Manual Testing Assessment

Online Taxi booking Application

What	How
What are the features need to implement? **CAB:** **1.What are the attributes needed for the cab table?** **Ans:** Vehicle ID, Vehicle no, Rate/Km, Vehicle type and Vehicle capacity. **2. What are the functions can perform in cab table?** **Ans:** Insertion of vehicle details by admin. **3. What is the database use for this project?** **Ans:** MySQL is preferred for this project**.** **4. What are the benefits of the application to cabs?** **Ans:** It is useful in the way that cabs can access this application remotely. **5. What is the procedure to register for this application?** **Ans:** Cabs can register to this application with valid vehicle proofs and owners valid Id proofs.	**How are the features to be implemented?** **CAB:** **1.How are the attributes helpful for the cab table?** **Method 1: By mentioning vehicle id as primary key.** **Method 2:** By mentioning vehicle no as primary key. **2. How functions will be useful for cab table?** **Method 1:** Using insertion and deletion functions of vehicle details is enough. **Method 2:** Using insertion, deletion, and updating functions of vehicle details is enough**.** **Method 3: Using the insertion and search function of vehicle details will be more useful.** **3. How and which database is useful?** **Method 1:** H2 data can be preferred. **Method 2: MySQL database can be preferred.** **Method 3:** Oracle database can be preferred. **4. How is the application beneficial to the cabs?** **Method 1:** It is beneficial to cabs in the way that they can access it remotely as web application. **Method 2:** It is beneficial to cab in the wat that they can access it remotely as mobile application. **Method 3: It is beneficial to cabs in the way that they can access it remotely as both web and mobile application.** **5. How is the procedure to register useful?** **Method 1:** Registering with valid proof of vehicle details. **Method 2: Registering with valid proofs of owner and vehicle details.**

PASSENGER: **1.What are the attributes needed for the ticket table?** **Ans:** Passenger Id, Passenger, pickup point, drop point and Vehicle Id. **2.What are the functions can perform in bus table?** **Ans:** Insertion of passenger details and can search the vehicle details by passenger. **3. What are the benefits of this application to users?** **Ans:** Passengers or users can access this application remotely and they can book their travel, update their travel and cancel their travel remotely. **4. What is the procedure to register for this application?** **Ans:** Passenger can register to this application with any one of valid Id proof.	**PASSENGER:** **1.How are the attributes helpful for the passenger table?** **Method 1: By mentioning Passenger id as primary key.** **Method 2:** By mentioning Passenger name as primary key. **2. How functions will be useful for passenger table?** **Method 1:** Using insertion and deletion functions of Passenger details is enough. **Method 2:** Using insertion, deletion, and updating functions of bus details is enough**.** **Method 3: Using the insertion function of passenger details will be more useful.** **3. How is the application beneficial to the passengers?** **Method 1:** It is beneficial to passengers in the wat that they can access it remotely as web application. **Method 2:** It is beneficial to passengers in the wat that they can access it remotely as mobile application. **Method 3: It is beneficial to passengers in the wat that they can access it remotely as both web and mobile application.** **4. How is the procedure to register useful?** **Method 1:** Registering with two valid proofs of passenger. **Method 2: Registering with one valid proof of passenger.**

BOOKING DETAILS: **1.What are the attributes needed for the booking table?** **Ans:** Booking Id, booking status, total amount, passenger Id and Vehicle Id. **2.What are the functions can perform in bus table?** **Ans:** Insertion of booking details.	**BOOKING DETAILS:** **1.How are the attributes helpful for the passenger table?** **Method 1: By mentioning Booking id as primary key.** **Method 2:** By mentioning Booking name as primary key. **2. How the functions be performed on the booking table?** **Method 1:** Using insertion and deletion functions of Passenger details is enough. **Method 2:** Using insertion, deletion, and updating functions of bus details is enough**.** **Method 3: Using the insertion function of booking details will be more useful.**

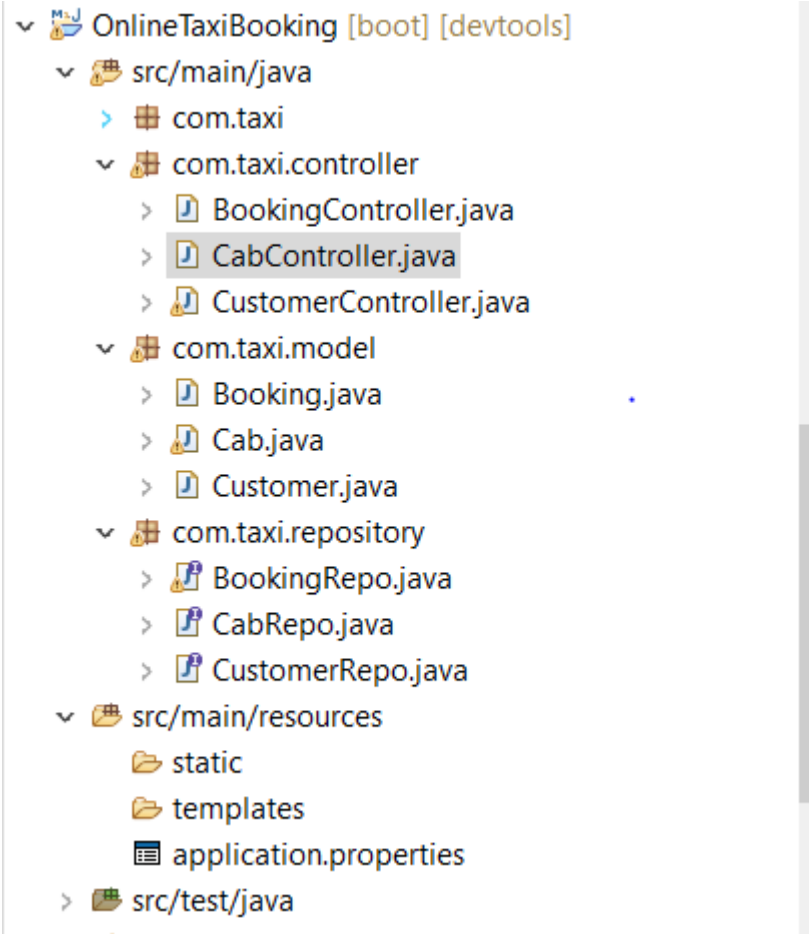
Why	Why not
Why are attributes needed for the bus table? **CAB:** **1.Why are the attributes helpful for the cab table?** **Method 1: By mentioning vehicle id as primary key.** **It is easy to access the vehicle details.** **2. Why functions will be useful for cab table?** **Method 3: Using the insertion and search function of vehicle details will be more useful.** **Because giving all these functions to users, it will be useful for them, and it will make the application user friendly.** **3. Why and which database is useful?** **Method 2: MySQL database can be preferred.** **Because MySQL is a free licensed database.** **4. Why is the application beneficial to the cabs?** **Method 3: It is beneficial to cabs in the way that they can access it remotely as both web and mobile application.** **Because it is a web and mobile application, they access it with system or laptop or with mobiles.** **5. Why is the procedure to register useful?** **Method 2: Registering with valid proofs of owner and vehicle details.** **Because it is essential to validate the user detail.**	**Why aren't other attributes needed for the bus table?** **CAB:** **1. Why aren't the attributes helpful for the cab table?** **Method 2:** By mentioning vehicle no as primary key. **It is not convenient to access the data.** **2. Why aren't these functions will not be useful for cab table?** **Method 1:** Using insertion and deletion functions of vehicle details is enough. **Method 2:** Using insertion, deletion, and updating functions of vehicle details is enough**.** **Because these functions are not necessary for the application.** **3. Why aren't these databases is not useful?** **Method 1:** H2 data can be preferred. **Method 3:** Oracle database can be preferred. **Because Oracle is paid licensed and H2 is not much more efficient than MySQL.** **4. Why aren't this feature is not beneficial to the cabs?** **Method 1:** It is beneficial to cabs in the way that they can access it remotely as web application. **Method 2:** It is beneficial to cab in the wat that they can access it remotely as mobile application. **Because it is not accessible in multiple ways or devices.** **5. Why isn't this procedure to register is not useful?** **Method 1:** Registering with valid proof of vehicle details. **Because this detail is not much enough to validate about the details and background of users.**

PASSENGER: **1. Why are the attributes helpful for the passenger table?** **Method 1: By mentioning Passenger id as primary key.** **It is easy to access the passenger details.** **2. Why will the functions be useful for passenger table?** **Method 3: Using the insertion function of passenger details will be more useful.** **Because giving all these functions to users, it will be useful for them, and it will make the application user friendly.** **3. Why is the application beneficial to the passengers?** **Method 3: It is beneficial to passengers in the wat that they can access it remotely as both web and mobile application.** **Because it is a web and mobile application, they access it with system or laptop or with mobiles.** **4. Why is the procedure to register useful?** **Method 2: Registering with one valid proof of passenger.** **Because it is essential to validate the user detail.**	**PASSENGER:** **1.Why are these attributes not helpful for the passenger table?** **Method 2:** By mentioning Passenger name as primary key. **It is not easy to access the passenger details.** **2. Why are these functions not useful for the passenger table?** **Method 1:** Using insertion and deletion functions of Passenger details is enough. **Method 2:** Using insertion, deletion, and updating functions of bus passengers is enough**.** **Because these functions are not necessary for the application.** **3. Why is this feature not beneficial to the passengers?** **Method 1:** It is beneficial to passengers in the wat that they can access it remotely as web application. **Method 2:** It is beneficial to passengers in the wat that they can access it remotely as mobile application. **Because it is not accessible in multiple ways or devices.** **4. Why is the procedure to register not useful?** **Method 1:** Registering with two valid proofs of passenger. **Because this detail is not much enough to validate about the details and background of users.**

BOOKING DETAILS: **1. Why are the attributes helpful for the booking table?** **Method 1: By mentioning Booking id as primary key.** **It is easy to access the booking details.** **2. How the functions be performed on the booking table?** **Method 3: Using the insertion function of booking details will be more useful.** **Because giving all these functions to users, it will be useful for them, and it will make the application user friendly.**	**BOOKING DETAILS:** **1. Why are these attributes not helpful for the booking table?** **Method 2:** By mentioning Booking name as primary key. **It is not easy to access the booking details.** **2. How the functions be performed on the booking table?** **Method 1:** Using insertion and deletion functions of Passenger details is enough. **Method 2:** Using insertion, deletion, and updating functions of bus details is enough**.** **Because these functions are not necessary for the application .**

Spring boot code

Folder structure



Entity package:

Booking.java

```
package com.taxi.model;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinTable;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@Entity

@Table

@NoArgsConstructor
```

```
@AllArgsConstructor

@Getter

@Setter

@ToString

public class Booking {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int bookingId;

private String pickupLocation;

private String dropLocation;

private String bookingStatus;

@ManyToOne(targetEntity = Customer.class, cascade = CascadeType.MERGE)

private Customer customer;

@ManyToOne(targetEntity = Cab.class,cascade = CascadeType.MERGE)

@JoinTable(name = "cab_booking", joinColumns = @JoinColumn(name =
"booking_Id"), inverseJoinColumns = @JoinColumn(name = "cab_Id"))

private Cab cab;

}
```

Customer.java

```
package com.taxi.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;
```

```java
import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

@Entity

@Table

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int customerId;

private String customerName;

private String phnNo;

}
```

Cab.java

```java
package com.taxi.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.ManyToMany;

import jakarta.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

@Entity

@Table

public class Cab {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private int cabId;

private String cabNo;

private String cabType;

private int cabCapacity;

private int ratePerkm;

}
```

Repository package:

Booking repoitory.java

```
package com.taxi.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.taxi.model.Booking;

public interface BookingRepo extends JpaRepository<Booking, Integer>{


Booking findByBookingId(int bookingId);





}
```

CustomerRepository.java

```
package com.taxi.repository;

import java.util.List;
```

```java
import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.taxi.model.Customer;

public interface CustomerRepo extends JpaRepository<Customer,
Integer>{


Customer findByCustomerId(int customerId);


@Query("select customerId from Customer")

public List<Integer> getCustomerId();

}
```

CabRepository.java

```java
package com.taxi.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.ManyToMany;

import jakarta.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;
```

```java
import lombok.Setter;

import lombok.ToString;

@NoArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

@Entity

@Table

public class Cab {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int cabId;

private String cabNo;

private String cabType;

private int cabCapacity;

private int ratePerkm;

}
```

Controller package:

BookingController.java

```java
package com.taxi.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import com.taxi.model.Booking;
import com.taxi.repository.BookingRepo;
@RestController
@CrossOrigin(" http://localhost:3000/")
public class BookingController {
@Autowired
private BookingRepo repo;
@PostMapping("/addBooking")
public Booking addBooking(@RequestBody Booking booking) {
return repo.saveAndFlush(booking);
}
@GetMapping("/getAllbooking")
public List<Booking> findAllBooking() {
return repo.findAll();
}
@GetMapping("/getBookingById/{bookingId}")
Booking findBusId(@PathVariable("bookingId") int bookingId) {
```

```
return repo.findByBookingId(bookingId);
}

}
```

CustomerController.java

```
package com.taxi.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
import com.taxi.model.Cab;
import com.taxi.model.Customer;
import com.taxi.repository.CustomerRepo;
@RestController
@CrossOrigin(" http://localhost:3000/")
public class CustomerController {

@Autowired
private CustomerRepo repo;

@GetMapping("/allCustomers")
```

```
List<Customer> getAllcustomers(){
return repo.findAll();
}

@GetMapping("/getAllCustomerId")
List<Integer> findAllCustomerIds() {
return repo.getCustomerId();
}
@GetMapping("/getCustById/{customerId}")
Customer findCabId(@PathVariable("customerId") int customerId) {
return repo.findByCustomerId(customerId);
}
}
```

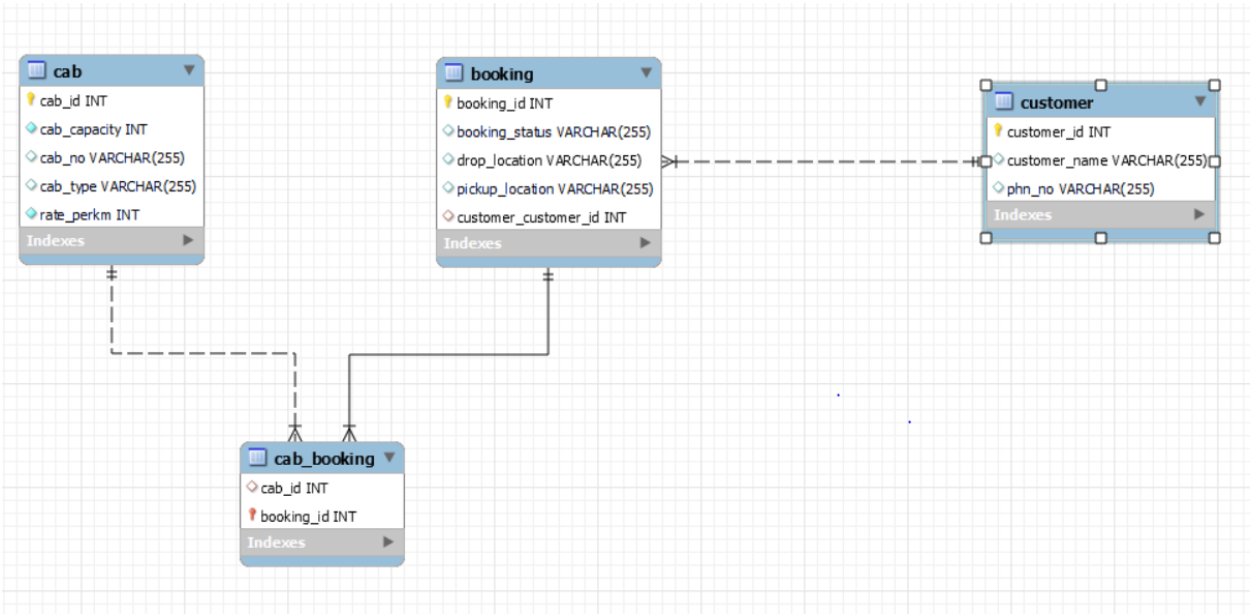
CabController.java

```
package com.taxi.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

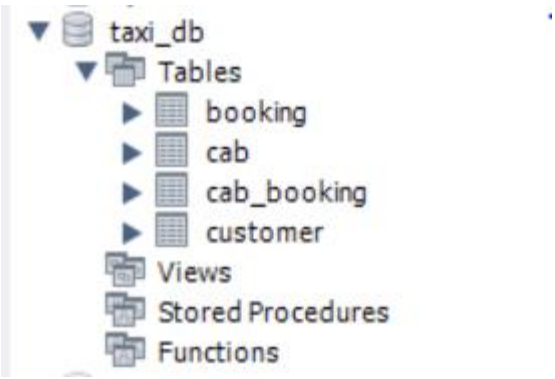
```java
import com.taxi.model.Cab;

import com.taxi.repository.CabRepo;

@RestController

@CrossOrigin(" http://localhost:3000/")

public class CabController {


@Autowired

private CabRepo repo;


@PostMapping("/addCab")

public Cab addcab(@RequestBody Cab cab) {

return repo.save(cab);

}


@GetMapping("/allCabs")

List<Cab> getAllCab() {

return repo.findAll();

}


@GetMapping("/getCabById/{cabId}")

Cab findCabId(@PathVariable("cabId") int cabId) {

return repo.findByCabId(cabId);

}
```

```
@GetMapping("/getAllCabId")

List<Integer> findAllCabIds() {

return repo.getCabId();

}



}
```

Reverse Enginering for database



Database Structure

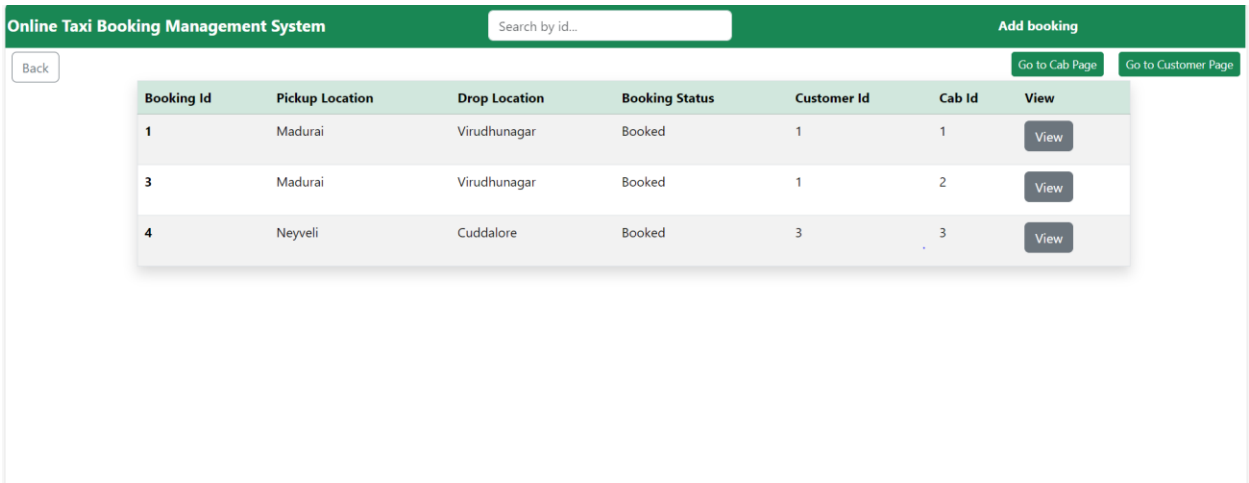


React js Frontend

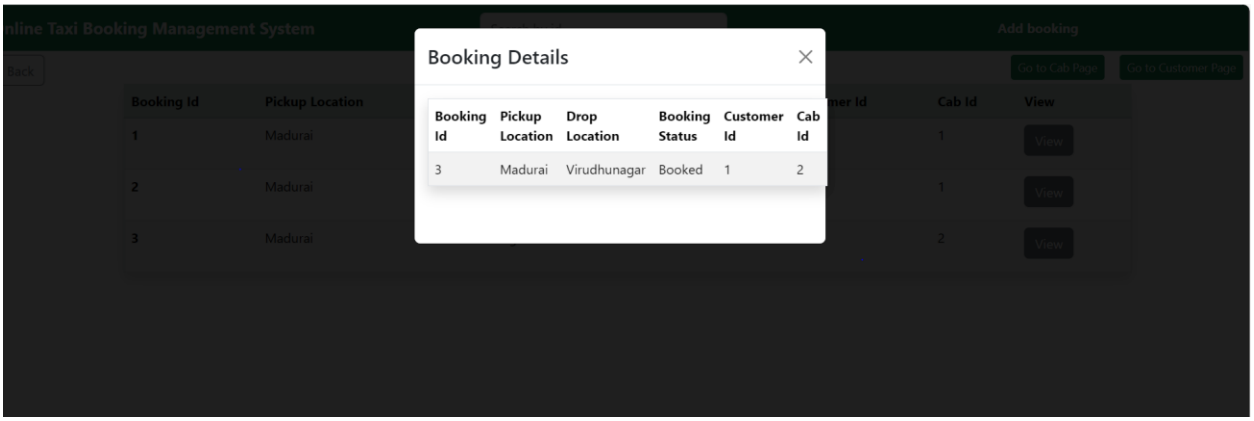
Booking table view page

By clicking on to the customer page button, it will redirect to the customer home page.

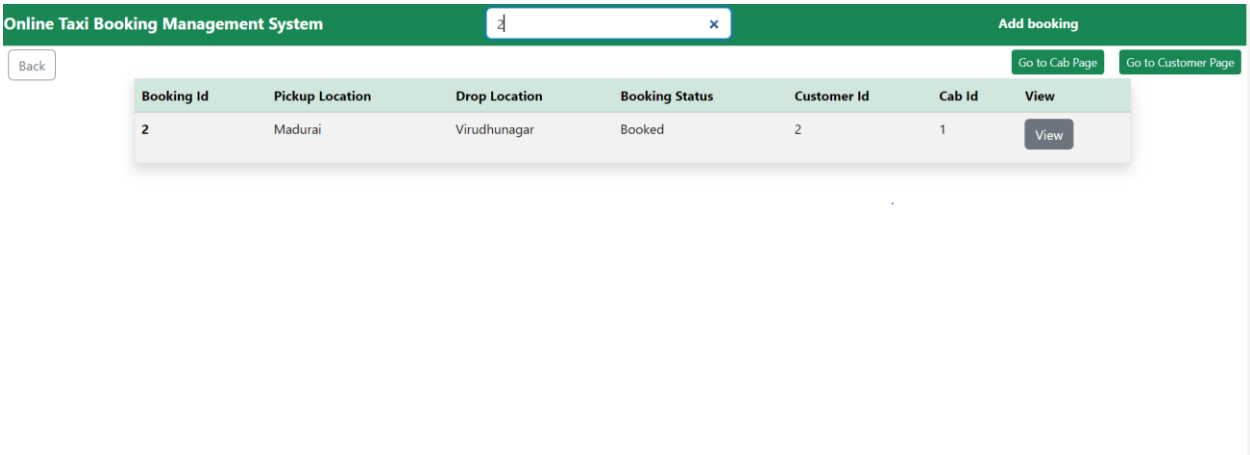
By clicking on to the cab page button, it will redirect to the cab home page.



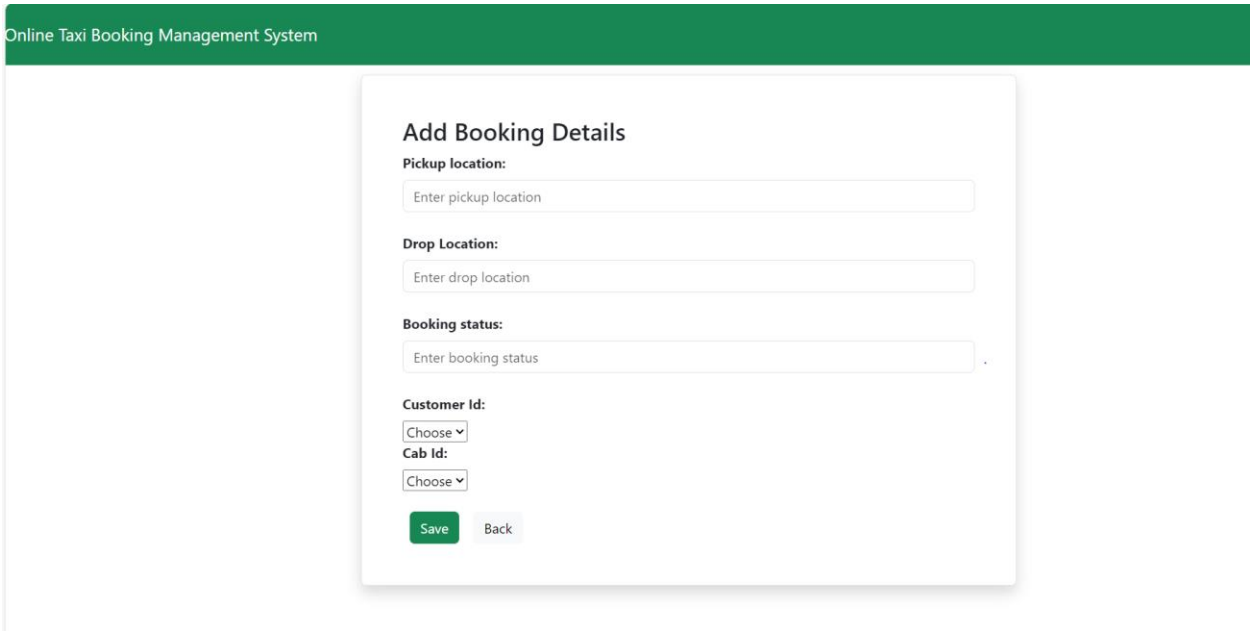
By clicking the view button, we can see the details of the particular booking detail.



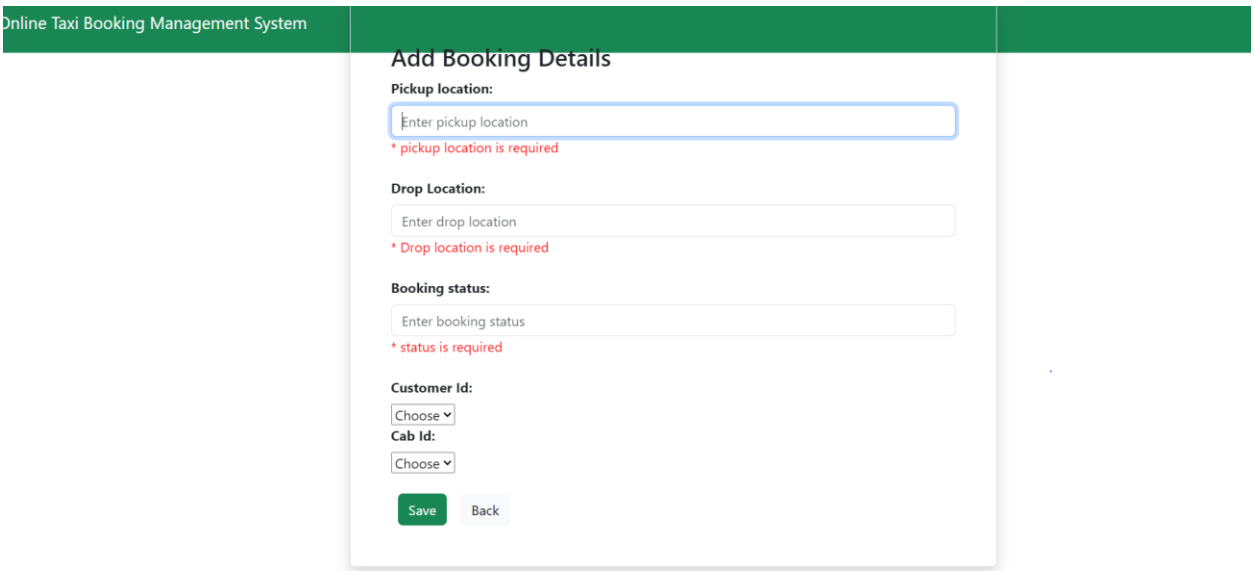
Through search bar, we can search booking by id



By clicking on add booking link, we can book a taxi by giving required details.



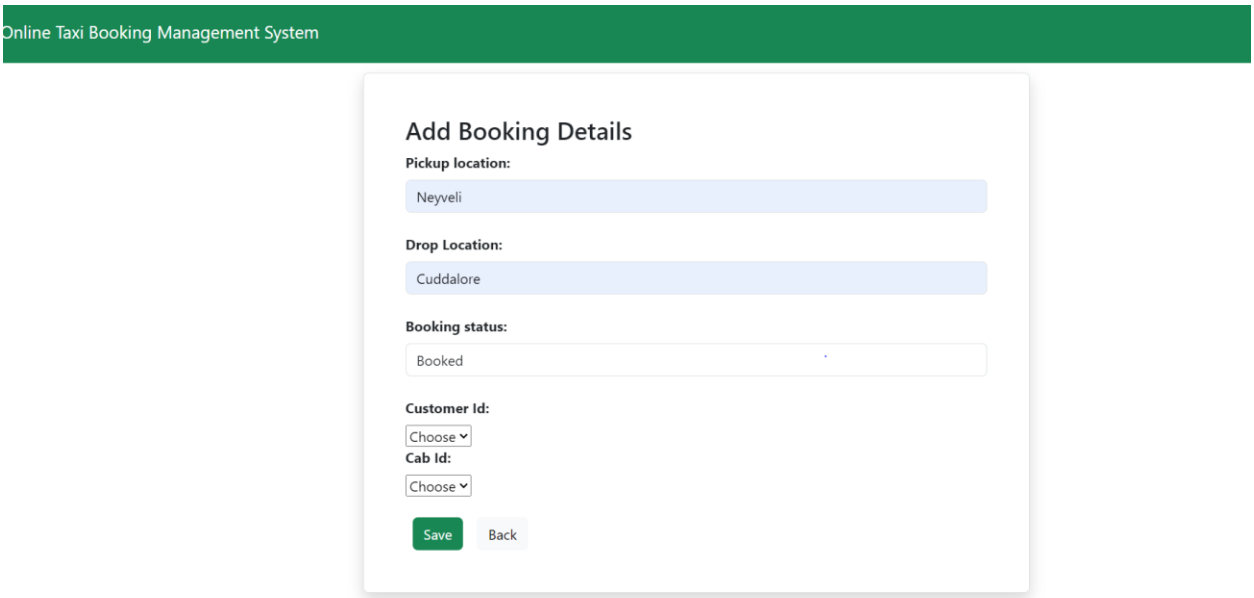
Here, a simple validation have done.



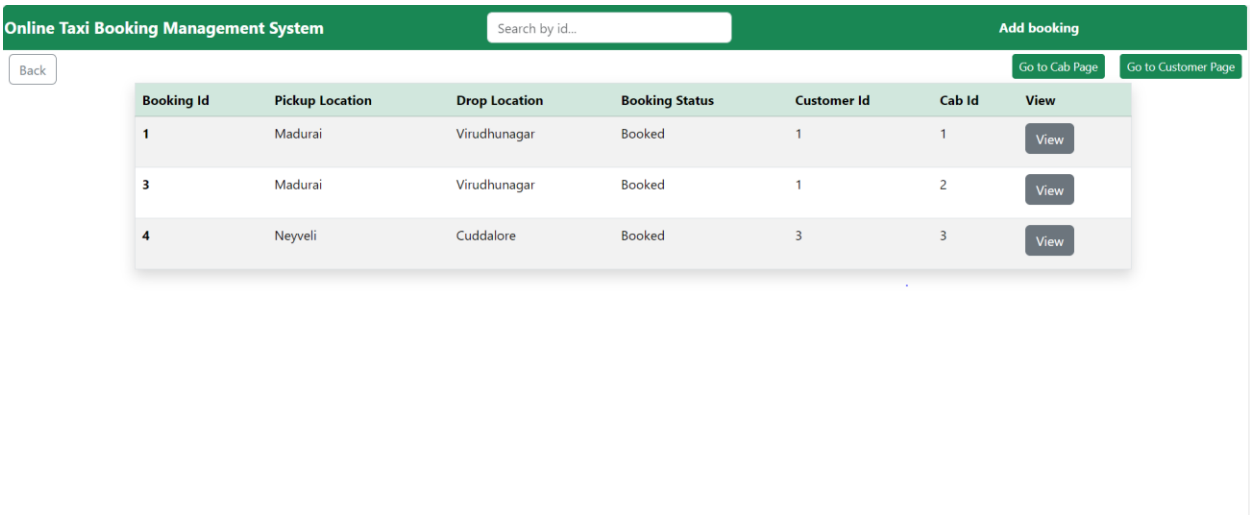
Here, we cannot book an already booked taxi.

One customer can book many taxi's.

Here, taxi is booked by giving details.



Here data is added and it is reflected in view page.

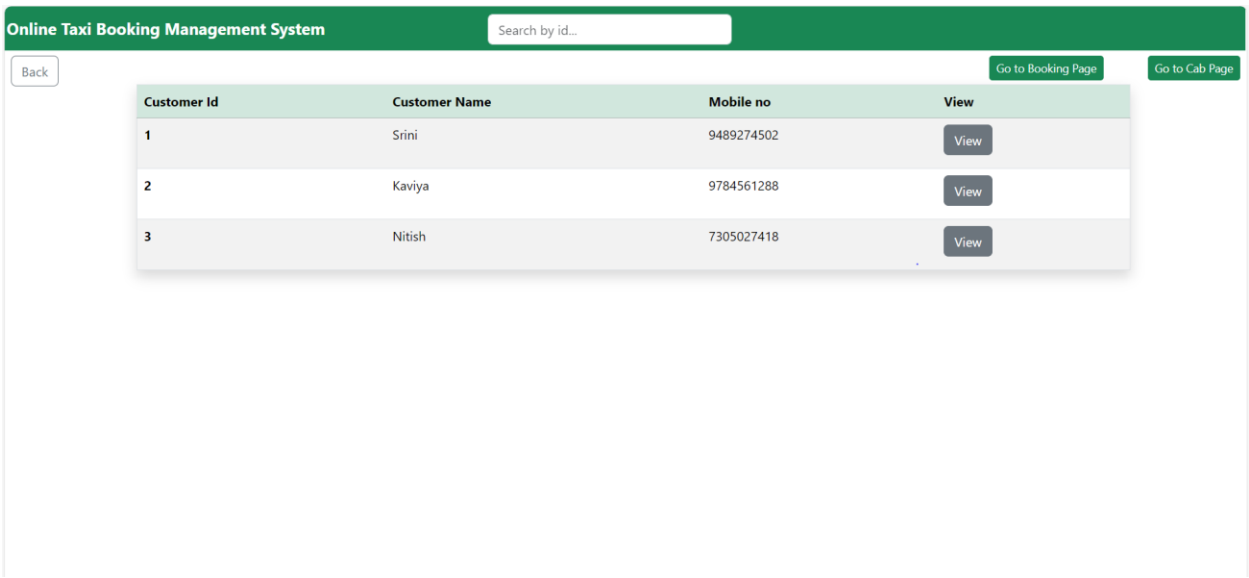


Customer details view page

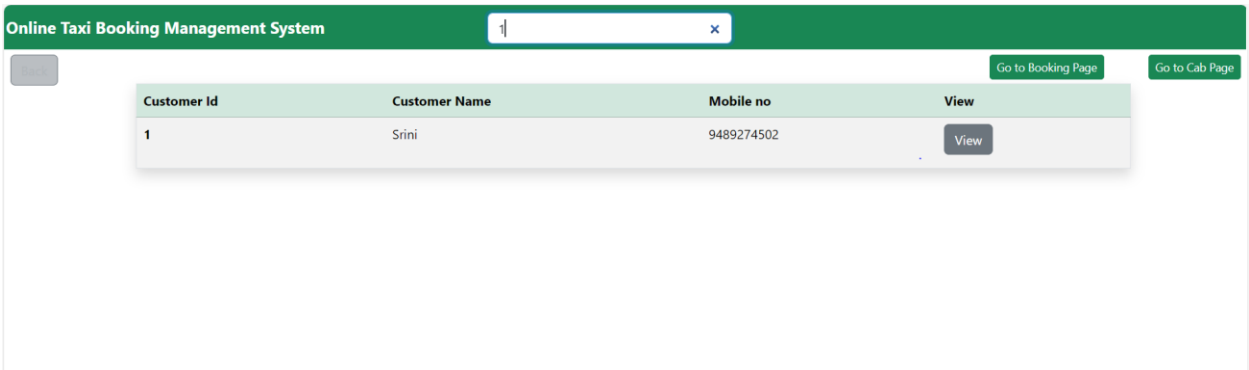
By clicking on to the back button, it will redirect to the booking home page.

By clicking on to the booking page button, it will redirect to the booking home page.

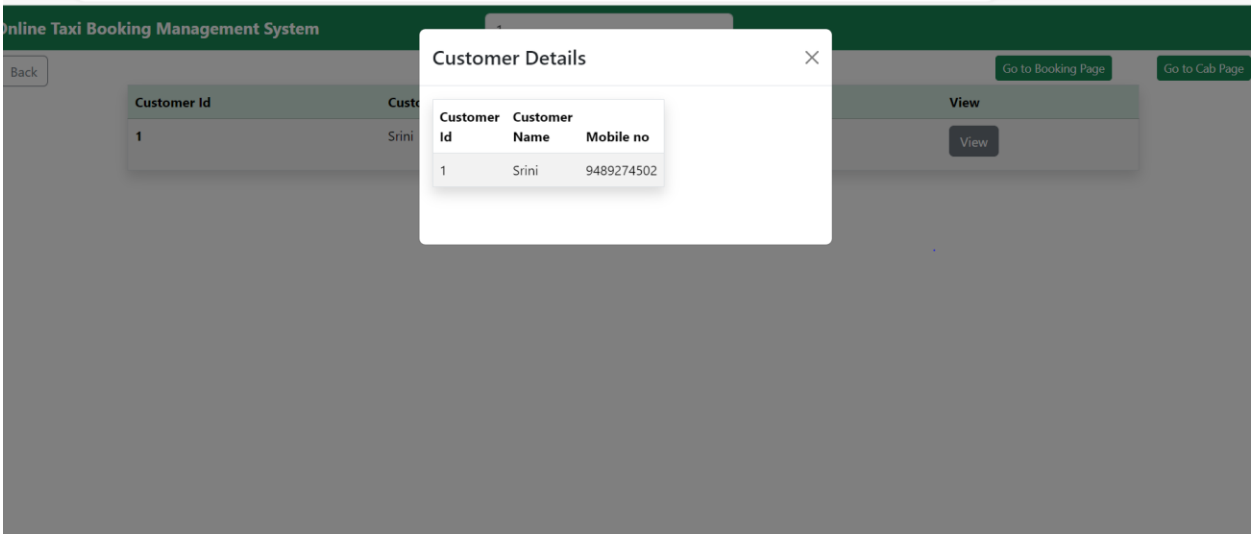
By clicking on to the cab page button, it will redirect to the cab home page.



Through search bar, we can search customer by id



By clicking the view button, we can see the deatils of the particular customer

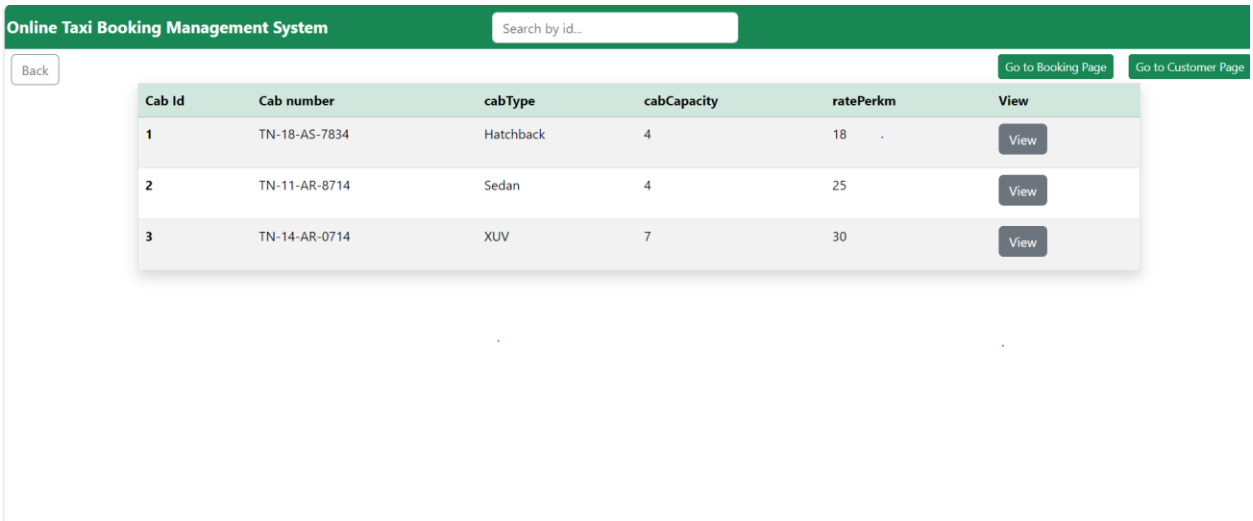


Here it is a cab view page.

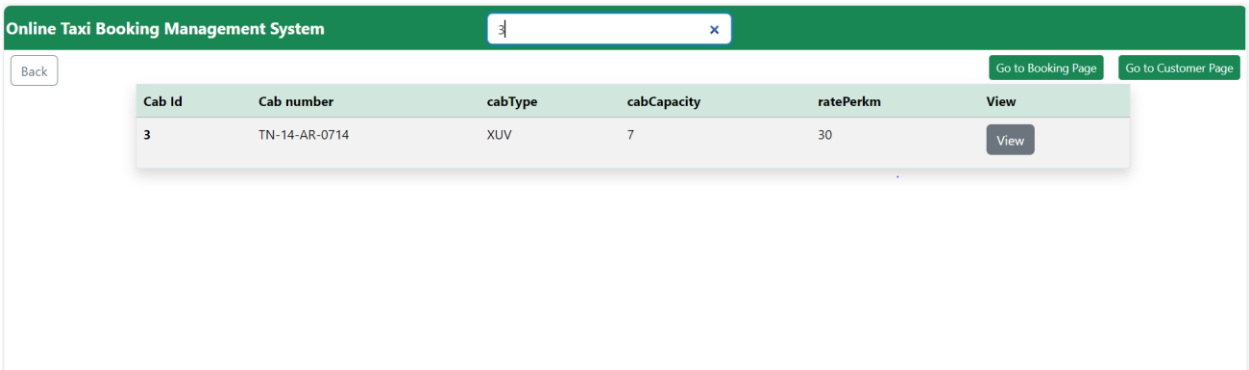
By clicking on to the back button, it will redirect to the booking home page.

By clicking on to the booking page button, it will redirect to the booking home page.

By clicking on to the customer page button, it will redirect to the customer home page.



Through search bar, we can search cab by id



By clicking the view button, we can see the details of the particular cab detail.