Software Engineering

ICT2405

Rithika Hettiarachchi - 0110
Lecturer: Miss Mahesha Thejani
30 August 2022

Part 01

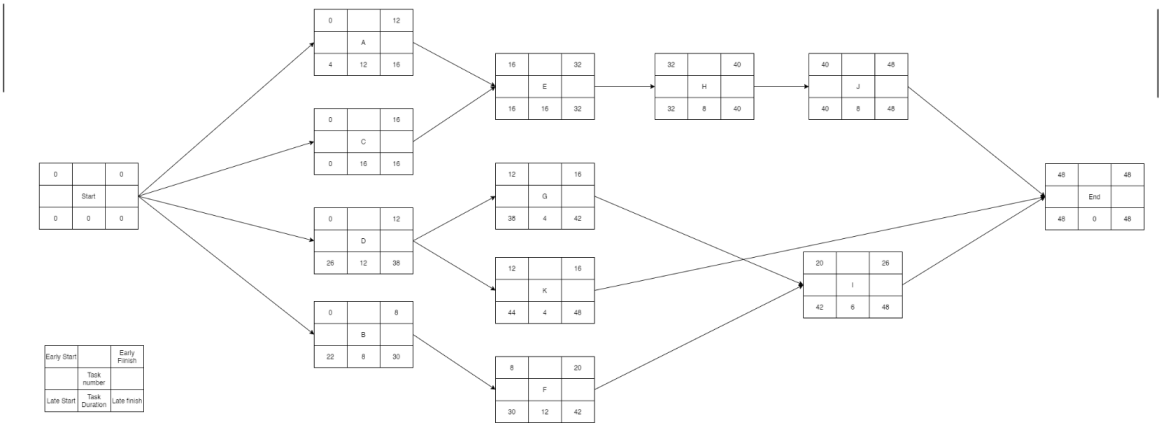
Question 1

The following tasks, labelled A to K, need to be completed in order to undertake staff induction for a Local Authority. The durations of tasks are measured in weeks and the costs of tasks are in Rs ’000

ID	Dur (Weeks)	Preceded by:	Cost (Rs.’000):
A	12	-	560
B	8	-	720
C	16	-	40
D	12	-	120
E	16	A,C	200
F	12	B	120
G	4	C,D	40
H	8	E	160
I	6	F,G	80
J	8	H	200

Administrative and accommodation overheads including project management fees are £4000 per week for the duration of the project.

A. Find the critical path, duration, and total cost for this project.



1. Critical Path can be written as follows – C, E, H, J
2. Duration will be 48 weeks to complete the project.
3. Total cost to complete this project will be

    Management fees 48 * £4000 = £192,000

    Cost of tasks =

    (560 000 + 720 000 + 40 000 + 120 000 + 40 000 + 160 000 + 80 000 + 200 000) / 276 {converting to pounds} = 1 920 000 / 276 = £6,953

    Total cost = 192,000 + 6,953 = £198,953

B. If each task requires one trainer, then, if all tasks take place at their earliest start times, how many trainers are required for the project?

Trainer 01 - A, E, H, J

Trainer 02 - C

Trainer 03 - D, G, I

Trainer 04 - K

Trainer 05 – B, F

As you can see in the above answer, I have used 5 trainers even though that there are 11 tasks, that because some tasks are preceded by or dependent on some task.

Part 02

Case study

The software to be developed is to support the operations of Fiesta Cab Service (FCS), specifically the administration of booking and rental vehicles for long distance trips and special functions. The customer should be able to book the vehicle at least 3 days prior to the event. This service doesn’t contain any real time booking facility or the location handling (GPS, etc.)

FCS operates a number of subsidiaries across the country where each has a collection of vehicles and rents them to customers in the geographical area assigned to them according to the requirements set while a customer makes a booking. FCS has a number of supervisory staff who are responsible for customer bookings.

The entire region FCS operates in is divided into geographical areas, one for each subsidiary, and each customer is delivered vehicles from the subsidiary for the geographical area in which the customer is located. Every subsidiary has at least 1 customer in its area. Each customer is located in one place only, i.e. in only one subsidiary's area. Operator should be able to create, edit, delete customers (Both regular and Business) and search customers by mobile number.

Each vehicle (uniquely identified by a registration number) is available at only one subsidiary which may, however, rent out more than one vehicle.

The subsidiary location of vehicles does not ever change (a simplifying assumption).

The driver allocated to a booking event does not change except for rare occasions (e.g. Illness of drivers). A minimum of at least two alternative drivers are kept at each subsidiary (again, another simplifying assumption).

Customers make bookings. Some customers, however, have not yet made any booking. A customer can be either one of two types, Business (customers who approach the company on a regular basis) or Personal (customers who approach the organization occasionally). Customer will select the type of booking (Pick / Pick & Drop), date, time, current location and destination. A booking consists of vehicles (at least one), specifying the type of vehicle and the date of delivery. Operator will display all vehicles available for

booking and confirm the booking by updating vehicle, supervisor and driver information. She/he is able to display all Driver details group by city.

When a vehicle is delivered, the vehicle booking is marked as delivered by entering the date of vehicle delivery. For Pick & Drop bookings, the vehicle return date is also recorded with the other booking details.

The customer is liable to pay for the hire at the time of vehicle delivery. (payment handling is not expected to develop in your system)

Each customer has a vehicle booking limit. This limit is the number of vehicles rented which may not be exceeded.

A confirmed customer booking is always notified to the customer 1 hour following the time of vehicle booking. The notification consists of Venue (Pick / Pick & Drop), Vehicle, Driver and Booking details.

The company employs supervisory staff. Each supervisory staff overlooks customer bookings that are directed from the one or more subsidiaries which are allocated exclusively to that staff.

In addition, each vehicle at a subsidiary is assigned to a particular supervisory staff member who is responsible for checking the vehicle condition before and after the hire, but not all supervisory staffs are responsible for checking any vehicles.

System should be able to display all vehicles which are maintained by a particular supervisor.

The system should be able to generate the below reports.

- Display all pick booking including vehicle information along driver details
- Generate a list of top 5 subsidiaries which have hired out the most numbers of vehicles for last 30days.
- Allow the system to enter a customer ID and produce a report that details all hires off that customer in a suitable format.
- Display all customers who have not place any orders for last 100 days.

Task 1 - Requirement Engineering

1.1 Functional Requirements

1. Creating Records

F1	**Creating New Customer, Driver, Supervisor, Vehicle and Bookings**
Summary	Creating new customers, driver, supervisor where the id is auto generated but the rest are inputted. Whereas creating the Vehicle its registration number is used as the id to uniquely identify them all. After all are creating a new booking can be set by a supervisor by adding all the required data.
Pre-condition	Application must be running
Post-condition	-
Input	Required Details of the Customer, Supervisor, Vehicle, Driver and Booking
Process	After the Details are passed form the controller layer to the service later the details will be verified before they are been added to the system.
Output	If the details are faulty then an error message is shown while if the details are correct then the db will return the added data as the response

2. Updating Records

F2	**Updating the entities (Customer, Driver, Supervisor, Vehicle and Bookings)**
Summary	If the user decides to edit the data that's been stored in the database or if an service layer method needs to update the data that's stored in the db it should be possible
Pre-condition	Application must be running and entity should be already added
Post-condition	-
Input	The unique Id and the updated data as the request body
Process	Will find the record for the given id and if found then it will set the data as given by the request body

F2	Updating the entities (Customer, Driver, Supervisor, Vehicle and Bookings)
Process	Will find the record for the given id and if found then it will set the data as given by the request body
Output	If data was not found or if the updating data is not correct then an error message us shown otherwise a response with the updated data is shown

3. Delete Records

F3	Deleting the entities (Customer, Driver, Supervisor, Vehicle and Bookings)
Summary	If the user decides to delete the data that's been stored in the database or if an service layer method needs to delete the data that's stored in the db it should be possible
Pre-condition	Application must be running and entity should be already added
Post-condition	-
Input	The unique Id
Process	Will find the record for the given id and if found then it will delete the data
Output	It will delete the data from the databases and return an empty response.

4. Reading the entities

F4	Reading the entities (Customer, Driver, Supervisor, Vehicle and Bookings)
Summary	If the user decides to read the data that's been stored in the database or if an service layer method needs to read the data that's stored in the db it should be possible
Pre-condition	Application must be running and entity should be already added
Post-condition	-
Input	The unique Id or any other attribute that we are searching by.
Process	Will find the record for the given id or attribute and return it and if found then it will delete the data

F4	Reading the entities (Customer, Driver, Supervisor, Vehicle and Bookings)
Output	The data will be retrieved if it existed. Or an empty array will be returned.

5. Searching Customer By Phone Number

F4	Search Customer by phone number
Summary	Searching for customers by their registered
Input	The unique Id or any other attribute that we are searching by.
Pre-condition	Application must be running and entity should be already added
Post-condition	-
Process	Will find the customer record for the given mobile nd return it and if found then it will delete the data
Output	The data will be retrieved if it existed. Or an empty array will be returned.

6. Generating Reports

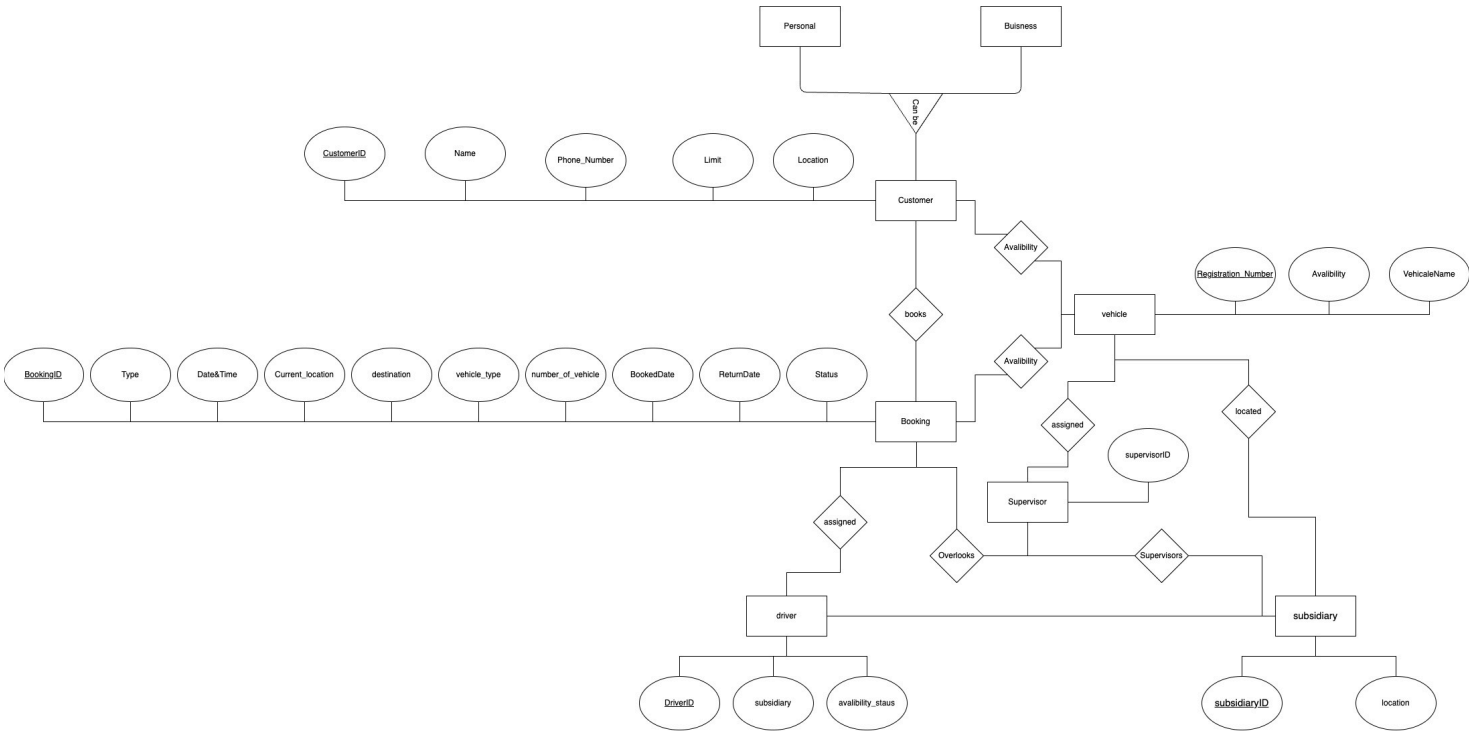
F5	Report Generation
Summary	Generating reports based on the data that's existing in the databases
Input	-
Pre-condition	Application must be running and All entities(Customer, driver, supervisors, bookings) should be already added
Post-condition	-
Process	Getting the required data from database sending it to the service layer where a json response is created as the results.
Output	The required result if the data for it is present otherwise would return an empty array or a error message.

1.2 Non-Functional Requirements

- Speed - Having multiple subsidiary and each one of them having multiple bookings, this creates huge amount of data, when loading these data it should be quick
- Reliability of the generated reports and the CRUD operations for the entities
- Portability how well it can work from different operating systems.
- Limit of bookings - A unreasonable and unrealistic number of booking at one time can’t placed by the customer
- Searching and Displaying Data should be easy.

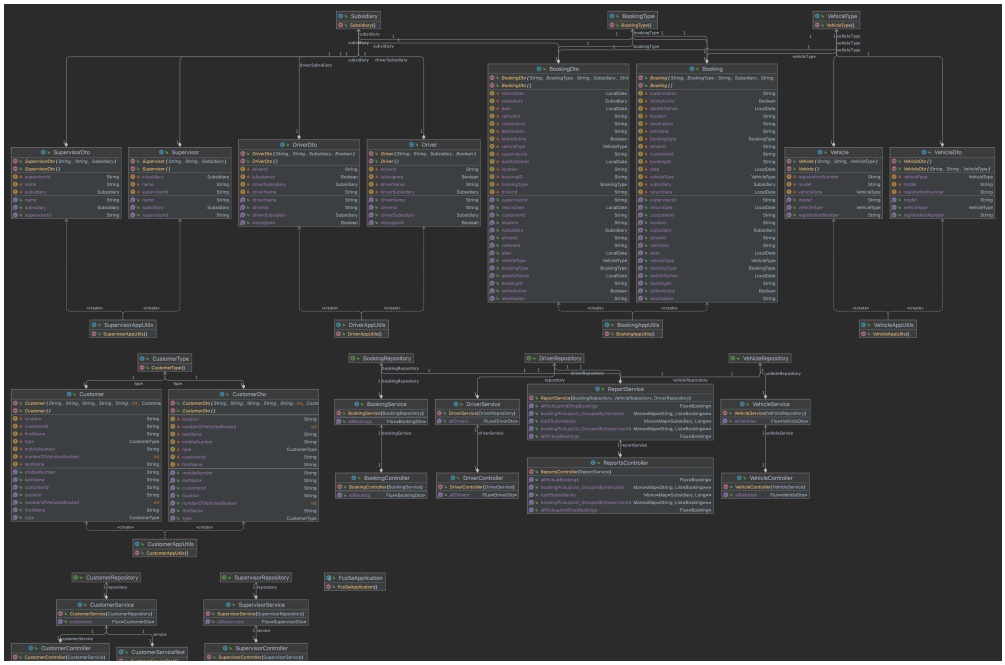
Task 2: Design the System

2.1 Evidences for the designed system



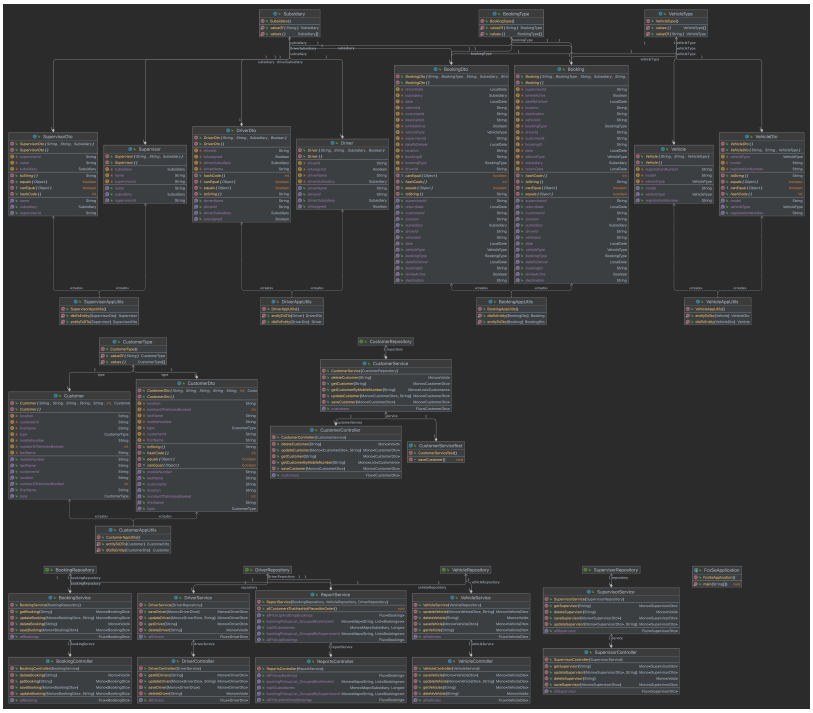
Entity Relationship Diagram

Figure 2.1 Entity Relationship diagram



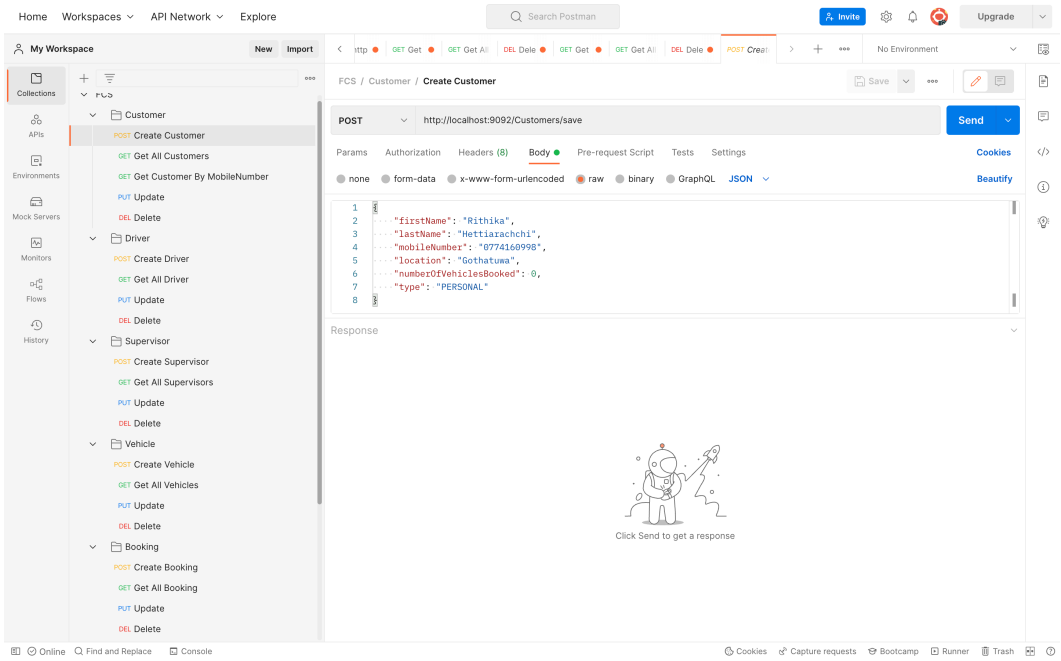
Class diagram

Figure 2.2.1 Java Class Diagram



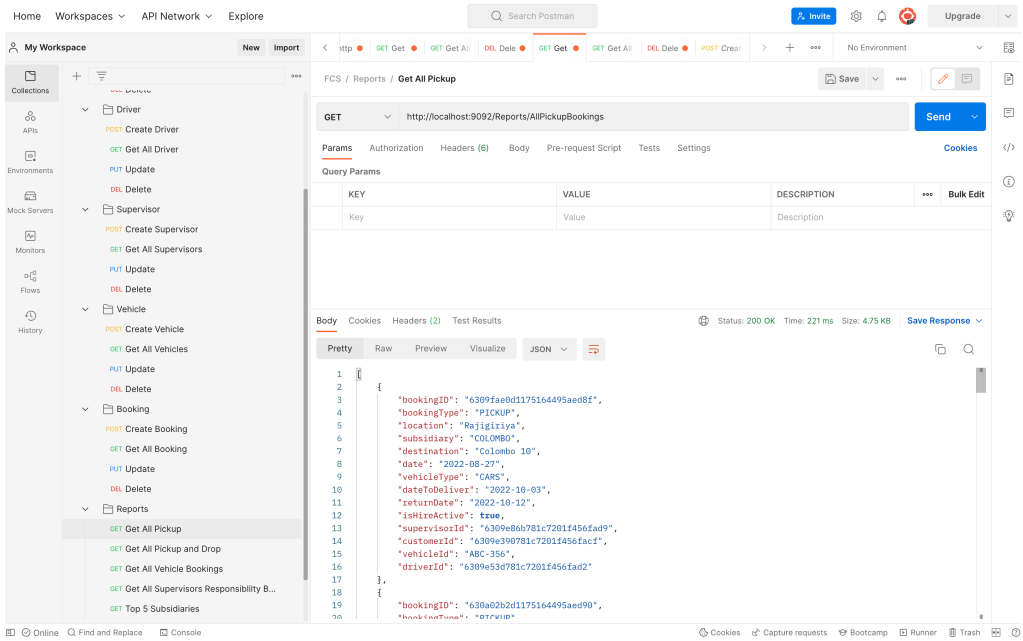
Class diagram with all the methods

Figure 2.2.2 Java Class Diagram with methods



PostMan Collection Customer

Figure 2.3.1 PostMan Collection



Postman continued

Figure 2.3.2 PostMan Collection

2.2 Usage of OCL in the UML diagramming

Unified Modelling Language (UML) is a standard way to visualise the design of a system. Object Constraint Language (OCL) is Official Part of UML. Enables one to describe expressions and constraints on object-oriented models and other object modelling artefacts.

There are couple of expression types in OCL

1. Invariants
2. Initialisation expressions
3. Derived elements
4. Query operations
5. Operation Contracts

Examples of OCL in the given Scenario:

Context Customer :: insert()
pre: Supervisor.supervisorId; Vehicle.vehicleId; Driver.driverId;
Customer.customerId;

Showing a representation of pre condition when placing a bookings. When booking placed there should be values for the supervisorId, vehicleId, driverId and customerId in the system before a booking is placed. Where this is a “operation contrast”

Context Booking inv: LocalDate.now <= returnDate;

Specifies that the date that the vehicle should be delivered should be either greater or equals to the present date where this is an “invariant expression”

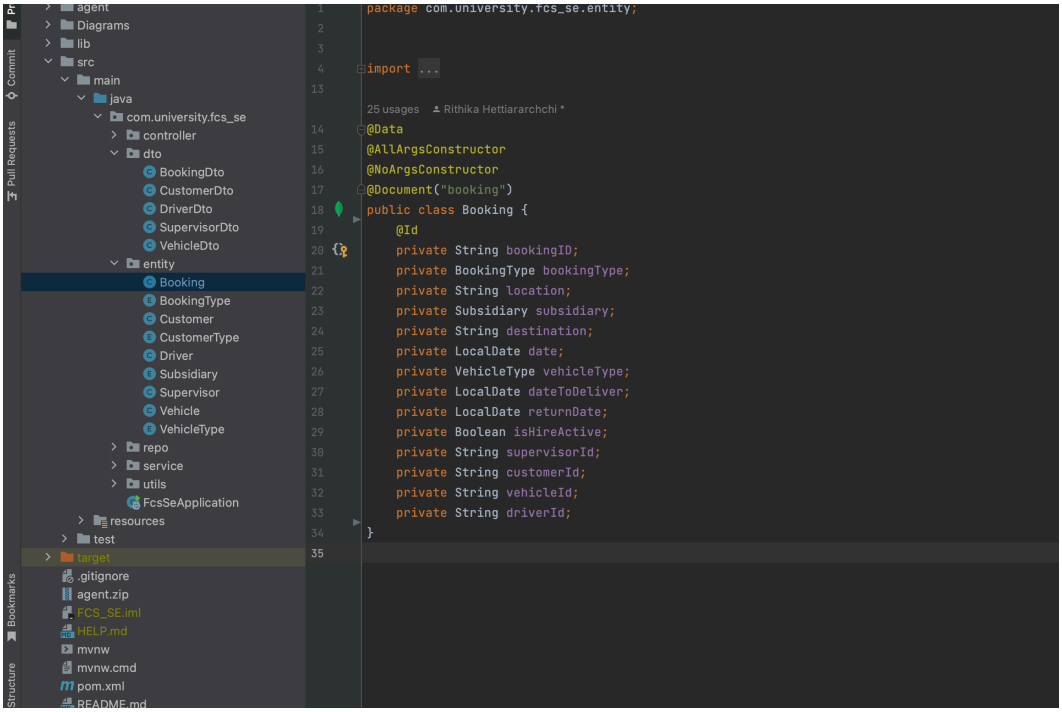
Context Customer :: numberOfVehiclesBooked : int : 10

Sets a default limit of every customer that the maximum number of booking that they can book is 10 therefor this is a “initialisation expression”.

Task 3: Evidences

3.1. Important code Lines and Evidences

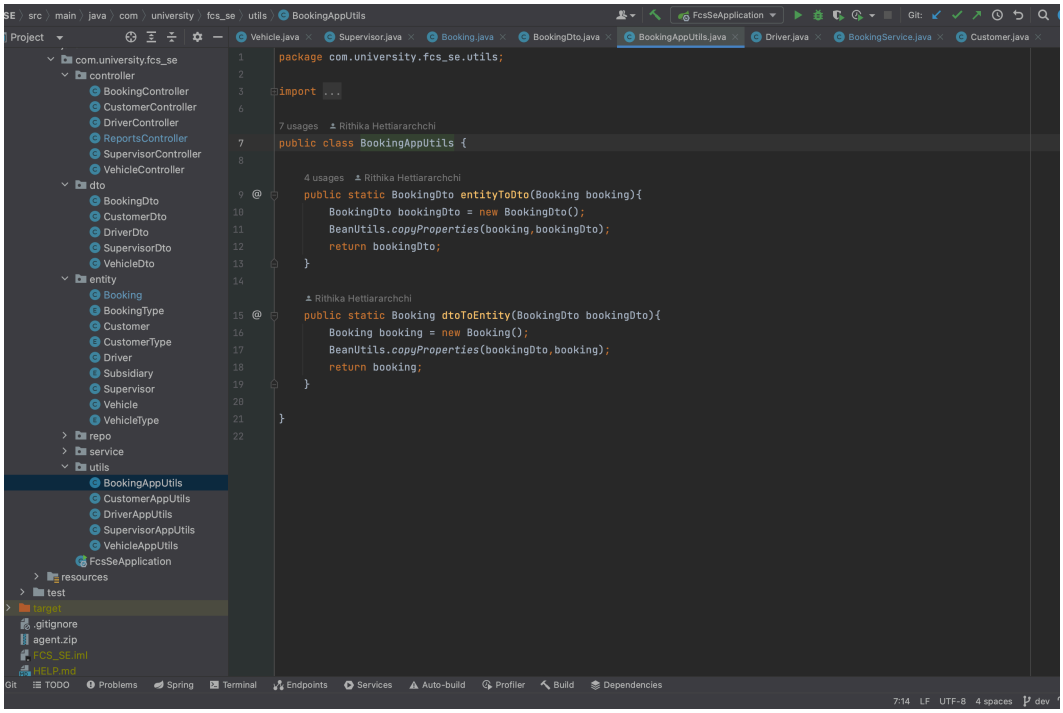
Below shows the Evidences for the build FCS Application. The project uses Reactive Programming along with the reactive mongoDB. to make it functional and managing the data it using Reactive Streams and Reactor APIs. As show below.



All the entities and DTO's

Figure 3.1 Evidences - Entities and DTO's

Even though DTO and Entity look similar the term "Data Transfer Object" (DTO) is defined quite unambiguously, the term "Entity" is interpreted differently in various contexts. Entity is class mapped to table. DTO is class mapped to "view" layer mostly. What needed to store is entity & which needed to 'show' on web page is DTO. But the reason to use DTO is due to security.

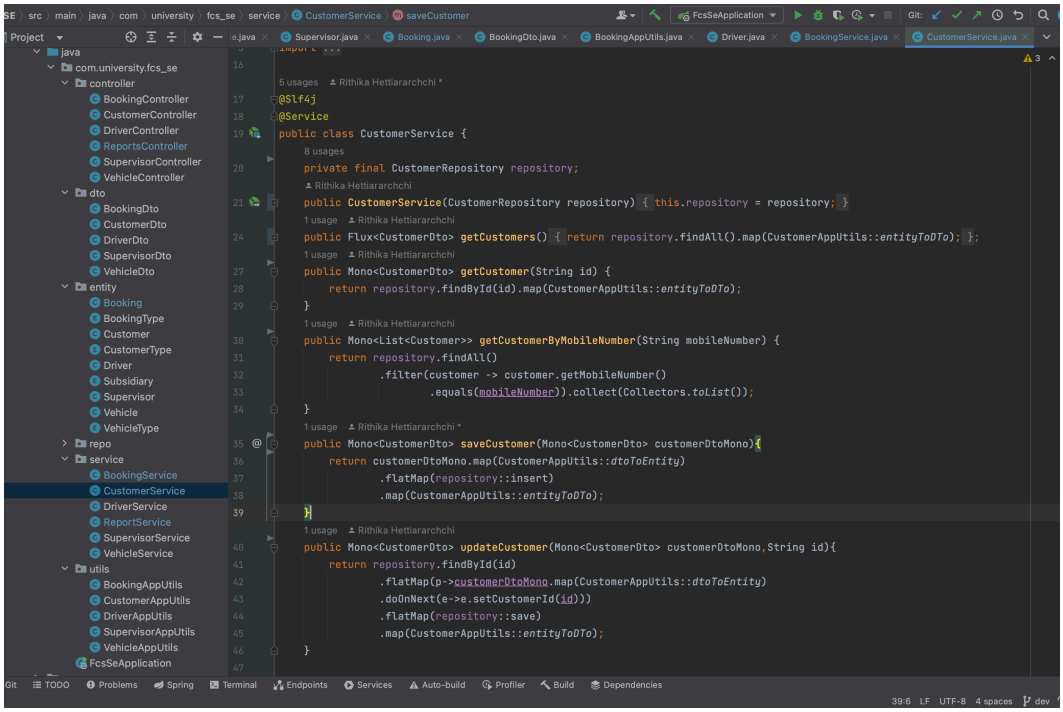


AppUtils Copying data from DTO to Entity and Vice Versa

Figure 3.2 Evidences - AppUtils

When we are taking the advantage of using DTOs on RESTful APIs written in Java (and on Spring Boot), is that they can help hiding implementation details of domain objects (aka. entities). Exposing entities through endpoints can become a security issue if we do not carefully handle what properties can be changed through what operations.

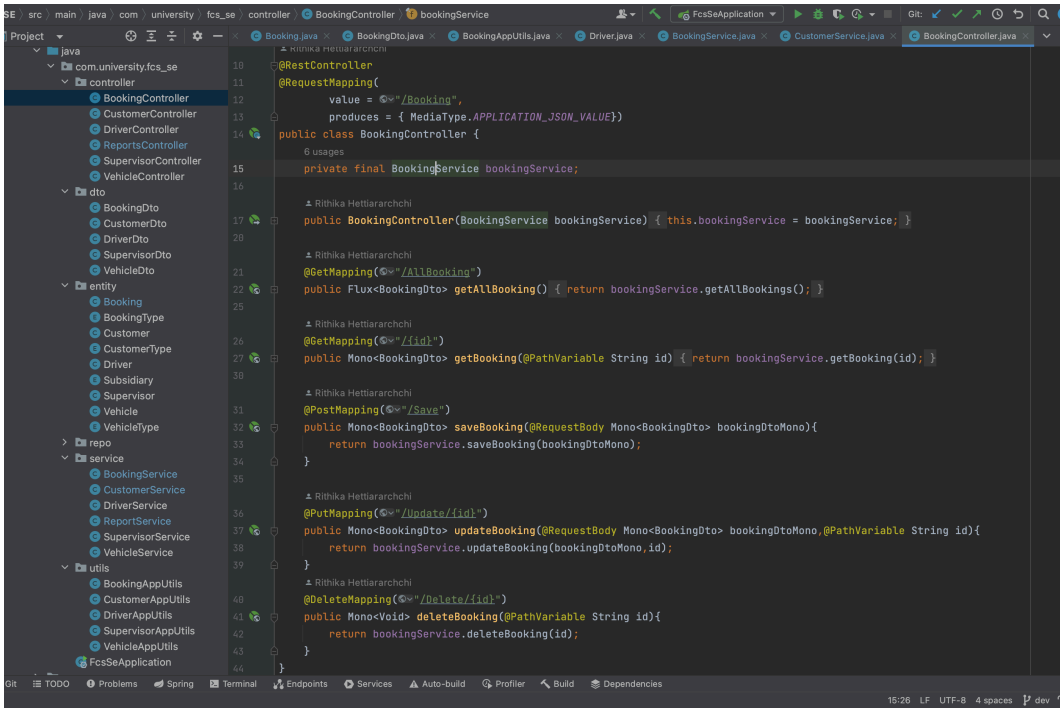
When using DTOs is to map the DTO to the entity which can done by the Beans.copyProperties () or by a model mapper and vice-versa. Each Entity has a AppUtils for them which can be seen in Figure 3.2.



Customer Service Class CRUD operations

Figure 3.3 Evidences - Service Layer

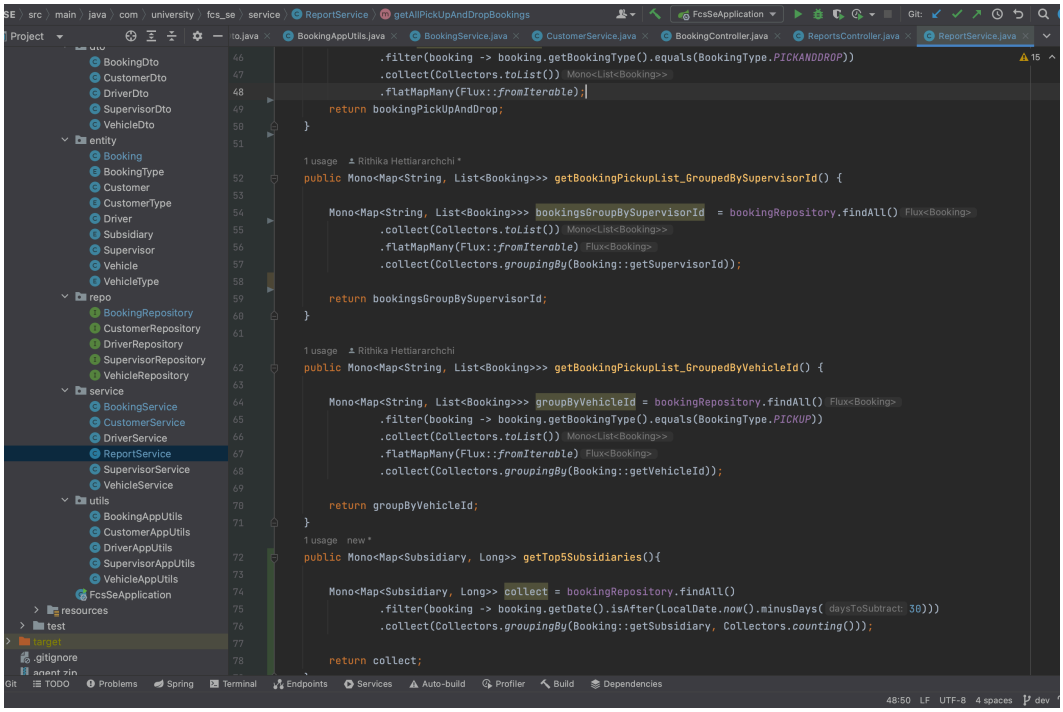
Developing the Service Layer as we don't want the methods in the service layer to block, the service methods return reactive types. Figure 3.3 show CustomerService. @Service annotation used to mark the class as a service provider. So overall @Service annotation is used with classes that provide some business functionalities. Spring context will autodetect these classes when annotation-based configuration and class-path scanning is used. Add the spring-context dependency in your pom.



Booking Controller

Figure 3.4 Evidences - Controller

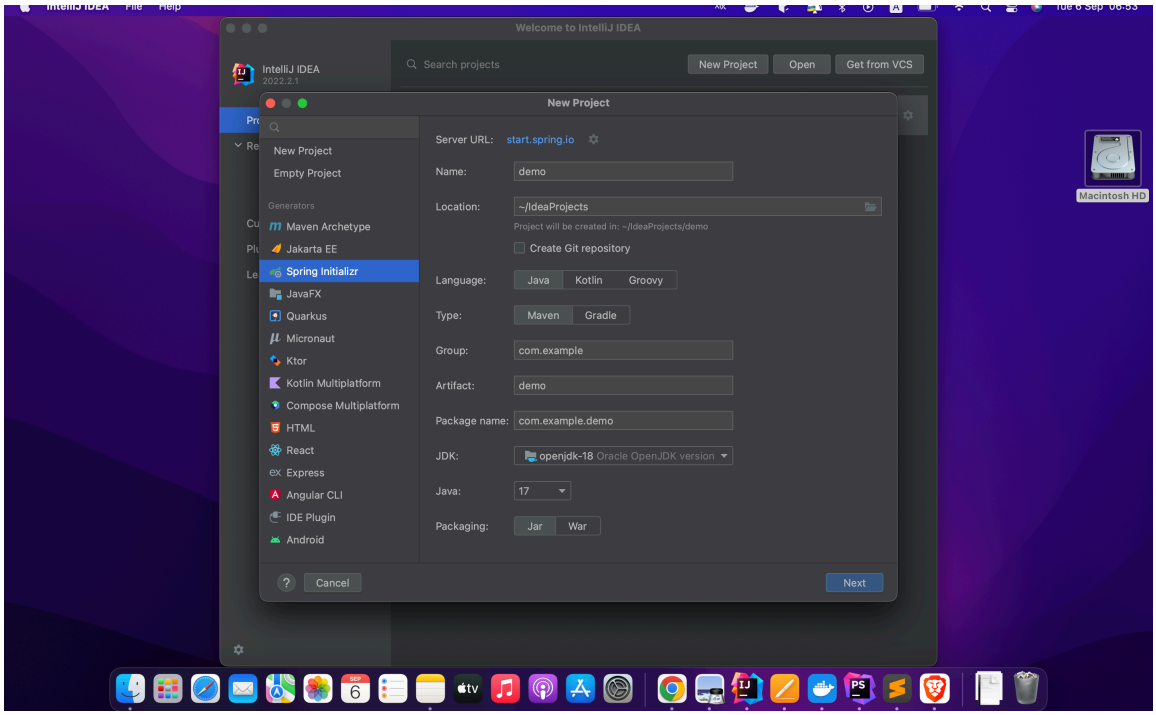
The controller will handle the navigation between the different views. The @Controller annotation extends the use-case of @Component and marks the annotated class as a business or presentation layer. When a request is made, this will inform the DispatcherServlet to include the controller class in scanning for methods mapped by the @RequestMapping annotation. But in this case I have used @RestController. The @RestController annotation in Spring is essentially just a combination of @Controller and @ResponseBody. This annotation was added during Spring 4.0 to remove the redundancy of declaring the @ResponseBody annotation in your controller.



Reports Service Class

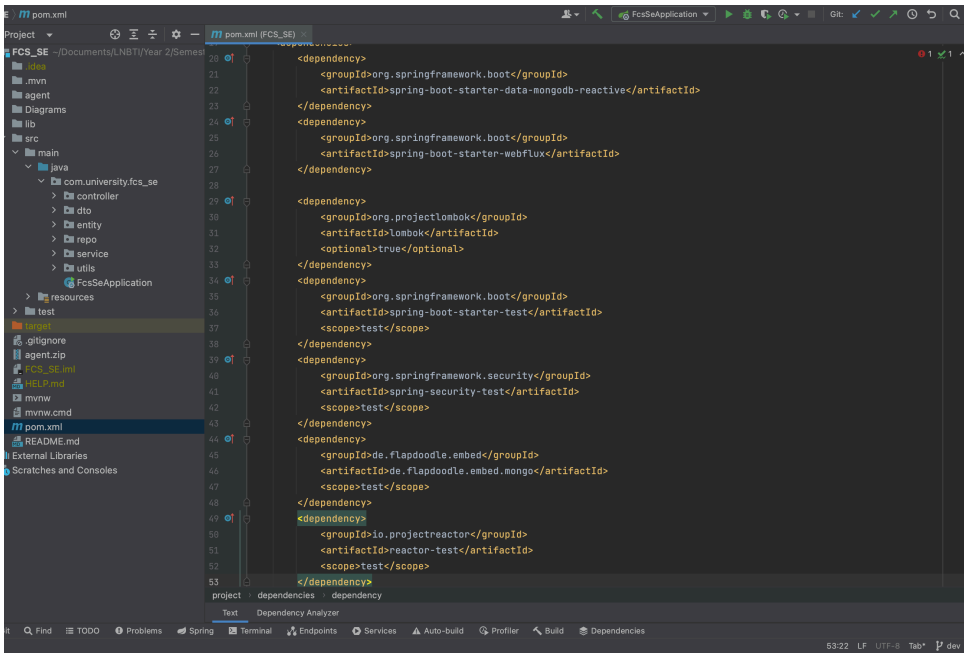
Figure 3.5 Evidences - Report Generation

This class has written methods for the use case requirements of generating Reports. Example here all bookings are taken and then they are been grouped by each supervisorId. So the user can see when bookings the supervisor is responsible for.



Spring Initialiser integrated with IntelliJ

Figure 3.6 Features of selected IDE - Integration of Spring initialiser



Maven Dependancy manager

Figure 3.7 Features of selected IDE - Maven Dependency manager

Task 4: Testing

4.1 Different Stages of testing

There are four stages of testing that need to be completed before the software application can be cleared for use

1. Unit testing
2. Integration testing
3. System testing
4. Acceptance testing

1. Unit Testing

Unit testing is done in the first round of testing, the system separated modules or Units will undergo testing to see if they, individually, ensure that component are fulfilling functionalities or not.

2. Integration testing

Integration testing is done to test the modules when integrated to verify that they work as expected. to test the modules which are working fine individually don't have issues when they are integrated. In Spring bot it’s all about running in Application Context and run tests. As the spring Framework does have a dedicated test module for integration testing. It is known as spring-test. This makes a use of JUnit 5 and mockito.

3. System testing

System Testing means testing the system as a whole. All the modules/ components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. With the main idea is the delivering high quality product.

4. Acceptance testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

4.2 Test plan

FCS Test Plan	
Introduction	FCS is a cab rental application that building using spring-Boot + Reactive MongoDB. Used to seamlessly manage the data cause its non-blocking and tend to be more efficient because they're not typing up processing while waiting for stuff to happen.
Test Items	• Customer • Bookings • Driver • Supervisor • Reports
Features to be test	• CRUD for Customer, Booking, Driver, Supervisor, Vehicle • Reports • FilteredData
Test Environments	Device : MacBook Pro M1 OS : Mac OS Monterey IDE. : IntelliJ Idea 2022 Database Manager : Mongo Compass
Test Approach	White Box testing and BlackBox testing
Test Deliverables	• Test Plan • Test Environment • Test Summary • Test Results • Test Evaluation Report

Table 4.1 Test Plan

4.3 Usage of White-Box and Black Box Designing

Black-Box Testing

Black box testing is used to test the system against external factors responsible for software failures. This testing approach focuses on the “input” that goes into the software, and the “output” that is produced. The testing team does not cover the inside details such as code, server logic, and development method.

White-Box Testing

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. Unit test is considered as white box testing.

Test Cases

Test Case	
Test Unit : Inserting Customer Data	Tester Rithika Hettiarachchi
Test Case ID : 01	Tester Type: Black Box Testing
Description: Creating a new Customer record and Verifying through DB	Test Execution Date: 6th September 2022
Title : Creating New Customer Record	Test Execution Time : 18:58

Table 4.2.1 Test Case 01

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes

1	Request with the new Customer data	{
		"firstName": "Rithika",
		"lastName": "Hettiarachchi",
		"mobileNumber": "0774160998",
		"location": "Gothatuwa",
		"numberOfVehiclesBooked": 0,
		"type": "PERSONAL"
		}
			[
			{
			"customerId": “{UniqueId}”,
			"firstName": "Rithika",
			"lastName": "Hettiarachchi",
			"mobileNumber": "0774160998",
			"location": "Gothatuwa",
			"numberOfVehiclesBooked": 0,
			"type": "PERSONAL"
			}
			]
				[
				{
				"customerId": "63174cc-c60442d33296adaa1",
				"firstName": "Rithika",
				"lastName": "Hettiarachchi",
				"mobileNumber": "0774160998",
				"location": "Gothatuwa",
				"numberOfVehiclesBooked": 0,
				"type": "PERSONAL"
				}
				]
					Success	When successfully entered the database the data is return as a response

2	Checking DB to verify	-	Database containing the data that was entered	Database containing the data that was entered	Success	

Table 4.2.2 Test Case 01 Steps

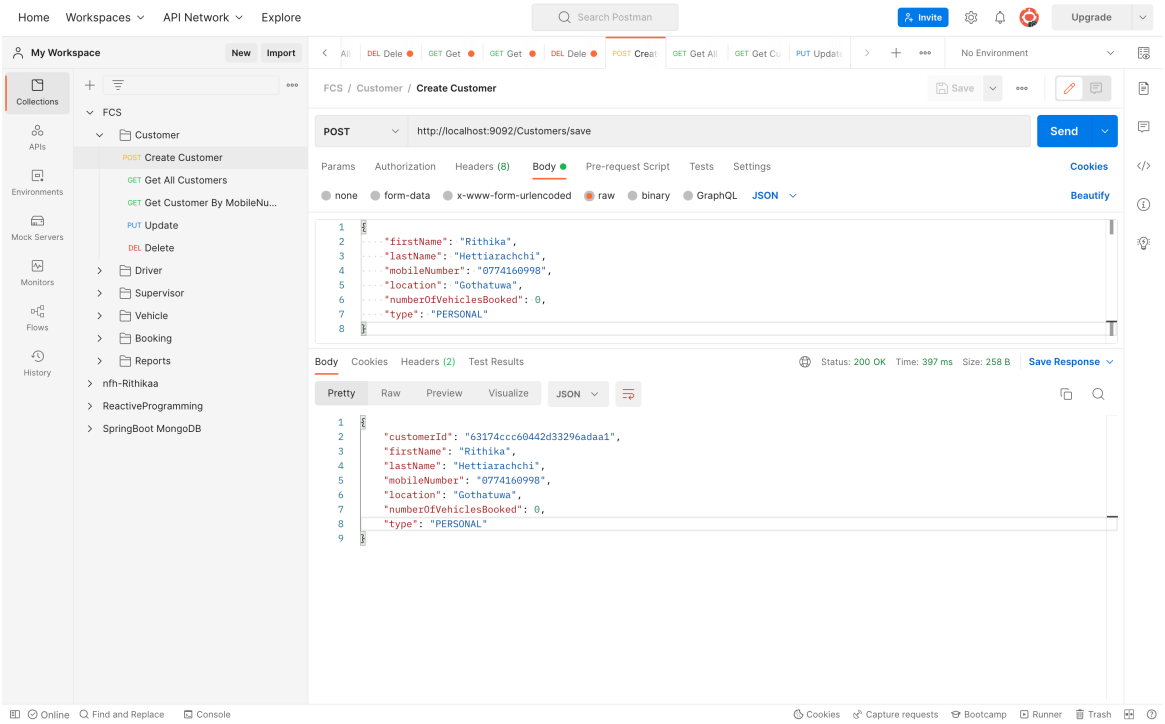


Figure 4.1.1 Test Case 01 Evidences - Postman

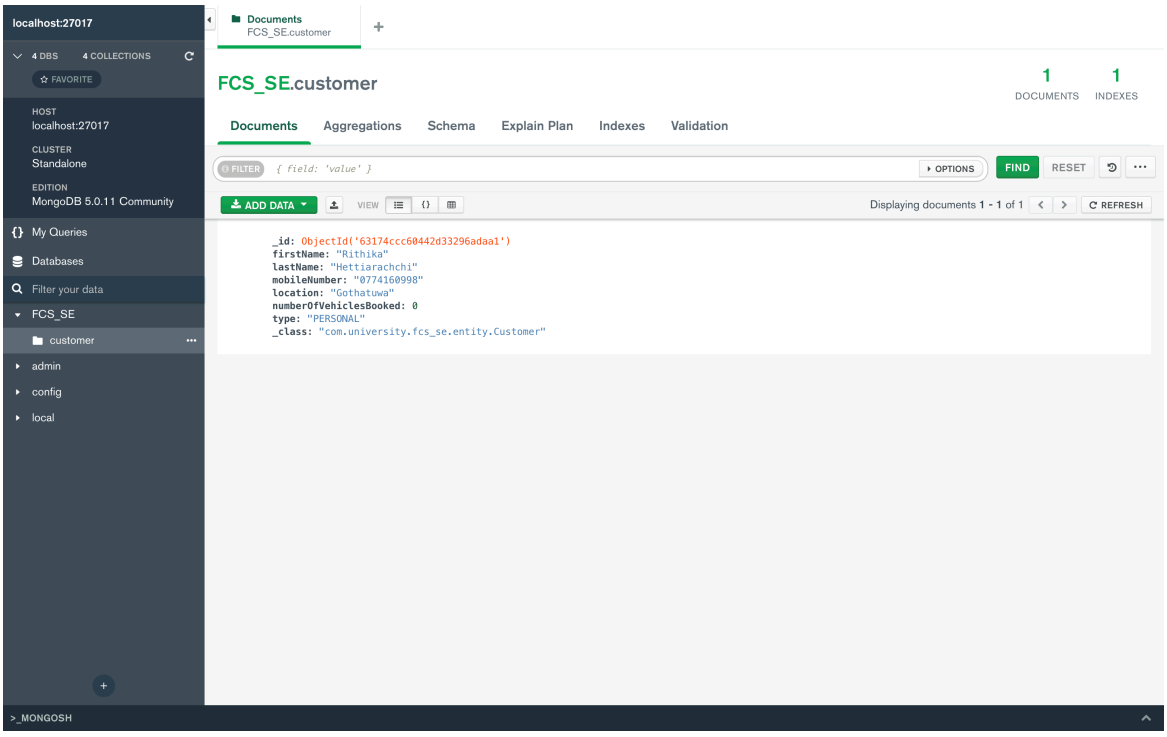


Figure 4.1.2 Test Case 01 Evidences Data - Mongo Compass

Test Case	
Test Unit : Reading All Customer Data	Tester Rithika Hettiarachchi
Test Case ID : 02	Tester Type: Black Box Testing
Description : Getting all Customers	Test Execution Date: 6th September 2022
Title : All Customers	Test Execution Time : 19:26

Table 4.3.1 Test Case 02

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes

1	Request	-	[
			{
			"customerId": “{UniqueId}”,
			"firstName": "Rithika",
			"lastName": "Hettiarachchi",
			"mobileNumber": "0774160998",
			"location": "Gothatuwa",
			"numberOfVehiclesBooked": 0,
			"type": "PERSONAL"
			}
			]
				[
				{
				"customerId": "63174cc-c60442d33296adaa1",
				"firstName": "Rithika",
				"lastName": "Hettiarachchi",
				"mobileNumber": "0774160998",
				"location": "Gothatuwa",
				"numberOfVehiclesBooked": 0,
				"type": "PERSONAL"
				}
				]
					Success	Since there is only one record only one is return

Table 4.3.2 Test Case 02 Steps

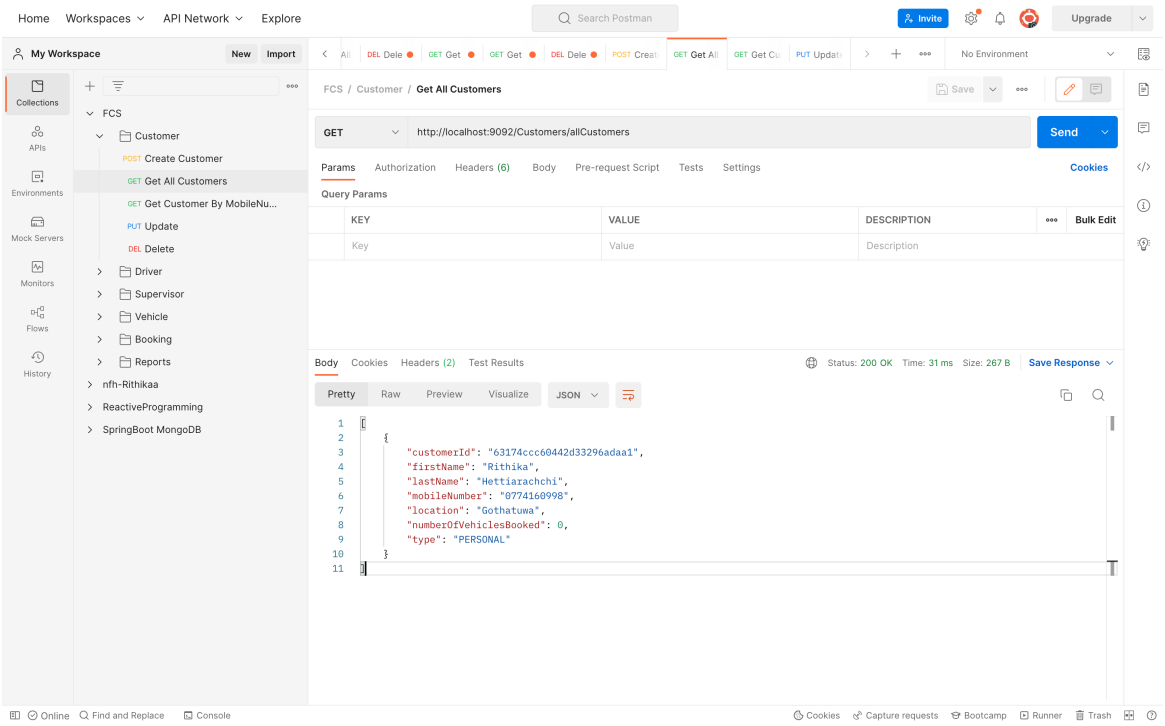


Figure 4.2 Test Case 02 Evidences Data - Postman

Test Case	
Test Unit : Searching Customer Data By Mobile Number	Tester Rithika Hettiarachchi
Test Case ID : 03	Tester Type: Black Box Testing
Description : Getting Customers By Mobile Number	Test Execution Date: 6th September 2022
Title : Customers By Mobile Number	Test Execution Time : 19:35

Table 4.4.1 Test Case 03

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes

1	Request along with phone number	774160998	[
			{
			"customerId": "63174ccc60442d33296adaa1",
			"firstName": "Rithika",
			"lastName": "Hettiarachchi",
			"mobileNumber": "0774160998",
			"location": "Gothatuwa",
			"numberOfVehiclesBooked": 0,
			"type": "PERSONAL"
			}
			]
				[
				{
				"customerId": "63174cc-c60442d33296adaa1",
				"firstName": "Rithika",
				"lastName": "Het-tiarachchi",
				"mobileNumber": "0774160998",
				"location": "Gothatuwa",
				"numberOfVehicles-Booked": 0,
				"type": "PERSONAL"
				}
				]
					Success	Since there is only one record only one is return

Table 4.4.2 Test Case 03 Steps

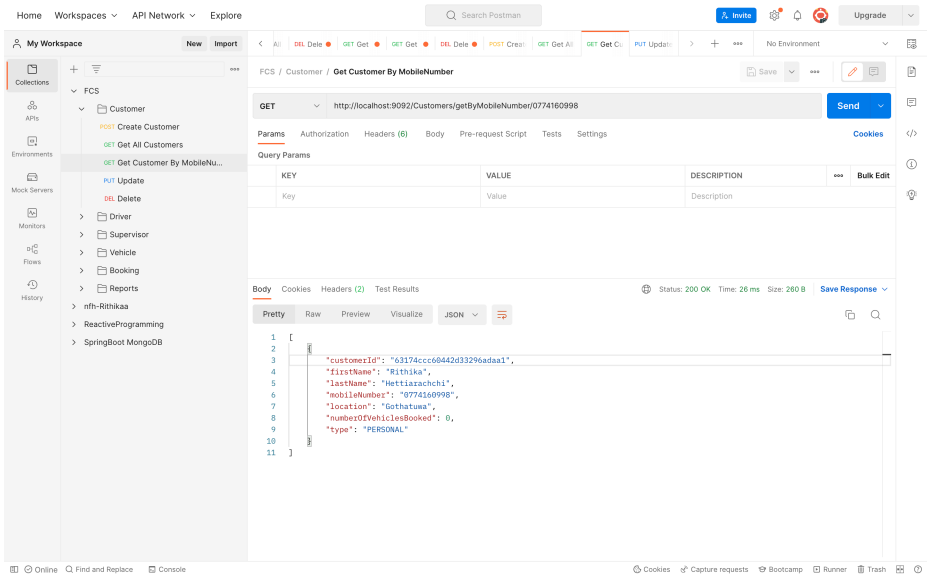


Figure 4.3 Test Case 03 Evidences Data - Postman

Test Case	
Test Unit : Updating Customer	Tester Rithika Hettiarachchi
Test Case ID : 04	Tester Type: Black Box Testing
Description : Updating Customer Record and verifying it from the database	Test Execution Date: 6th September 2022
Title : Updating Customer Record	Test Execution Time : 19:44

Table 4.5.1 Test Case 04

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request along with Customer ID, Updated data	{ "firstName": "Rithika", "lastName": "Hettiarachchi", "mobileNumber": "0773920998", "location": "Colombo 5", "numberOfVehiclesBooked": 0, "type": "BUSINESS" }	{ "customerId": "63174ccc60442d33296adaa1", "firstName": "Rithika", "lastName": "Hettiarachchi", "mobileNumber": "0773920998", "location": "Colombo 5", "numberOfVehiclesBooked": 0, "type": "BUSINESS" }	{ "customerId": "63174cc-c60442d33296adaa1", "firstName": "Rithika", "lastName": "Hettiarachchi", "mobileNumber": "0773920998", "location": "Colombo 5", "numberOfVehiclesBooked": 0, "type": "BUSINESS" }	Success	Phone Number and the customer type was changed here
2	Checking DB to verify	-	Database containing the data that was updated	Database containing the data that was updated	Success	

Table 4.5.2 Test Case 04 Steps

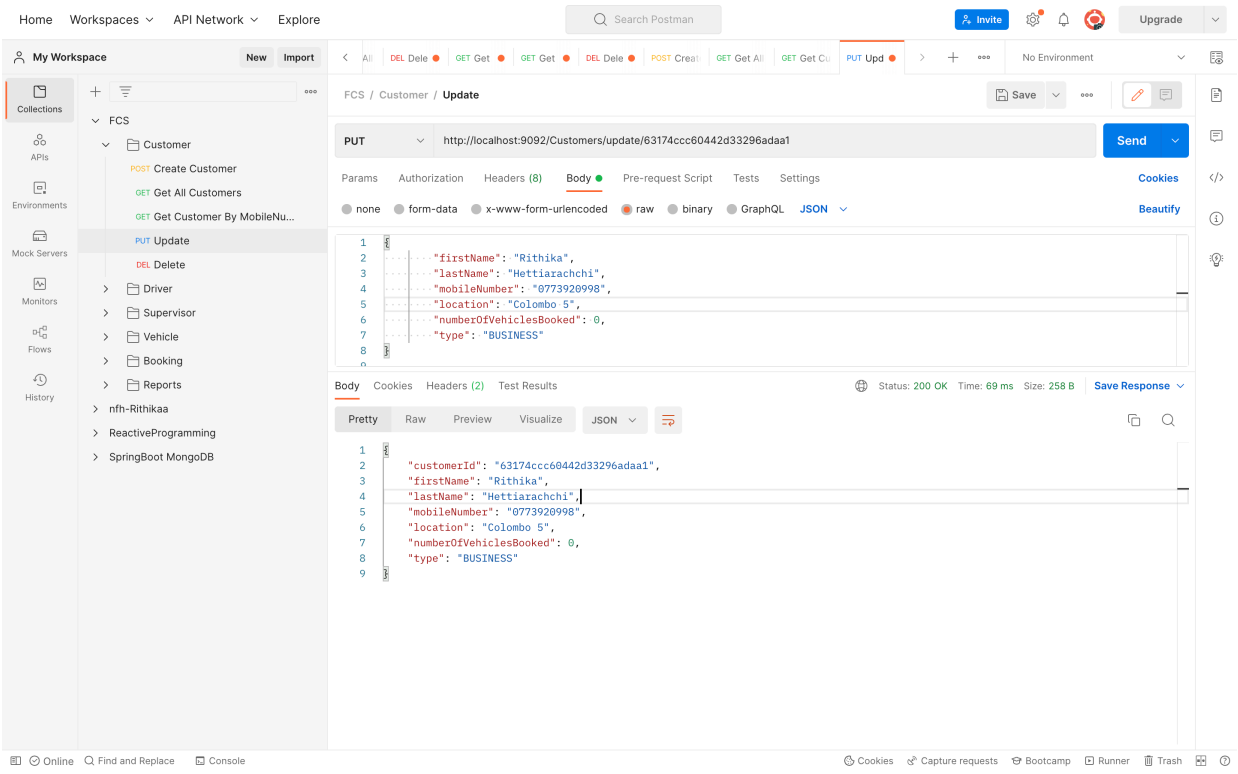


Figure 4.4.1 Test Case 04 Evidences Data - Postman

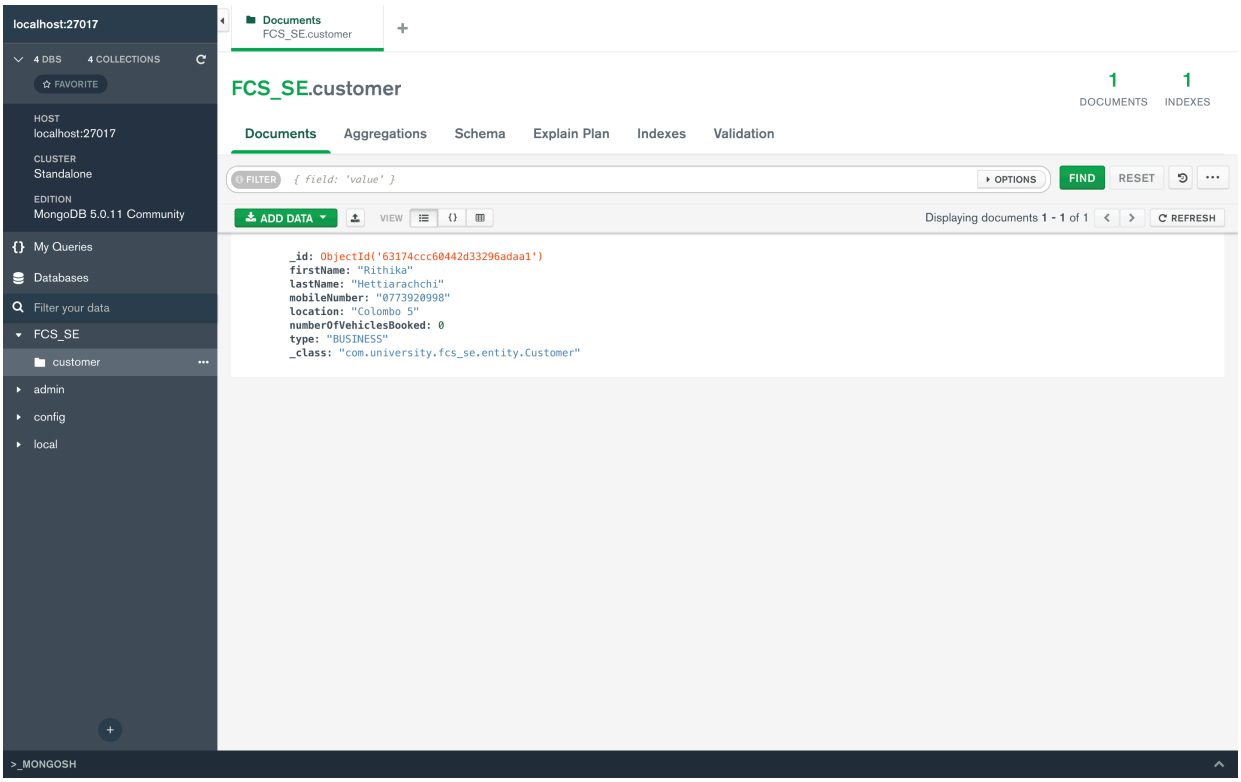


Figure 4.4.2 Test Case 04 Evidences Data - MongoDB

Test Case	
Test Unit : Deleting Customer	Tester Rithika Hettiarachchi
Test Case ID : 05	Tester Type: Black Box Testing
Description : Deleting Customer Record and verifying it from the database	Test Execution Date: 6th September 2022
Title : Deleting Customer Record	Test Execution Time : 19:44

Table 4.6.1 Test Case 05

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request along with Customer ID	-			Success	Empty Value is return when the record is deleted
2	Checking DB to verify	-	Database containing the data that was deleted	Database containing the data that was deleted	Success	

Table 4.6.2 Test Case 05 Steps

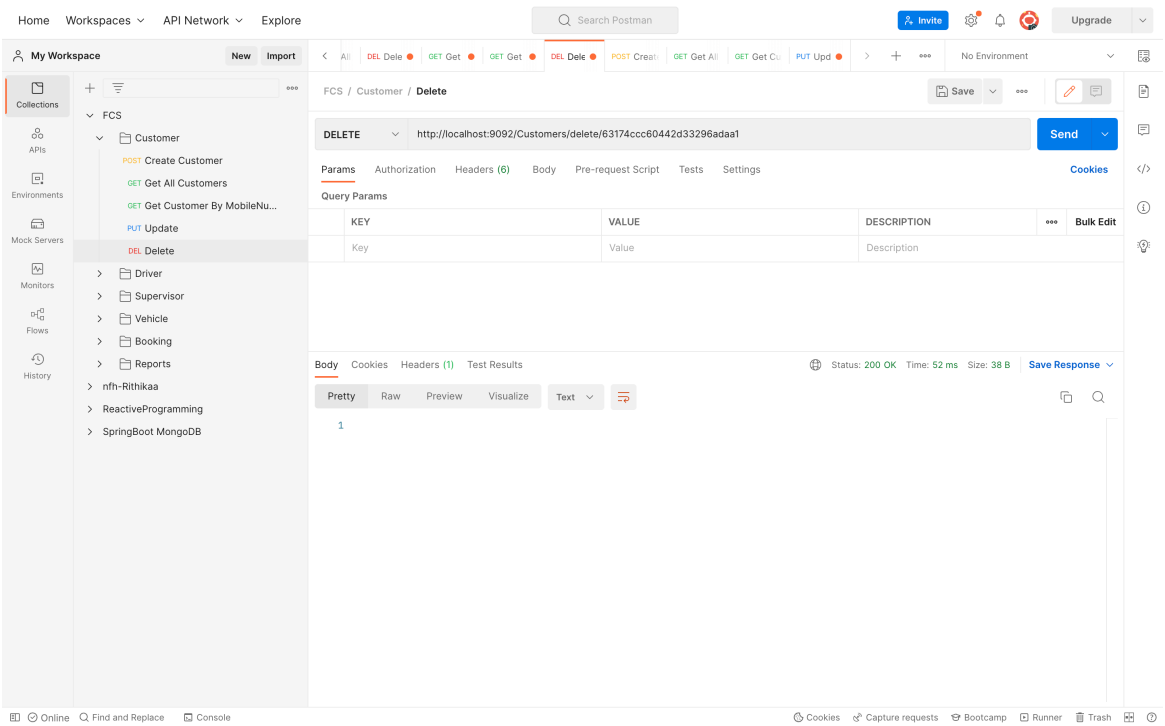


Figure 4.5.1 Test Case 05 Evidences Data - Postman

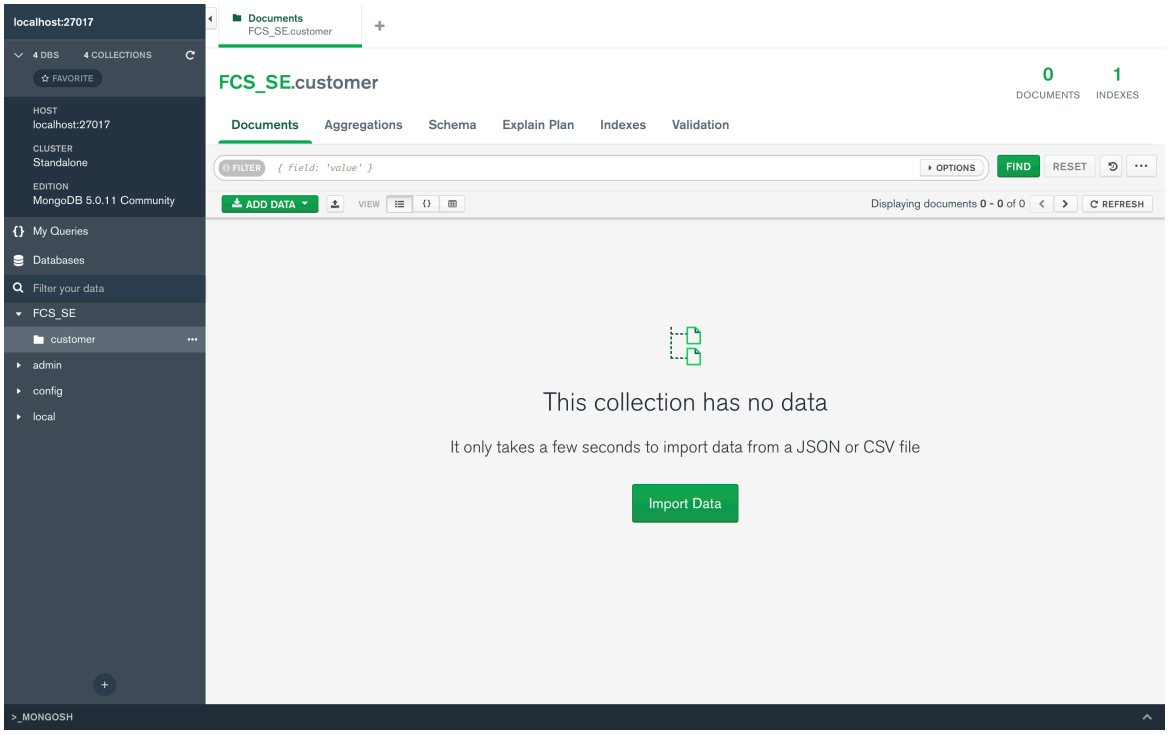


Figure 4.5.2 Test Case 05 Evidences Data - Mongo Compass

Test Case	
Test Unit : Creating Booking	Tester Rithika Hettiarachchi
Test Case ID : 06	Tester Type: Black Box Testing
Description :Placing a New Booking	Test Execution Date: 7th September 2022
Title : Creating A New Booking Record	Test Execution Time : 07:45

Table 4.7.1 Test Case 06

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request along with booking data	{ "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType":"SUV", "dateToDeliver":"2022-09-08", "returnDate":"2022-09-12", "isHireActive": "true", "supervisorId": "6317fc5460442d33296adaa4", "customerId":"6317fbe560442d33296adaa2", "vehicleId" :"ADE-216" , "driverId" :"6317fe1960442d33296adaa5" }	{ "bookingID": "{Unique Id}", "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }	{ "bookingID": "6317fe4060442d33296adaa6", "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }	Success	When the booking is successfully added the data entered to the db is return.
2	Checking DB to verify	-	Database containing the data that was Entered	Database containing the data that was Entered	Success	

Table 4.7.2 Test Case 06 Steps

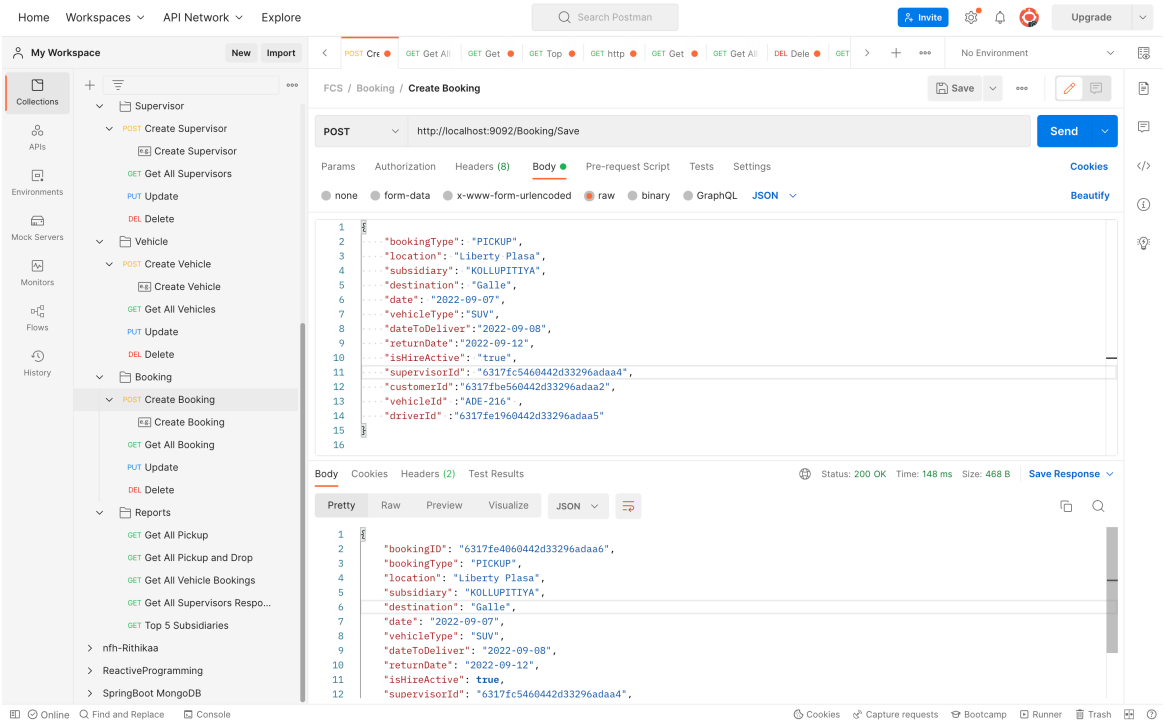


Figure 4.6.1 Test Case 06 Evidences Data - Postman

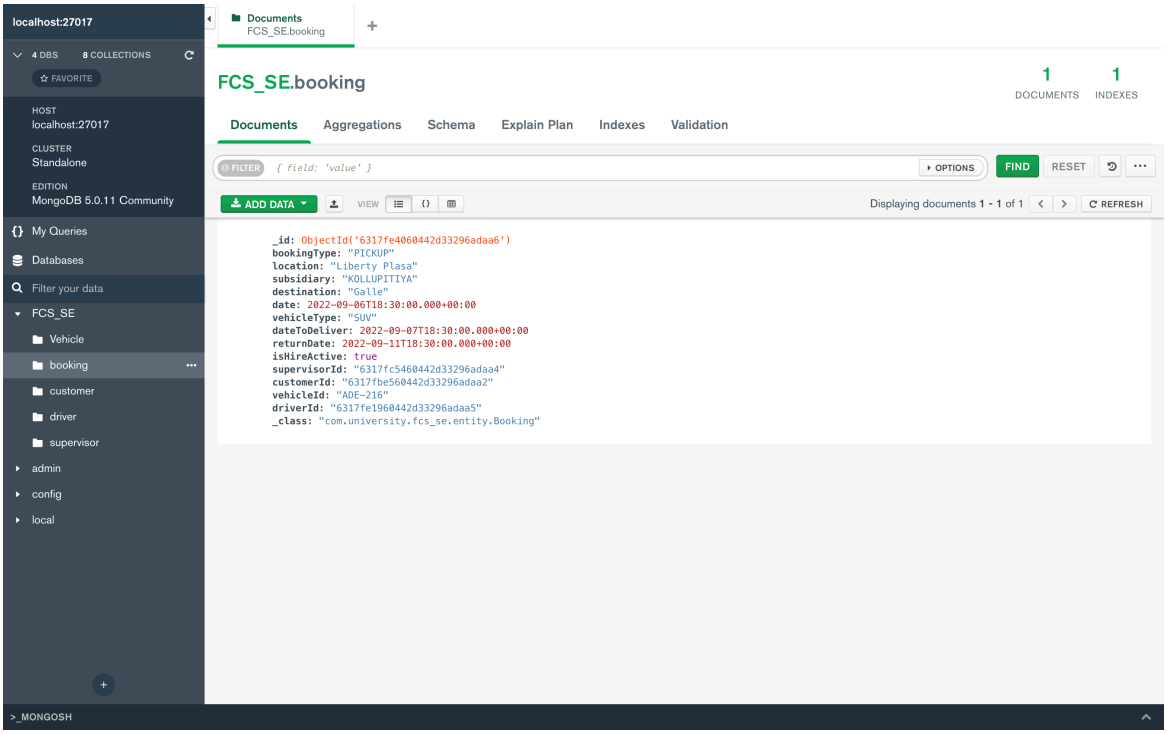


Figure 4.6.2 Test Case 06 Evidences Data - Mongo Compass

Test Case	
Test Unit : PickUp Booking	Tester Rithika Hettiarachchi
Test Case ID : 07	Tester Type: Black Box Testing
Description : Searches the entire booking collection and filters out the bookings with type of PICKPUP	Test Execution Date: 7th September 2022
Title : Get PickingUP Bookings	Test Execution Time : 07:52

Table 4.8.1 Test Case 07

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request	-	[{ "bookingID": "6317fe4060442d33296adaa6", "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }]	[{ "bookingID": "6317fe4060442d33296adaa6" , "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPI-TIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317f-be560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }]	Success	Bookings that have the booking type of pickup is returned

Table 4.8.2 Test Case 07 Steps

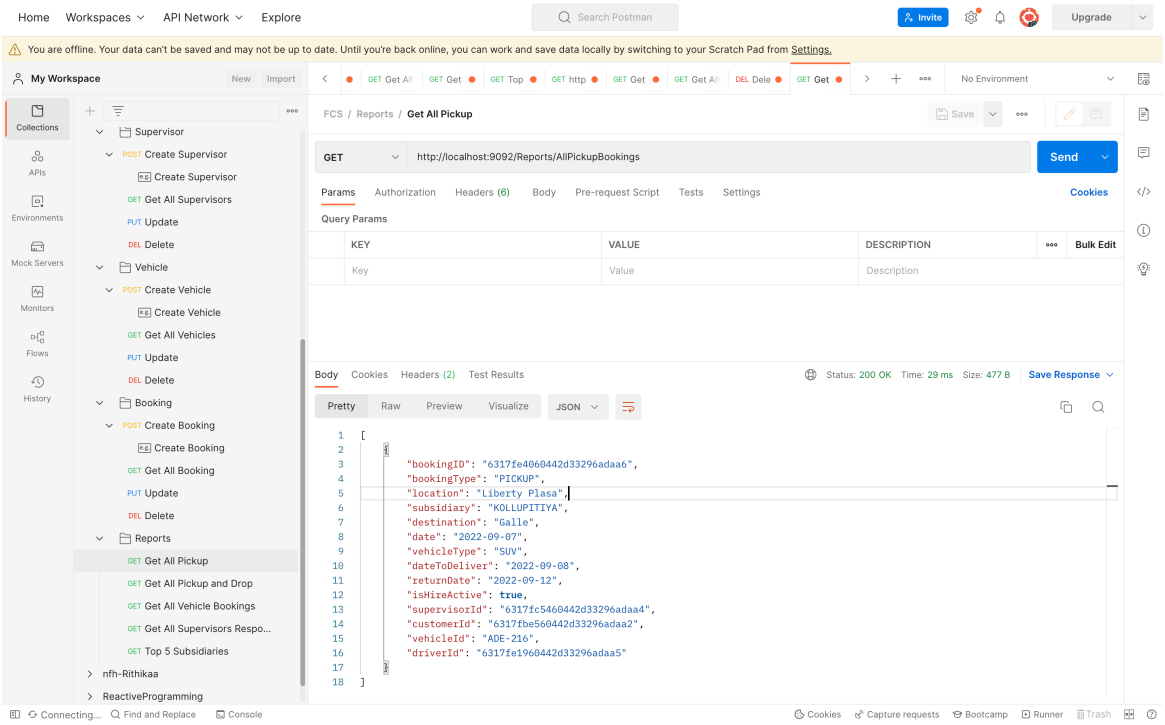


Figure 4.7 Test Case 07 Evidences Data - Postman

Test Case	
Test Unit : Pickup And Drop Bookings	Tester Rithika Hettiarachchi
Test Case ID : 08	Tester Type: Black Box Testing
Description : Searches the entire booking collection and filters out the bookings with type of PICKPUPANDDROP	Test Execution Date: 7th September 2022
Title : Get all PICKPUPANDDROP Bookings	Test Execution Time : 08:02

Table 4.9.1 Test Case 08

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request	-	[]	[]	Success	As I have not entered any data with a booking pickup and drop booking types a null is returned

Table 4.9.2 Test Case 08 Steps

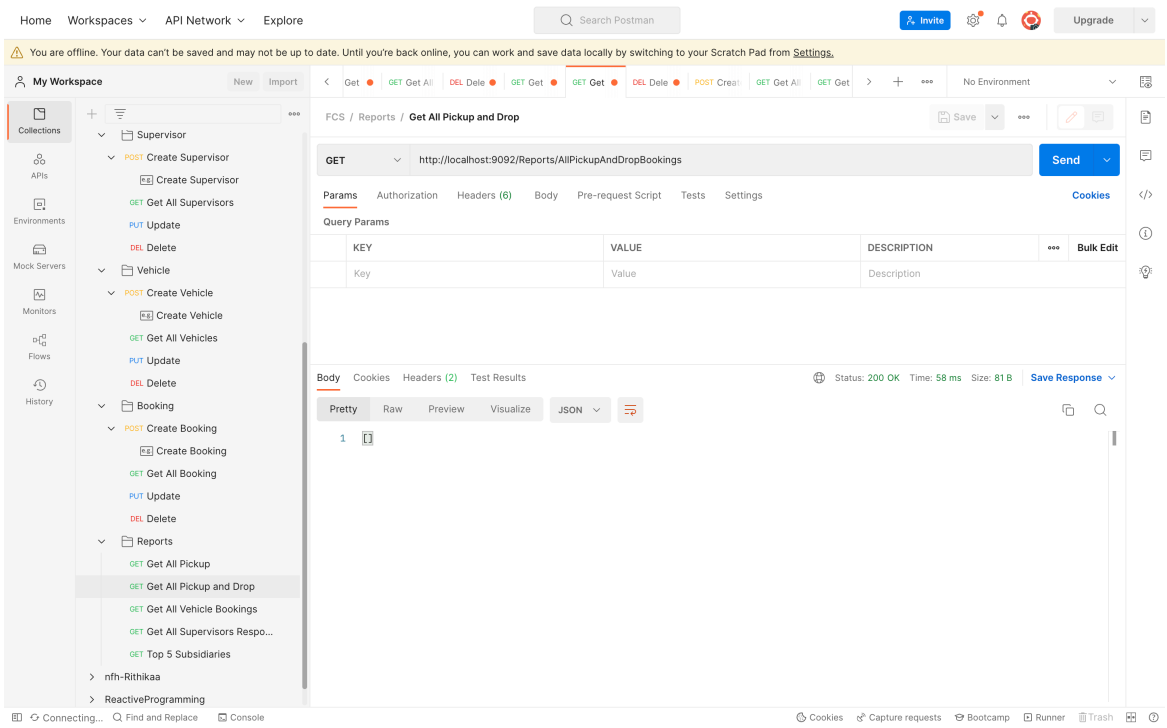


Figure 4.8 Test Case 08 Evidences Data - Postman

Test Case	
Test Unit : Grouping bookings by Supervisor	Tester Rithika Hettiarachchi
Test Case ID : 09	Tester Type: Black Box Testing
Description : returning a result of all the bookings a supervisor is responsible about	Test Execution Date: 7th September 2022
Title : Grouping bookings by Supervisor	Test Execution Time : 08:52

Table 4.10.1 Test Case 09

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Request	-	{ "6317fc5460442d33296adaa4": [{ "bookingID": "6317fe4060442d33296adaa6", "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }, { "bookingID": "63180d8660442d33296adaa7", "bookingType": "PICKUP", "location": "Ulife", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-14", "returnDate": "2022-09-18", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }] }	{ "6317fc5460442d33296adaa4": [{ "bookingID": "6317fe4060442d33296adaa6", "bookingType": "PICKUP", "location": "Liberty Plasa", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-08", "returnDate": "2022-09-12", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }, { "bookingID": "63180d8660442d33296adaa7", "bookingType": "PICKUP", "location": "Ulife", "subsidiary": "KOLLUPITIYA", "destination": "Galle", "date": "2022-09-07", "vehicleType": "SUV", "dateToDeliver": "2022-09-14", "returnDate": "2022-09-18", "isHireActive": true, "supervisorId": "6317fc5460442d33296adaa4", "customerId": "6317fbe560442d33296adaa2", "vehicleId": "ADE-216", "driverId": "6317fe1960442d33296adaa5" }] }	Success	As there are is only two bookings that I have entered under one supervisor, those are returned as a list inside a Map of supervisor and bookings

Table 4.10.2 Test Case 09 Steps

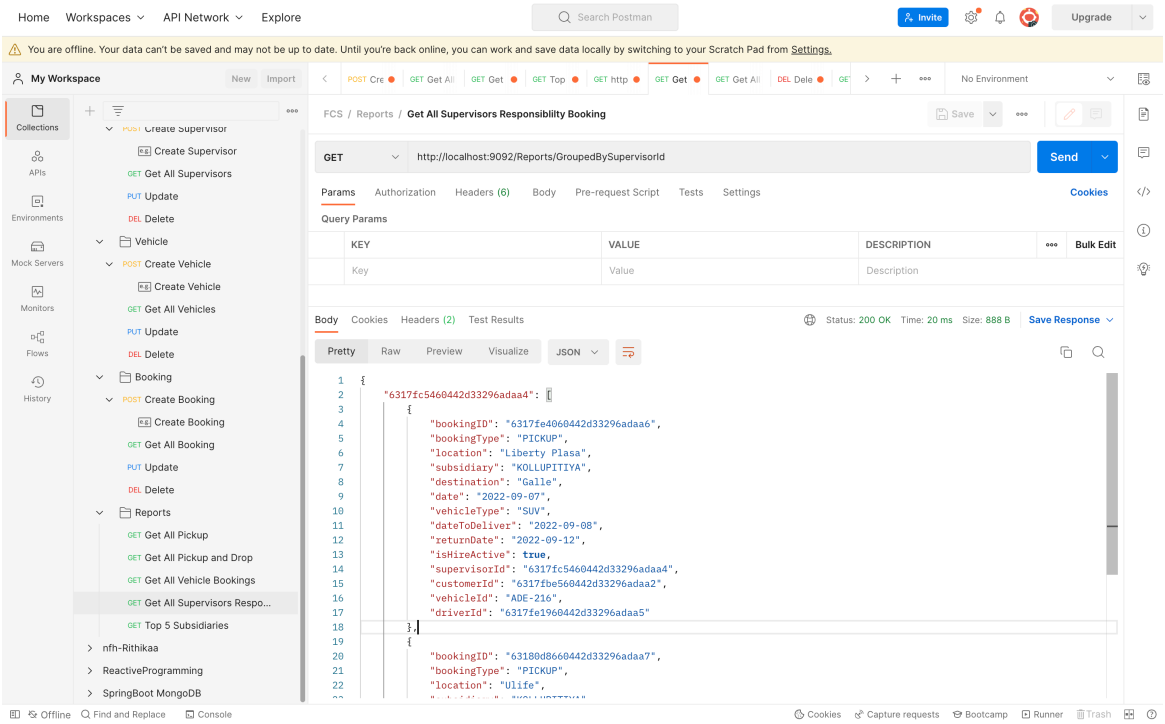


Figure 4.8 Test Case 09 Evidences Data - Postman

Test Case	
Test Unit : Get Top 5 Subsidiaries	Tester Rithika Hettiarachchi
Test Case ID : 10	Tester Type: Black Box Testing
Description : Searches the entire booking collection and collects the number of bookings under a subsidiaries within 100 days	Test Execution Date: 7th September 2022
Title : Get Top 05 subsidiaries	Test Execution Time : 09:21

Table 4.11.1 Test Case 10

Step No	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes

1	Request	-	{
			"KOLLUPITIYA": 2
			}
				{
				"KOLLUPITIYA": 2
				}
					Success	As I have only entered two subsidiaries under kollupitiya those data are will be returned

Table 4.11.2 Test Case 10 Steps

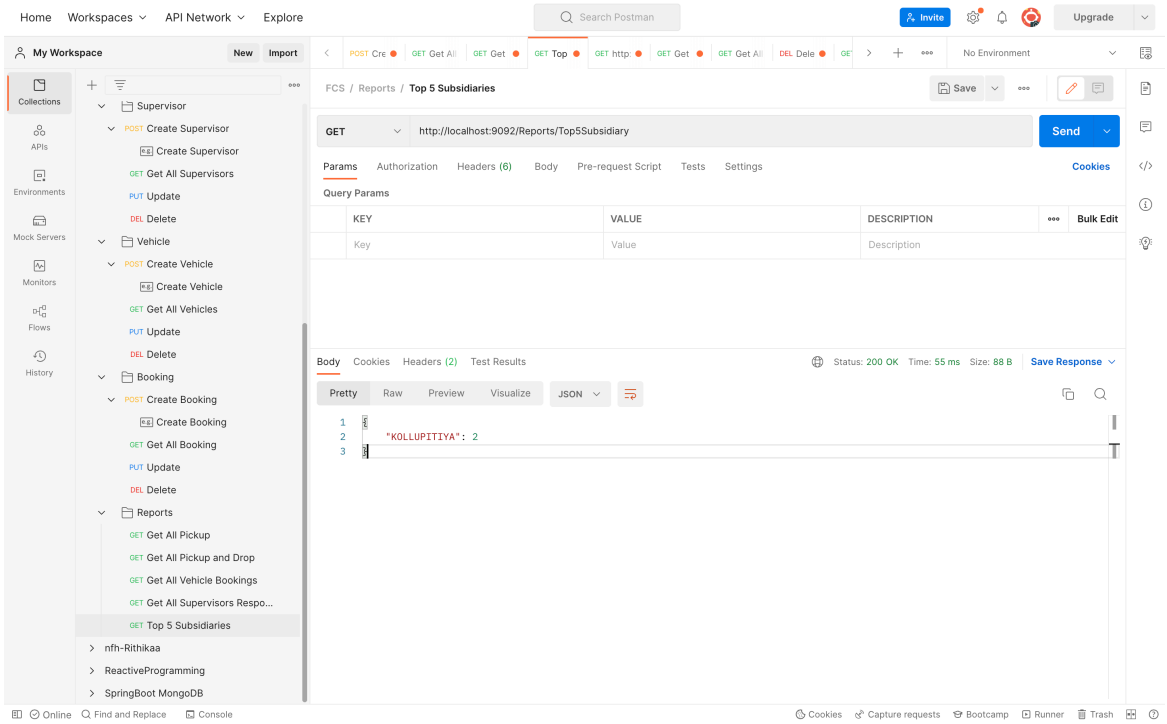


Figure 4.8 Test Case 10 Evidences Data - Postman

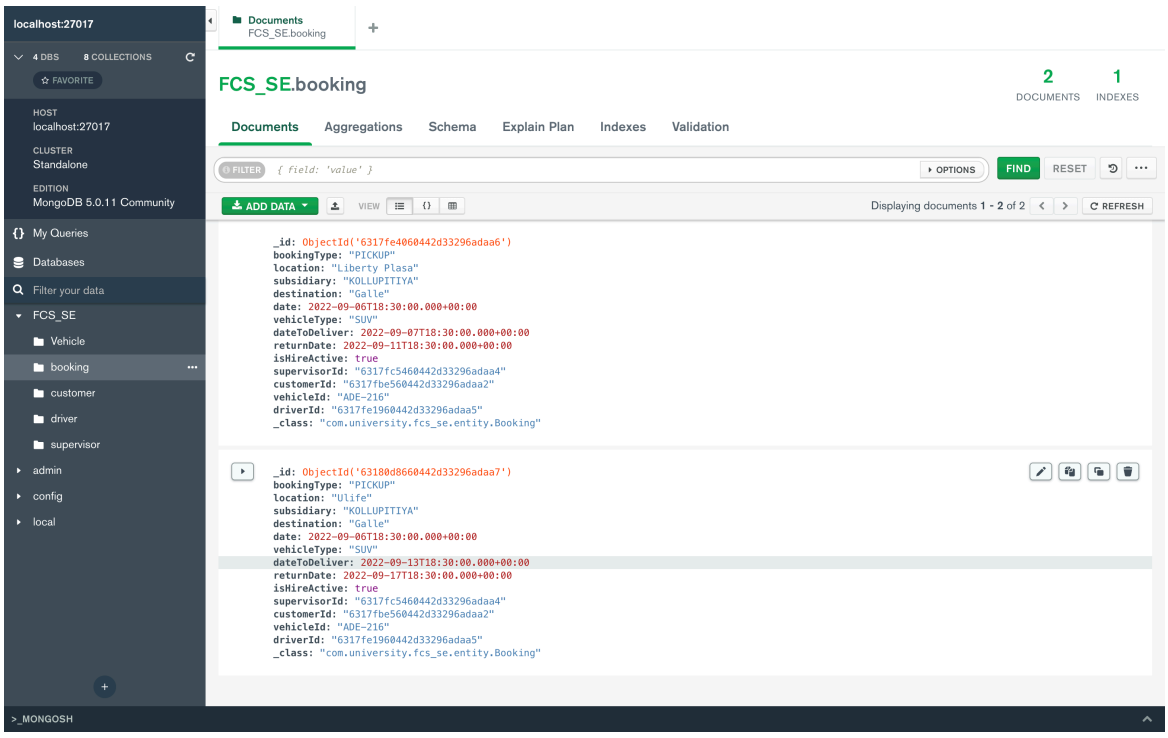


Figure 4.8 Test Case 10 Evidences Data - MongoCompass

4.4 user documentation

As the application contains only the backend using springBoot and the frontend is not yet build due to time limitation, I have instead a configured an API documentation which is available at localhost:9092/v3/swagger-ui.html

4.5 Software maintenance strategies

Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs. Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting issues or bugs, to boost performance, and more. The four different types of software maintenance are each performed for different reasons and purposes. A given piece of software may have to undergo one, two, or all types of maintenance throughout its lifespan.

The four types are:

- Corrective Software Maintenance
- Preventative Software Maintenance
- Perfective Software Maintenance
- Adaptive Software Maintenance

1. Corrective Software Maintenance

Corrective software maintenance is what one would typically associate with the maintenance of any kind. Correct software maintenance addresses the errors and faults within FCS applications that could impact various parts of the software, including the design, logic, and code. These corrections usually come from bug reports that were created by users or customers – but corrective software maintenance can help to spot them before your customers do, which can help your brand's reputation.

2. Preventative Software Maintenance

Preventative Software Maintenance helps to make changes and adaptations to FCS software so that it can work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. These services can include optimising code revamping the software and updating documentation as needed.

Preventative software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

3. Perfective Software Maintenance

Perfective software maintenance focuses on the evolution of requirements and features that existing in my system. As users interact with FCS applications, they may notice things that I might have not seen or suggest new features that they would like as part of the software, which could become future projects or enhancements. Perfective software maintenance takes over some of the work, both adding features that can enhance user experience and removing features that are not effective and functional. This can include features that are not used or those that do not help you to meet your end goals.

4. Adaptive Software Maintenance

Adaptive software maintenance becomes important when the environment of your software changes. This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system. Sometimes, adaptive software maintenance reflects organisational policies or rules as well. Updating services, making modifications to vendors, or changing payment processors can all necessitate adaptive software maintenance.

Task:5 Project Management

5.1 Importance

Software project management focuses on developing a product that will have a positive effect on an organisation. Without project management, a software development team may begin working on a project without any clear vision or guidance, resulting in more frequent errors and confusion.

Part of software project management involves making everyone involved aware of the purpose of the project and what steps are required to meet the end goal. Project management involves the use of various policies, procedures and principles to plan, implement and complete a project. To ensure a satisfactory result, projects must begin with specified parameters designed to produce the desired outcome.

Project Management can be divided into 5 phases:

1. **Initiating :** all the relevant project stakeholders will analyse the business case to decide whether or not to give the project the green light. If they all agree on starting the project, it's time to create a project charter, otherwise known as the project initiation document (PID). This text outlines the goal and requirements of the project
2. **Planning :** The second phase is critical to the project's success because it concentrates on developing the roadmap. This phase usually begins with setting the project objectives
3. **Executing :** This is the process where the team develops and completes project deliverables. preceded by a kickoff meeting where all the teams involved in the project are informed about their responsibilities. During this phase, the Project Manager needs to develop the team, assign resources, and execute the project management plan.
4. **Monitoring and Controlling :** ensuring that everything is closely aligned with the previously defined project management plan
5. **Closing :** represents the project in its completed state

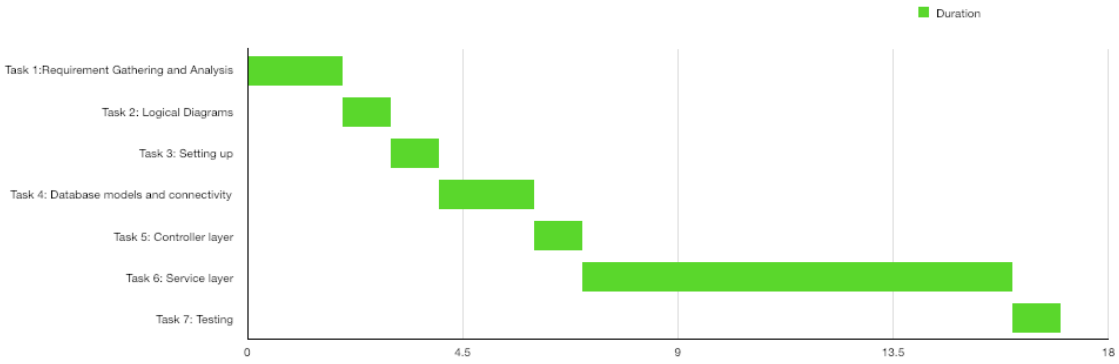


Figure 5.1 Grant chart of FCS