



PROJECT REPORT

BIKE RENTAL MANAGEMENT SYSTEM

Prepared By:

Kirti (IIT2020142)
Abhinandan (IIT2020119)
Shivam Harjani (IIT2020121)
Aditi (IIT2020138)

Introduction

An online bike rental system that allows customers to search for available rental bikes in a more efficient manner than the old one, saving time and energy. Admin and User are the two parts that make up this renting system. Admins can go in to add, amend, and delete information on bikes, as well as update the list of them. He or she has access to all users' bookings and may manage their bookings and transactions. Users may create an account on the website, log in, check the availability of bikes, book the bike of their preference, and pay for it.

How to run

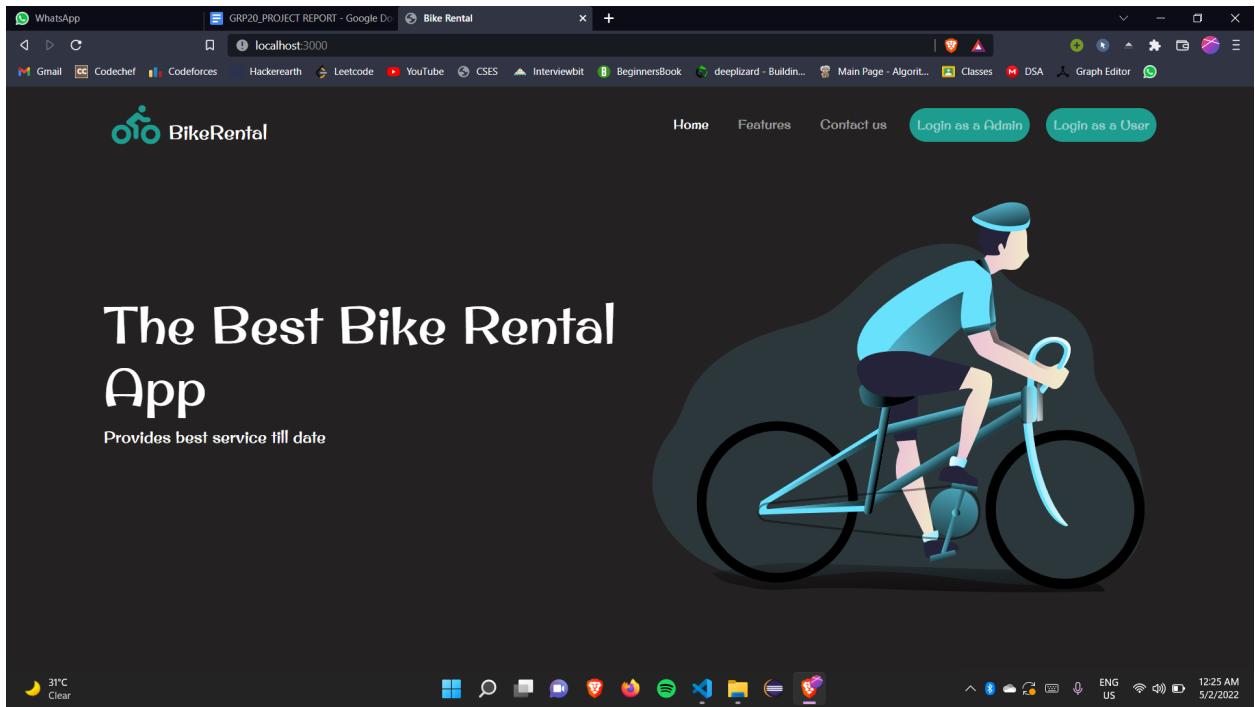
1. Run the bike_rental_database.sql in MySql workbench
2. Change the password in app.js to your database password.
3. run command npm start in your terminal
4. Your project will run on http://localhost:3000/

The screenshot shows the Visual Studio Code interface with the following details:

- Code Editor:** The main area displays the `app.js` file, which contains JavaScript code for a Node.js application. The code includes logic for adding bikes and handling user bookings.
- Terminal:** The bottom-left terminal window shows the command line output of running the project:

```
PS C:\Users\Abhinandan\Desktop\Bike_Rental_System\Bike_Rental_System-main> npm start
> bike_rental_system@1.0.0 start
> nodemon app.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
```
- File Explorer:** The right-hand sidebar shows the project structure under "BIKE_RENTAL_SYSTEM-MAIN". It includes files like `index.html`, `plusbikes.html`, `updatepass.html`, and `userregister.html`. A "views" folder contains `app.js` and other static files.
- Bottom Status Bar:** Shows the current line (Ln 329), column (Col 38), and selected text (6 selected). It also displays system status icons for battery, signal, and network.



Functionalities

- **Admin**

1. Login
2. Update bike information
3. Manage invoice
4. Provide transactions
5. Change of passwords
6. Update invoice
7. Access to all bookings
8. Updating details

- **User**

1. Login
2. Booking a bike

3. Update self credentials
4. Canceling a bike
5. Receive invoice
6. Receive transaction
7. Change password
8. Selecting bike as per of given type

Entities along with the schema

1. Admin

Attributes:

id	PRIMARY KEY,
username	CANDIDATE KEY,
password ,	
email	

2. User

Attributes:

id	PRIMARY KEY,
username	CANDIDATE KEY,
contact number,	
password,	
license	CANDIDATE KEY,
email,	
address	

3. Bike

Attributes:

id	PRIMARY KEY,
name,	
cost,	
feature,	
model,	
description,	
image	

4. Bookings

Attributes:

id	PRIMARY KEY,
username	FOREIGN KEY references user(username),
password,	
email	

Relationships

1. Admin to User
2. User to Bikes
3. User to Bookings
4. Admin to Bookings
5. Admin to Bikes
6. Bookings to Bikes

Tables

1. Admin accounts table

The screenshot shows the MySQL Workbench interface with the 'accounts' table selected. The table has columns: id, username, password, and email. The data grid shows 10 rows of account information.

	id	username	password	email
1	test	test	test@test.com	
2	IT2020142	karti	singh.kr2002@gmail.com	
3	hello	hello	singh.kr2002@gmail.com	
4	hihi	hi	singh.kr16@gmail.com	
5	sushil	sushil	sushil@sushil	
6	shruti	shruti	shruti@shruti	
7	pqr	pqr	pqr @pqr	
8	pqr	pqr	pqr @pqr	
9	weak	weak	weka@weak	
10	hello	hello	hello @hello	

Below the table, the 'Output' pane shows the results of the query: 'SELECT * FROM nodelogin.accounts LIMIT 0, 1000'. It indicates 10 row(s) returned and a duration of 0.000 sec / 0.000 sec.

2. Bikes table

The screenshot shows the MySQL Workbench interface with the 'add_bikes' table selected. The table has columns: id, name, description, model, cost, features, file_data. The data grid shows 9 rows of bike information.

	id	name	description	model	cost	features	file_data
2	2	Tvs activa	bhhdifdf	1990	7	dual disc	11.000
3	3	TVS activa	nkjdfdf	1990	8	dual disc	11.000
4	4	honda scooter	fudifdfdf	1890	90	brake system	11.000
5	5	pulsar	fvgdfdf	2009	24	brake system	11.000
6	6	activaa	bqdfdfdf	2013	67	braking system	11.000
7	7	tvs	ghggy	1234	67	dual	11.000
8	8	pqr	ngflgfg	2019	12	dual disc	11.000
9	9	activa lucor	jnfjkgf	2345	567	dual disc	11.000

Below the table, the 'Output' pane shows the results of two queries: 'SELECT * FROM nodelogin.accounts LIMIT 0, 1000' and 'SELECT * FROM nodelogin.add_bikes LIMIT 0, 1000'. Both indicate 10 row(s) returned and a duration of 0.000 sec / 0.000 sec.

3. User accounts table

The screenshot shows the MySQL Workbench interface with the 'admin' schema selected. The 'Tables' pane shows the 'admin' table. The 'Result Grid' pane displays the following data:

	id	username	password	email
1	kirti	kirti	kirti@kirti.com	
2	krishna	krishna	krishna@k.com	
3	abhi	abhi	abhi@abhi	
4	aditi	aditi	aditi@aditi	
5	test	test	test@test	

The 'Output' pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
1	00:07:45	SELECT * FROM nodelogin.accounts LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
2	00:08:06	SELECT * FROM nodelogin.add_bikes LIMIT 0, 1000	8 row(s) returned	0.016 sec / 0.000 sec
3	00:08:33	SELECT * FROM nodelogin.admin LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

4. Bookings table

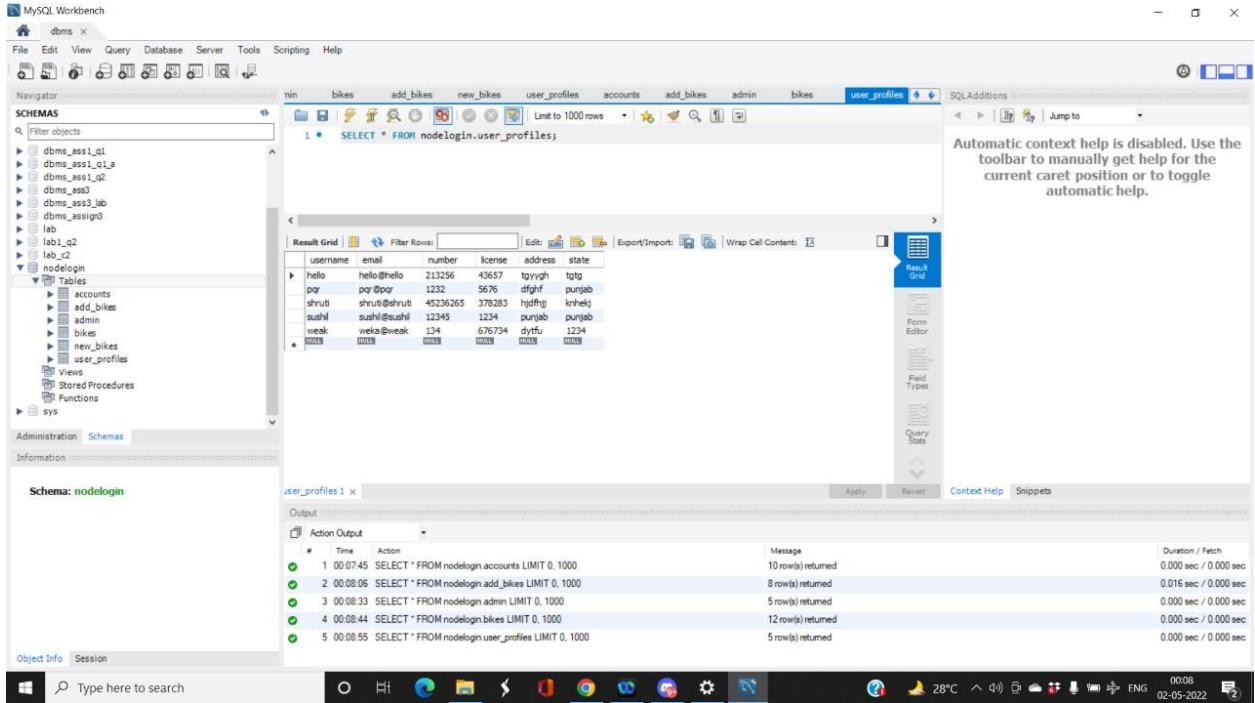
The screenshot shows the MySQL Workbench interface with the 'bikes' schema selected. The 'Tables' pane shows the 'bikes' table. The 'Result Grid' pane displays the following data:

	id	name	mobile	address	license	model	area	email
3	hh	566	fdff	dfdfdfdf	def	dfdf	dfdfg@hh	
4	kerti	1234	bt	ighy	ghykm	overt	kar@kerti	
5	kerti	1234	bt	ighy	ghykm	overt	kar@kerti	
6	kapal	5677	ludh	hu	pun	pun	kapal@kapal	
7	kapal	5677	ludh	hu	pun	pun	kapal@kapal	
8	krishna	8956	gul	lc	guj	guj	krish@krish	
9	ashita	7890	luc	luc	luc	luc	ashita@ashita	
10	shruti	1244	nbagar	guj7	pun	pun	shruti@shruti	
11	test	1325	fgfifgf	cfc	df	dd	dd@dd	
12	ttd	122	1223@fgv	12234	ydfh	ydf	hoh@njdj	
16	sushi	143143	hdffg	132	lp009	jhbjh	dv@dhf	
18	hello	234	föngif	335	gfh	fög	dgg@dg	

The 'Output' pane shows the execution history:

#	Time	Action	Message	Duration / Fetch
1	00:07:45	SELECT * FROM nodelogin.accounts LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
2	00:08:06	SELECT * FROM nodelogin.add_bikes LIMIT 0, 1000	8 row(s) returned	0.016 sec / 0.000 sec
3	00:08:33	SELECT * FROM nodelogin.admin LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
4	00:08:44	SELECT * FROM nodelogin.bikes LIMIT 0, 1000	12 row(s) returned	0.000 sec / 0.000 sec

5. User profile table



As taking into account the consideration of functional dependencies between different attributes, typically between a primary key and a non primary key. The tables used in our project contain only atomic values, proving them in 1NF form. And the repeated occurrences have also been removed as all non primary keys are directly dependent on the primary key, so it's in 2NF form. And as there is no partial dependency so it is in 3NF form.

SQL Code:

```
-- CREATE DATABASE IF NOT EXISTS `nodelogin` DEFAULT CHARACTER
SET utf8 COLLATE utf8_general_ci;
USE `nodelogin`;

-- CREATE TABLE IF NOT EXISTS `accounts` (
--   `id` int(11) NOT NULL AUTO_INCREMENT,
--   `username` varchar(50) NOT NULL,
--   `password` varchar(255) NOT NULL,
```

```
-- `email` varchar(100) NOT NULL,  
-- PRIMARY KEY (`id`)  
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;  
  
-- INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES  
(1, 'test', 'test', 'test@test.com');  
-- ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password  
BY 'kirti123#';  
-- CREATE TABLE IF NOT EXISTS `admin` (  
-- `id` int(11) NOT NULL AUTO_INCREMENT,  
-- `username` varchar(50) NOT NULL,  
-- `password` varchar(255) NOT NULL,  
-- `email` varchar(100) NOT NULL,  
-- PRIMARY KEY (`id`)  
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;  
  
-- INSERT INTO `admin` (`id`, `username`, `password`, `email`) VALUES (1,  
'kirti', 'kirti', 'kirti@kirti.com');  
  
-- CREATE TABLE IF NOT EXISTS `bikes` (  
-- `id` int(11) NOT NULL AUTO_INCREMENT,  
-- `username` varchar(50) NOT NULL,  
-- `password` varchar(255) NOT NULL,  
-- `email` varchar(100) NOT NULL,  
-- PRIMARY KEY (`id`)  
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;  
  
-- CREATE TABLE IF NOT EXISTS `bikes` (
```

```
-- `id` int(11) NOT NULL AUTO_INCREMENT,  
-- `name` varchar(50) NOT NULL,  
-- `mobile` int(20) NOT NULL,  
-- `address` varchar(100) NOT NULL,  
-- `license` varchar(100) NOT NULL,  
-- `state` varchar(100) NOT NULL,  
-- `area` varchar(100) NOT NULL,  
-- `email` varchar(100) NOT NULL,  
-- PRIMARY KEY (`id`)  
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

```
-- SELECT * FROM bikes where name="test";
```

```
-- CREATE TABLE IF NOT EXISTS `add_bikes` (  
-- `id` int(11) NOT NULL AUTO_INCREMENT,  
-- `name` varchar(50) NOT NULL,  
--  
-- `description` varchar(200) NOT NULL,  
-- `model` int(100) NOT NULL,  
-- `cost` int(100) NOT NULL,  
-- `features` varchar(200) NOT NULL,  
-- `file_data` MEDIUMBLOB NOT NULL,  
-- PRIMARY KEY (`id`)  
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

```
-- CREATE TABLE IF NOT EXISTS `user_profiles` (  
--  
-- `username` varchar(50) NOT NULL,
```

```

-- `email` varchar(200) NOT NULL,
-- `number` int(100) NOT NULL,
-- `license` int(100) NOT NULL,
-- `address` varchar(200) NOT NULL,
-- `state` varchar(200) NOT NULL,
-- 
-- 
-- PRIMARY KEY (`username`)
-- ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

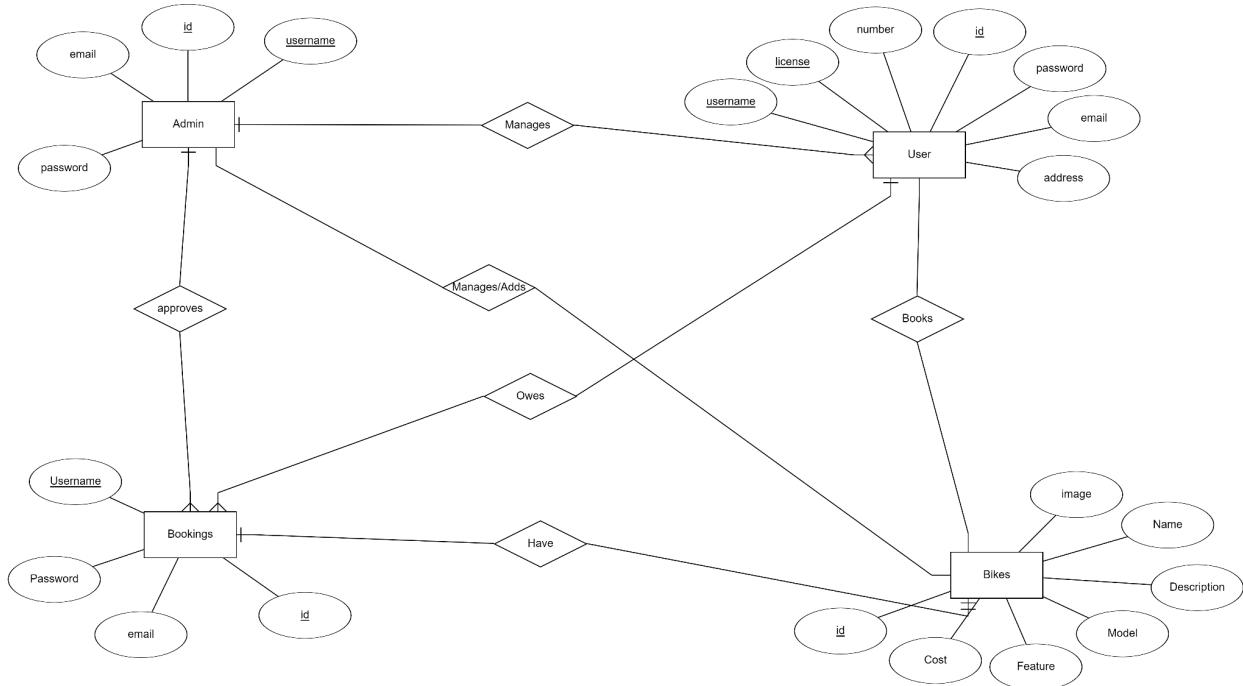
```

```

ALTER TABLE bikes
RENAME COLUMN state to model;

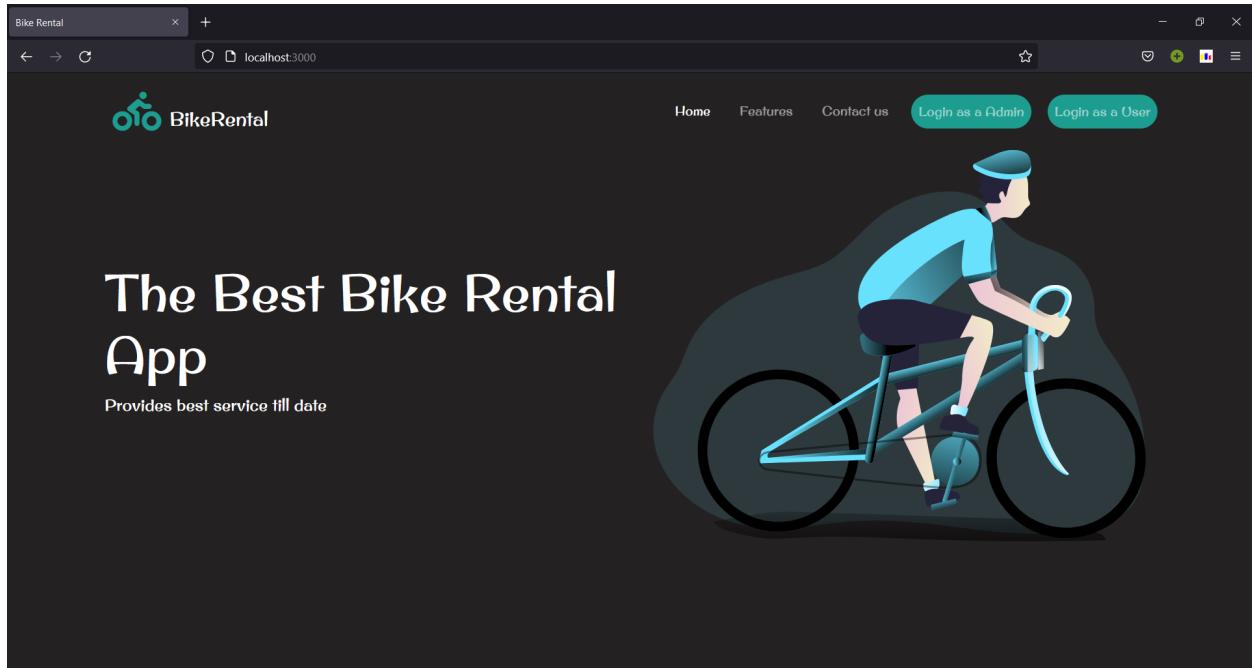
```

Entity-Relationship Diagram

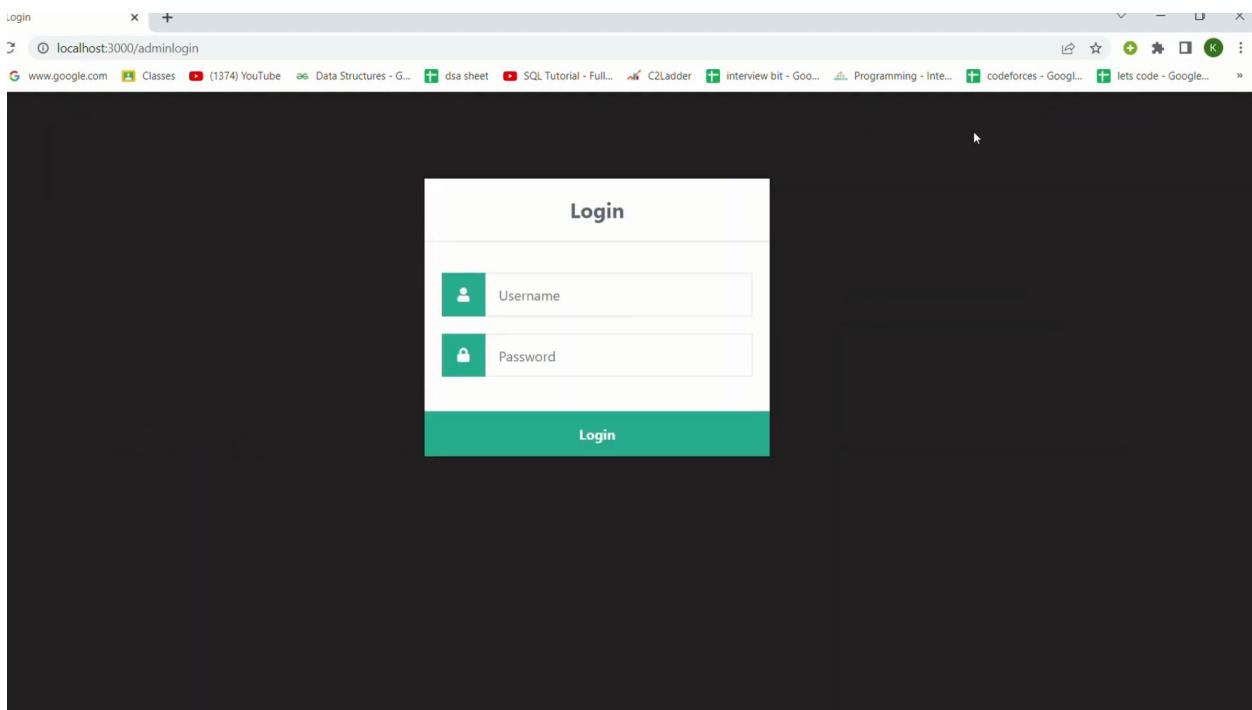


Screenshots

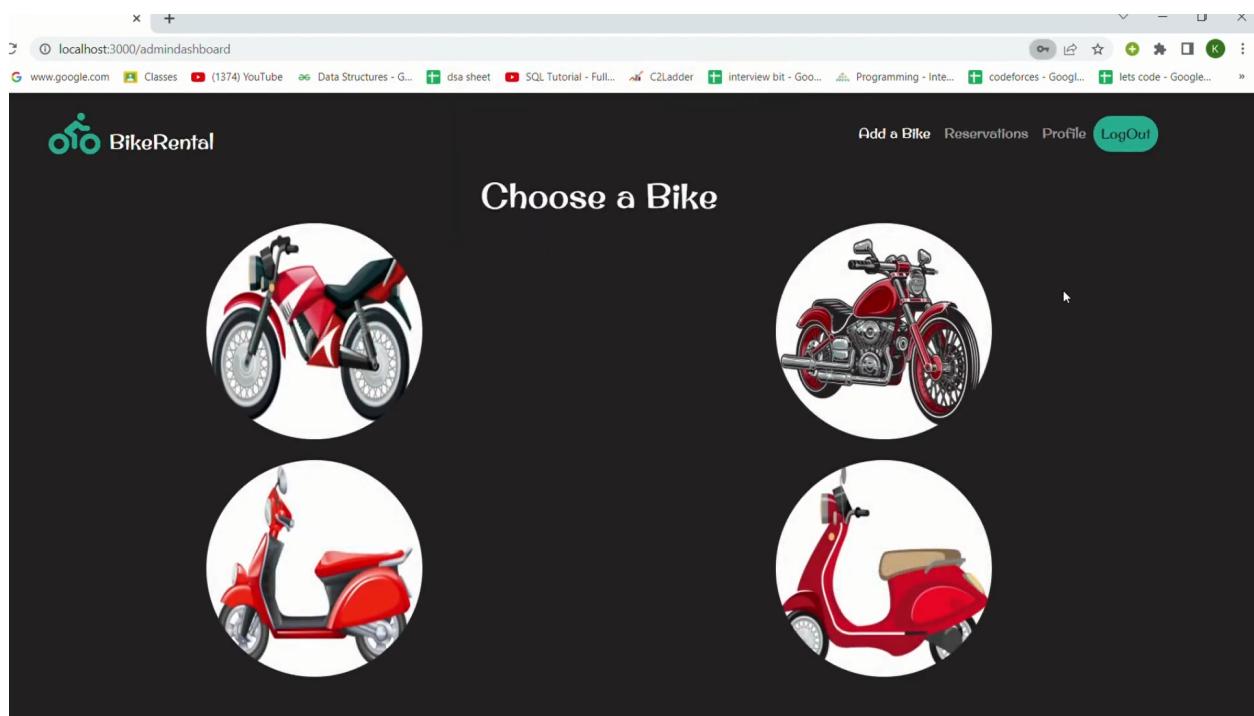
1. Home Page



2. Admin Login



3. Admin Dashboard



4. Available Bikes

The screenshot shows a web browser window with the URL `localhost:3000/addbikes`. The page title is "BikeRental". At the top right are links for "Add a bike", "Reservations", "Profile", and "LogOut". The main heading is "Add a Bike". Below it are three cards displaying bike details:

Bike Type	Model	Cost	Features
TVS activa	1990	8	dual disc
TVS activa	2013	67	braking system
honda scooter	1890	90	brake system
tvs	1234	67	dual

At the bottom right is a green "Add" button with a white plus sign icon.

5. Adding a Bike

Add Bike

name

desc

model

cost

features

Choose File No file chosen

Add

activa lucor

jnfjgkfg

2345

567

dual disc

Choose File No file chosen

Add

6. Added Bike

The screenshot shows a web browser window for the 'BikeRental' application at localhost:3000/addbikes. The page has a dark theme with white cards for each bike entry. Each card contains the bike's name, model, cost, and features.

Bike Model	Model	Cost	Features
TVS activa	1990	8	dual disc
TVS activa	1990	8	dual disc
honda scooter	1890	90	brake system
activa	2013	67	braking system
tvs	1234	67	dual
activa lucor	1234	67	braking system

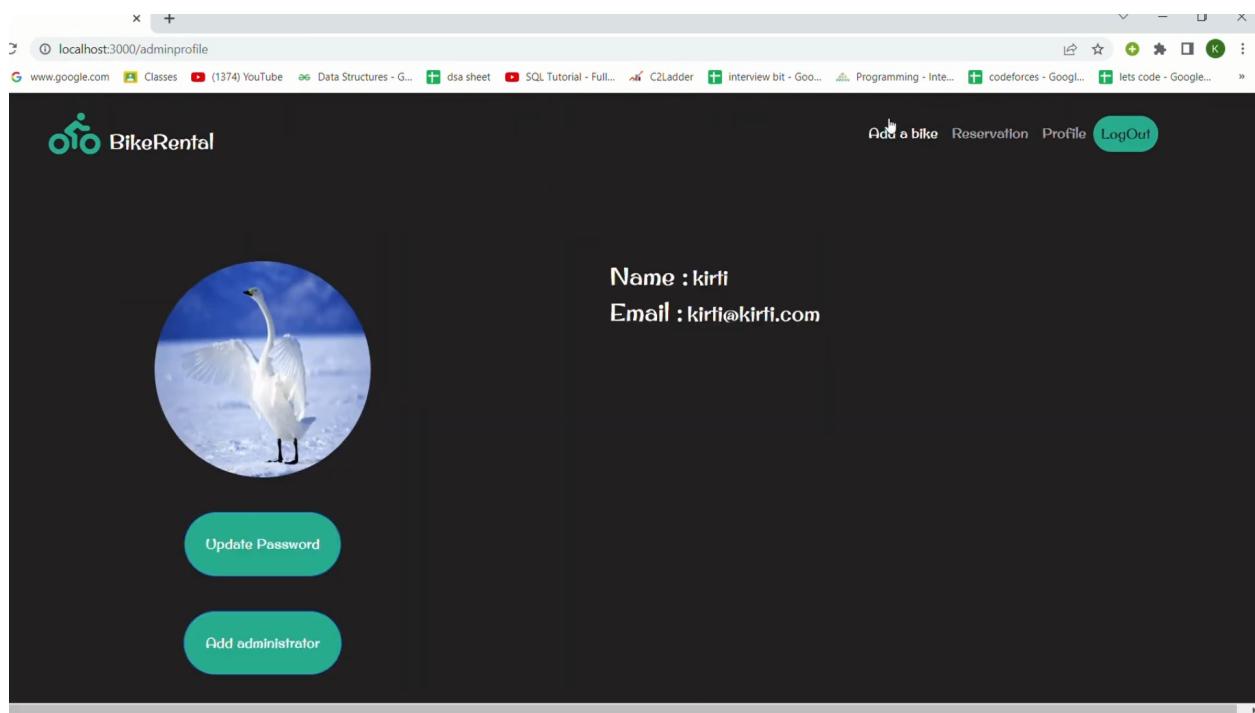
Add

7. List of all bookings available to admin

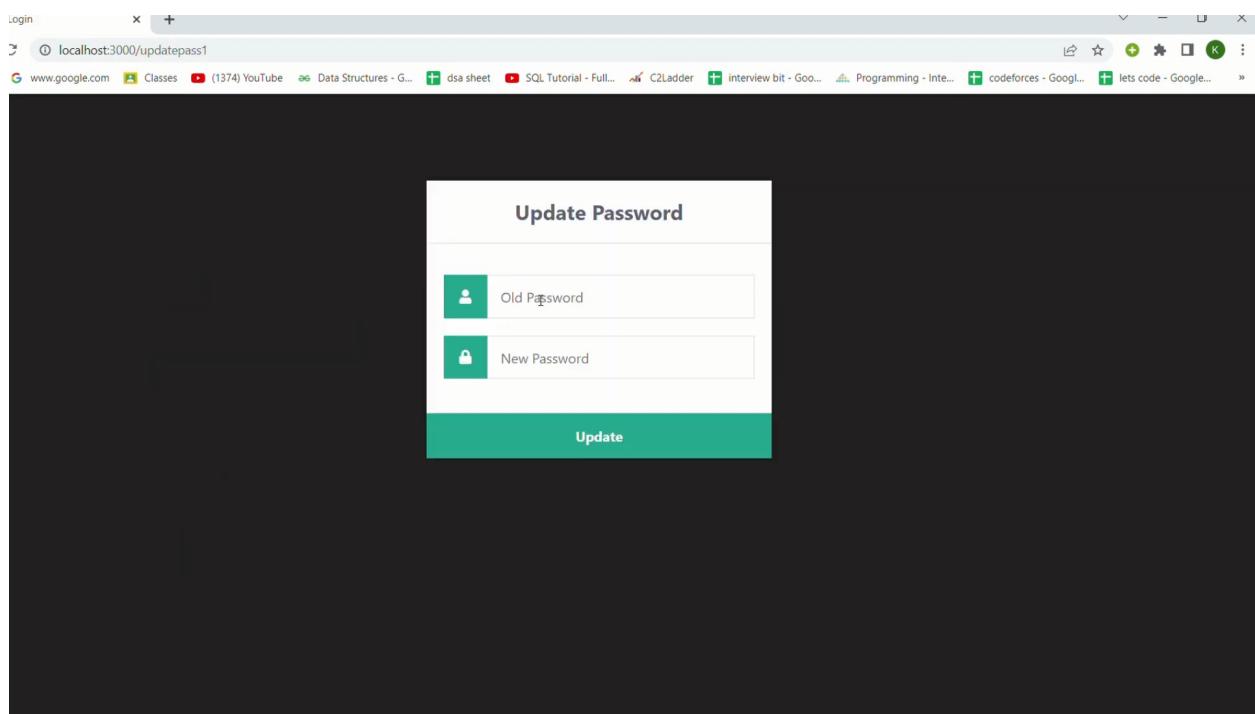
The screenshot shows a web browser window for the 'BikeRental' application at localhost:3000/reservation. The page displays a table of reservations with columns for Name, Mobile, Address, License, Model, Area, and Email.

#	Name	Mobile	Address	License	Model	Area	Email
1	hh	566	frdr	dfdfdfd	def	dfdfd	dfgfg@hh
2	kirli	1234	bti	lghy	ghjnm	qwert	kirli@kirli
3	kirli	1234	bti	lghy	ghjnm	qwert	kirli@kirli
4	kajal	5677	ludh	hui	pun	pun	kajal@kajal
5	kajal	5677	ludh	hui	pun	pun	kajal@kajal
6	krishna	8956	guj	lic	guj	guj	krish@krish
7	ashita	7890	luc	luc	luc	luc	ashita@ashita
8	shruti	1244	nbager	gu67	pun	pun	shruti@shruti
9	test	1325	fgfgrg	cfc	df	dd	ddodd
10	tsd	122	1223vfgv	12234	jdfuh	jdfj	hdh@njd
11	sushil	143143	hdhg	132	lp009	jhbdfh	dvehdh

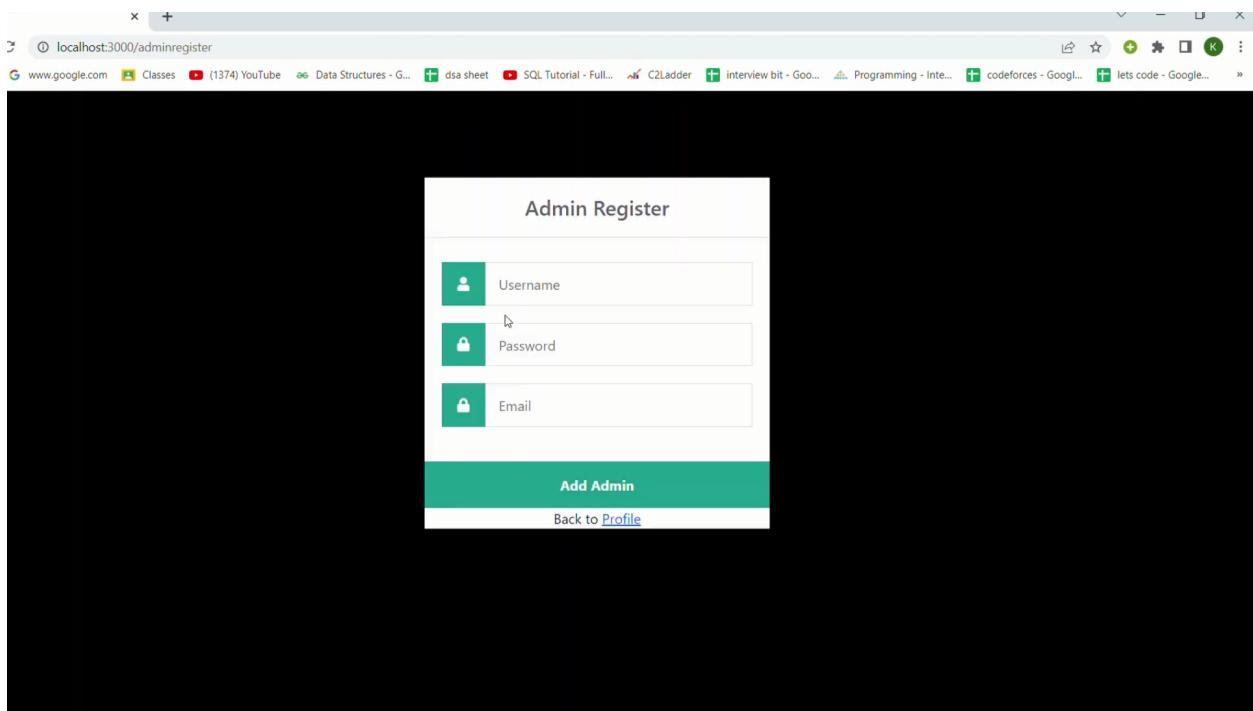
8. Admin Profile



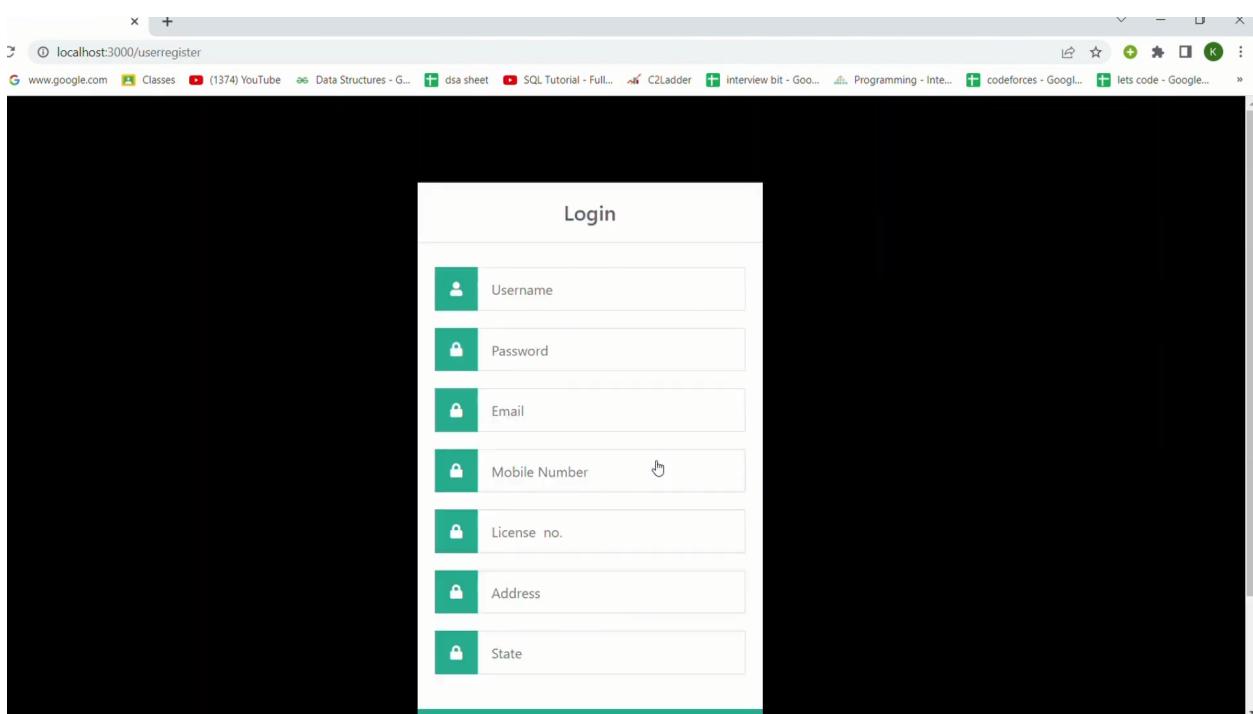
9. Password Update for Admin



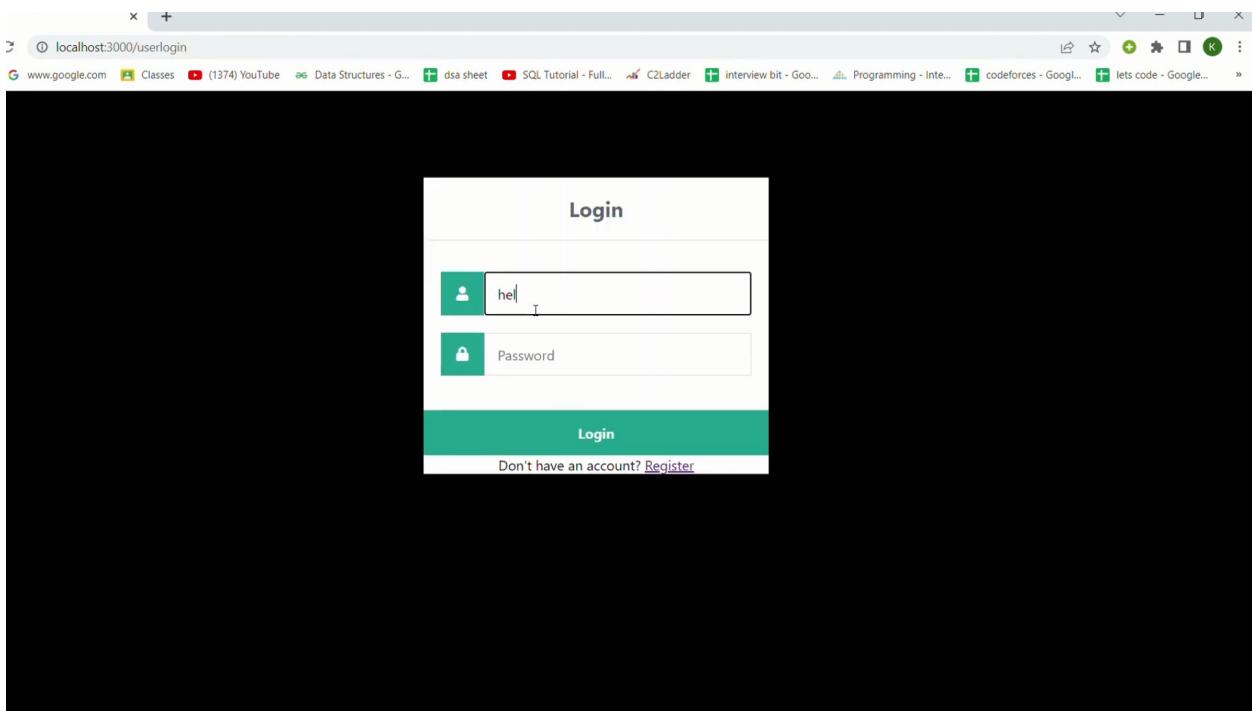
10. Admin Registration



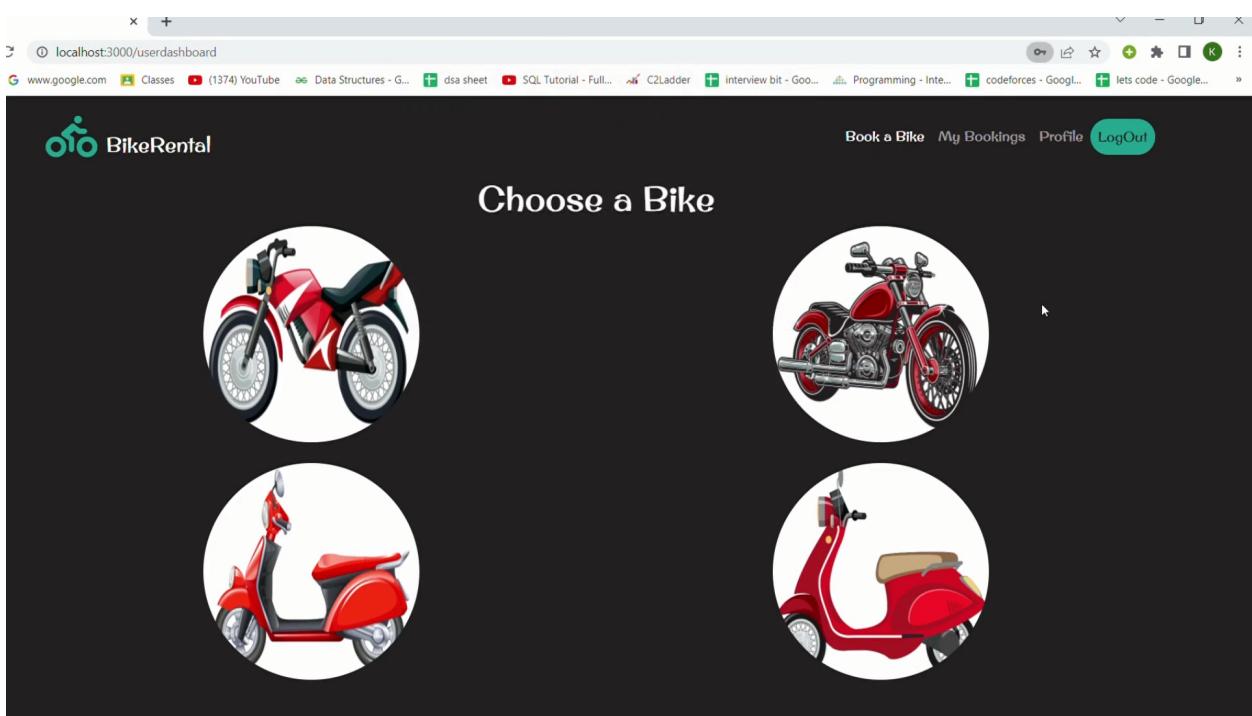
11. User Registration



12. User Login



13. User Dashboard



14. List of bikes available for rent

The screenshot shows a web browser window with the title "Bike" and the URL "localhost:3000/bikes". The page displays five cards, each representing a different bike model. Each card contains the bike's name, a placeholder image, its details (Model, Cost, Features), and a "Book Now" button.

Bike Name	Image Placeholder	Model	Cost	Features	Action
Isar	[Placeholder]	2009	24	brakes: brake system	Book Now
activa	[Placeholder]	2013	67	braking system	Book Now
tvs	[Placeholder]	1234	67	dual	Book Now
activa lucor	[Placeholder]	2345	567	dual disc	Book Now

15. User buying a bike

The screenshot shows a web browser window with the title "localhost:3000/buybike". A modal dialog box is open, titled "Buy a bike". It contains a form with six fields, each with a lock icon and a placeholder text:

- hello_1
- Mobile
- address
- license
- model
- area
- Email

16. Bike bookings of a user

A screenshot of a web browser window titled "localhost:3000/mybooking". The page displays a table titled "My Bookings" with one row of data. The columns are labeled: #, Name, Mobile, Address, License, Model, Area, and Email. The data row shows:

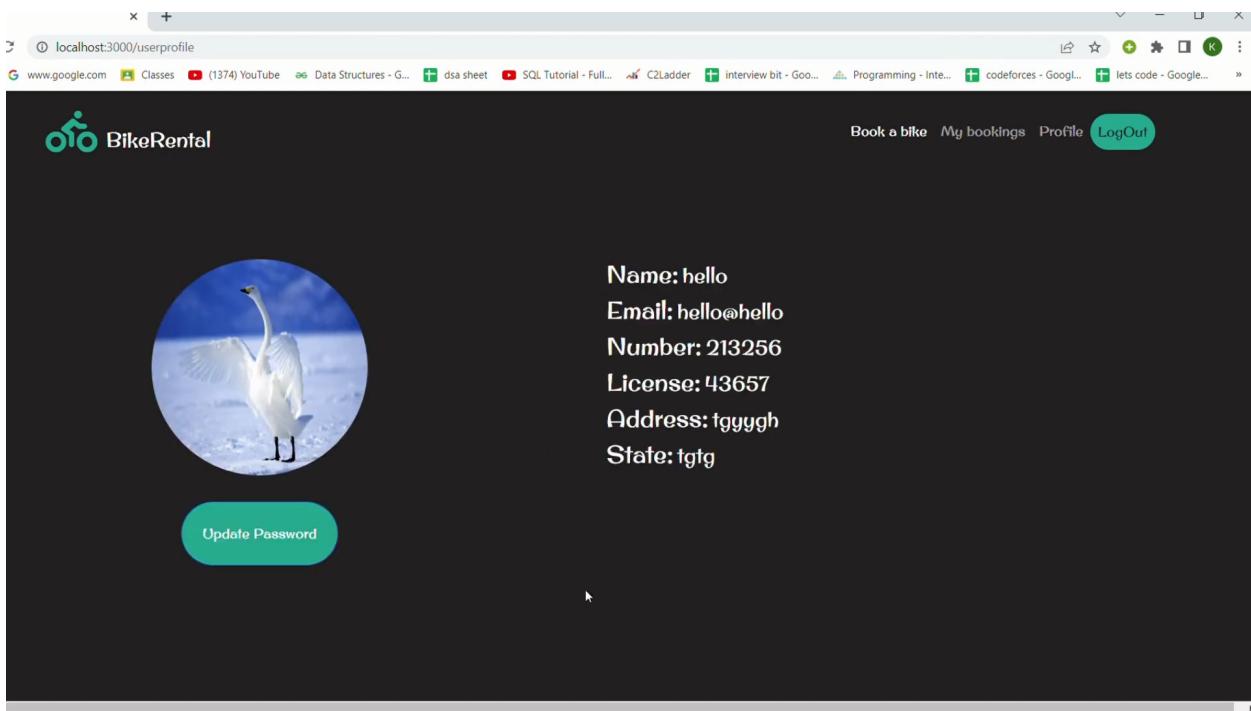
#	Name	Mobile	Address	License	Model	Area	Email
1	hello	345	difhgfh	5468	2345	hutyu	rttgejjf

Below the table are two buttons: "Cancel Booking" and "Print Invoice". The "Cancel Booking" button is highlighted with a mouse cursor.

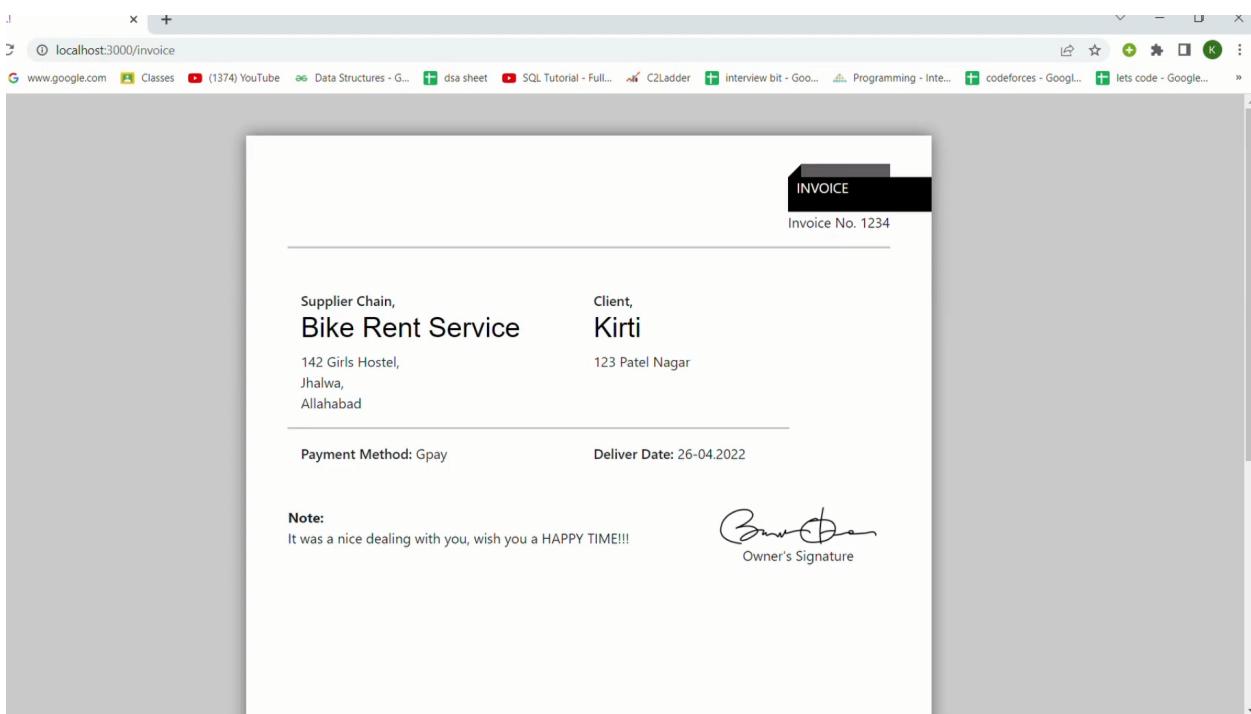
17. Cancellation of a booking

A screenshot of a web browser window titled "localhost:3000/mybooking". The page displays a table titled "My Bookings" with no data found. The message "No Data Found" is centered below the table.

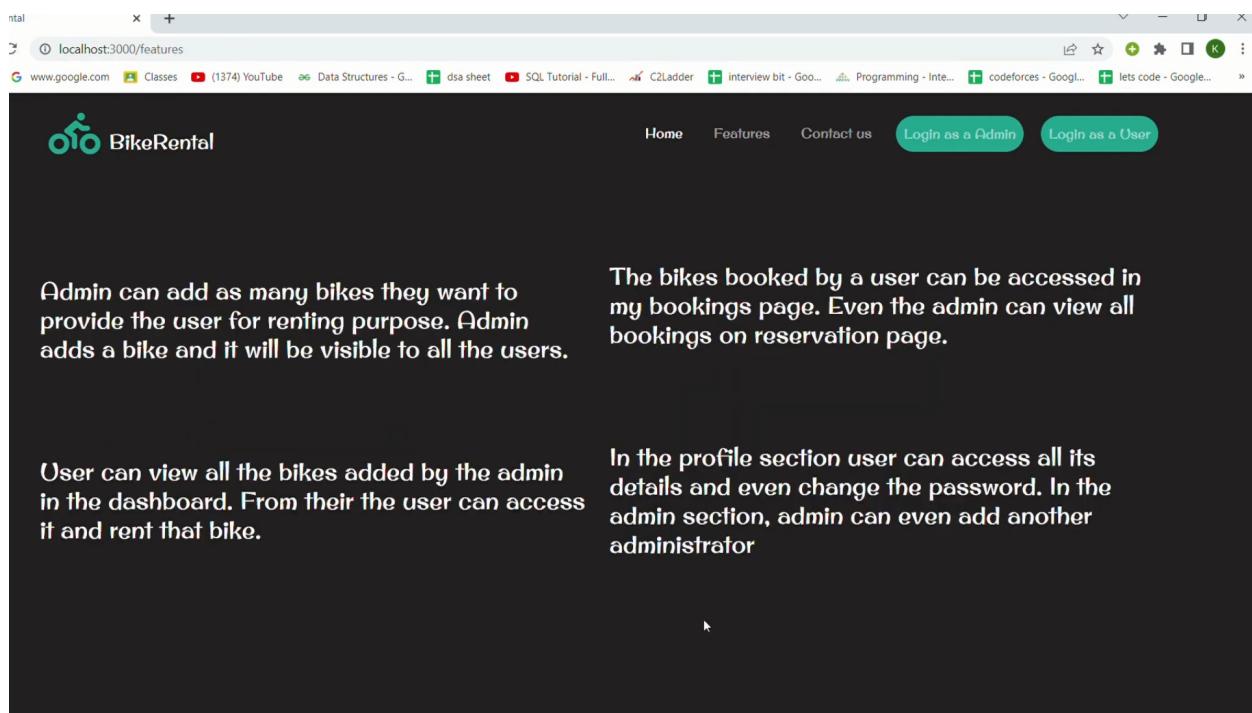
18. User profile page



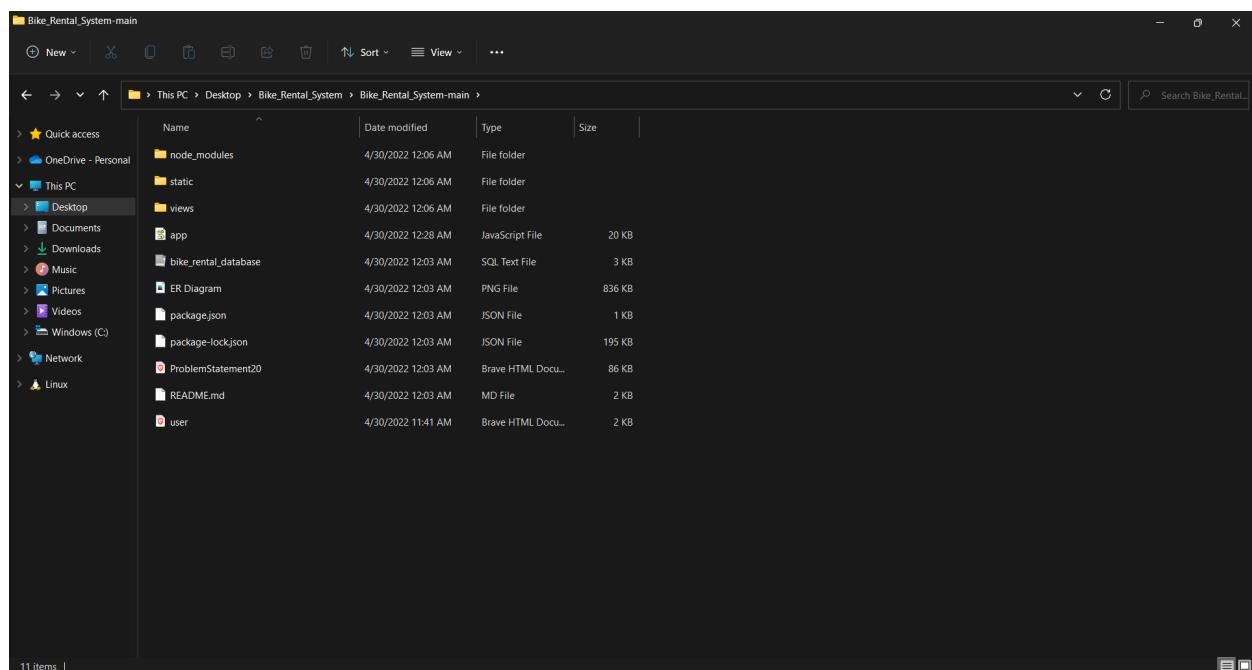
19. Invoice of the booking



20. Features page



21. Pdf of bill by the name user.pdf will be downloaded in the same directory



Scope

- It functions as a system for storing, processing and handling customer information and the information of the most valuable assets which are the bikes for the company.
- The web administrator is the one who has full control and authority to control the privilege and update the proposed system from time-to-time. In addition, they are also required to ensure that the different types of user access via their permitting level.
- The admin are restricted to view, update or delete any information about the registered members' information such as car reservation history. They are allowed to view and update details of customers as well as of themselves.
- In addition, the registered members are allowed to update their personal information if necessary with their own valid username and password through the proposed system. Each of the updated personal information will be stored and kept in the database and can be categorized as confidential information.
- The reports are generated by the proposed system based on the management requirement such as print and view the invoice.
- The management is able to improve and provide the better services from time-to-time or make it as future references.
- The proposed system provides a two tier login concept. It means that different users will login with different permission or privilege. Therefore the users are able to access different functionality of the proposed system.