

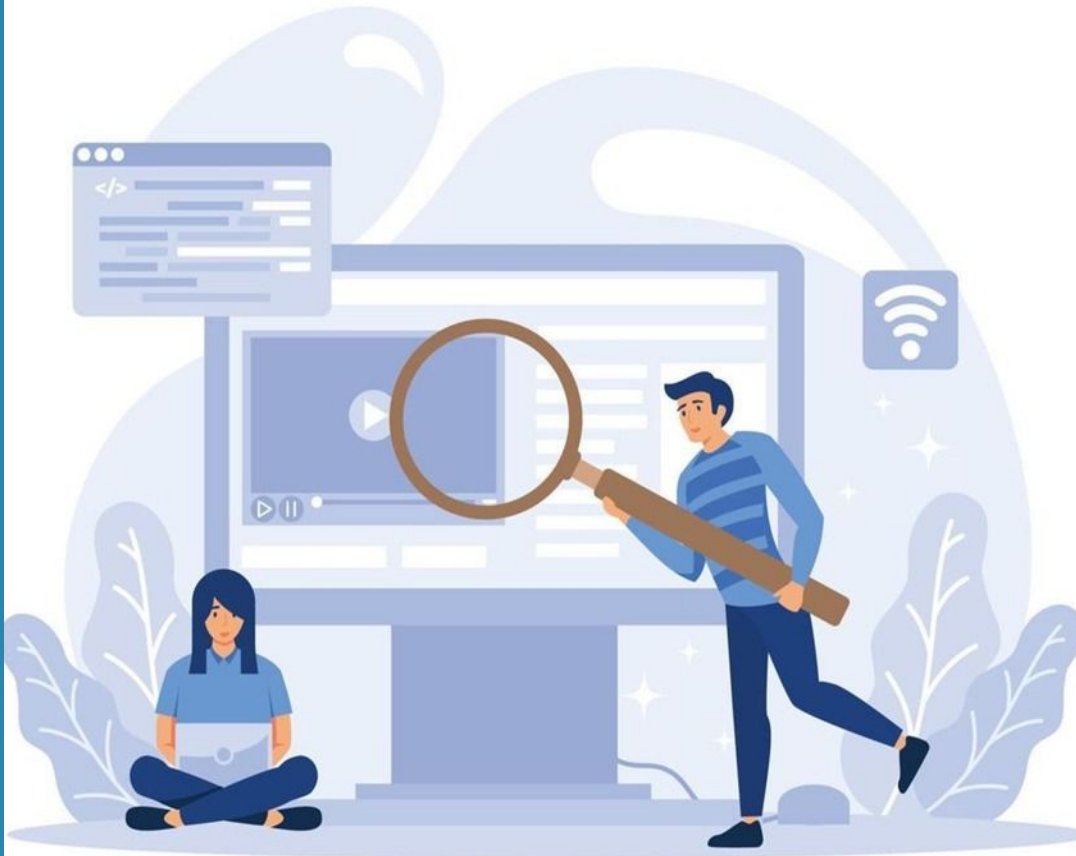


ONLINE BUS BOOKING SYSTEM

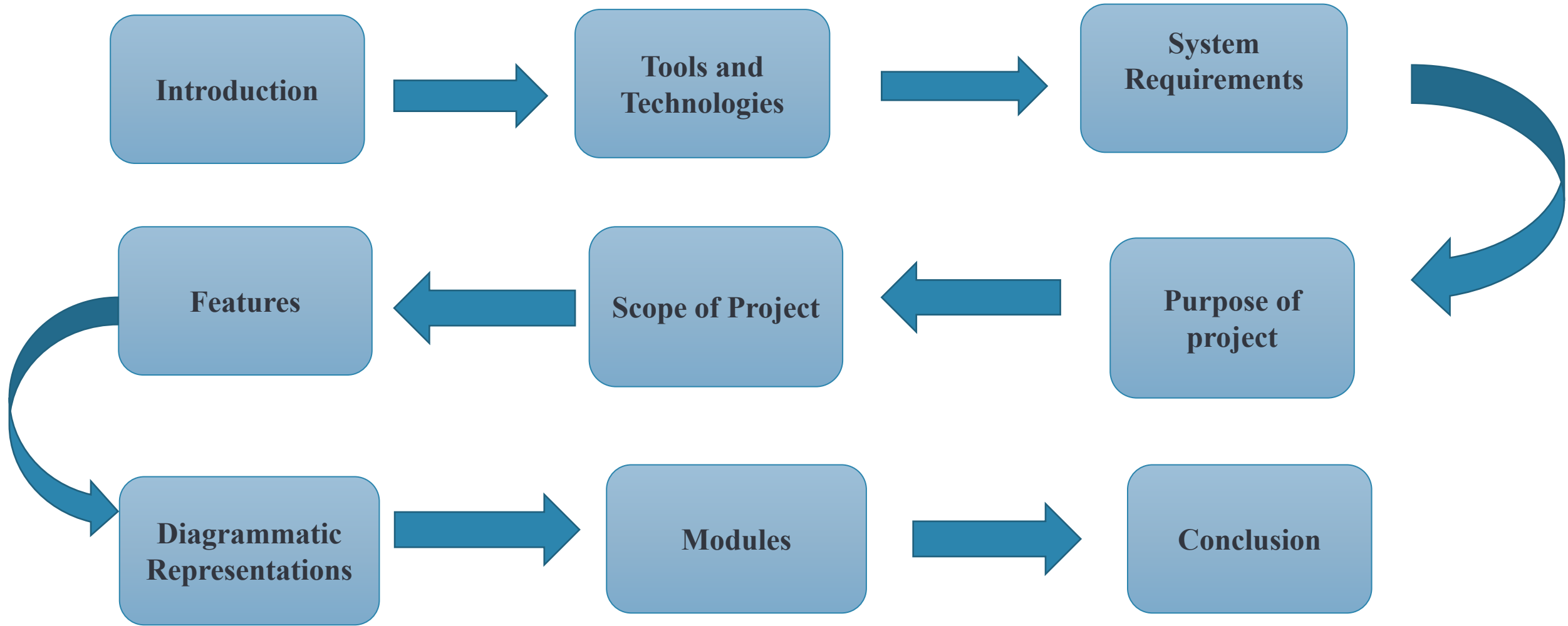
-GROUP 6

Group-6

- Harsh Rathod -2563088
- Swaroop A V -2562956
- Sulagna Chatterjee -2563990
- Priya Singha -2564017
- Sinchana Rao -2564007
- Pavithra S -2564183
- Rajalakshmi H -2564069
- V Kishen Kumar -2564315



Contents



Tools And Technologies

Front End

- ☐ Angular
- ☐ Html
- ☐ CSS
- ☐ BOOTSTRAP/ TailwindCSS

Back End

- ☐ MySQL
- ☐ Spring Boot
- ☐ Postman
- ☐ Rapid API

System Requirements

The Basic System Requirements for Running this project are listed below:

- ☐ Visual studio, Eclipse, Postman and MySQL are to be installed to the system
- ☐ 1 GB Random Access Memory
- ☐ 200 MB of Free Space on Hard Disk
- ☐ Microsoft Windows 8,10,11 or Linux or Equivalent OS
- ☐ Web Browser (Microsoft Edge, Google Chrome, Firefox)





Introduction

INTRODUCTION

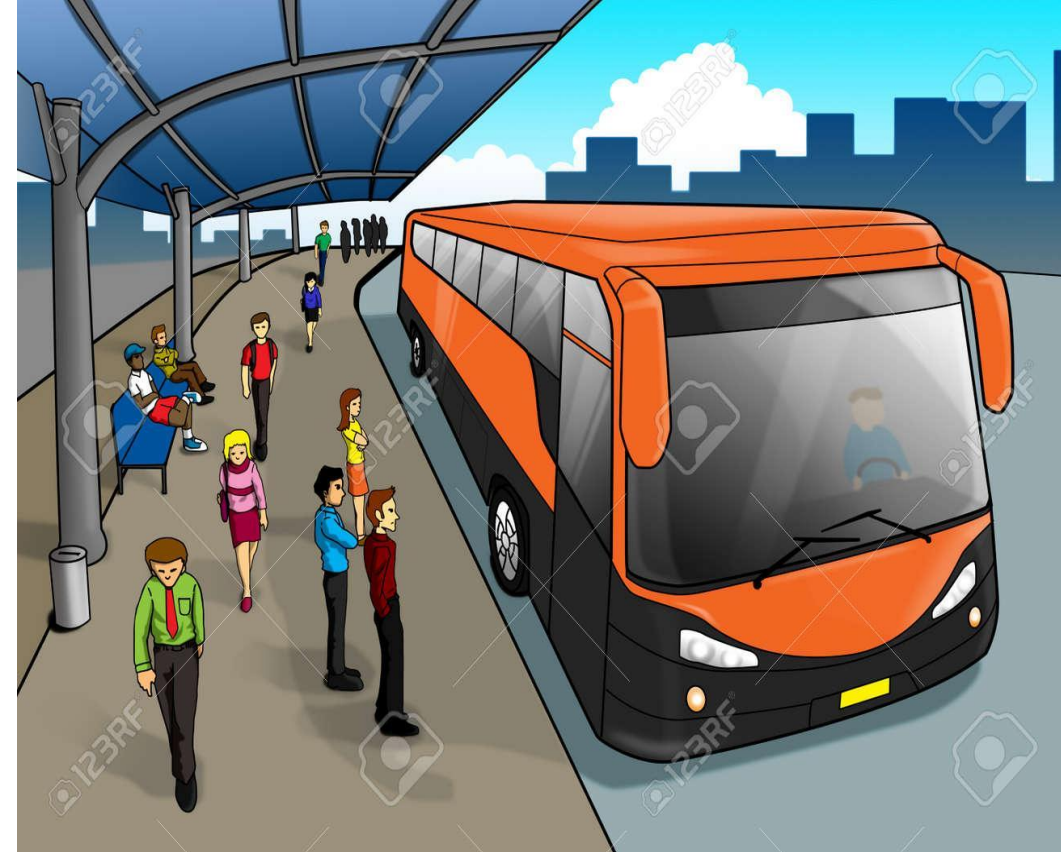
- An online bus booking system is a web-based application that allows users to book bus tickets and manage their reservations. This system is built using Spring Boot Java, Angular, and Junit, which are popular technologies for developing robust and scalable web applications.
- A bus booking system is a mobile or web software solution designed to provide customers with a personalized easy-to-utilize user experience for booking and purchasing tickets online. It stores customers personal data records, scheduled routes, frequent trips, drop points, and other information

PURPOSE:

- An online bus booking system serves to streamline the process of reserving bus tickets, making it convenient for passengers to search for routes, select seats. It benefits both passengers and bus operators by offering real-time information, reducing manual effort, and improving overall efficiency.
- It will allow the passengers to enjoy the booking of bus tickets from the present position through the internet. They will be provided with the bus routes as they will be provided with the bus routes along with some other facilities like booking the tickets based on their comfort level, the time of arrival and departure, and canceling the tickets.
- The administrator can handle various aspects like applying the offers, changing the facilities according to price, can monitor various other things. The Travel Agency can also use this application for managing their ticket booking service.

Scope

- ❑ This Bus Ticket Booking application can be used by any Travel Agent to issue tickets to customers.
- ❑ It also helps the customer to enquire about the availability of seats in a particular bus at a particular date from a particular location.
- ❑ It will also provide the facility to check the timings and schedule of the buses along with the ticket price.



Modules

- ☐ Home Module
- ☐ Admin Register Module
- ☐ Admin Login Module
- ☐ Location Module
- ☐ Bus Module
- ☐ User Register Module
- ☐ User Login Module
- ☐ Booking Module



Features Available

Admin

- Home
- Register
- Login
- Schedule
- Booked List
- Maintenance

User

- Home
- Register
- Login
- Booking

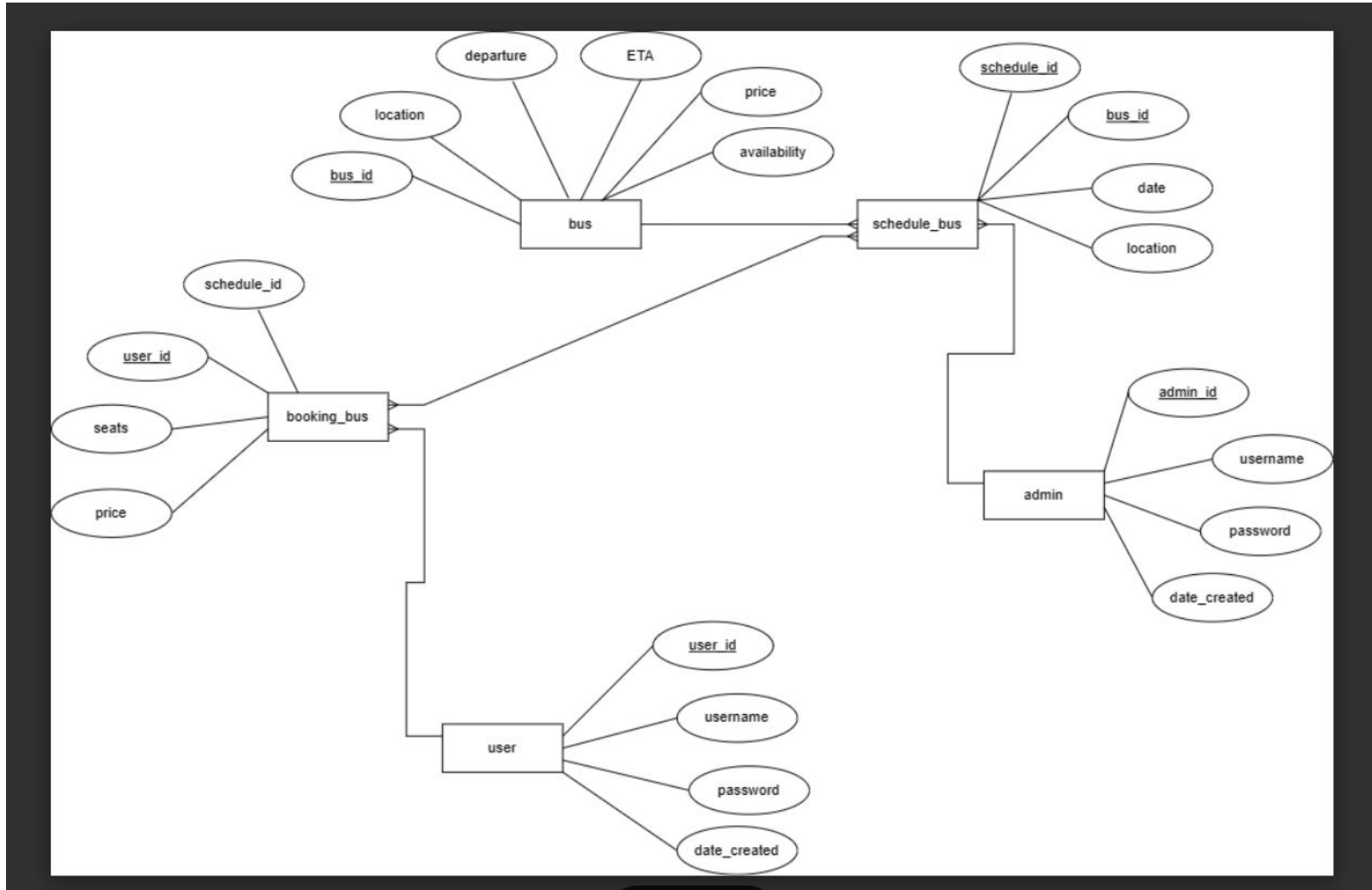
Bus

- Bus ID
- Availability
- Location
- ETA
- Price
- Departure

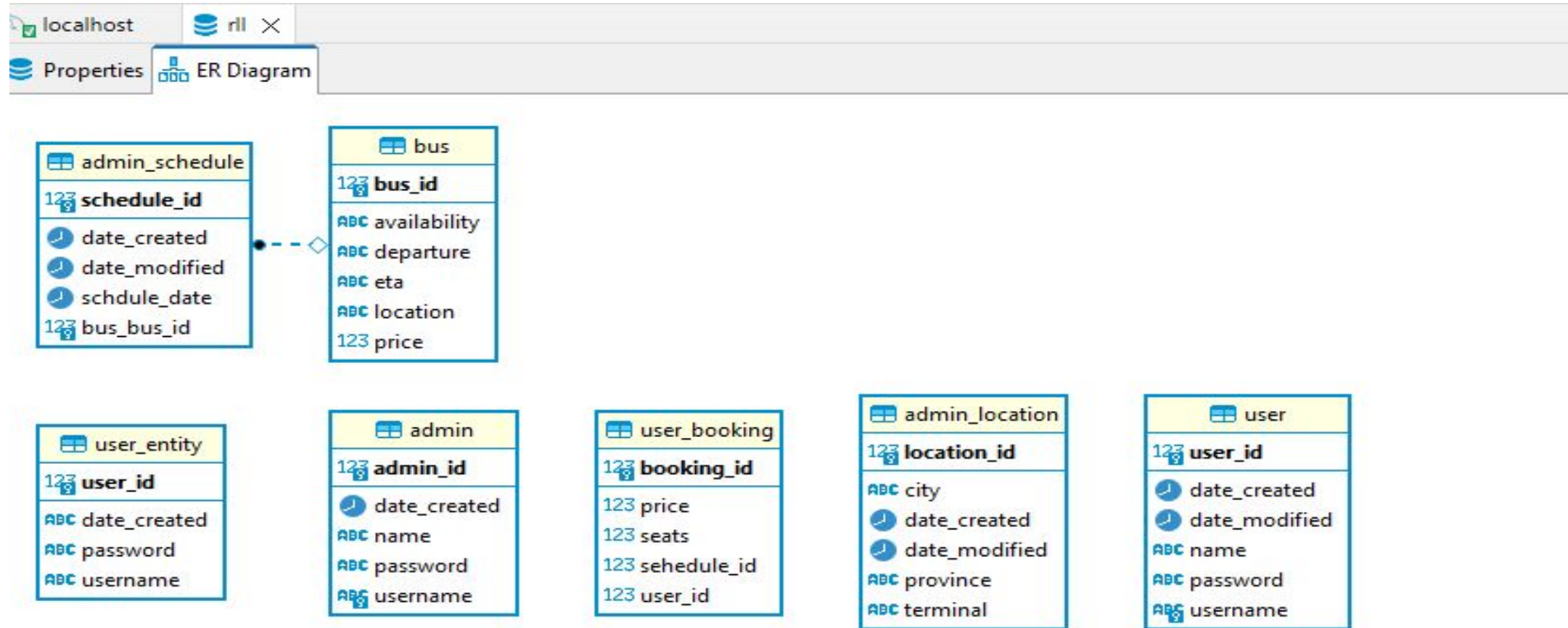
Booking

- Price
- Schedule ID
- User ID
- Seats
- Action – Edit and Delete

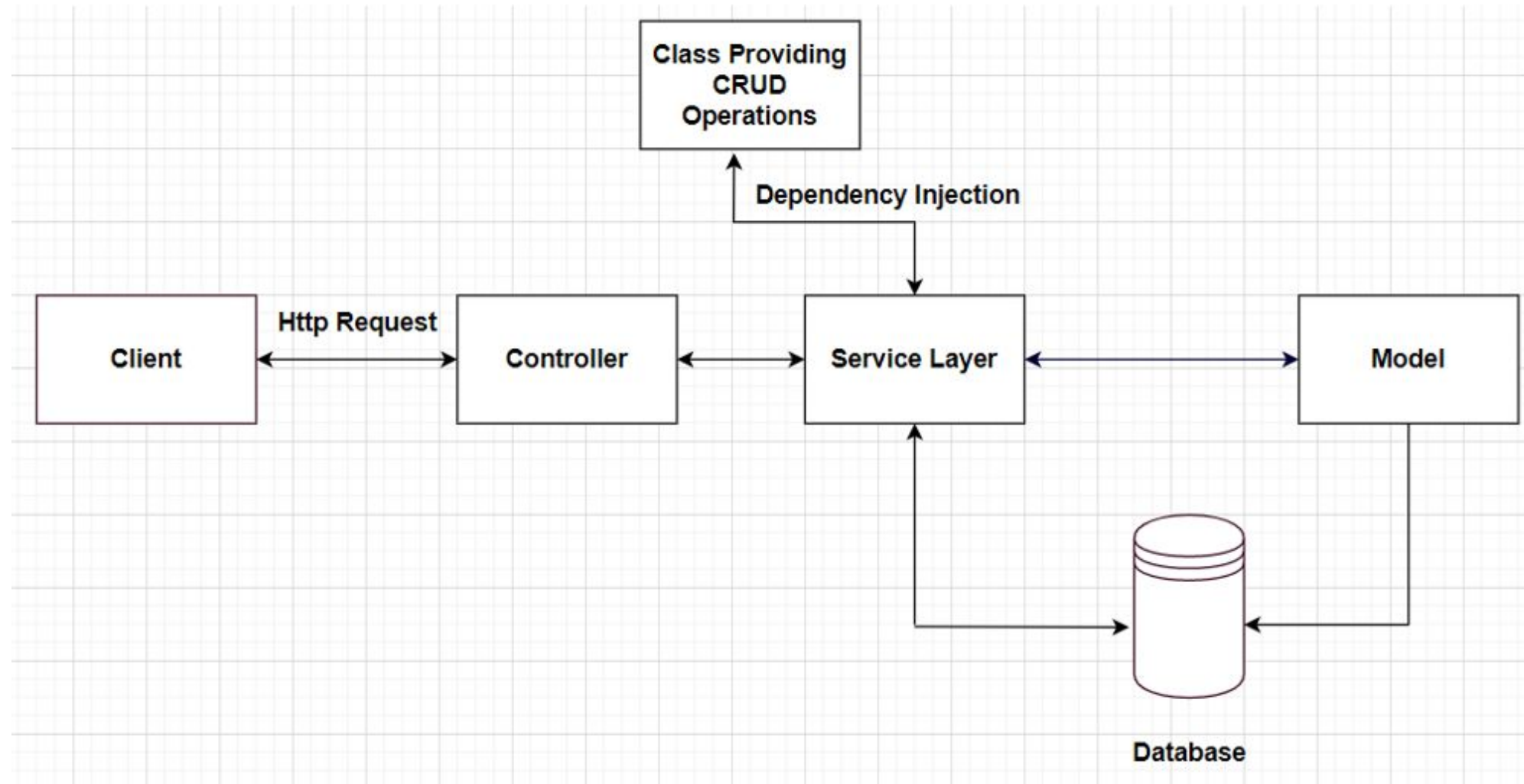
Schema Diagram



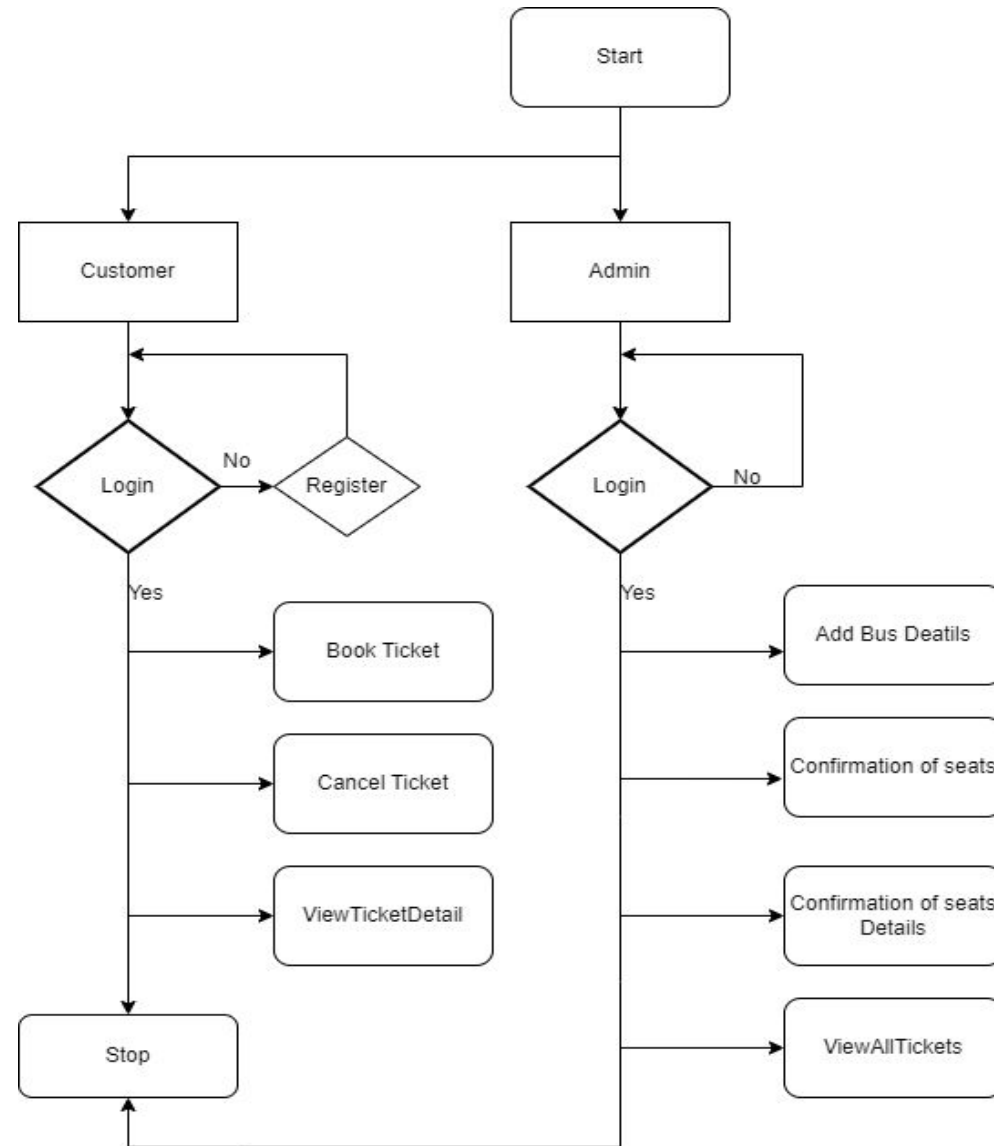
Database Schema



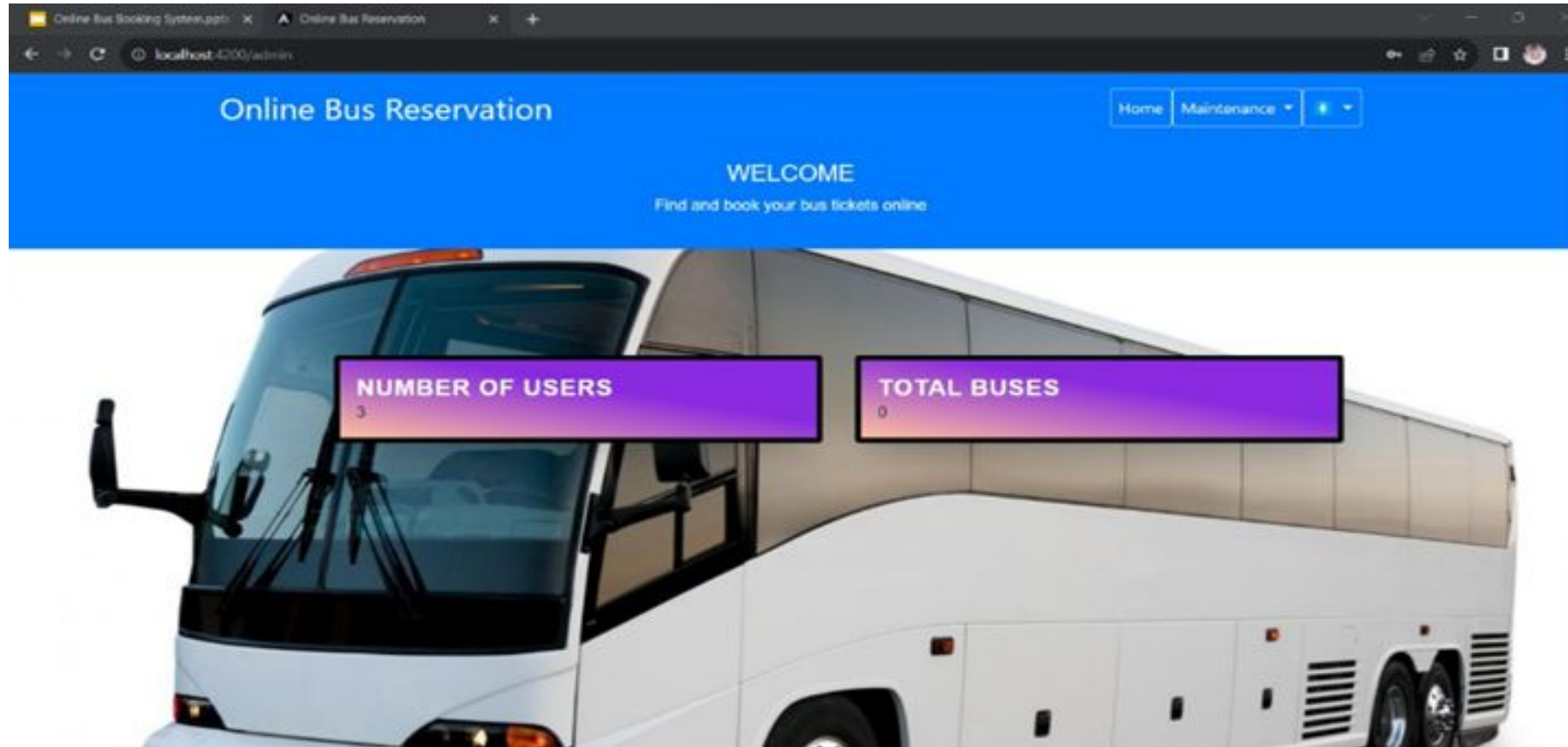
Spring Boot



Flow Chart



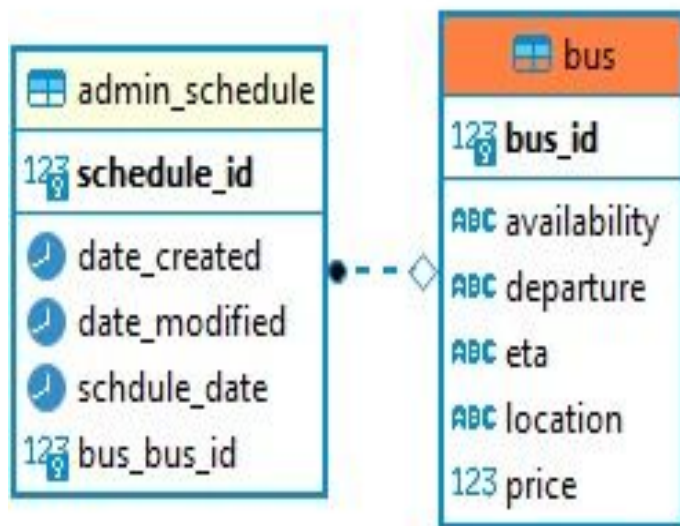
ADMIN HOME PAGE



FRONTEND - LOGIC

```
export class HomeComponent implements OnInit {  
  totalNumberOfBuses: Number;  
  totalNumberOfUsers: Number;  
  constructor(private homeService: HomeService) {  
    this.totalNumberOfBuses = 0;  
    this.totalNumberOfUsers = 0;  
  }  
  ngOnInit(): void {  
    this.homeService.totalNumberOfBuses().subscribe((totalNumberOfBuses) => {  
      this.totalNumberOfBuses = totalNumberOfBuses;  
    });  
    this.homeService.totalNumberOfUsers().subscribe((totalNumberOfUsers) => {  
      this.totalNumberOfUsers = totalNumberOfUsers;  
    });  
  }  
}
```

Database Schema



Column Name	#	Data Type	Not Null	Auto Increment	Key
bus_id	1	bigint	[v]	[v]	PRI
availability	2	varchar(255)	[]	[]	
departure	3	varchar(255)	[]	[]	
eta	4	varchar(255)	[]	[]	
location	5	varchar(255)	[]	[]	
price	6	bigint	[]	[]	

BACKEND - LOGIC

```
@GetMapping("/totalNumberOfUsers")  
public ResponseEntity<Long> totalNumberOfUsers() {  
    return new ResponseEntity<Long>(this.userService.totalNumberOfUsers(),  
    HttpStatus.OK);
```

```
@GetMapping("/totalNumberOfBuses")  
public ResponseEntity<Long> totalNumberOfBuses() {  
    return new ResponseEntity<Long>(this.bs.totalNumberOfBuses(), HttpStatus.OK);
```

Backend - Rest API

GET ⌵ http://localhost:8080/bus/totalNumberOfBuses Send ➤

200 OK · 16 ms

Description Headers Query **Body** Auth Options ⚠

Text **JSON** JSON Tree Form URL-Encoded

Multipart GraphQL

Info Request **Response**

Headers Text JSON Tree **JSON Text** Raw

1 2

Pretty Print JSON

```
1 {"totalNumberOfBuses": " "}
```

GET ⌵ http://localhost:8080/user/totalNumberOfUsers Send ➤

200 OK · 44 ms

Description Headers Query **Body** Auth Options ⚠

Text **JSON** JSON Tree Form URL-Encoded

Multipart GraphQL

Info Request **Response**

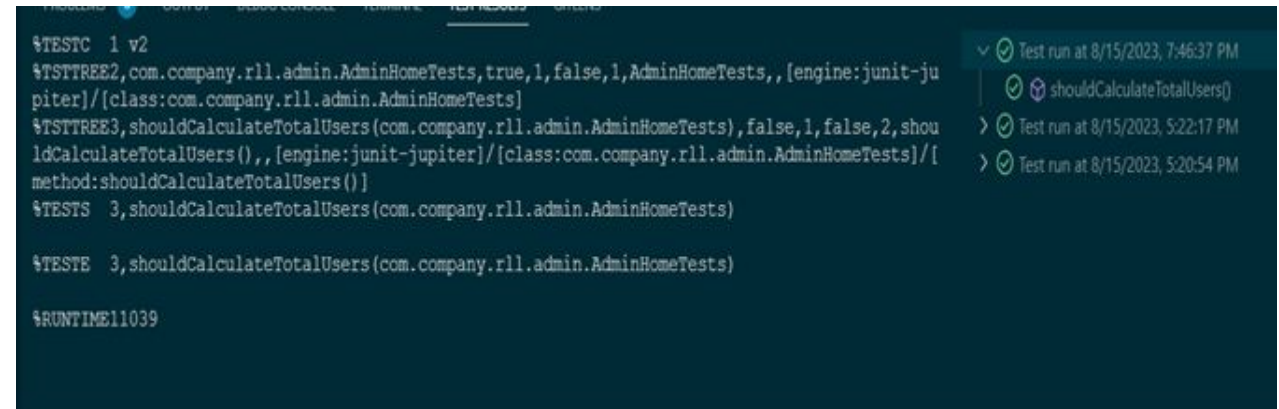
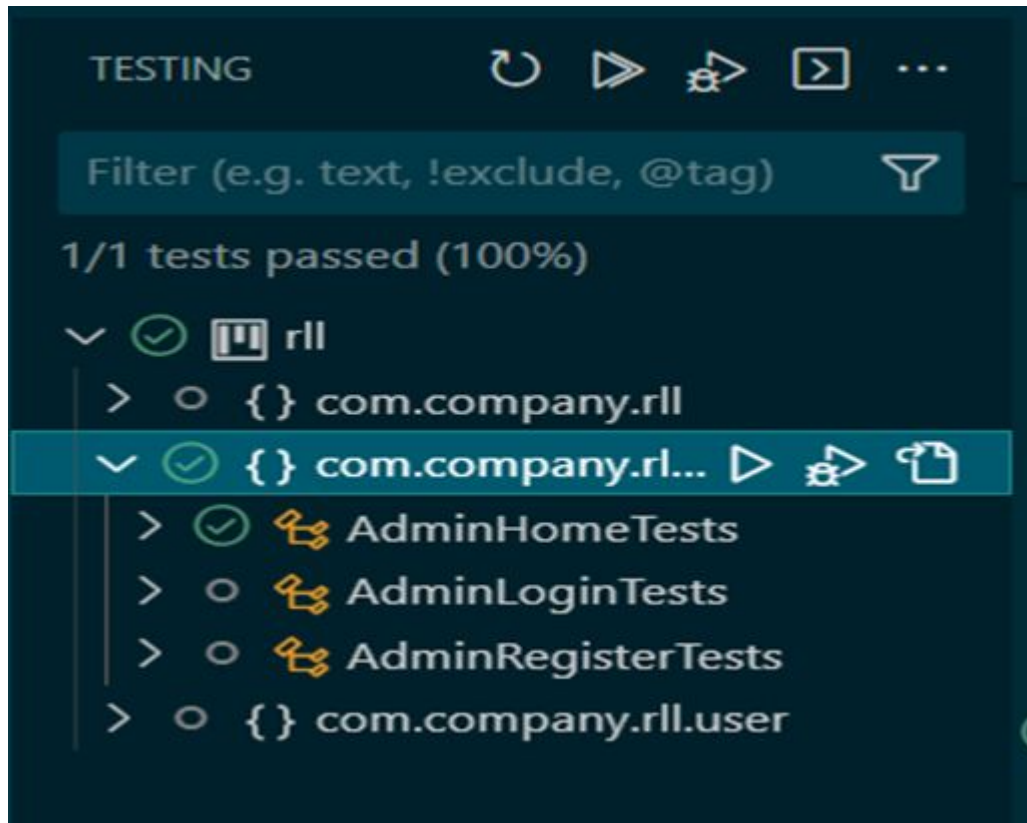
Headers Text JSON Tree **JSON Text** Raw

1 3

Pretty Print JSON

```
1 {"totalNumberOfUsers": " "}
```


Testing - Admin Home Page



Frontend - User Home Page



Database Schema



user	
123	user_id
🕒	date_created
🕒	date_modified
ABC	name
ABC	password
ABC	username

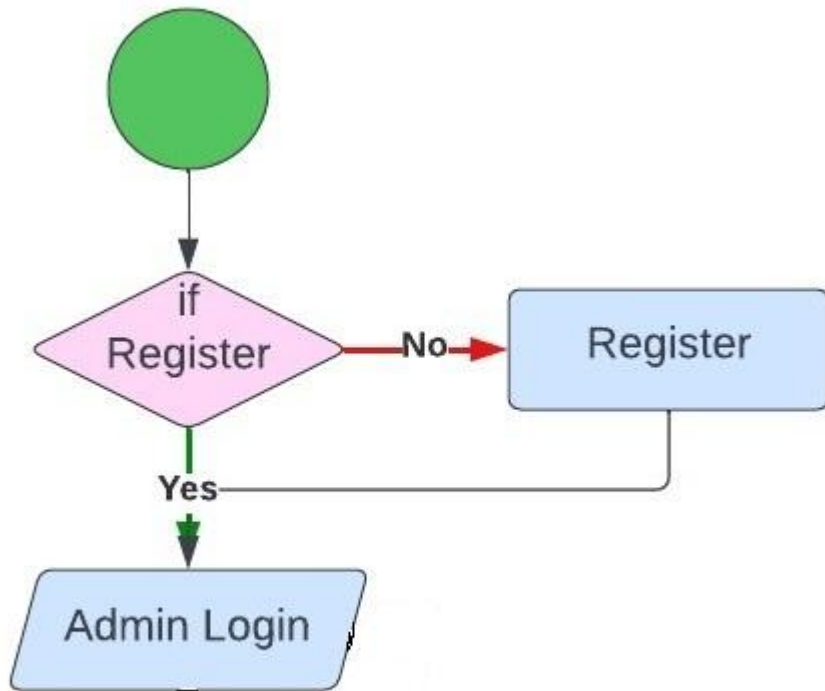
Column Name	#	Data Type	Not Null	Auto Increment	Key
123 user_id	1	bigint	[v]	[v]	PRI
🕒 date_created	2	datetime(6)	[]	[]	
🕒 date_modifi...	3	datetime(6)	[]	[]	
ABC name	4	varchar(255)	[]	[]	
ABC password	5	varchar(255)	[]	[]	
ABC username	6	varchar(255)	[]	[]	UNI

Purpose: To secure access as administrator.

- ❑ To use the bus booking system admin need to register by using Name as name, email as username and valid password as password.
- ❑ This module is carefully designed to ensure a seamless and secure registration experience, enabling efficient control over various aspects of the application

DATABASE

Flow Diagram



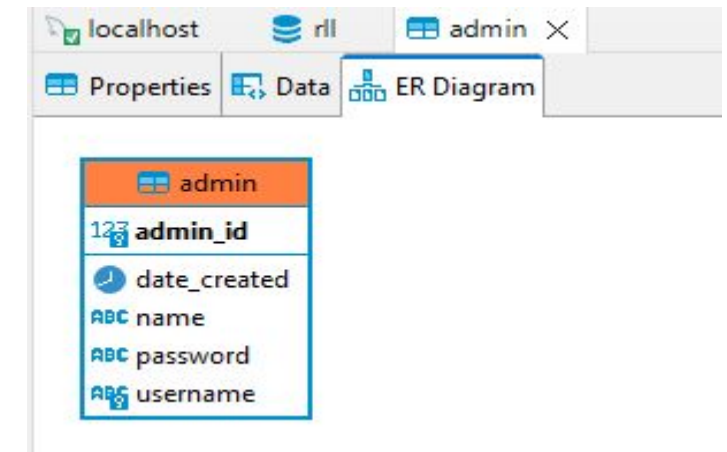
Table

Table: admin

Columns:

<u>admin_id</u>	bigint AI PK
date_created	datetime(6)
name	varchar(255)
password	varchar(255)
username	varchar(255)

Database Schema



Front-end: Admin Register Page

Registration Form

Name

Email

Password

Registration Form

Name

Email

Password

Back-end:Rest-Api

The screenshot displays a REST client interface with a dark theme. On the left, the 'New Request' tab is active, showing a POST request to `http://localhost:8080/admin/register`. The 'Body' tab is selected, and the 'JSON' format is chosen. The JSON body is `{ "name": "sulagna chatterjee", "username": "sulu27@gmail.com", "password": "1234" }`. A 'Pretty Print JSON' button is visible. On the right, the 'Response' tab is active, showing a successful 201 Created response with a status of 897 ms. The response body is the same JSON object as the request. The interface includes tabs for Description, Headers, Query, Body, Auth, and Options, as well as buttons for Text, JSON, JSON Tree, Form URL-Encoded, Multipart, and GraphQL.

POST `http://localhost:8080/admin/register` Send

Description Headers Query Body Auth Options

Text **JSON** JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {  
2   "name": "sulagna chatterjee",  
3   "username": "sulu27@gmail.com",  
4   "password": "1234"  
5 }
```

201 Created · 897 ms

Info Request Response

Headers Text JSON Tree **JSON Text** Raw

```
1 {  
2   "name": "sulagna chatterjee",  
3   "username": "sulu27@gmail.com",  
4   "password": "1234"  
5 }
```

Filter expression...

Testing: Admin Register page

TESTING

Filter (e.g. text, !exclude, @tag)

2/2 tests passed (100%)

- ✓ rll 3.5s
 - > ○ {} com.company.rll
 - ✓ {} com.company.rll.admin 3.5s
 - > ○ AdminHomeTests
 - > ○ AdminLoginTests
 - ✓ AdminRegisterTests 3.5s
 - ✓ shouldRegisterSampleAdmin() 3.4s
 - ✓ shouldDeleteSample...
 - > ○ {} com.company.rll.user

PROBLEMS 6 OUTPUT TEST RESULTS TERMINAL GITLENS DEBUG CONSOLE

```
%TESTC 2 v2
%TSTTREE2,com.company.rll.admin.AdminRegisterTests,true,2,false,1,AdminRegisterTests,,[engine:junit-jupiter]/[class:com.company.rll.admin.AdminRegisterTests]
%TSTTREE3,shouldRegisterSampleAdmin(com.company.rll.admin.AdminRegisterTests),false,1,false,2,shouldRegisterSampleAdmin(),,[engine:junit-jupiter]/[class:com.company.rll.admin.AdminRegisterTests]/[method:shouldRegisterSampleAdmin()]
%TSTTREE4,shouldDeleteSampleAdmin(com.company.rll.admin.AdminRegisterTests),false,1,false,2,shouldDeleteSampleAdmin(),,[engine:junit-jupiter]/[class:com.company.rll.admin.AdminRegisterTests]/[method:shouldDeleteSampleAdmin()]
%TESTS 3,shouldRegisterSampleAdmin(com.company.rll.admin.AdminRegisterTests)

%TESTE 3,shouldRegisterSampleAdmin(com.company.rll.admin.AdminRegisterTests)

%TESTS 4,shouldDeleteSampleAdmin(com.company.rll.admin.AdminRegisterTests)

%TESTE 4,shouldDeleteSampleAdmin(com.company.rll.admin.AdminRegisterTests)

%RUNTIME27948
```

Git Commit

```
PROBLEMS 6 OUTPUT TERMINAL GITLENS DEBUG CONSOLE

commit bc6927a4136ce98d431710aca77d5fa26bdbe2c3
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Sat Aug 12 10:46:16 2023 +0530

    update test

commit f6fb64f6e5d00b414a6dfae1da33f0d13652162c
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Thu Aug 10 15:06:37 2023 +0530

    :...skipping...
commit bc6927a4136ce98d431710aca77d5fa26bdbe2c3
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Sat Aug 12 10:46:16 2023 +0530

    update test

commit f6fb64f6e5d00b414a6dfae1da33f0d13652162c
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Thu Aug 10 15:06:37 2023 +0530

    adminregistration testing

commit 3c9631cb0e4c76e521166161d3864a4497e30307
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Thu Aug 10 12:25:12 2023 +0530

    new update

commit 56f803a086e3c5210d61d9fd131c8c6f8f2b9f0f
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Wed Aug 9 21:29:10 2023 +0530

    update

commit 412ec446755a3a4ad87bd9d249efb0fec9bf6efb
Author: sulagna <sulagnachatterjee27@gmail.com>
Date: Wed Aug 9 10:49:32 2023 +0530

    new update

commit 5e1bdbe7d33341211ccd2994a041f6cd6940dd91
Author: sulagna <sulagnachatterjee27@gmail.com>
```

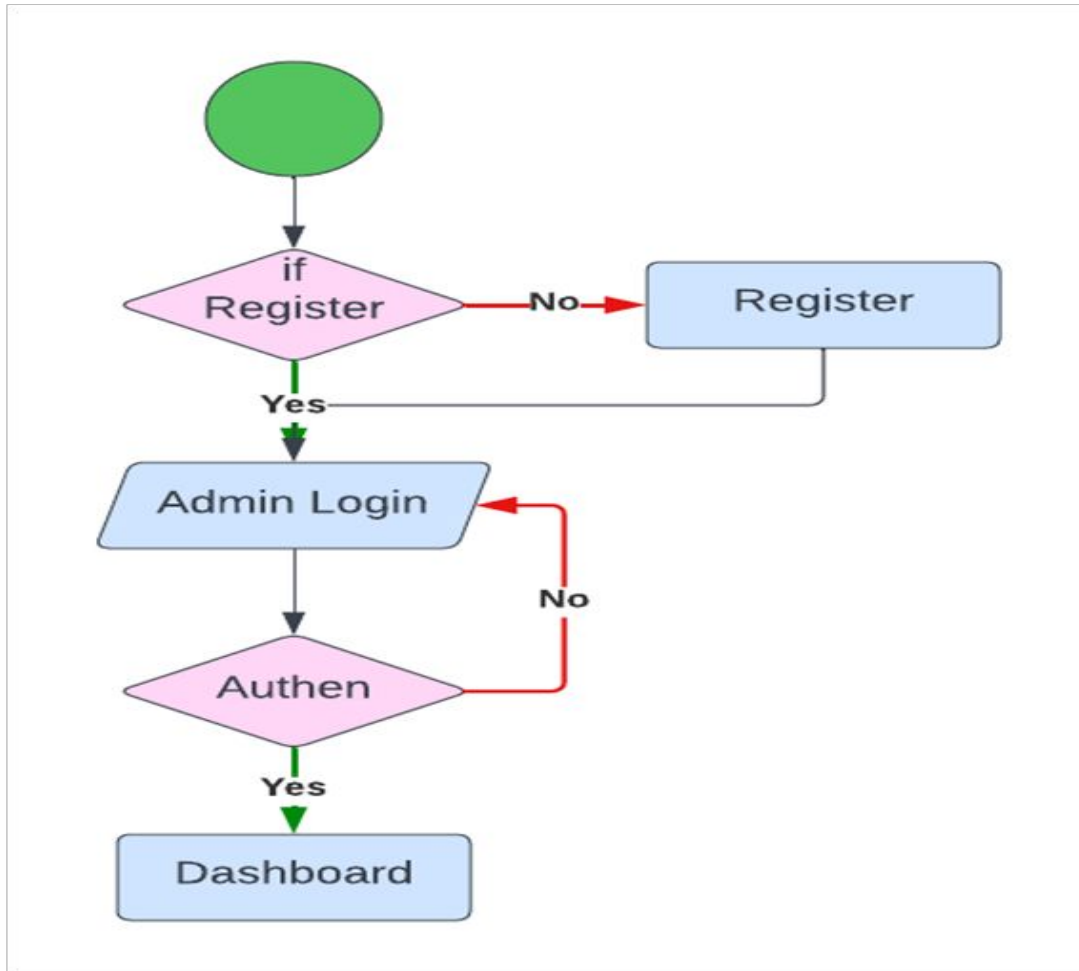

Admin Login Module

Purpose: Secure access for administrators to the admin dashboard

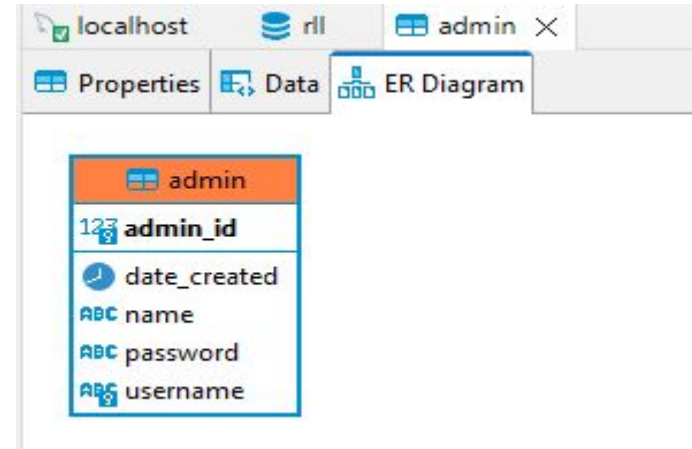
- ❑ To use the system admin have to log in to the system by entering the admin username and password to get access to the system.
- ❑ This module is carefully designed to ensure a seamless and secure login experience, enabling efficient control over various aspects of the application

Database

Flow Diagram



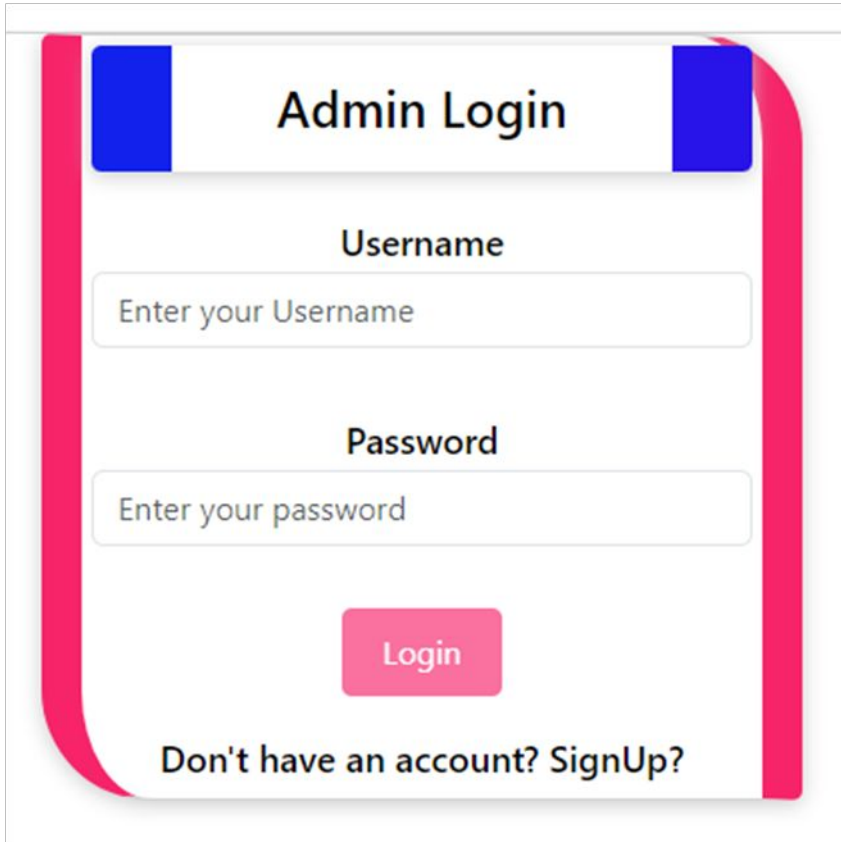
Database Schema



Table

	Column Name	#	Data Type	Not Null	Auto Increment	Key
Columns	admin_id	1	bigint	[v]	[v]	PRI
Constraints	date_created	2	datetime(6)	[]	[]	
Foreign Keys	name	3	varchar(255)	[]	[]	
References	password	4	varchar(255)	[]	[]	
Triggers	username	5	varchar(255)	[]	[]	UNI
Indexes						
Statistics						

Fronted: Admin Login page



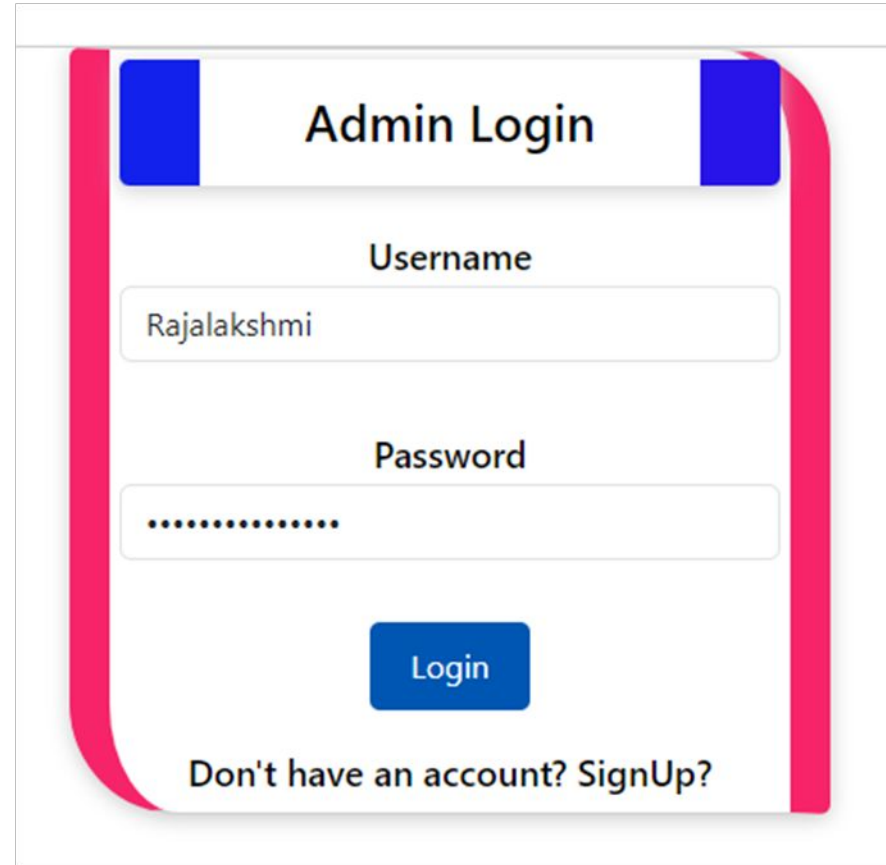
Admin Login

Username

Password

Login

Don't have an account? [SignUp?](#)



Admin Login

Username

Password

Login

Don't have an account? [SignUp?](#)

Admin Login page

Frontend Logic

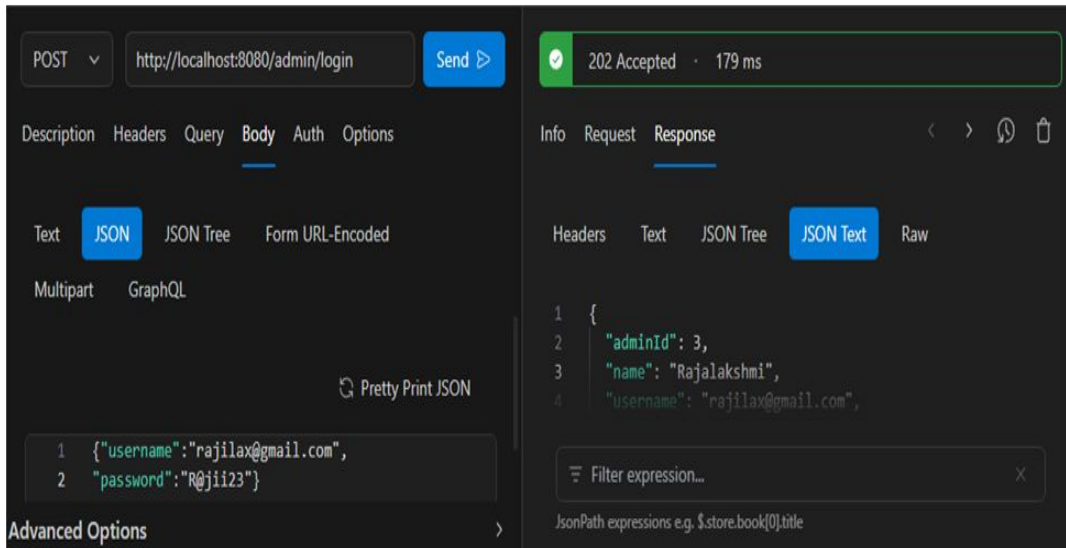
```
login(admin: NgForm) {  
  this.LoginService.login(admin.value).subscribe({  
    next: (admin) => {  
      if (admin != null) {  
        this.adminService.isLoggedIn = true;  
        this.router.navigate(['/admin']);  
      }  
    },  
    error: (err: Error) => {  
      alert(err.name + "\n" + err.message);  
    }  
  });  
}
```

Backend Logic

```
public AdminEntity login(AdminEntity adminEntity) {  
  Optional<AdminEntity> result = adminRepository.findByUsername(  
    adminEntity.getUsername()  
  );  
  if (result.isPresent()) {  
    if (adminEntity.getPassword().equals(result.get().getPassword())) {  
      return result.get();  
    }  
  }  
  return null;  
}
```

Backend : Rest API

Login Success



This screenshot shows a successful POST request to `http://localhost:8080/admin/login` in Postman. The status is `202 Accepted` with a response time of `179 ms`. The request body is a JSON object: `{ "username": "rajilax@gmail.com", "password": "R@jii23" }`. The response body is a JSON object: `{ "adminId": 3, "name": "Rajalakshmi", "username": "rajilax@gmail.com", ... }`.

```
POST http://localhost:8080/admin/login
```

202 Accepted · 179 ms

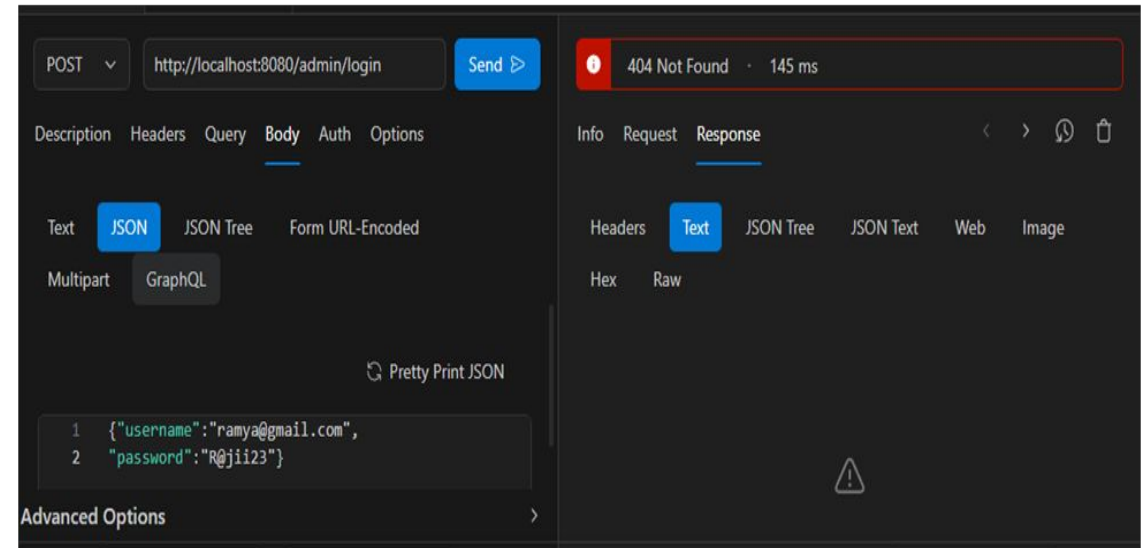
JSON

```
1 { "username": "rajilax@gmail.com",
2   "password": "R@jii23" }
```

JSON Text

```
1 {
2   "adminId": 3,
3   "name": "Rajalakshmi",
4   "username": "rajilax@gmail.com",
  ... }
```

Invalid Login (404 Not Found)



This screenshot shows a failed POST request to `http://localhost:8080/admin/login` in Postman. The status is `404 Not Found` with a response time of `145 ms`. The request body is a JSON object: `{ "username": "ramya@gmail.com", "password": "R@jii23" }`. The response body is empty, and a warning icon is visible at the bottom right.

```
POST http://localhost:8080/admin/login
```

404 Not Found · 145 ms

JSON

```
1 { "username": "ramya@gmail.com",
2   "password": "R@jii23" }
```

Text








Warning icon

Testing: Admin Login Page

TESTING

Filter (e.g. text, !exclude, @tag)


4/4 tests passed (100%)

- ✓  rll 5.9s
- ✓  {} com.company.rll 5.9s
 - ✓  AdminLoginTests 5.9s
 - ✓  contextLoads() 6.0ms
 - ✓  testGreet() 155ms
 - ✓  testadminLogin() 5.5s
 - ✓  testadminLogin2() 186ms

```
@Test
void shouldLoginSampleAdmin1() throws Exception {
    MvcResult result =
        this.mockmvc.perform(
            MockMvcRequestBuilders
                .put(urlTemplate: "/admin/login")
                .content(content: "{\"username\": \"admin@example.com\", \"password\": \"admin\"}")
                .contentType(MediaType.APPLICATION_JSON)
                .accept(MediaType.APPLICATION_JSON)
        )
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andReturn();
    AdminEntity admin =
        this.objectMapper.readValue(
            result.getResponse().getContentAsString(),
            valueType: AdminEntity.class
        );
    assertEquals(expected: "Admin", admin.getName());
}
```

Location Module

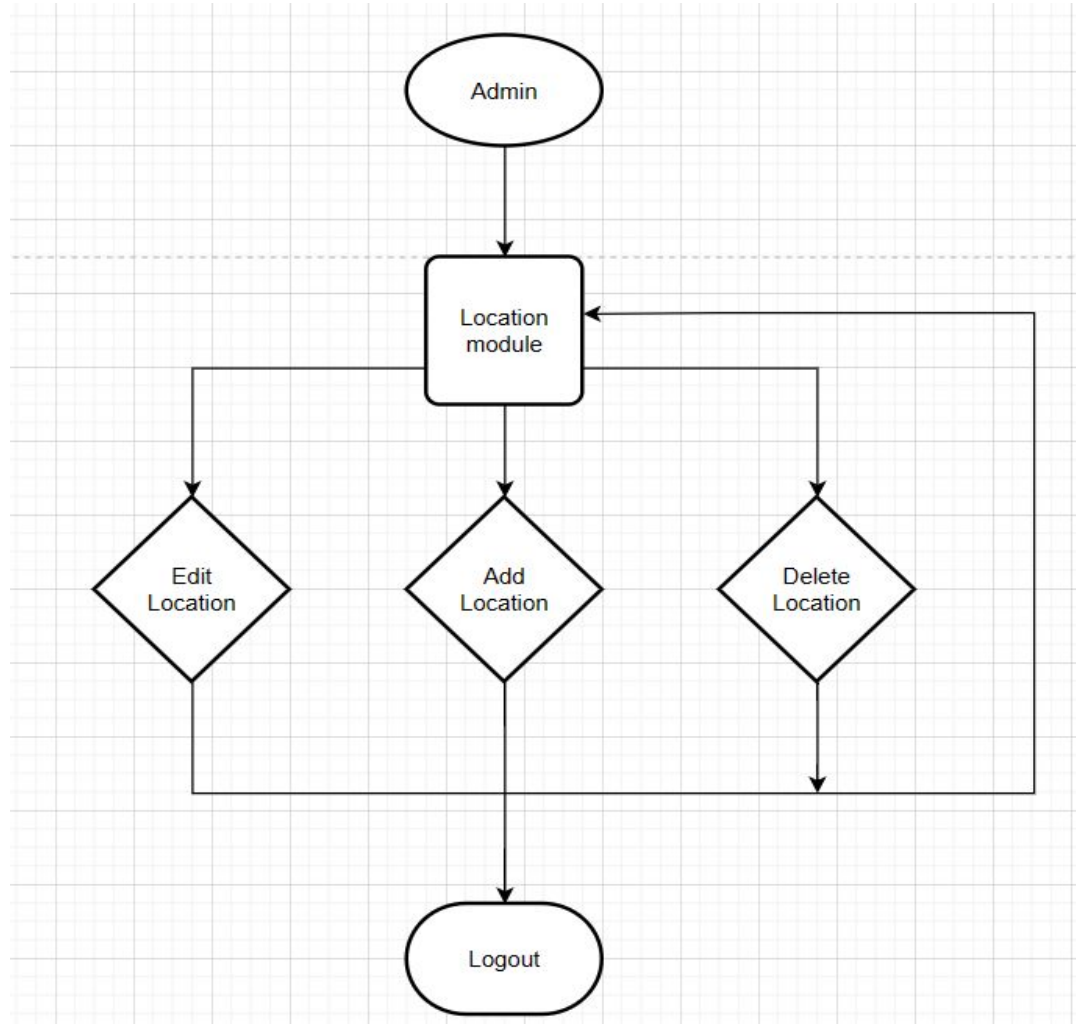
Online Bus Reservation

Home Maintenance 

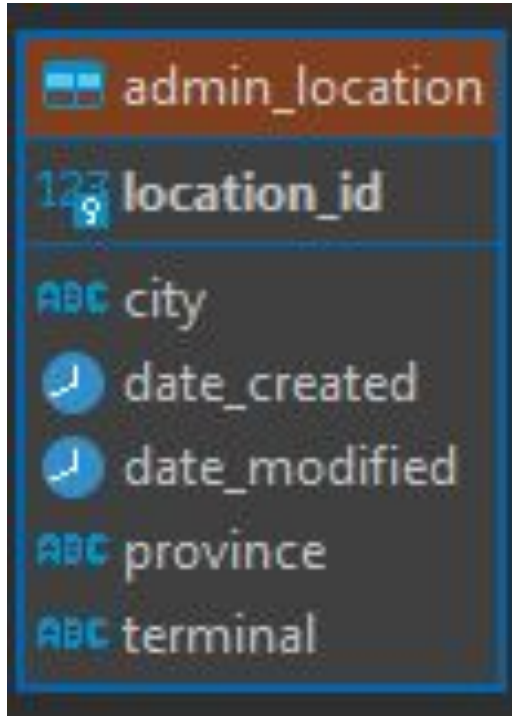
Add new +

Location Id	Terminal	City	Province	Action	
1		Delhi	Delhi	Edit	Delete
2		Mumbai	Maharashtra	Edit	Delete
3		Kolkata	West Bengal	Edit	Delete
4		Bandalore	Karnataka	Edit	Delete
5		Chennai	Tamil Nadu	Edit	Delete
6		Hyderabad	Telangana	Edit	Delete
7		Pune	Maharashtra	Edit	Delete
8		Ahmedabad	Gujrat	Edit	Delete
9		Surat	Gujrat	Edit	Delete
10		Prayagraj	Uttar Pradesh	Edit	Delete

Location Module - Flow



Location Module - Database Schema



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
location_id	1	bigint	[v]	[v]	PRI		auto_increment
city	2	varchar(255)	[v]	[]			
date_created	3	datetime(6)	[v]	[]			
date_modifi...	4	datetime(6)	[v]	[]			
province	5	varchar(255)	[v]	[]			
terminal	6	varchar(255)	[v]	[]			

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
location_id	1	bigint	[v]	[v]	PRI		auto_increment
city	2	varchar(255)	[v]	[]			
date_created	3	datetime(6)	[v]	[]			
date_modifi...	4	datetime(6)	[v]	[]			
province	5	varchar(255)	[v]	[]			
terminal	6	varchar(255)	[v]	[]			

Location Module - Backend end points

location-controller

PUT /admin/location/update_one

POST /admin/location/add_one

GET /admin/location/total_count

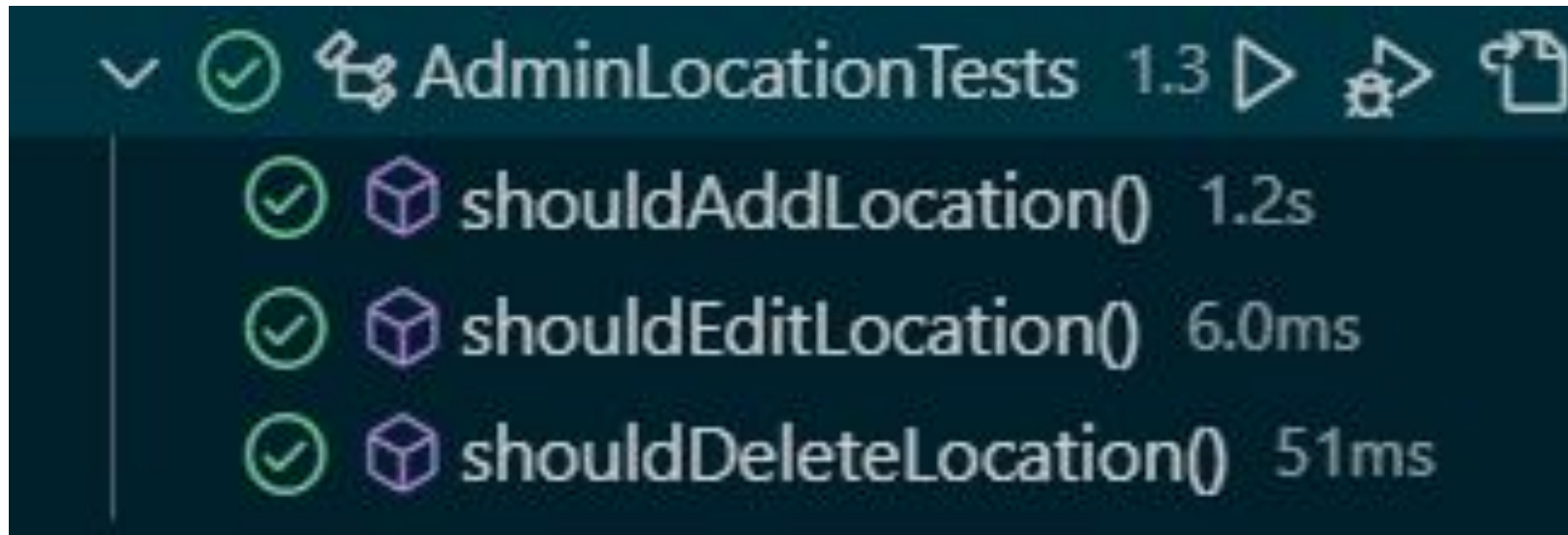
GET /admin/location/read_one/{locationId}

GET /admin/location/read_multiple/{pageNumber}/{count}

DELETE /admin/location/remove_one/{locationId}

Location module - Backend testing

Below is screen of passing test cases



- ❑ We are working on a module that need to manage information about buses, their locations, schedules, prices, and availability. This information is essential for providing efficient transportation services to users.
- ❑ CRUD(create,read ,update delete) operations are done.

```
+-----+
| BusEntity |
+-----+
| bus_id (PK) |
| location    |
| departure   |
| ETA         |
| price       |
| availability |
+-----+
```

Backend Logic of User Bus Module

BusController class

```
@PostMapping(value = "/addbus", consumes =
MediaType.APPLICATION_JSON_VALUE)
public String addbus(@RequestBody BusEntity b) {
    return bs.addbus(b);
}

@GetMapping(value = "/viewAllbuses", produces =
MediaType.APPLICATION_JSON_VALUE)
public List<BusEntity> showallbuses() {
    return bs.showallbuses();
}
```

BusService class

```
// adding
public String addbus(BusEntity b) {
    if (b != null) {
        br.save(b);
        return "Bus Added";
    } else {
        return "Bus was not added";
    }
}

//deleting
public String deletebus(long bus_id) {
    br.deleteById(bus_id);
    return "Bus Removed";
}
```

Backend-Rapid API

The screenshot displays the Backend-Rapid API interface. On the left, the 'Request' tab is active, showing a GET request to `http://localhost:8080/bus/viewAllbuses`. The 'Body' tab is selected, displaying a JSON object:

```
{  "location": "Banglore",  "departure": "8am",  "eta": "8am",  "price": 240,}
```

. A 'Pretty Print JSON' button is visible. On the right, the 'Response' tab is active, showing a 200 OK status with a response time of 119 ms. The 'JSON Text' tab is selected, displaying the response body:

```
[  {    "bus_id": 1,    "location": "Banglore",    "departure": "8am",    "eta": "8am",    "price": 240.  }]
```

. A search bar labeled 'Filter expression...' is at the bottom right.

GET `http://localhost:8080/bus/viewAllbuses` Send

Description Headers Query **Body** Auth Options

Text **JSON** JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {
2   "location": "Banglore",
3   "departure": "8am",
4   "eta": "8am",
5   "price": 240,
6 }
```

200 OK · 119 ms

Info Request **Response**

Headers Text JSON Tree **JSON Text** Raw

```
1 [
2   {
3     "bus_id": 1,
4     "location": "Banglore",
5     "departure": "8am",
6     "eta": "8am",
7     "price": 240.
8   }
9 ]
```

Filter expression...

JsonPath expressions e.g. `$store.book[0].title`

Front-End logic

```
ngOnInit() {

this.busService.viewAllBus().subscribe((response)=>{this.buses=response;}, (error)=>{console.log removeBus(busNo: number) {g(error);}}));

}

this.busService.deletebus(busNo).subscribe((response)=>{
    if (response === 'Bus Removed') {
        this.router.navigateByUrl('/', { skipLocationChange: true }).then(() =>
{this.router.navigate([this.currenturl]);});
    } else {
        alert(`Could not delete`);
    }
}, (error) => {
    console.error('API Error:', error);
});
```



localhost:4200/user/bus



List of Buses

Bus Number

Location

Departure

ETA

Price

Availability

Go to Home

Test Cases

PROBLEMS **6** OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS GITLENS

```
%TESTC 3 v2
%TSTTREE2,com.company.r11.BusTests,true,3,false,1,BusTests,,[engine:junit-jupiter]/[class:com.
company.r11.BusTests]
%TSTTREE3,testFindAllBuses(com.company.r11.BusTests),false,1,false,2,testFindAllBuses(),,[engi
ne:junit-jupiter]/[class:com.company.r11.BusTests]/[method:testFindAllBuses()]
%TSTTREE4,testAddBus(com.company.r11.BusTests),false,1,false,2,testAddBus(),,[engine:junit-jup
iter]/[class:com.company.r11.BusTests]/[method:testAddBus()]
%TSTTREE5,testBusEntityConstructor(com.company.r11.BusTests),false,1,false,2,testBusEntityCons
tructor(),,[engine:junit-jupiter]/[class:com.company.r11.BusTests]/[method:testBusEntityConstr
uctor()]
%TESTS 3,testFindAllBuses(com.company.r11.BusTests)

%TESTE 3,testFindAllBuses(com.company.r11.BusTests)

%TESTS 4,testAddBus(com.company.r11.BusTests)

%TESTE 4,testAddBus(com.company.r11.BusTests)

%TESTS 5,testBusEntityConstructor(com.company.r11.BusTests)

%TESTE 5,testBusEntityConstructor(com.company.r11.BusTests)

%RUNTIME50653
```

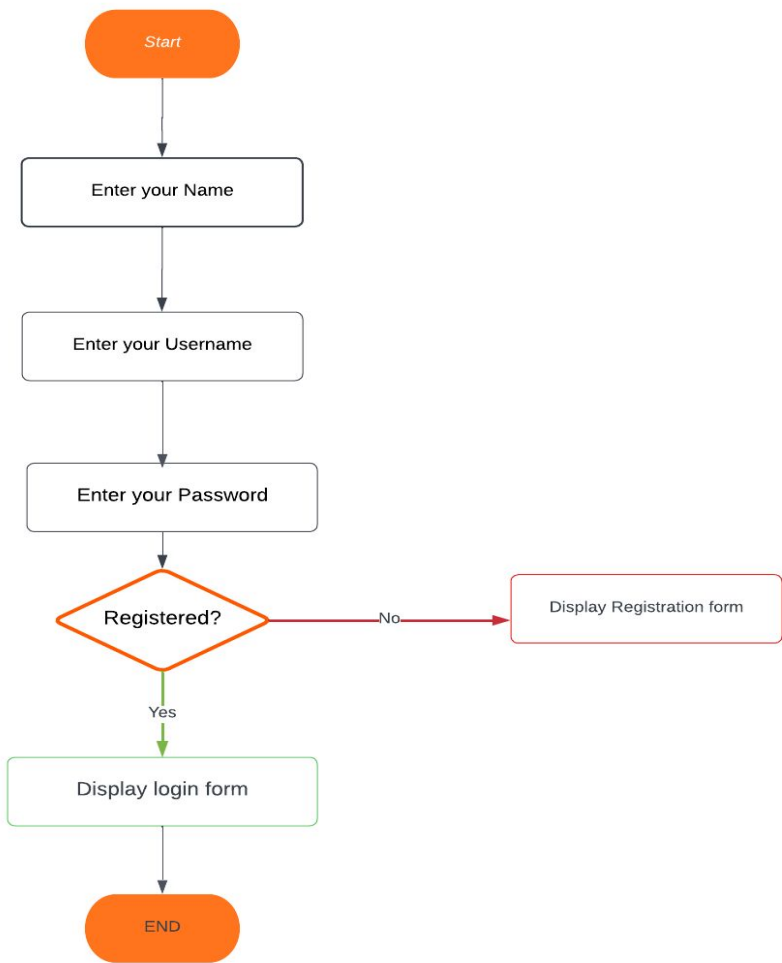
✓ Test run at 8/15/2023, 7:50:43 PM

- ✓ testAddBus()
- ✓ testBusEntityConstructor()
- ✓ testFindAllBuses()

User Registration Module

Priya Singha

Flow Diagram



User Entity

User	
<u>user_id</u>	
name	
username	
password	
date_created	
date_modified	

Frontend

Register User

Name

Enter your Name

Username

Enter your username

Password

Enter your Password

Register

Register User

Name

Priya Singha

Username

priya123singha@gmail.com

Password

....

Register

Backend

- **User Registration**

```
public UserEntity register(UserEntity u) {  
    Optional<UserEntity> r = this.userRepository.findById((long) u.getUserId());  
    if (!r.isPresent()) {  
        return this.userRepository.save(u);  
    }  
    return null;  
}
```

- **Controller**

```
@PostMapping(  
    value = "/register",  
    consumes = MediaType.APPLICATION_JSON_VALUE,  
    produces = MediaType.APPLICATION_JSON_VALUE  
)  
public ResponseEntity<UserEntity> register(@RequestBody UserEntity u) {  
    return new ResponseEntity<UserEntity>(  
        this.userService.register(u),  
        HttpStatus.CREATED  
    );  
}
```

Rapid API

The screenshot displays the RapidAPI client interface with a dark theme. At the top, there are tabs for Java files: `UserEntity.java`, `UserRepository.java`, `UserController.java`, and `UserService.java`. The active tab is `Request 2`.

The main interface is divided into two panels. The left panel shows the request configuration:




- Method: `POST`
- URL: `http://localhost:8080/user/register`
- Buttons: `Send` and `Send >`
- Tabs: `Description`, `Headers`, `Query`, `Body` (selected), `Auth`, `Options`
- Body format: `JSON` (selected), `JSON Tree`, `Form URL-Encoded`, `Multipart`, `GraphQL`
- Buttons: `Pretty Print JSON`
- Request body (line 1): `{"username": "Priya Singha", "password": "Priya23"}`
- Bottom section: `Advanced Options` with a right arrow, `Request snippet`, `JavaScript` (selected), `XMLHttpRequest` (selected), and icons for code and copy.

The right panel shows the response details:

- Status: `200 OK`, Time: `101 ms`
- Tabs: `Info`, `Request` (selected), `Response`
- Response format: `JSON Text` (selected), `Headers`, `Text`, `JSON Tree`, `Raw`
- Response body (lines 1-4):

```
1 {  
2   "username": "Priya Singha",  
3   "password": "Priya23"  
4 }
```
- Filter bar: `Filter expression...` with a close button and a note: `JsonPath expressions e.g. $.store.book[0].title`
- Bottom section: `JSON to` and `TypeScript` (selected) with icons for code and copy.

Testing

✓  UserRegisterTests 3.3s
✓  testApiWithJsonInput() 3.2s
✓  shouldDeleteSampleAdmin() 184ms

PROBLEMS **6** OUTPUT TEST RESULTS TERMINAL GITLENS DEBUG CONSOLE

%TESTS 3, testApiWithJsonInput (com.company.r11.user.UserRegisterTests)

%TESTE 3, testApiWithJsonInput (com.company.r11.user.UserRegisterTests)

%TESTS 4, shouldDeleteSampleAdmin (com.company.r11.user.UserRegisterTests)

%TESTE 4, shouldDeleteSampleAdmin (com.company.r11.user.UserRegisterTests)

%RUNTIME26529

Proof of integration and Screenshot of your commits

```
elp UserController.java - rll - Visual Studio Code

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL GITLENS

commit 7ed857e0ecd1ca724e0d012468eade5354ccf52d
Author: Priya S <priya123singha@gmail.com>
Date: Thu Aug 10 14:33:20 2023 +0530

    User Register testing Done

commit 4fcde33b25f1c9d6467135ebfa65907b65bf4272
Author: Priya S <priya123singha@gmail.com>
Date: Wed Aug 9 17:24:39 2023 +0530

    FRONTEND MENOR ISSUE RESOLVED

commit f0023b027f32cde4b88409a72080a56058a9ff14
Author: Priya S <priya123singha@gmail.com>
Date: Wed Aug 9 17:02:05 2023 +0530

    edit frontEnd

commit 10f36e88f52d1845d786b3ab0dce2ce2ad19b365
Author: Priya S <priya123singha@gmail.com>
Date: Wed Aug 9 15:39:31 2023 +0530

    Done User Registration frontEnd and backEnd

commit 75c762eaf9114a35e392652a4b14294daf38aebf
Author: Priya S <priya123singha@gmail.com>
Date: Wed Aug 9 11:05:44 2023 +0530

    Edit frontend

commit cc8a73dcbefcb34daa07cd6c9a20103f6e2929d9
Author: Priya S <priya123singha@gmail.com>
Date: Tue Aug 8 20:03:13 2023 +0530

    User Register

commit 0f5a7b52c8f18edb16ef4e8168c39a50bb81e60d
Author: Priya S <priya123singha@gmail.com>
Date: Tue Aug 8 19:22:23 2023 +0530

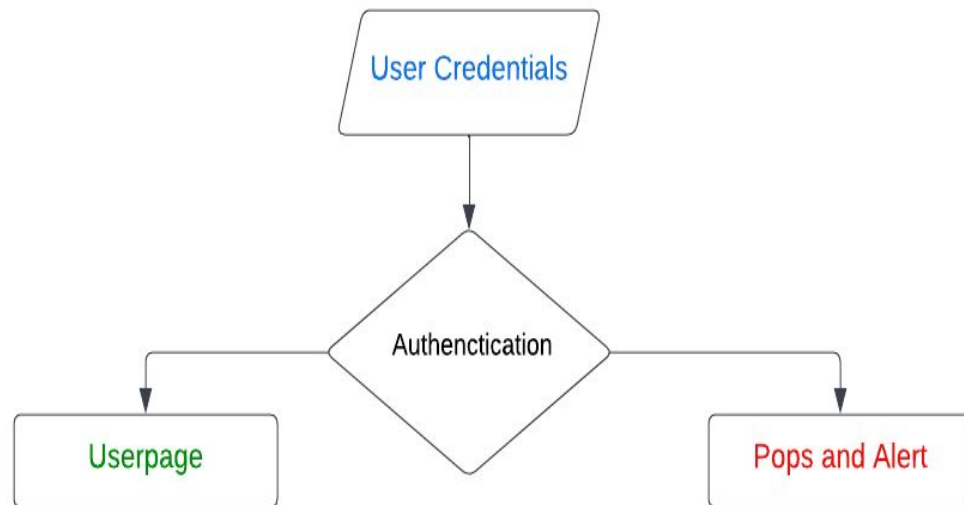
    user registration

commit 4d356c73230fee5bc6a28b2b318979a2b57b35dd
Author: Priya S <priya123singha@gmail.com>
```

User- Login

-Swaroop A V

An Overview



User Entity

<i>User</i>
<u><i>user_id</i></u>
<i>name</i>
<i>username</i>
<i>password</i>
<i>date_created</i>
<i>date_modified</i>

Backend Logic of user-login

- User Authentication

```
public UserEntity loginvalidation(String username, String password) {  
    Optional<UserEntity> u = this.userRepository.findByUsername(username);  
    if (u.isPresent()) {  
        if (u.get().getUsername().equals(username) && u.get().getPassword().equals(password)) {  
            return u.get();  
        }  
    }  
    return null;  
}
```

- Controller

```
@PostMapping("/login")  
public ResponseEntity<Object> loginvalidation(@RequestBody UserEntity u) {  
    UserEntity user = this.userService.loginvalidation(u.getUsername(), u.getPassword());  
    if (user != null) {return new ResponseEntity<Object>(user, HttpStatus.OK);}  
    return new ResponseEntity<Object>(null, HttpStatus.NOT_FOUND);  
}
```


API endpoints

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/user/login`. The request body is a JSON object: `{"username": "user@example.com", "password": "user"}`. The response is a 200 OK status with a response time of 1056 ms. The response body is a JSON object: `{"userId": 1, "name": "User", "username": "user@example.com", "password": "user", "dateModified": 1692008010000, "dateCreated": 1692008010000}`.

PUT `http://localhost:8080/user/login` Send

200 OK · 1056 ms

Description Headers Query Body Auth Options

Text **JSON** JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {"username": "user@example.com", "password": "user"}
```

Info Request Response

Headers Text JSON Tree **JSON Text**

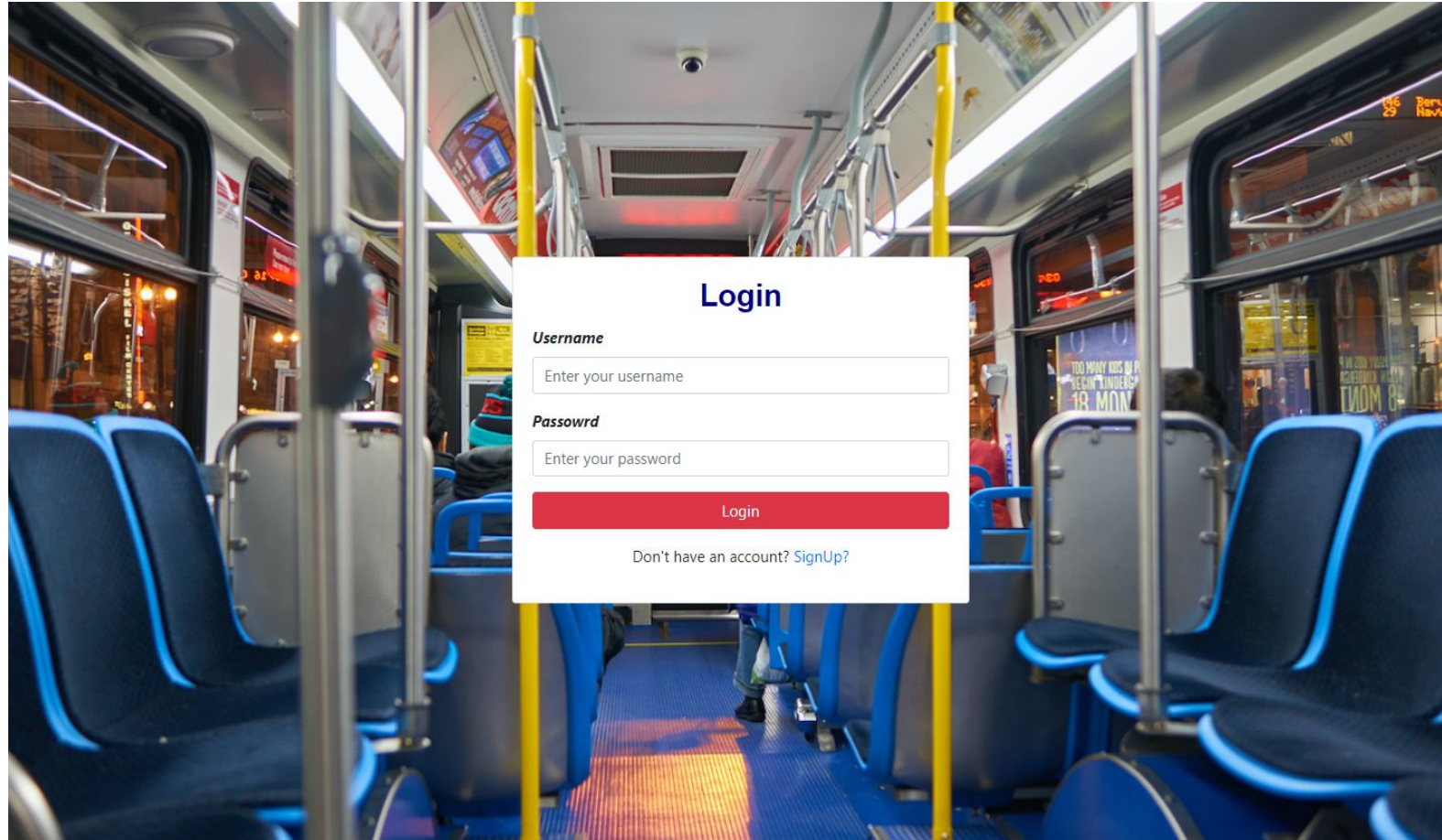
```
1 {
2   "userId": 1,
3   "name": "User",
4   "username": "user@example.com",
5   "password": "user",
6   "dateModified": 1692008010000,
7   "dateCreated": 1692008010000
8 }
```

Connecting to the front -end

```
onSubmit(user: User) {console.log(user);  
this.loginService.login(user).subscribe((user) => {  
if (user !== null) {this.router.navigate(['/user']);}  
else {alert(`OOPS You have missed  
somewhere with your credentials`);}}});
```

- Retrieving the authentication result

```
login(user: User): Observable<User> {return  
this.http.put<User>(`${this.loginurl}/user/login  
`, user);}
```



Test Cases

- Test Cases with json inputs

@Test

```
void shouldLoginSampleUser1() throws Exception {  
    MvcResult result =this.mvc.perform(  
        MockMvcRequestBuilders  
            .put("/user/login")  
            .content("{\"username\":\"user@example.com\",\"password\":\"user\"}")  
            .contentType(MediaType.APPLICATION_JSON)  
            .accept(MediaType.APPLICATION_JSON))  
        .andExpect(MockMvcResultMatchers.status().isOk()).andReturn();  
    UserEntity user =this.objectMapper.readValue(  
        result.getResponse().getContentAsString(),UserEntity.class);  
    assertEquals("User", user.getName());}
```

Git commits

```
Swaroop@DESKTOP-J6JSLD2 MINGW64 ~/Desktop/EclipseWorkspace/r11 (main)
$ git log --author="Swaroop A V" --oneline
4cfc39c Minor Changes in Bus Html file
e9ee8a8 Login Tests Issues resolved
6f014ee FRONTEND bus delete function and viewing all bus done
af8c985 FRONTEND minor fixes
8729688 BACKEND:Logintest cases executed
e613d49 BACKEND:user registration merge conflict resolved
89a1baa FRONTEND:Userlogin Final
0f94ff5 FRONTEND:Userlogin working
30f74e0 FRONTEND user login
```

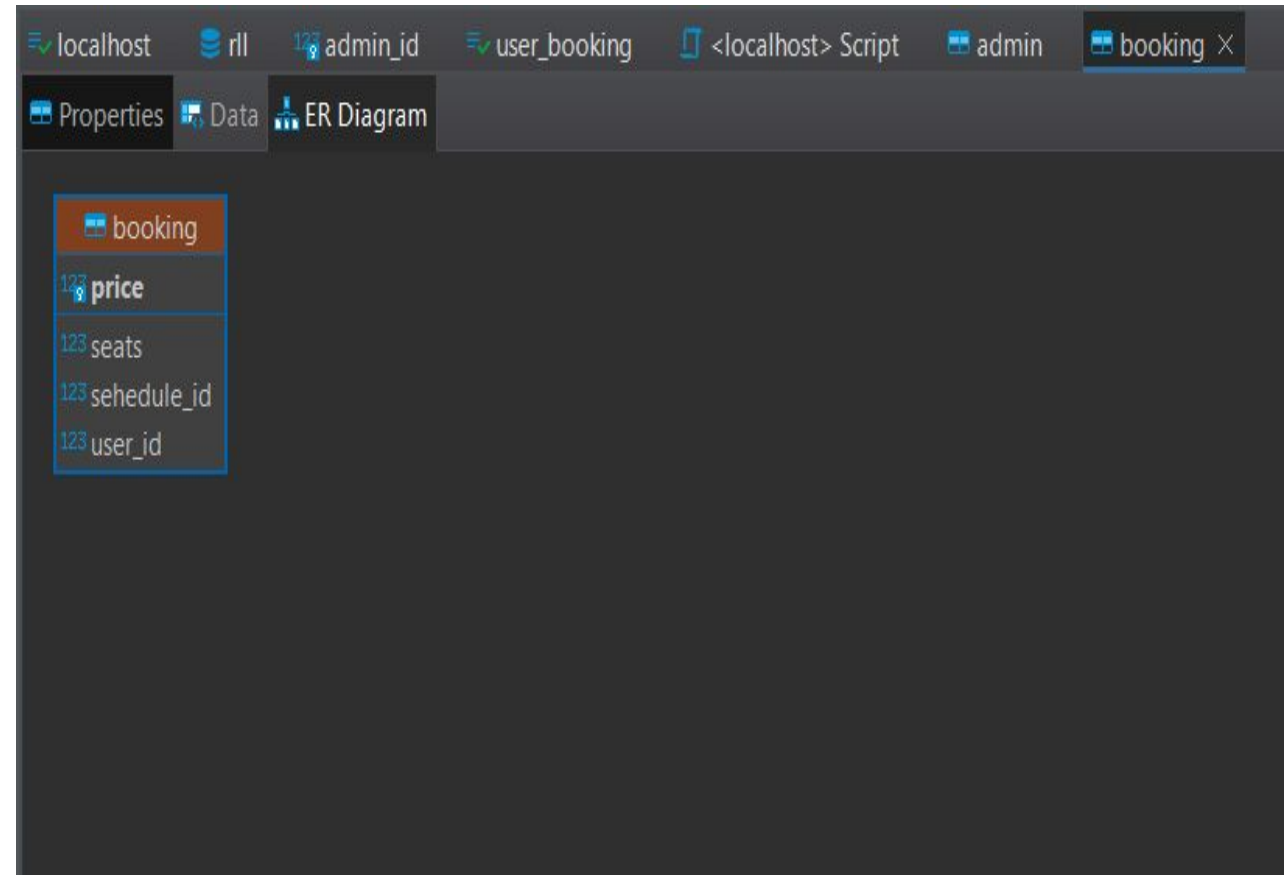
USER BOOKING BUS:

-Kishen Kumar

AIM:

- To Show Bus Booking
- To Edit and Delete the booking bus

Booking Entity



Backend

```
public class BookingEntity {  
    @Column(name = "price")  
    private long price;
```

```
    @Column(name = "seats")  
    private long seats;
```

```
    @Column(name = "user_id")  
    private long user_id;
```

```
    @Column(name = "sehedule_id")  
    private long sehedule_id;
```

```
    @Id  
    private long bookingId;
```

```
}
```

FrontEnd

```
<td>  
    <ng-container *ngIf="editIndex !== i; else editCell">  
        {{ booking.price *booking.numSeats }}  
    </ng-container>  
    <ng-template #editCell>  
        <input [(ngModel)]="editedBooking.price" class="form-control" />  
    </ng-template>  
</td>
```


OUTPUT

Online Bus Booking System.pptx

Online Bus Reservation

localhost:4200/user/booking

Home

Bus Booking

Number of Seats	User ID	Schedule ID	Price	Action
-----------------	---------	-------------	-------	--------

List Of Buses

Online Bus Reservation

[Home](#)


Bus Booking

Number of Seats	User ID	Schedule ID	Price	Action
1	1	101	50	Edit Delete
2	2	102	100	Edit Delete

[List Of Buses](#)

Backend - Java Spring Boot

```
[INFO] Running com.company.rll.BookingTests
2023-08-16T00:16:11.426+05:30 INFO 11840 --- [main] t.c.s.AnnotationConfigContextLoaderUtils : Could not detect default configuration classes for test class [com.company.rll.BookingTests]: BookingTests does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
2023-08-16T00:16:11.450+05:30 INFO 11840 --- [main] .b.t.c.SpringBootTestContextBootstrapper : Found @SpringBootConfiguration com.company.rll.RllApplication for test class com.company.rll.BookingTests
```



Spring Boot v3.1.2

```
2023-08-16T00:16:11.538+05:30 INFO 11840 --- [main] com.company.rll.BookingTests : Starting BookingTests
using Java 17.0.8 with PID 11840 (started by Admin in F:\rll)
2023-08-16T00:16:11.538+05:30 INFO 11840 --- [main] com.company.rll.BookingTests : No active profile set, falling back to 1 default profile: "default"
2023-08-16T00:16:12.091+05:30 INFO 11840 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-08-16T00:16:12.122+05:30 INFO 11840 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 31 ms. Found 5 JPA repository interfaces.
2023-08-16T00:16:12.313+05:30 INFO 11840 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-08-16T00:16:12.328+05:30 INFO 11840 --- [main] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2023-08-16T00:16:12.328+05:30 INFO 11840 --- [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver set up: ignoring JPA class transformer
2023-08-16T00:16:12.328+05:30 INFO 11840 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-2 - Starti
```

🔍 You, 3 hours ago Ln 20, Col 27 Spaces: 2 UTF-8 CRLF {} TypeScript

Test Cases

The screenshot shows an IDE window with the 'TEST RESULTS' tab selected. The left pane displays the test execution output, and the right pane shows a summary of the test runs.

Test Results Output (Left Pane):

```
() , , [engine:junit-jupiter]/[class:com.company.r11.BookingTests]/[method:testUpdateBooking()  
]  
%TSTTREE7, testDeleteBooking (com.company.r11.BookingTests), false, 1, false, 2, testDeleteBooking  
() , , [engine:junit-jupiter]/[class:com.company.r11.BookingTests]/[method:testDeleteBooking()  
]  
%TESTS 3, testGetBookingByUserId (com.company.r11.BookingTests)  
  
%TESTE 3, testGetBookingByUserId (com.company.r11.BookingTests)  
  
%TESTS 4, testShowAllBookings (com.company.r11.BookingTests)  
  
%TESTE 4, testShowAllBookings (com.company.r11.BookingTests)  
  
%TESTS 5, testCreateBooking (com.company.r11.BookingTests)  
  
%TESTE 5, testCreateBooking (com.company.r11.BookingTests)  
  
%TESTS 6, testUpdateBooking (com.company.r11.BookingTests)  
  
%TESTE 6, testUpdateBooking (com.company.r11.BookingTests)  
  
%TESTS 7, testDeleteBooking (com.company.r11.BookingTests)  
  
%TESTE 7, testDeleteBooking (com.company.r11.BookingTests)  
  
%RUNTIME18124
```

Test Results Summary (Right Pane):

- Test run at 8/16/2023, 12:18:53 AM
 - testCreateBooking()
 - testDeleteBooking()
 - testGetBookingByUserId()
 - testShowAllBookings()
 - testUpdateBooking()
- Test run at 8/14/2023, 7:30:14 PM
 - shouldLoginSampleAdmin1()
 - shouldLoginSampleAdmin2()
- Test run at 8/14/2023, 7:25:38 PM
 - testCreateBooking()
 - testDeleteBooking()
 - testGetBookingByUserId()
 - testShowAllBookings()
 - testUpdateBooking()

IDE Status Bar: You, 3 days ago | Ln 9, Col 50 | Spaces: 2 | UTF-8 | CRLF | {} Java

REST API

The screenshot shows an IDE with several Java files open: `BookingController.java`, `BusController.java`, `Request 1`, `BookingEntity.java`, `BookingRepository.java`, `BookingService.java`, and `BookingTe`. The `Request 1` tab is active, displaying a REST client interface. The request method is `GET` and the URL is `http://localhost:8080/bookings/2`. The `Body` tab is selected, showing a JSON body: `{ "user_id": 1000 }`. The `Response` tab is also active, showing a `200 OK` status with a response time of `27 ms`. The `JSON Tree` view is selected, but it displays a warning: "The response body is empty, there is no JSON to be shown".

bookingController.java `BusController.java` Request 1 `BookingEntity.java` `BookingRepository.java` `BookingService.java` `BookingTe`

GET `http://localhost:8080/bookings/2` Send

Description Headers Query Body Auth Options

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

1 `{ "user_id": 1000 }`

Info Request Response

200 OK · 27 ms

Headers Text JSON Tree JSON Text Web Image Hex Raw

The response body is empty, there is no JSON to be shown

THANK YOU