

Project Documentation

SmartStay (Real Estate Application)

Parmar Lakhman, Makwana Vaibhav

March 28, 2024

Contents

1	Introduction	2
1.1	Project Overview	2
1.2	Technologies Used	2
2	System Architecture	3
2.1	Database Design	3
3	API Documentation	4
3.1	api/users	4
3.2	register a user	5
3.3	delete a user	6
3.4	get all users only by admin role	7
3.5	Authorized Dealer registering a property	8
3.6	Authorized Dealer can delete nth Property	9
3.7	Authorized Dealer can view nth property	10
3.8	Authorized Dealer can view all his Properties	11
3.9	View Property Details by Authorized User of any Role	12
3.10	Add Property Details	13
3.11	Delete Property Details	14
3.12	Customer can add property in his Bookmark list	15
3.13	Customer can delete a Property from his Bookmark list	16
3.14	Customer can view Properties in his Bookmark list	17
3.15	Endpoint Lists	18
4	Version Control	19
4.1	GitHub Repositories	19
5	Conclusion	20

Chapter 1

Introduction

1.1 Project Overview

1.2 Technologies Used

Chapter 2

System Architecture

2.1 Database Design

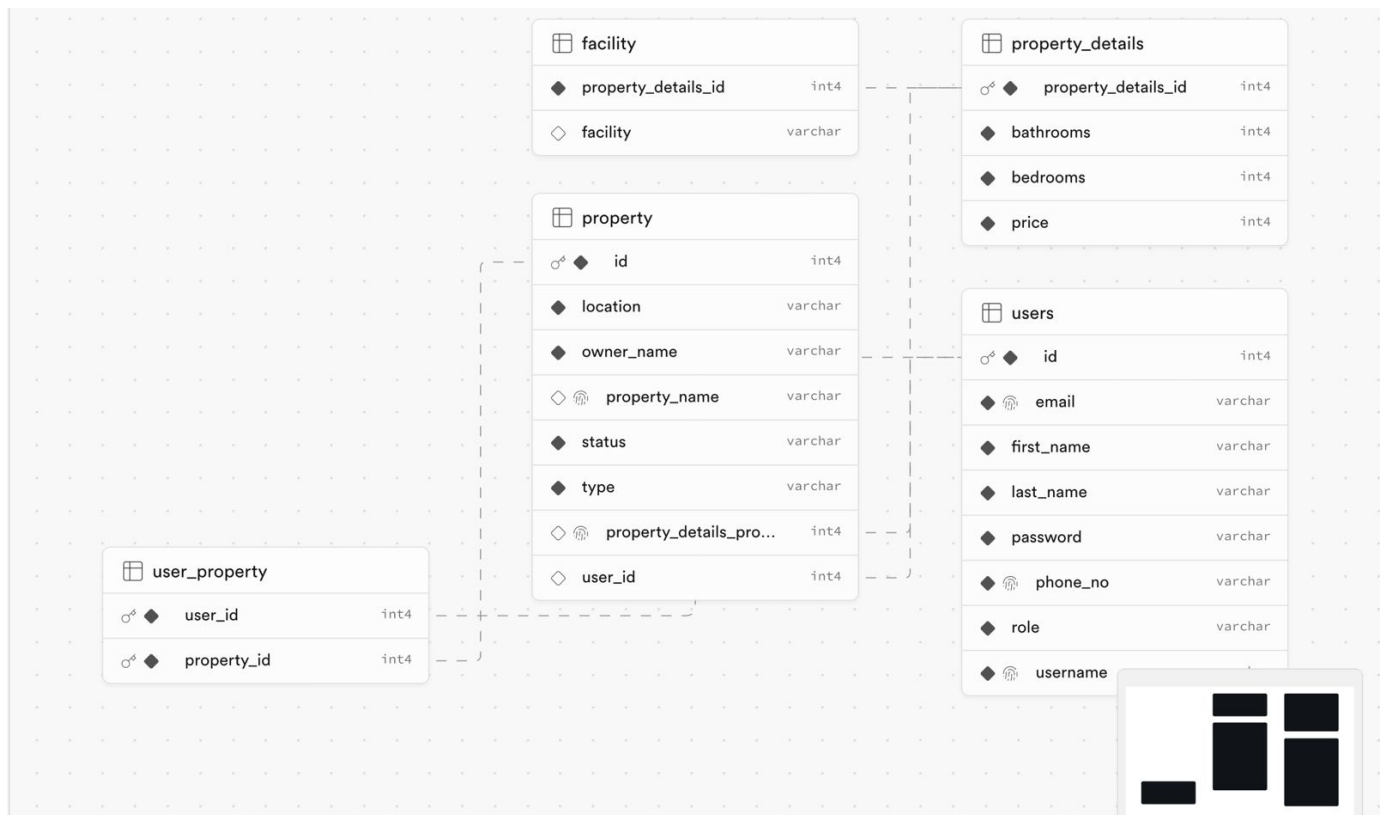


Figure 2.1: Database Design

Chapter 3

API Documentation

3.1 api/users

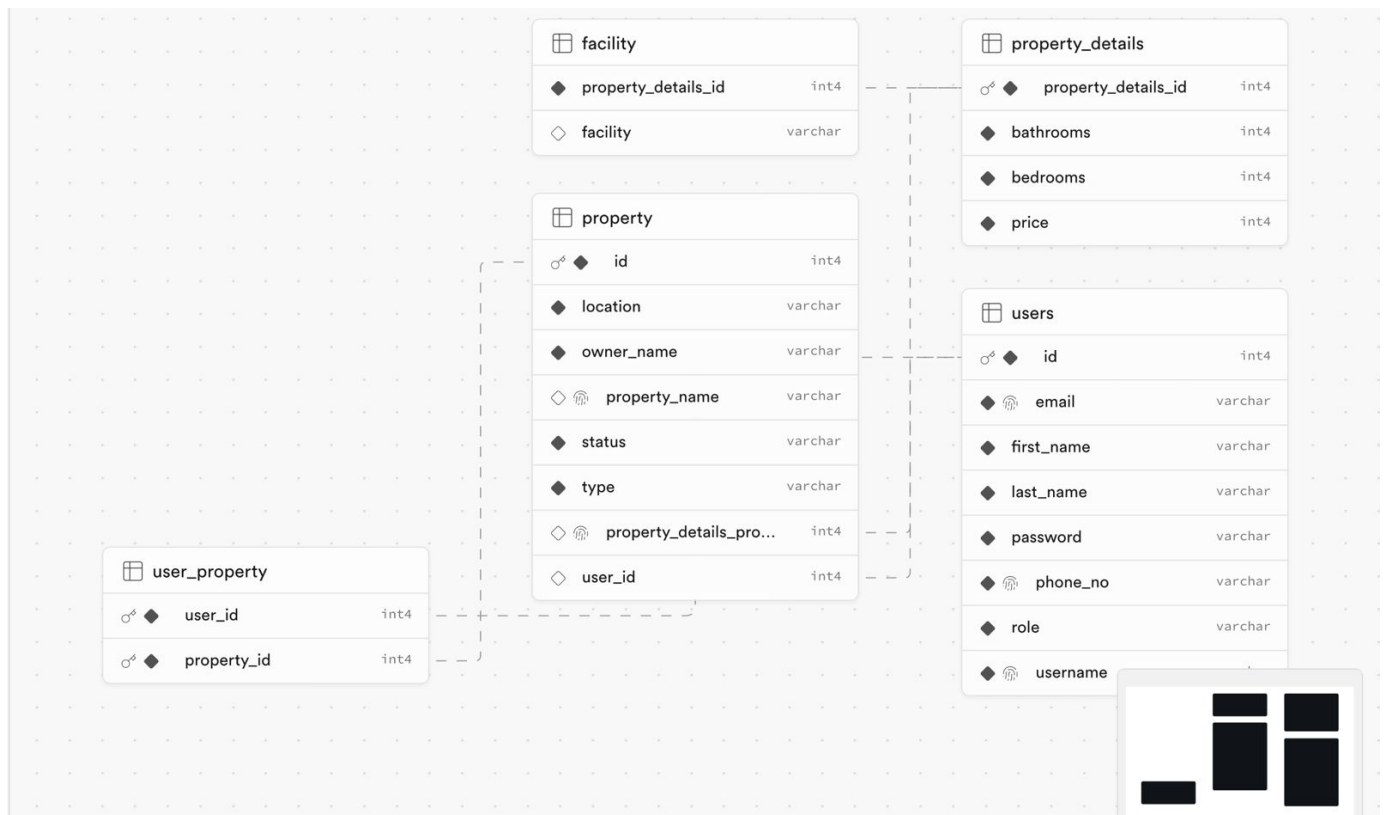


Figure 3.1: api/users

3.2 register a user

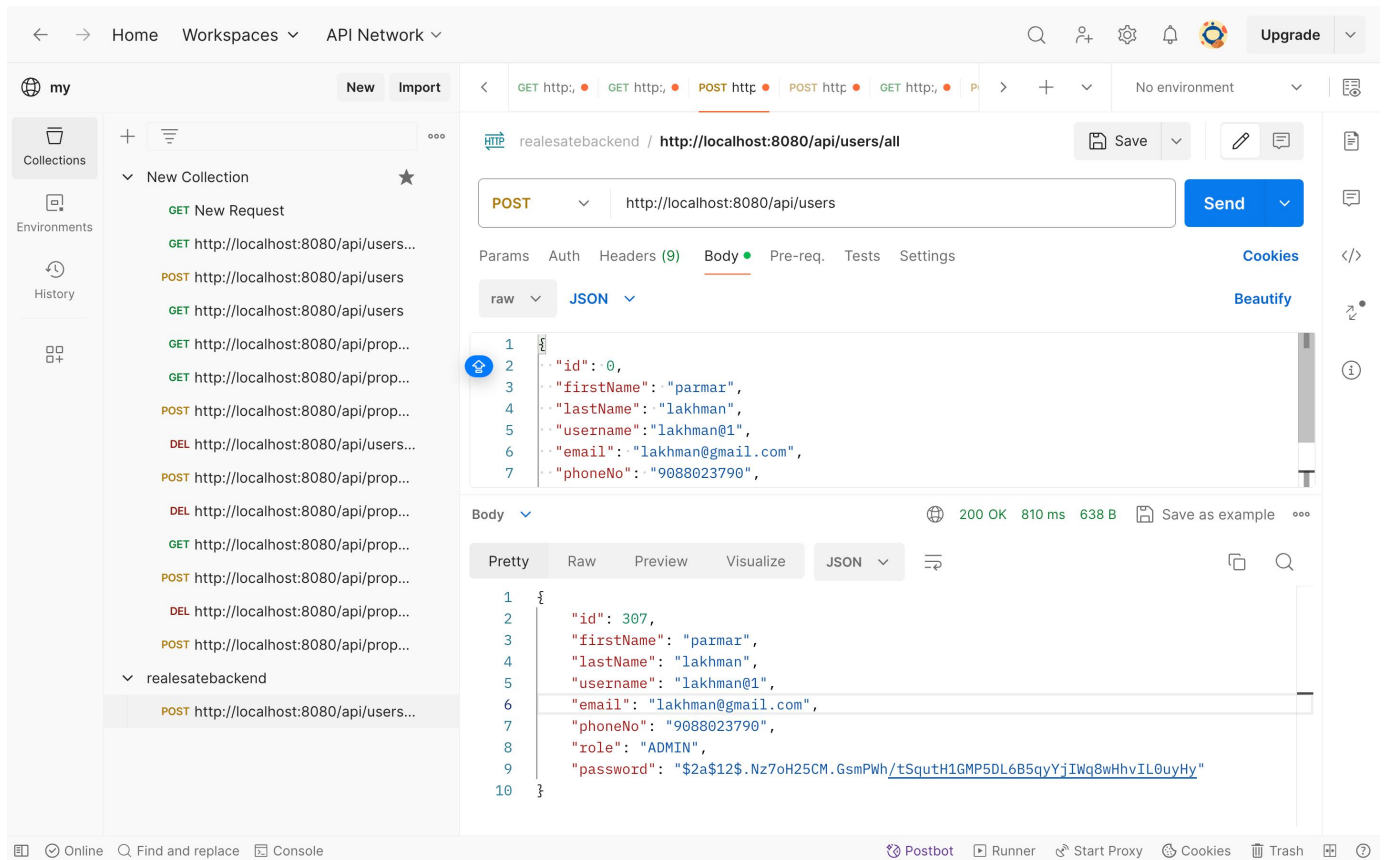


Figure 3.2: api/users/id

3.3 delete a user

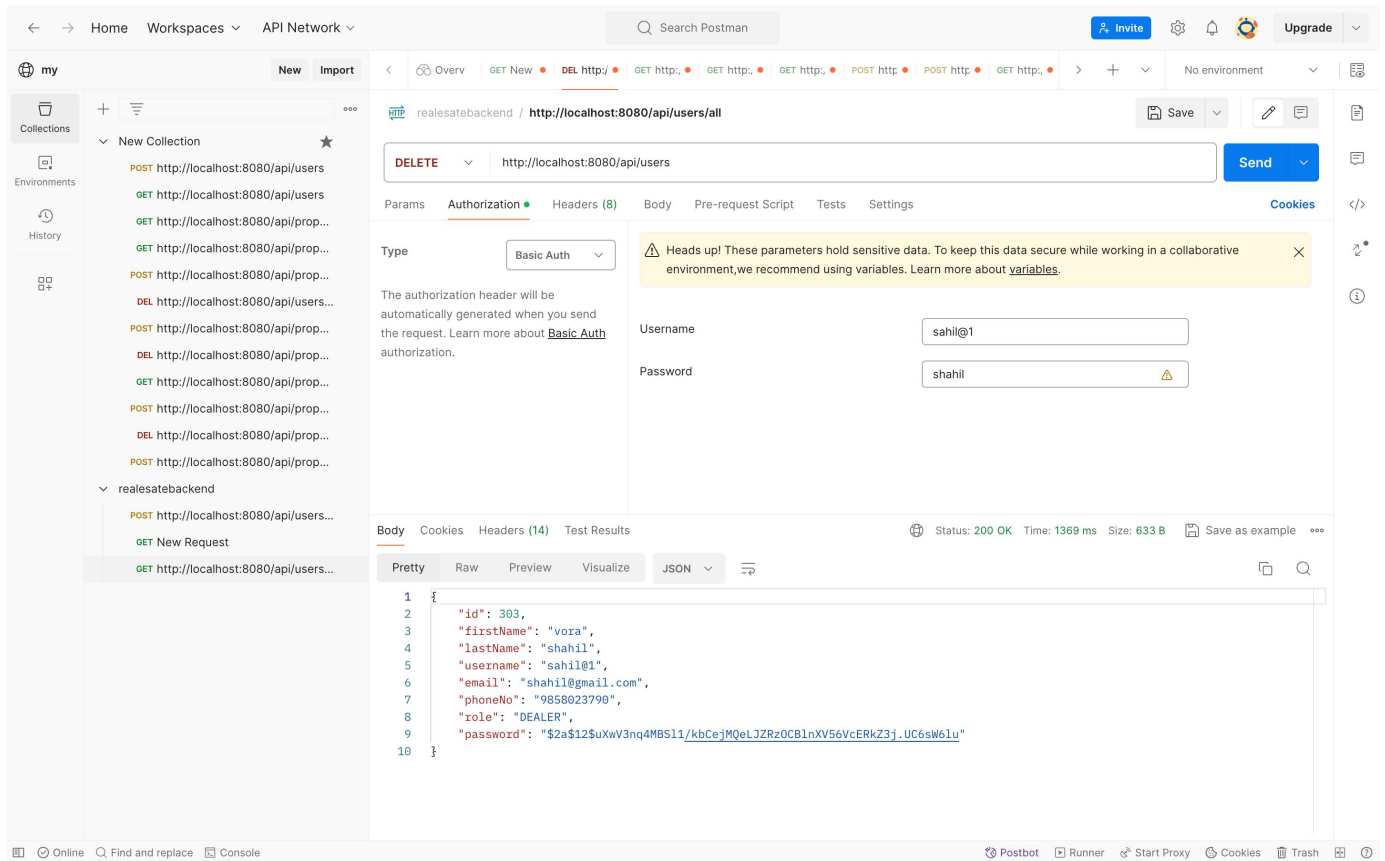


Figure 3.3: api/users

3.4 get all users only by admin role

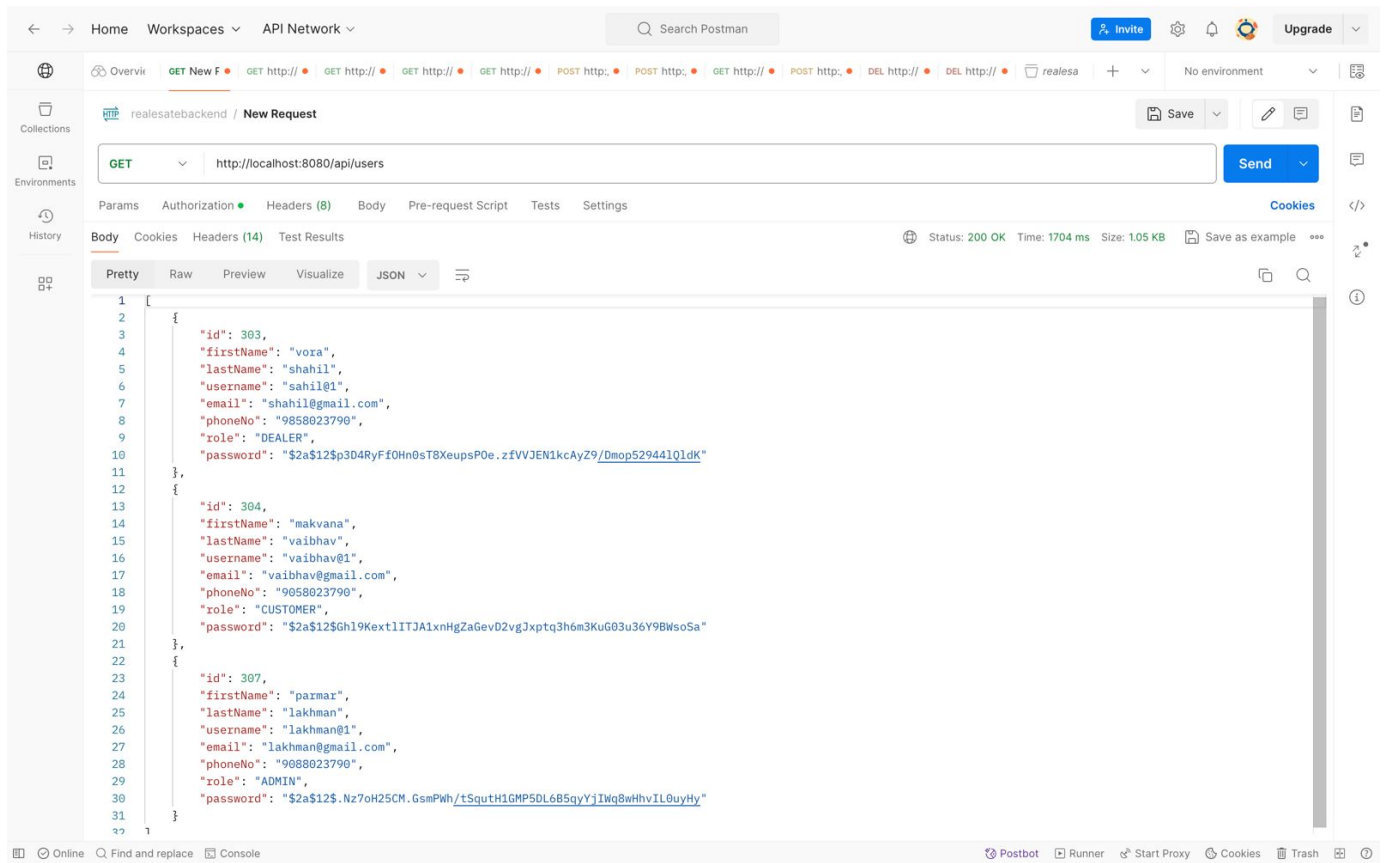


Figure 3.4: api/users

3.5 Authorized Dealer registering a property

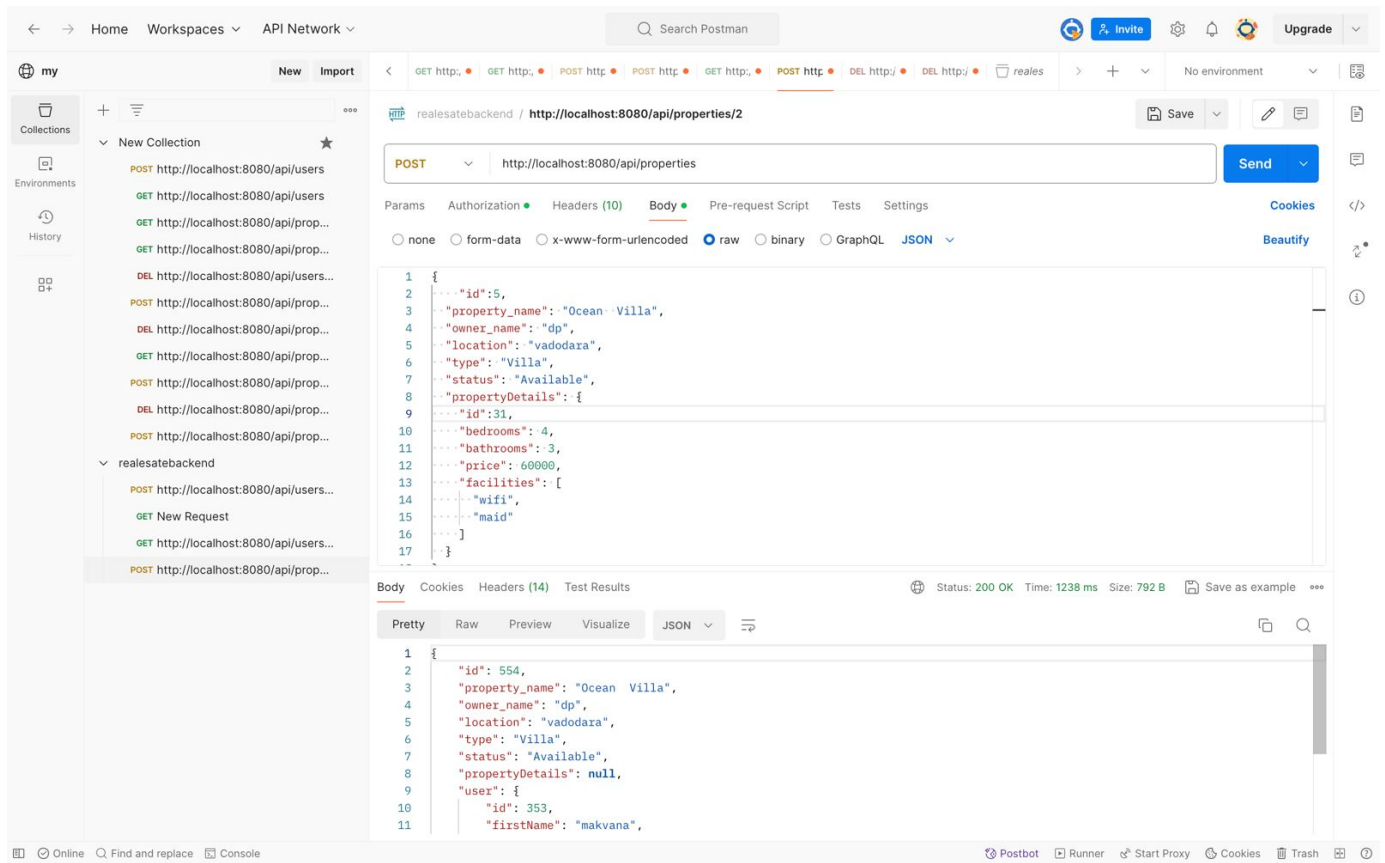


Figure 3.5: api/properties

3.6 Authorized Dealer can delete nth Property

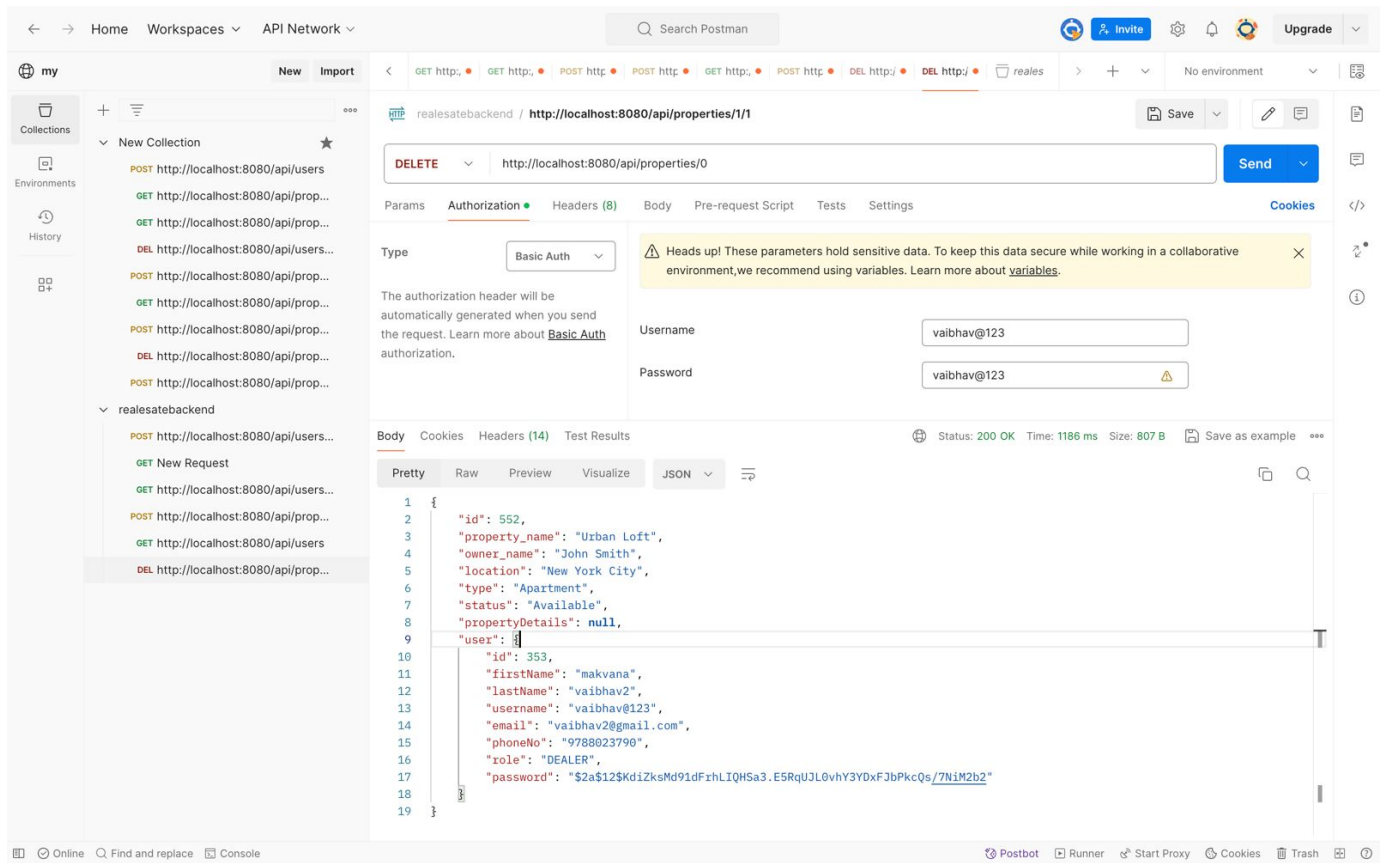


Figure 3.6: api/properties/id

3.7 Authorized Dealer can view nth property

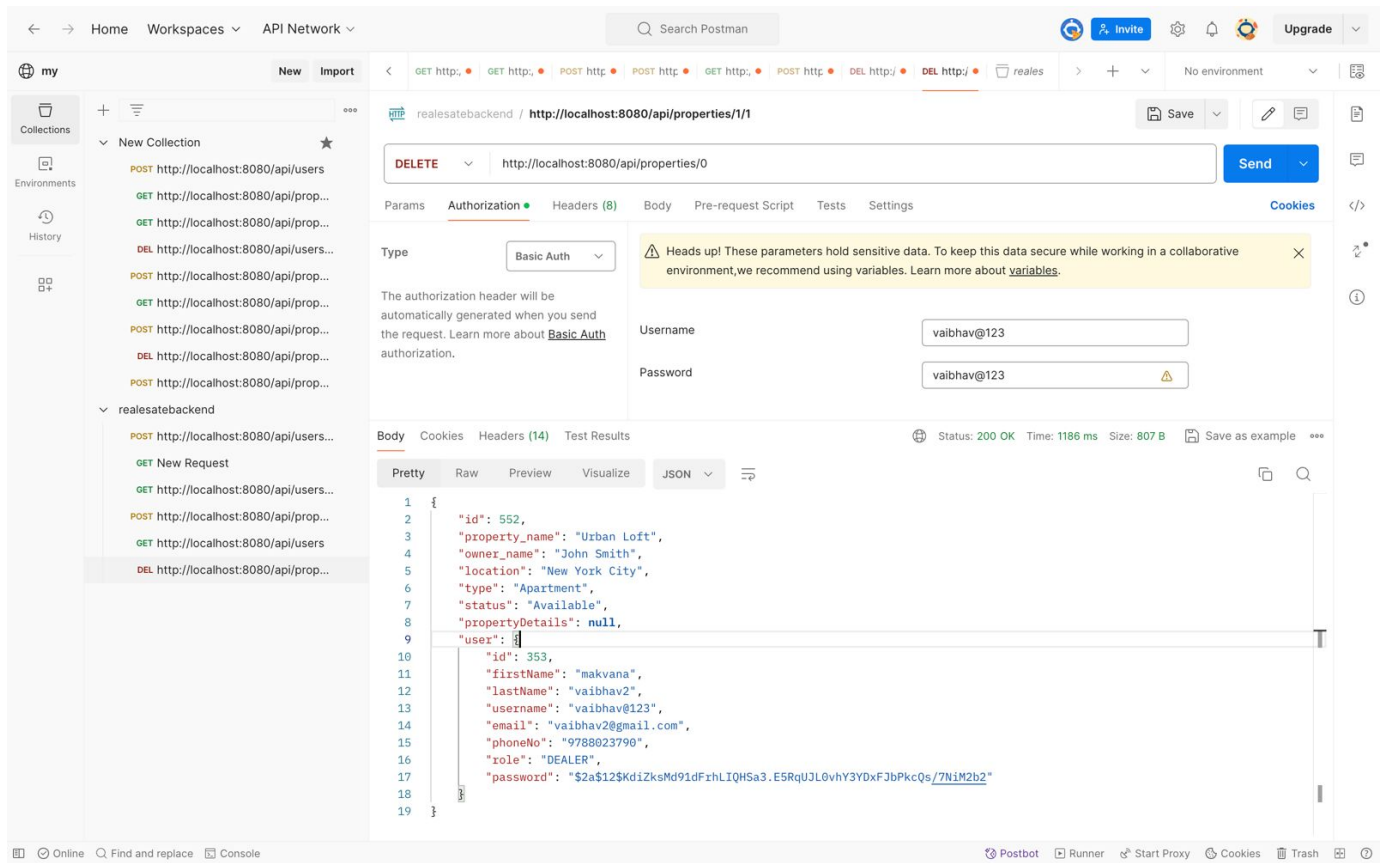


Figure 3.7: api/properties/id

3.8 Authorized Dealer can view all his Properties

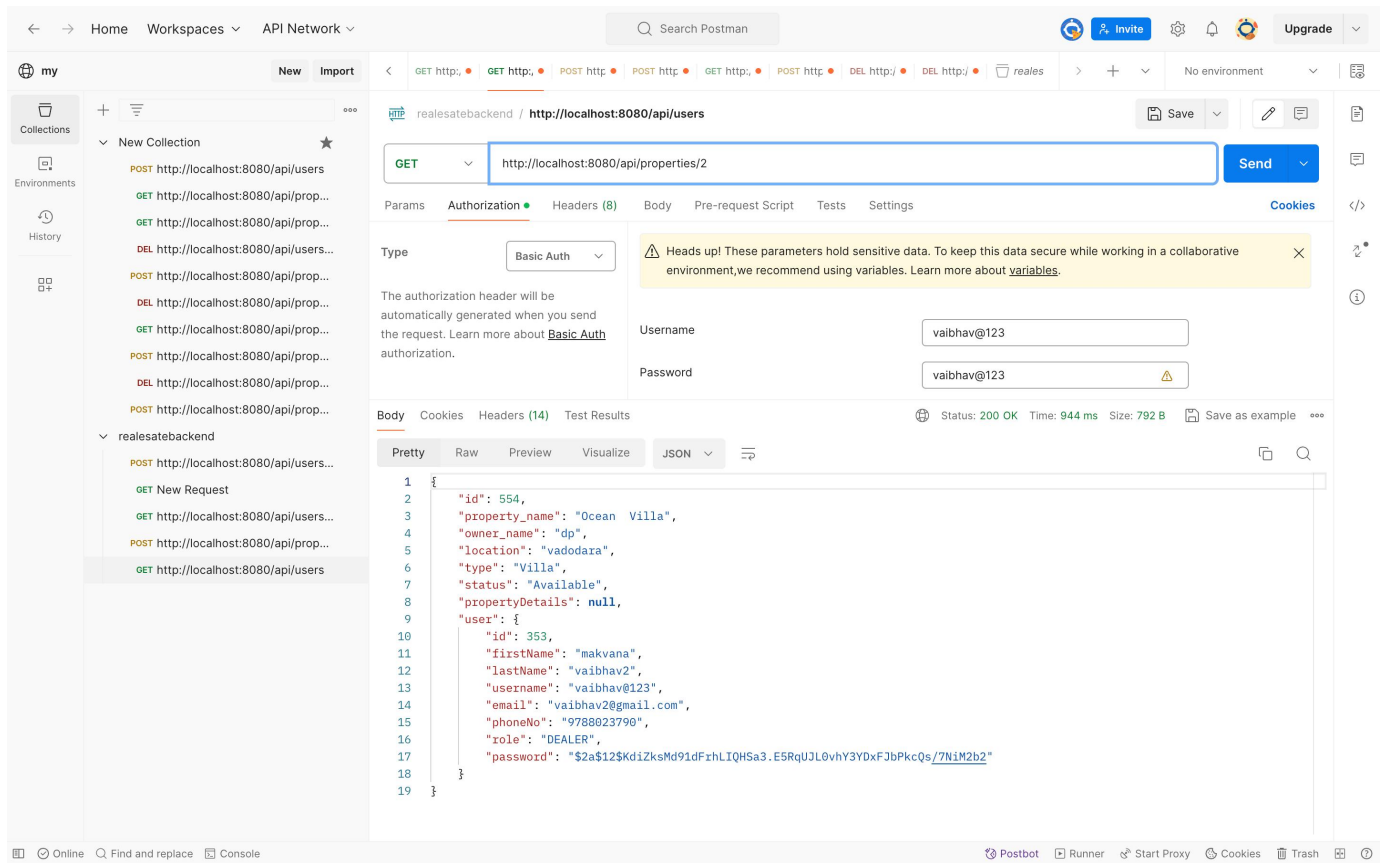


Figure 3.8: api/properties

3.9 View Property Details by Authorized User of any Role

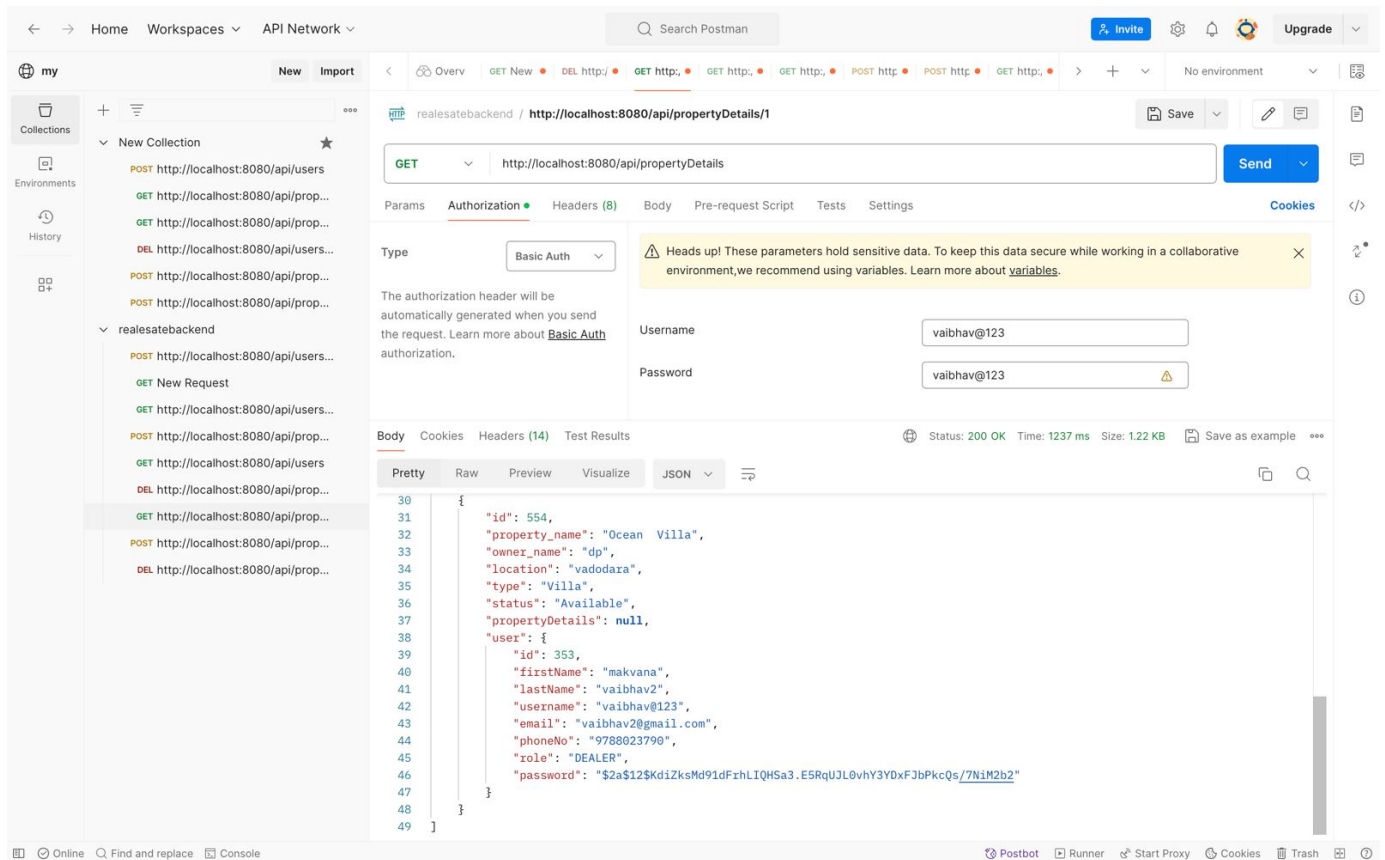


Figure 3.9: api/propertyDetails

3.10 Add Property Details

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/api/propertyDetails/554`. The request body is a JSON object with the following structure:

```
1 {
2   "id": 1,
3   "bedrooms": 4,
4   "bathrooms": 4,
5   "price": 1200,
6   "facilities": ["wifi", "maid"]
7 }
```

The response status is `200 OK` with a time of `1427 ms` and a size of `502 B`. The response body is a JSON object with the following structure:

```
1 {
2   "id": 153,
3   "bedrooms": 4,
4   "bathrooms": 4,
5   "price": 1200,
6   "facilities": [
7     "wifi",
8     "maid"
9   ]
10 }
```

Figure 3.10: api/propertyDetails

3.11 Delete Property Details

★

se...

'op...

'op...

se...

'op...

'op...

se...

se...

'op...

se...

'op...

'op...

'op...

'op...

DELETE

http://localhost:8080/api/propertyDetails/554

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Type

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about [Basic Auth](#) authorization.

Username


vaibhav@123

Password

vaibhav@123

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Response



Click Send to get a response

Figure 3.11: api/propertyDetails/id

3.12 Customer can add property in his Bookmark list

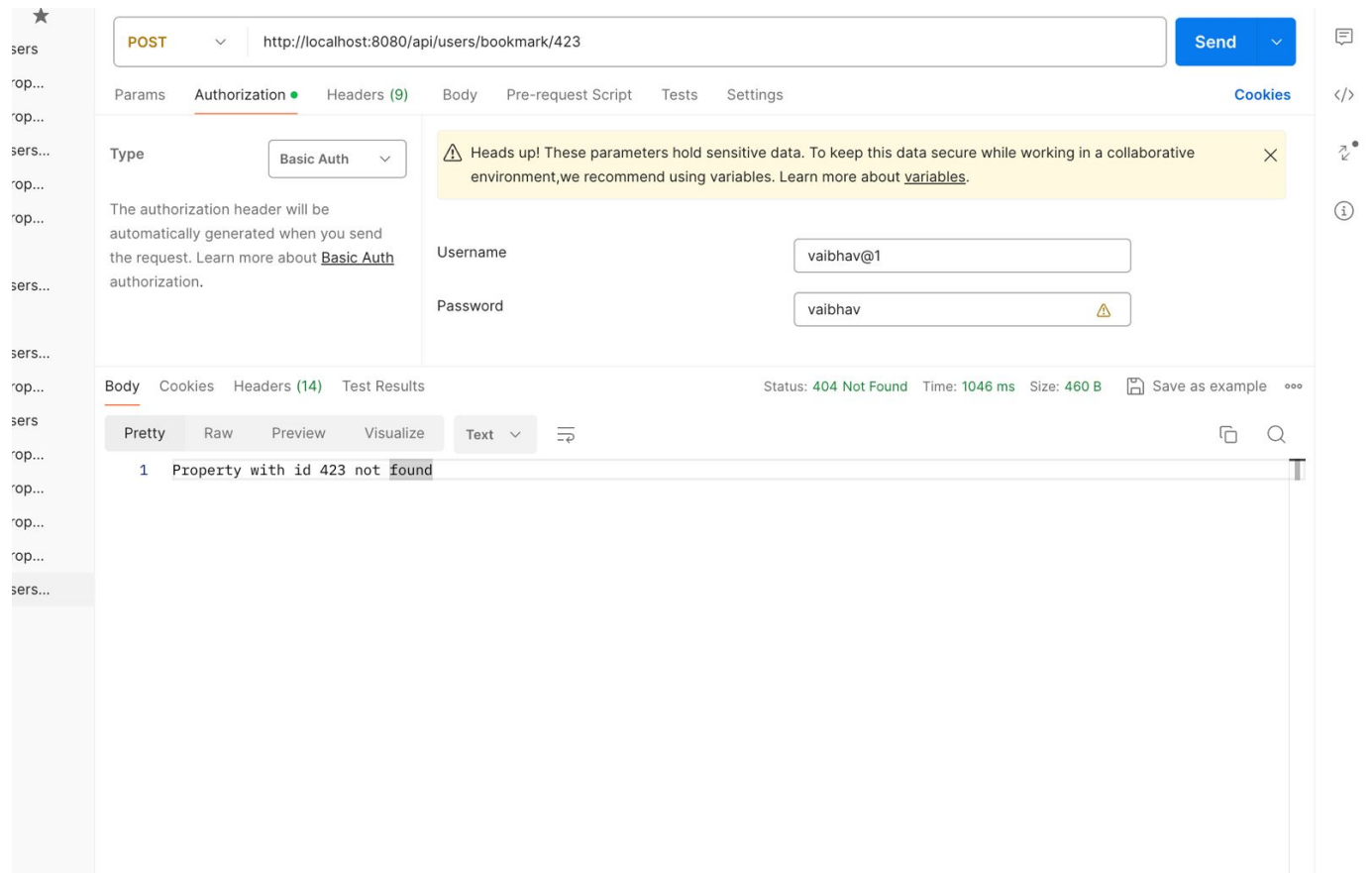


Figure 3.12: api/users/bookmark/id

3.13 Customer can delete a Property from his Bookmark list

The screenshot displays a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/api/users/bookmark/553
- Authorization:** Basic Auth (Username: vaibhav@1, Password: vaibhav)
- Response Status:** 200 OK, Time: 1429 ms, Size: 874 B
- Response Body (JSON):**

```
1 {
2   "id": 553,
3   "property_name": "Hillside Retreat",
4   "owner_name": "vamja vijaybhai",
5   "location": "Munnar",
6   "type": "Cottage",
7   "status": "Booked",
8   "propertyDetails": {
9     "id": 202,
10    "bedrooms": 4,
11    "bathrooms": 4,
12    "price": 1200,
13    "facilities": [
14      "maid"
15    ]
16  },
17  "user": {
18    "id": 353,
19    "firstName": "makvana",
20  }
21 }
```

Figure 3.13: api/users/bookmark/id

3.14 Customer can view Properties in his Bookmark list

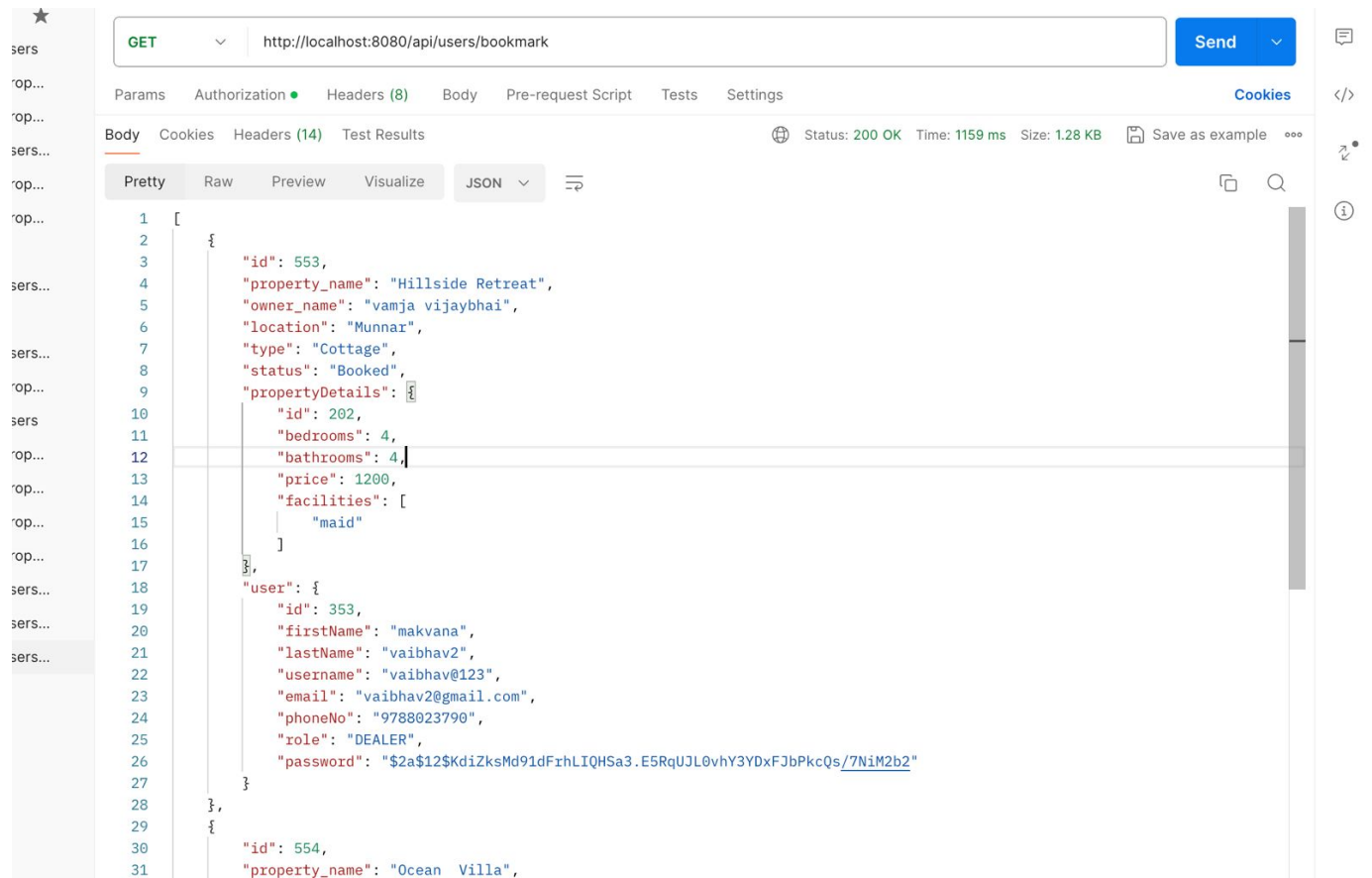


Figure 3.14: api/users/bookmark

3.15 Endpoint Lists


















SmartStay		
 / [GET]		Controller
 /api/properties [GET]		PropertyController
 /api/properties/ [POST]		PropertyController
 /api/properties/{n_th_property} [GET]		PropertyController
 /api/properties/{propertyId} [DELETE]		PropertyController
 /api/users [GET]		UserController
 /api/users [POST]		UserController
 /api/users/ [DELETE]		UserController
 /api/users/all [GET]		UserController
 /api/users/bookmark [GET]		UserController
 /api/users/bookmark/{book_id} [DELETE]		UserController
 /api/users/bookmark/{id} [POST]		UserController
 /api/users/{User_id} [DELETE]		UserController
 /api/propertyDetails/ [GET]		PropertyDetailsController
 /api/propertyDetails/{propertyId} [GET]		PropertyDetailsController
 /api/propertyDetails/{propertyId} [POST]		PropertyDetailsController
 /api/propertyDetails/{propertyId} [DELETE]		PropertyDetailsController

Figure 3.15: End Points

Chapter 4

Version Control

4.1 GitHub Repositories

- SmartStay Backend Lakhman's Repo: <https://github.com/lakhman108/smartstay.git>
- SmartStay Backend Vaibhav's Repo: <https://github.com/Vaibhav31mak/smartstay.git>

Chapter 5

Conclusion

In conclusion, the SmartStay real estate application developed using Spring Boot offers a comprehensive and secure platform for property owners and tenants to connect seamlessly. Through the use of advanced technologies and robust security features, the application simplifies the property management process and enhances the overall user experience while ensuring data confidentiality, integrity, and availability.

By leveraging the power of Spring Boot, the application ensures high performance, scalability, and security. The use of Spring Boot's dependency management, auto-configuration, and embedded server capabilities has greatly simplified the development process, allowing for faster deployment and easier maintenance.

Additionally, the integration of various security features such as user authentication, authorization, and secure communication protocols provides a safe environment for both property owners and tenants to interact. The application's security measures, coupled with its intuitive user interface and responsive design, ensure a seamless and protected experience across devices.

Overall, the SmartStay real estate application stands as a testament to the capabilities of Spring Boot in building modern, efficient, and secure web applications. Its successful implementation underscores the potential of Spring Boot in creating innovative solutions for real-world challenges while prioritizing the security and privacy of its users' data.