



SOFE 3700U Data Management Systems

Final Report

CarShare - Car Rental Reservation System

Professor: Dr. Khalid A. Hafeez

TA: Taghreed Alghamdi, Heyam Abunaseer

Group#: 25

Date: November 29th, 2021

First Name	Last Name	Student Number
David	Fung	100767734
Saaruca	Kugarajh	100751441
Nathan	Yohannes	100749914
Anish	Patel	100751489

Table of Contents

Table of Contents	2
Introduction	3
Background and Motivation	4
State of Art	4
Problem Statement	4
Solution Relevance	5
Distinguishing Features	6
System Design	7
Design Process	7
Use Cases	9
Test Cases	12
Diagrams	13
Sources of Data	14
Observation and Analysis	17
Conclusion	18
References	20
Appendices	21
A1. Installation Guide and Project Files	21
A2. Phase II Views	21
Contribution Matrix	26

Introduction

Transportation is an essential part of our everyday life. From travelling to running errands, it is important to be able to get from one place to another. Car rental companies aim to simplify transportation by offering many options to fit the needs of the consumer. The Car Rental industry in Canada has shown consistent revenue growth since the last five years [1]. This can be attributed to consumers spending more time on leisure activities such as domestic travel. However, creating a car rental company requires massive amounts of funding in order to be able to afford a large inventory of cars to rent out [2].

CarShare was created as a project that avoids this financial concern by utilizing a peer-to-peer network instead. The CarShare system allows hosts to list their cars online as a rental vehicle for customers to use. The initial prototype uses a local database and functioning web pages that request information from this database. This structure allows CarShare to be hosted online and be used anywhere in the world that is connected to the internet. There are many advantages to a peer-driven online booking system for cars that distinguishes it from a traditional car rental company that uses their own fleet of vehicles. This report will cover the background and motivation of this project, the system design, and the observations and analysis during the creation of this CarShare system.

Background and Motivation

State of Art

Currently, car rental businesses require clients to go through a lengthy process in verifying identification and insurance, which often have long wait times, with prices ranging from \$50 to \$150 per day. For example, if you wanted to rent a Toyota Prius from the company Hertz, it would cost \$112 per day from this Friday, to next Friday for a total of almost \$1100 [3].

A study conducted by the Business Development Bank of Canada (BDC) examined the new work-from-home lifestyle that companies offered since the start of the pandemic. After surveying more than 700 small businesses and 2000 Canadians, they determined that 74% of small-medium business owners will offer their employees work-from-home, 55% of employees prefer work-from-home, and 54% of employees say that work-from-home is a determining factor in applying and accepting a new job [4].

Furthermore, the growing statistics of people working from home indicate the usage of everyday commuter cars have lowered considerably, as people don't need to drive to work as often [5].

Problem Statement

Renting cars is often expensive and requires a lengthy process to complete. Additionally, the ongoing pandemic has resulted in an increase in the work-from-home model and consequently, in several unused cars [5]. Through creating a car rental system, Carshare strives to provide a more affordable and efficient rental service to clients while providing a source of

income for these people who are now engaging in a work-from-home lifestyle. Carshare is a peer-to-peer network of car sharing, where hosts can list their cars online, and customers can rent out any car that fits their needs.

Solution Relevance

Carshare implements a peer-to-peer network sharing model. Clients can sign up to be a host and/or a customer. Hosts are able to list their car online and view reservations made for their listing. Customers can view car listings, and make or delete reservations as they wish to. The system provides different filters to view different sets of car listings.

Carshare peer-to-peer model addresses the difficulties clients face when trying to find cars available for a fair price. Hosts are able to set up their own prices when posting their vehicles for rent. This will create a wide selection of vehicles at different price points in which customers can choose that best fit their personal and financial needs.

Moreover, by creating a peer-to-peer economy, the process required to rent a car is simplified to a peer-to-peer transaction. Thus, the time required for renting a car is heavily reduced.

The database system also attributes to the work-from-home model that is becoming more popular in the workforce. The amount of unused cars previously used for work commute has drastically increased due to the pandemic and the reduced need to commute. Through listing one's own car, hosts are able to generate another form of reliable income through renting their unused cards to customers.

Distinguishing Features

A notable feature to the Carshare system is the implementation of a peer-to-peer sharing network. Carshare works as a facilitator for host to customer communication and provides a platform to rent property to customers, rather than providing the goods themselves. This resembles the model Airbnb, which is a platform that allows people to rent out their residence. This sharing economy model has been adapted to suit the car rental system, which has rarely been done.

Another important feature is the multiple view provided to customers when they are viewing listings. The different views provided to browse listings serve as different filters that customers can use to find a car that best suits their individual needs. This aids in customizing their searches and creating an individualized user experience.

System Design

Design Process

The first step in designing the CarShare system was to create a set of requirements which are captured below in Table 1. These sets of requirements were then used to create a system overview that captures the flow of information from the requirements. The data necessary for the CarShare service was then brainstormed and organized. These became columns for the database that was further organized into tables (Customers, Cars, Hosts, Vehicles). Development then began on the front end which required data from the CarShare database. Front & back end integration was implemented with a REST API to submit data such as vehicle information, and retrieve data such as cars available for renting.

The languages used to create this website are HTML, CSS, PHP and MYSQL. HTML and CSS is used to design and create the user interface and the overall website design. PHP was used to implement a REST API, which facilitates information transactions between the client and the server. The MYSQL database is used to hold all information regarding the customer signup, host signup, bookings and vehicle listings.

We had to examine the larger scope of our website and determine which API to use. Our API options were SOAP and REST, in which we had to weigh the pros and cons of each one to make our choice. The deciding factor was that the REST API is incredibly effective for client-server models, which is the foundation for our reservation system.

REST web services are provided through the signup features for both customers and hosts. The user submits a form with the required information (name, email, telephone etc.)

which will then be retrieved using the POST web service to create a new customer or host, which is stored in their respective table within the database.

A JSON document is updated whenever an entry is added or removed from the database. A function is executed when the user signs up as a host, signs up as a customer, makes or deletes a reservation, or creates a listing. This creates a change to their corresponding tables in the database. When a change is created, the JSON document is updated to retrieve the items within that table.

When designing the system, 10 views were made to create complex website functionality. These views are provided as web services that allow users to filter searches when browsing both listings and reservations. Appendix A2 provides more information on the services provided through examples.

The system has a total of 20 web pages, 10 of which are .php pages that make up the 10 views. These 10 pages are accessible through the vehicles page. The vehicles page displays a table with all entries in the vehicles table. If an entry is added, then the table gets updated. Under the table shows the views and a description for each one. Each view can be accessed with a button. View 1 takes the data from bookings which is an inner join with hosts and customers and displays the data on a table. View 2 gets the user to input an offset year and then displays the vehicles which fall into the offset range entered. View 3 shows all the vehicles which are above the average cost and view 4 gets all the customer bookings. View 5 lists all the hosts and their vehicles which have been booked at least once. View 6 sorts the listings by cost per day and view 7 requires the user to input 2 locations which the listings will filter by. View 8 requires the user to go through a dropdown menu which lists all the types of vehicles and once the user chooses one, it will display a filtered table with all vehicles that are the type selected.

View 9 filters the vehicles by a range of years which the user inputs and view 10 displays all reservations after a certain date which the user enters. CarShare's homepage welcomes the user to the website by giving them a preview of the listings, option to sign up if they would like to put their car up for rent and an option to view all vehicles. All pages of the website have a navbar which lets them navigate to different parts of the website. CarShare also has an about us page which gives the user a little backstory on how CarShare came to life. There is a signup page for both customers and hosts, and also a login page for both. The sign up page for customers requires their first and last name, address, phone number and email address. Once they have entered the required information, the customers table in the database gets updated with a new entry. The sign up page for hosts requires their Social Insurance Number (SIN), first and last name, address, phone number, email address, and Vehicle Identification Number (VIN). It is the same process as the customers sign up but the entry gets added to the hosts table. For the login, it is the same process for customers and hosts. Once the user enters their credentials, the website checks the database to see if an account exists or not. If it does exist, the user has logged in successfully and if it does not, then the user is prompted to enter the correct information or create an account. After the user has logged in, they can access a listings page or a bookings page. It depends on the type of user because if the user has logged in as a customer, they can go to the bookings page and create a reservation. If the user has logged in as a host, then they can go to the listings page and update their listings.

Use Cases

Use Case	Description
UC-1: Manage Listings	The host is able to create new car listings by providing information such as the car model, location, availability, etc. Listings can be deleted by the host at any time. Information can also be modified by the host.
UC-2: Reservations	Customers are able to make reservations for a car rental. This reservation can be deleted at any time before the date of the reservation.
UC-3: Confirm Reservations	An email confirmation should be sent to a customer when making or removing a reservation.
UC-4: View Reservations	Hosts should be able to view reservations made for their car.
UC-5: Signup	Users should be able to create an account and input their information to use the reservation system.
UC-6: Login	Users should be able to login to the system with a username and password.
UC-7: View Listings	Customers should be able to view available cars, and the relevant information about each listing should be displayed.
UC-8: Search Listings	Customers should be able to search for specific cars using different categories such as car model, location, availability, etc.

Table 1. The requirements for the CarShare system were captured as use cases in this table

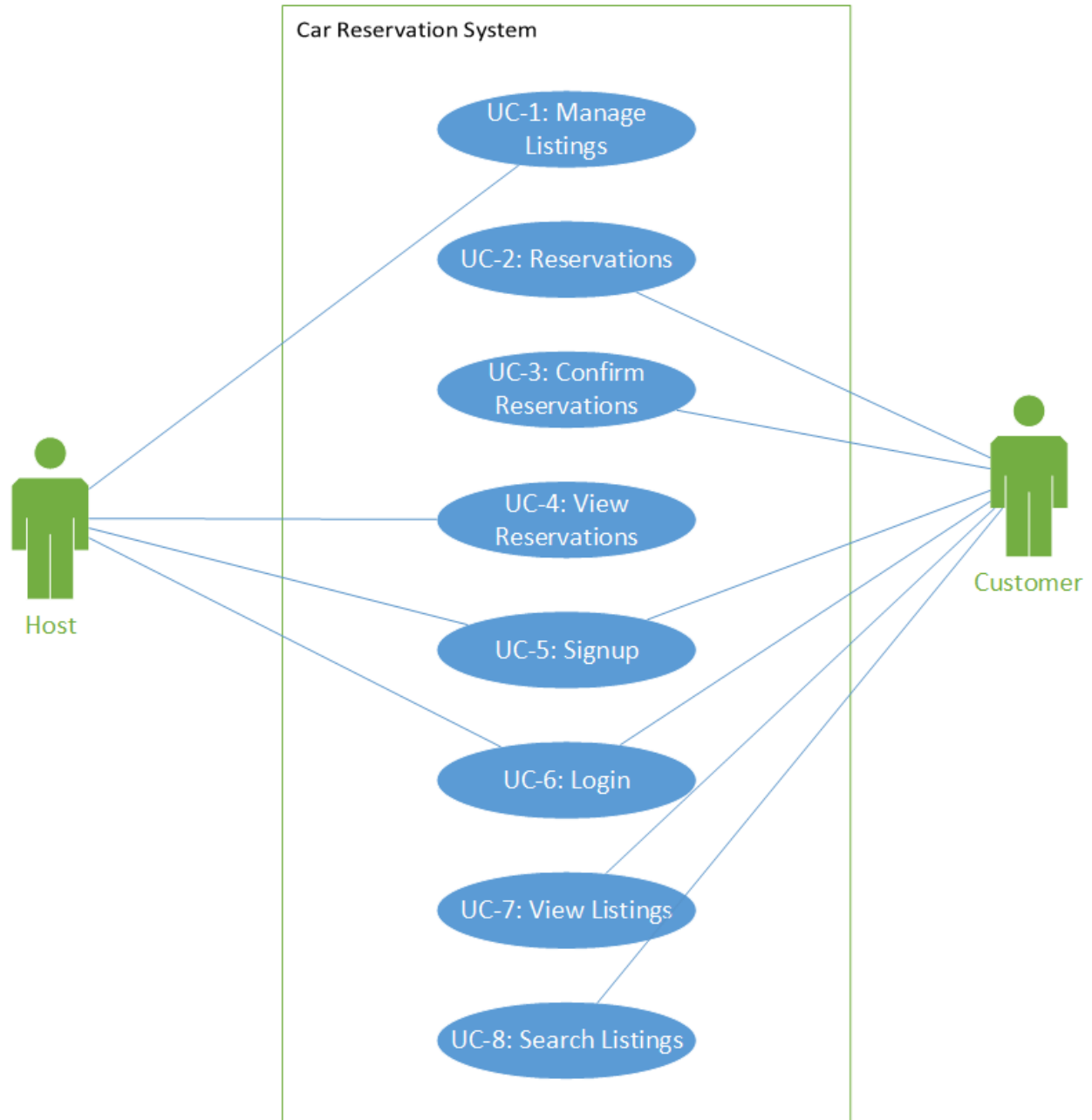


Figure 1. An alternative view to the use cases that displays user interaction with the appropriate requirements

Test Cases

Enter offset years

CLICK FOR VIEW 2 AFTER ENTERING OFFSET

Vehicle ID	Host ID	Brand	Model	Year	Type	Location	Daily Cost
1324	99102658	BMW	M4	2019	Sports Car	Toronto	250
2004	67899055	Audi	A4	2020	Sedan	Oshawa	175
3345	48915659	Dodge	Caravan	2021	Van	Brampton	275

```
"SELECT * FROM vehicles WHERE v_year > ANY (SELECT AVG(v_year - ". $offset .") FROM vehicles) GROUP BY v_type";
```

Figure 2. An example of a query being tested and the results showing the appropriate view after executing

View 1
Get all reservations, all information

Booking ID	Host ID	Customer ID	Vehicle ID	Start Date	End Date	Host ID	Host Name	Host Address	Host Telephone	Host Email
345244	12345678	77889900	8888	2023-08-01	2023-08-07	12345678	John Doe	123 Main St	416-123-4567	john.doe@xyz.com
677890	45678901	90123456	9999	2023-09-15	2023-09-22	45678901	Jane Smith	456 Elm St	416-987-6543	jane.smith@abc.com
789012	89012345	23456789	1000	2023-10-01	2023-10-05	89012345	Mike Johnson	789 Oak St	416-555-1234	mike.j@def.com
123456	56789012	34567890	1111	2023-11-01	2023-11-08	56789012	Sarah Lee	321 Pine St	416-222-3333	sarah.lee@ghi.com
234567	67890123	45678901	2222	2023-12-01	2023-12-07	67890123	David Kim	654 Birch St	416-777-8888	david.kim@jkl.com
345678	78901234	56789012	3333	2024-01-01	2024-01-05	78901234	Emily White	987 Cedar St	416-444-5555	emily.white@mno.com
456789	89012345	67890123	4444	2024-02-01	2024-02-06	89012345	Chris Brown	101 Maple St	416-333-4444	chris.brown@pqr.com

```
"SELECT * FROM bookings INNER JOIN hosts ON bookings.hosts_id=hosts.h_id INNER JOIN customers ON bookings.customer_id=customers.c_id";
```

Figure 3. An example of a query being tested and the webpage showing the appropriate information from the database

Diagrams

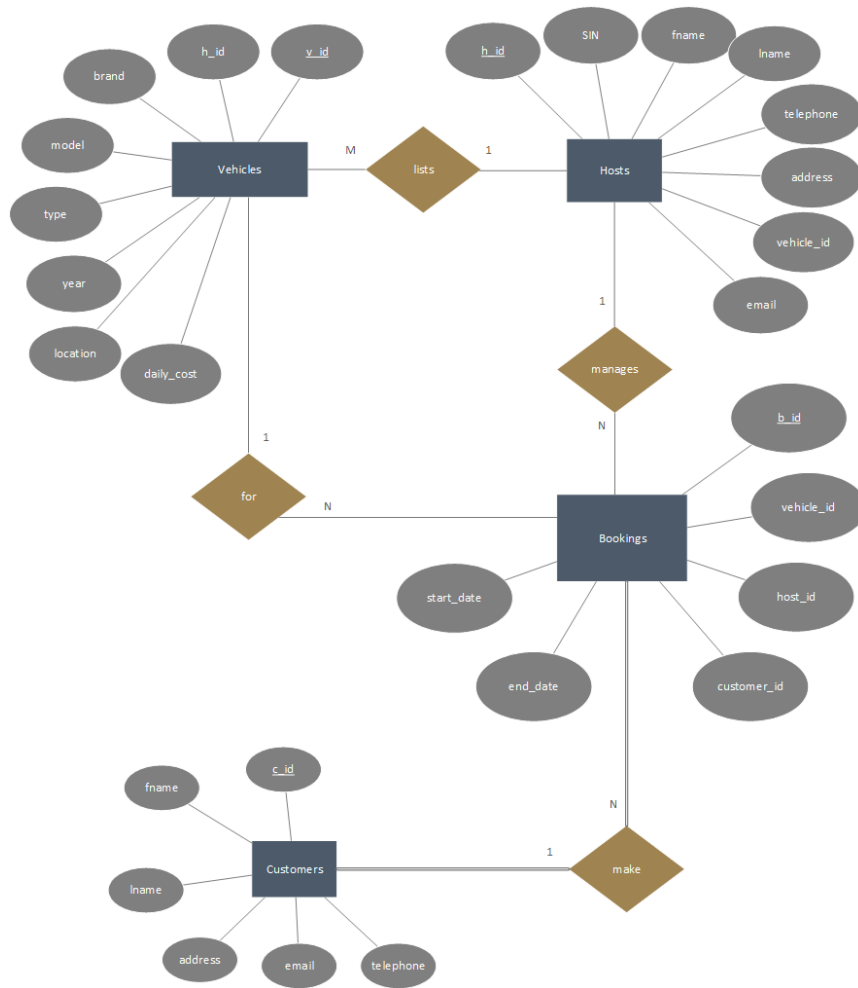


Figure 4. The ER diagram that captures the overall structure of the database and showing the relationships between the tables

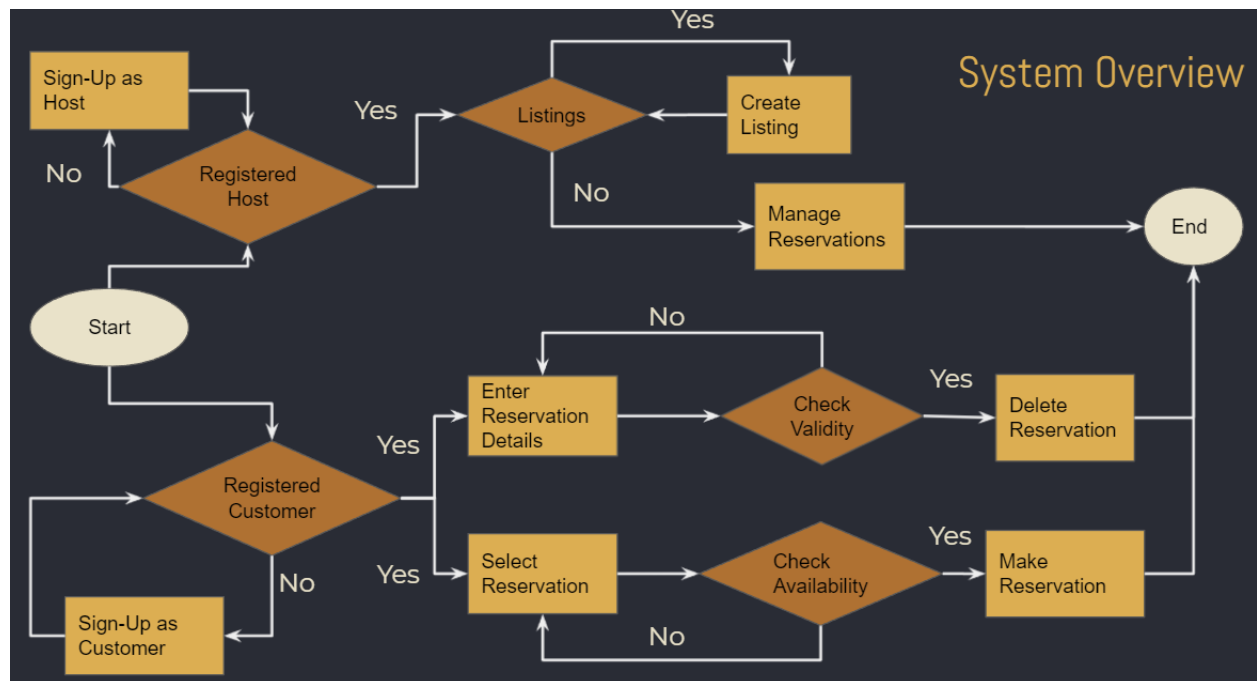


Figure 5. The system overview was created using the use cases and shows the flow of information in this diagram

Sources of Data

The data stored in the database is provided by customers and hosts. These are defined by the tables hosts, customers, bookings and vehicles. Sample data was provided for the purposes of system testing and development.

Customers					
c_id: integer	fname: string	lname: string	address: string	telephone: string	email: string
1121477	Andrew	White	55 Simcoe Street	647-634-3356	andrew_white@gmail.com
1005896	Anish	Patel	1800 Simcoe Street	647-999-5555	anishpatel@gmail.com
1778953	Suzan	Smith	44 Bridle Path	416-782-5189	suzansmith77@gmail.com
5962189	John	Doe	88 Toronto Blvd	437-891-8515	johndoe@gmail.com
7780458	Marie	Novak	7180 Markham Road	416-872-6156	marienovak@hotmail.com

8825692	Ronald	Forbes	37 Garfield Drive	905-456-1567	forbesronald@gmail.com
---------	--------	--------	-------------------	--------------	------------------------

Table 2. The customers table displayed above shows the column types and sample data of customers

Vehicles						
v_id: integer	host_id : int	brand: string	model: string	v_year: int	type: string	location: string
2004	67899055	Audi	A4	2020	Sedan	Oshawa
6633	77561892	Honda	Civic	2017	Sports Car	Markham
1324	99102658	BMW	M4	2019	Sports Car	Toronto
3345	48915659	Dodge	Caravan	2021	Van	Brampton
8988	15918748	Land Rover	Range Rover	2008	SUV	Waterloo
9008	48516698	Toyota	Prius	2014	Hybrid	Vaughan

Table 3. The vehicles table displayed above shows the column types and sample data of vehicles

Hosts							
h_id: int	SIN: int	fname: string	lname: string	address: string	telephone: string	email: string	vehicle_id: int
67899055	490943566	Phillip	Reed	21 Checker Street	435-645-2234	phillipreed@gmail.com	2004
77561892	512605778	David	Fung	107 Bridle Path	647-781-6318	davidfung@gmail.com	6633
99102658	419765205	Elle	Amber	47 High Point Lane	647-178-4186	elleamber@yahoo.ca	1324
48915659	418915115	Laura	Dawson	4182 Nelson Road	516-485-1569	ldawson@gmail.com	3345
15918748	415895288	Erin	Stone	48 Riverside Road	416-784-8651	erinstone@gmail.com	8988
48516698	597853895	Johnathan	Green	59 Windpath Road	647-581-5459	johnathang@gmail.com	9008

Table 4. The hosts table displayed above shows the column types and sample data of hosts

Bookings					
b_id	host_id: int	customer_id: int	vehicle_id: int	start_date: date	end_date: date
123456	67899055	1121477	2004	2020-12-20	2020-12-25
234567	77561892	1005896	6633	2021-11-07	2021-11-17
435567	99102658	1778953	1324	2022-01-03	2022-01-14
745567	48915659	5962189	3345	2021-12-24	2022-01-03
546544	15918748	7780458	8988	2022-04-20	2022-04-27
657886	48516698	8825692	9008	2022-02-22	2022-03-02

Table 5. The bookings table displayed above shows the column types and sample data of bookings

Observation and Analysis

CarShare as a service is a solution to several modern problems. The niche it fills is highly applicable to a wide range of people while keeping simplicity in all aspects of design. Maintaining ease of use is critical both on the Car-Hosts and Car-Renters sides to ensure that those who need a temporary car service and people who would like to rent their vehicle have a simple and safe way to do so. CarShare provides a solution to many prevalent problems that by honouring the principle of simplicity and user experience, it poses a potential to reach levels of ubiquity similar to that of Airbnb, Hertz, Enterprise, Etc.

CarShare's development had set-backs/slow points, which were the result of several factors. The scope of CarShare's reach, whether being a Provincial, National, International, or Global system was discussed heavily. Backend development proves to be an arduous task in order to keep user data secure while keeping data during a session and sharing information with a database.

As car share nears a production level, features that ensure fair-play between both parties should be addressed; this includes being able to report bad-actors. In confirmed cases of misconduct, an internal integrity meter should reflect such actions. In response to this integrity measure, accounts may be blacklisted (banned), or temporarily restricted from using the service on account of keeping all users safe.

Conclusion

While creating the CarShare system, we faced many challenges along the process. Conceptualizing the user interface and user experience was the first big obstacle that we experienced. It is difficult to determine which functions were necessary to implement, how we could implement these functions, and how clients would be able to access these services. We managed to brainstorm use cases and referred to these use cases to ensure we met all our goals when constructing the user interface. We wanted to create an effective user experience and used a system diagram to make sure the interface was intuitive and functional. The next challenge we faced was to determine how client information was stored and retrieved from the database in order to complete these functions. We decided to implement a REST API as a solution to retrieve information from the database.

Some future improvements we could implement would be to create a single login for both hosts and customers. Initially, we faced a lot of issues when trying to merge the functionalities while creating the project, and we believe that this feature would be a great addition to improving user experience by simplifying the interface. The next improvement would be able to allow hosts to make and delete reservations, since only customers currently have this capability. We believe this is a necessary feature for hosts as well as it provides more autonomy to their own listings. Our final step would be to launch the project online by hosting the website on a public server and utilizing a cloud platform to host our database. A cloud platform such as Amazon's relational database services would offer great scalability, fast performance, high availability, and high security.

This project has been a long journey and we have learned a lot on how to implement databases and create a functioning front end web application. Our group was able to effectively solve the many challenges along the way by using creative solutions or breaking down the problem into smaller steps. We believe our project has great potential and offers many advantages when compared to traditional car rental options, and hope that one day it can serve as a prototype for a real peer-driven car rental system.

References

- [1] "Car Rental in Canada - Market Research Report." *Industry Research Reports*. Nov. 2021. Available: <https://www.ibisworld.com/canada/market-research-reports/car-rental-industry/>. Accessed on: Oct. 20, 2021.
- [2] "Car Rental Market Size, Share & COVID-19 Impact Analysis, By Vehicle Type (Luxury Cars, Executive Cars, Economy Cars, SUVs and MUVs). By Application Type (Local Usage, Airport Transport, Outstation and Others), By Rental Duration Type (Short-Term, Long-Term), and Regional Forecast, 2020-2027." *Market Research Report*. Apr. 2021. Available: <https://www.fortunebusinessinsights.com/car-rental-market-105117>. Accessed on: Oct. 20, 2021.
- [3] Hertz Reservations Vehicles. Available: <https://www.hertz.ca/reservations/vehicles>. Accessed on: Nov. 29, 2021.
- [4] "Remote work is here to stay: BDC study." Jun. 2021. Available: <https://www.bdc.ca/en/about/mediaroom/news-releases/remote-work-here-stay-bdc-study>. Accessed on: Oct 20, 2021.
- [5] "Remote working and online shopping could drive 14 million cars off US roads - permanently". *World Economic Forum*. Aug. 2020. Available: <https://www.weforum.org/agenda/2020/08/remote-working-online-shopping-millions-cars-off-us-roads/>. Accessed on: Oct 20, 2021.

Appendices

A1. Installation Guide and Project Files

The installation guide and project files are included in this GitHub link:

<https://github.com/anishp22/DataManagementFinal>

A2. Phase II Views

View 1: Computes a join of at least three tables

The screenshot shows a database query editor with a tab labeled 'vehicles'. The SQL query is as follows:

```
1 SELECT * FROM bookings
2 INNER JOIN hosts
3 on bookings.host_id=hosts.h_id
4 Inner JOIN customers
5 on bookings.customer_id=customers.c_id;
```

Below the query editor is the 'Result Grid' showing the output of the query. The grid has 12 columns: b_id, host_id, customer_id, vehide_id, start_date, end_date, h_id, SIN, fname, lname, address, and teleph. The data is as follows:

b_id	host_id	customer_id	vehide_id	start_date	end_date	h_id	SIN	fname	lname	address	teleph
234567	77561892	1005896	6633	2021-11-07	2021-11-17	77561892	512605778	David	Fung	107 Bridle Path	647-
123456	67899055	1121477	2004	2020-12-20	2020-12-25	67899055	490943566	Phillip	Reed	21 Checker S...	435-
435567	99102658	1778953	1324	2022-01-03	2022-01-14	99102658	419765205	Elle	Amber	47 High Point ...	647-
745567	48915659	5962189	3345	2021-12-24	2022-01-03	48915659	418915115	Laura	Dawson	4182 Nelson ...	516-
546544	15918748	7780458	8988	2022-04-20	2022-04-27	15918748	415895288	Erin	Stone	48 Riverside ...	416-
657886	48516698	8825692	9008	2022-02-22	2022-03-02	48516698	597853895	Johnathan	Green	59 Windpath ...	647-

Returns a view of the inner join of bookings with hosts and customers.

View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause

The screenshot shows a database query editor with a tab labeled 'vehicles'. The SQL query is as follows:

```
1 SELECT *
2 FROM vehicles
3 WHERE v_year > ANY (
4     SELECT AVG(v_year - 2 )
5     FROM vehicles)
6 GROUP BY v_type;
```

Below the query editor is the 'Result Grid' showing the output of the query. The grid has 8 columns: v_id, host_id, brand, model, v_year, v_type, and location. The data is as follows:

v_id	host_id	brand	model	v_year	v_type	location
1324	99102658	BMW	M4	2019	Sports Car	Toronto
2004	67899055	Audi	A4	2020	Sedan	Oshawa
3345	48915659	Dodge	Caravan	2021	Van	Brampton
*	NULL	NULL	NULL	NULL	NULL	NULL

Returns a view of vehicles with a year greater than any avg minus 2 and groups vehicles with the same model.

View 3: A correlated nested query

```

1 • SELECT brand, model, v_year, v_type, daily_cost
2   FROM vehicles WHERE daily_cost > (
3   SELECT AVG(daily_cost) FROM vehicles);

```

brand	model	v_year	v_type	daily_cost
BMW	M4	2019	Sports Car	250
Dodge	Caravan	2021	Van	275
Land Rover	Range Rover	2008	SUV	355

Returns a view of all vehicles with a daily cost that is greater than the average daily cost.

View 4: Uses a FULL JOIN

```

1 • SELECT * from customers
2   Full Join bookings
3   on c_id=bookings.customer_id;

```

c_id	fname	lname	address	telephone	email	b_id	host_id	customer_id	vehicle_id	start
1005896	Anish	Patel	1800 Simcoe St...	647-999-5555	anishpatel@gmail.com	234567	77561892	1005896	6633	2021
1121477	Andrew	White	55 Simcoe Street	647-634-3356	andrew_white@gmail.com	123456	67899055	1121477	2004	2020
1778953	Suzan	Smith	44 Bridle Path	416-782-5189	suzansmith77@gmail.com	435567	99102658	1778953	1324	2022
5962189	John	Doe	88 Toronto Blvd	437-891-8515	johndoe@gmail.com	745567	48915659	5962189	3345	2021
7780458	Marie	Novak	7180 Markham ...	416-872-6156	marienovak@hotmail.com	546544	15918748	7780458	8988	2022
8825692	Ronald	Forbes	37 Garfield Drive	905-456-1567	forbesronald@gmail.com	657886	48516698	8825692	9008	2022

Full join combines the results of both left and right outer joins. It will contain records from both tables and fill in NULLs for the missing matches on either side.

View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT

```

1 • SELECT v_id, host_id
2 FROM vehicles
3 UNION
4 SELECT vehicle_id, host_id
5 FROM bookings
6

```

Result Grid

	v_id	host_id
▶	8988	15918748
	9008	48516698
	3345	48915659
	2004	67899055
	6633	77561892
	1324	99102658

Views the combined results of vehicle id and host id in vehicles and booking table.

View 6:

bookings bookings customers vehicles vehicles - table vehicles car_rental.vehicles

```

1 • SELECT * FROM car_rental.vehicles
2 ORDER BY daily_cost;
3

```

Result Grid

	v_id	host_id	brand	model	year	type	location	daily_cost
▶	9008	48516698	Toyota	Prius	2014	Hybrid	Vaughan	125
	2004	67899055	Audi	A4	2020	Sedan	Oshawa	175
	6633	77561892	Honda	Civic	2017	Sports Car	Markham	200
	1324	99102658	BMW	M4	2019	Sports Car	Toronto	250
	3345	48915659	Dodge	Caravan	2021	Van	Brampton	275
	8988	15918748	Land Rover	Range Rover	2008	SUV	Waterloo	355
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Orders the vehicles table by increasing cost.

View 7:

bookings bookings customers **vehicles** vehicles - table vehicles car_rental.vehicles

Limit to 1000 rows

```

1 • SELECT * FROM car_rental.vehicles
2 WHERE location = "Oshawa" OR location = "Toronto";
3

```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	v_id	host_id	brand	model	year	type	location	daily_cost
▶	1324	99102658	BMW	M4	2019	Sports Car	Toronto	250
	2004	67899055	Audi	A4	2020	Sedan	Oshawa	175
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Returns a view of all available vehicles in Oshawa or Toronto.

View 8:

bookings bookings customers vehicles **vehicles** car_rental.vehicles vehicles

Limit to 1000 rows

```

1 • SELECT * FROM car_rental.vehicles
2 WHERE v_type = "Sports Car";

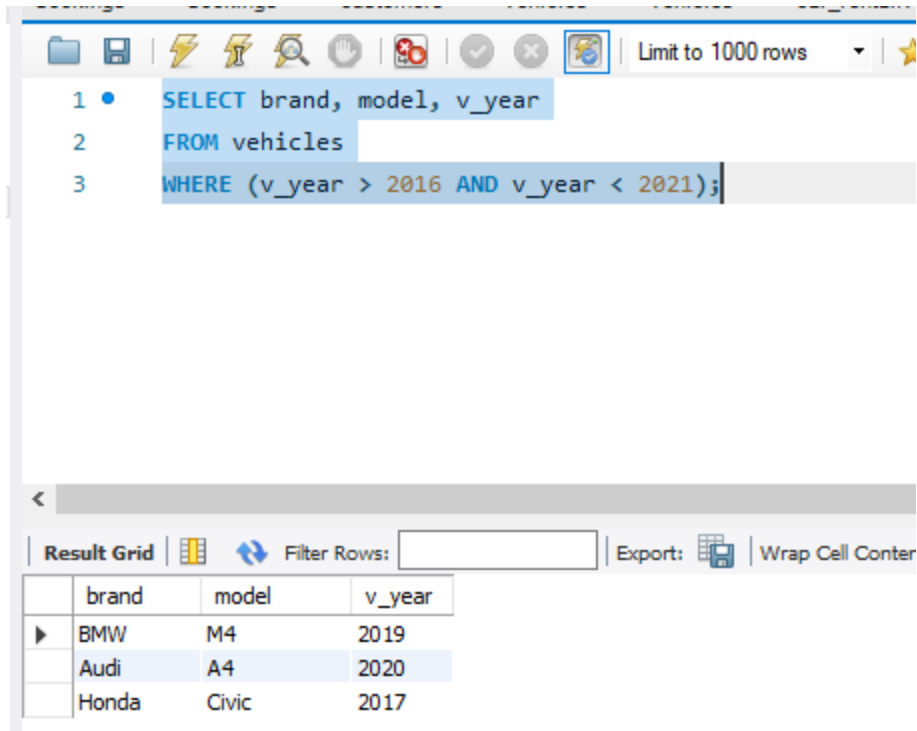
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

	v_id	host_id	brand	model	v_year	v_type	location	daily_cost
▶	1324	99102658	BMW	M4	2019	Sports Car	Toronto	250
	6633	77561892	Honda	Civic	2017	Sports Car	Markham	200
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Returns a view of all available vehicles of the type "Sports Car".

View 9:



The screenshot shows a SQL query editor with the following query:

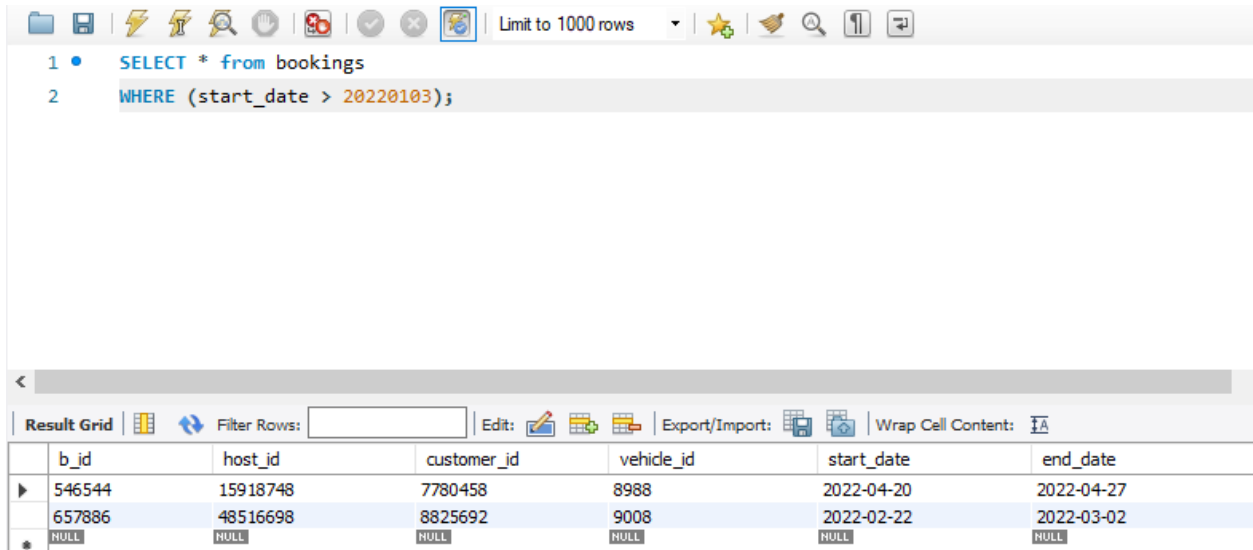
```
1 • SELECT brand, model, v_year
2 FROM vehicles
3 WHERE (v_year > 2016 AND v_year < 2021);
```

The result grid below the query shows the following data:

	brand	model	v_year
▶	BMW	M4	2019
	Audi	A4	2020
	Honda	Civic	2017

Returns a view that filters cars between the years 2016 and 2021.

View 10:



The screenshot shows a SQL query editor with the following query:

```
1 • SELECT * from bookings
2 WHERE (start_date > 20220103);
```

The result grid below the query shows the following data:

	b_id	host_id	customer_id	vehide_id	start_date	end_date
▶	546544	15918748	7780458	8988	2022-04-20	2022-04-27
	657886	48516698	8825692	9008	2022-02-22	2022-03-02
*	NULL	NULL	NULL	NULL	NULL	NULL

Returns a view of all bookings with the start date later than 2022-01-03.

Contribution Matrix

Student Name	Saaruca Kugarajh	David Fung	Anish Patel	Nathan Yohannes
Student Number	100751441	100767734	100751489	100749914
Phase 1				
Overview	25%	25%	25%	25%
Problems Addressed	25%	25%	25%	25%
Goals and Motivation	25%	25%	25%	25%
Related Work	25%	25%	25%	25%
Methodology and Plan	25%	25%	25%	25%
Contribution (%)	25%	25%	25%	25%
Phase 2				
Relational Schema	0%	80%	0%	20%
Sample Data	0%	0%	80%	20%
Views	20%	20%	20%	40%
ER Diagram	80%	0%	0%	20%
Contribution (%)	25%	25%	25%	25%
Phase 3				
Presentation	25%	25%	25%	25%
Final Report	30%	30%	20%	20%
Website Design	20%	20%	30%	30%
Contribution (%)	25%	25%	25%	25%
Total Contribution	25%	25%	25%	25%