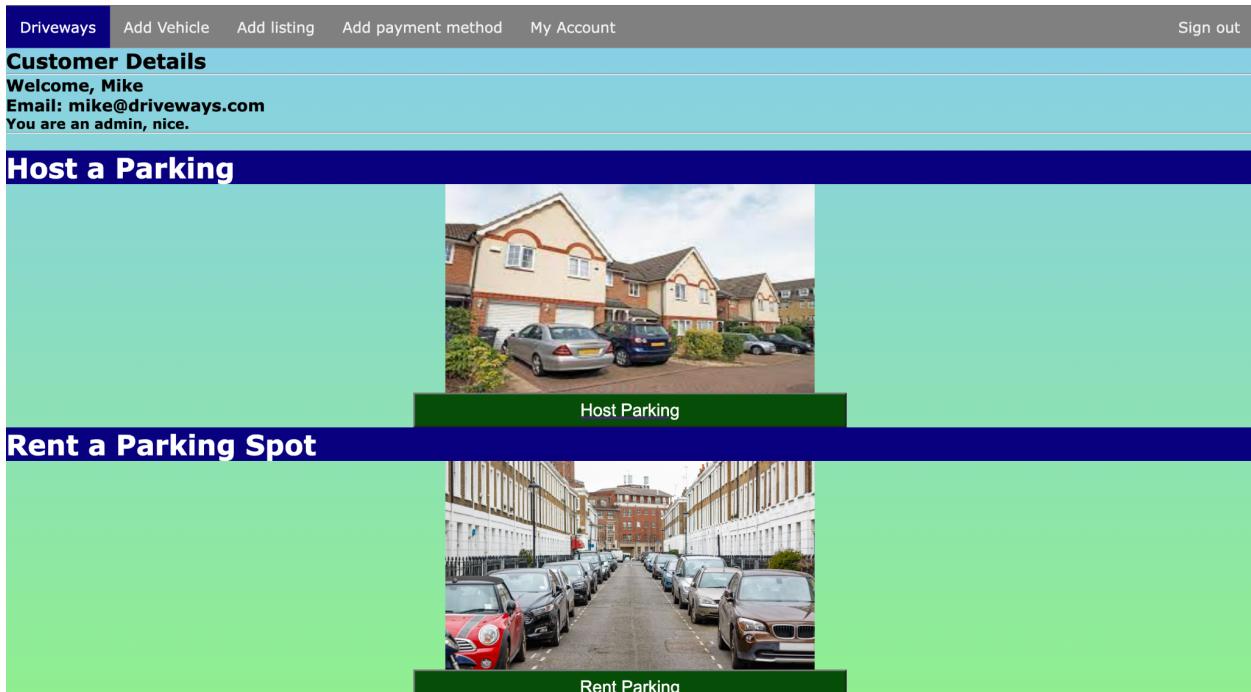


Rent Parking

localhost:8080/driveways/hostparking.jsp

3.1.2. Account creation

The snapshot shows what it looks like to log in as an admin:



The snapshot shows what admin can do and user cannot to a listing on listing page:

User Perspective:

Find Parking

Driveways Add Vehicle Add listing Add payment method My Account Sign out

				Search	Submit
Number	Street	City	Zipcode	Rent this	
123	1st Street	San Jose	95112	Reserve	
123	2nd Street	San Jose	951112	Reserve	
123456	3rd Street	San Jose	95112	Reserve	
10125	Longfellow Ave	Oakland	94603	Reserve	
1217	S Almaden Ave	San Jose	95110	Reserve	
2706	Meadowfaire Dr	San Jose	95111	Reserve	
1721	California St	Mountain View	94041	Reserve	
2540	Howare Ave	San Carlos	94070	Reserve	
1129	San Anselmo Ave	Millbrae	94030	Reserve	
1	E Julian St	San Jose	95112	Reserve	

Back

Admin Perspective:

Find Parking

Driveways Add Vehicle Add listing Add payment method My Account Sign out

				Search	Submit
Number	Street	City	Zipcode	Rent this	Admin Powers
123	1st Street	San Jose	95112	Reserve	Delete Listing
123	2nd Street	San Jose	951112	Reserve	Delete Listing
123456	3rd Street	San Jose	95112	Reserve	Delete Listing
10125	Longfellow Ave	Oakland	94603	Reserve	Delete Listing
1217	S Almaden Ave	San Jose	95110	Reserve	Delete Listing
2706	Meadowfaire Dr	San Jose	95111	Reserve	Delete Listing
1721	California St	Mountain View	94041	Reserve	Delete Listing
2540	Howare Ave	San Carlos	94070	Reserve	Delete Listing
1129	San Anselmo Ave	Millbrae	94030	Reserve	Delete Listing
1	E Julian St	San Jose	95112	Reserve	Delete Listing

Back

The snapshots show payment information requested and contained in the database:

Card Number:

Exp Date:

Submit

Back

MySQL Workbench

Query 5: SELECT * FROM driveway.paymentinformation;

userId	CardNumber	ExpDate	depositInfo
4	132045024	NULL	Bank Account Information
5	327208572	NULL	Bank Account Info.
6	430966566	NULL	Bank Account Info.
7	207281073	NULL	Bank Account Info.
8	891100594	NULL	Bank Account Info.
9	638207995	NULL	Bank Account Info.
10	526986566	NULL	Bank Account Info.
11	942985666	NULL	Bank Account Info.
12	576294234	NULL	Bank Account Info.
13	547966870	NULL	Bank Account Info.
16	87694321	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

3.1.3. Display parking information

The snapshot shows the Google API implementation on the dashboard page:

This screenshot shows the Driveways dashboard interface. At the top, there is a navigation bar with links for "Driveways", "Add Vehicle", "Add listing", "Add payment method", "My Account", and "Sign out". Below the navigation bar, the title "Customer Details" is displayed, followed by a welcome message "Welcome, Gon" and an email address "Email: gon@gmail.com". A large section titled "Parking Spots Available:" displays a map of the San Francisco Bay Area with several red location markers indicating available parking spots. The map includes labels for major cities like San Francisco, Daly City, San Mateo, Hayward, Fremont, Mountain View, and San Jose, along with various towns and roads. Below the map, a section titled "Host a Parking" is visible, featuring a green background and a "Reserve" button.

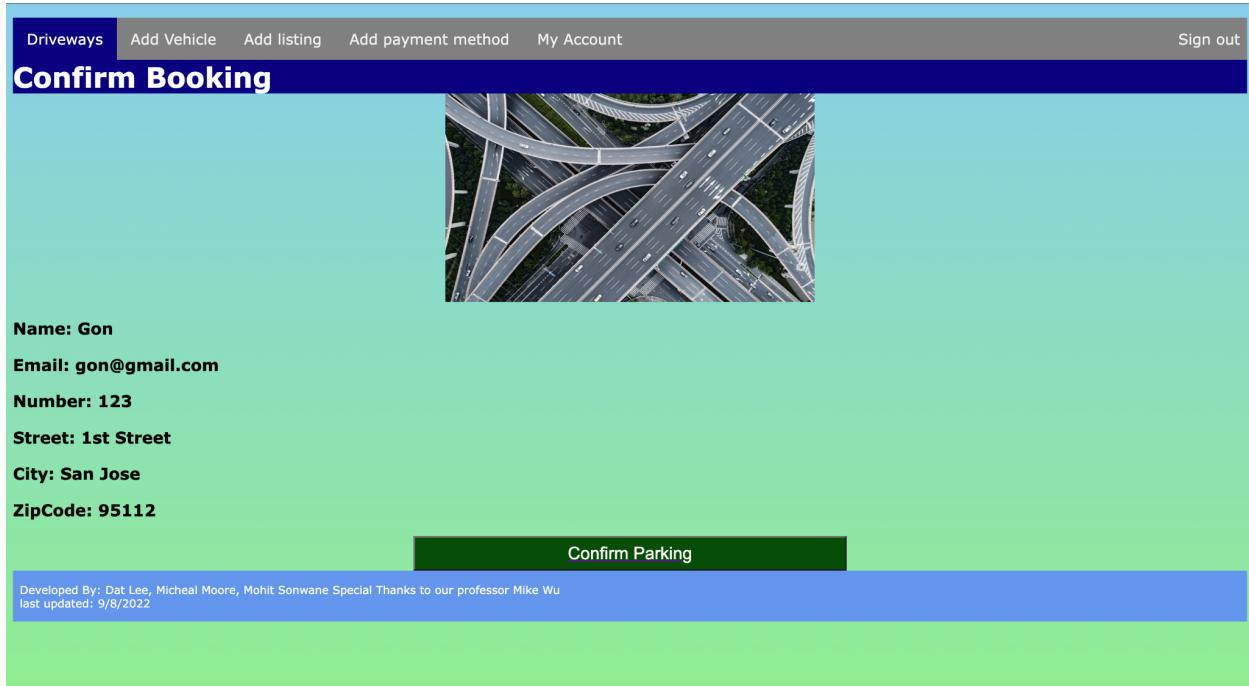
The snapshot shows the listing page:

This screenshot shows the Driveways listing page. The top navigation bar is identical to the dashboard, with links for "Driveways", "Add Vehicle", "Add listing", "Add payment method", "My Account", and "Sign out". Below the navigation bar, the title "Find Parking" is displayed. A search bar with the placeholder "Search" and a "Submit" button are present. The main content area displays a grid of parking listings. Each listing includes a "Number", "Street", "City", "Zipcode", and a "Rent this" button. The "Rent this" button is highlighted with a dark green background and white text. The grid contains the following data:

Number	Street	City	Zipcode	Rent this
123	1st Street	San Jose	95112	Reserve
123	2nd Street	San Jose	951112	Reserve
123456	3rd Street	San Jose	95112	Reserve
10125	Longfellow Ave	Oakland	94603	Reserve
1217	S Almaden Ave	San Jose	95110	Reserve
2706	Meadowfaire Dr	San Jose	95111	Reserve
1721	California St	Mountain View	94041	Reserve
2540	Howare Ave	San Carlos	94070	Reserve
1129	San Anselmo Ave	Millbrae	94030	Reserve
1	E Julian St	San Jose	95112	Reserve

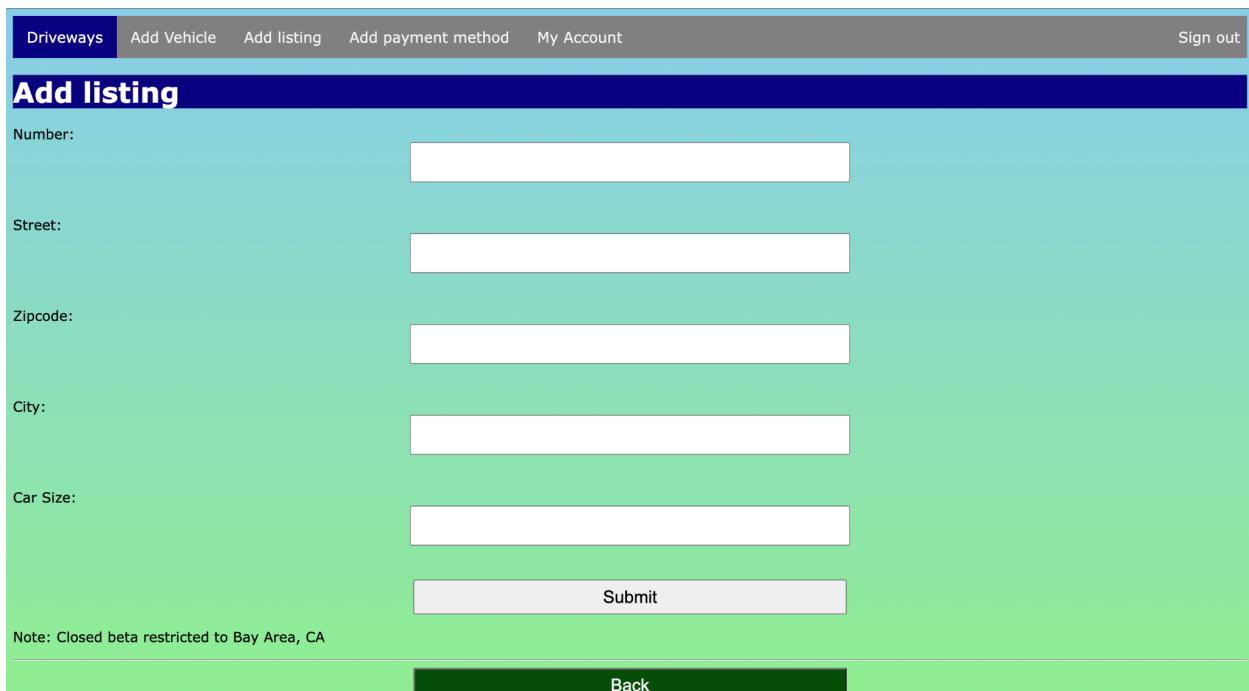
A "Back" button is located at the bottom of the grid.

This snapshot shows the parking information when clicking “Reserve”:



3.1.4. Add a listing, vehicle information and rent a listing

The snapshots shows the add listing page, add vehicle page and add payment info page:



Driveways Add Vehicle Add listing Add payment method My Account Sign out

Vehicle Information

Make:

Model:

Color:

License Plate:

Driveways Add Vehicle Add listing Add payment method My Account Sign out

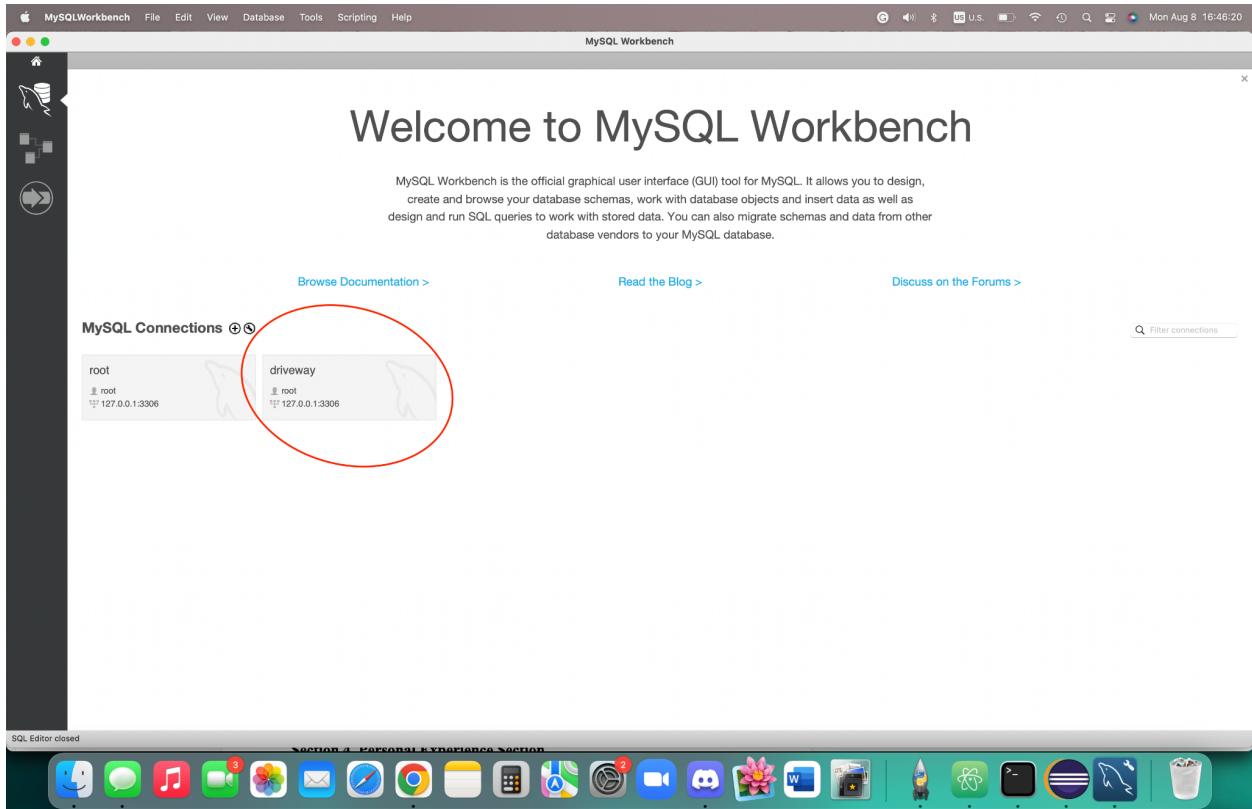
Add Payment Method

Card Number:

Exp Date:

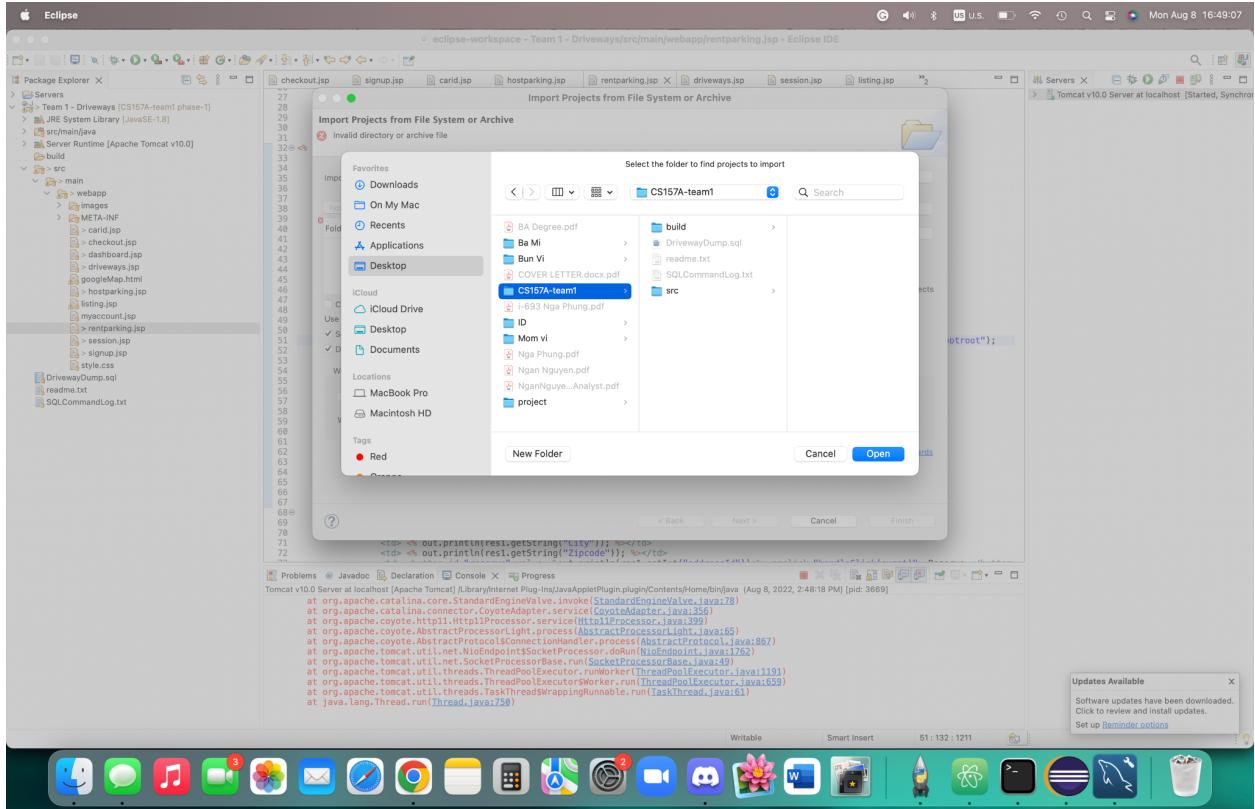
3.2. How to set up and run the system

3.2.1. (In MySQL Workbench) Create a connection named “driveway”



3.2.2. (In MySQL Workbench) Import the DrivewayDump.sql database from the zip file

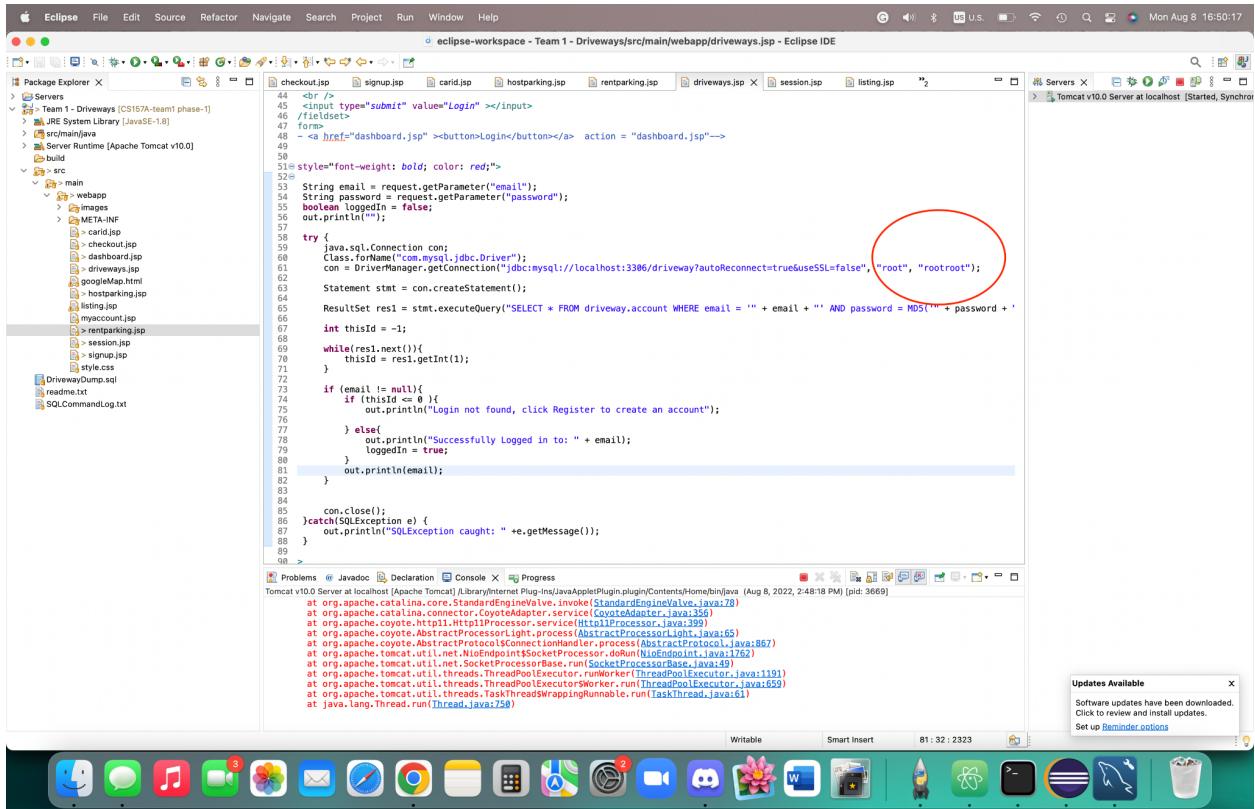
3.2.3. (In Eclipse) Open project from File System → choose the folder contained in the zip file named “CS157A-team1”



3.2.4. (In Eclipse) Correct the username and password to your own credentials in every file in order for the SQL connection to work *The default codes have our credentials which are “root” & “Root123!” → Correct it to your own to have the whole thing run properly*** OR change your database password to match with our password for a quicker access to the application.**

For example:

Driveways 36



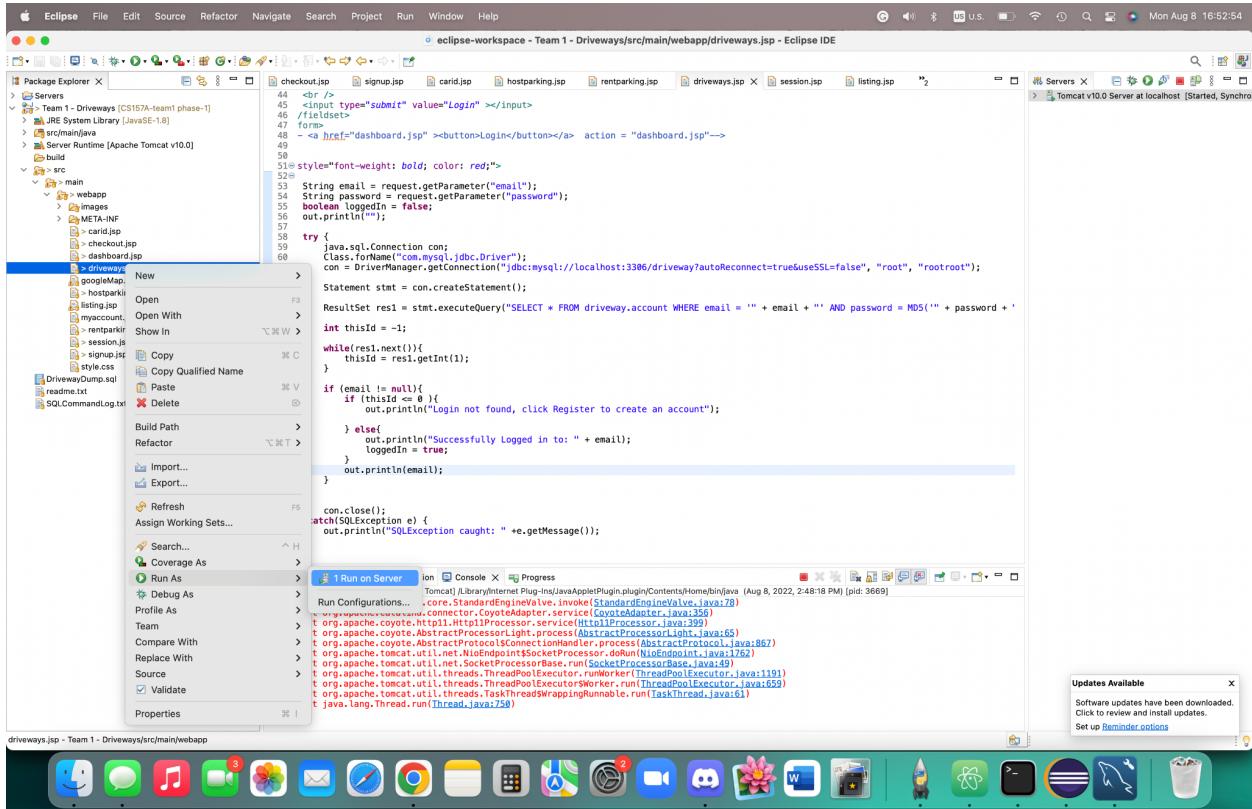
```
44 <br />
45 <input type="submit" value="Login" /></input>
46 </fieldset>
47 <form>
48   -> href="dashboard.jsp" >button>Login</button></a> action = "dashboard.jsp">
49 
50 
51   style="font-weight: bold; color: red;">
52 <@
53   String email = request.getParameter\("email"\);
54   String password = request.getParameter\("password"\);
55   boolean loggedIn = false;
56   out.println\(""\);
57 
58   try {
59     Class.forName\("com.mysql.jdbc.Driver"\);
60     con = DriverManager.getConnection\("jdbc:mysql://localhost:3306/driveway?autoReconnect=true&useSSL=false", "root", "rootroot"\);
61     Statement stmt = con.createStatement\(\);
62     ResultSet res1 = stmt.executeQuery\("SELECT \* FROM driveway.account WHERE email = '" + email + "' AND password = MD5\(" + password + "\)"\);
63     int thisId = -1;
64     while\(res1.next\(\)\){
65       thisId = res1.getInt\(1\);
66     }
67     if \(email != null\){
68       if \(thisId <= 0\){
69         out.println\("Login not found, click Register to create an account"\);
70       } else{
71         out.println\("Successfully Logged in to: " + email\);
72         loggedIn = true;
73       }
74     }
75     out.println\(email\);
76   }
77 
78   con.close\(\);
79 }catch\(SQLException e\) {
80   out.println\("SQLException caught: " +e.getMessage\(\)\);
81 }
82 
83 
84 
85 
86 
87 
88 
89 >
```

The screenshot shows the Eclipse IDE interface with the code editor open to a JSP file named 'driveways.jsp'. The code implements a login logic using JDBC to connect to a MySQL database. A red circle highlights the line of code where the connection is closed after executing the query:

```
con.close();
```

The Eclipse interface includes a Package Explorer, Servers view, and a Problems view showing a stack trace for a SQLException. The status bar at the bottom right indicates the time as 8:32:2323.

3.2.5. (In Eclipse) Run the driveways.jsp with TomCat server



3.2.6. Follow described steps in Section 3.1. to use the web application

Section 4. Personal Experience

4.1. Dat Le

The project is the first real-world application that I have ever had throughout the entire college career. This is also the first time I learned about databases and how a project from Eclipse can connect to a database. By working on this project, I learned how to design a system structure and an ERD, which are essential tools for any software engineer. Also, after completing the project, I now understand the basics of CSS, HTML, and JavaScript. The Driveway project motivated me to pick up and learn about MySQL and other programming languages in two weeks through several online sources. GitHub was also a new experience for me. I have had an account since 2019 and never had a chance to use it. Through doing the project, I finally had a chance to practice using Git and GitHub so as to create a better and smoother working environment for the team. I never recognized that my limit was this far. The GUI of our project is still simple, but this is no doubt one of the proudest innovations that I have ever made in my life. My team is in perfect chemistry with hardworking and persistent mindsets, we never complained about obstacles and challenges when coming across one while building the web application. There is no regret when I have a chance to be in the team.

4.2. Michael Moore

This was my first time working with any database application, and it was a great learning experience. It was also my first time working with JSP, Tomcat, and MySQL specifically. I found the interaction between JSP and MySQL to be very specific, and getting the

syntax correct was sometimes difficult. I spent much of my time getting familiar with JSP and MySQL interactions on one specific part of the project. However, once I was able to figure out one portion of the project, the code was often applicable in many other sections of the project as well. This project was also a great lesson on how to use (and how not to use) Github. Only once we got to the end of the project was I really aware of how to fully utilize the power of branching, merging, and general version control. My teammates were great and really made the fast-paced nature of this summer semester easier to handle.

4.3. Mohit Sonwane

This was my first interesting and challenging group project in which I used a SQL database, and used up to 4 programming languages including: html, css, javascript, jsp. Thus, this project gave me a fruitful experience in web development using SQL databases, and will add to my skill set for my future career. The part I enjoyed the most was to design the ERD diagram of our database, and to learn on how to interact with the database. However, the challenging part for me was to understand how github works, setting up 3 tier architecture, and writing JSP code to retrieve data from SQL databases. Moreover, I had been using VScode/Atom IDE for web development in the past which had so many features to write code smoothly and quickly. Using Eclipse IDE was not a pleasant experience. However, with the help of my panthers, and browsing on the internet , especially stack overflow, and www3 I was able to overcome the problems I was facing. My teammates were very supportive and were open to other people's ideas. Hence, we were able to work efficiently based on the system design and functional specs we

proposed prior to implementation. The only thing I felt was that, as everyone was learning some skills and Git for the first time, we used GitHub in a messy way and did redundant work. But, through proper communication and tracking each other's progress, we were able to solve the problems we faced.