Test Document

for

Roomble

Version 1.0

Prepared by

Group #: 7 **Group Name: Marauders**

Aarsh Jain	**230015**	**aarshjain23@iitk.ac.in**
Aritra Ambudh Dutta	**230191**	**aritraad23@iitk.ac.in**
Aritra Ray	**230193**	**aritrar23@iitk.ac.in**
Bhukya Vaishnavi	**230295**	**bhukyav23@iitk.ac.in**
Bikramjeet Singh	**230298**	**bsingh23@iitk.ac.in**
Hitarth Makawana	**230479**	**hitarthkm23@iitk.ac.in**
Shlok Jain	**230493**	**jainshlok23@iitk.ac.in**
Ronav Puri	**230815**	**ronavgp23@iitk.ac.in**
Rathod Ayushi	**230844**	**rathoday23@iitk.ac.in**
Saksham Verma	**230899**	**sakshamv23@iitk.ac.in**
Surepally Pranaysriharsha	**231057**	**surepally23@iitk.ac.in**

Course: **CS253**

Mentor TA: ***Nij Bharatkumar Padariya***

Date: **04/04/2025**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
V1.0	Aarsh Jain Aritra Ambudh Dutta Aritra Ray Bhukya Vaishnavi Bikramjeet Singh Hitarth Makawana Shlok Jain Ronav Puri Rathod Ayushi Saksham Verma Surepally Pranaysriharsha	First Version of the Testing Document	04/04/2025

1 Introduction

Test Strategy:

We used manual testing to test our web application, **Roomble,** wherein each of the testers individually tested the application for various inputs or cases and matched the obtained result with the desired one.
The testers were keen to have a precise check on all the functional and non-functional requirements to be fulfilled by the application as per the **SRS Document.**

Testing Period:

Testing was an integral part of the development process, and we manually tested the code, but a major portion of the testing was done after the implementation process.
The partial testing during development made it easier for us to find errors during the final testing since any bugs encountered were easy to tackle and fix as a result of the knowledge of the code structure as well as the possible edge cases in mind. This made the testing efficient and less time-consuming.

Testers of the App:

Developers were the main testers of the main application due to lack of time and size of the team. However we ensured that the persons who developed a certain component did not test the same so that it was tested efficiently and with neutrality. Being the developers, we were well aware of the various input/test cases and hence carried out the testing extensively, ensuring the robustness of the platform. The various errors identified were corrected as and when discovered.

Coverage Criteria:

We have used functional and non-functional coverage. For these, we tried to stick to the **SRS Document** to guide our way through system testing.

Tools used for Testing:

We used **Postman** to conduct comprehensive testing of our web platform's APIs. It is a widely recognized and powerful API testing tool that enables developers to design, test, and document APIs efficiently. The key advantages that lured us into using it for testing as well as for manual checking during the development process:

- **Intuitive Interface:** Postman provides a clean and user-friendly interface, which greatly simplifies the process of creating, sending and analyzing API requests. It allows the testers to build test cases without dealing with complex configurations.
- **Seamless Integration:** Postman integrates seamlessly with various development workflows, supporting environments for automated testing, API documentation, and mock server setups. This ensures a smooth transition between the development and testing phases.
- **Robust Feature Set:** Postman offers a range of advanced features, including support for authentication mechanisms, automated testing with scripts, detailed request history tracking, and many more. Although we did not use all the features exhaustively, the few we used greatly helped us in streamlining the testing process.
- **Cross-Platform & Cloud Support:** Postman is available as a desktop application and a cloud-based tool, allowing developers to sync their work across devices and collaborate efficiently.

By leveraging this tool, we ensured thorough API testing of the web platform.

2 Unit Testing

1a. SignUp-Tenant-Backend

In this unit we tested the backend function through which a new tenant can sign up on **Roomble.**

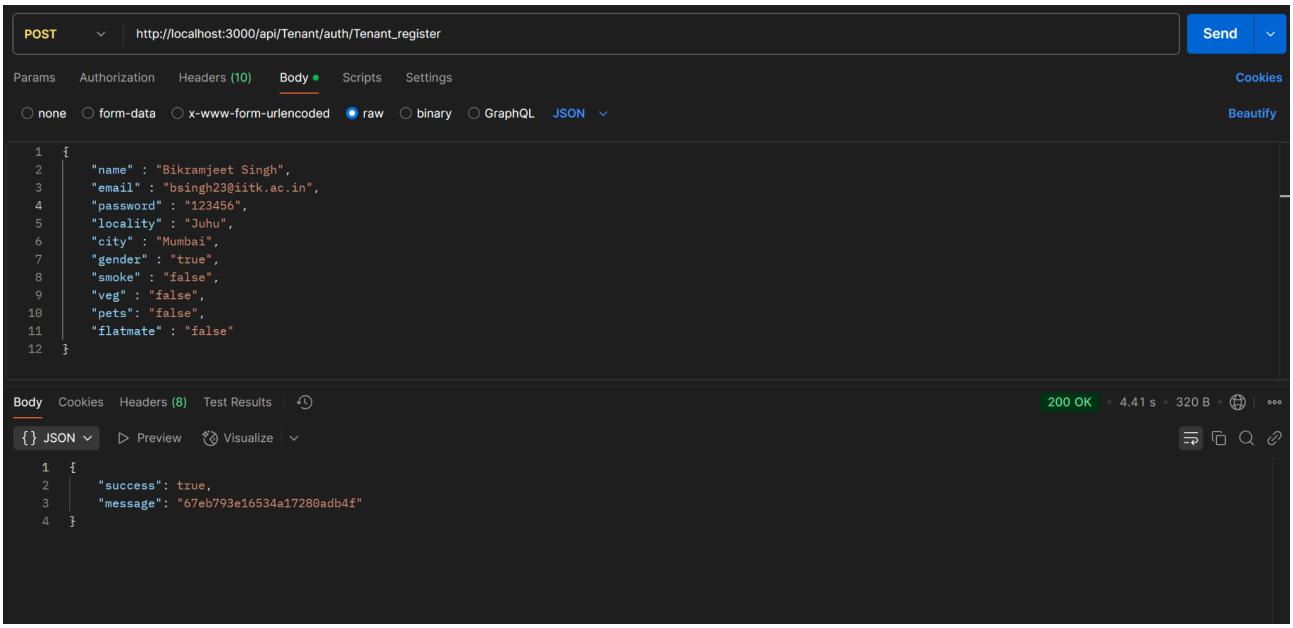
Unit Details: Testing of this unit ensures that a new user is registered correctly with the entered details. It also ensures that no two users are registered with the same email.

Test Owner: Bikramjeet Singh

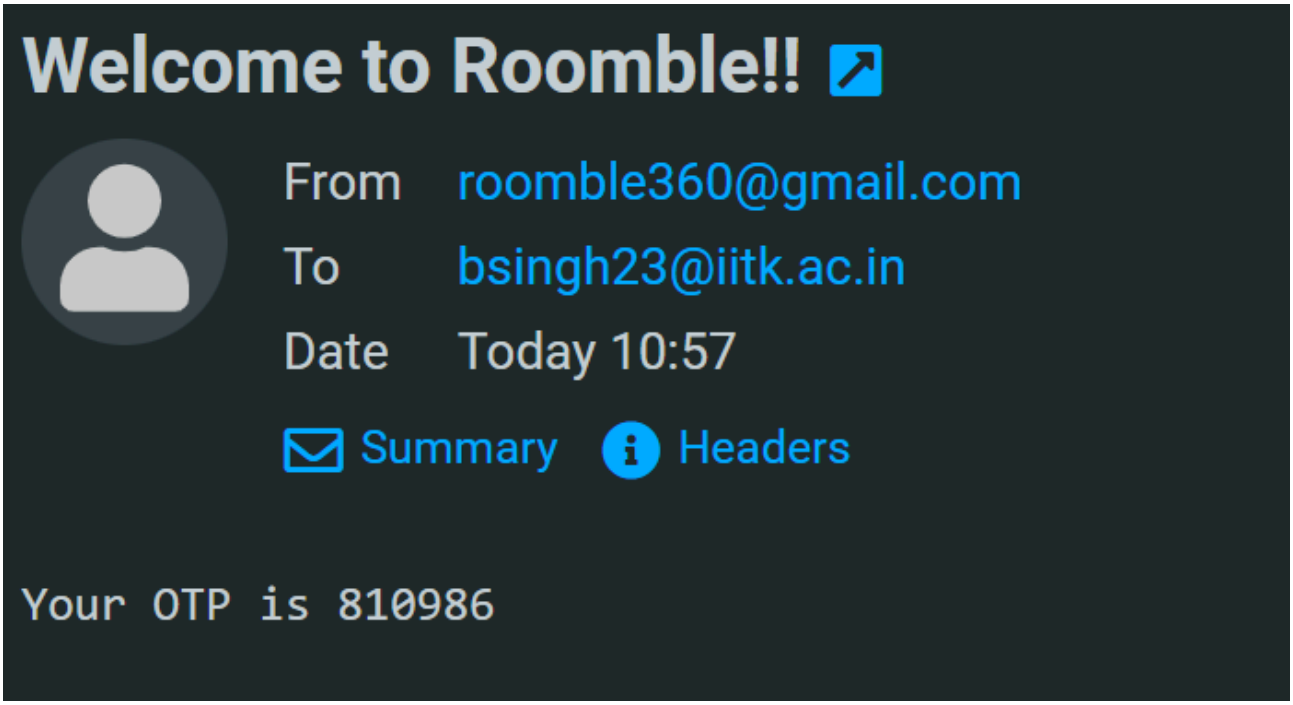
Test Date: 01/04/2025

Test Results: A new user was correctly registered. An object of the class “Tenant” was created. If the email already existed a prompt was shown alerting the user about the same.

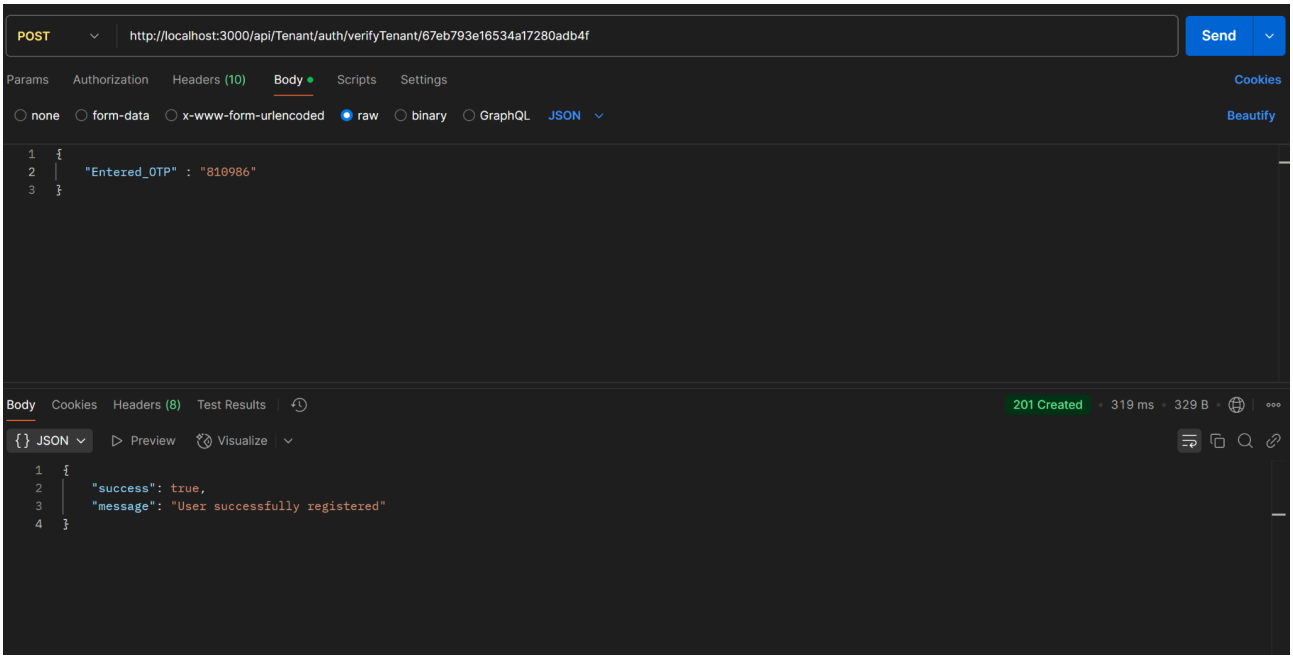
- New Email:



For a new email, at first an ID is generated and an email is sent with an OTP which is used for verification.



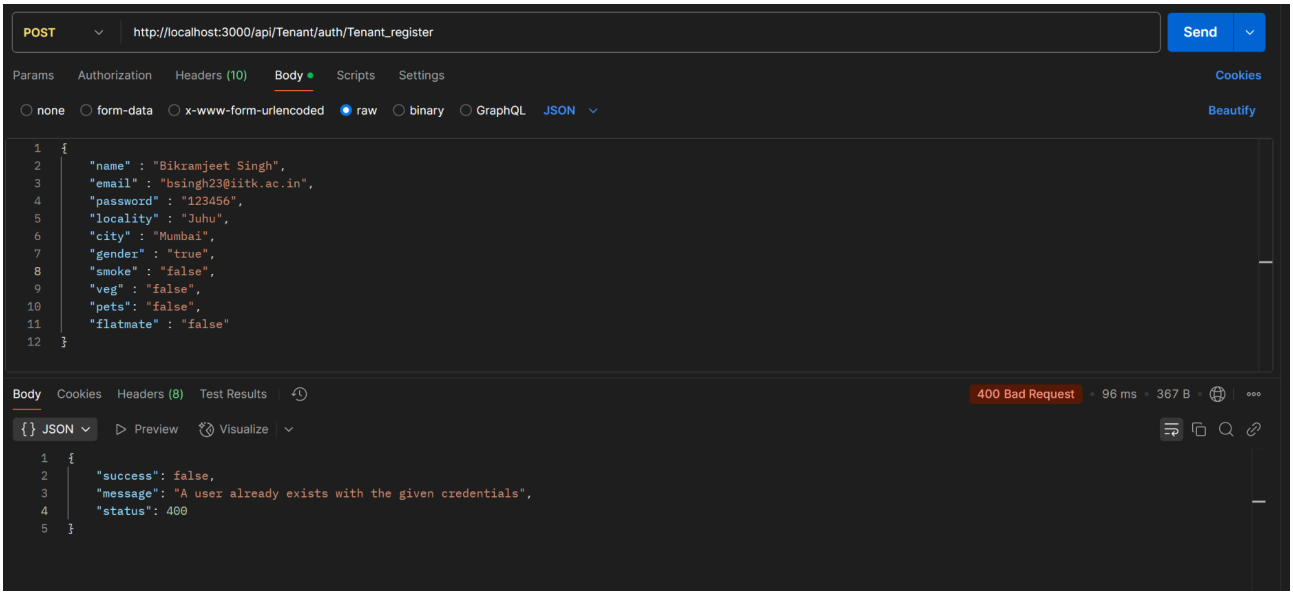
After sending the OTP along with the ID in the params
The user is successfully registered.



The user along with his/her details, is available now in the Tenant Database.

```
_id: ObjectId('67eb7a3d16534a17280adb52')
name : "Bikramjeet Singh"
type : "tenant"
email : "bsingh23@iitk.ac.in"
password : "$2b$10$mAoIeheiepmJthU0Y0hx7ucfBrVNv1W1Lct.8hfPmTGIXrUqOtJ3O"
locality : "Juhu"
city : "Mumbai"
gender : true
smoke : false
veg : false
pets : false
flatmate : false
description : "This user hasn't setup a description yet"
▸ conversations : Array (empty)
▸ bookmarks_tenants : Array (empty)
▸ bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
▸ reviews : Array (empty)
__v : 0
```

- Existing Email:



The user is prompted regarding the existence of an account with the same credentials.

Structural Coverage: Functional Coverage (related to register route in the backend), Decision coverage (covers the check for existing user with the same email).
Additional Comments: None

1b. SignUp-Tenant-Frontend

In this unit we tested the frontend part of the signup-tenant functionality.

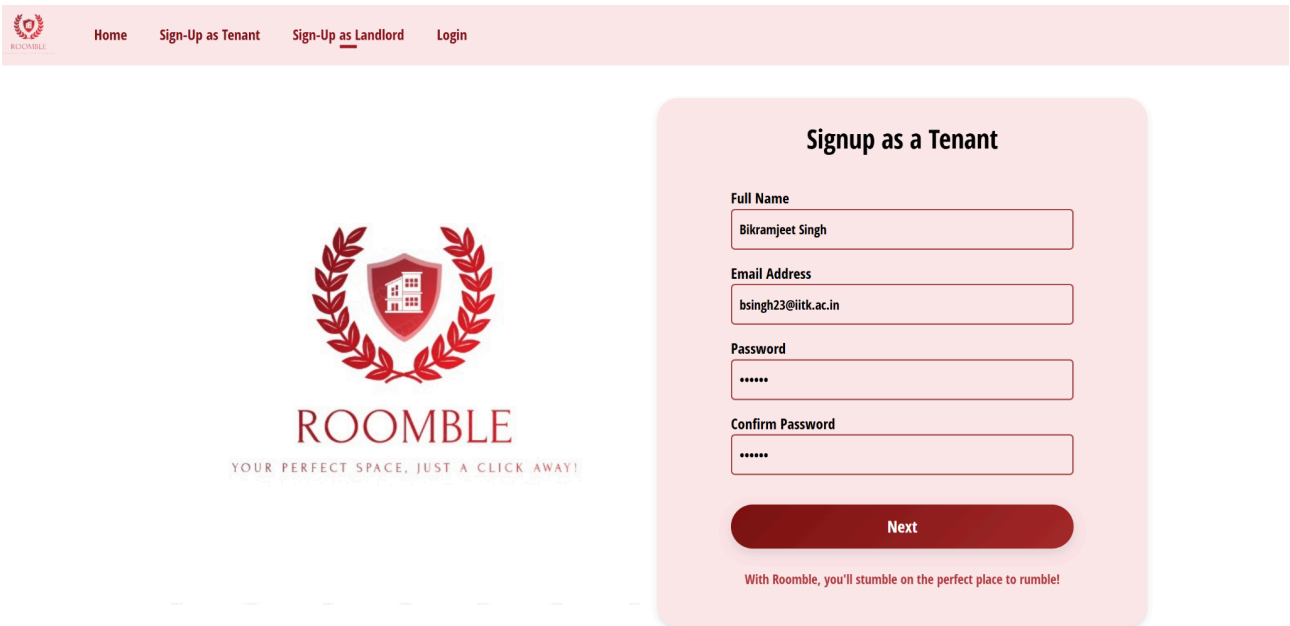
Unit Details: Testing of this unit ensures that a new user can easily sign up through the user-friendly interface of our application.

Test Owner: Bikramjeet Singh

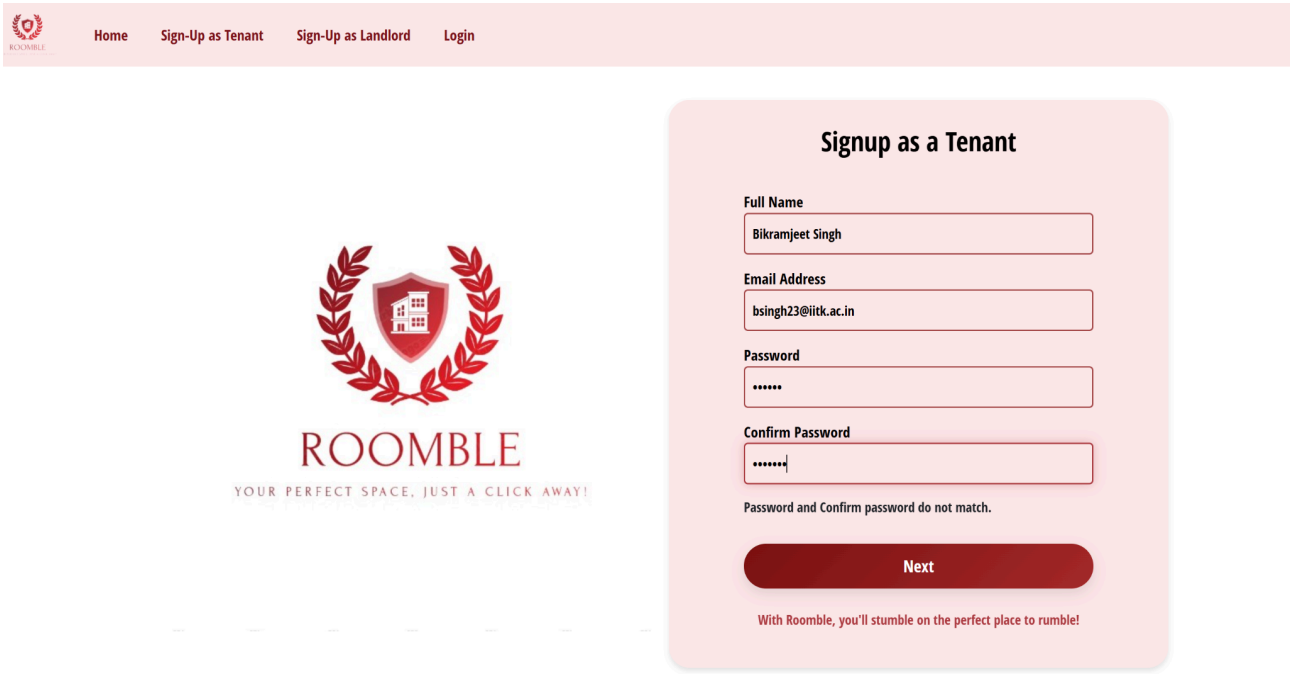
Test Date: 01/04/2025

Test Results: A new tenant can easily sign-up by first entering the email, password and the necessary details.

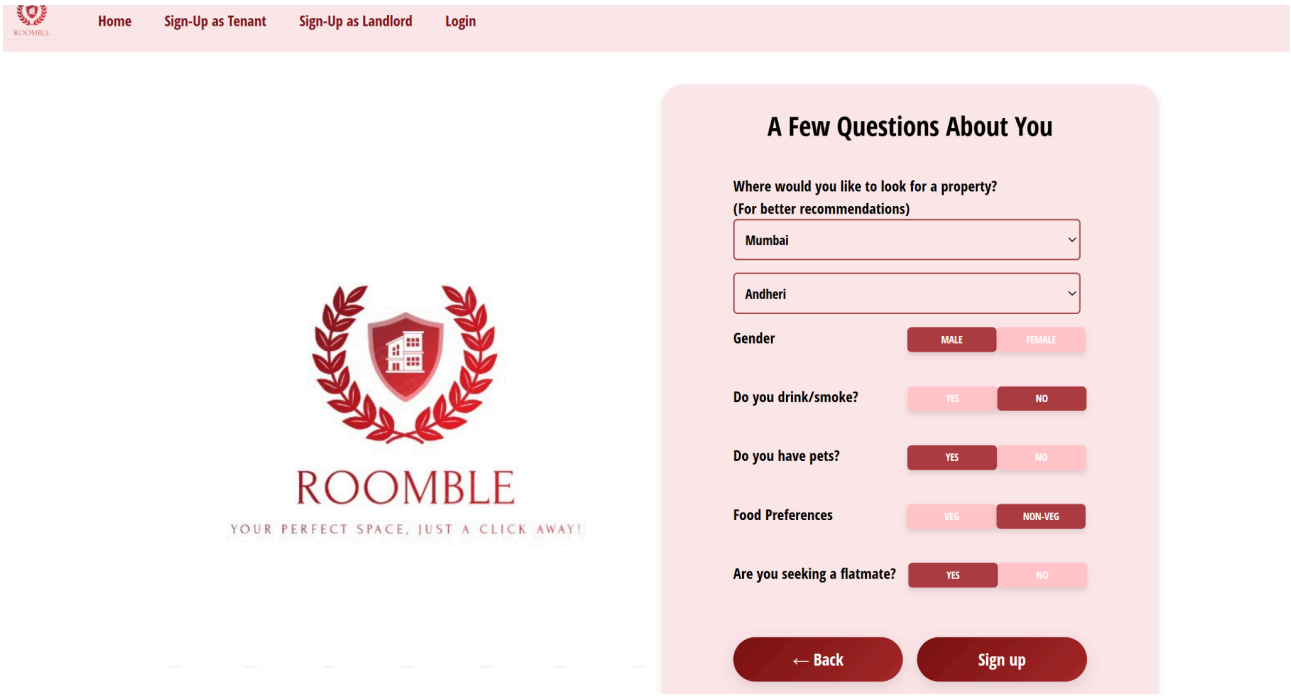
- New Email:



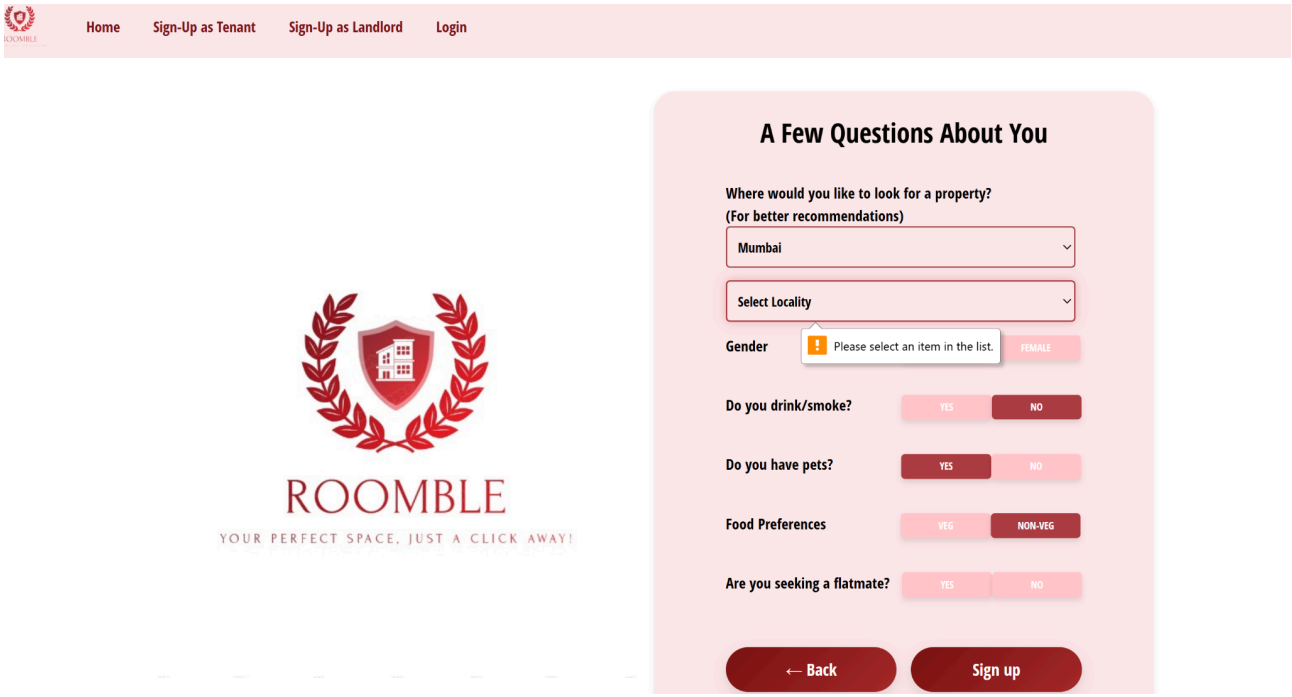
User enters the email, password, and the matching password



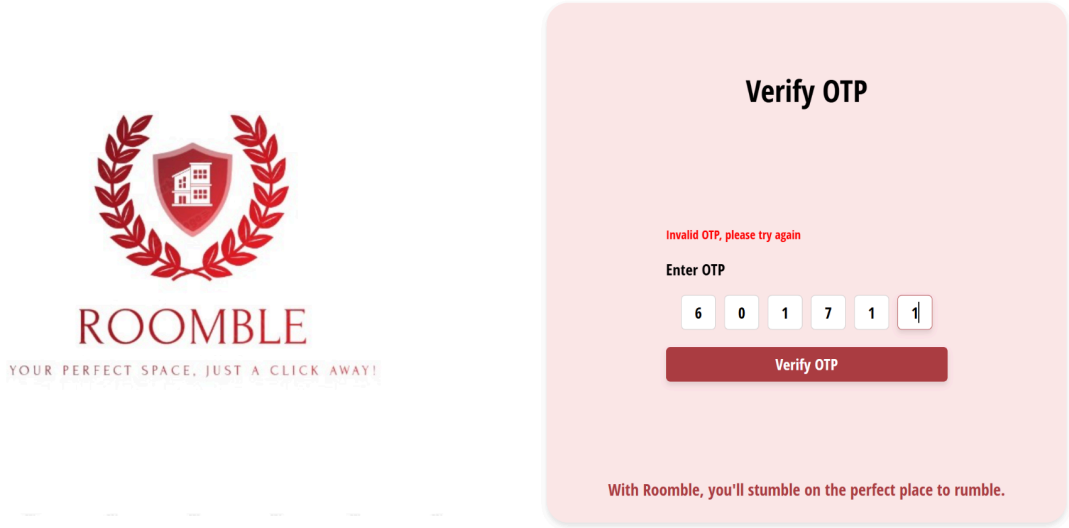
If the passwords do not match the user is prompted to correct it.



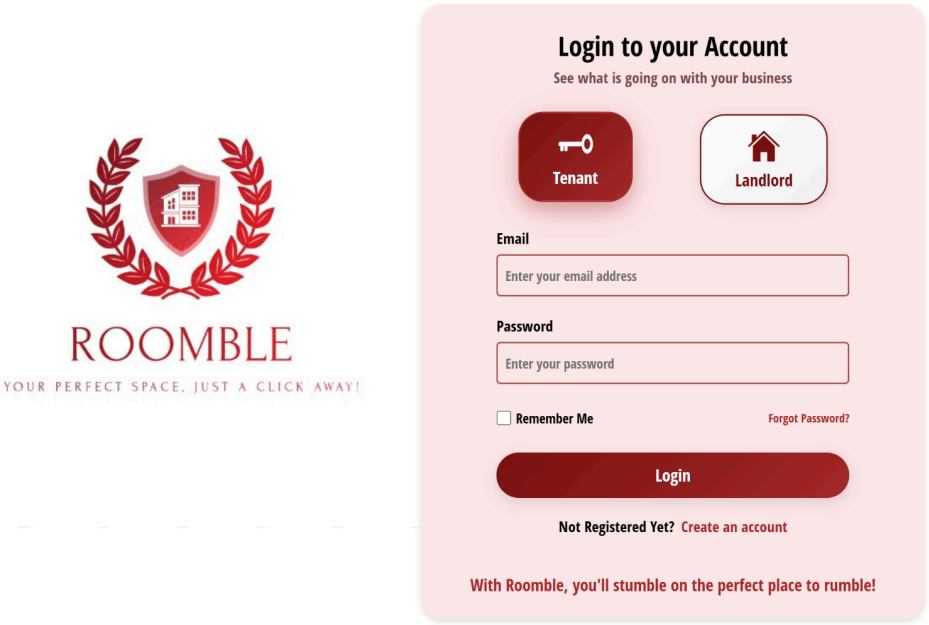
Afterwards the user is asked to fill a form with some necessary details if he/she misses some of them a related prompt alerts the user regarding the same.



Afterward, an OTP is sent to the concerned mail of the user and the user is directed to the following page, wherein he/she needs to enter the correct OTP, entering the wrong OTP gives an error message.



Entering the correct OTP, redirects the user to the login page:



1c. SignUp-Landlord-Backend

In this unit, we tested the backend function that ensures that a new landlord can sign up on **Roomble.**

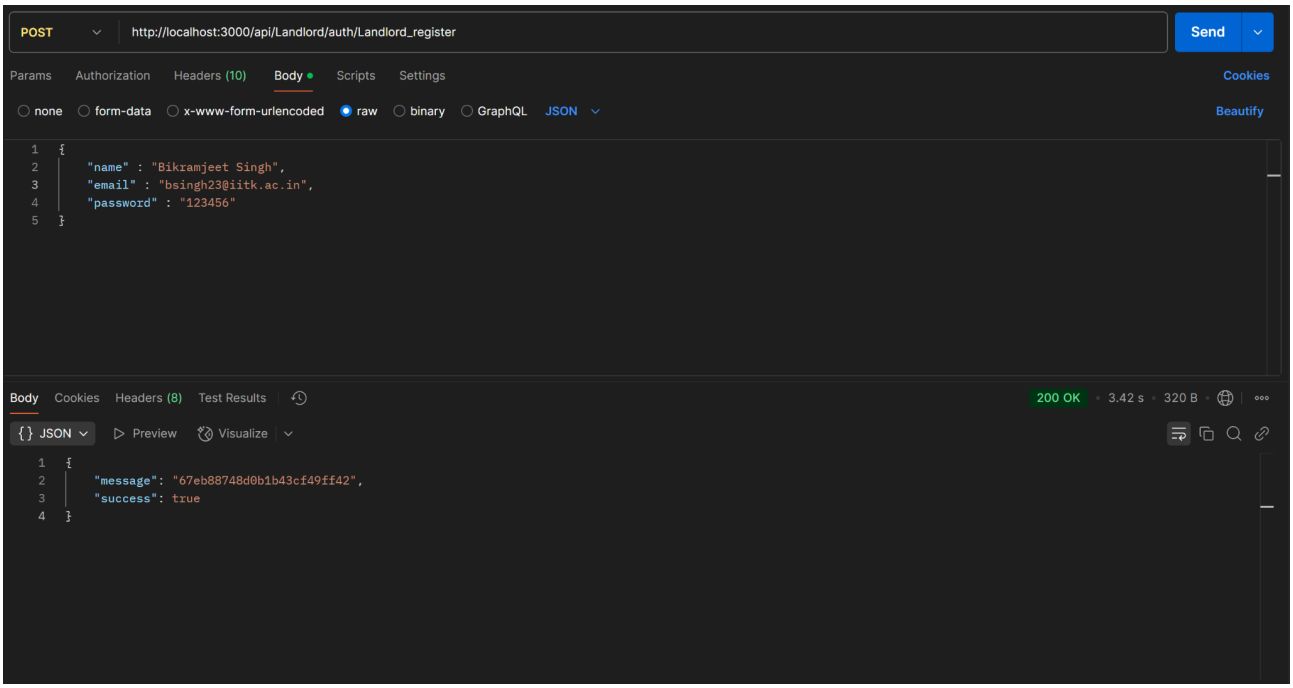
Unit Details: Testing of this unit ensures that a new user is registered correctly with the entered details. It also ensures that no two users are registered with the same email.

Test Owner: Bikramjeet Singh

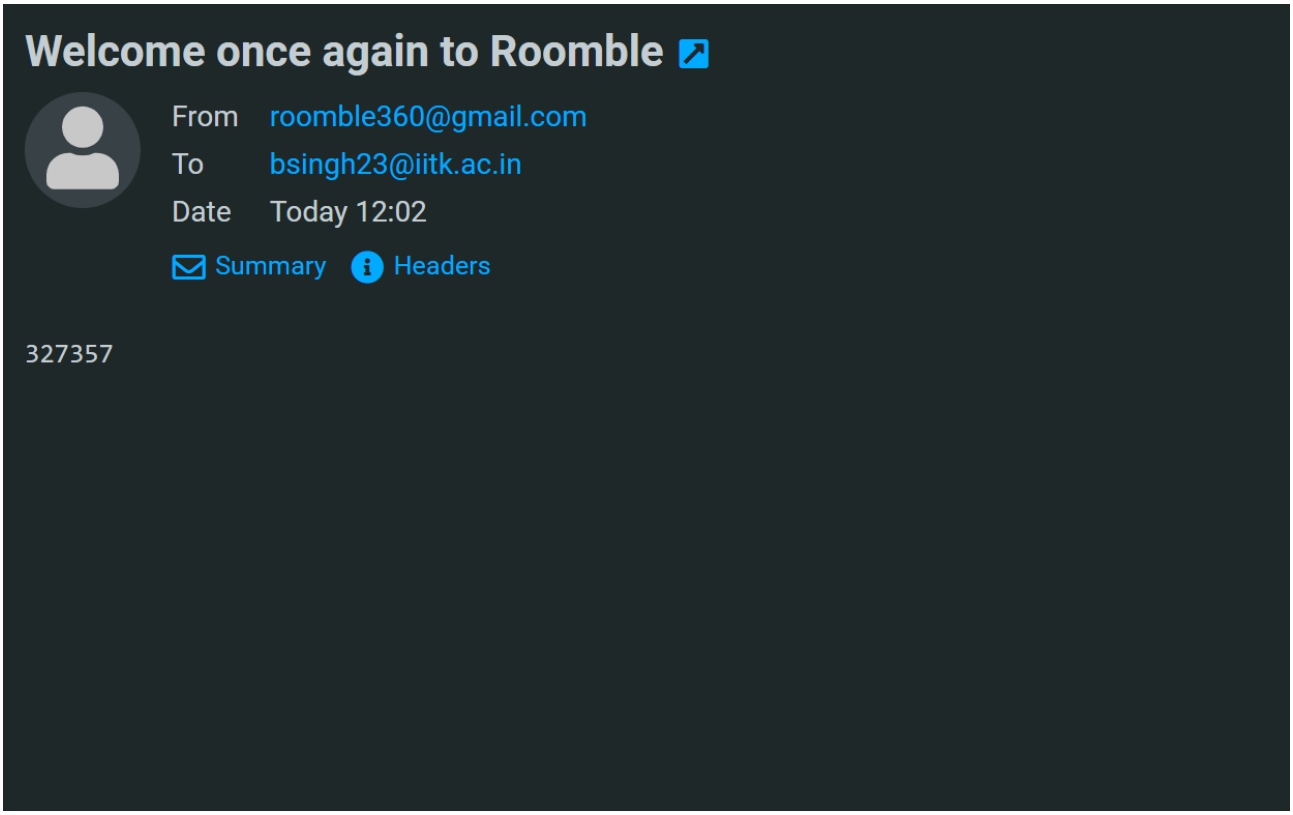
Test Date: 01/04/2025

Test Results: A new user was correctly registered. An object of the class “Tenant” was created. If the email already existed a prompt was shown alerting the user about the same.

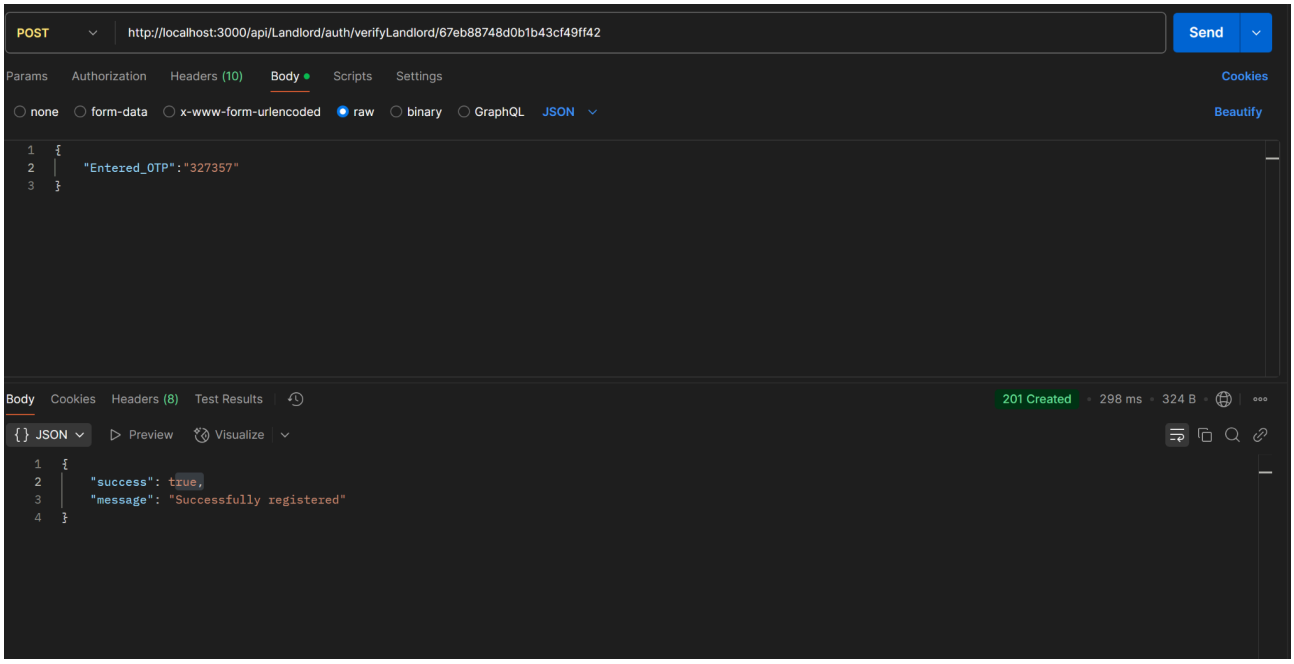
- New Email:



For a new email, at first, an ID is generated and an email is sent with an OTP, which is used for verification.

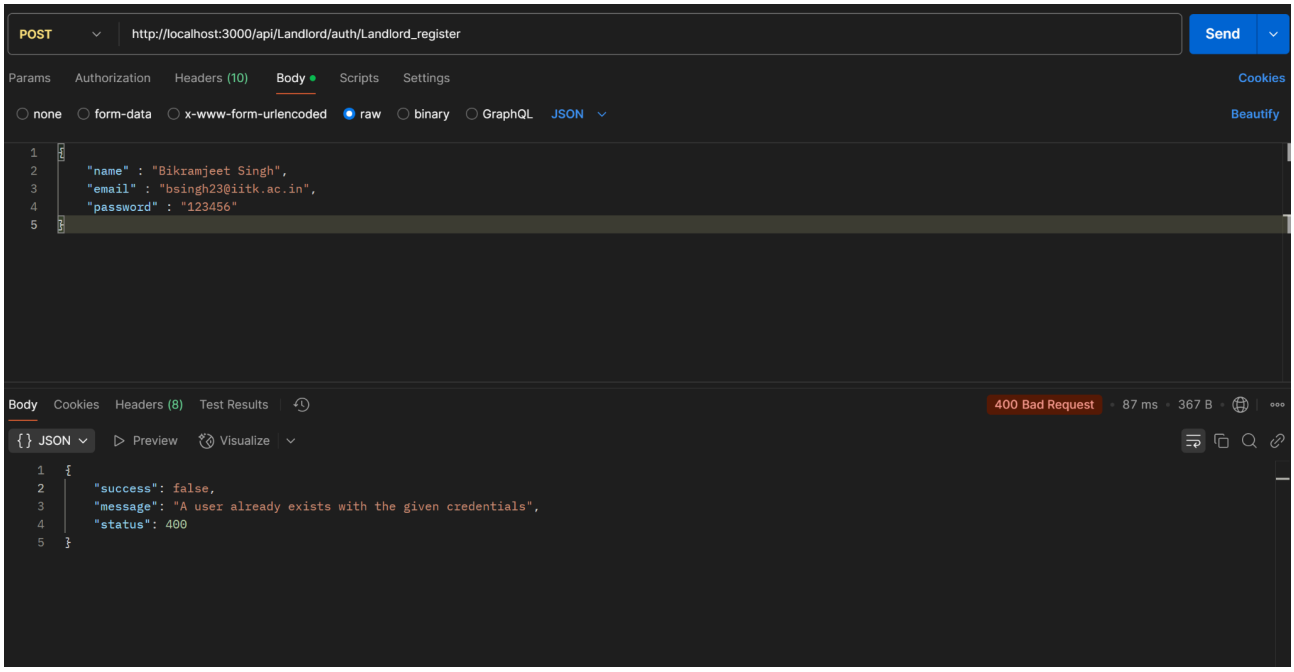


After sending the OTP along with the ID in the params
The user is successfully registered.



The user along with his/her details, is available now in the Tenant Database.

- Existing Email:
  The user is prompted regarding the existence of an account with the same credentials.



Structural Coverage: Functional Coverage (related to register route in the backend), Decision coverage (covers the check for existing user with the same email).
Additional Comments: None

1d. SignUp-Landlord-Frontend

In this unit we tested the frontend part of the signup-landlord functionality.

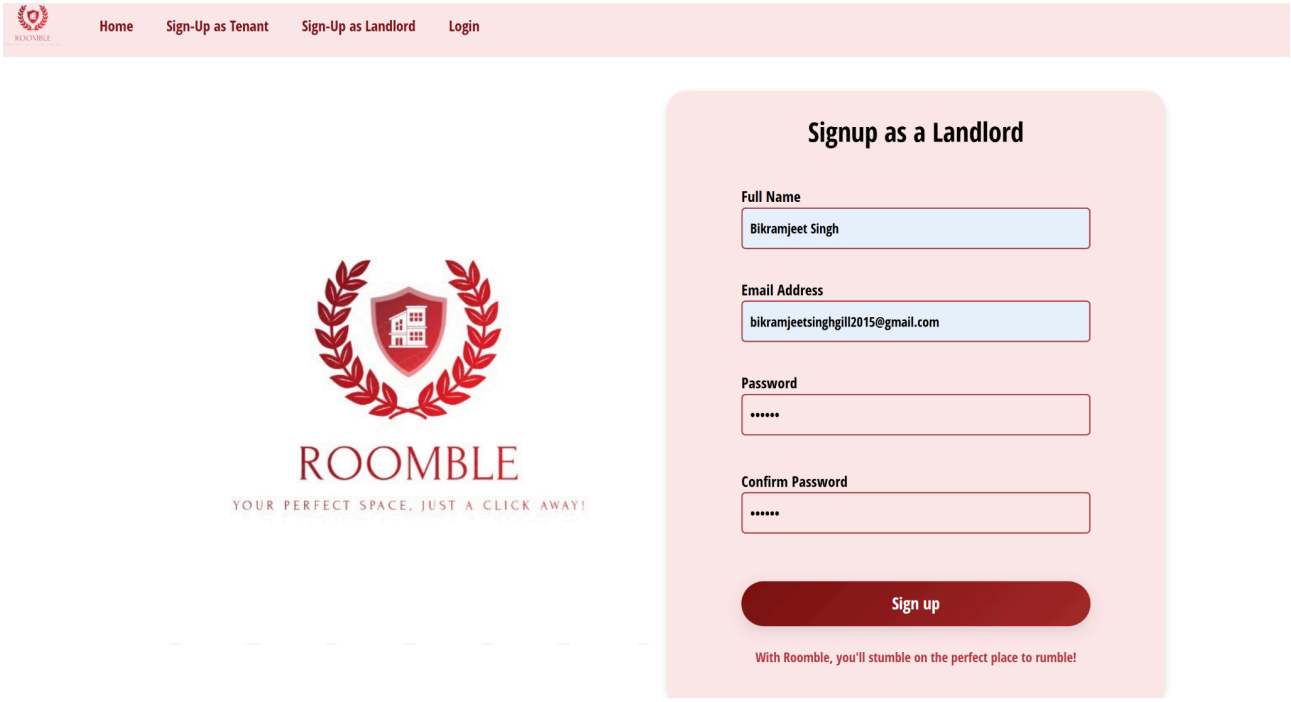
Unit Details: Testing of this unit ensures that a new user can easily sign-up through the user-friendly interface of our application.

Test Owner: Bikramjeet Singh

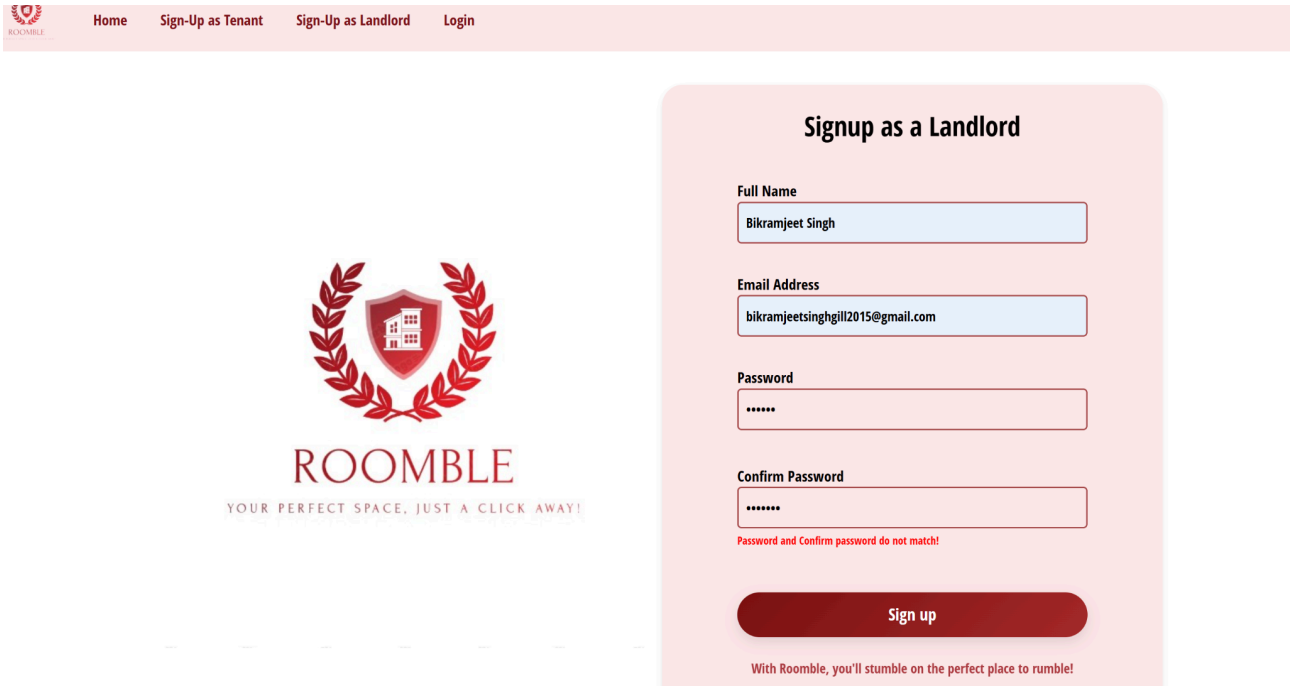
Test Date: 01/04/2025

Test Results: A new tenant can easily sign-up by first entering the email, password and the necessary details.

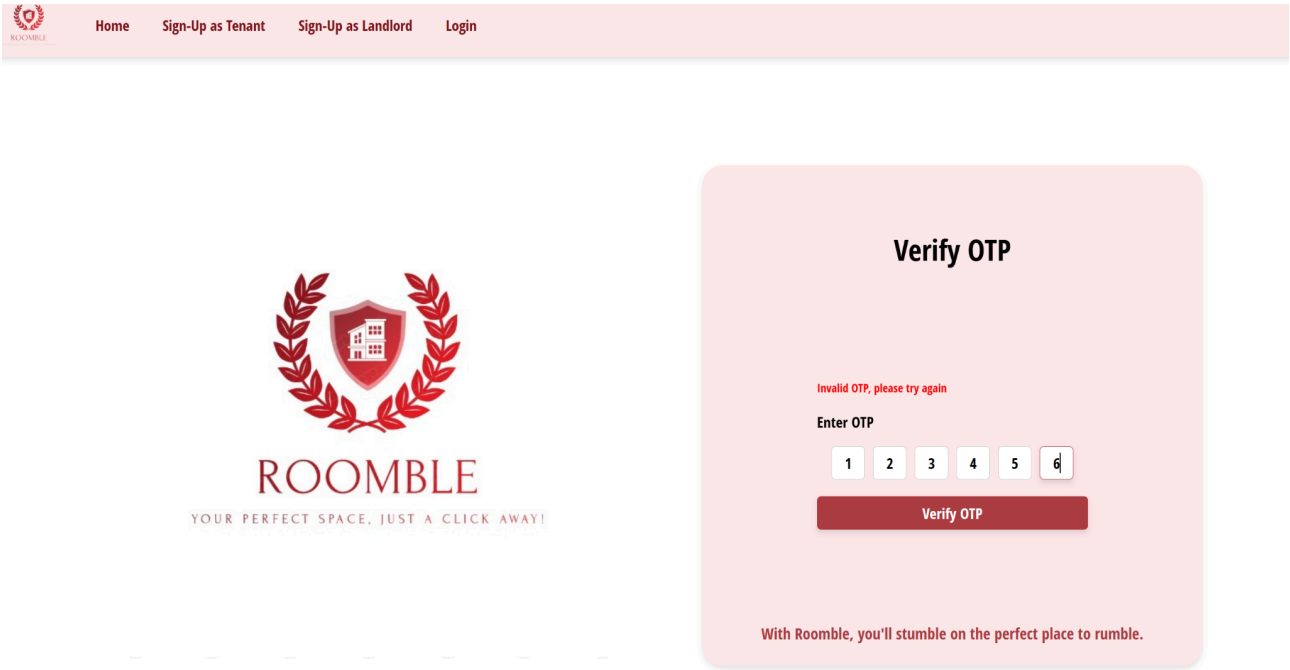
- New Email:



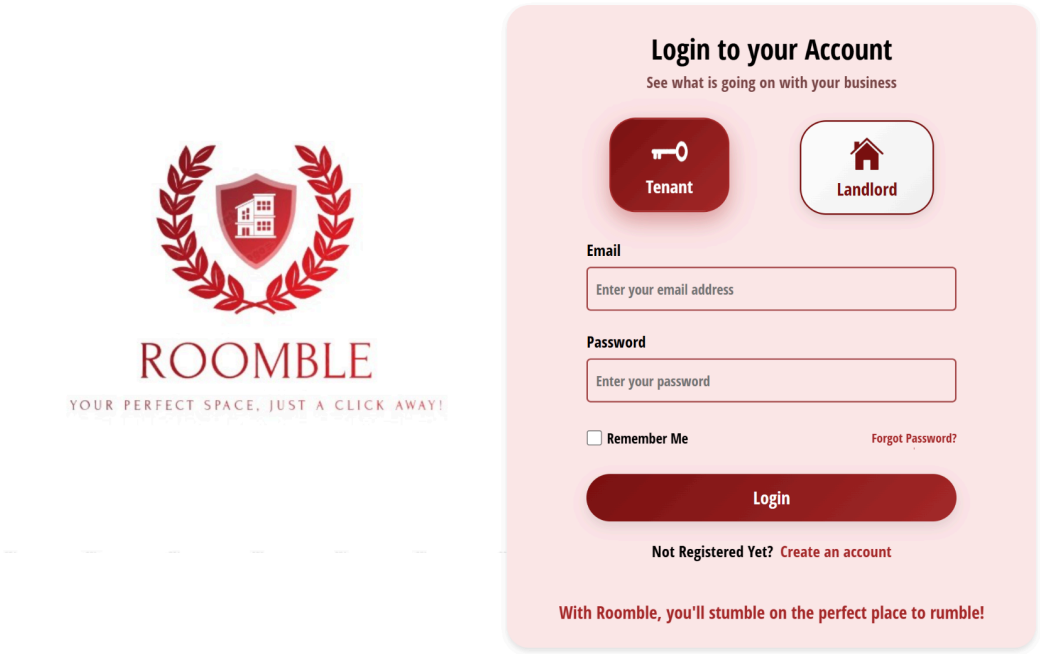
User enters the email, password, and the matching password



If the passwords do not match the user is prompted to correct it.



Afterward, an OTP is sent to the concerned mail of the user and the user is directed to the following page, wherein he/she needs to enter the correct OTP,entering the wrong OTP gives an error message.



Entering the correct OTP, redirects the user to the login page:

2a. Login-Tenant-Backend

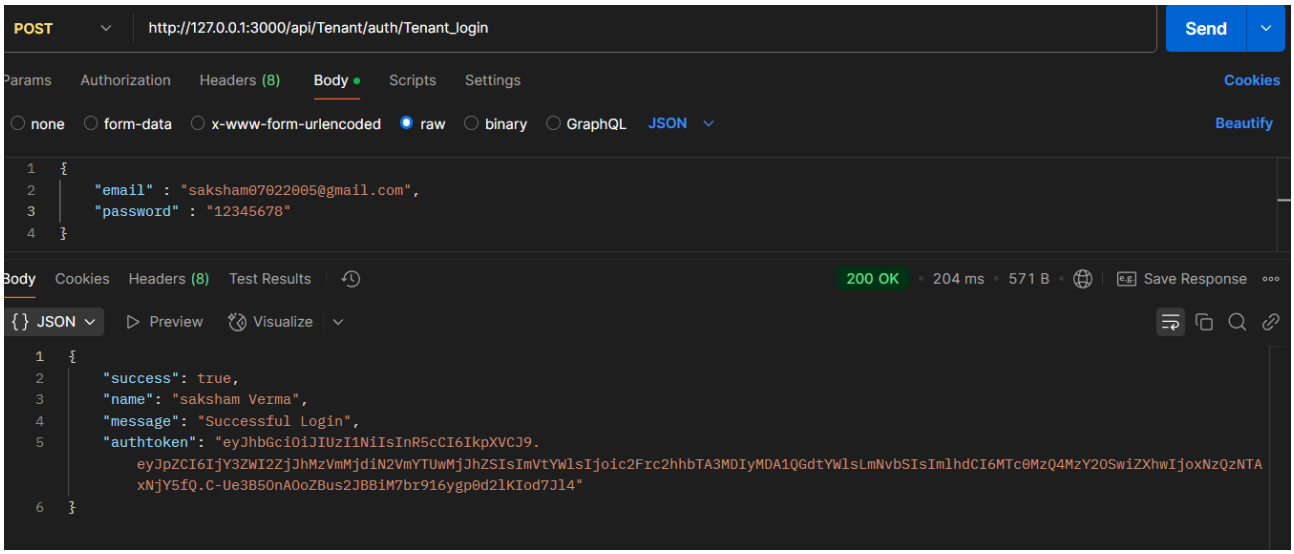
In this unit we have tested the backend function that logs in a Tenant.

Unit Details: Testing of this unit ensures that a valid Tenant is allowed to sign in, if the entered details are correct. It also ensures that an unauthenticated Tenant is not allowed to sign in.

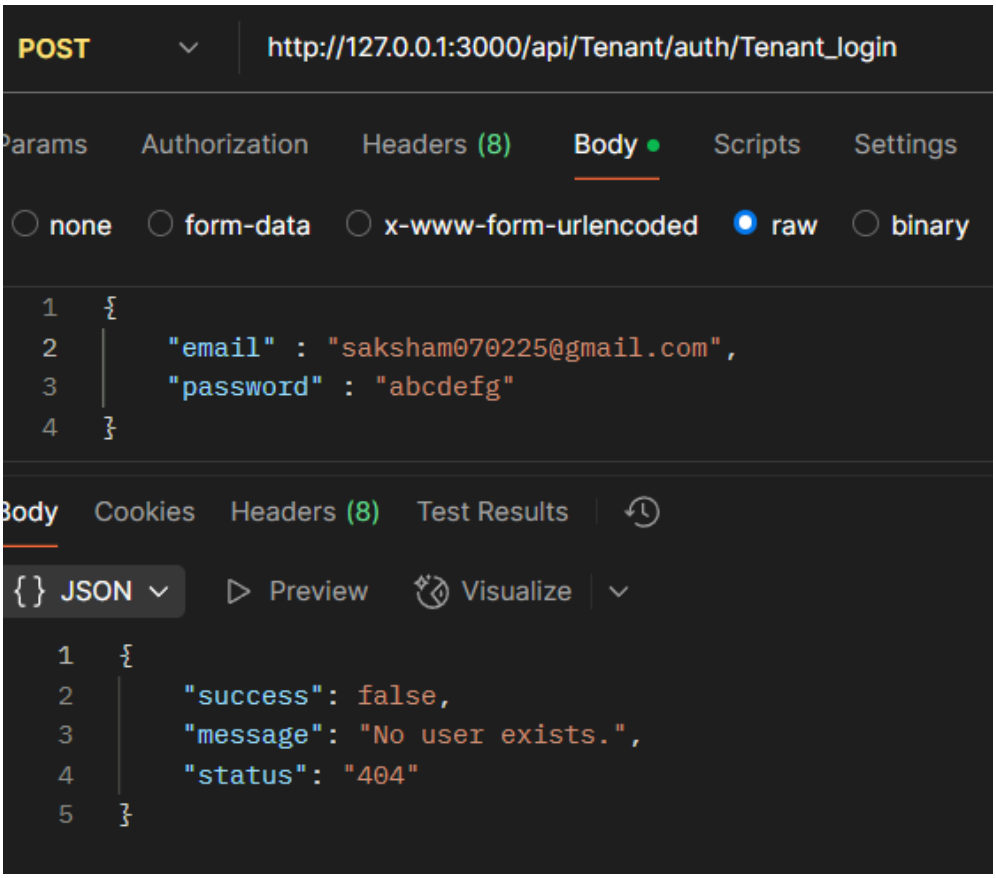
Test Owner: Saksham Verma

Test Date: 01/04/2025

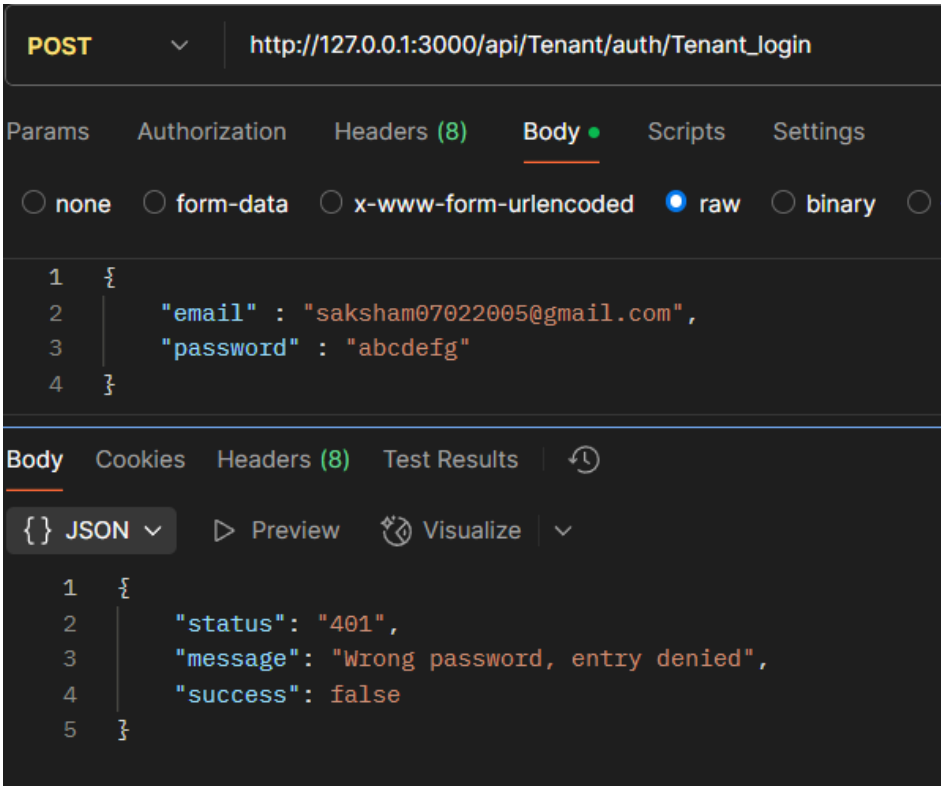
Test Results: An authenticated Tenant(valid details) was allowed to log in successfully. An unauthenticated Tenant was prevented from logging in.



Upon successful login, the tenant receives an Authtoken with success = true



If the User isn't registered, he receives success = false, with message = "No user exists"



If the User tries to Login with wrong Password, he receives success = false, and message = "Wrong password, entry denied"

Structural Coverage: Functional coverage (covers Tenant_Auth), Branch Coverage checks for Tenant Existence and valid credentials

Additional Comments: None

2b. Login-Tenant-Frontend

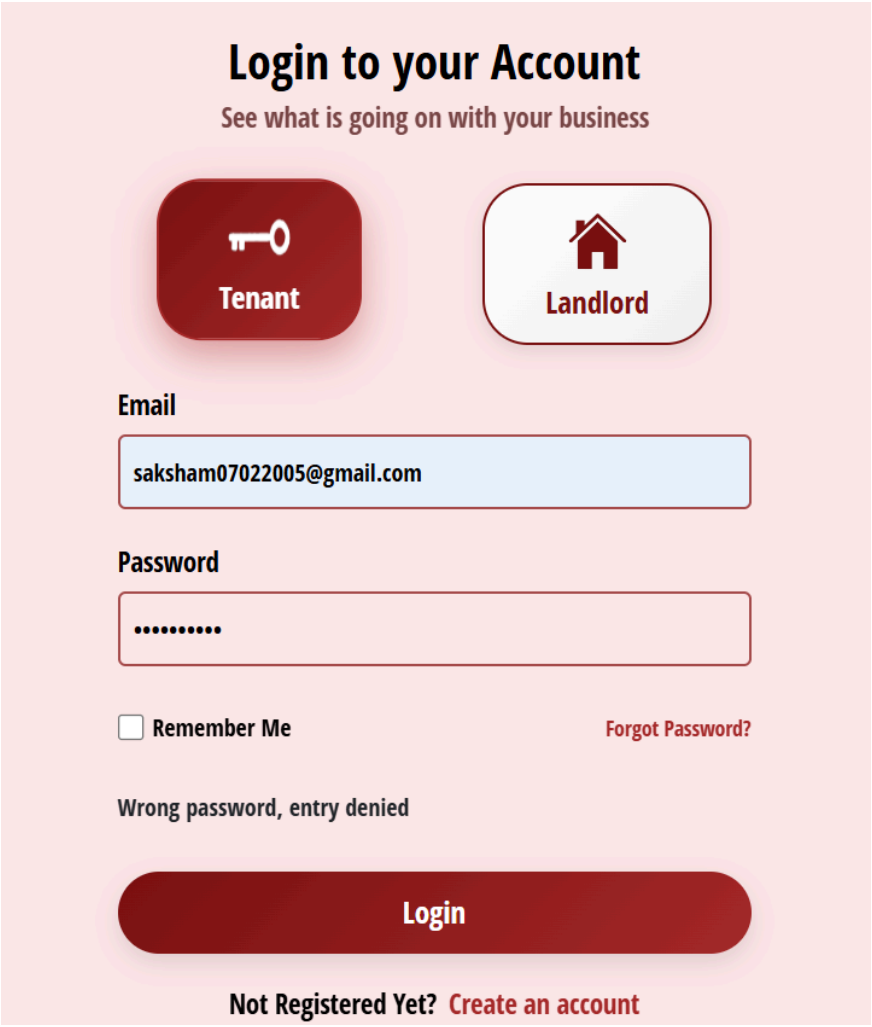
In this unit we have tested the backend function that logs in a Tenant

Unit Details: Testing of this unit ensures that a valid Tenant is allowed to sign in, if the entered details are correct. It also ensures that an unauthenticated Tenant is not allowed to sign in.

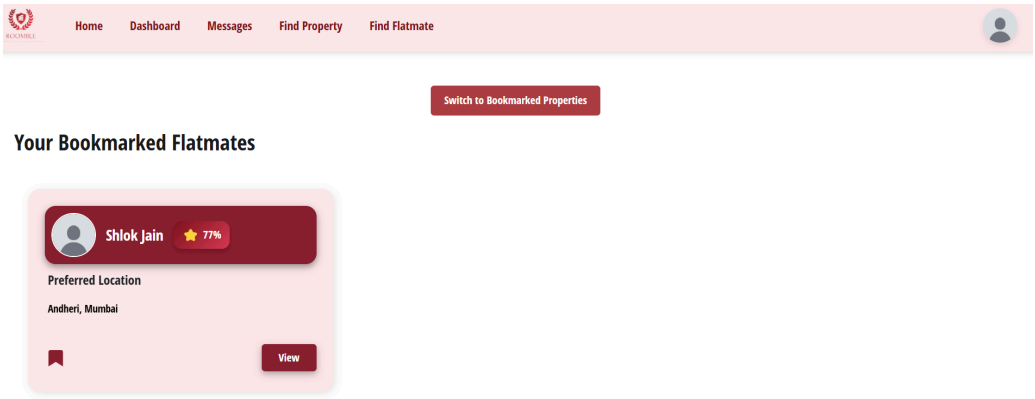
Test Owner: Saksham Verma

Test Date: 01/04/2025

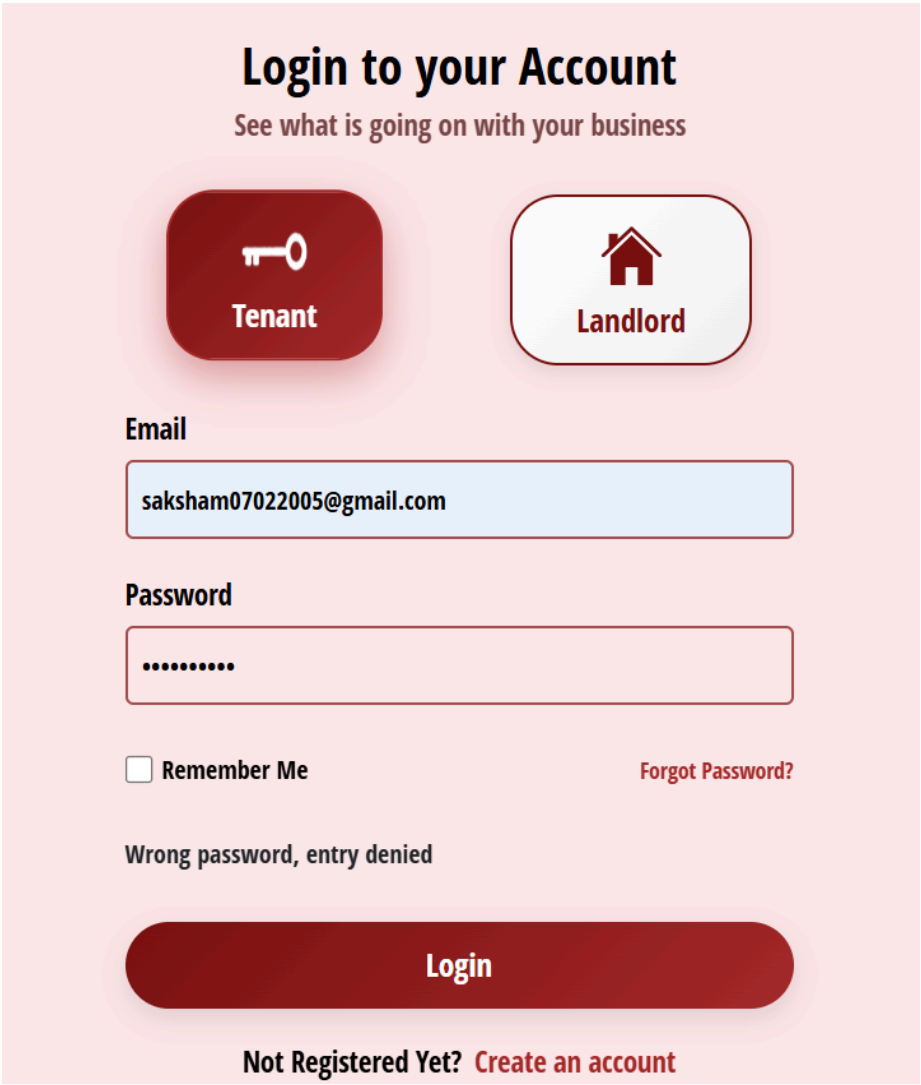
Test Results: An authenticated Tenant(valid details) was allowed to log in successfully. An unauthenticated Tenant was prevented from logging in.



Upon entering wrong password, the Tenant is not allowed to Log in.



Upon entering correct credentials a Tenant is redirected to his Homepage.



Upon entering a wrong Password the Tenant isn't allowed to log in

Structural Coverage: Functional coverage (covers Tenant_Auth), Branch Coverage checks for Tenant Existence and valid credentials

Additional Comments: None

2c. Login-Landlord-Backend

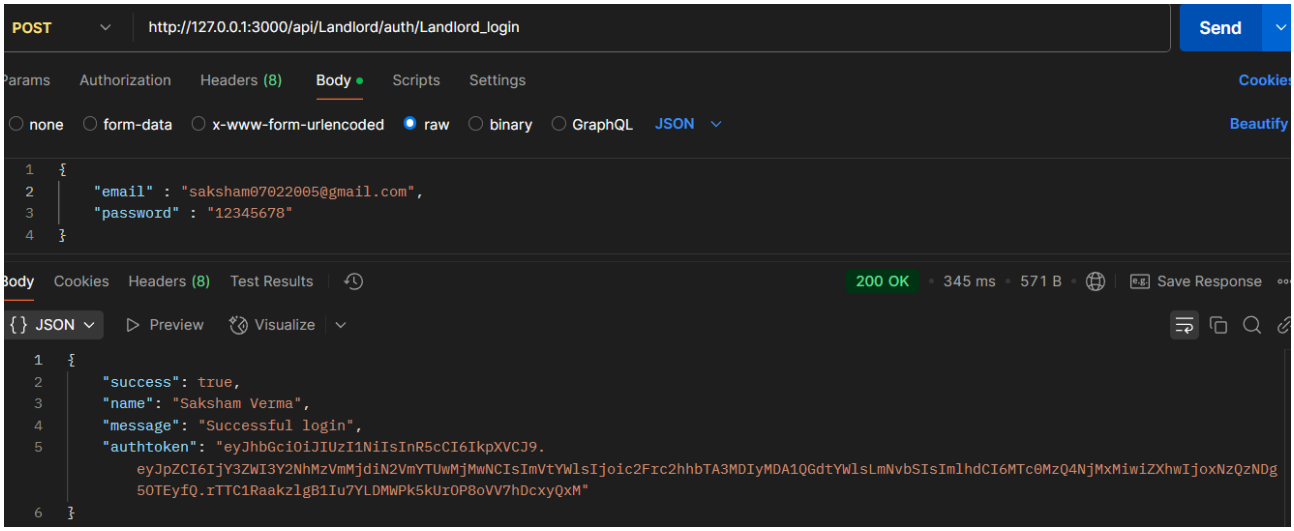
In this unit we have tested the backend function that logs in a Landlord

Unit Details: Testing of this unit ensures that a valid Landlord is allowed to sign in, if the entered details are correct. It also ensures that an unauthenticated Landlord is not allowed to sign in.

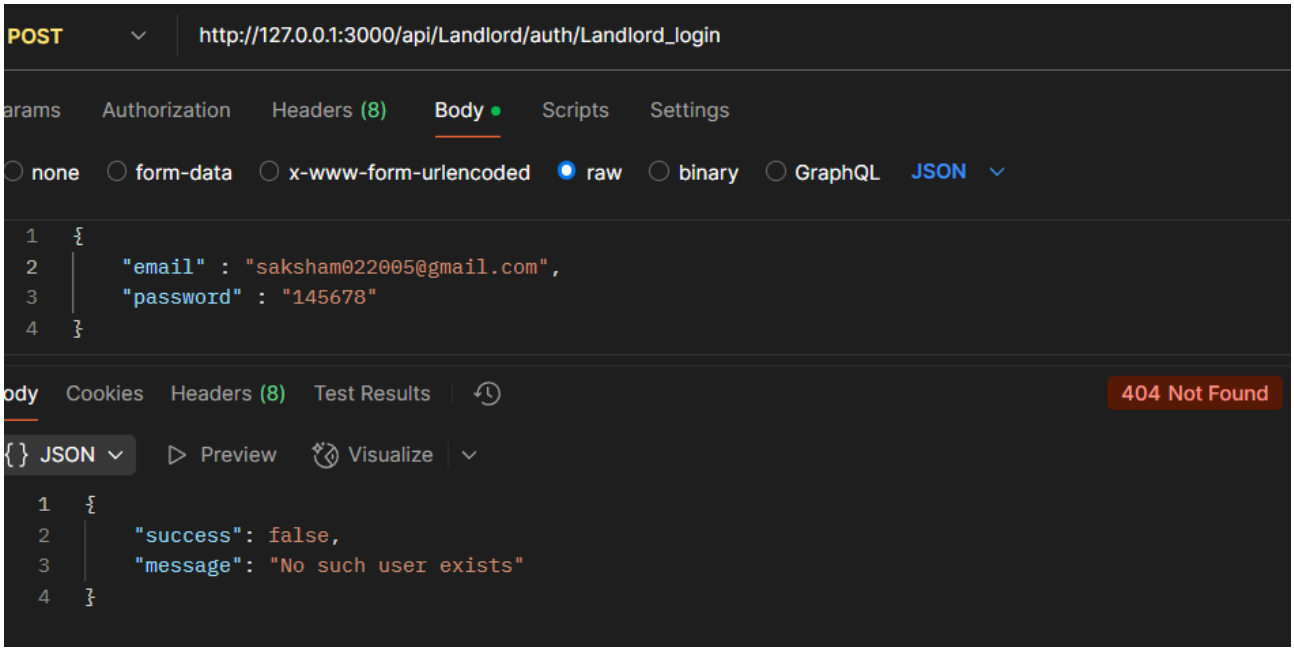
Test Owner: Saksham Verma

Test Date: 01/04/2025

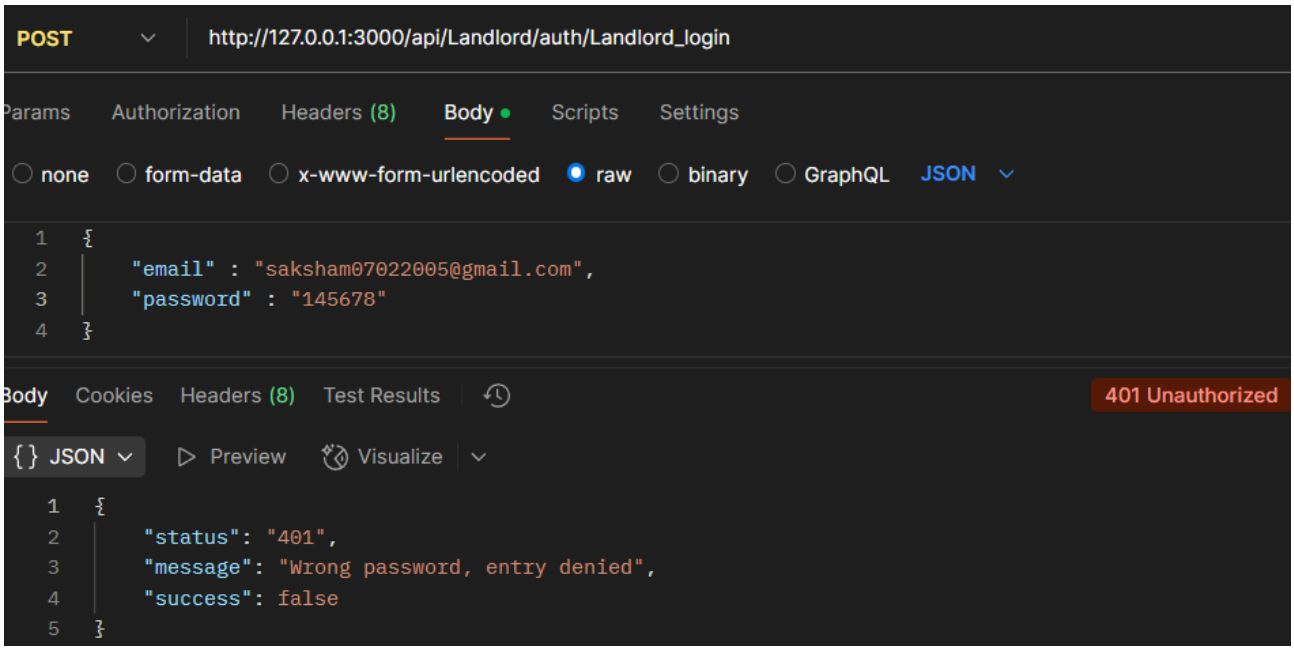
Test Results: An authenticated Landlord(valid details) was allowed to log in successfully. An unauthenticated Landlord was prevented from logging in.



Upon entering correct credentials, a success code of 200 and authtoken is returned



Upon entering an invalid email, a success code of 200 is returned



Upon entering wrong password, a success code of 401 is returned

Structural Coverage: Functional coverage (covers Landlord_Auth), Branch Coverage checks for Landlord Existence and valid credentials

Additional Comments: None

2d. Login-Landlord-Frontend

In this unit we have tested the frontend function that logs in a Landlord

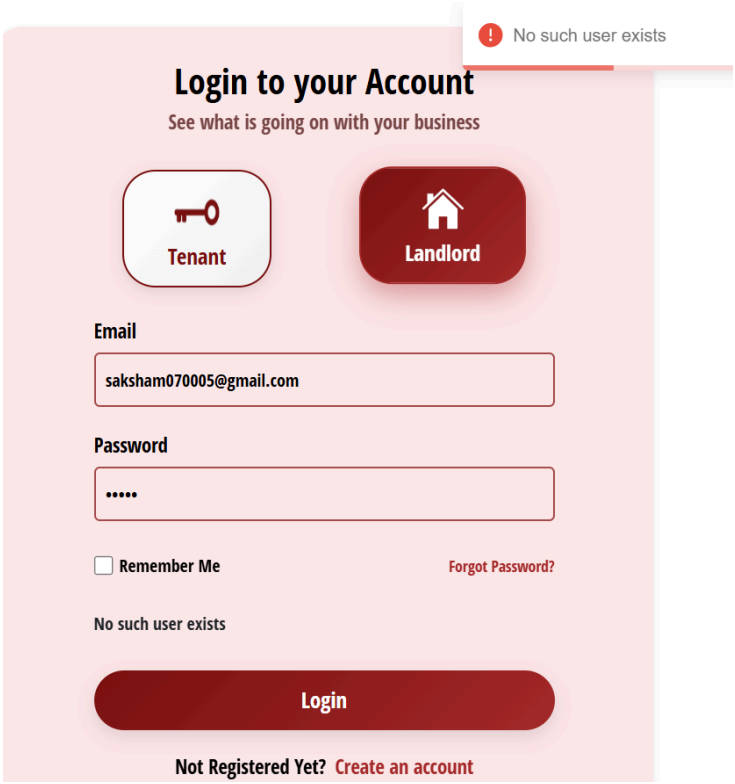
Unit Details: Testing of this unit ensures that a valid Landlord is allowed to sign in, if the entered details are correct. It also ensures that an unauthenticated Landlord is not allowed to sign in.

Test Owner: Saksham Verma

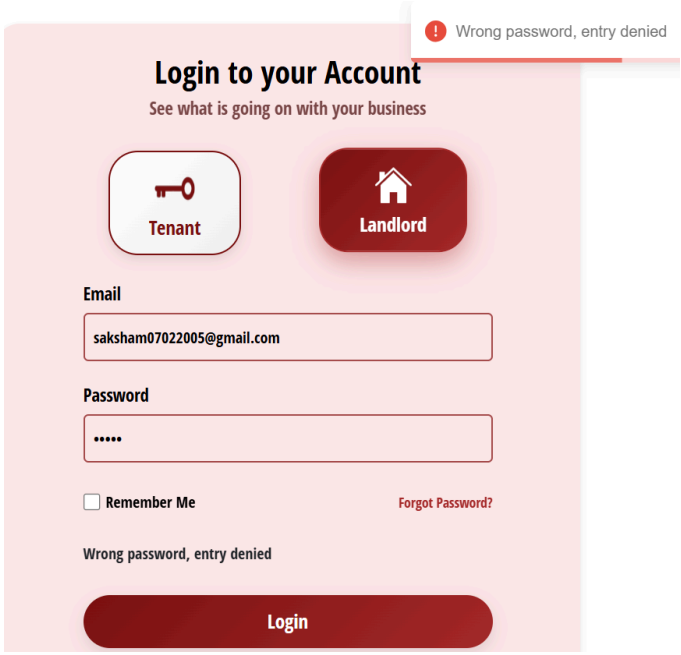
Test Date: 01/04/2025

Test Results: An authenticated Landlord(valid details) was allowed to log in successfully. An unauthenticated Landlord was prevented from logging in.

Upon entering correct credentials the Landlord is directed to his Homepage



Upon entering wrong email , a message showing "No such user exists" is displayed



Upon entering the wrong password, a message showing "Wrong Password" is displayed

Structural Coverage: Functional coverage (covers Landlord_Auth), Branch Coverage checks for Landlord Existence and valid credentials
Additional Comments: None

3a. Forgot Password-Backend

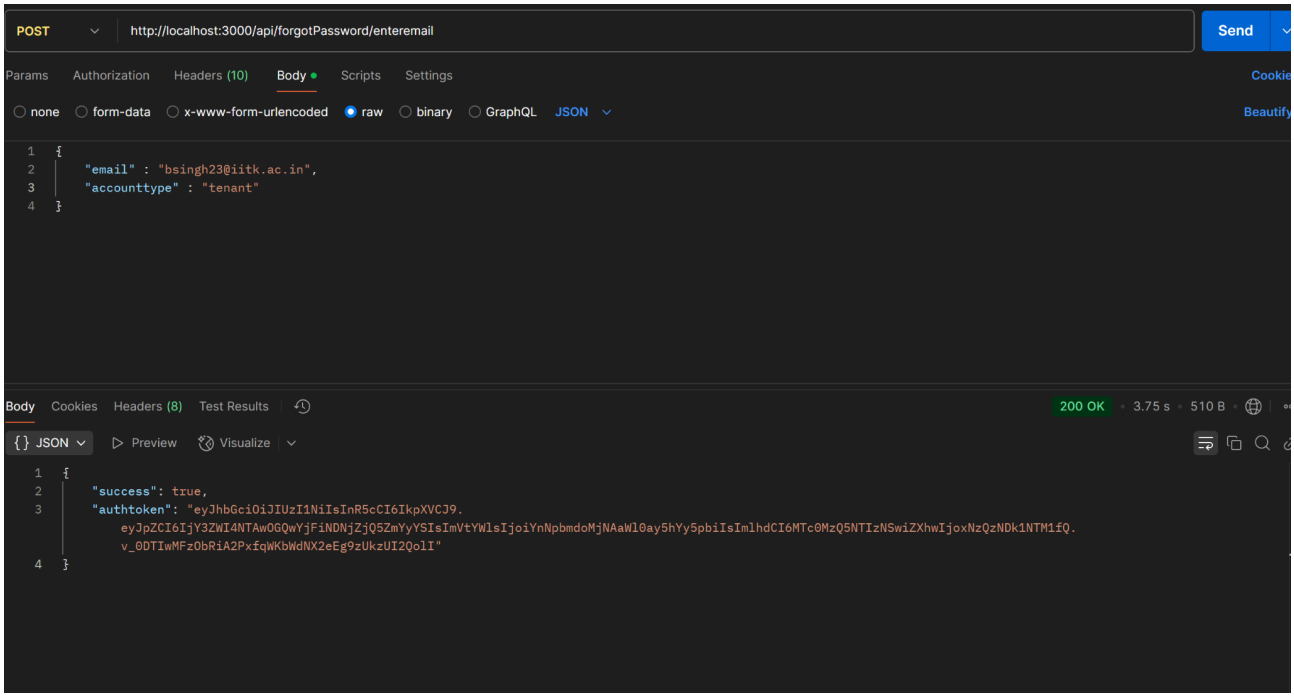
In this unit we have tested the backend route, which allows the user to get a new password in case he/she forgets it.

Unit Details: Testing of this unit ensures that if a user forgets his/her password, he/she can opt for making a new one using his/her registered email ID, and the new password gets updated in the backend.

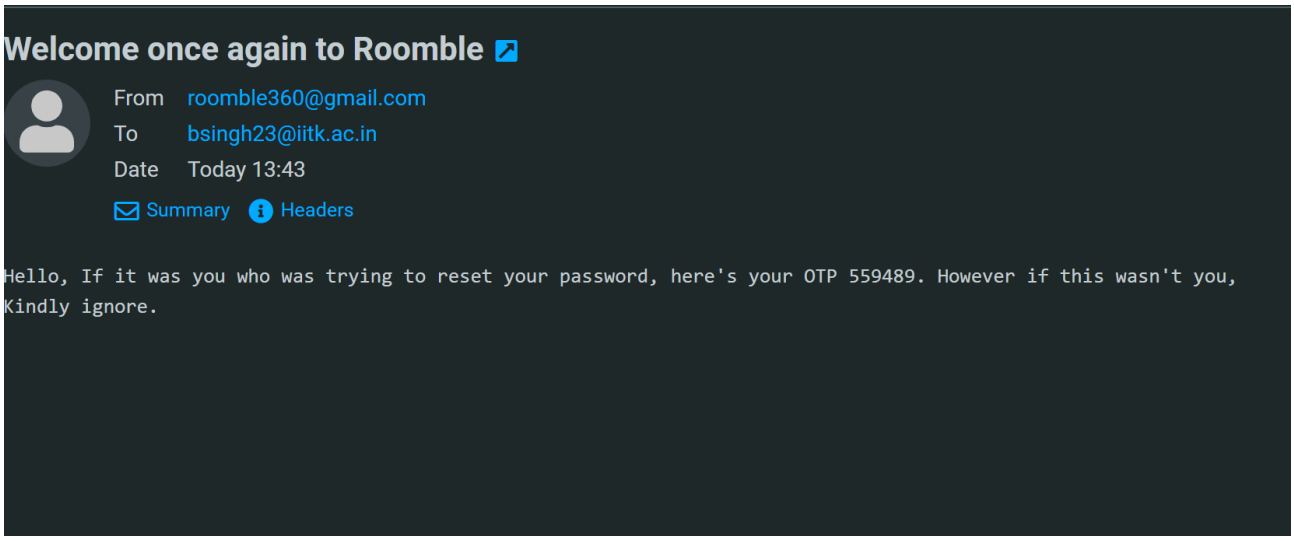
Test Owner: Bikramjeet Singh

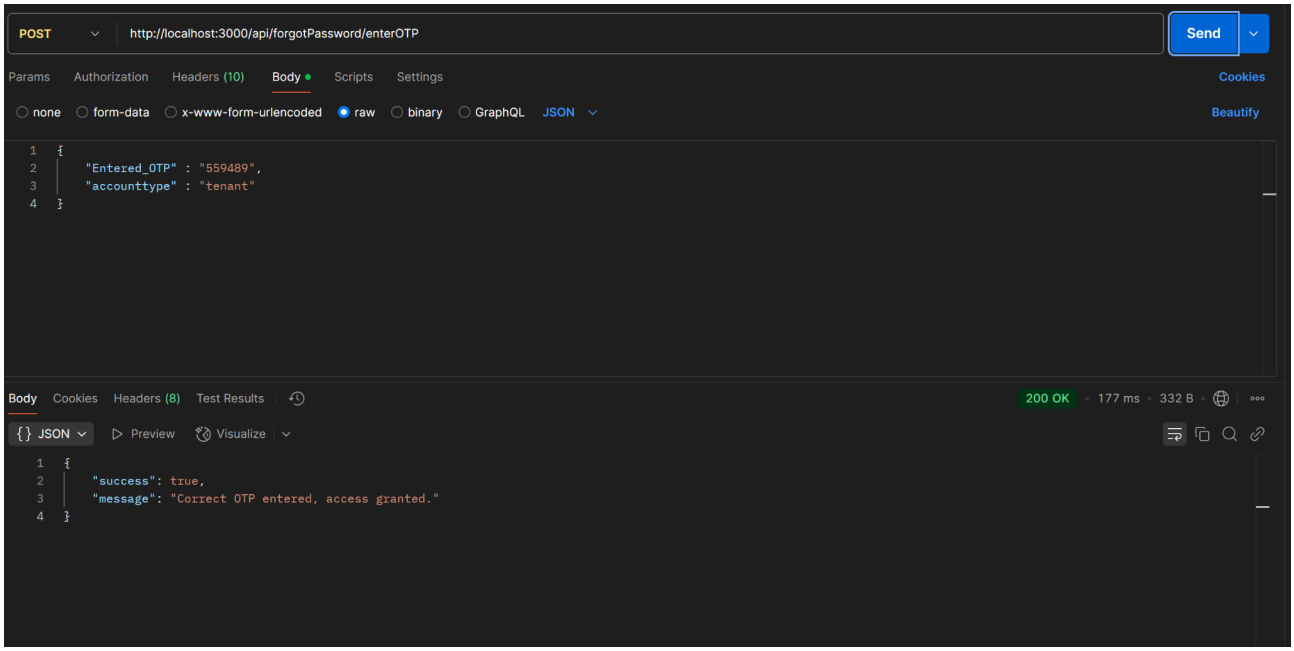
Test Date: 01/04/2025

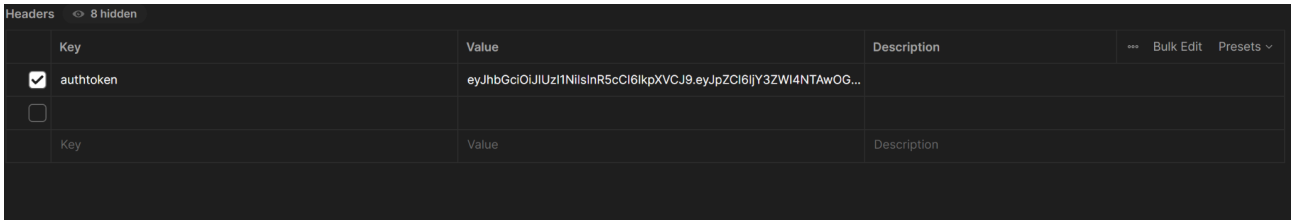
Test Results: Using this route the password was successfully changes in the backend.



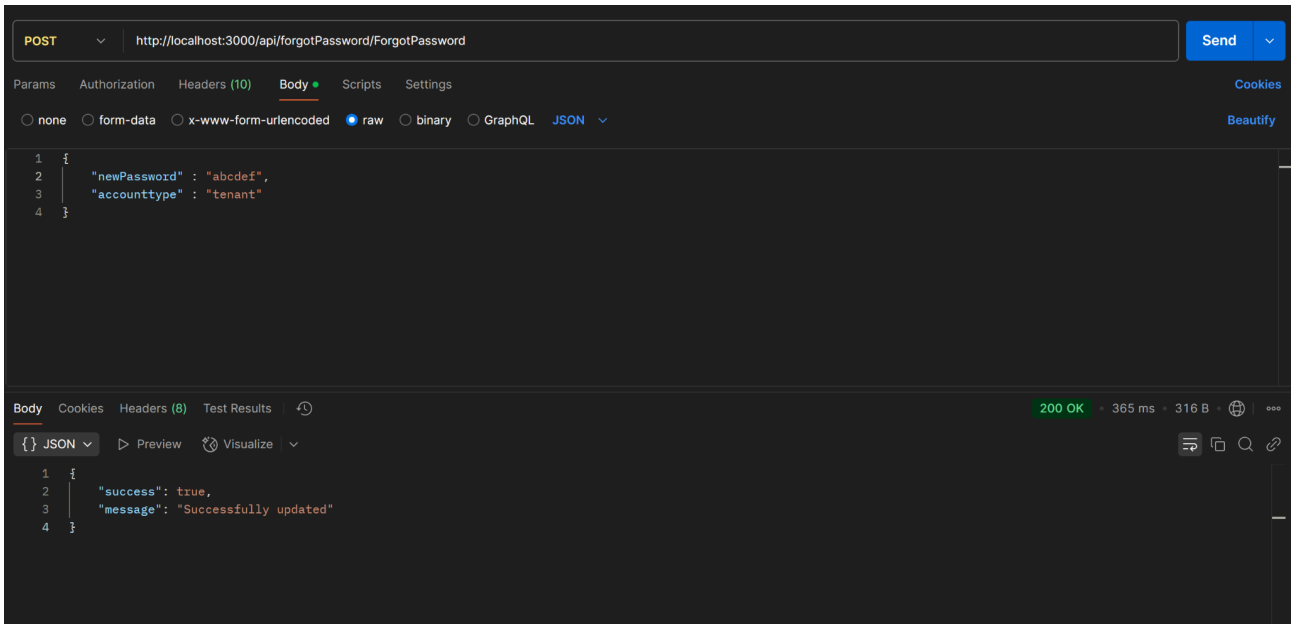
On sending the email and the accounttype in a POST request an OTP is sent to the email and an auth-token is generated for safety concerns.


Welcome once again to Roomble
From roomble360@gmail.com
To bsingh23@iitk.ac.in
Date Today 13:43
Summary
Headers
Hello, If it was you who was trying to reset your password, here's your OTP 559489. However if this wasn't you, Kindly ignore.


POST
http://localhost:3000/api/forgotPassword/enterOTP
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"Entered_OTP" : "559489",
"accounttype" : "tenant"
}
Body
Cookies
Headers (8)
Test Results
200 OK
177 ms
332 B
JSON
Preview
Visualize
{
"success": true,
"message": "Correct OTP entered, access granted."
}



On sending the correct OTP in the body alongwith the previously generated auth-token in the header the access to change the password is granted.



On sending the new-password in the body, the password gets successfully updated. This is also depicted in the backend as well:

password : `"$2b$10$3X1vjrc/dUOdWy2Ae1WKmOETVR.ANuh9Bqxckmwt1IyR37Wmsx/vO"`

This is the previously hashed password

password : `"$2b$10$0M.dz1aQMZsGG7.wVWAdoefO7JvWoOVB7OHuytHaptWg65W8v6166"`

This is the new password after changing it through Postman.

Structural Coverage: The above testing majorly addresses the functional coverage part, the branch coverage has been done in the frontend part immediately following this part.
Additional Comments: None

3b. Forgot Password-Frontend

In this unit, we have tested the feature that allows the user to get a new password he/she forgets through the user-friendly interface of our application.

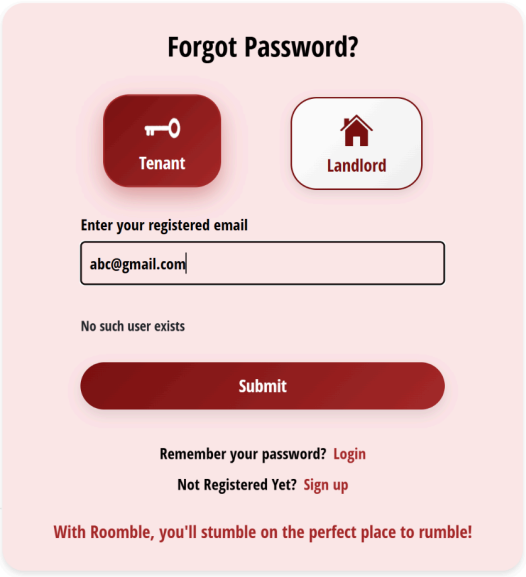
Unit Details: Testing of this unit ensures that if a user forgets his/her password,d he /she can easily make a new one using his/her registered email ID.

Test Owner: Bikramjeet Singh

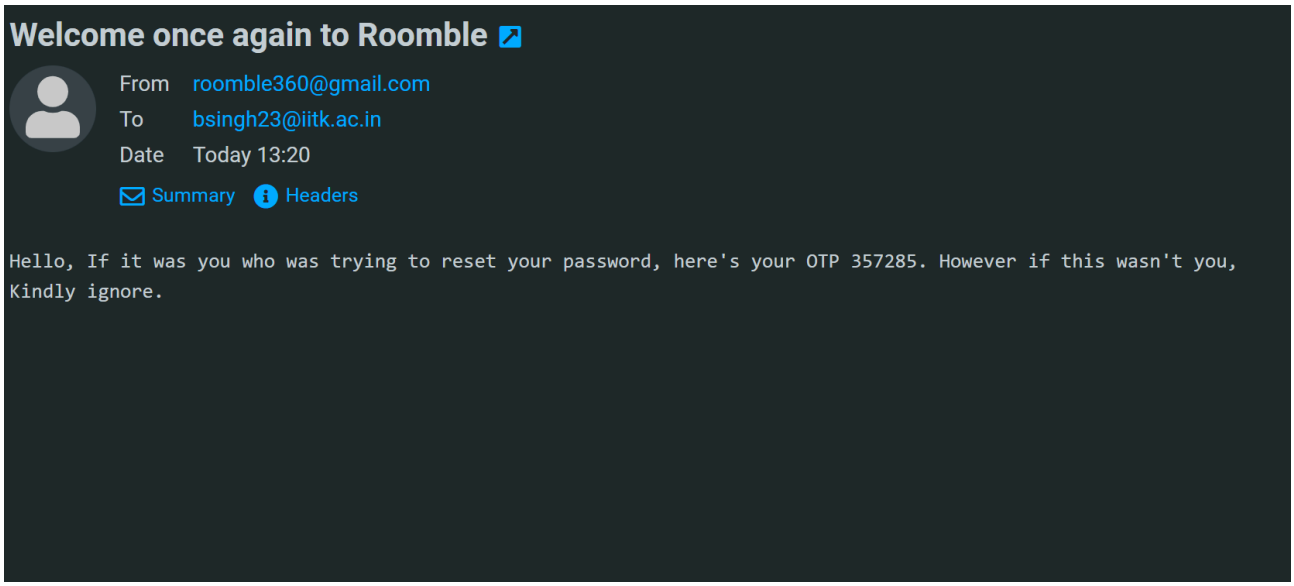
Test Date: 01/04/2025

Test Results: Using this feature a user was able to change his password.

On clicking Forgot password on the Login Page user is redirected here where he/she needs to enter his/her registered email.

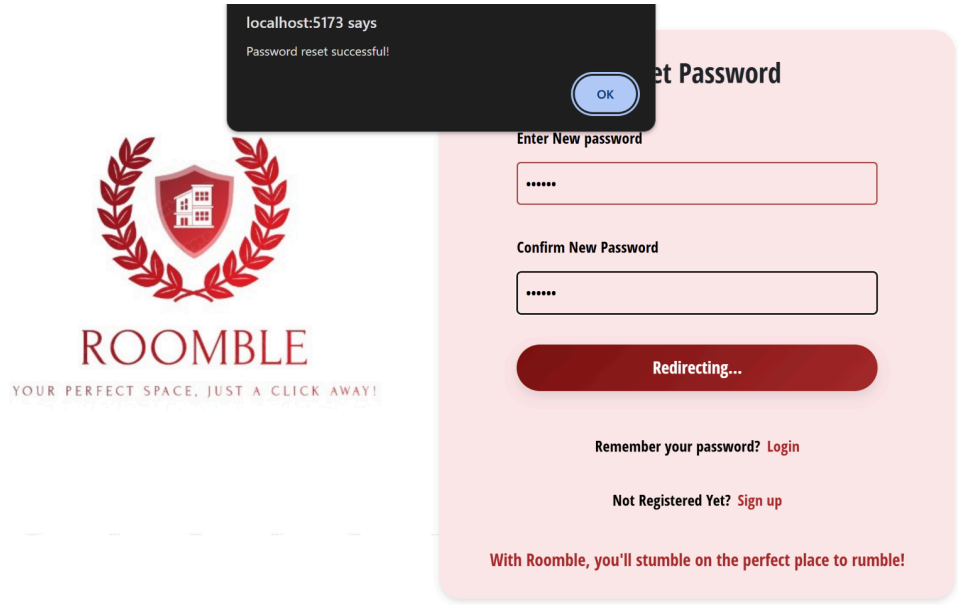


If the user is not registered he will get a No such user exists message



Entering the correct email sends an OTP to the concerned email.

Entering the correct OTP leads to the above page where user can enter and confirm the new password.



After doing so, an alert appears on the screen indicating success of the process. After this the user is redirected to the login page where he/she can use his/her new password.

Structural Coverage: Functional Coverage and Branch Coverage for invalid email and OTP.
Additional Comments: None

4a. View Self profile-Backend

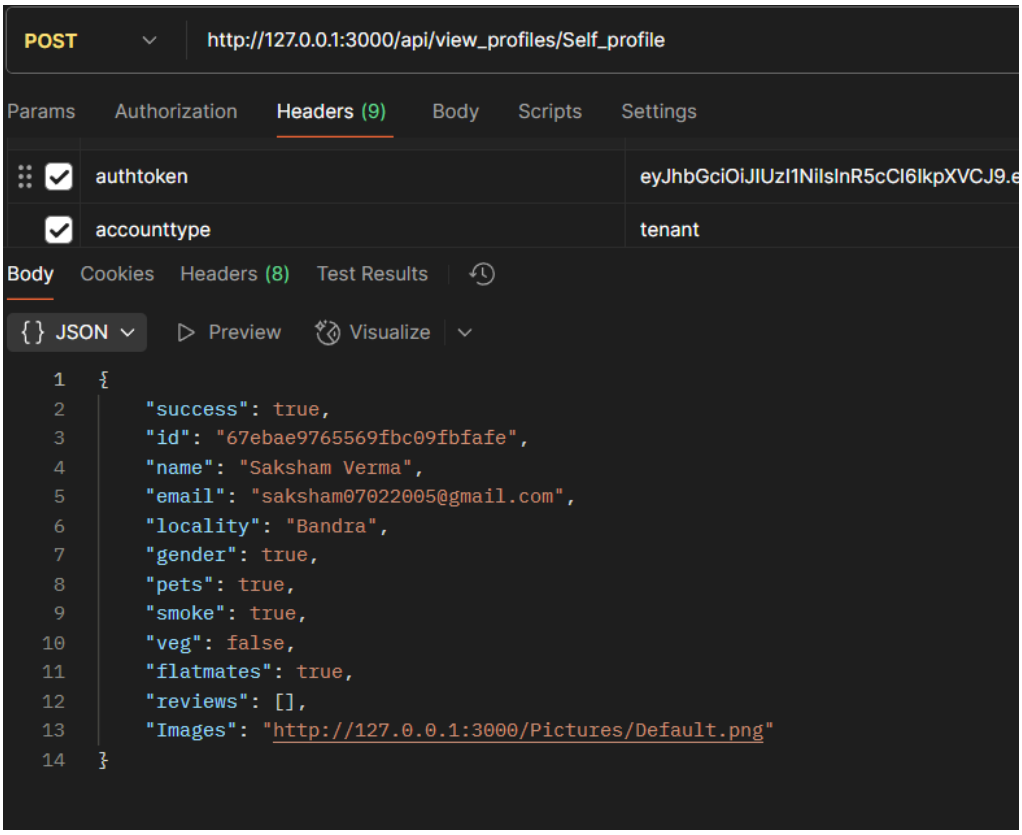
In this unit, we have tested the backend function that allows the user to view his own profile

Unit Details: Testing of this unit ensures that a users profile is fetched correctly to them

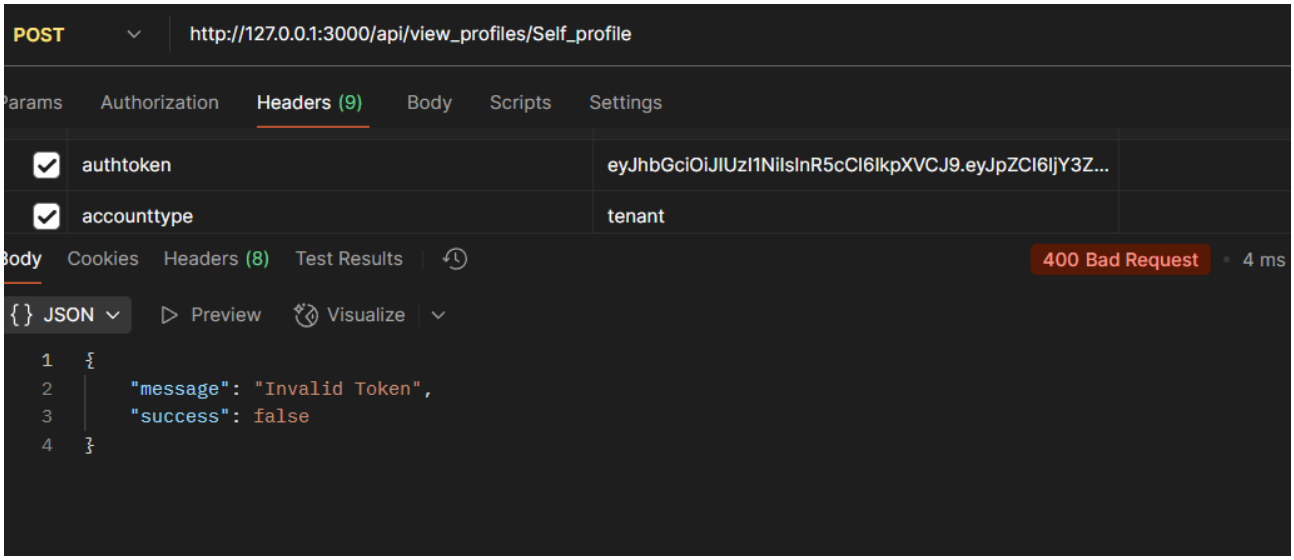
Test Owner: Rathod Ayushi

Test Date: 01/04/2025

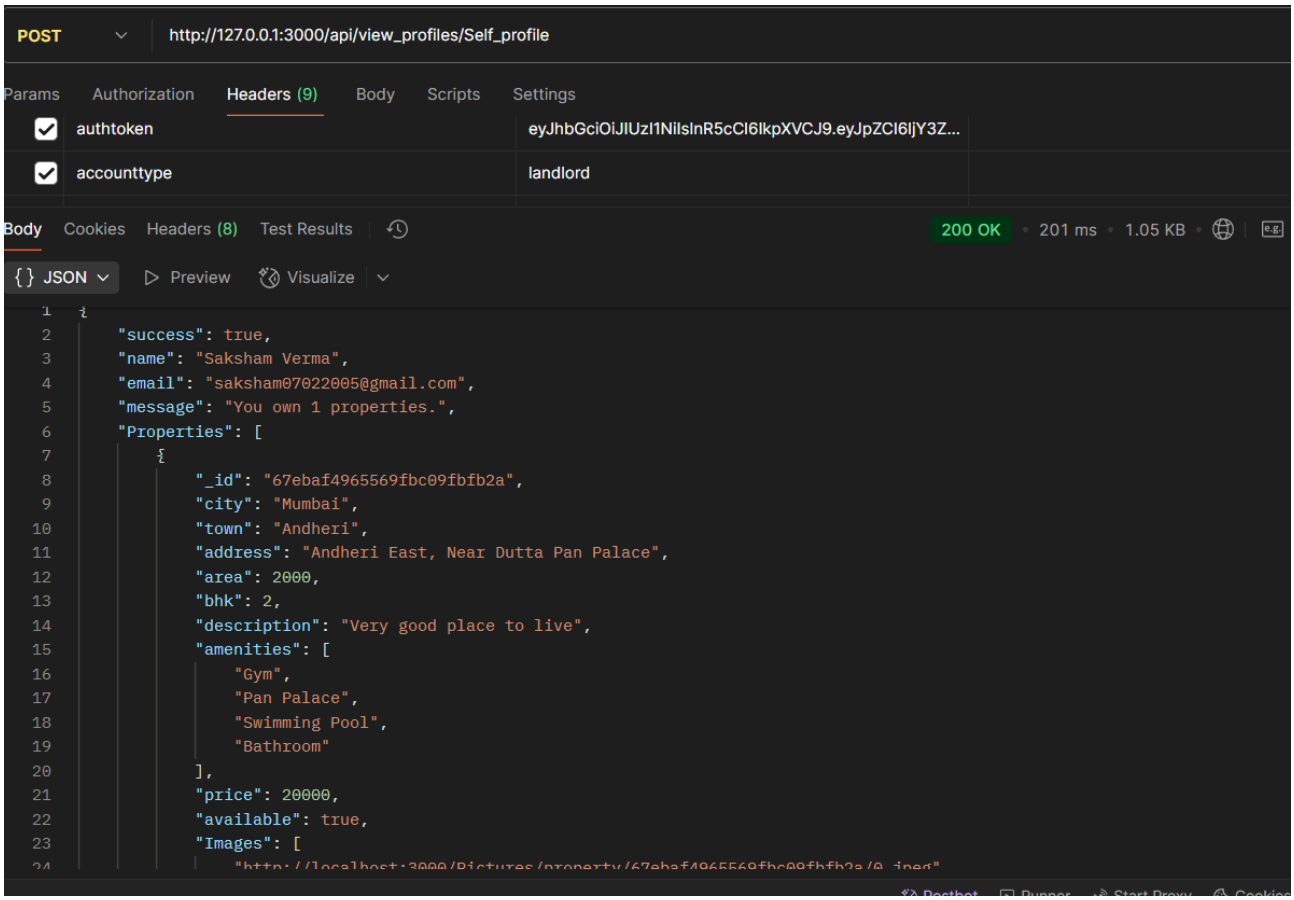
Test Results: Using this feature a user was able to correctly fetch his own profile.



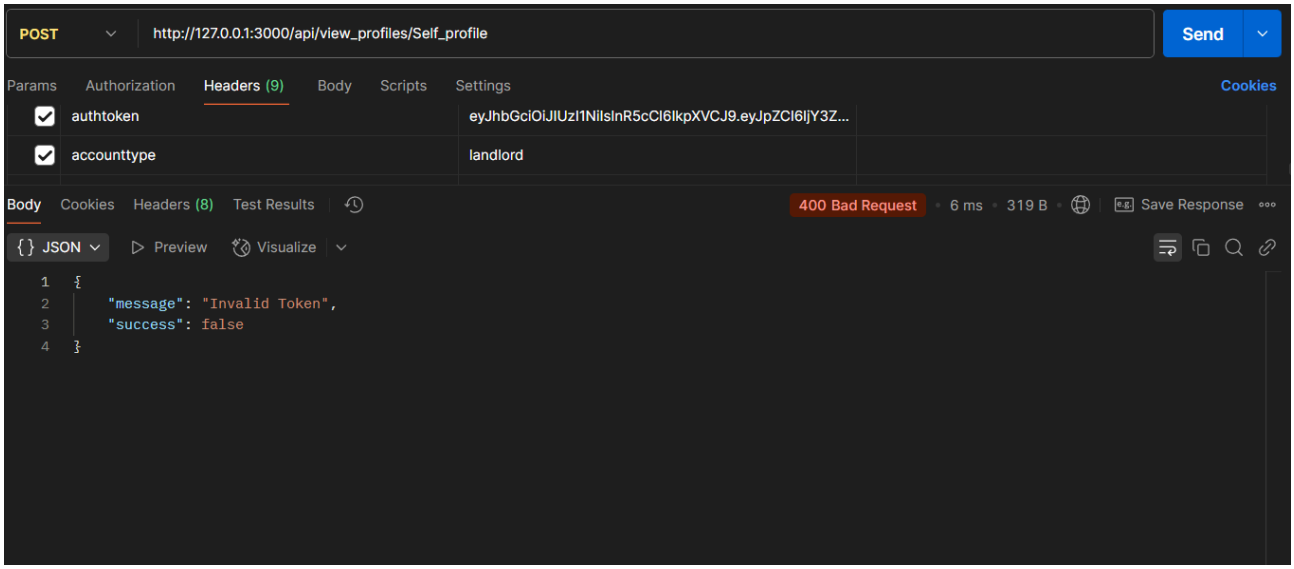
If a tenant with a valid authtoken sends this request, he gets his own profile



If a tenant who hasn't logged in or doesn't have a valid authtoken tries to access his profile, he gets a response as above



If a landlord with a valid authtoken sends this request, he gets his own profile



If a landlord who hasn't logged in or doesn't have a valid authtoken tries to access his profile, he gets a response as above

Structural Coverage: Functional Coverage and Branch Coverage on whether a user has logged in.

Additional Comments: None

4b. View Self profile-Frontend

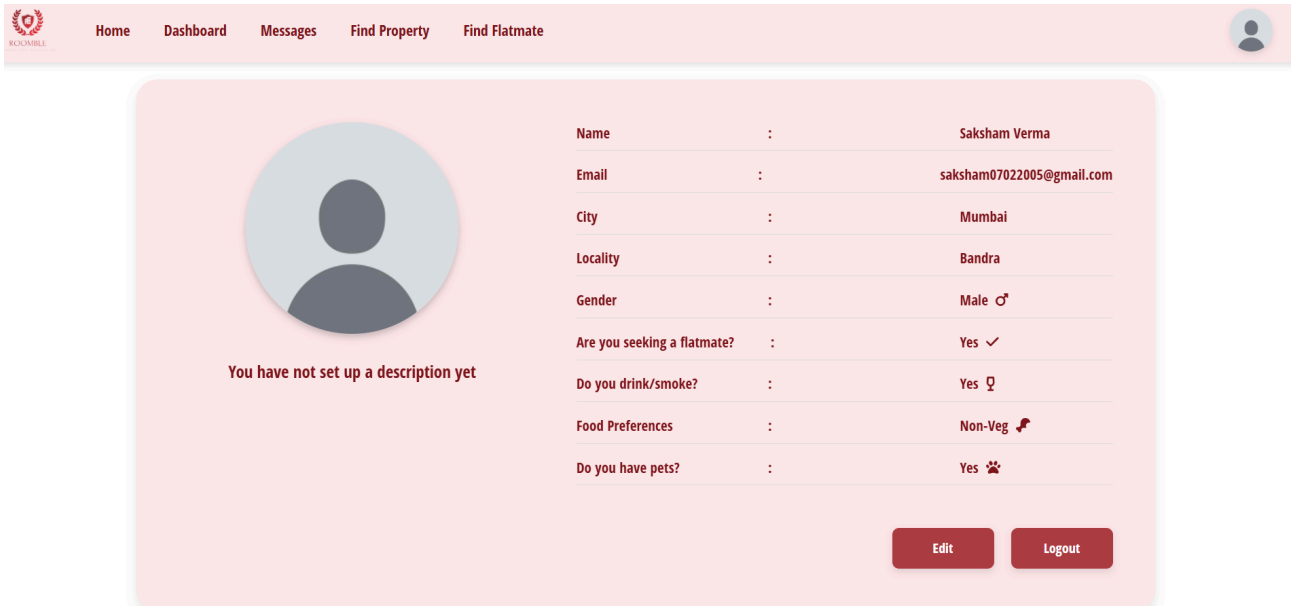
In this unit, we have tested the frontend function that allows the user to view his own profile

Unit Details: Testing of this unit ensures that a users profile is shown correctly to them

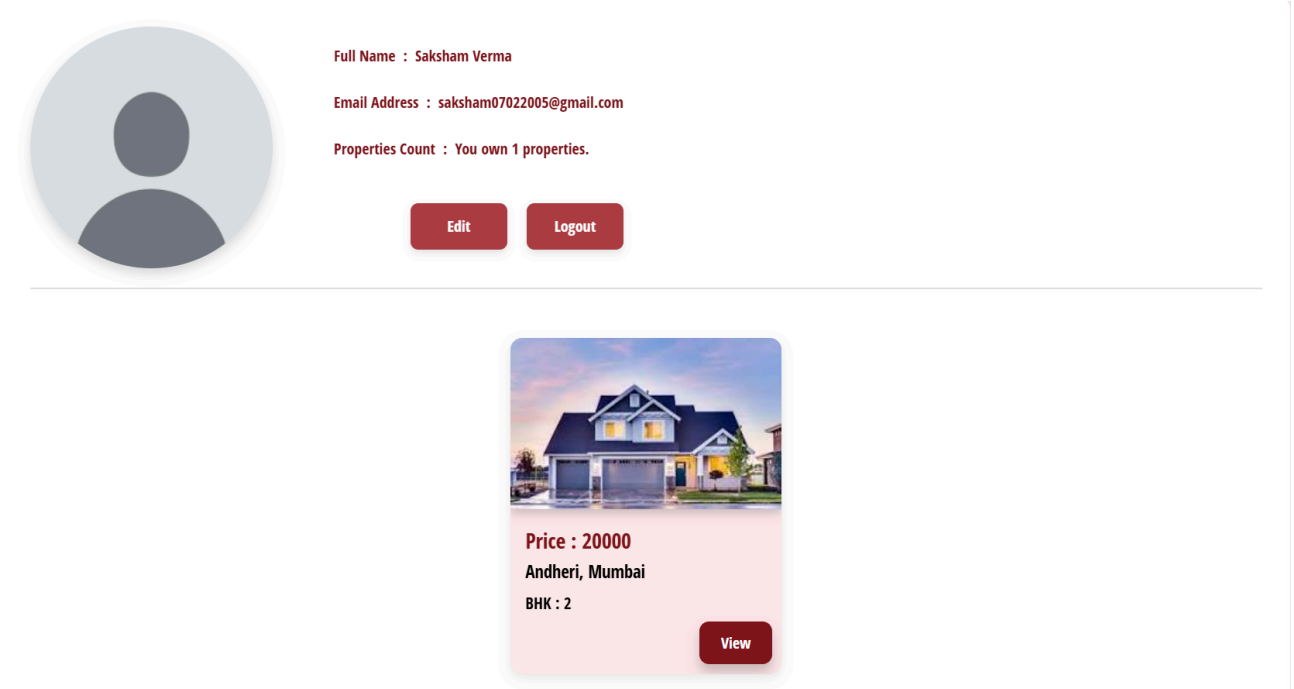
Test Owner: Saksham Verma

Test Date: 01/04/2025

Test Results: Using this feature a user was able to correctly view his own profile.



If a tenant is logged in he can correctly view his profile



If a tenant is logged in, he can correctly view his profile

This is the output shown to a user who isn't logged in, but tries to view his profile

Structural Coverage: Functional Coverage and Branch Coverage on whether a user has logged in.

Additional Comments: None

5a. View Others profile-Backend

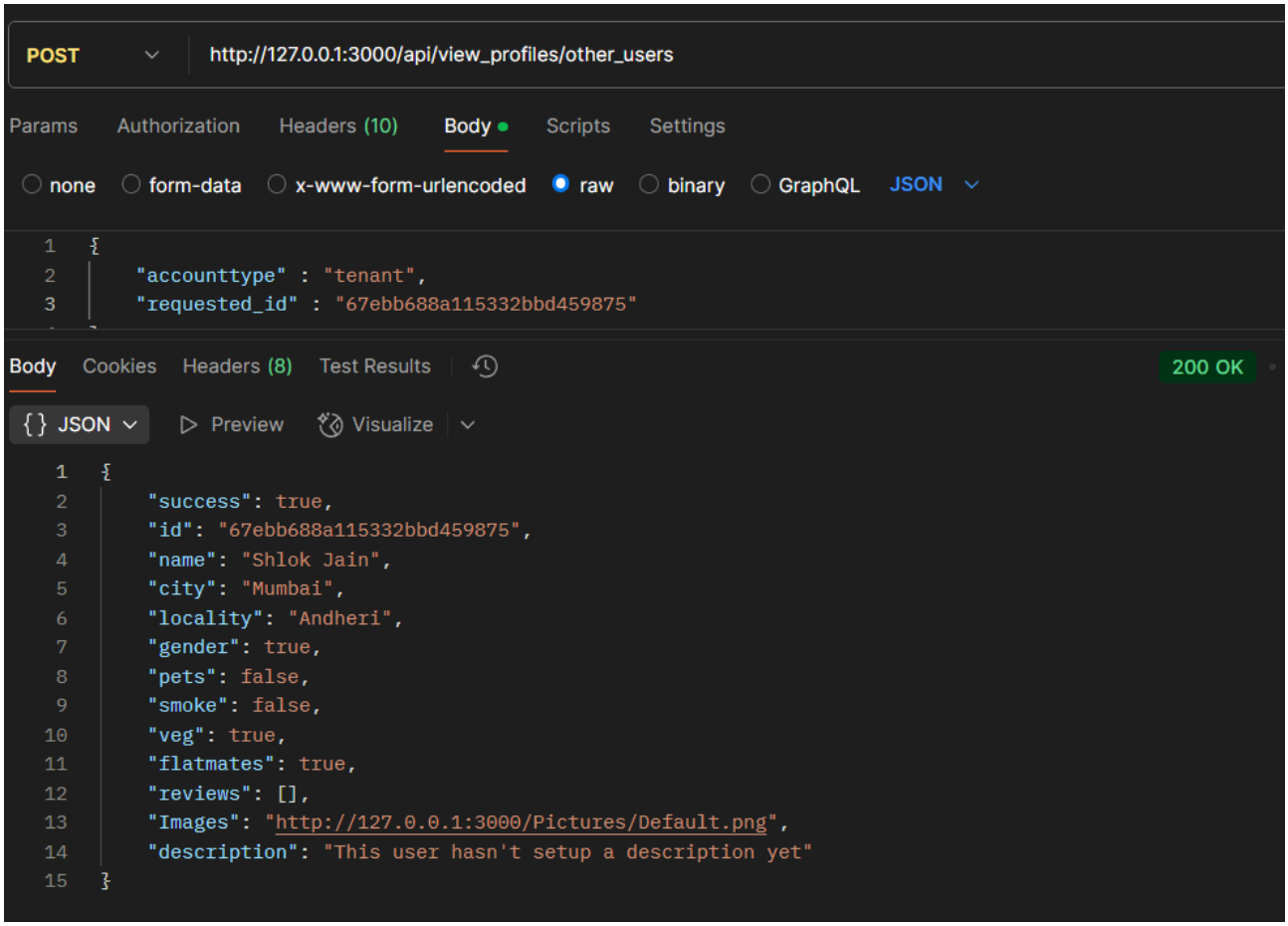
In this unit, we have tested the frontend function that allows the user to view his own profile

Unit Details: Testing of this unit ensures that a user's profile is shown correctly to other users

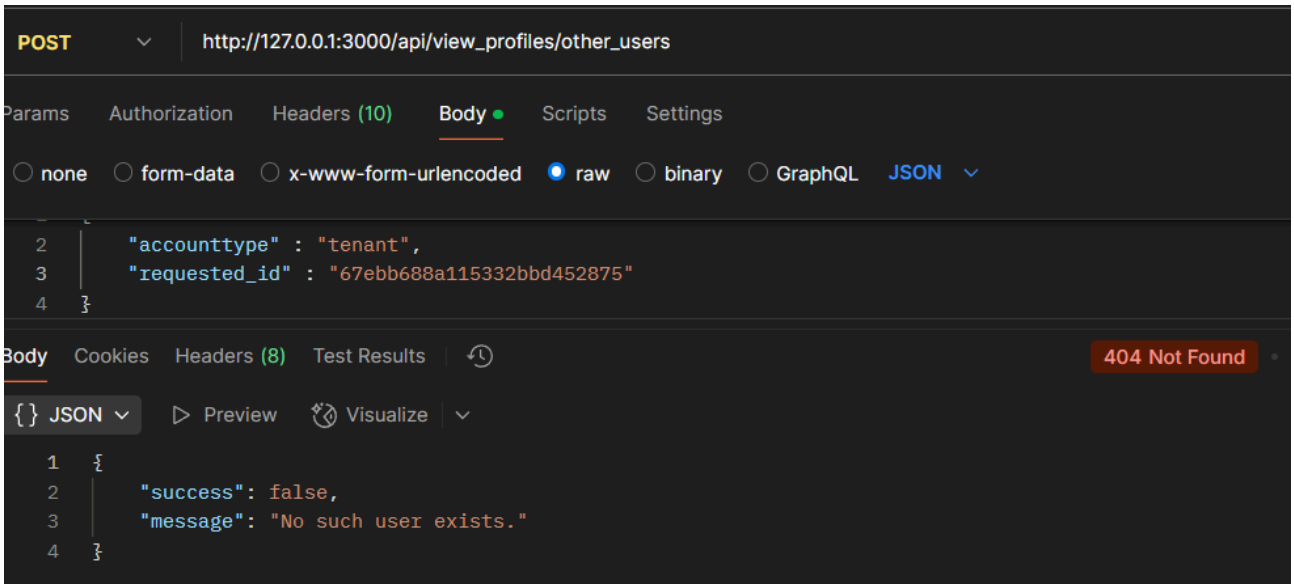
Test Owner: Saksham Verma

Test Date: 01/04/2025

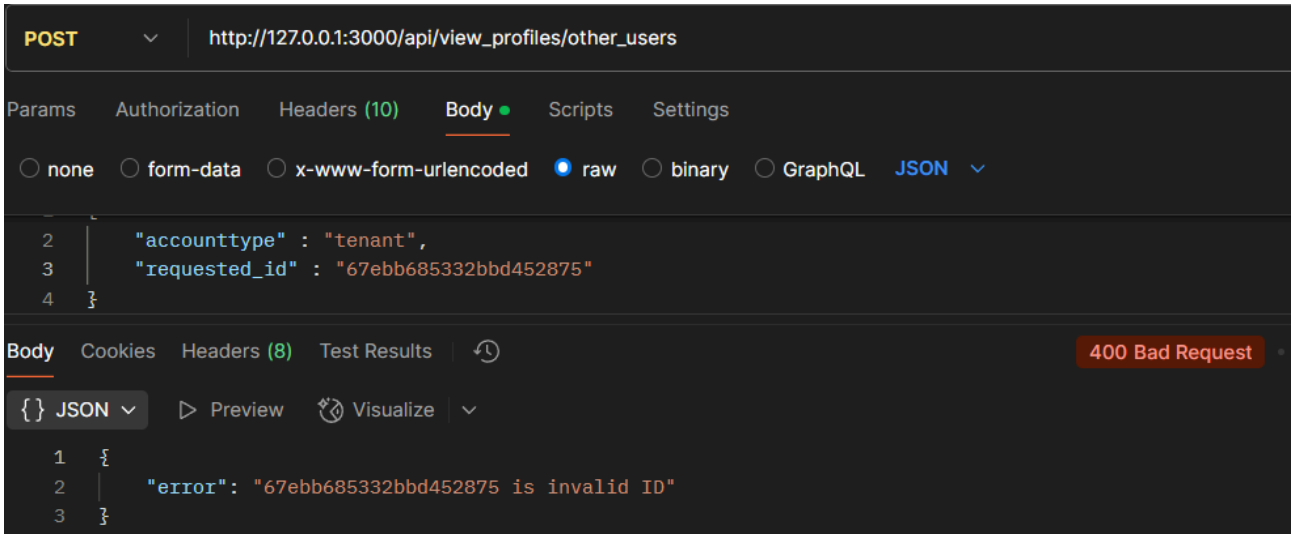
Test Results: Using this feature a user was able to correctly view other profiles.



If the correct ID of a tenant / landlord is entered, their profile is retrieved



If a wrong ID of tenant/landlord is entered, a message displaying “No such user exists”



If an invalid ID is entered(MongoDB follows certain ID syntaxes) a status code of 400 is shown.

Structural Coverage: Functional Coverage and Branch Coverage on wrong ID being asked, ID not existing in backend and a valid ID entered.

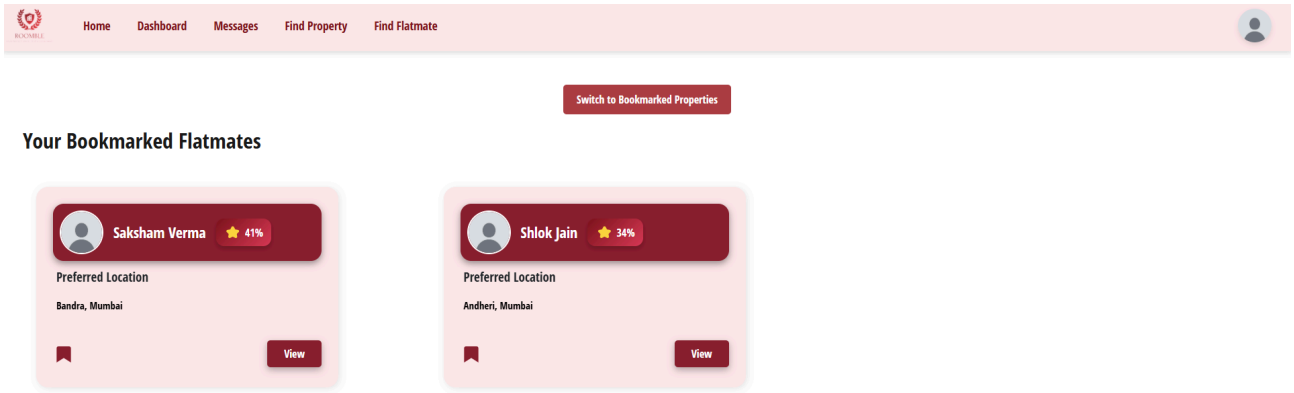
Additional Comments: None

6. Navbar

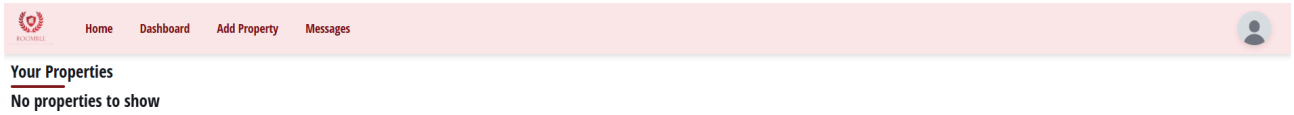
In this unit, we have tested the frontend function that allows the user to navigate to different pages through Navbar.

Unit Details: Testing of this unit ensures that a user can move through different pages using Navbar.

Test Owner: Ronav Puri
Test Date: 01/04/2025



The Navbar for Tenant has the following options.



The Navbar for Landlord has the following options.

Test Results: Clicking on the Navbar options successfully redirects the user to different pages, and the NavBar options change for different types of users.

Structural Coverage: Functional coverage and Branch coverage on fetching relevant details to be displayed on the navbar based on the usertype and page elements.

Additional Comments: None

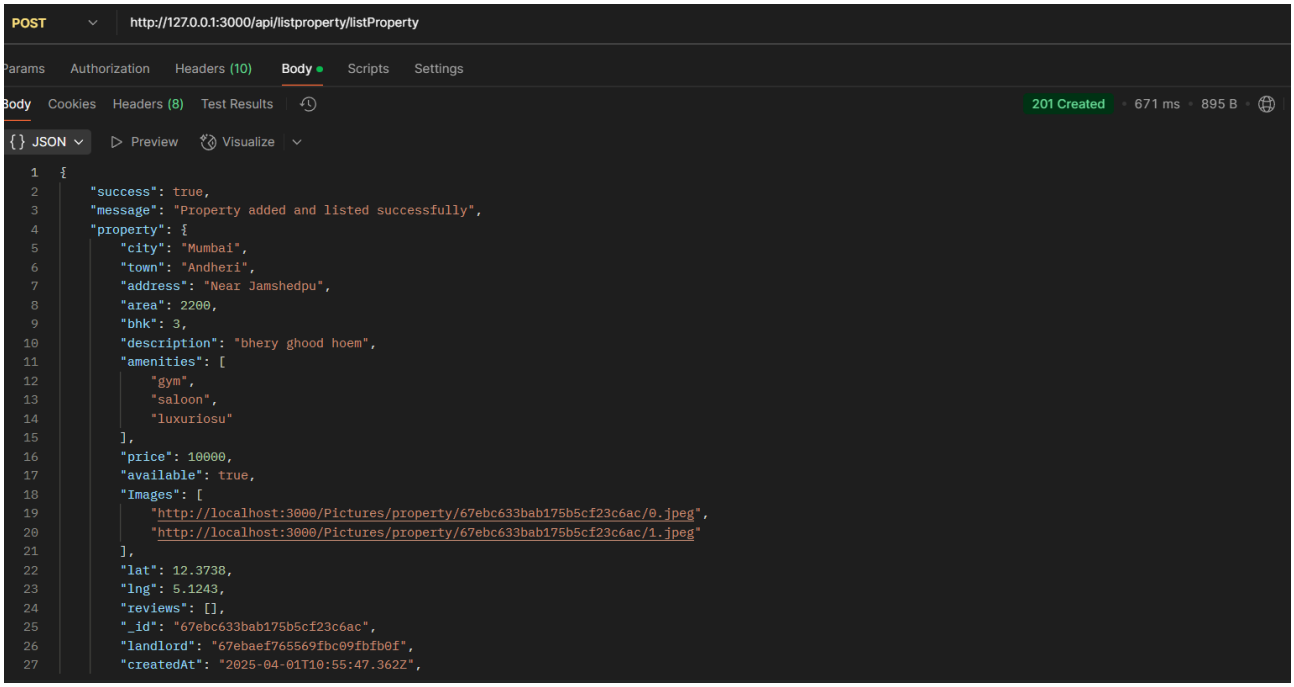
7. Add-Property-Backend

In this unit, we have tested the backend function which enables a landlord to add properties for renting.

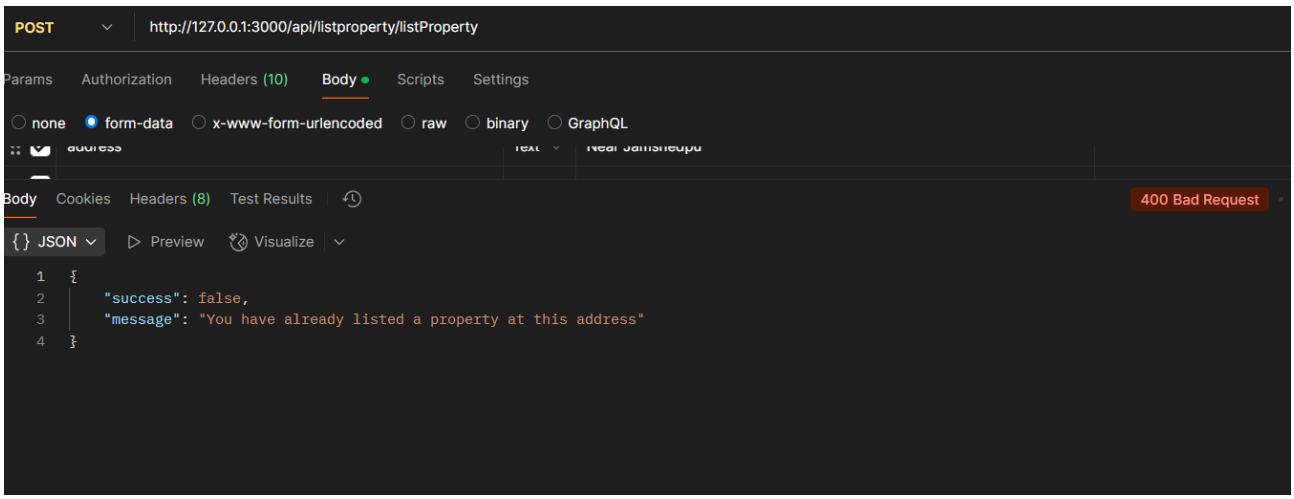
Unit Details: Testing of this unit ensures that a landlord can successfully list a property.

Test Owner: Saksham Verma
Test Date: 01/04/2025

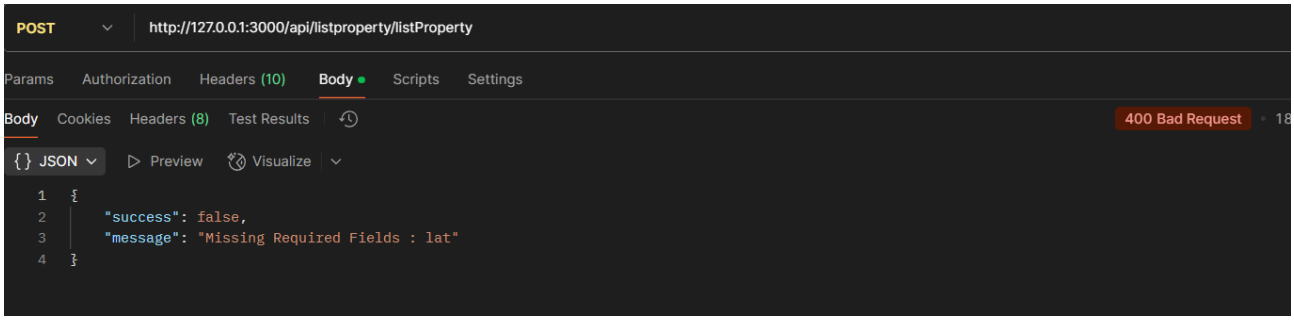
Test Results: Using this test, the software ensures that a landlord is able to add properties and also throw an error whenever something unexpected is given to it.



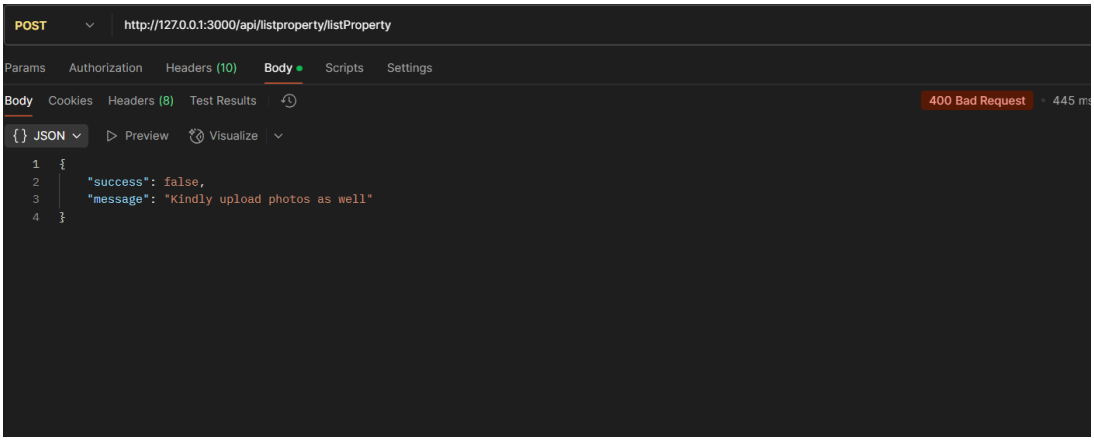
A user can successfully list properties by entering all given fields.



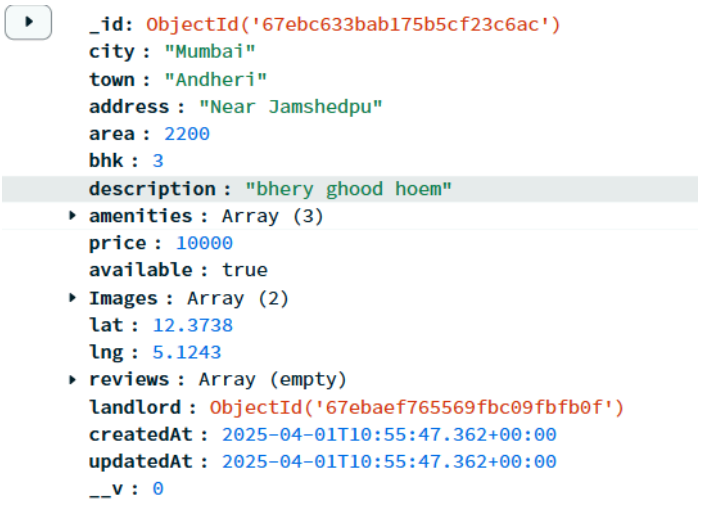
If a landlord tries to add 2 properties at the same address, then the software throws an error



If a landlord misses to fill any fields then a message is displayed indicating the missing fields



If a landlord uploads a property without Images, a message is displayed, prompting the landlord to upload Images



Upon successful property addition a new document was created in our database

Structural Coverage: Functional Coverage and Branch Coverage on whether all details are filled correctly while adding a property, Images Uploaded, Image size and number of Images uploaded and whether property already exists.
Additional Comments: None

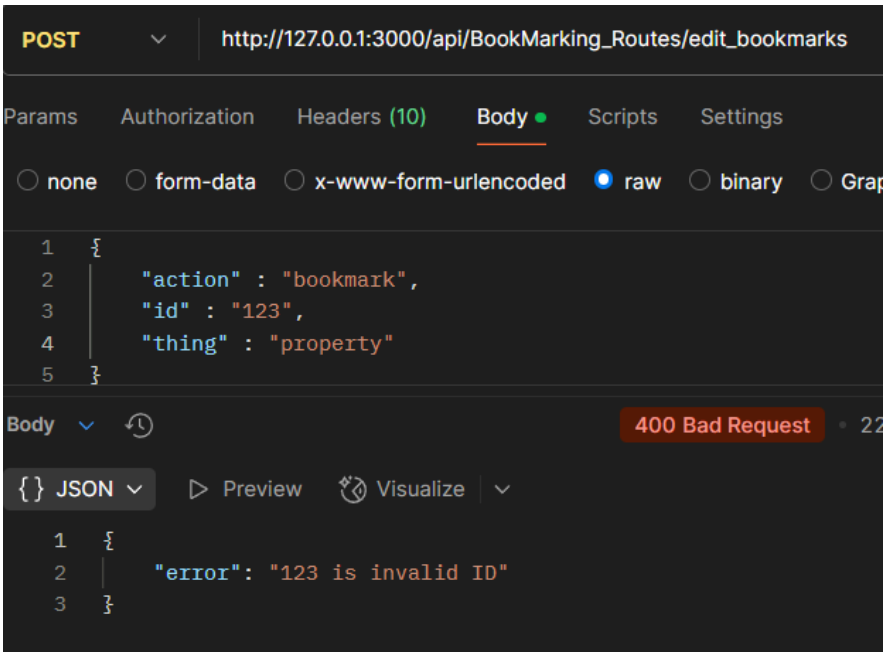
8. Bookmarks-Backend

In this unit, we have tested the backend function which enables a Tenant to bookmark and unmark other flatmates and Properties.

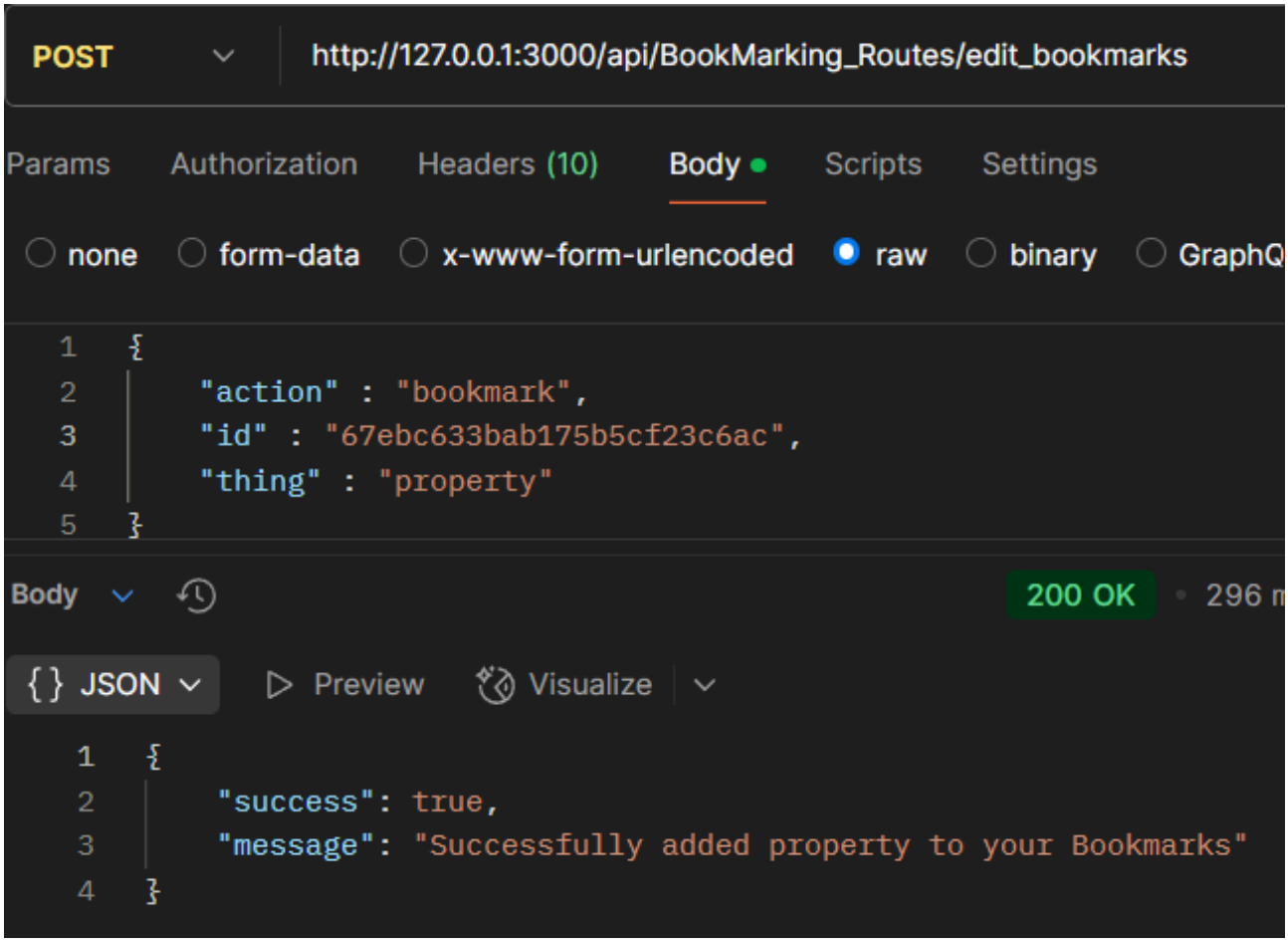
Unit Details: Testing of this unit ensures that a tenant can successfully bookmark and unmark a property and flatmate

Test Owner: Saksham Verma
Test Date: 01/04/2025

Test Results: Using this test, the software ensures that a tenant will be able to bookmark properties and other flatmates and also throw an error whenever a wrong ID is tried sent to backend for processing.

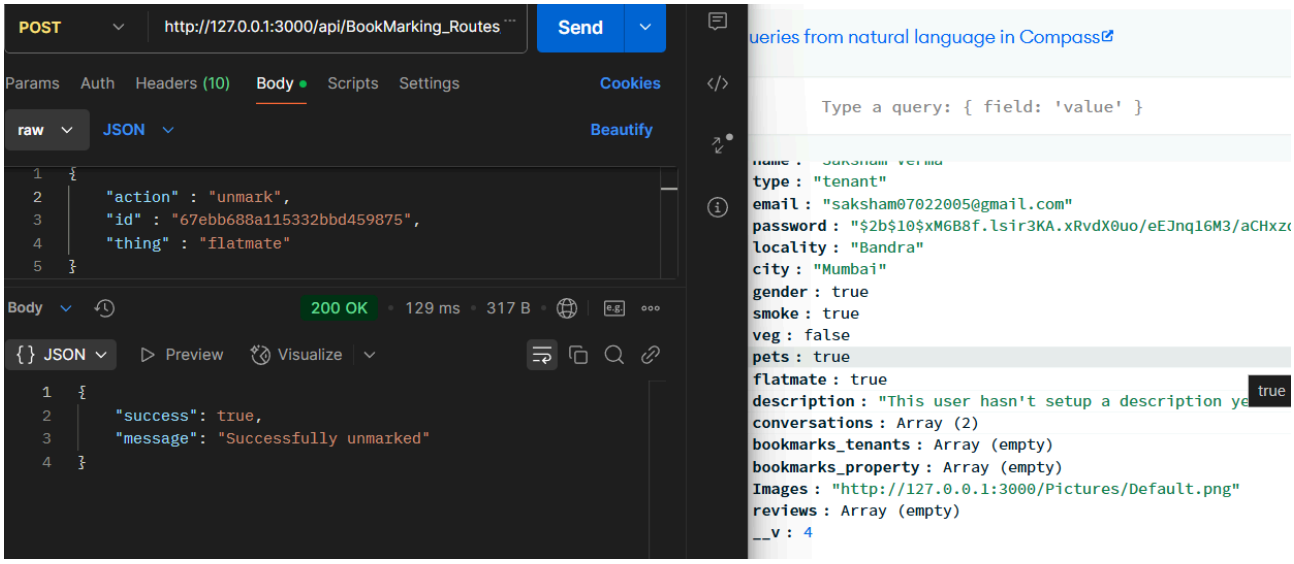


An error is thrown whenever a wrong ID is entered

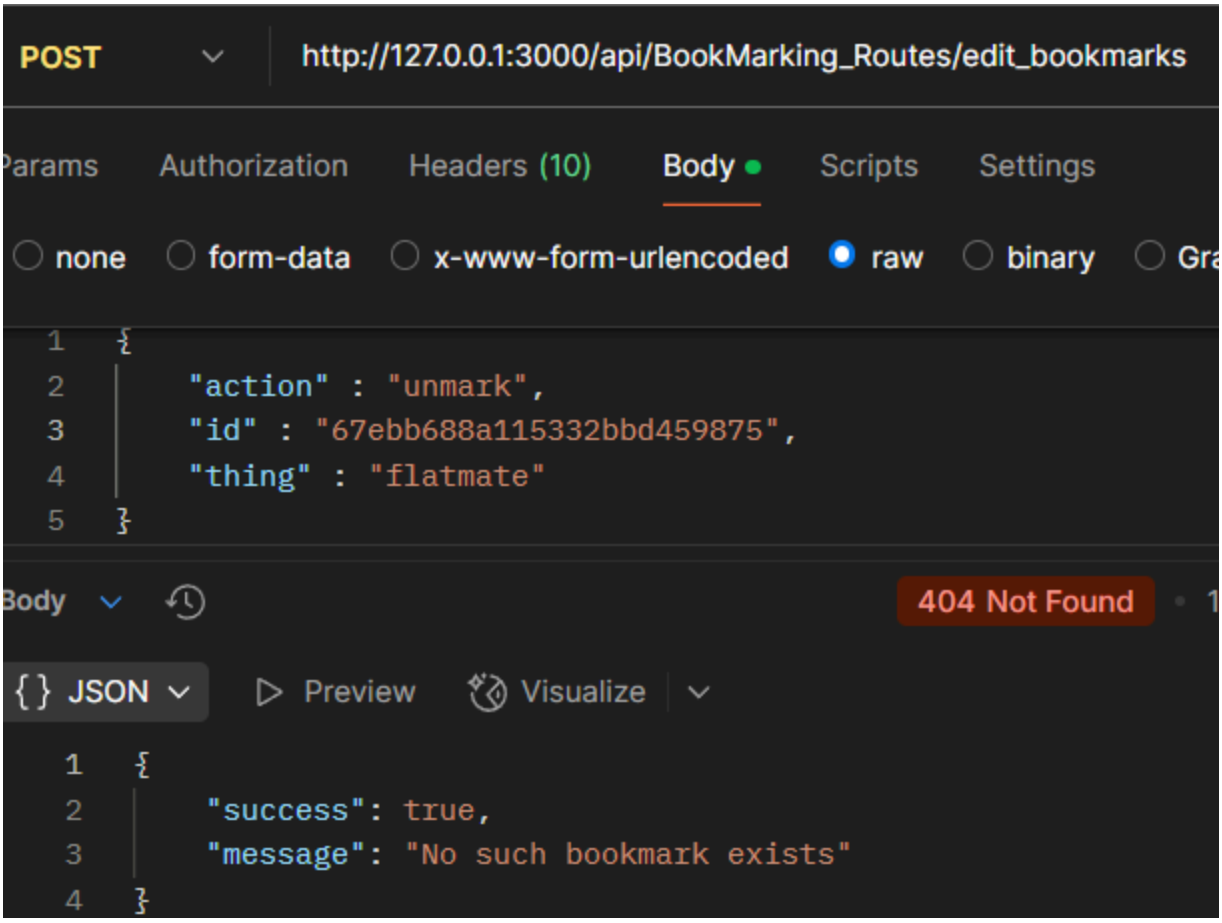


The user can successfully add bookmarks to the property by entering correct property id

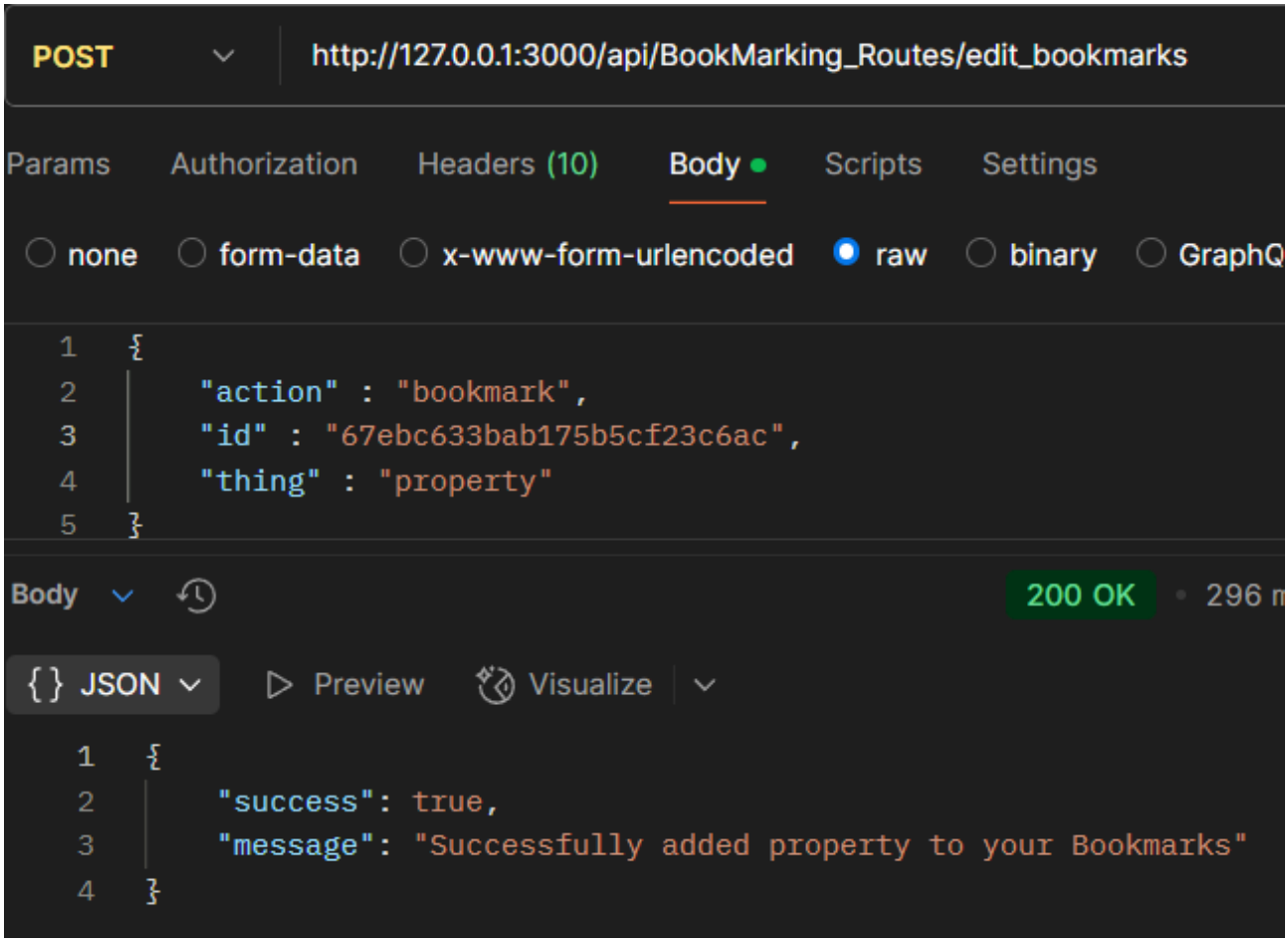
Upon Successful addition the Bookmark is added to our database and also a successful response is given to the client

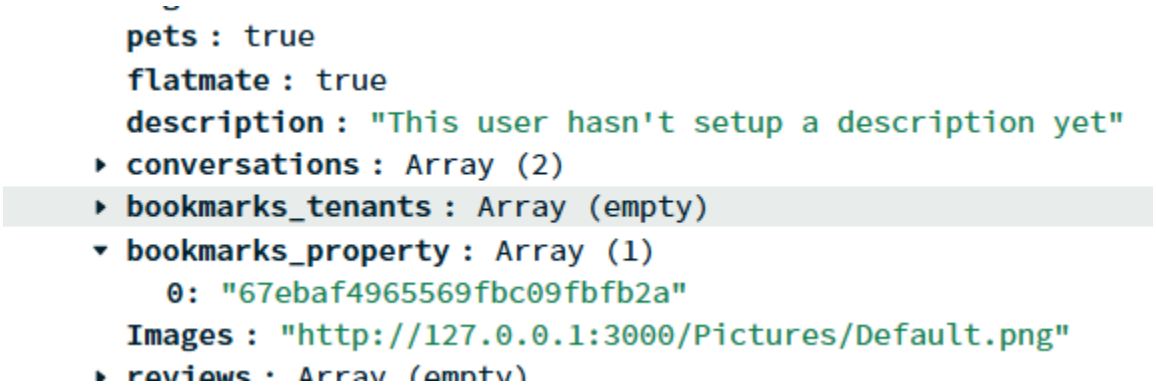


Upon successful Unmarking of a flatmate a success code of 200 is returned and also removed from our database



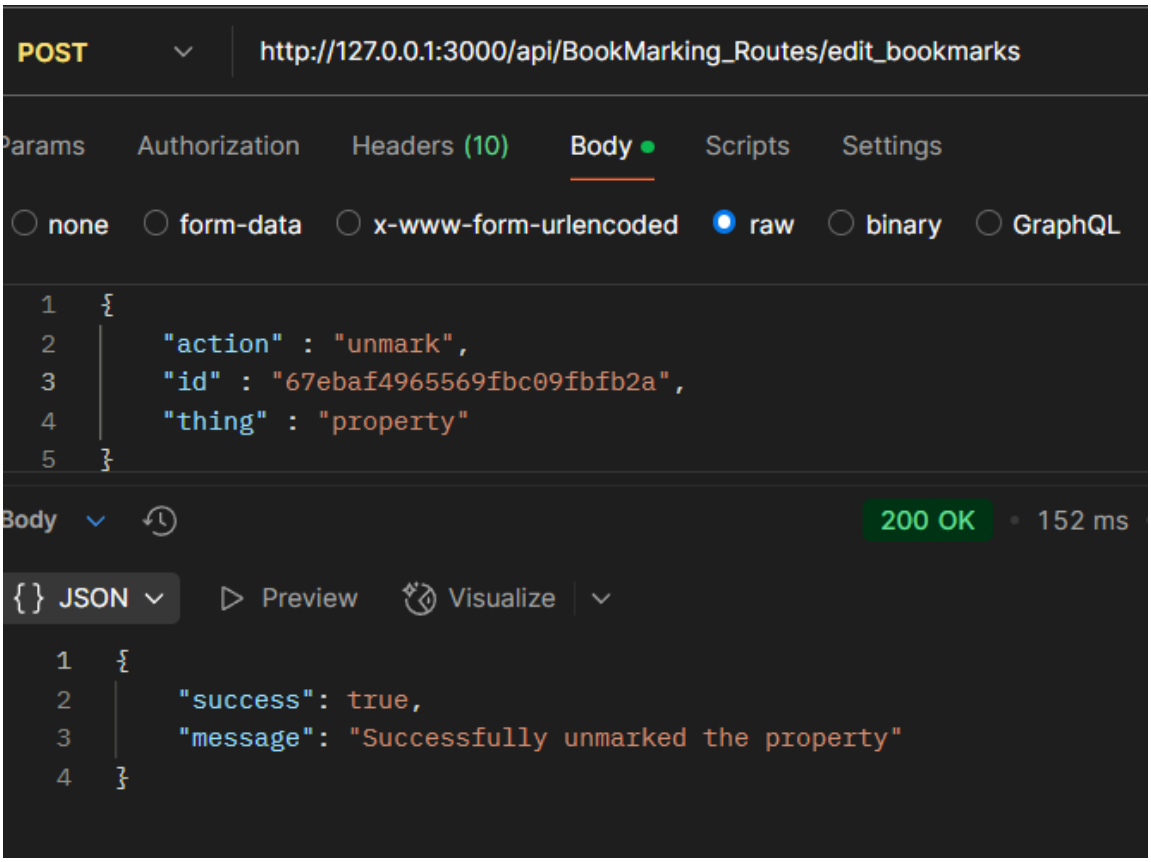
If someone tries to unmark a flatmate who isn't bookmarked in the first place, status code of 404 is returned along with a message which says the same



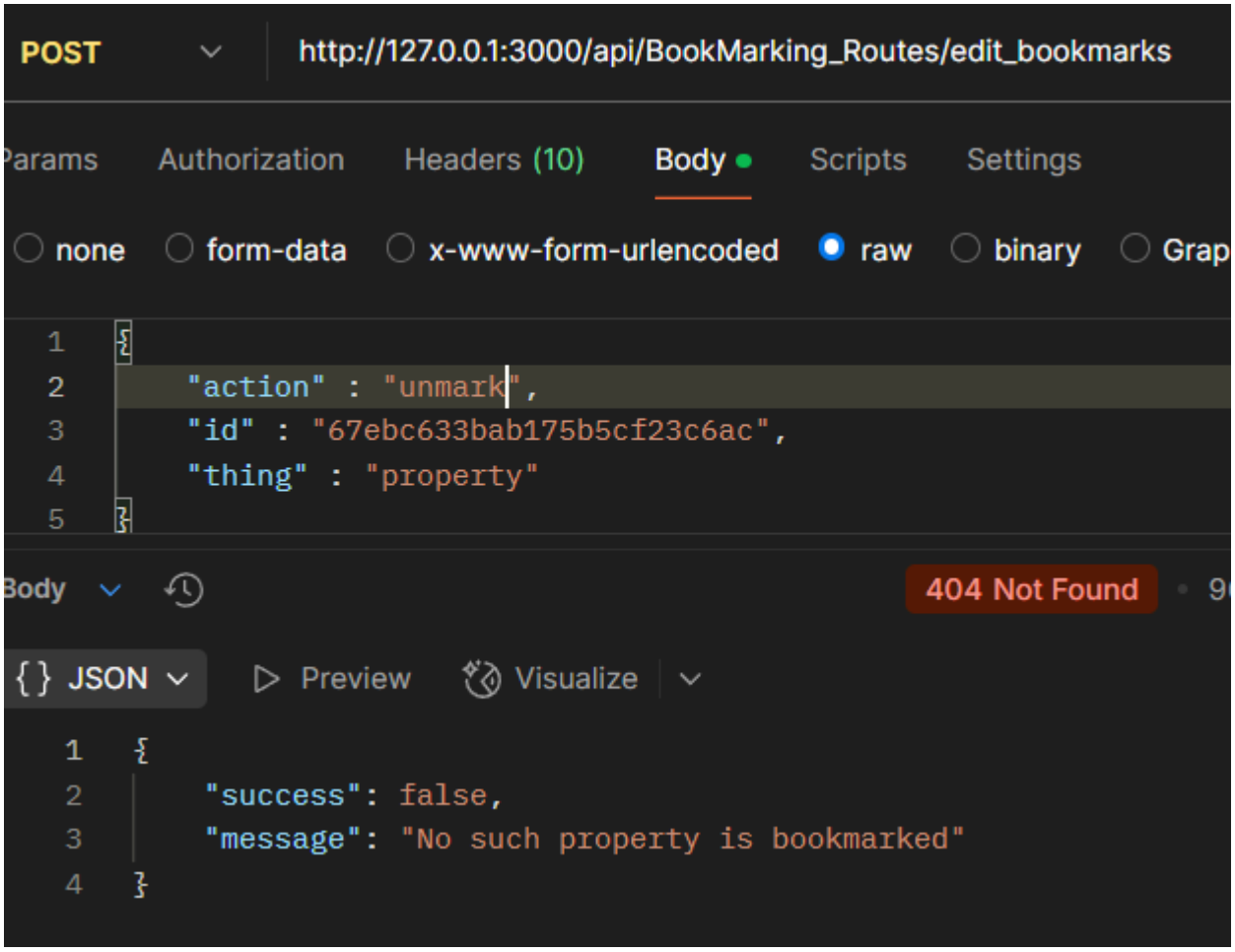


Successfully bookmarking a property

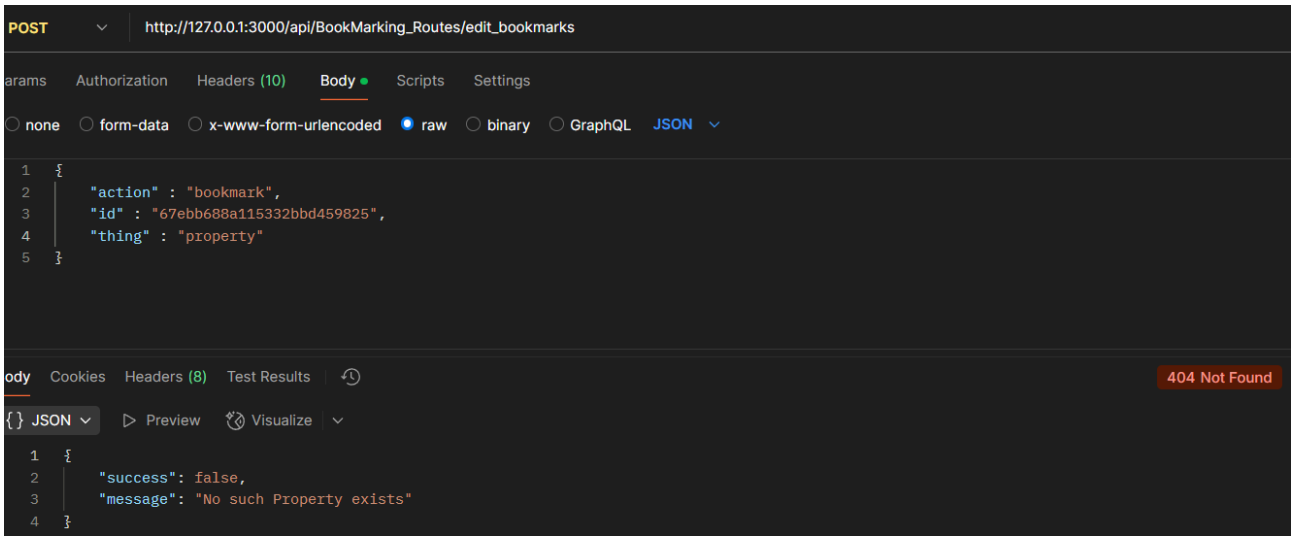
```
_id: ObjectId('67ebae9765569fbc09fbfafe')
name : "Saksham Verma"
type : "tenant"
email : "saksham07022005@gmail.com"
password : "$2b$10$xM6B8f.lsir3KA.xRvdX0uo/eEJnq16M3/aCHxzq1/l1bWUyN8K72"
locality : "Bandra"
city : "Mumbai"
gender : true
smoke : true
veg : false
pets : true
flatmate : true
description : "This user hasn't setup a description yet"
conversations : Array (2)
bookmarks_tenants : Array (empty)
bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
reviews : Array (empty)
```

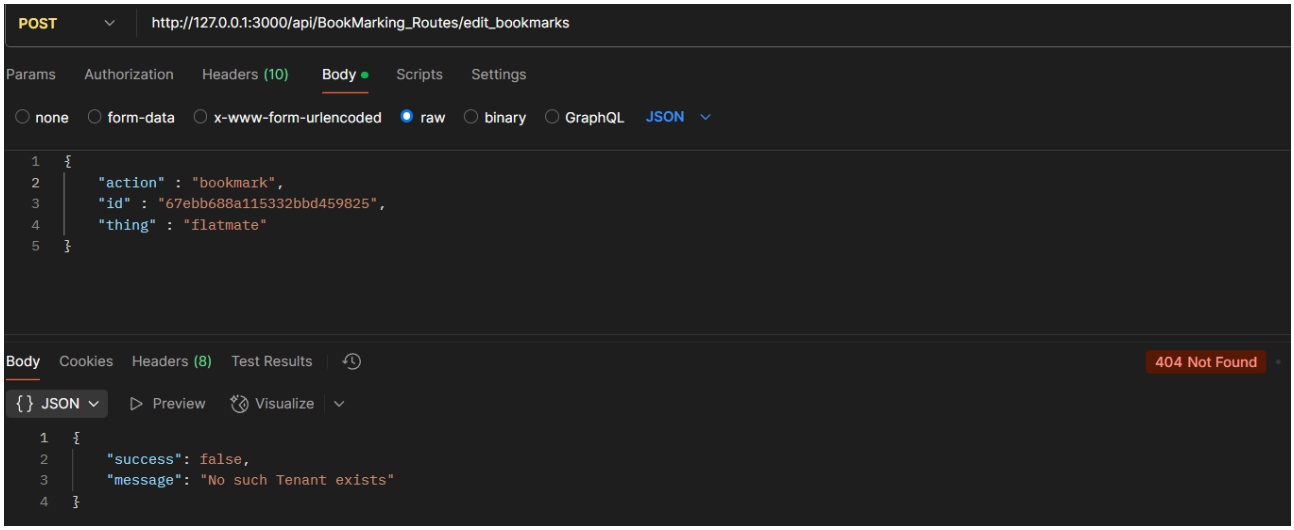


Upon successful Unmark of property a successful response is sent to client and the property is also removed from the database



Trying to unmark a property which isn't bookmarked in the first place





The system always check whether a Flatmate or Property exists with the entered ID, if not an error message is returned with a success code of 404

Structural Coverage: The branch coverage covers the cases when a property which doesn't exists in bookmarks is trying to be unmarked, Wrong syntax of ID, An ID which doesn't exists.

Additional Comments: None

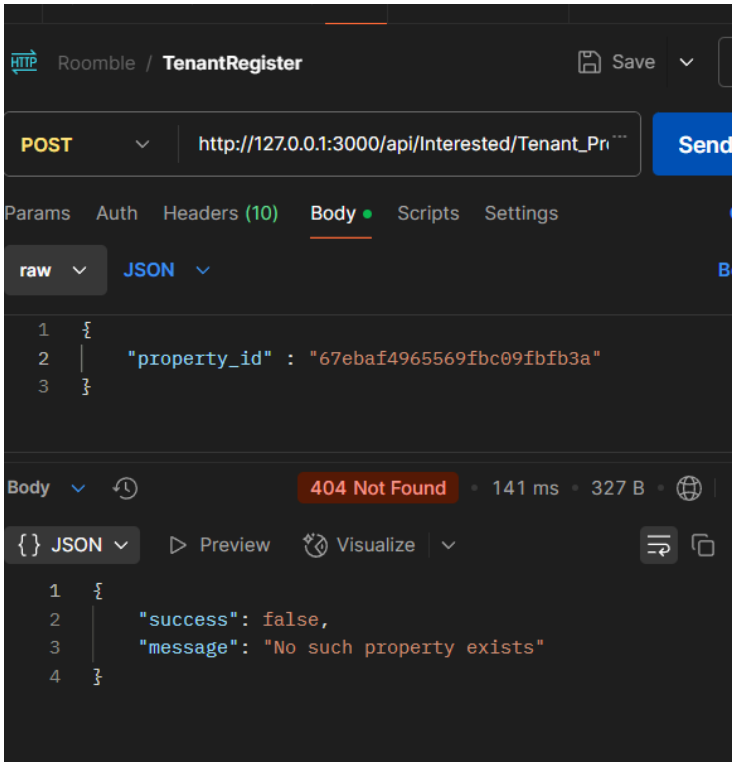
10. Interested in a Property- Backend

In this section we have tested the backend function that notifies a Landlord whenever a Tenant is interested in his Property.

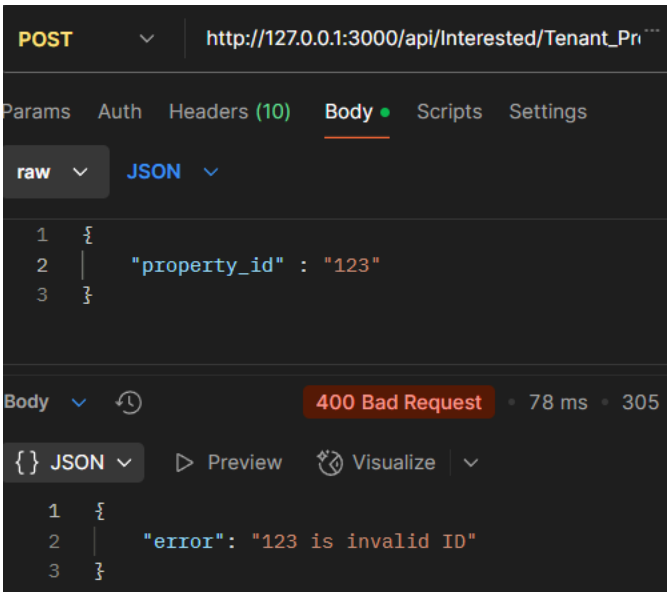
Unit Details: Testing of this unit ensures that the software can successfully notify a Landlord via mail, whenever a Tenant invokes this function.

Test Owner: Saksham Verma
Test Date : 03/04/2025

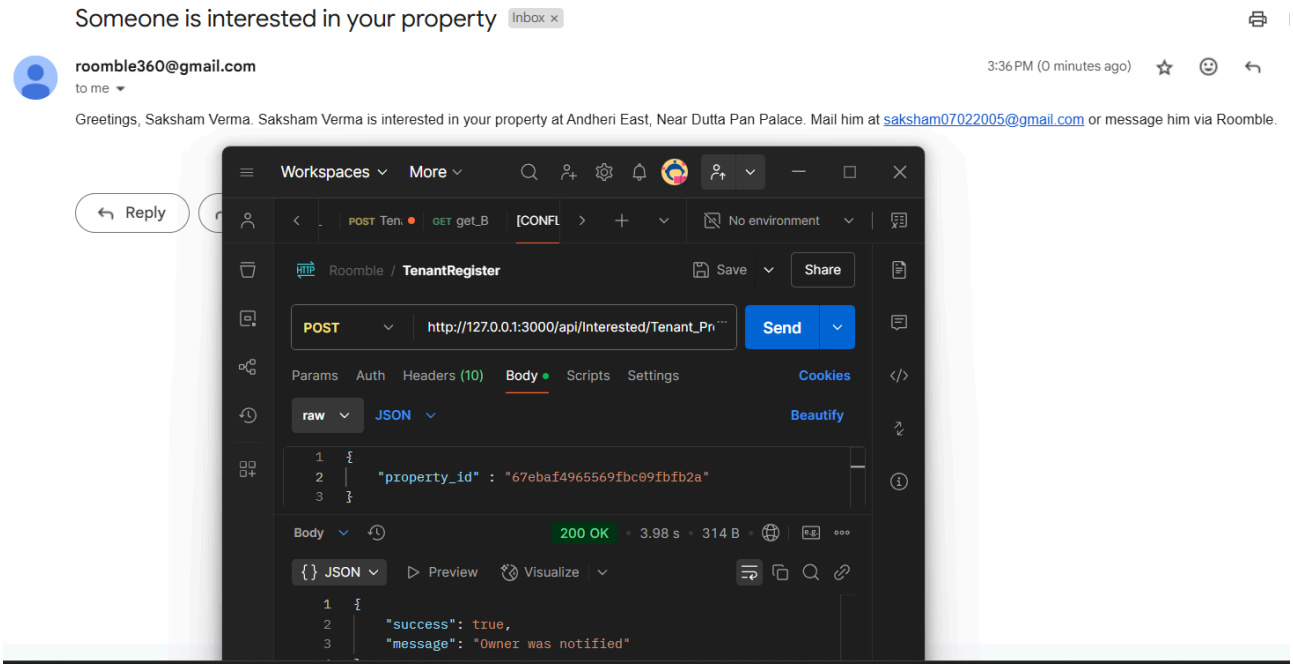
Test Results: Using this functionality the software was successfully able to notify the Landlord via mail whenever a Tenant invoked this function.



The software returns a status code of 404 whenever a wrong Property ID is entered which doesn't exists



The software returns a status code of 400 (Bad request) whenever an Invalid Property ID is entered (Syntax wise)



The owner of the property is notified if the property ID is correct and a status code of 200 is returned

Structural Coverage: This testing took into consideration the branch coverage since both the branches of whether the property exists or not were covered.

Additional Comments: None

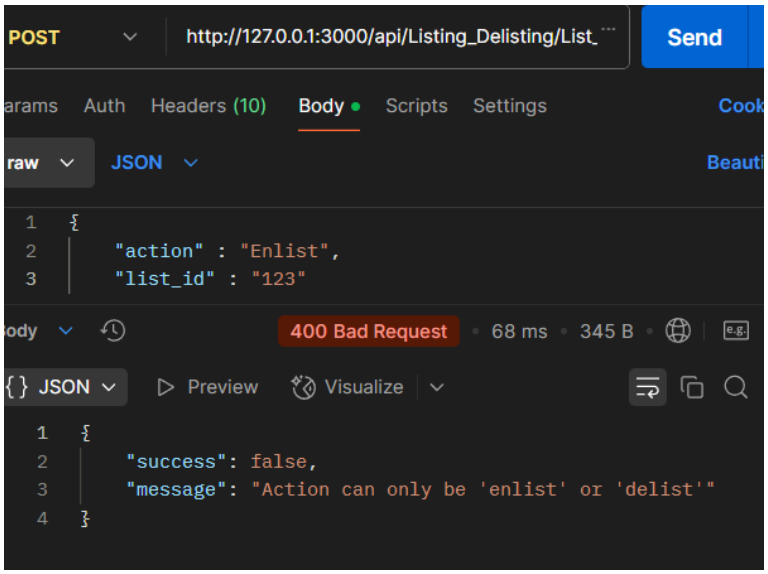
11. Enlisting and Delisting a Property- Backend

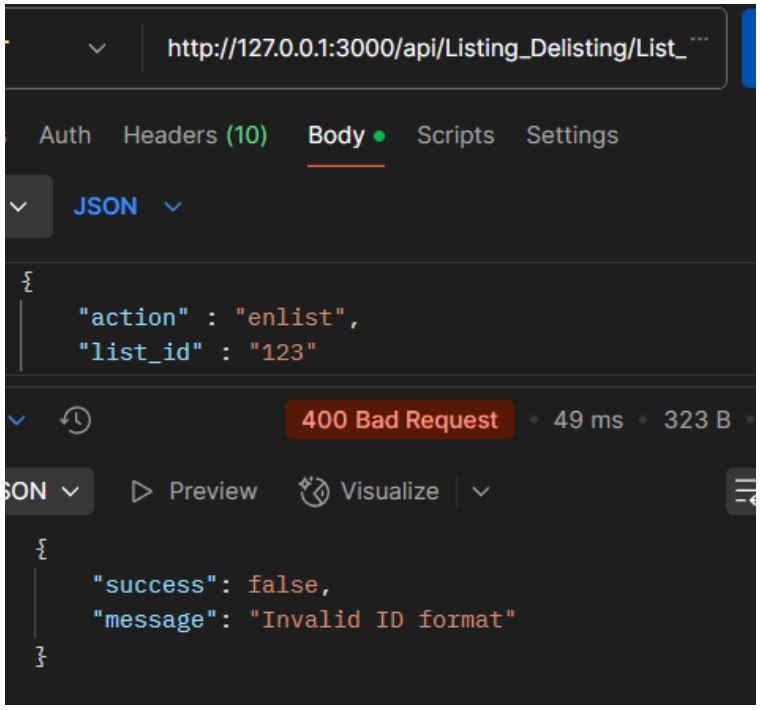
In this section we have tested the backend function that enables Landlord to successfully enlist and delist a property.

Unit Details: Testing of this unit ensures that a Landlord can successfully enlist and delist his properties.

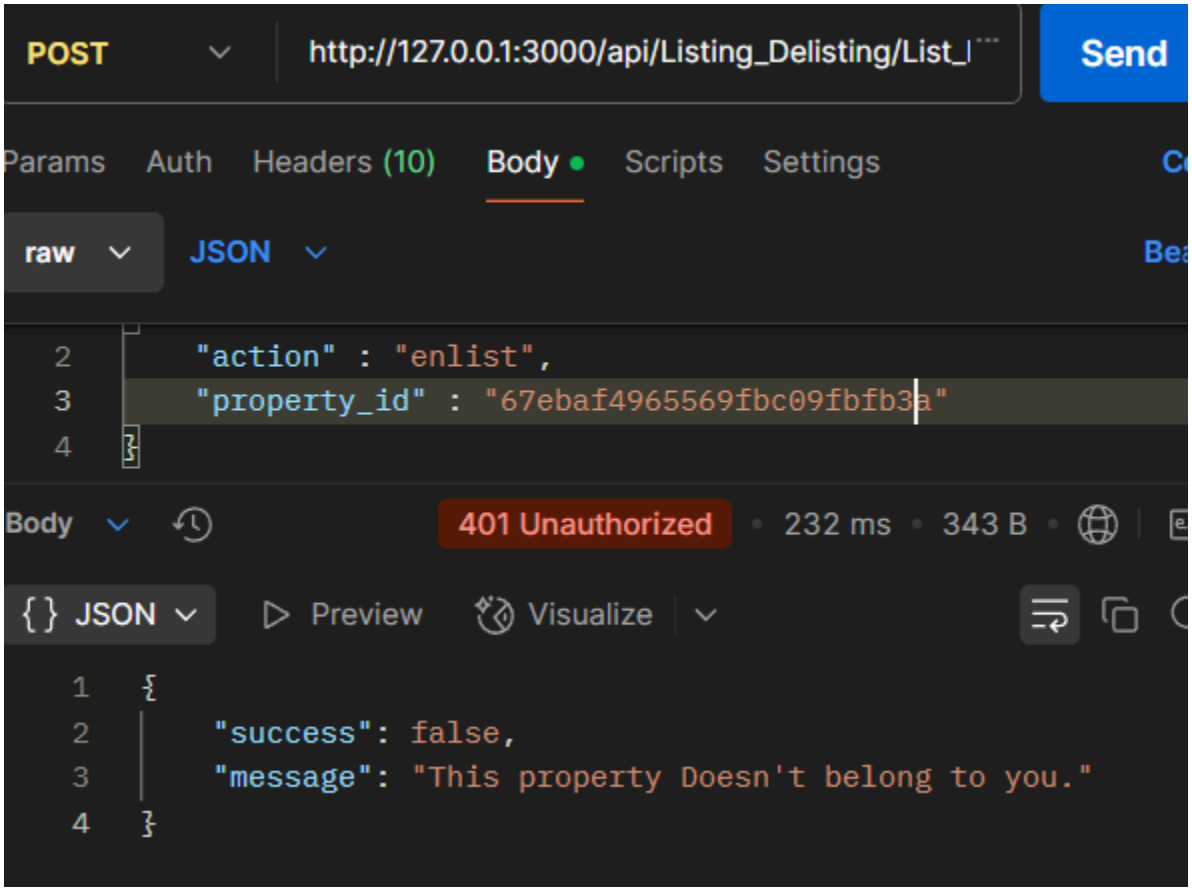
Test Owner: Saksham Verma
Test Date : 03/04/2025

Test Results: Using this functionality a Landlord was successfully able to Delist and enlist his properties and the software restricted People from enlisting / Delisting other peoples properties

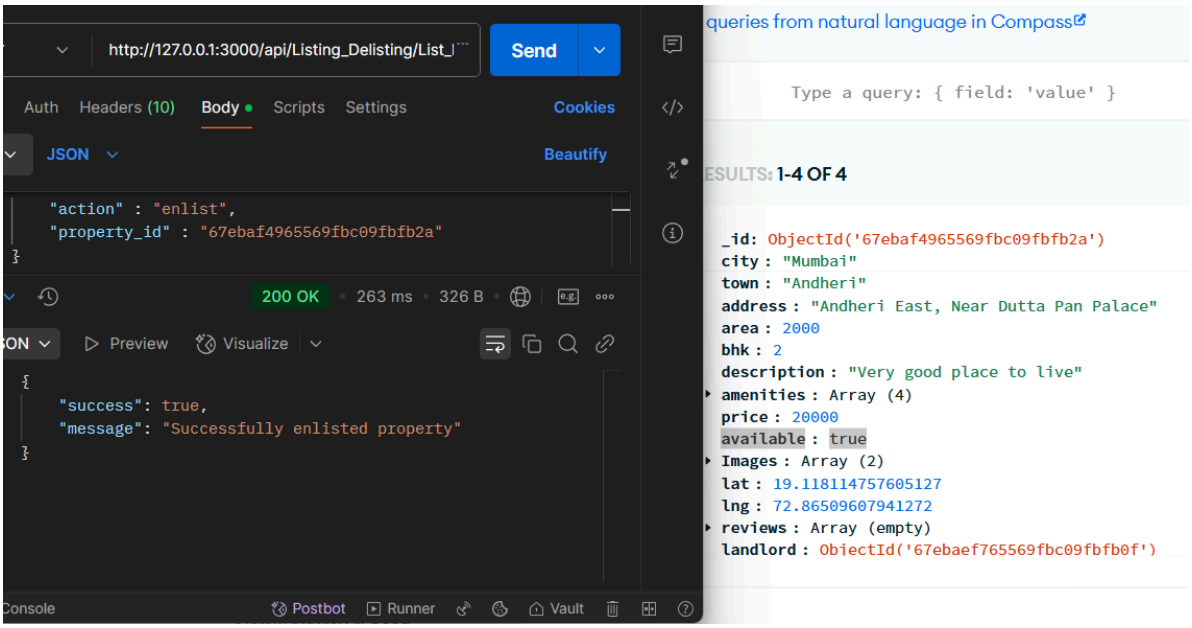




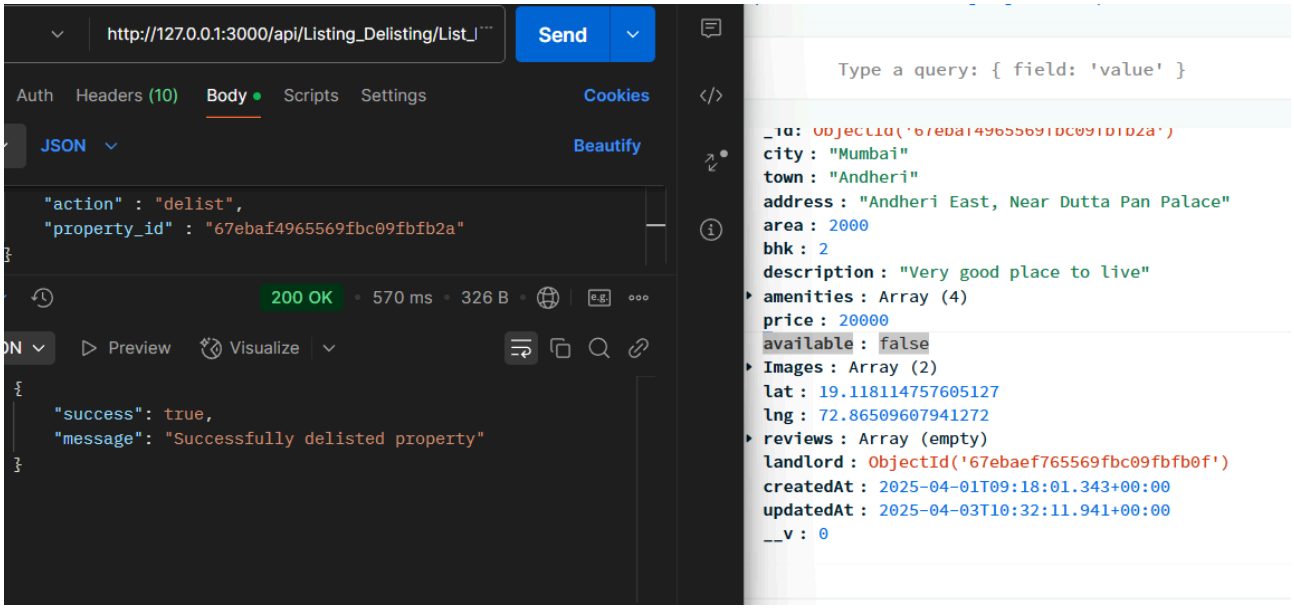
A status code of 400(Bad request) is sent whenever the user doesn't enter a valid ID or a valid Action



A status code of 401 is sent if the property doesn't exists in the Landlords_properties Array



If the property is owned by him, A status code of 200 is returned and the Available attribute is marked as true



Upon successful Delist, the available attribute is marked false and status code of 200 is returned

Structural Coverage: This testing took into consideration the branch coverage since both the branches of whether the property exists or not and Property belongs to him or not were covered.

Additional Comments: None.

12. DeleteProperty- Backend

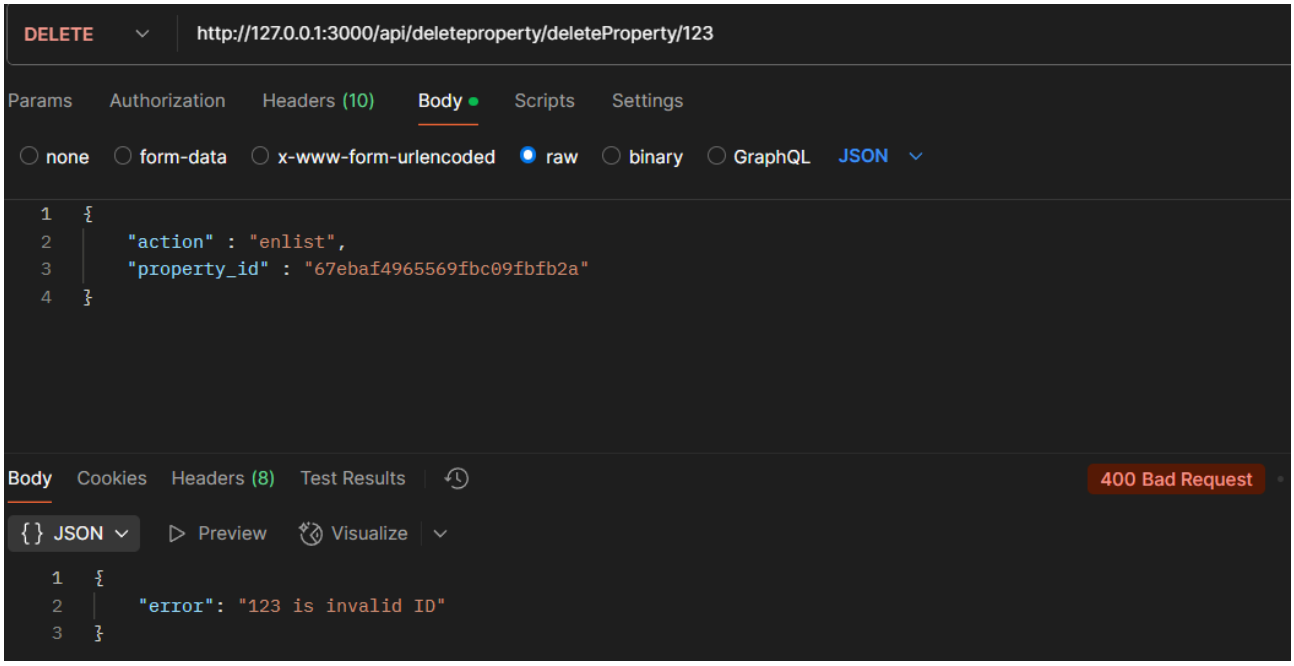
In this unit we have tested the backend function which allows a landlord to delete his own properties from our database

Unit Details: Testing of this unit ensures that a landlord can delete his properties

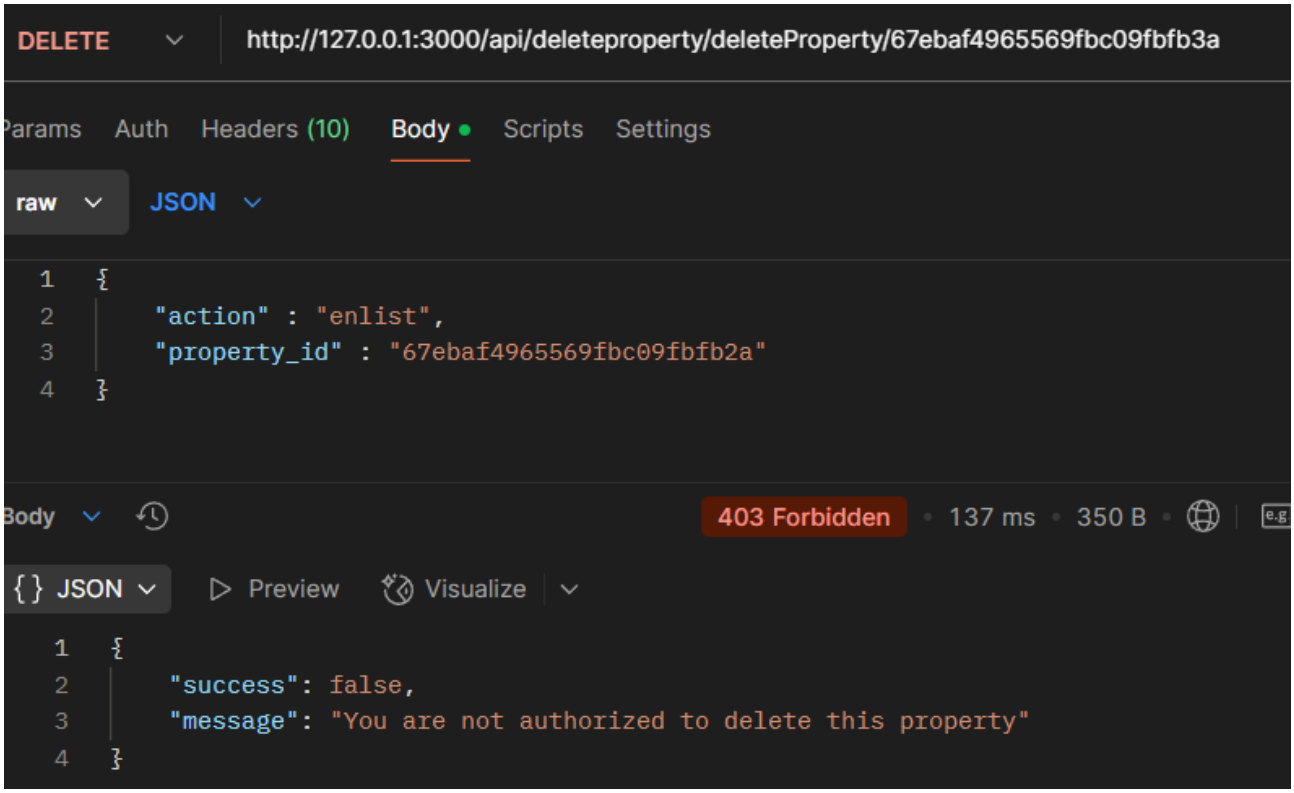
Test Owner: Saksham Verma

Test Date : 02/04/2025

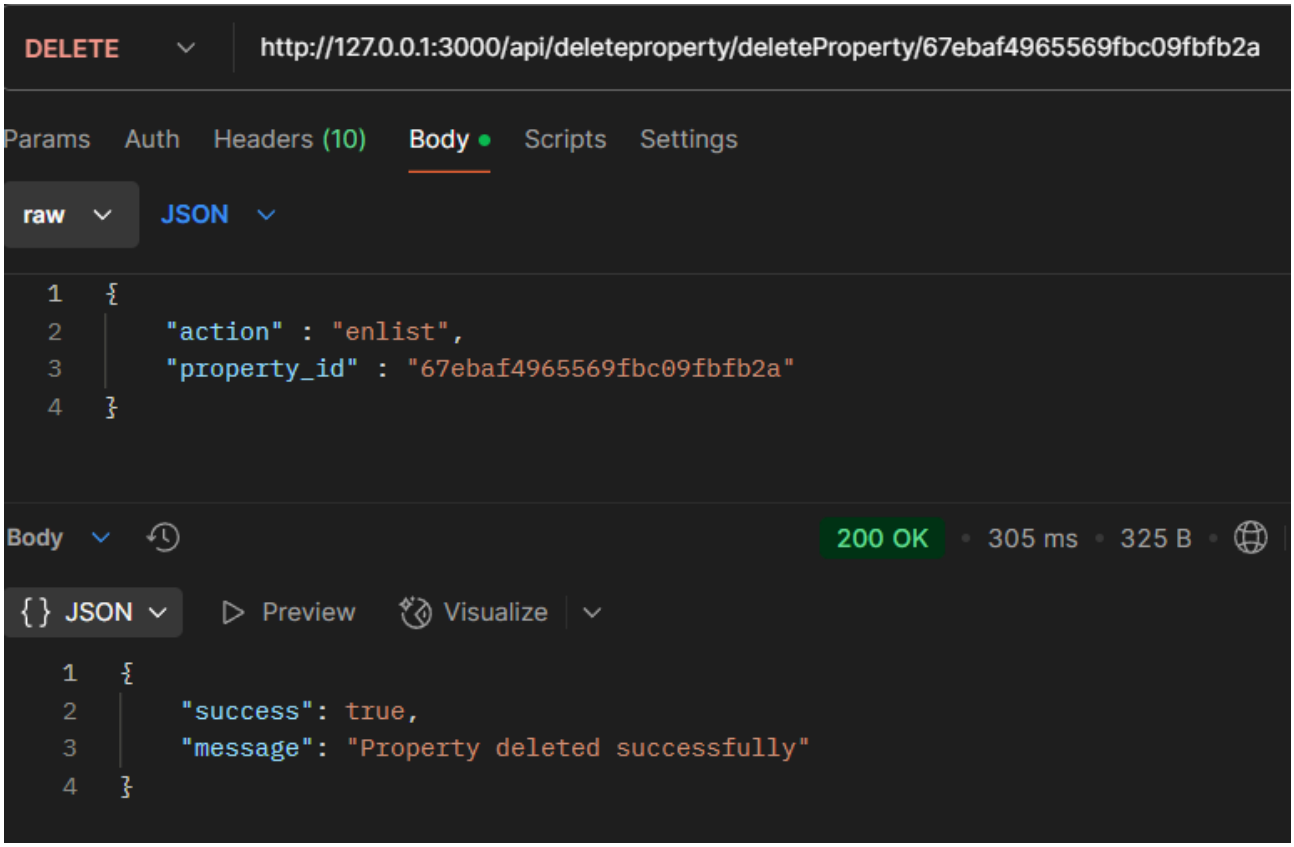
Test Results: Using this functionality the user was able to successfully able to delete the properties that he had listed



A status code of 400 is returned if the syntax of ID is invalid



A status code of 403(Forbidden) is returned if the given ID isn't in Tenant_propertyList



The property is successfully deleted if the property exists in his Tenant_propertyList

Structural Coverage: This testing took into consideration the branch coverage since both the branches of whether the property exists or not and whether the Property belongs to him or not were covered.

Additional Comments: None.

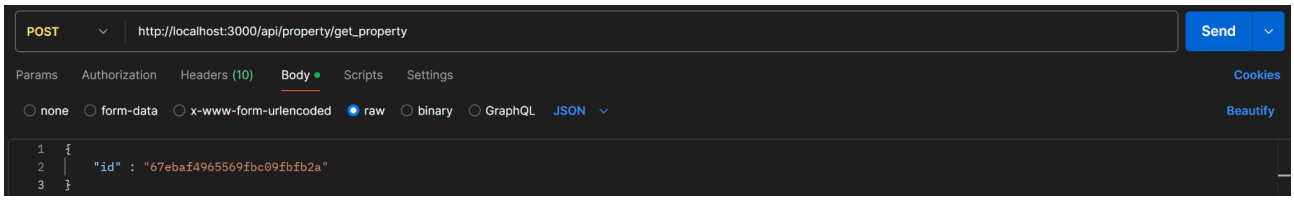
13. ViewProperty- Backend

In this unit, we have tested the backend function which allows a user to view the complete details of a property.

Unit Details: Testing of this unit ensures that the user can successfully view any property that has a valid ID in the database.

Test Owner: Rathod Ayushi
Test Date: 02/04/2025

Test Results: Using this functionality the user was able to successfully view the complete details about the property.

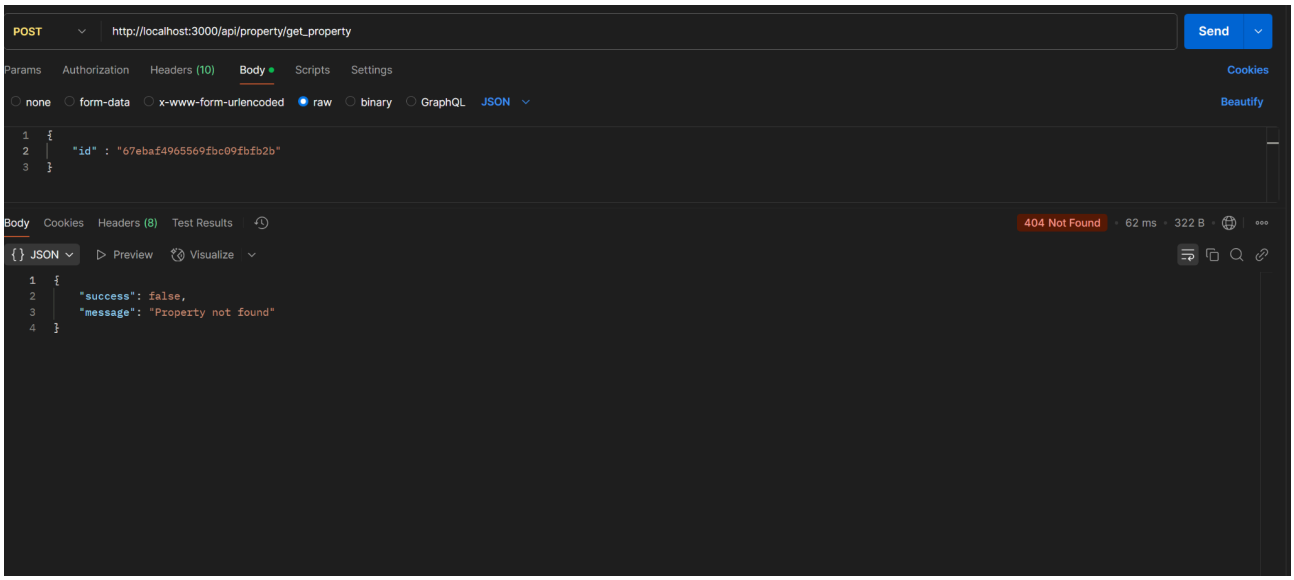


On performing the above POST request through Postman
We got the following results as a json-response:

This result matches with the details of the property available in the database:

```
_id: ObjectId('67ebaf4965569fbc09fbfb2a')
city : "Mumbai"
town : "Andheri"
address : "Andheri East, Near Dutta Pan Palace"
area : 2000
bhk : 2
description : "Very good place to live"
▸ amenities : Array (4)
price : 20000
available : true
▸ Images : Array (2)
lat : 19.118114757605127
lng : 72.86509607941272
▸ reviews : Array (empty)
landlord : ObjectId('67ebaef765569fbc09fbfb0f')
createdAt : 2025-04-01T09:18:01.343+00:00
updatedAt : 2025-04-01T09:18:01.343+00:00
__v : 0
```

Also, in case of searching for an ID which is not available in the database, we get the following response indicating property not found:



Structural Coverage: This testing took into consideration the branch coverage since both branches of whether the property exists or not were covered.

Additional Comments: Along with the details of the property available in the backend, the user also gets the navigating URL of the property, which can be used to locate the property on the map.

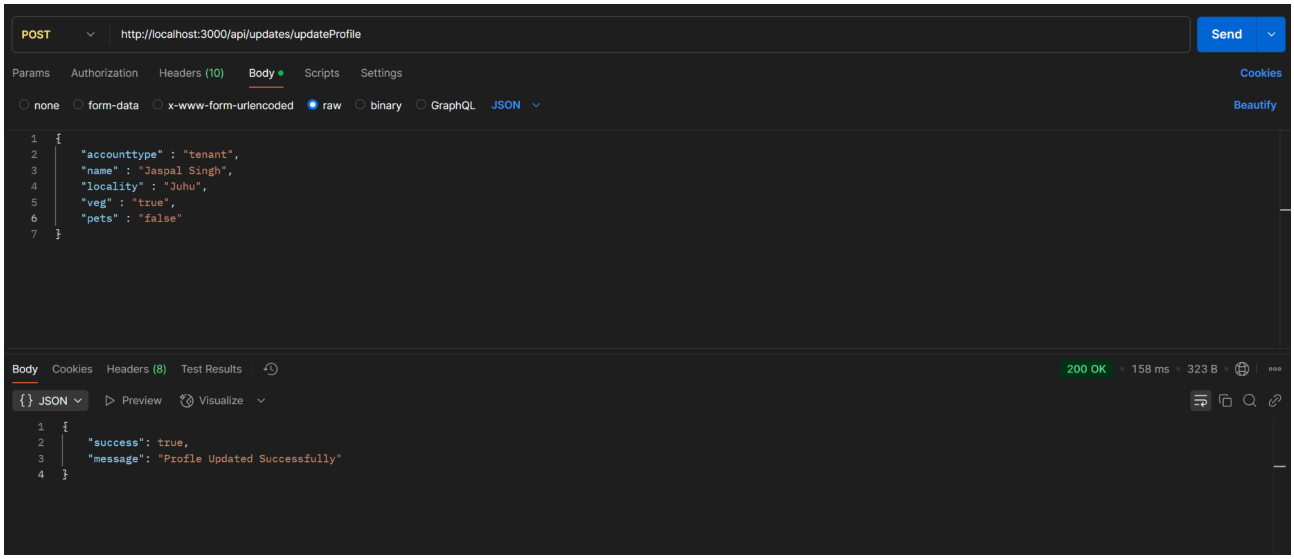
14. UpdateProfile-Backend

This unit tests the backend function that allows the user to update his/her profile, This functionality is available for both the landlord and the tenant.

Unit Details: This unit ensures that the user is able to update his/her profile by filling out the necessary details and that the changes are reflected in the backend database.

Test Owner: Bikramjeet Singh
Test Date: 02/04/2025

Test Results: The unit successfully updated the user profile by filling out the necessary details.

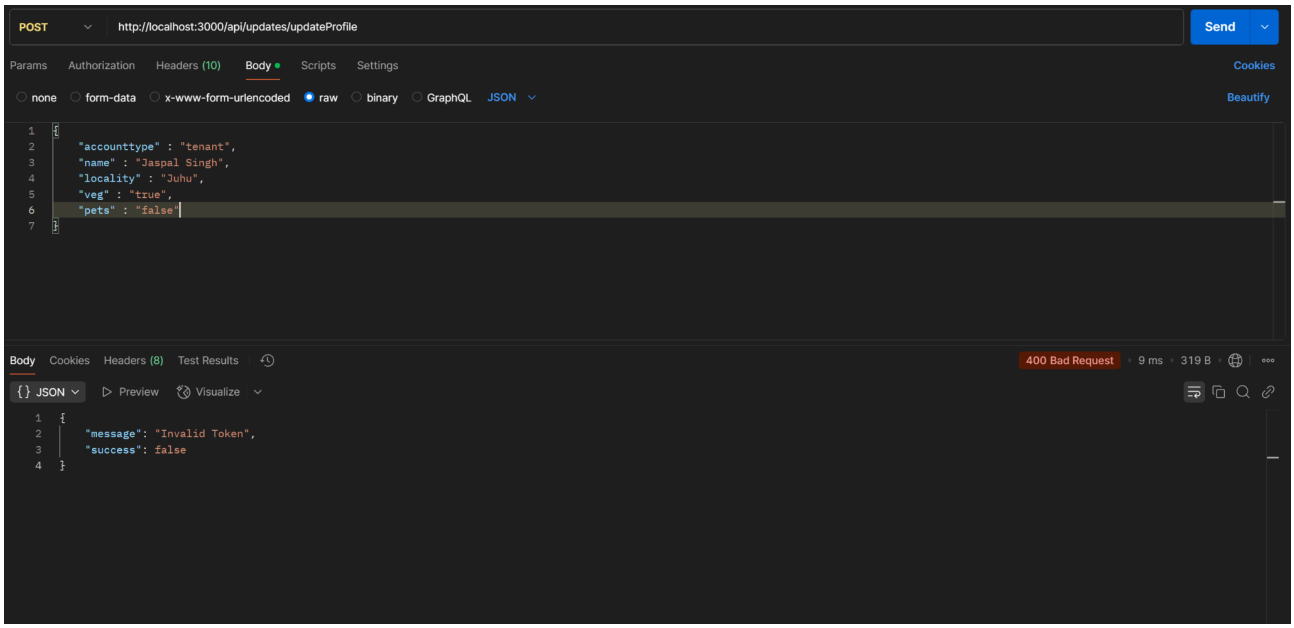


Upon doing a POST request with the account type of the account and the updated fields, the landlord profile was updated successfully, and changes also appeared in the database:

The above is before the update request; afterwards, it looked like this:


```
_id: ObjectId('67ec0f32e9f12674c62a5ad4')
name : "Jaspal Singh"
type : "tenant"
email : "bsingh23@iitk.ac.in"
password : "$2b$10$OQWpJJHaQMc.RW3ckygvtOIND3Pcv5luOrur4xZcJrSep0pv5K/8y"
locality : "Juhu"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : false
flatmate : true
description : "This user hasn't setup a description yet"
conversations : Array (empty)
bookmarks_tenants : Array (2)
bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
reviews : Array (empty)
__v : 1
```


Clearly, the changes requested in the POST request are reflected successfully in the database/backend.



Also, if the auth-token being sent in the headers is invalid/expired, then an appropriate error message is generated, here the auth-token was deliberately modified to test this branch.

Structural Coverage: This unit takes into account the branch coverage of this functionality by checking whether the auth token is valid or not.

Additional Comments: This route does not allow the user to change his/her password or email, we have those routes available in the backend but have not successfully integrated them with the frontend.

15. UpdateProperty-Backend

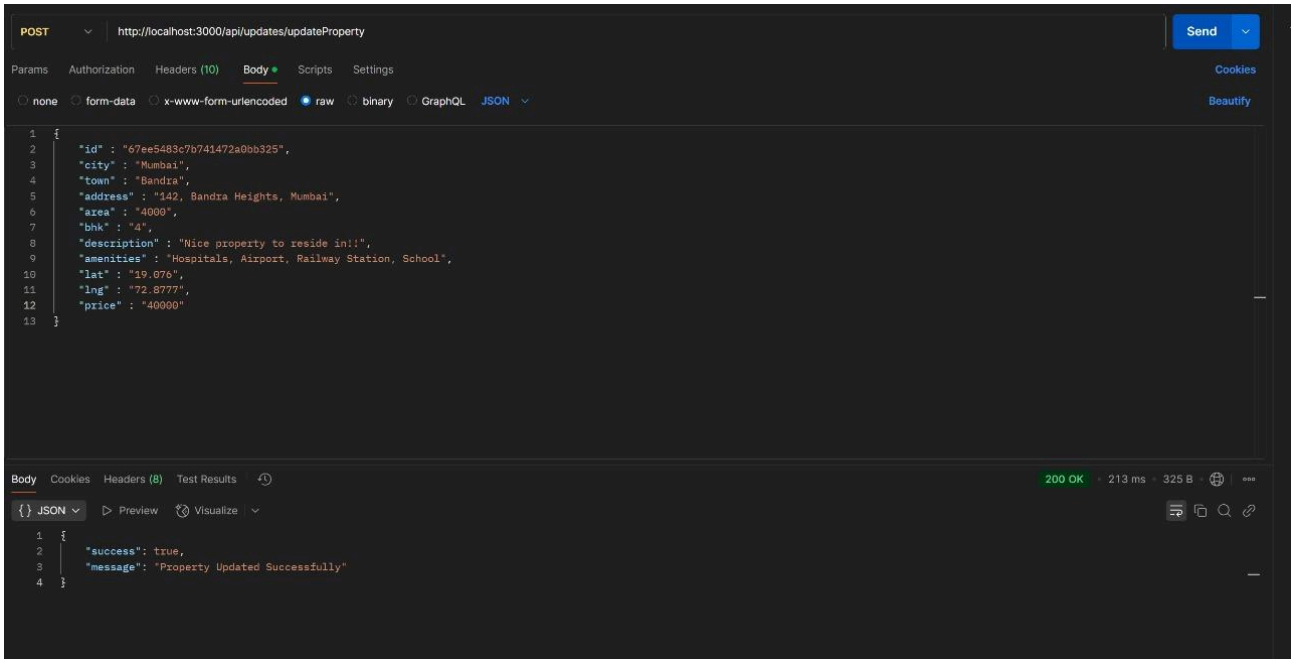
This unit tests the backend function, which allows the landlord to update his/her property by filling out the necessary details.

Unit Details: This unit ensures that the landlord can easily update any details regarding his/her property.

Test Owner: Bikramjeet Singh
Test Date: 03/04/2025

Test Results:

This unit was successful in updating the details of the property of a verified landlord, that is, the actual owner of the property.



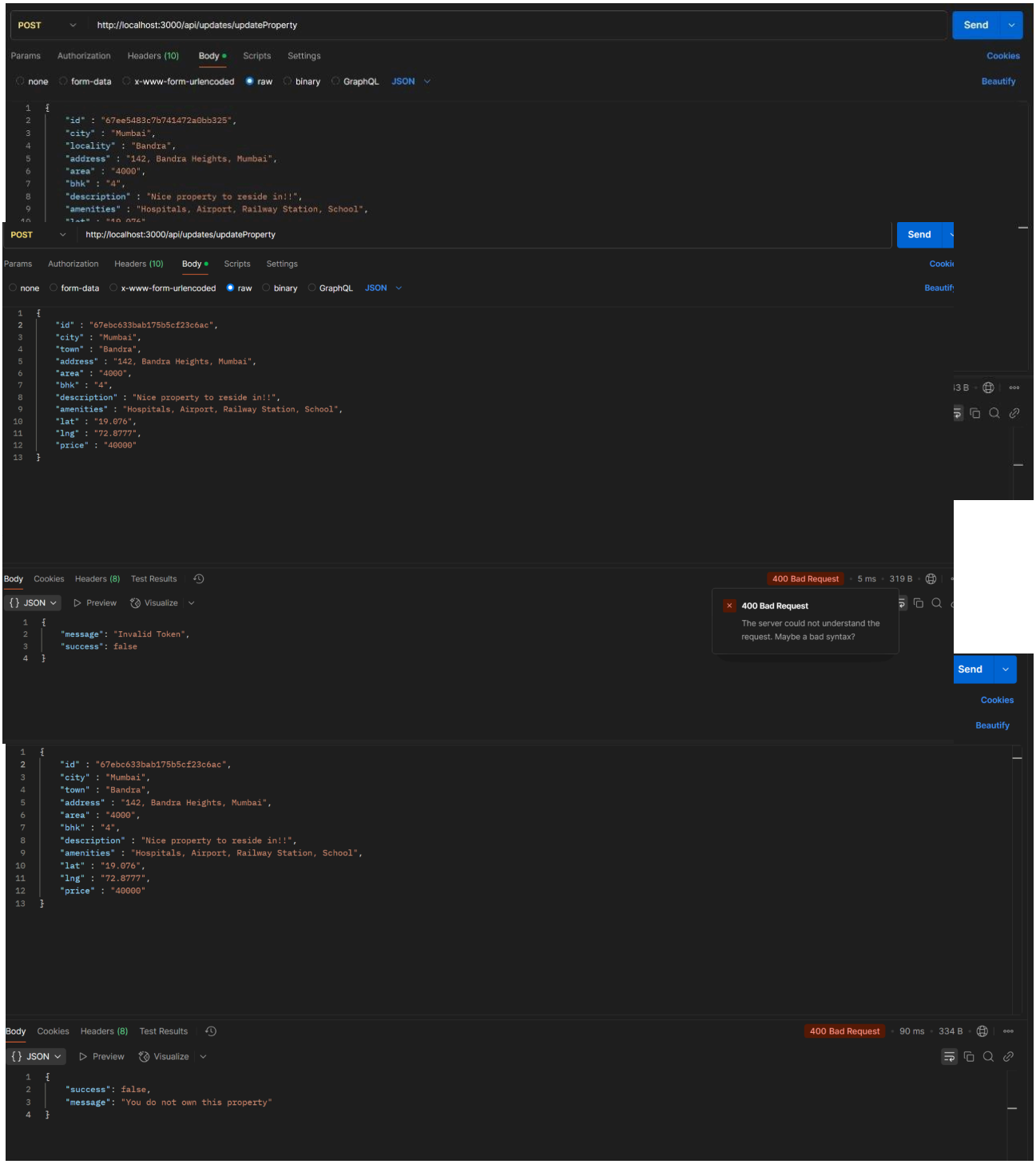
Upon initiating the above POST request, a success message indicating successful The update of the property was shown, which is also reflected in the database:

The above are the details of the property before the update; below are the same After the update:


```
_id: ObjectId('67ee5483c7b741472a0bb325')
city : "Mumbai"
town : "Bandra"
address : "142, Bandra Heights, Mumbai"
area : 4000
bhk : 4
description : "Nice property to reside in!!"
▸ amenities : Array (1)
price : 40000
available : true
▸ Images : Array (1)
lat : 19.076
lng : 72.8777
▸ reviews : Array (empty)
landlord : ObjectId('67ee47b5e7e3855709247863')
createdAt : 2025-04-03T09:27:31.258+00:00
updatedAt : 2025-04-03T09:33:00.230+00:00
__v : 1
```


Also in case the user misses some of the necessary fields for the property, then the user is shown the following error message:


```
_id: ObjectId('67ee5483c7b741472a0bb325')
city : "Mumbai"
town : "Juhu"
address : "143, Bandra Heights, Mumbai"
area : 4000
bhk : 4
description : "Nice Property to live-in, surrounded by a peaceful atmosphere alongwit…"
▸ amenities : Array (4)
price : 30000
available : true
▸ Images : Array (1)
lat : 19.076
lng : 72.8777
▸ reviews : Array (empty)
landlord : ObjectId('67ee47b5e7e3855709247863')
createdAt : 2025-04-03T09:27:31.258+00:00
updatedAt : 2025-04-03T09:27:31.258+00:00
__v : 0
```



Also, if any unverified landlord tries to update any property that is his/her auth-token is expired or invalid The following response is shown to the user:

Hence, we can conclude that if a verified landlord tries to update his/her owned property, then the property is successfully updated.

Structural Coverage: This particular test completes branch coverage since we are testing for invalid landlords, invalid properties, and valid properties that are not owned by the concerned landlord.

Additional Comments: None.

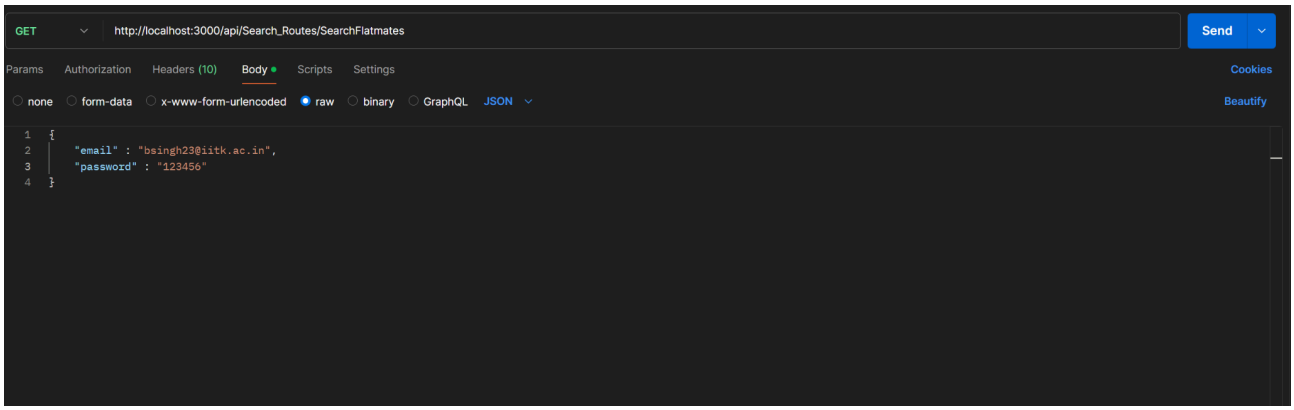
16. SearchingFlatmates-Backend

In this unit, we test the backend function that allows a tenant to search for possible flatmates by applying various filters of his/her choice.

Unit Details: The efficient and correct working of this unit will ensure that the tenant gets the list of flatmates that match his/her priorities and hence would be able to make an informed decision regarding that choice.

Test Owner: Bikramjeet Singh
Test Date: 03/04/2025

Test Results:
The testing of this unit showed that it is indeed working as expected:



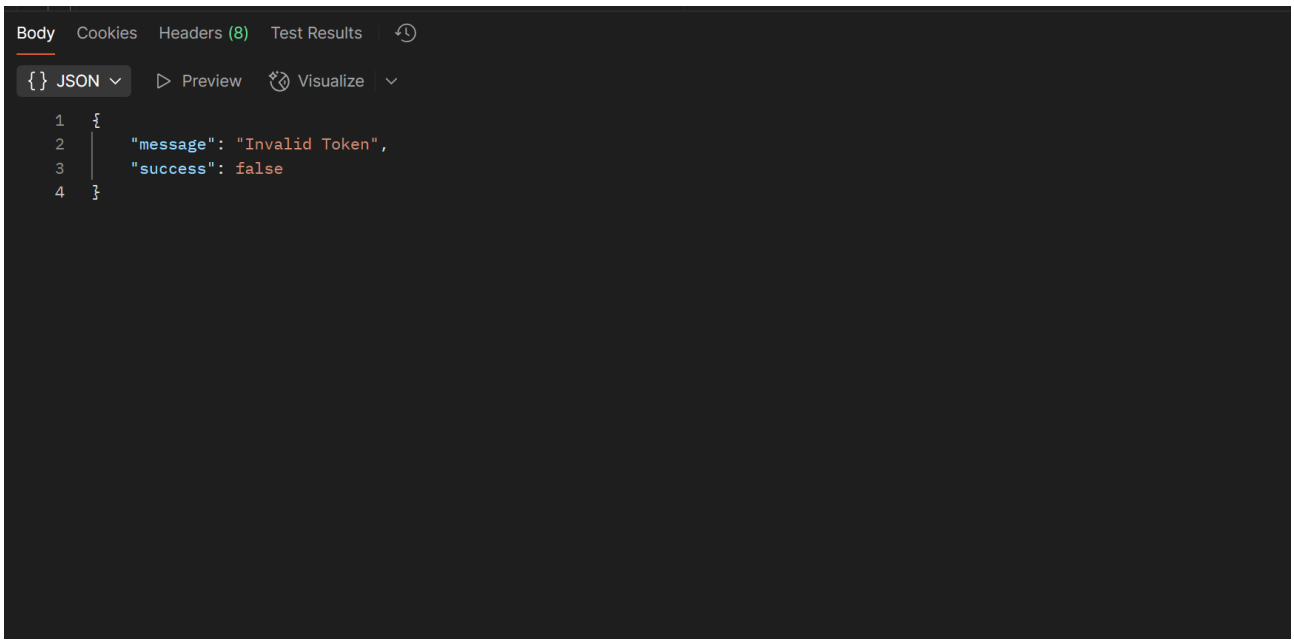
The above GET request was initiated in POSTMAN

The following list of flatmates was obtained. here no filter is being applied:

```
"success": true,
"message": "Found 4 matching tenants",
"data": [
    {
        "_id": "67ebae9765569fbc09fbfafe",
        "name": "Saksham Verma",
        "type": "tenant",
        "email": "saksham07022005@gmail.com",
        "locality": "Bandra",
        "city": "Mumbai",
        "gender": true,
        "smoke": true,
        "veg": false,
        "pets": true,
        "flatmate": true,
        "description": "This user hasn't setup a description yet",
        "conversations": [
            "67ebb708a115332bbd4598aa",
            "67ebc33dc268d28a44491018"
        ],
        "bookmarks_tenants": [],
        "bookmarks_property": [],
        "Images": "http://127.0.0.1:3000/Pictures/Default.png",
        "reviews": [],
        "__v": 4,
        "recommendationScore": 0.925,
        "bookmarked": true
    },
    {
        "_id": "67ebc4919a575e22057f036f",
        "name": "Hitarth Makawana",
        "type": "tenant",
        "email": "hitarthkm23@iitk.ac.in",
        "locality": "Andheri",
        "city": "Mumbai",
        "gender": true,
        "smoke": false,
        "veg": true,
        "pets": true,
        "flatmate": true,
        "description": "This user hasn't setup a description yet",
        "conversations": [
            "67ebc5a59a575e22057f03da"
        ],
        "bookmarks_tenants": [],
        "bookmarks_property": [
            "67ebc94c9a575e22057f043a"
        ],
        "Images": "http://127.0.0.1:3000/Pictures/Default.png",
        "reviews": [
```


Showing two out of the four due to space and clarity constraints.

Only a verified tenant is allowed to see the flatmates; if a tenant with an expired or invalid auth-token attempts to do so, he/she is denied :



```
Body  Cookies  Headers (8)  Test Results

{ } JSON  ▷ Preview  Visualize

{
    "message": "Invalid Token",
    "success": false
}
```


Structural Coverage: This testing took into account the functional coverage and

partial branch coverage since we did not mention here the result with the filter applied to it.

Additional Comments: This function also returns the filtered list of the possible flatmates, although it has not been shown here since it is apt to be shown in the integration part under the frontend part.

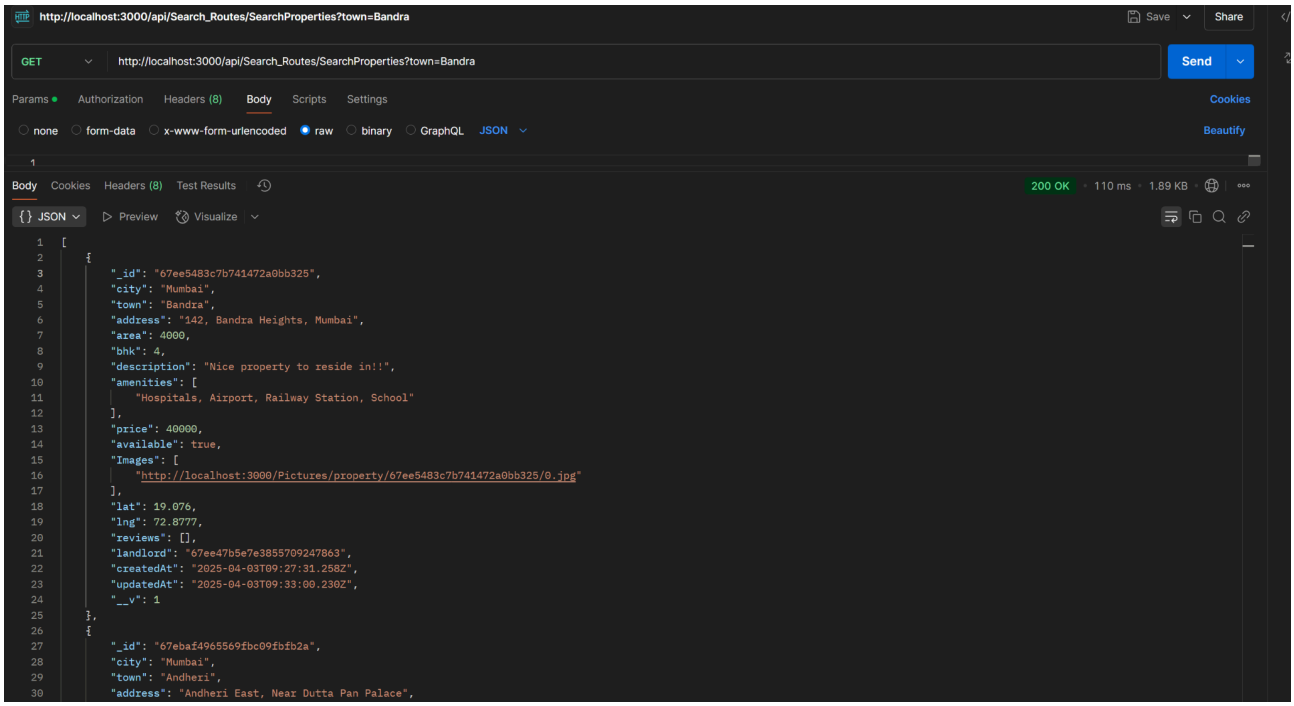
17. Searching Property-Backend

In this unit, we test the backend function that allows a verified tenant to search for properties as per his/her locality of choice and also filter on the basis of BHK, price, and area requirements.

Unit Details: The successful & efficient working of this unit must allow the tenant to search for the property based on the various filters.

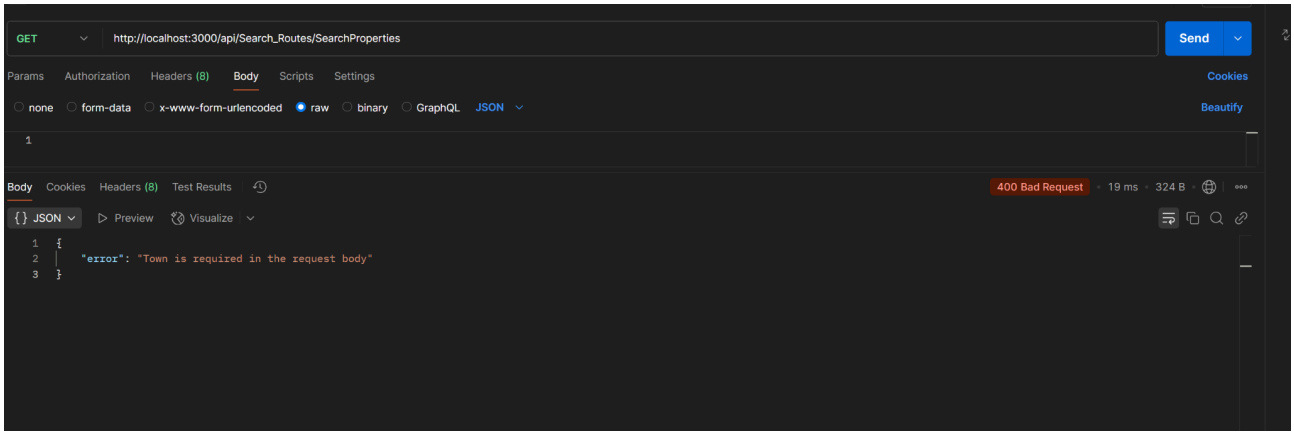
Test Owner: Bikramjeet Singh
Test Date: 03/04/2025

Test Results: The unit successfully allowed the tenant to search properties in the locality of his/her choice:

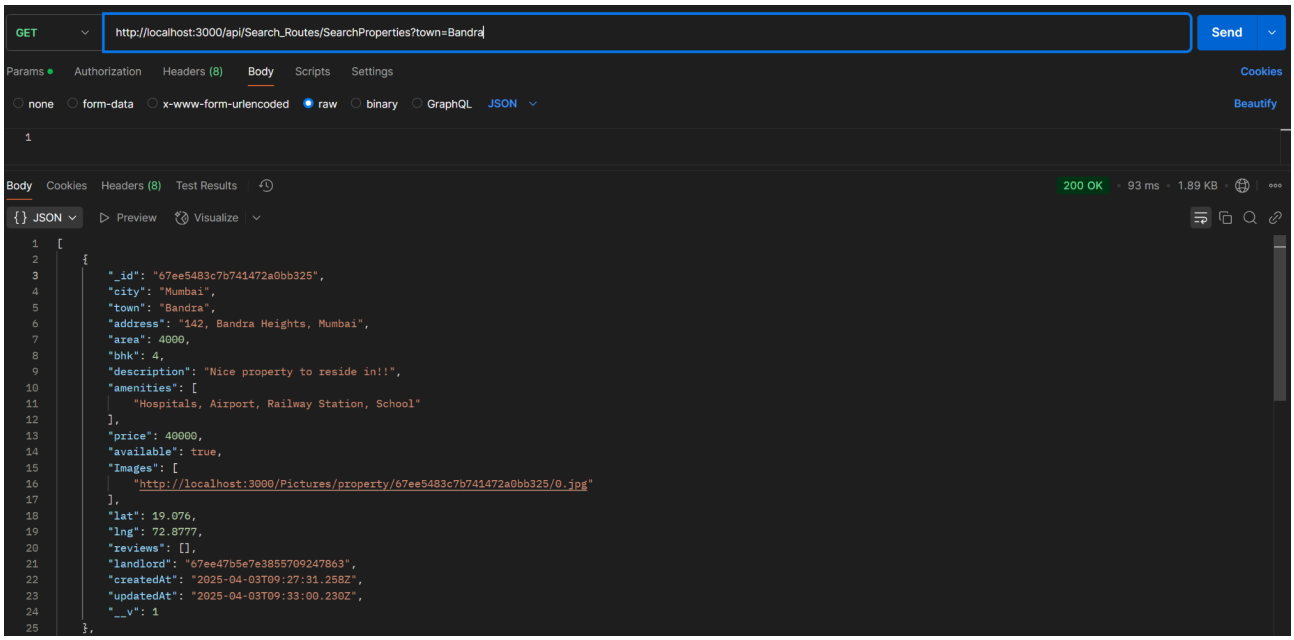


Upon initiating the GET request using POSTMAN, we get the responses according to the town that has been mentioned in the query string.

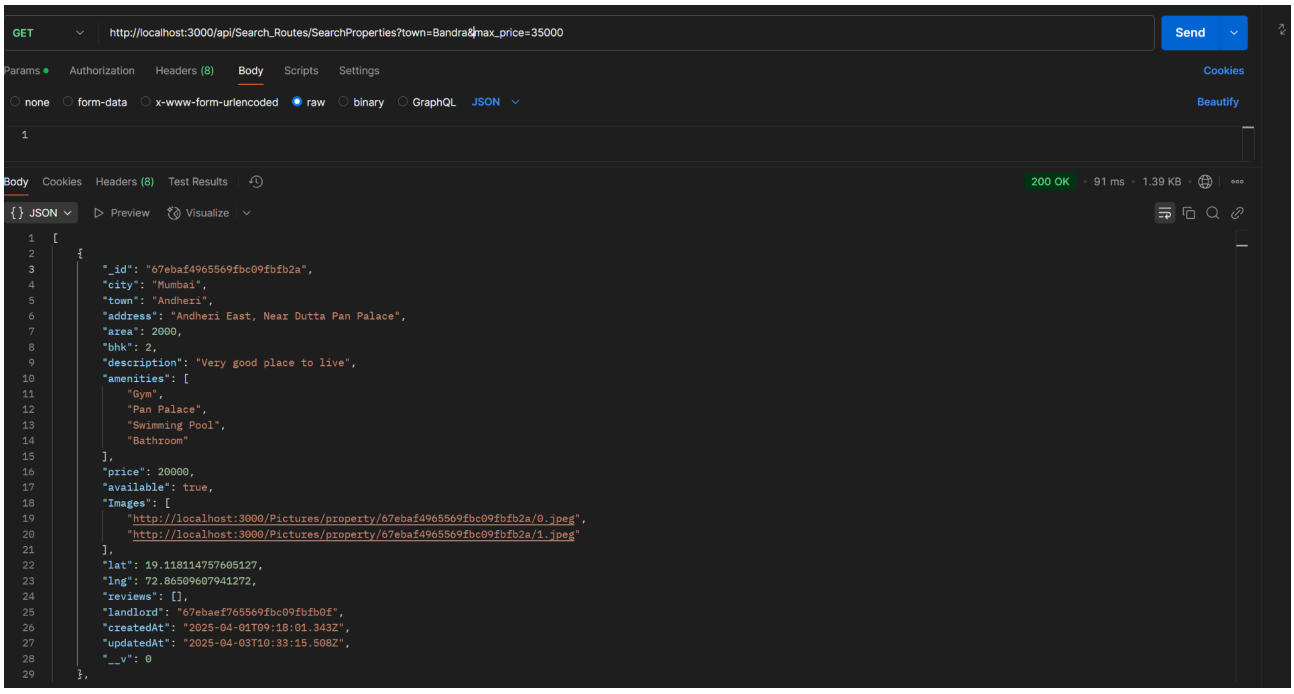
Also, in case, we do not send the town in the query string we get the following error indicating that it is indeed mandatory to send the town:



Applying filters also gives correct results:



Before applying any filter, we get the above result; however applying the filter say regarding the maximum price being 35000 provides the following result:



which clearly removes the initial property which had a price of more than 35000.

Structural Coverage: The testing covers the functional coverage and partial branch coverage.

Additional Comments: Although not explicitly shown here, there are many more filters that have been implemented to filter out properties on more specific choices of the user.

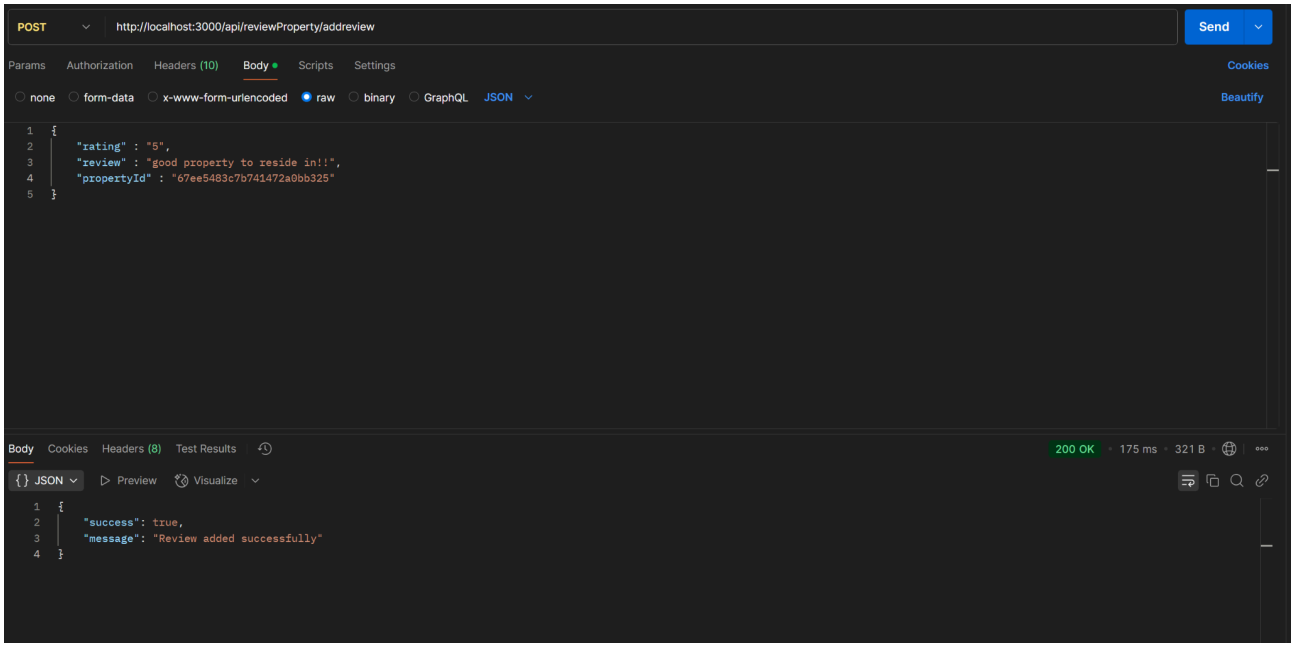
18. Review Property-Backend

This unit tests the backend function, which allows a user to give a review about any property.

Unit Details: This unit ensures the tenant is able to post a review about some property in the form of a rating out of 5 stars as well as descriptive writing about it.

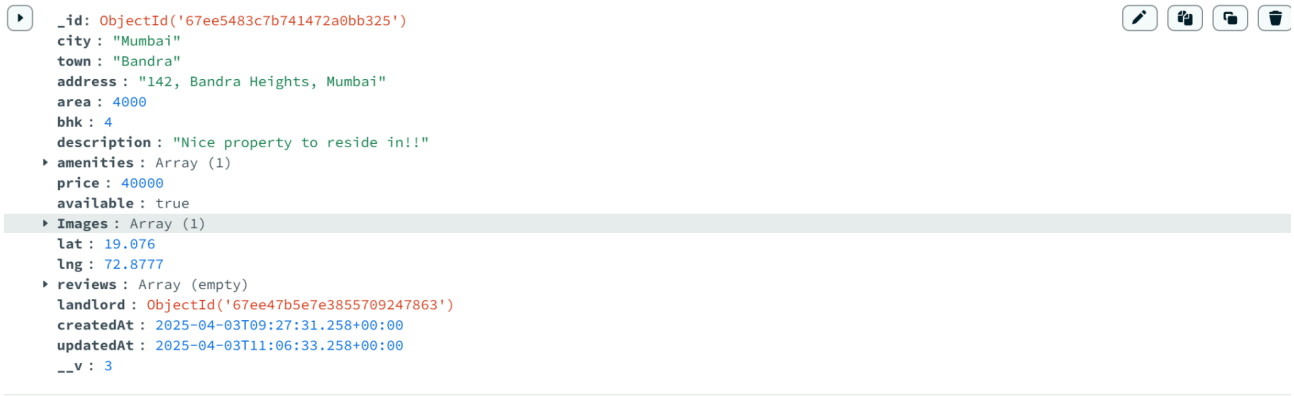
Test Owner: Bikramjeet Singh
Test Date: 03/04/2025

Test Results: The unit was successful in allowing the user to post a review about a certain property, including rating and the descriptive review.



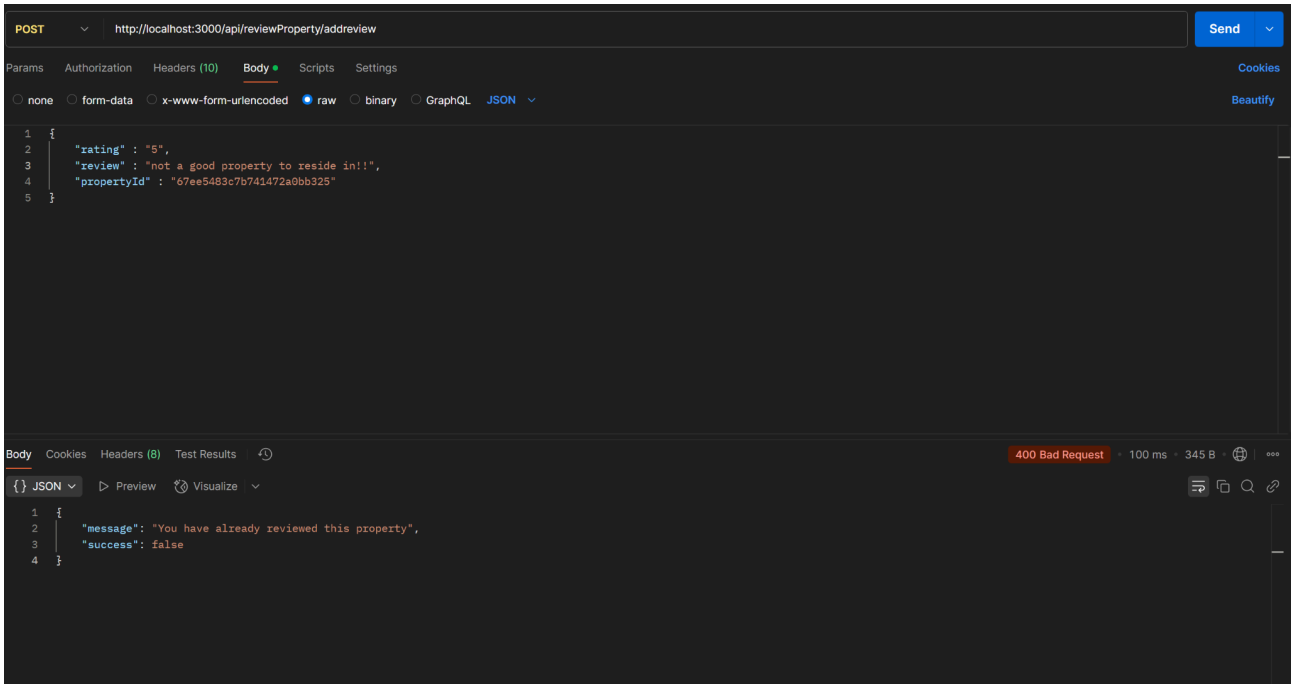
129392

The above POST request successfully updates the review in the database as seen clearly below:

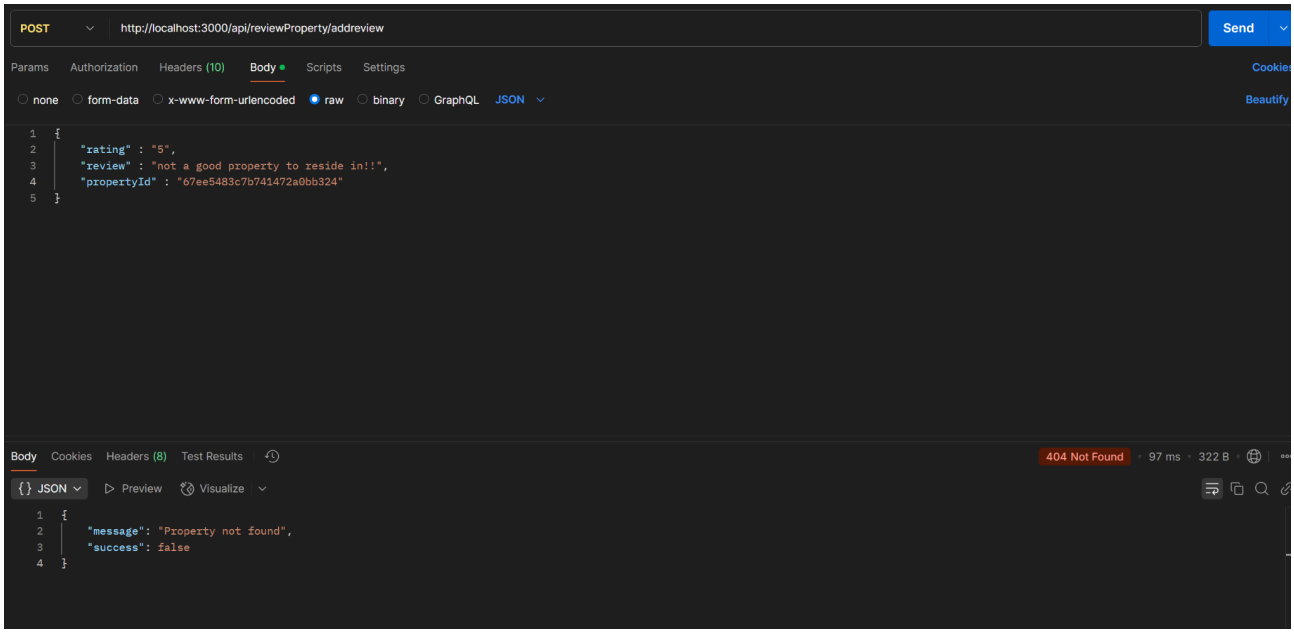


The array of reviews is empty before the POST request, but after the POST request, a review appears in the same array:

Also, if the user tries to review the same property twice/or more he is not allowed to:



Also, the user is not allowed to give a review to an invalid property:



Structural Coverage: This testing takes into account compleet branch coverage.

Additional Comments: None.

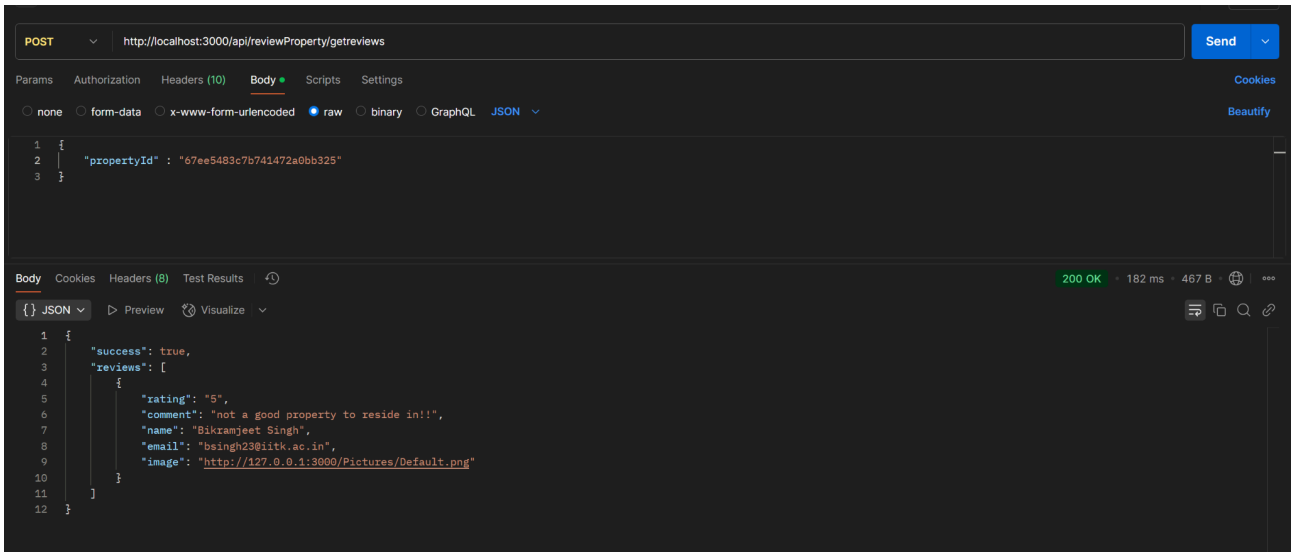
19. Get Property Reviews- Backend

This unit tests the backend function which basically allows a user to fetch and see the reviews about a certain property.

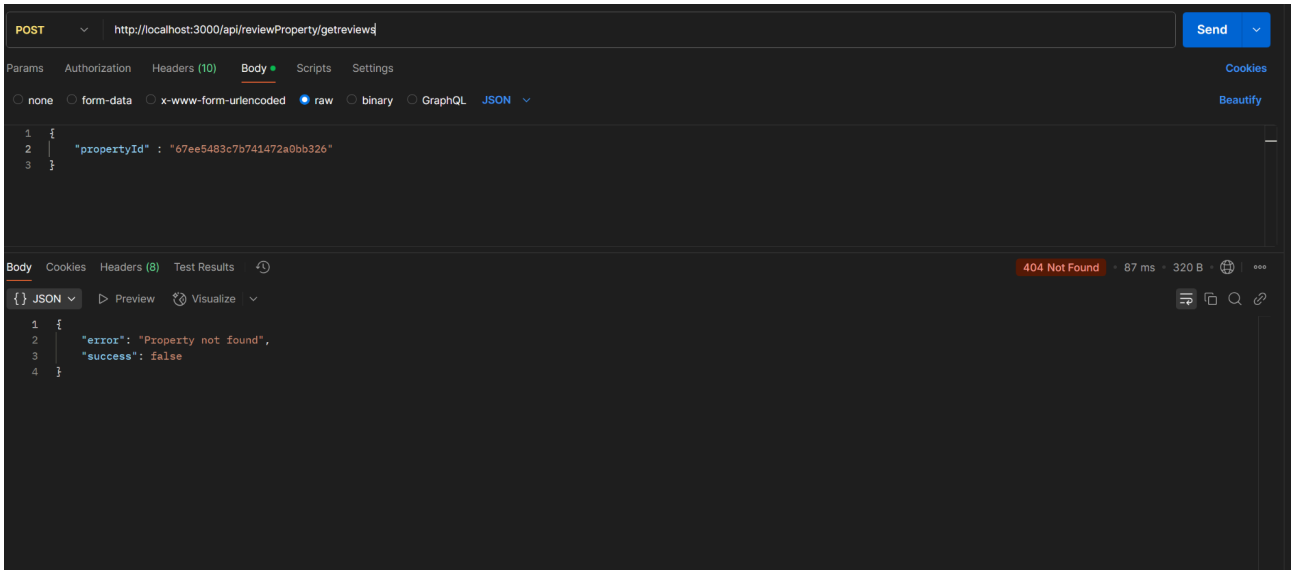
Unit Details : The successful working of this unit ensures that the user is able to see the reviews regarding a particular property along with the person who has posted the review.

Test Owner: Bikramjeet Singh
Test Date : 03/04/2025

Test Results: The unit was successfully tested and gave apt results when reviews of a particular property were demanded.



The above POST request generated the response in the above image, which correctly shows all the details about the review be it the rating, the reviewee or the content.



The above POST request which provides an invalid ID is provided an appropriate response regarding the same.

Structural Coverage: The above testing covers functional coverage and branch coverage.

Additional Comments: None.

20. Messaging and concerned functionalities- Backend

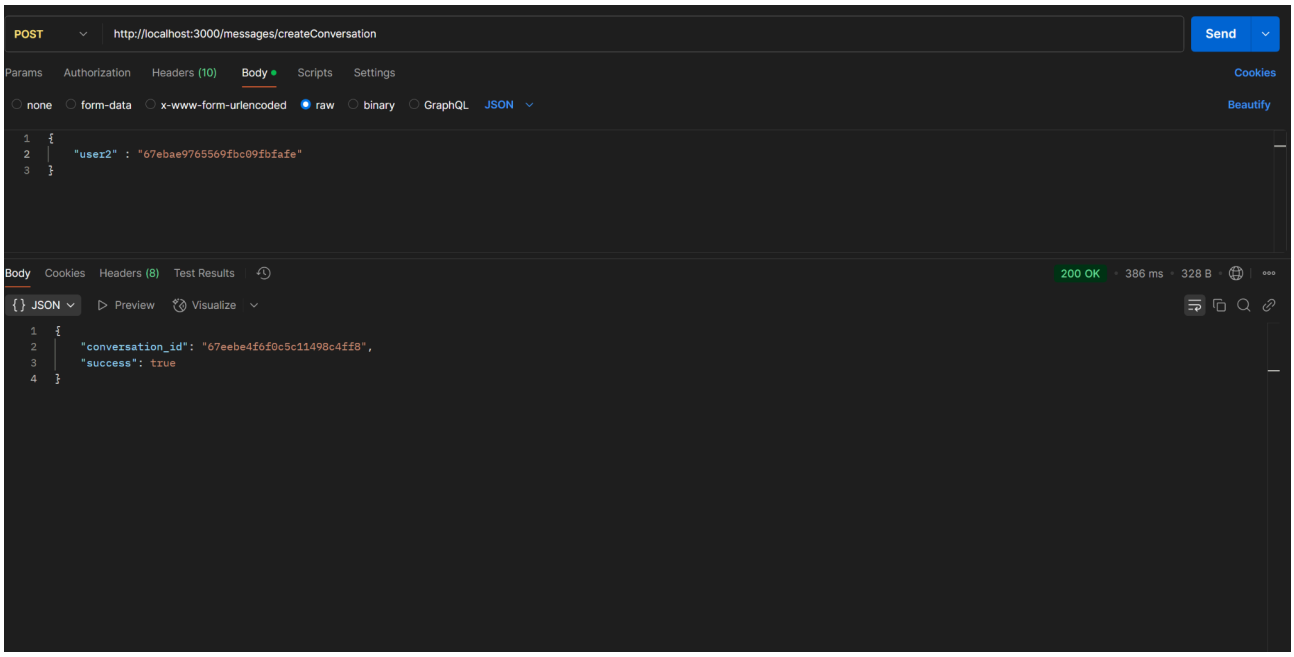
This unit tests the backend route, which allows the user to send a message to another user in real time.

Unit Details: The successful working of this unit ensures that the user can send/ receive a message to/from another user.

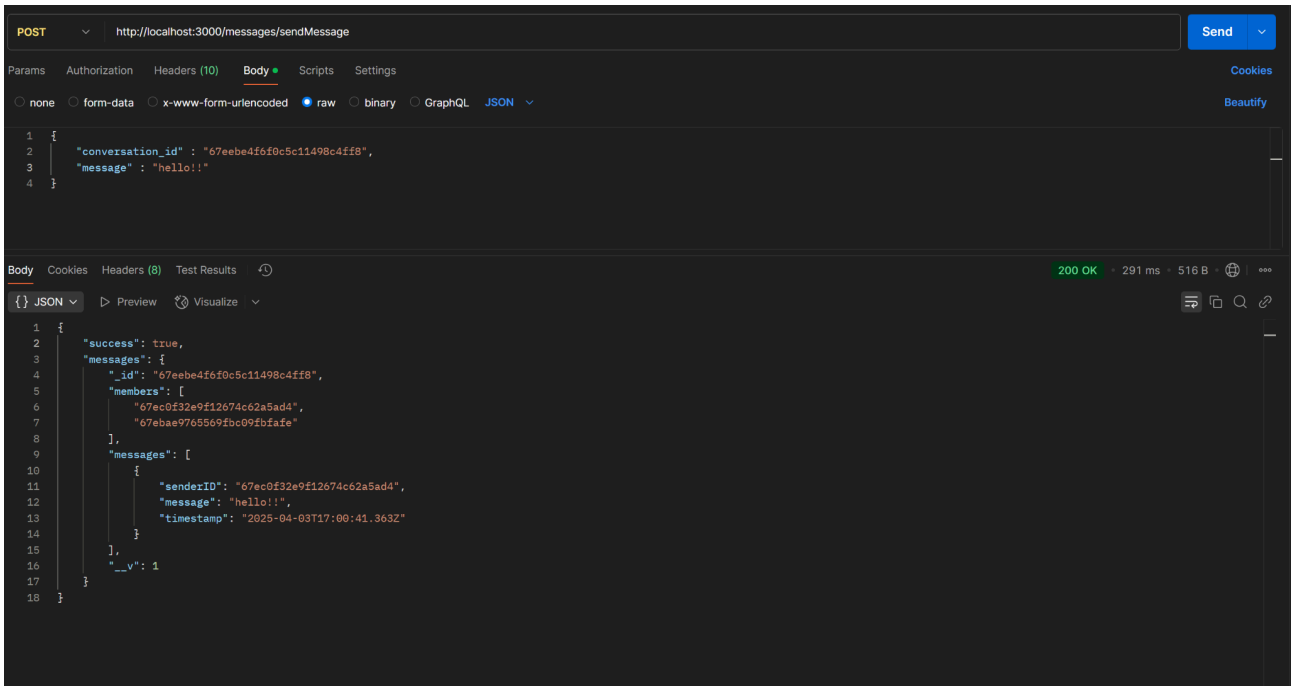
Test Owner: Bikramjeet Singh

Test Date: 03/04/2025

Test Results: The unit was successfully tested and the two users concerned were able to send and receive messages respectively.

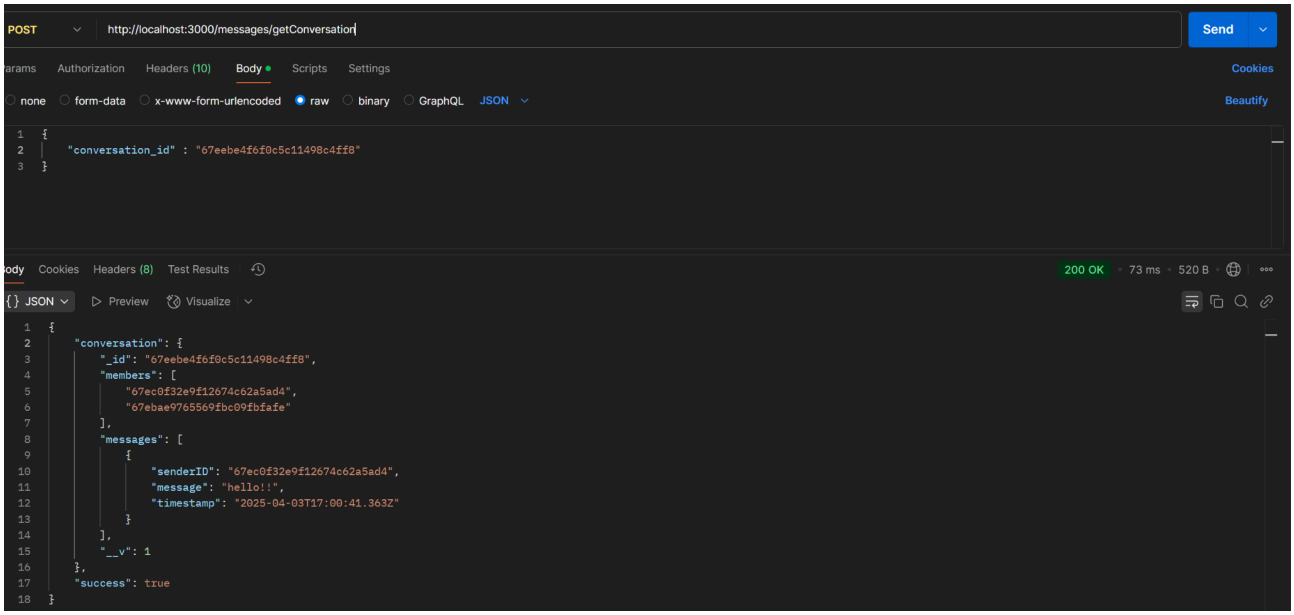


At first we need to create a conversation between to users if there does not exist any prior one else they won't be able to chat.

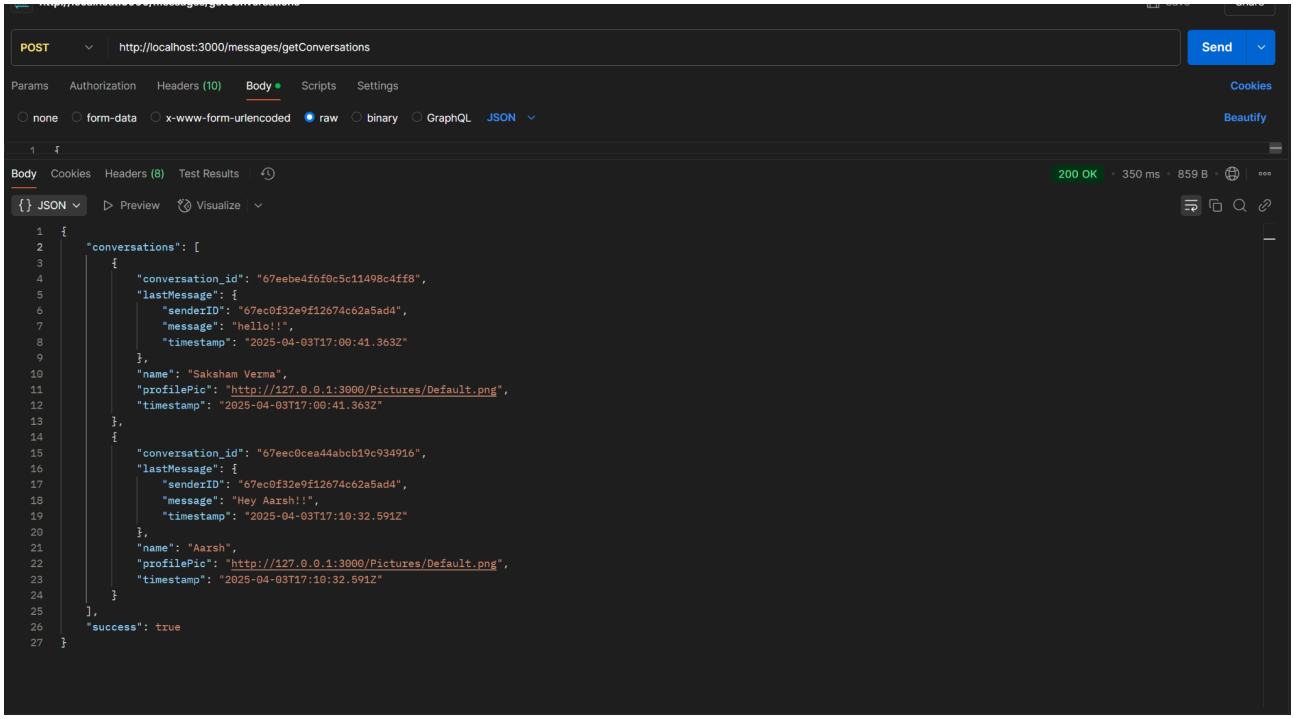


Now for sending the message we need to access the conversation_id created in the previous request which is stored permanently in the database, and also need to send the message in the body of the request.

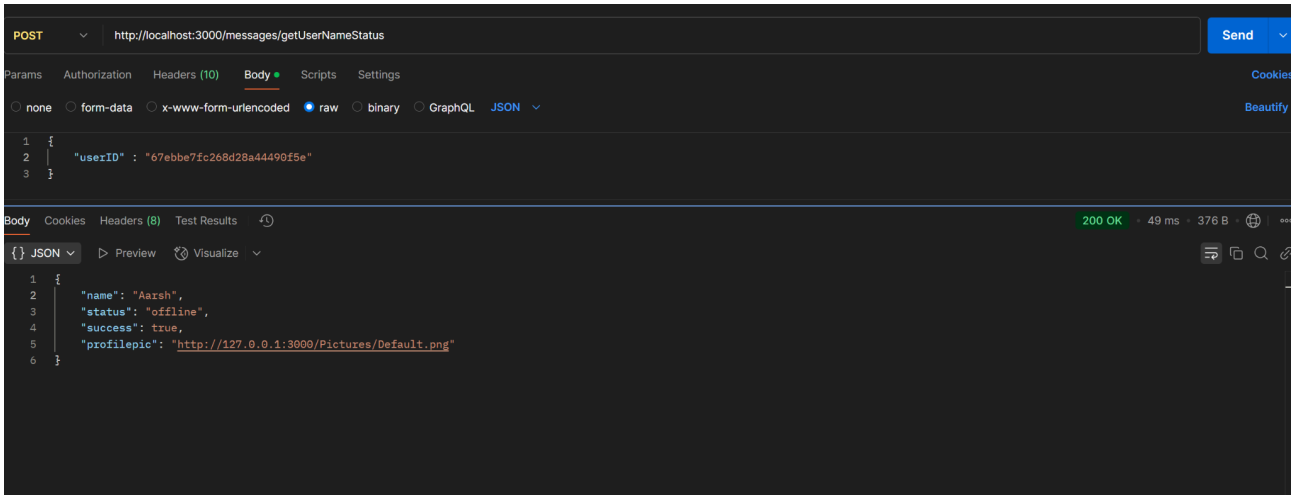
On success, we get the response as shown, along with timestamp and the other necessary details.



To fetch any conversation, we can do so by using the concerned route which gives the apt output as shown above.



To fetch all the conversations of a particular user, we can access the conversations attribute in the user model through the concerned route.



A user can also check if another user is offline or online at a certain moment.

All these tests indicate that the various routes concerned with messaging work appropriately producing correct results.

Structural Coverage: This testing takes into account the functional testing of the messaging route.
Additional Comments: None

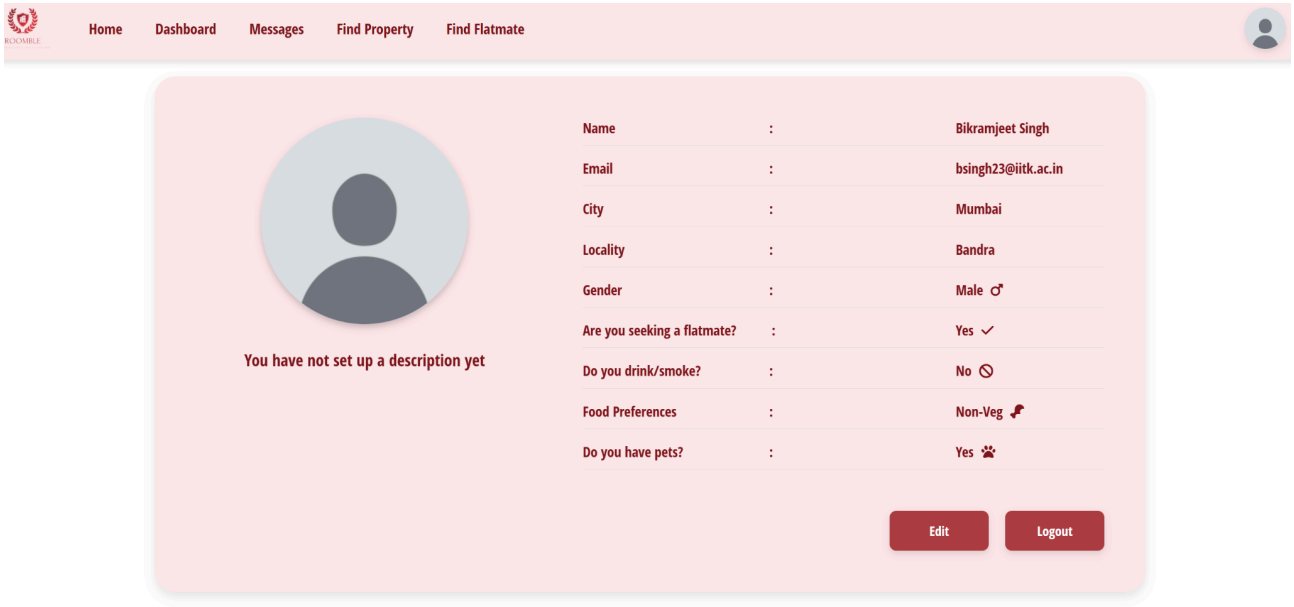
21. Logout- Frontend

This unit tests the correct working of the front-end functionality that allows the user to log out of the currently running session.

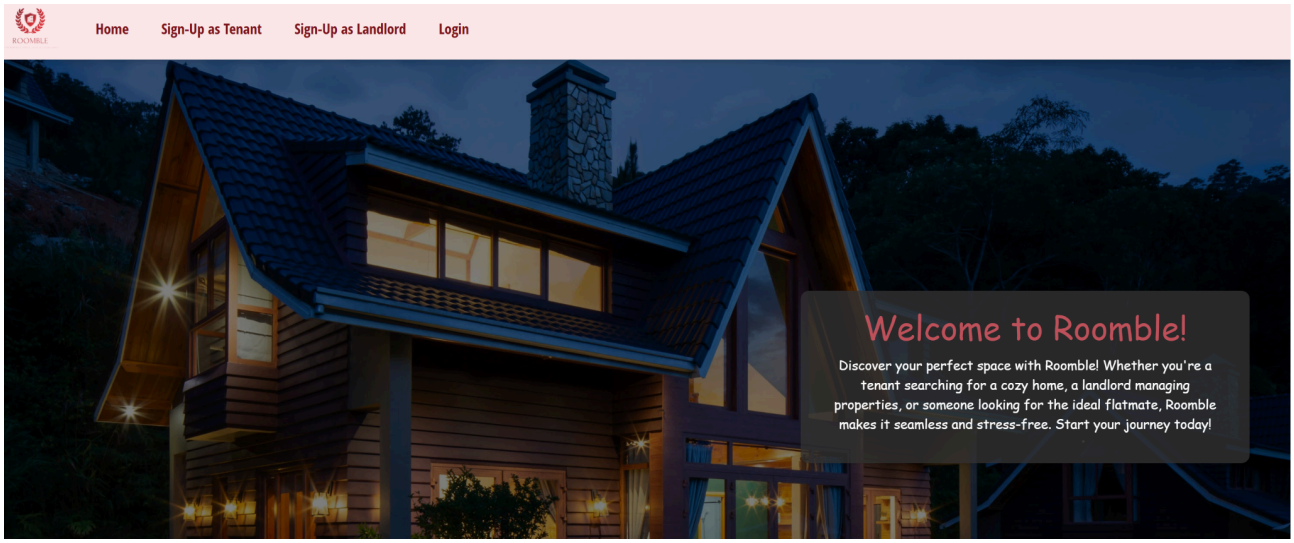
Unit Details: The correct working of the unit is essential so that the user can successfully logout which is an essential part of any web-application.

Test Owner : Bikramjeet Singh
Test Date: 03/04/2025

Test Results:
The unit works correctly since clicking it logs out a user successfully.



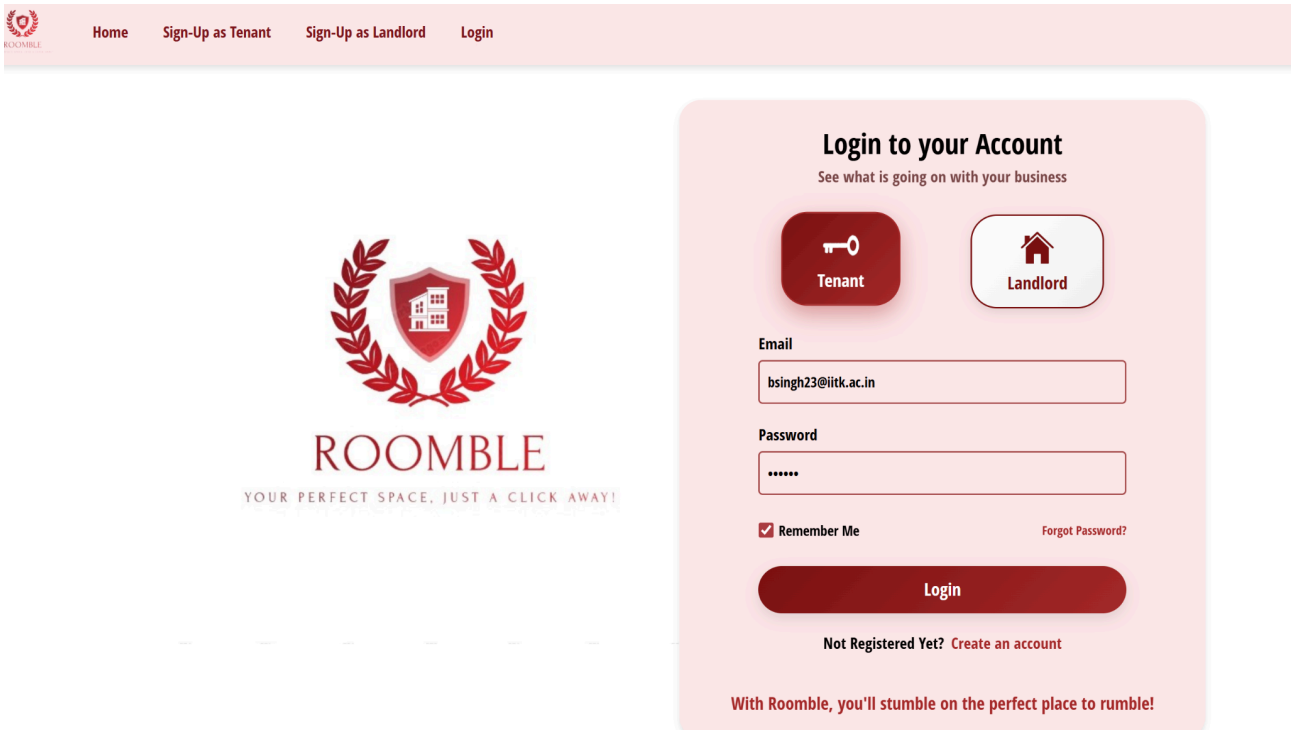
The logout button is available in the profile section of the user at the bottom right.



After clicking on logout the user is redirected to this page.

Structural Coverage: This testing covers appropriately the functionality of the logout button.

Additional Comments: Although "**Remember Me**" is a different functionality as such, it is also a highly related function to logout-login.

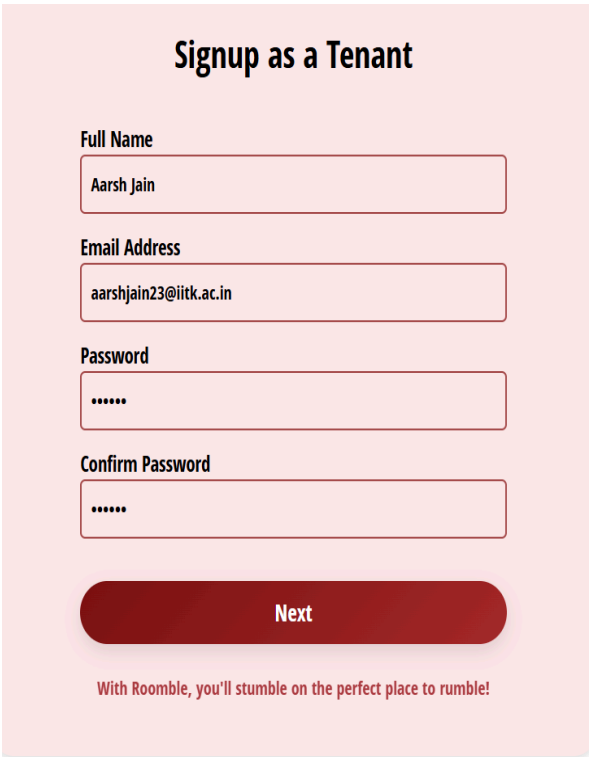


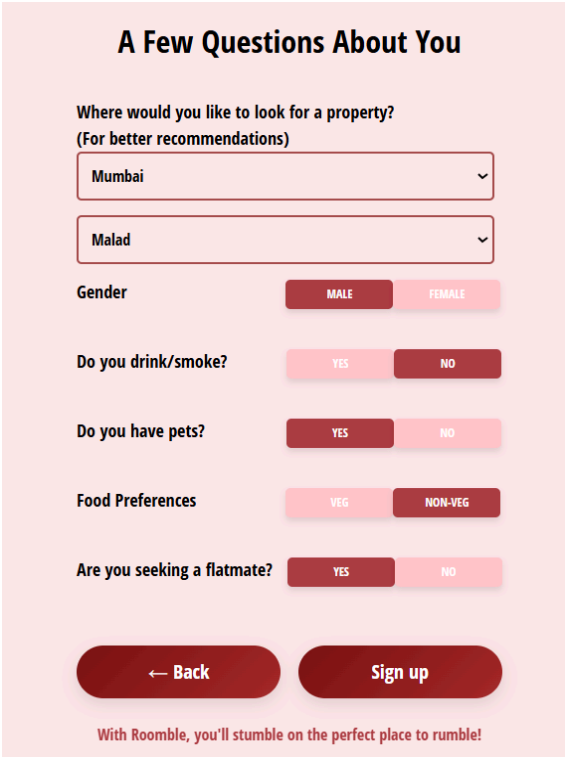
*The "**Remember Me**" button allows the user to store his/her credentials locally.*

3 Integration Testing

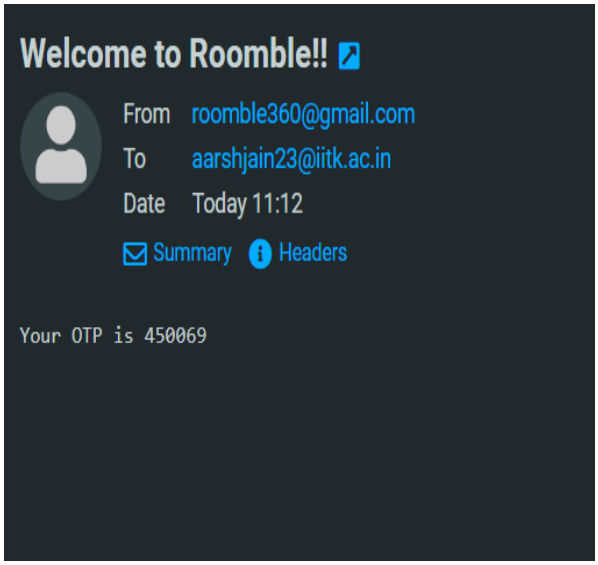
1. Sign Up Tenant

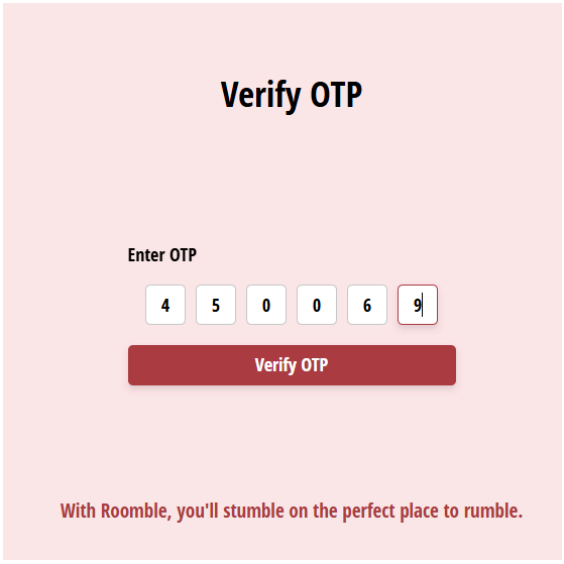
Module Details: This module tests the registration process for a new tenant. It verifies that upon successful sign up, all tenant details and choices are correctly stored in the backend database.
Test Owner: Aarsh Jain
Test Date: 01/04/2025



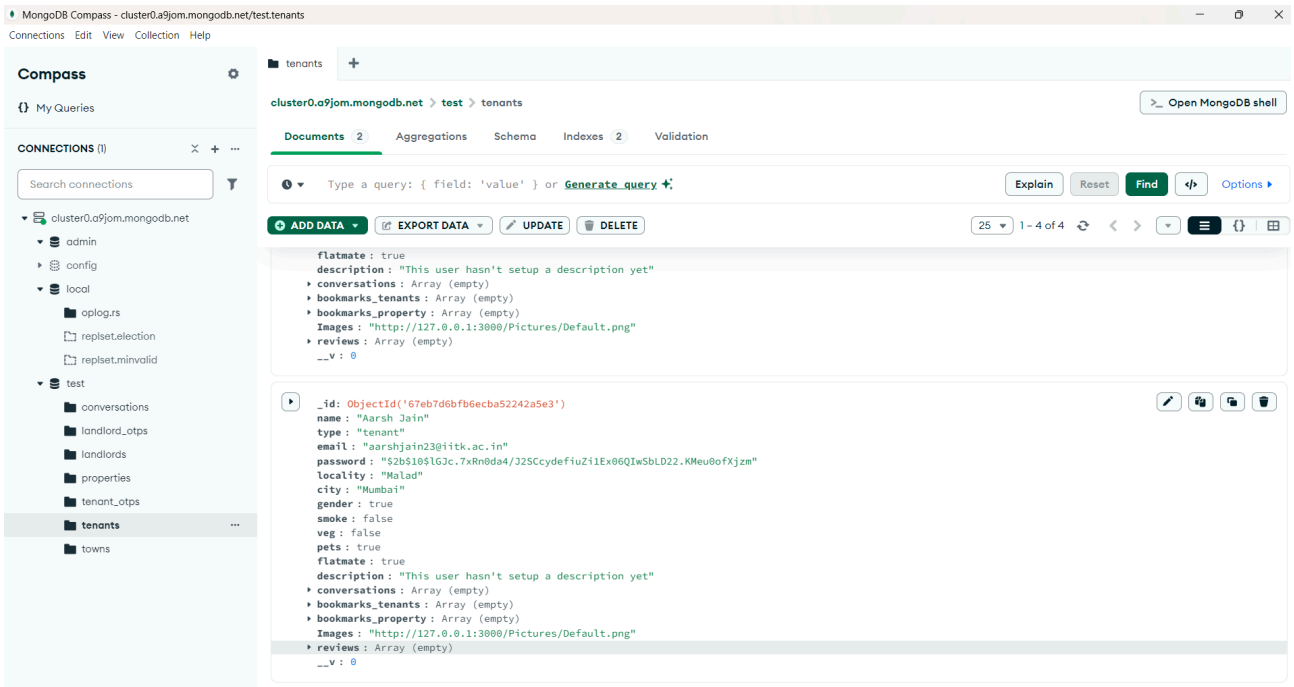


The user entered the following details, on clicking Sign-Up, the user will be redirected to the OTP page, where he has to enter the OTP sent to the registered email id.





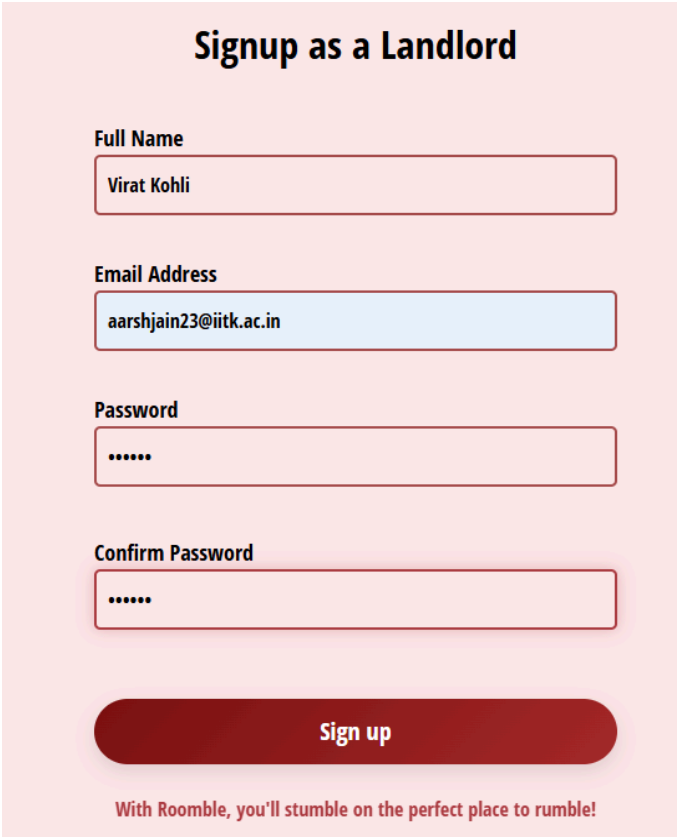
On entering the successful OTP received from the backend, the user should be redirected to the Login page, and details of the user should be stored in the backend.



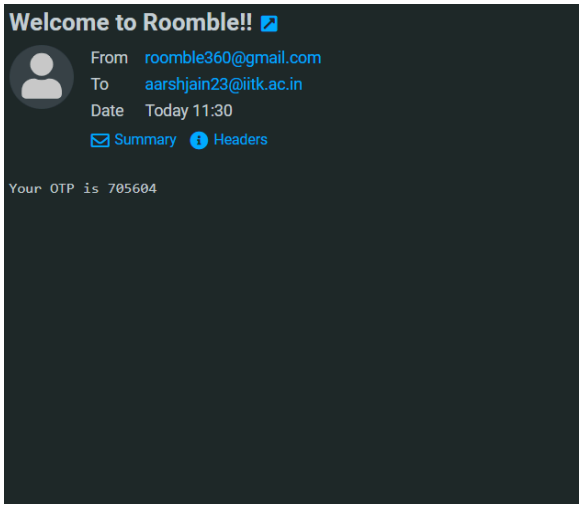
Test Results: The new tenant "Aarsh" was successfully registered, and his details were correctly saved in the backend user record.

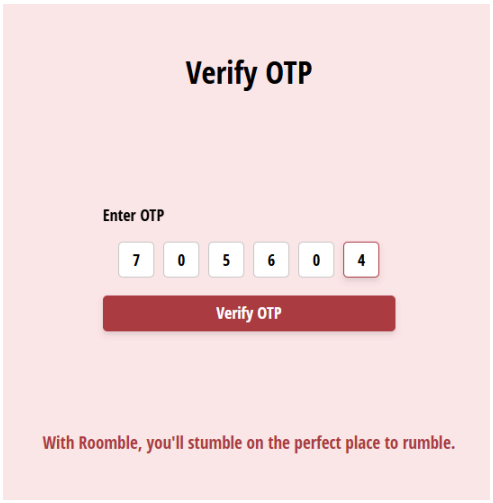
2. Sign Up Landlord

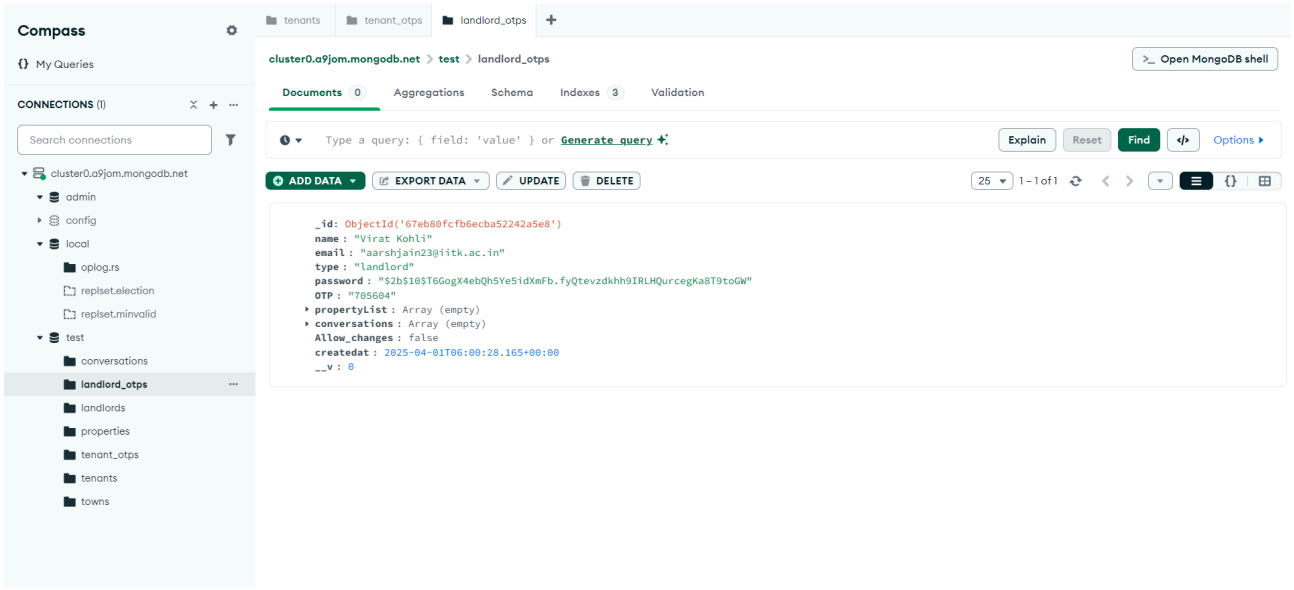
Module Details: This module tests the registration process for a new landlord. It verifies that upon successful sign up, all landlord details are correctly stored in the backend database.
Test Owner: Aarsh Jain
Test Date: 01/04/2025



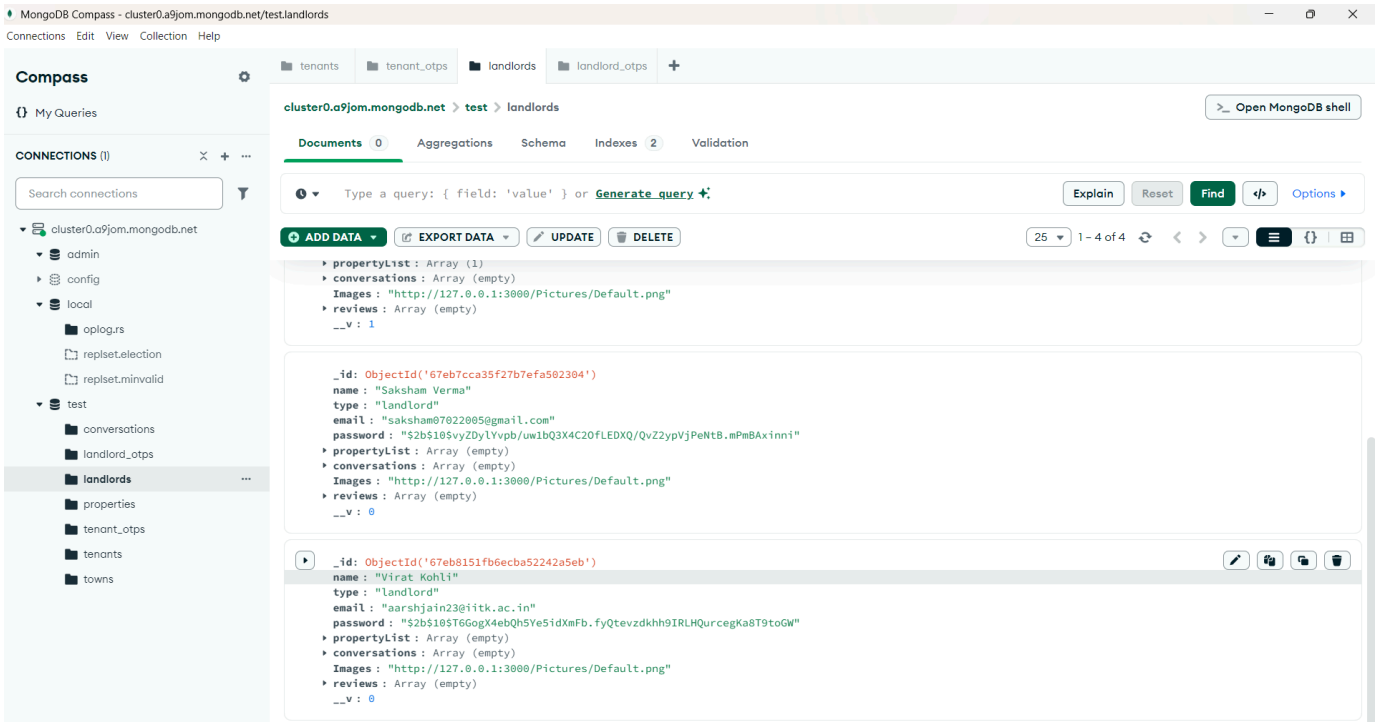
The user on clicking Sign Up gets redirected to the OTP page.







The OTP is successfully generated in the backend and mailed to the registered email ID, after entering the correct OTP user is redirected to the Login Page, and the User object with the correct User details is created in the backend



Test Results: The Landlord “Virat Kohli” is successfully registered.

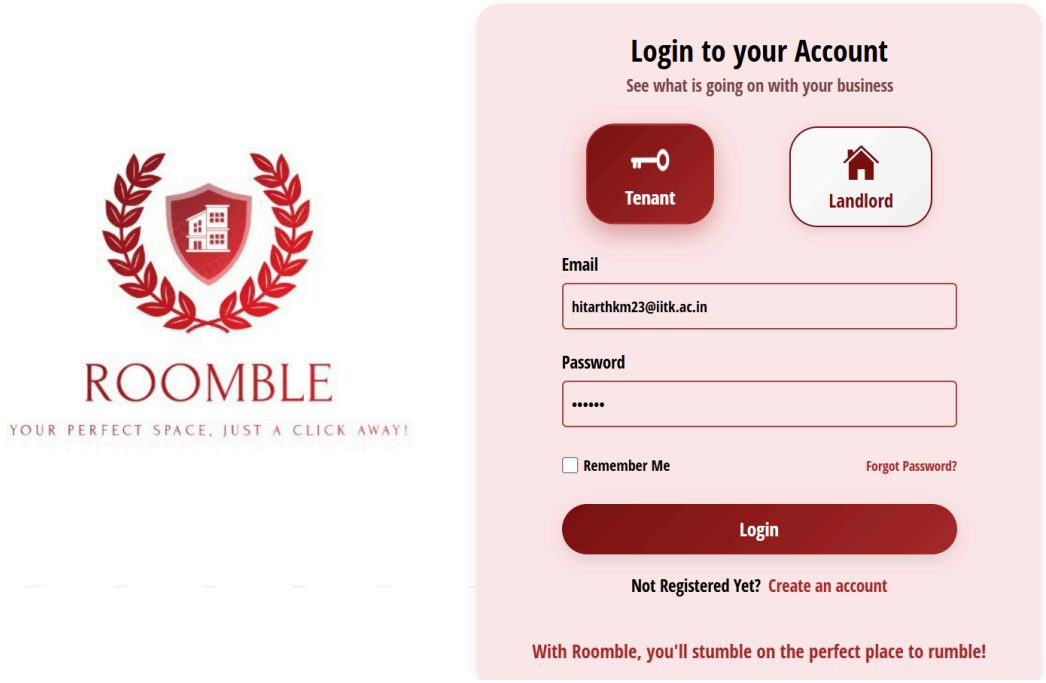
3. Login

Module Details: This module includes the tests for validating the details entered for login with the details stored in the database during signup and redirecting to respective user dashboards on successful validation.

Test Owner: Pranay

Test Date: 1/4/2025

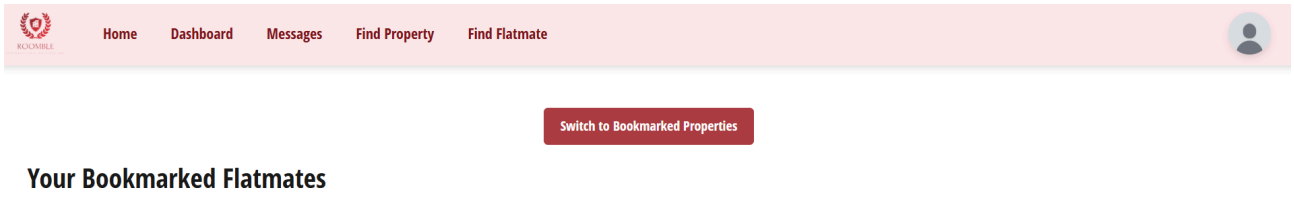
Test #1: Tenant Login



Tenant enters details that were entered during signup.

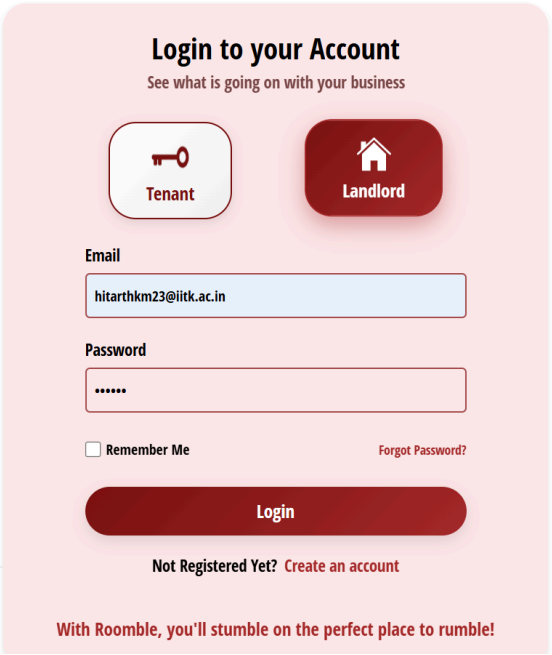

```
_id: ObjectId('67eb77a567dfab515f609f03')
name : "Hitarth Makawana"
type : "tenant"
email : "hitarthkm23@iitk.ac.in"
password : "$2b$10$TBbAwXLrw1qTD5i1Y3FKru27WrkEU8jWyC07/DyisNS3nA0CEtaNK"
locality : "Andheri"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : true
flatmate : true
description : "This user hasn't setup a description yet"
▸ conversations : Array (empty)
▸ bookmarks_tenants : Array (empty)
▸ bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
▸ reviews : Array (empty)
__v : 0
```


Data stored in MongoDB database to be validated with login form input, specifically email and password (hashed in MongoDB database for secure authentication).



The validation was successful, hence the tenant was redirected to tenant dashboard.

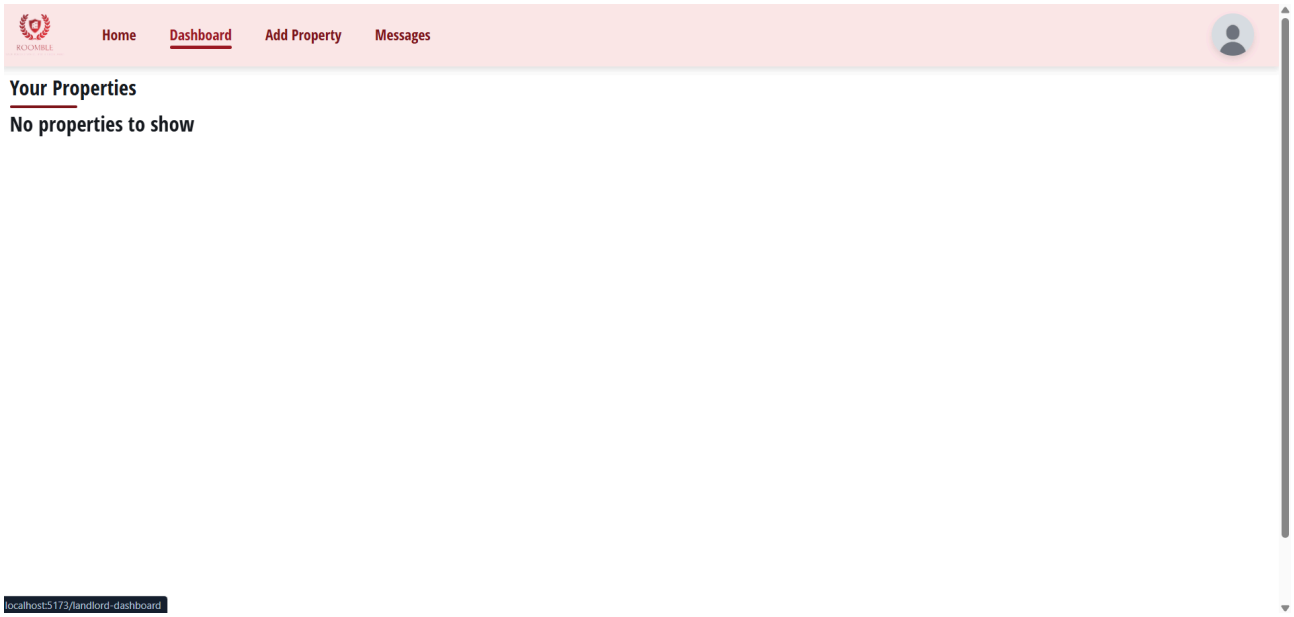
Test #2: Landlord Login



Landlord enters details that were entered during Signup

```
_id: ObjectId('67eb7aea67dfab515f609f1a')
name : "Hitarth Makawana"
type : "landlord"
email : "hitarthkm23@iitk.ac.in"
password : "$2b$10$TaWuoErY4yaajih/SQRDzulZ25niOgfHqQ9CK3FfLFMSagi3Rbupi"
propertyList : Array (empty)
conversations : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
reviews : Array (empty)
__v : 0
```


Data stored in MongoDB database to be validated with login form input, specifically email and password (hashed in MongoDB database for secure authentication).



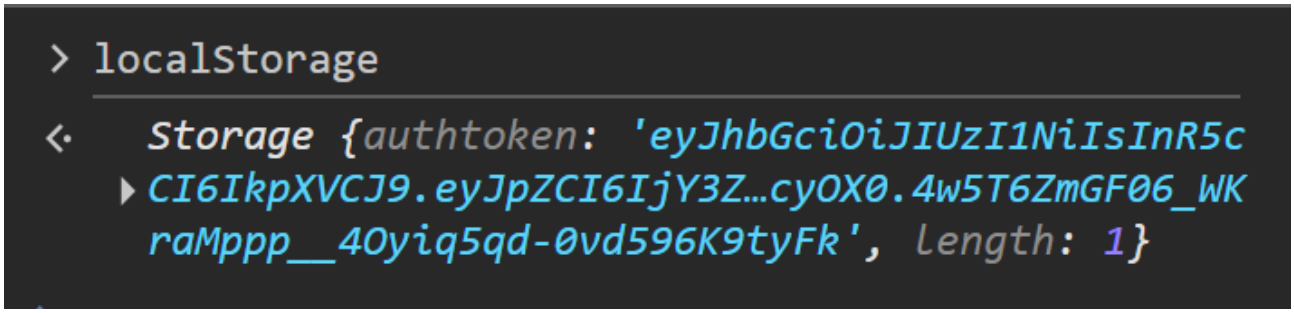
The validation was successful, hence the landlord was redirected to landlord dashboard.

Test Results: Both the login pages i.e. for tenant and landlord were successfully integrated with the backend hence the data entered in the login form was successfully validated with the backend database and both the tenant and landlord users were redirected to their respective dashboards after submitting the login form. The procedure flow is as follows:
Data entered in login form -> Data validated with MongoDB database -> User redirected to respective dashboard pages upon successful validation.
Additional Comments: The password shown in the MongoDB database is hashed in order to ensure secure authentication, however, the user needs to input the correct password to login.

4. Authtoken Login

Test #1: Tenant

Whenever a user logs into his account, he receives an auth token, which can be used to log him in again if he does so within 5 hours.

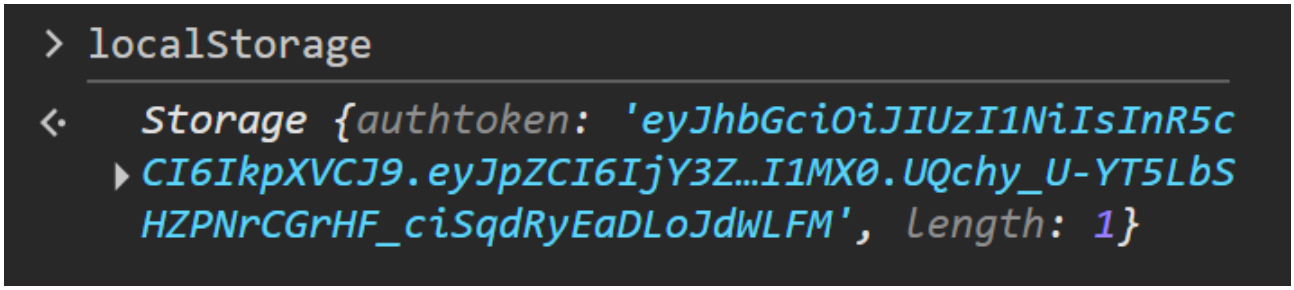


The auth token was stored successfully.

Successful login to the account on reload.

Test #2: Landlord

Whenever a landlord logs into his account, he receives an auth token, which can be used to log him in again if he does so within 5 hours.



The auth token was stored successfully.

Successful login to account on reload.

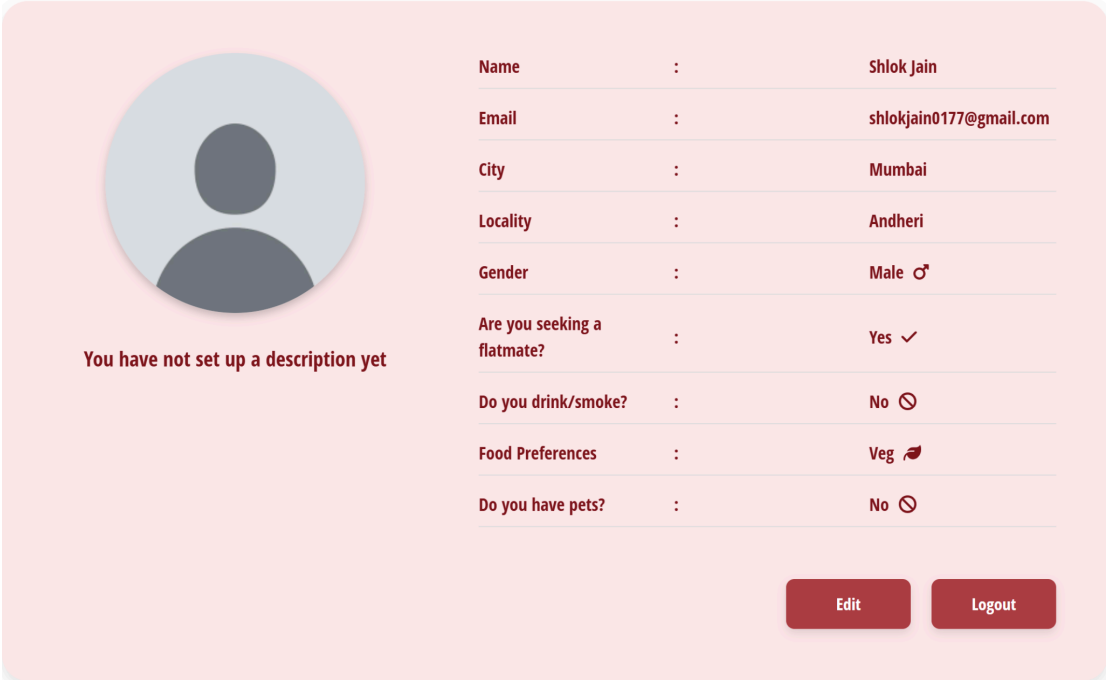
Test Owner: Shlok Jain
Test Date: 1/4/2025
Test Results: Feature working as expected

5. View Self Profile

Module Details: The feature to view self-profile can be accessed at /tenant-profile-page for the Tenant account. The details are consistent across backend and frontend.

Test #1: Tenant


```
_id: ObjectId('67eb6f65f35d79d0a99ccba3')
name : "Shlok Jain"
type : "tenant"
email : "shlokjain0177@gmail.com"
password : "$2b$10$sMRHLIW3rqCAlXeFlDwtW.QtwIUbMS778qhidfOt1N3EbYmOYPInC"
locality : "Andheri"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : false
flatmate : true
description : "This user hasn't setup a description yet"
▸ conversations : Array (empty)
▸ bookmarks_tenants : Array (empty)
▸ bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
▸ reviews : Array (empty)
__v : 0
```

Test #2: Landlord

For landlords, self-profile can be viewed at /landlord-profile-page. The details are consistent across backend and frontend.

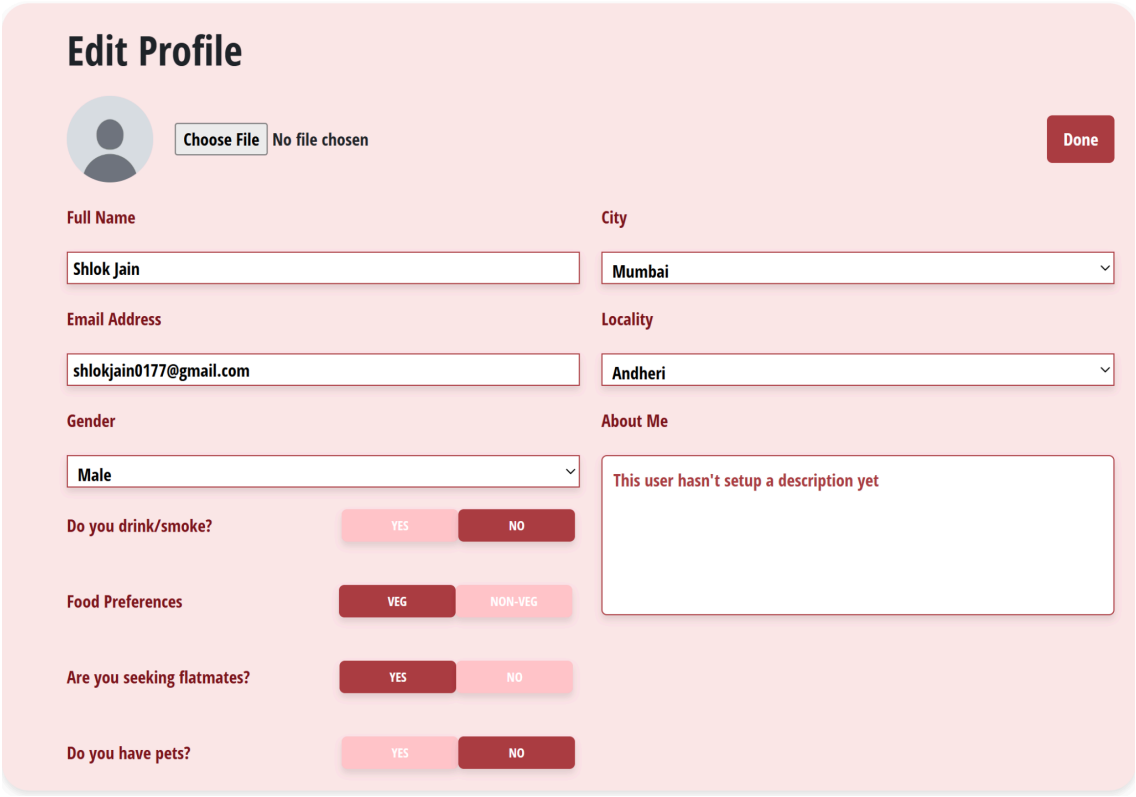
```
_id: ObjectId('67eb7acef35d79d0a99ccbd6')
name : "Shlok Jain"
type : "landlord"
email : "shlokjain0177@gmail.com"
password : "$2b$10$YNLFsAzhBIkQ393PD27P7u2V0v1z1Z.eZ.KnonYhbxP6P7ts/YzOO"
▸ propertyList : Array (empty)
▸ conversations : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
▸ reviews : Array (empty)
__v : 0
```


Test Owner: Shlok Jain
Test Date: 1/4/2025
Test Results: Working as expected

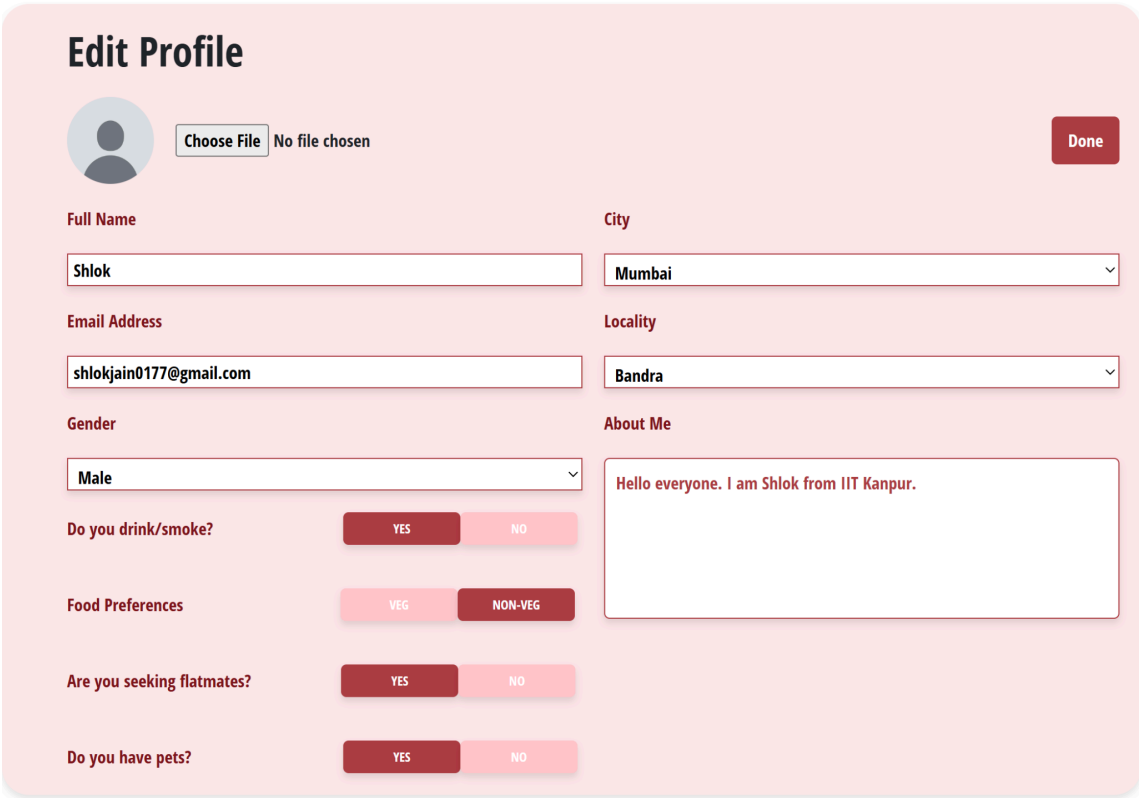
6. Edit Profile

Module Details: The tenant profile edit page can be used to edit the tenant's details and can be accessed at /tenant-edit-page.

Test #1: Tenant



The initial details are loading correctly.



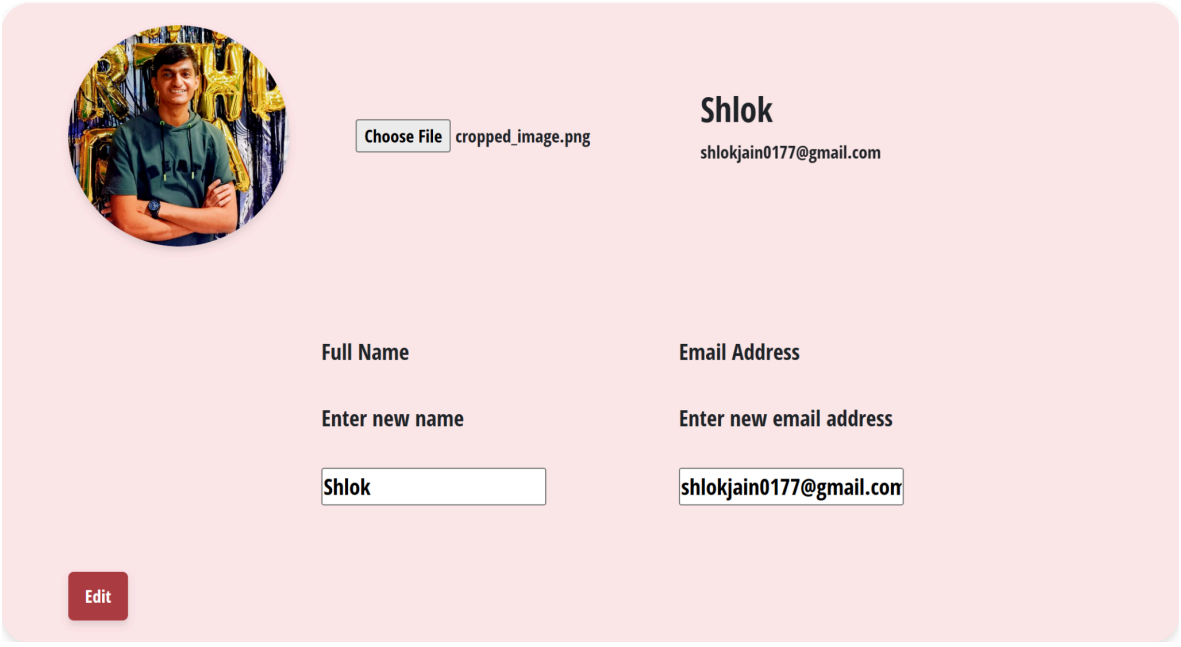
Now, if I change some details as shown and click the Done button, the details are sent to the backend API, which updates it accordingly. And the user is redirected to the profile page.

It is correctly updated in the backend, as shown.

Test #2: Landlord

The landlord profile edit page can be accessed at /landlord-edit-page.

The initial details are loading correctly.



Now, I will change the profile photo and name as shown. If I click the edit button, the details will be sent to the backend and updated accordingly.

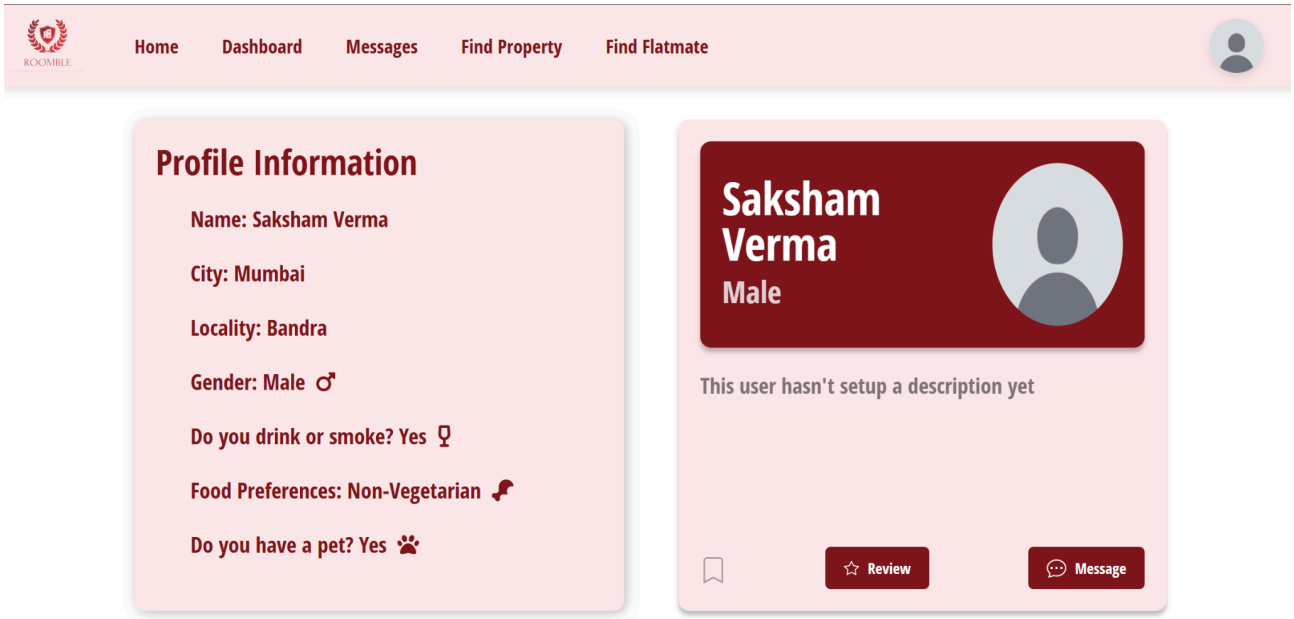
```
_id: ObjectId('67eb7acef35d79d0a99ccbd6')
name : "Shlok"
type : "landlord"
email : "shlokjain0177@gmail.com"
password : "$2b$10$YNLFsAzhBIkQ393PD27P7u2V0v1z1Z.eZ.KnonYhbxP6P7ts/YzOO"
▸ propertyList : Array (empty)
▸ conversations : Array (empty)
Images : "http://localhost:3000/Pictures/landlord/67eb7acef35d79d0a99ccbd6.png"
▸ reviews : Array (empty)
__v : 0
```


Also, the updated image is stored in the backend server folder of /Pictures/landlord/{id}.png, as shown.

Test Owner: Shlok Jain
Test Date: 1/4/2025
Test Results: The landlord and tenant's edit feature is working correctly.

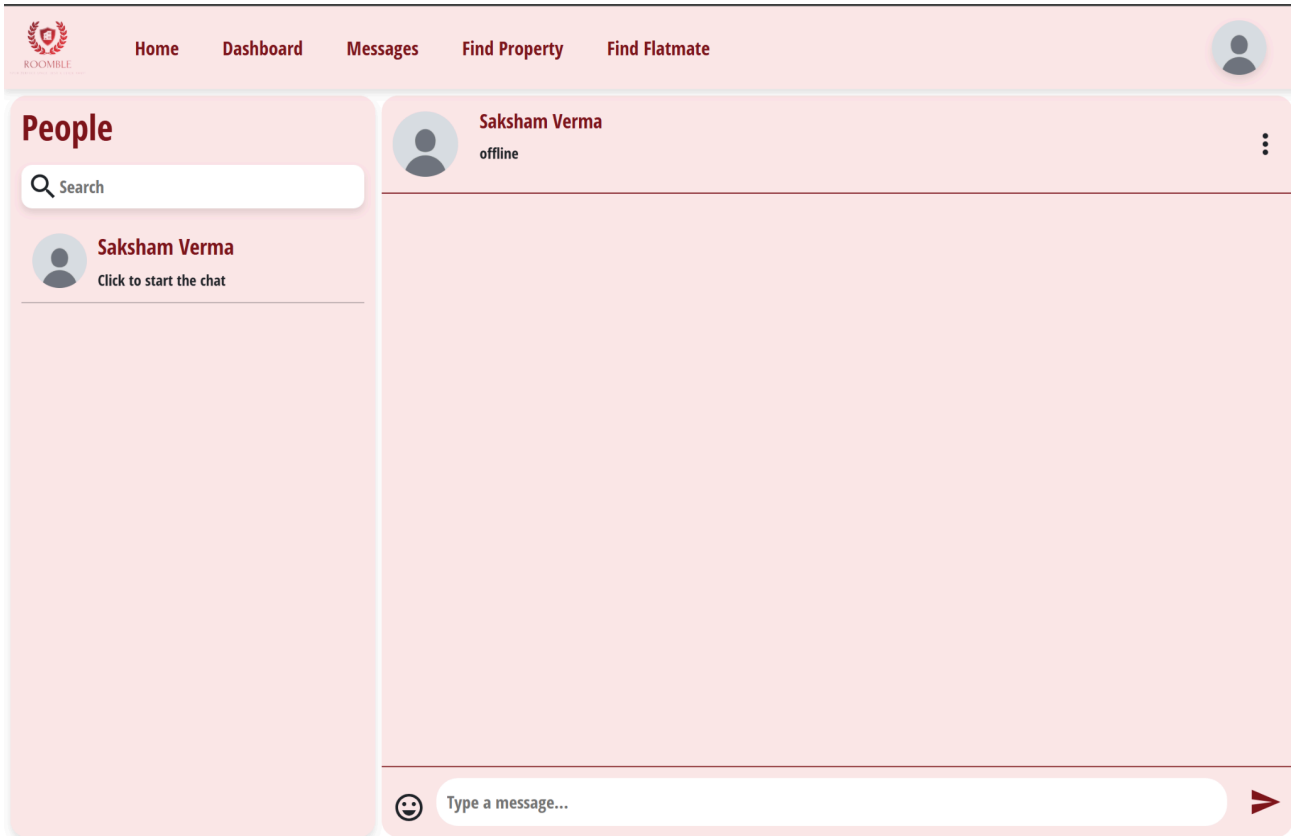
7. Messaging

Module Details: One of the important features of our app is the real-time chat feature between any two users. We store the conversation history between pairs of users in the conversation model. If you want to chat with another user, click the message button on their profile. This will create a new conversation and start appearing on the Messages page.

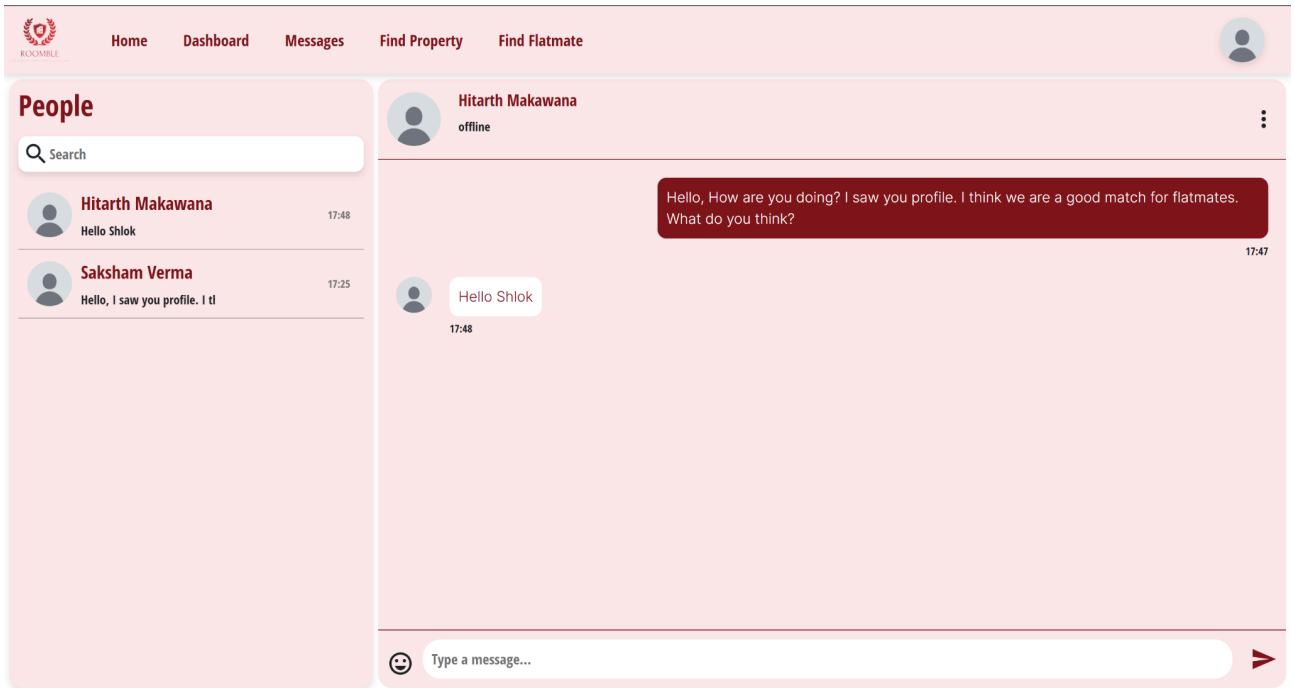


This creates a new element in the Conversations database and redirects the user to the chat page, as shown.


```
  _id: ObjectId('67ebb708a115332bbd4598aa')
▼ members : Array (2)
    0: ObjectId('67ebb688a115332bbd459875')
    1: ObjectId('67ebae9765569fbc09fbfafe')
▶ messages : Array (empty)
  __v : 0
```



The real-time messaging feature is also working correctly, the online/offline status is handled and updated correctly using web sockets (socket.io)

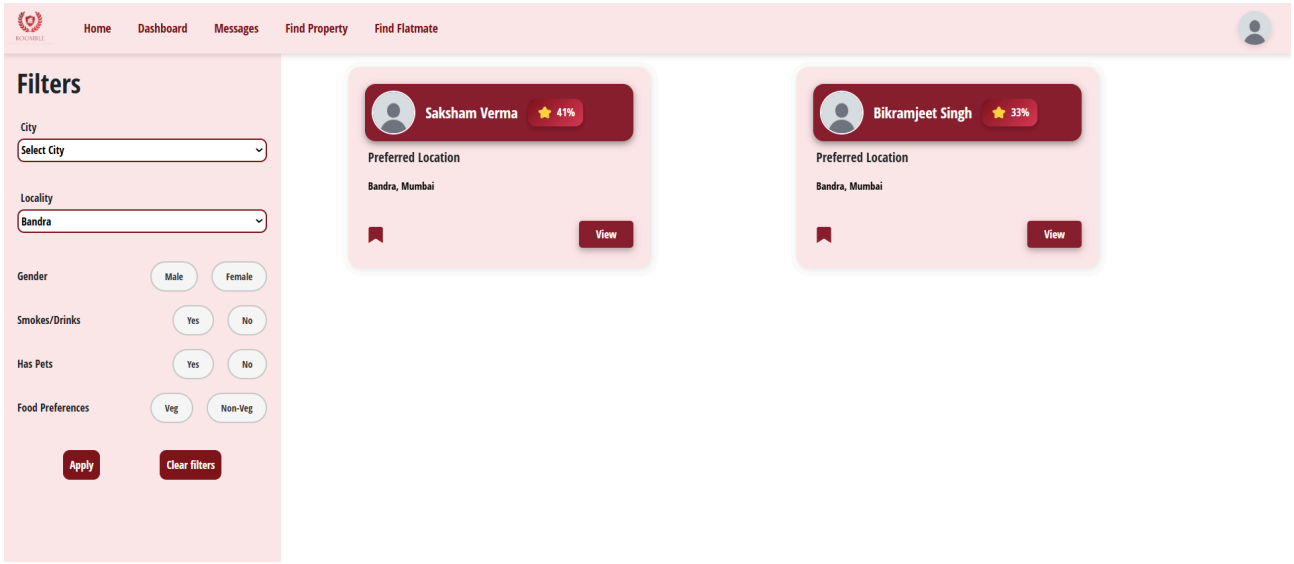


Test Owner: Shlok Jain
Test Date: 1/4/2025
Test Results: The full messaging pipeline works correctly with all its features. The message button on the profile can be used to start a conversation with another user. Then, WebSockets are being used for real-time communication and updating users' online/offline status, which are also tested and working as expected.

8. Find Flatmates

Module Details: The correctly filtered and sorted search results from the backend are successfully shown in the frontend
Test Owner: Aarsh Jain
Test Date: 01/04/2025

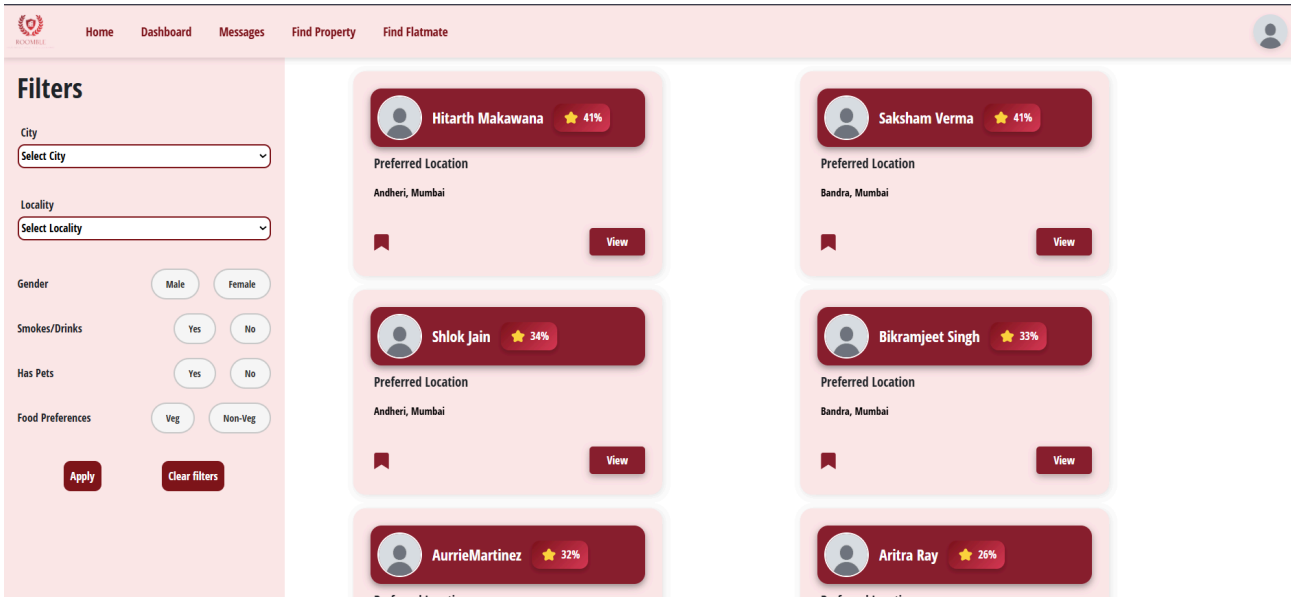
Test #1



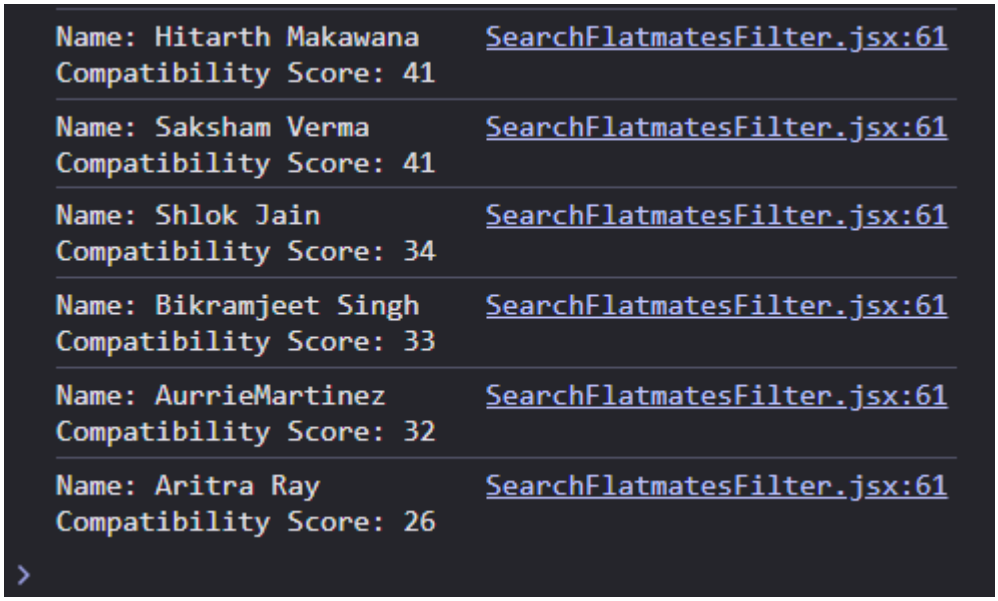
The search results are filtered to show only the flatmates with location preference "Bandra".

The backend sends the filtered data correctly and in sorted order of the compatibility score, which the frontend displays accurately.

Test #2



On clicking clear filters, the filters go back to default and we can see the non filtered search results correctly.



Checking with the backend we get the following sorted data as expected.

Test Results: The Find Flatmates Page is working correctly without any bugs.

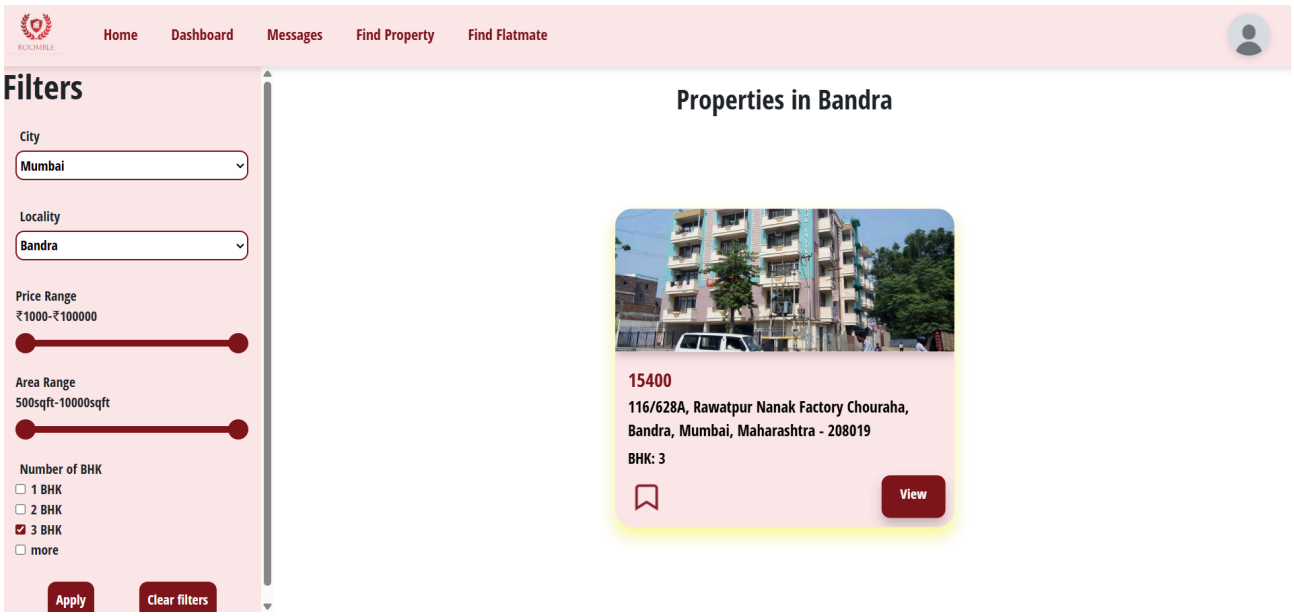
9. Adding/Removing Bookmarks

Module Details: This module includes the tests for adding and removing bookmarks for the properties and flatmates the tenant is interested in. The bookmarks are added to the MongoDB database and backend in an array entitled bookmarks_tenants and bookmarks_property.

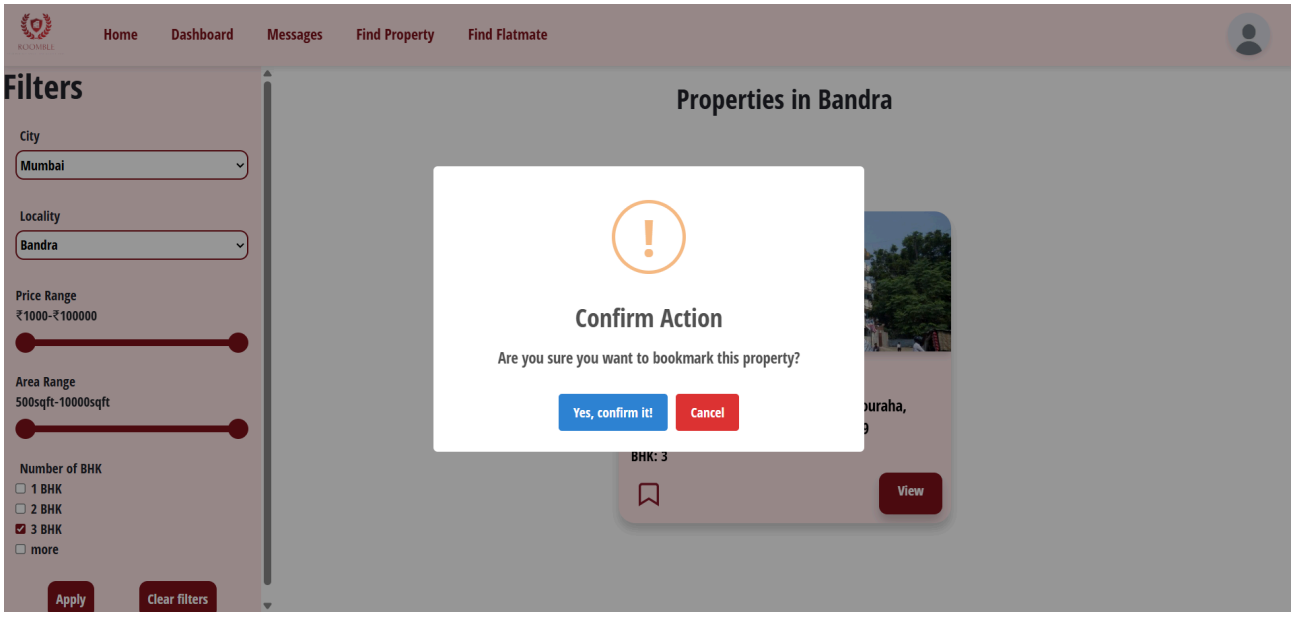
Test Owner: Hitarth Makawana

Test Date: 01/04/2025

Test #1: Testing add/delete bookmarks for properties



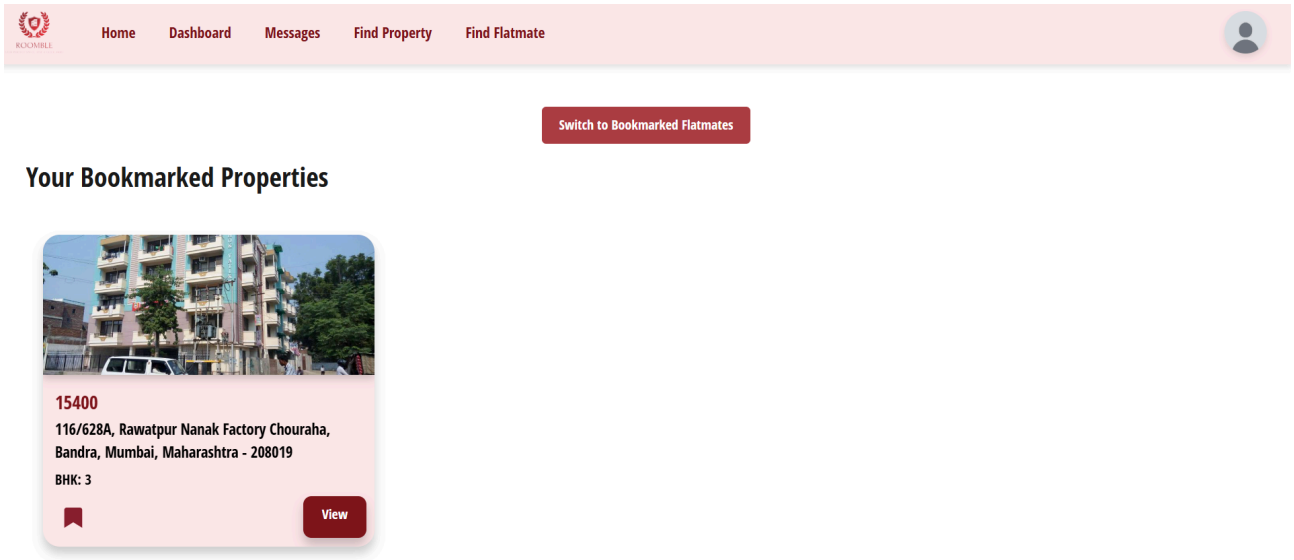
Property searched using Find Property and clicked on the bookmark icon for the interested property.



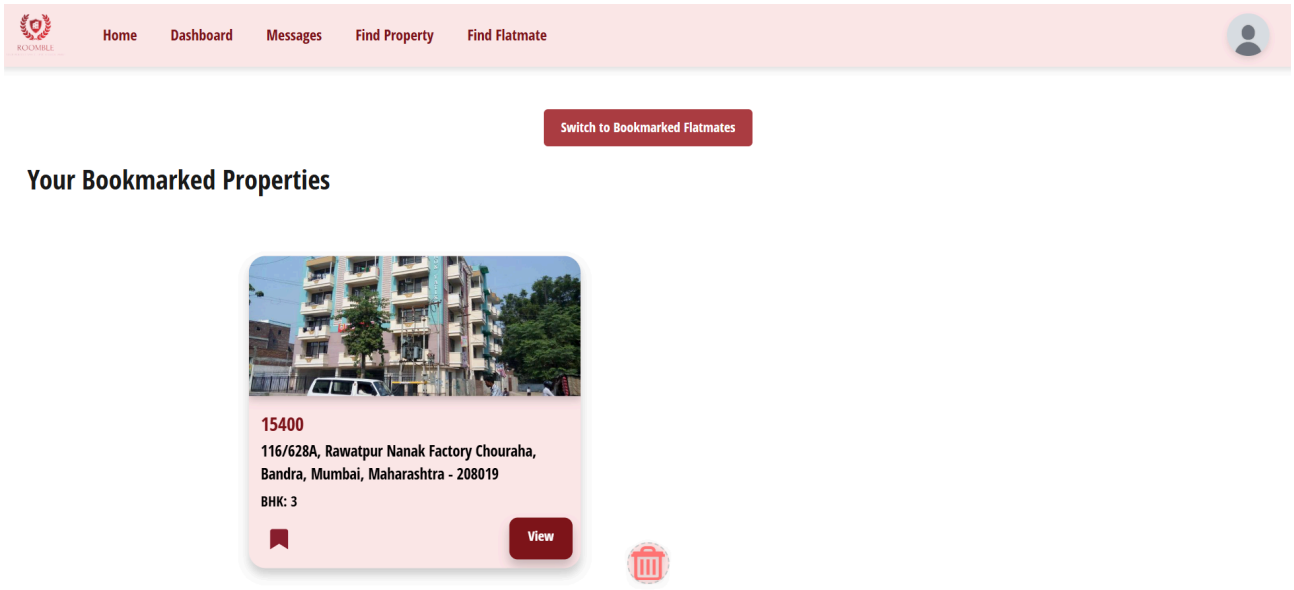
The tenant is prompted with a popup asking for confirmation for the bookmark to be added, the tenant clicks on "Yes, confirm it!" and the bookmark is showed on the tenant dashboard as well as the backend database is updated.


```
_id: ObjectId('67eb77a567dfab515f609f03')
name : "Hitarth Makawana"
type : "tenant"
email : "hitarthkm23@iitk.ac.in"
password : "$2b$10$TBbAwXLrw1qTD5i1Y3FKru27WrkEU8jWyC07/DyisNS3nA0CEtaNK"
locality : "Andheri"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : true
flatmate : true
description : "This user hasn't setup a description yet"
conversations : Array (empty)
bookmarks_tenants : Array (1)
  0: "67eb834d39dab323e423ab0f"
bookmarks_property : Array (1)
  0: "67eb81fa67dfab515f609f5c"
Images : "http://127.0.0.1:3000/Pictures/Default.png"
reviews : Array (empty)
__v : 5
```

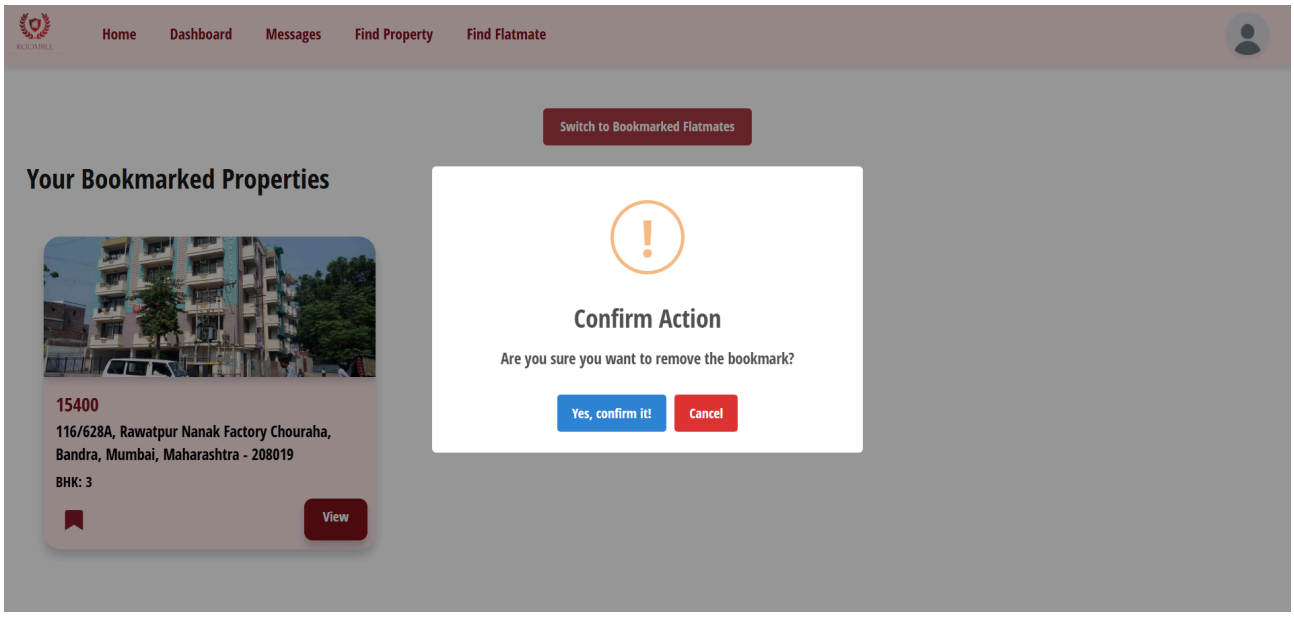

The MongoDB database is updated, the property id is added to the bookmarks_property array.



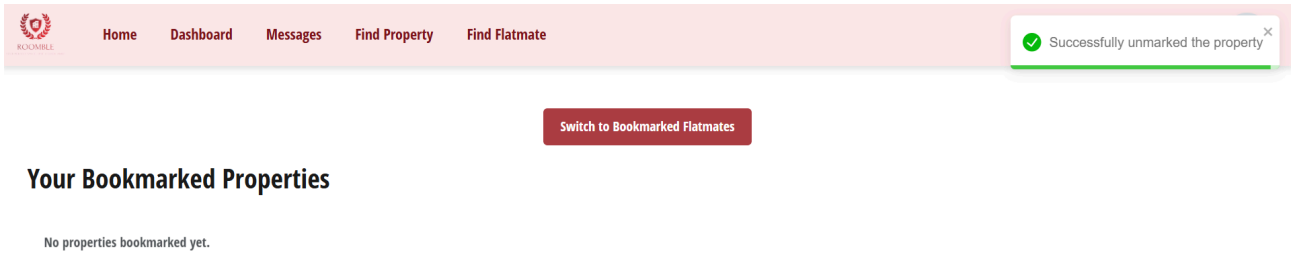
The Property is bookmarked and is visible on the tenant dashboard under the Bookmarked Properties heading.



The tenant can drag and drop the bookmarked property to the bin icon to remove the bookmark.

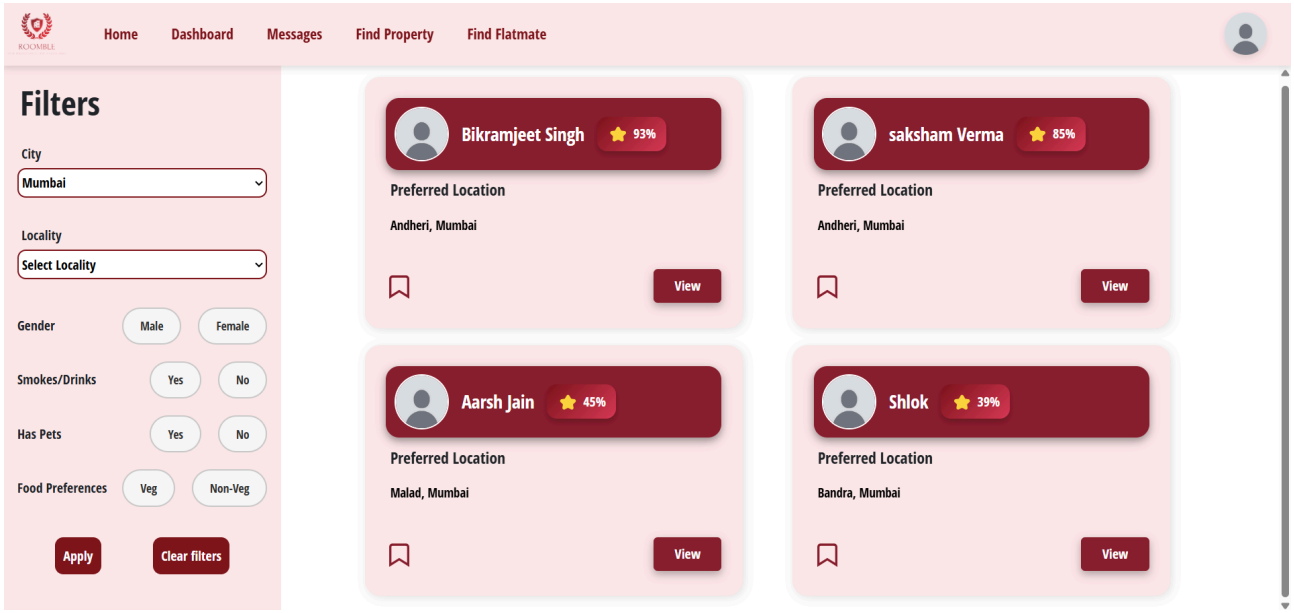


The tenant is prompted with the confirmation message for the bookmark to be removed. The tenant clicks on "Yes, confirm it!" and the bookmark is removed, the dashboard is updated and the property is no longer visible, the backend is updated too.

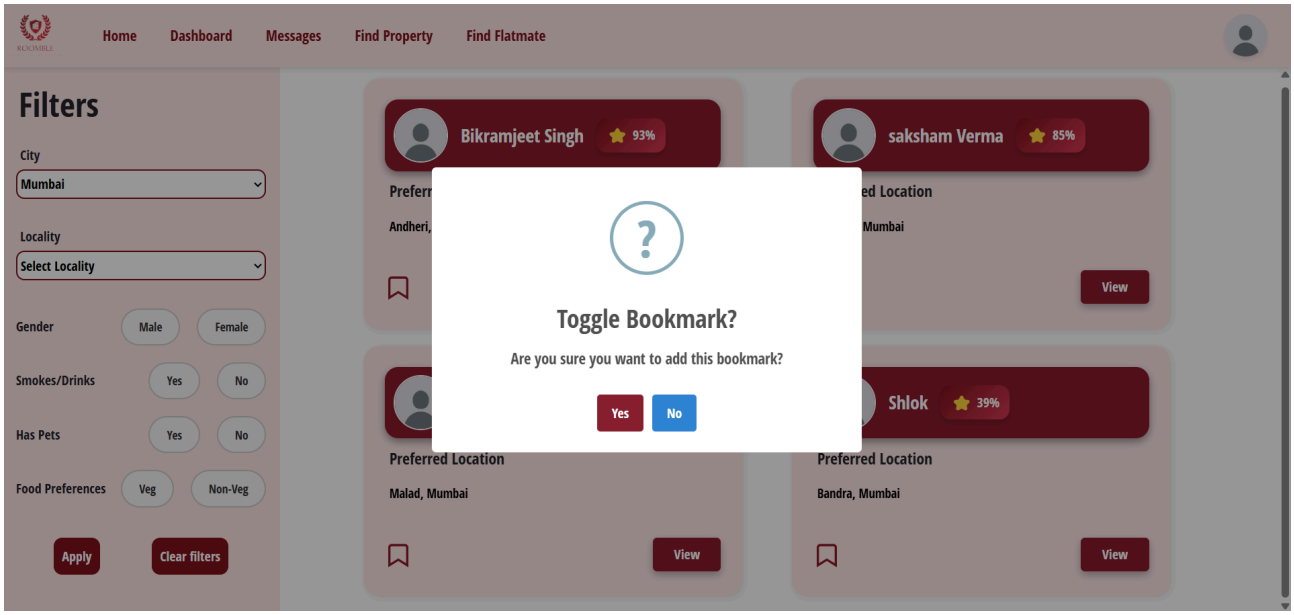


The user is prompted with the notification that the bookmark is unmarked successfully.

Test #2: Testing add/delete bookmarks for flatmates



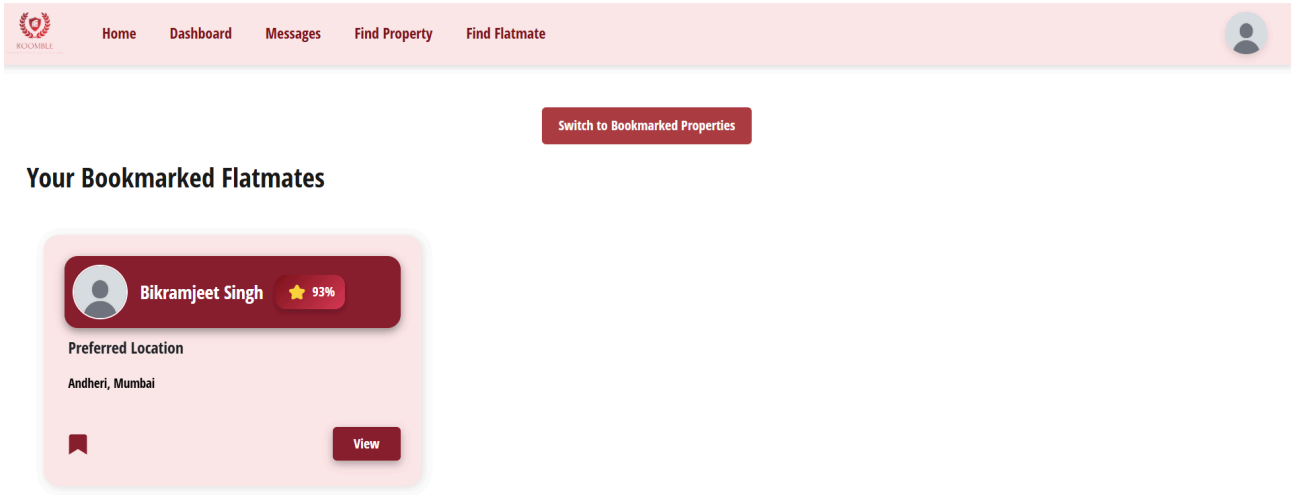
Flatmates searched using Find Flatmate, and clicked on the bookmark icon for the interested flatmate.



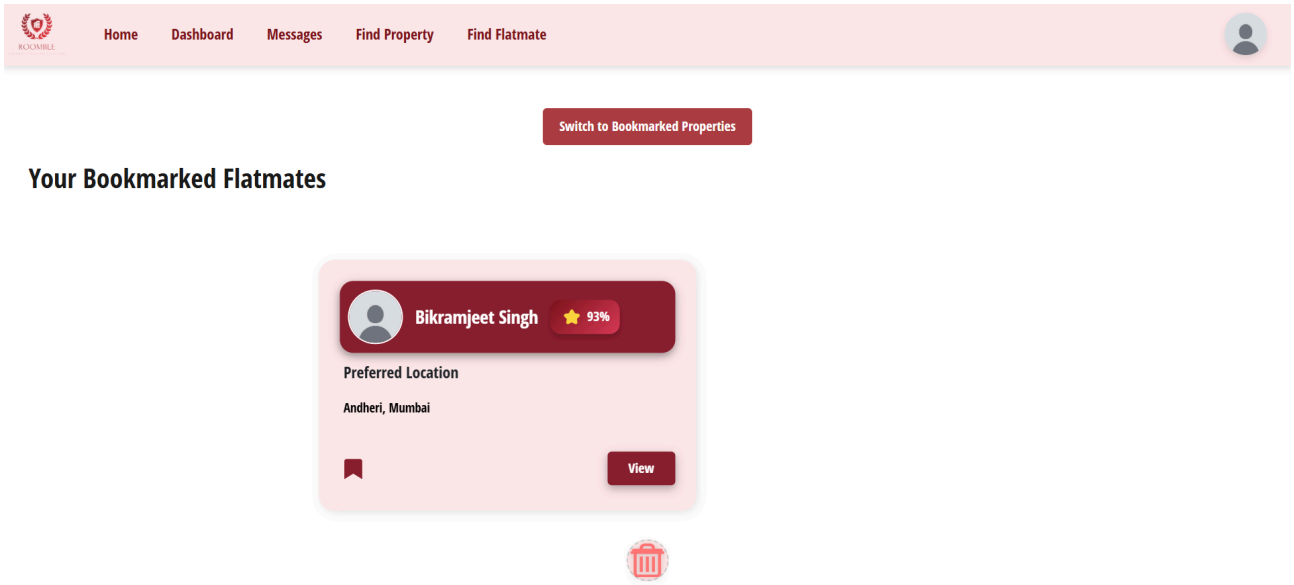
The tenant is prompted with a popup asking for confirmation for the bookmark to be added, the tenant clicks on “Yes, confirm it!” and the bookmark is showed on the tenant dashboard as well as the backend database is updated.

```
_id: ObjectId('67eb77a567dfab515f609f03')
name : "Hitarth Makawana"
type : "tenant"
email : "hitarthkm23@iitk.ac.in"
password : "$2b$10$TBbAwXLrw1qTD5i1Y3FKru27WrkEU8jWyC07/DyisNS3nA0CEtaNK"
locality : "Andheri"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : true
flatmate : true
description : "This user hasn't setup a description yet"
conversations : Array (empty)
bookmarks_tenants : Array (1)
  0: "67eb834d39dab323e423ab0f"
bookmarks_property : Array (1)
  0: "67eb81fa67dfab515f609f5c"
Images : "http://127.0.0.1:3000/Pictures/Default.png"
reviews : Array (empty)
__v : 5
```

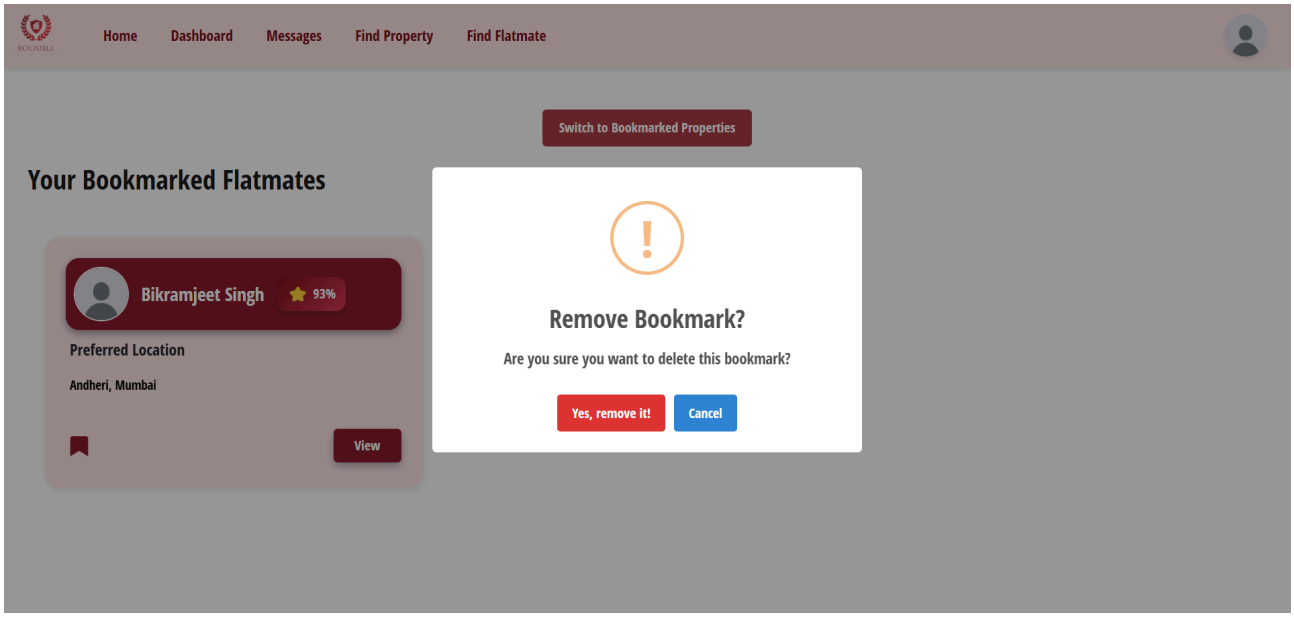

The MongoDB database is updated, the flatmate id is added to the bookmarks_tenants array.



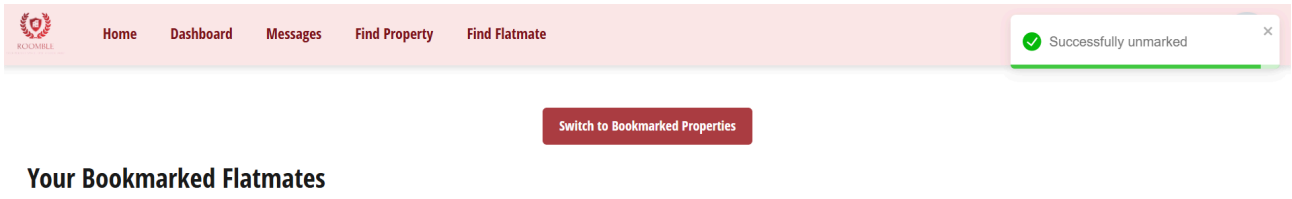
The Flatmate is bookmarked and is visible on the tenant dashboard under the Bookmarked Flatmates heading.



The tenant can drag and drop the bookmarked flatmate to the bin icon to remove the bookmark.



The tenant is prompted with the confirmation message for the bookmark to be removed. The tenant clicks on "Yes, confirm it!" and the bookmark is removed, the dashboard is updated and the flatmate is no longer visible, the backend is updated too.



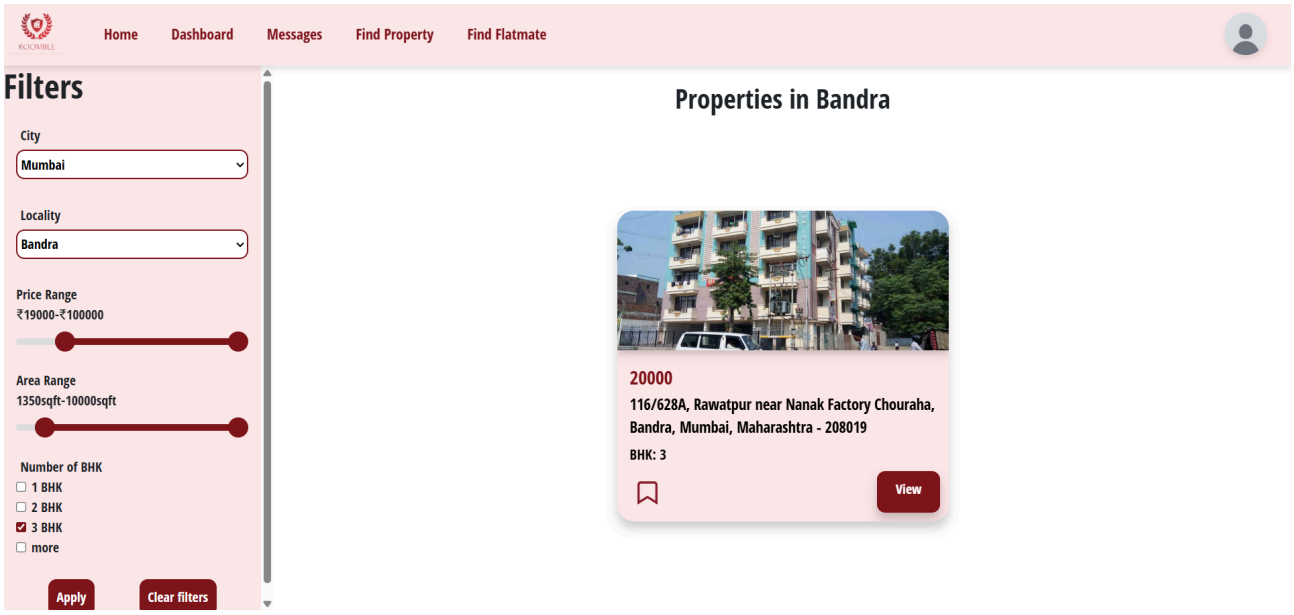
The user is prompted with the notification that the bookmark is unmarked successfully.

Test Results: The user clicks on bookmark icon for interested properties and flatmates, after which, the user is prompted with the confirmation message to add/delete bookmark, to which, when the user clicks on “Yes, confirm it!”, the bookmark is respectively added or removed, and the user is notified with a success message that the bookmark has been added or removed. Moreover, the bookmarked properties and flatmates are visible at the tenant dashboard. Similarly, the properties/flatmates are removed from the dashboard after removing the bookmarks respectively.

10. Find Property

The tenant can apply filters for finding a property of interest

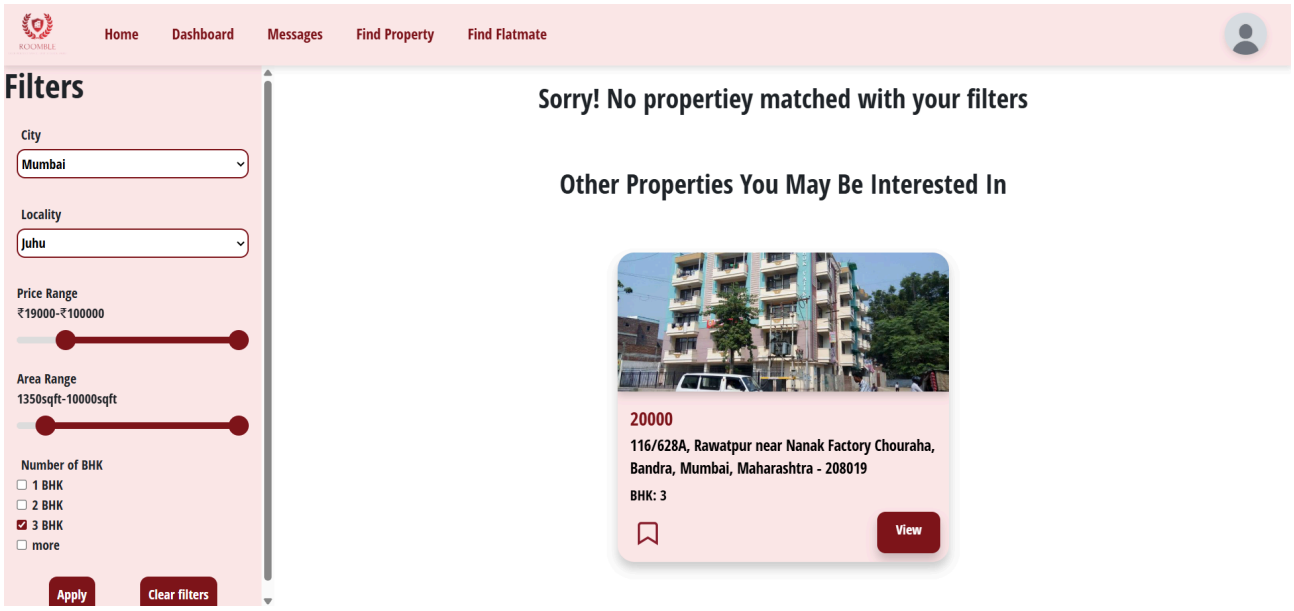
Module Details: This module contains the tests for validating the filters added during searching for a property of interest. The backend filters the properties based on filters provided during the search. The backend responds with an array of sorted properties according to the matched attributes with filters provided, which are displayed to the tenant. If no property satisfies the given filters, the search results display the message of property not found.
Test Owner: Hitarth Makawana
Test Date: 01/04/2025



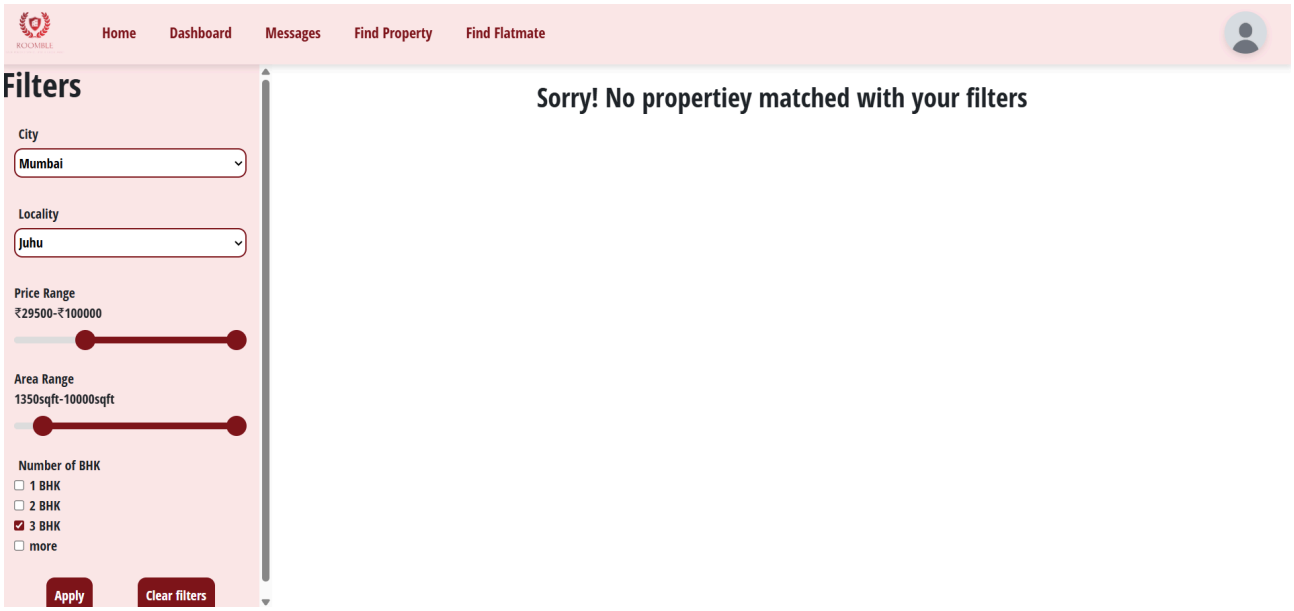
The tenant applies filters by choice and the search results display relevant properties


```
_id: ObjectId('67ebc94c9a575e22057f043a')
city : "Mumbai"
town : "Bandra"
address : "116/628A, Rawatpur near Nanak Factory Chouraha, Bandra, Mumbai, Mahara…"
area : 1400
bhk : 3
description : "This is a very good flat"
amenities : Array (3)
  0: "PS5 console"
  1: " no grass to touch"
  2: " pool table"
price : 20000
available : true
Images : Array (4)
  0: "http://localhost:3000/Pictures/property/67ebc94c9a575e22057f043a/0.png"
  1: "http://localhost:3000/Pictures/property/67ebc94c9a575e22057f043a/1.png"
  2: "http://localhost:3000/Pictures/property/67ebc94c9a575e22057f043a/2.png"
  3: "http://localhost:3000/Pictures/property/67ebc94c9a575e22057f043a/3.png"
lat : 19.05952552667832
lng : 72.83009995687377
reviews : Array (empty)
landlord : ObjectId('67ebc87b9a575e22057f041d')
createdAt : 2025-04-01T11:09:00.316+00:00
updatedAt : 2025-04-01T11:09:00.316+00:00
__v : 0
```


The backend database gives the properties array (here only single element) which is sorted according to the filters provided.



If the tenant applies some filters that don't match exactly but matches some other properties, then the user is suggested for the matched properties that he/she may be interested in.



If no property exists which satisfies the filters provided, then the search results show there is no property that matched the filters.

Test Results: When the tenant applies filters by choice, the database responds with an array of properties that are sorted according to the filters provided and the properties are displayed in the same order to the tenant. Moreover, if some filters don't match the exact property, but matches with some other properties, then the

user is suggested with the matched properties that he/she may be interested in. If the tenant gives some filters that do not match with any property, then the search results display a message that no property match the filters.

11. Adding and Viewing Reviews

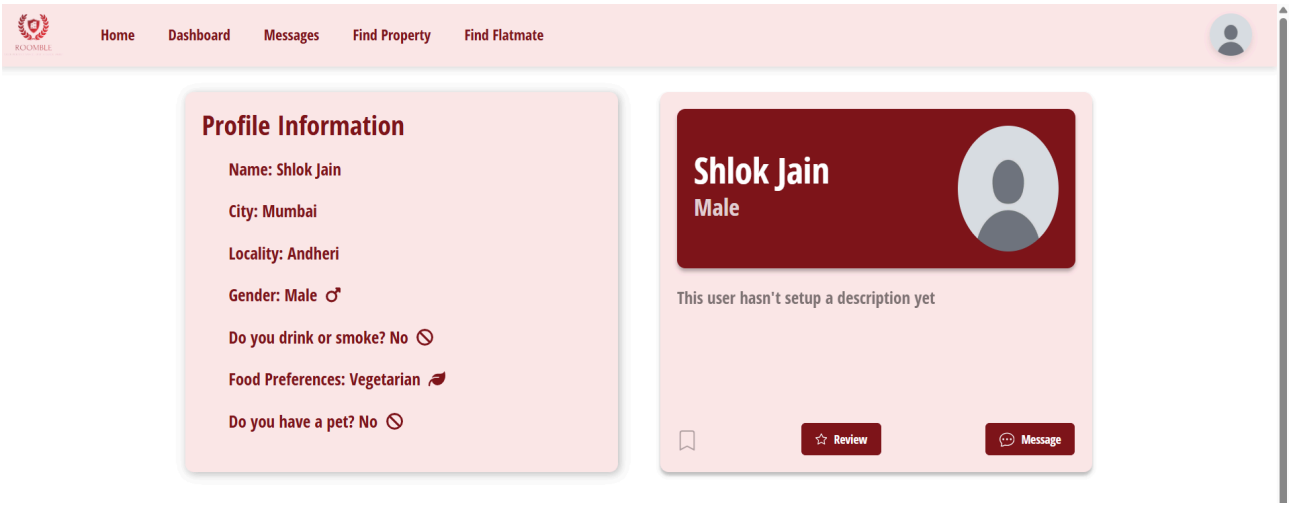
The tenant can review both flatmates as well as properties and it is open to all review system.

Module Details: This module contains tests for adding and viewing reviews for the flatmates and properties. When the tenant clicks on review, he can rate the flatmate/property out of five stars and write a review message.

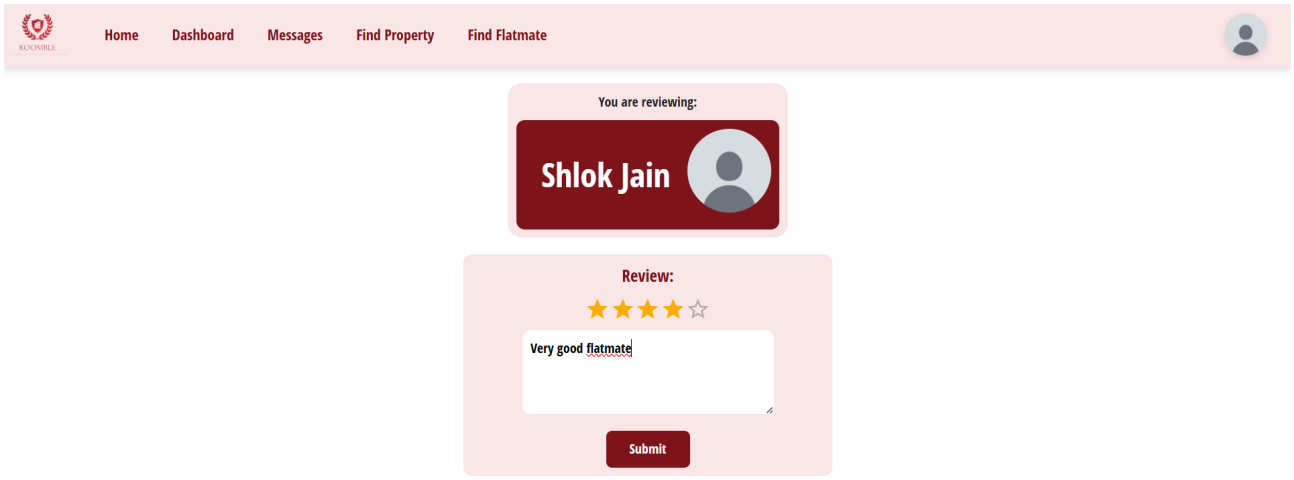
Test Owner: Hitarth Makawana

Test Date: 01/04/2025

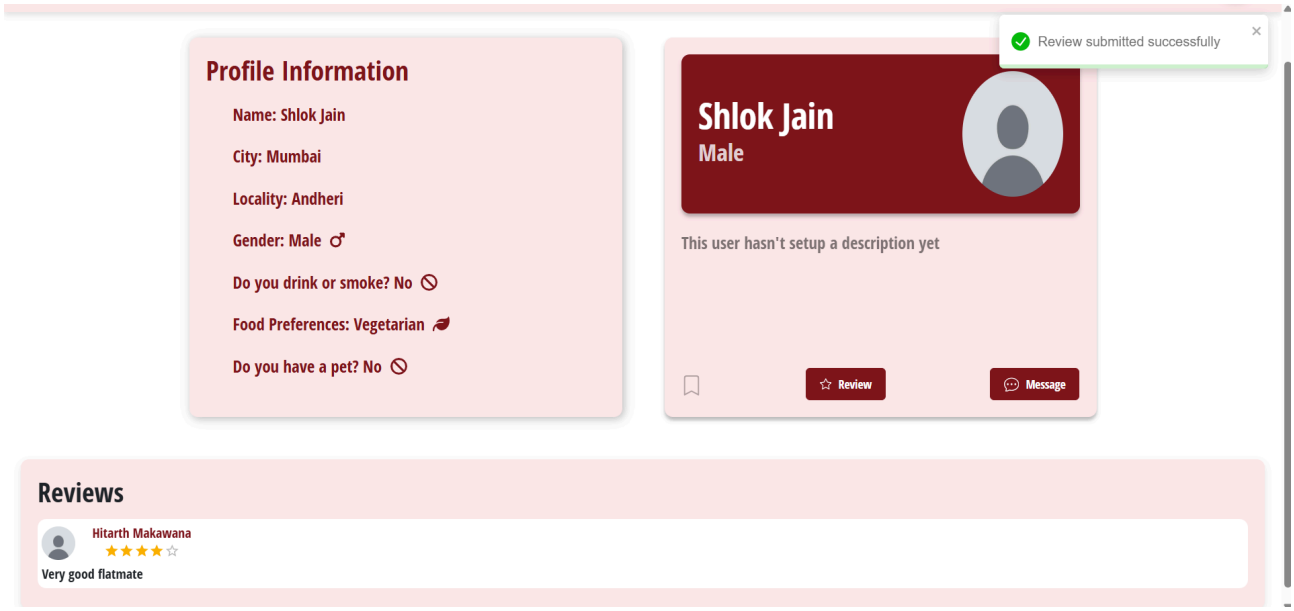
Test #1: Adding/Viewing review for a flatmate



The user clicks on review button and is prompted with a form to submit reviews.



The tenant can review the flatmate out of 5 stars, write a message, and click the submit button to submit the review.

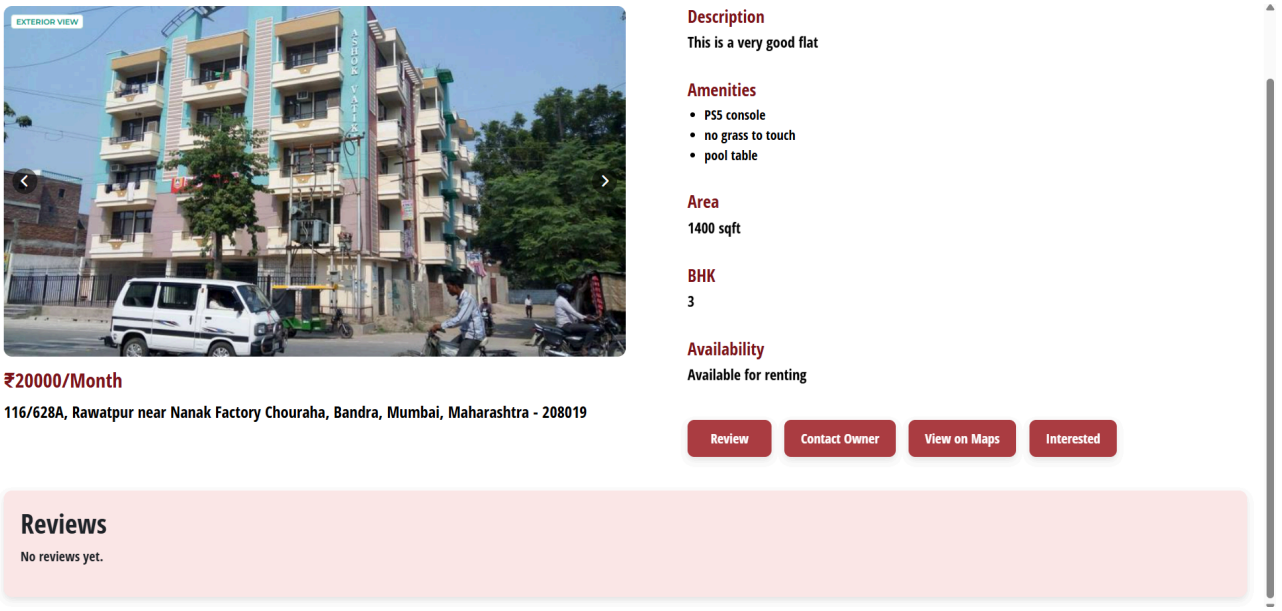


The review section of the flatmate is updated, and the current reviewer is prompted with a success message saying the review was submitted successfully.

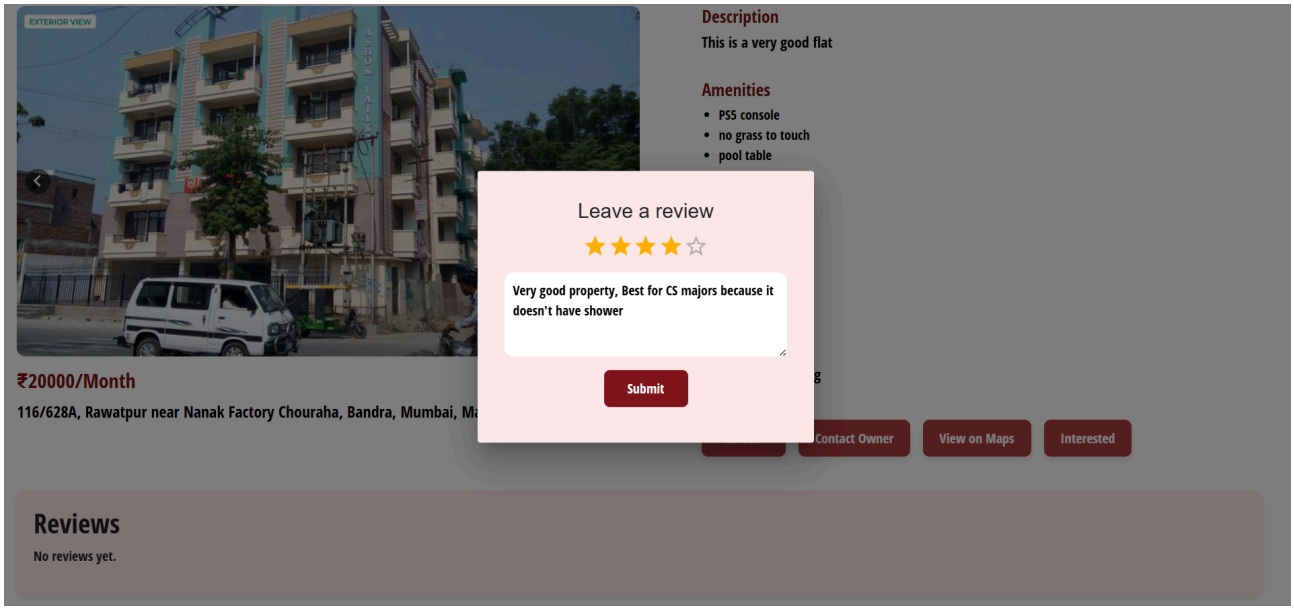
```
_id: ObjectId('67ebb688a115332bbd459875')
name : "Shlok Jain"
type : "tenant"
email : "shlokjain0177@gmail.com"
password : "$2b$10$f4iurr8NvkvyUE09melQ5Oa0s6d/ZxgHjYJxdxdfUdmPGI/L0vrES"
locality : "Andheri"
city : "Mumbai"
gender : true
smoke : false
veg : true
pets : false
flatmate : true
description : "This user hasn't setup a description yet"
▸ conversations : Array (2)
▸ bookmarks_tenants : Array (empty)
▸ bookmarks_property : Array (empty)
Images : "http://127.0.0.1:3000/Pictures/Default.png"
▾ reviews : Array (1)
  ▾ 0: Object
      reviewer : ObjectId('67ebc4919a575e22057f036f')
      rating : 4
      comment : "Very good flatmate"
      reviewertype : "tenant"
__v : 4
```


The MongoDB database is updated and the details of the reviewer and the review added by him/her is stored in the database.

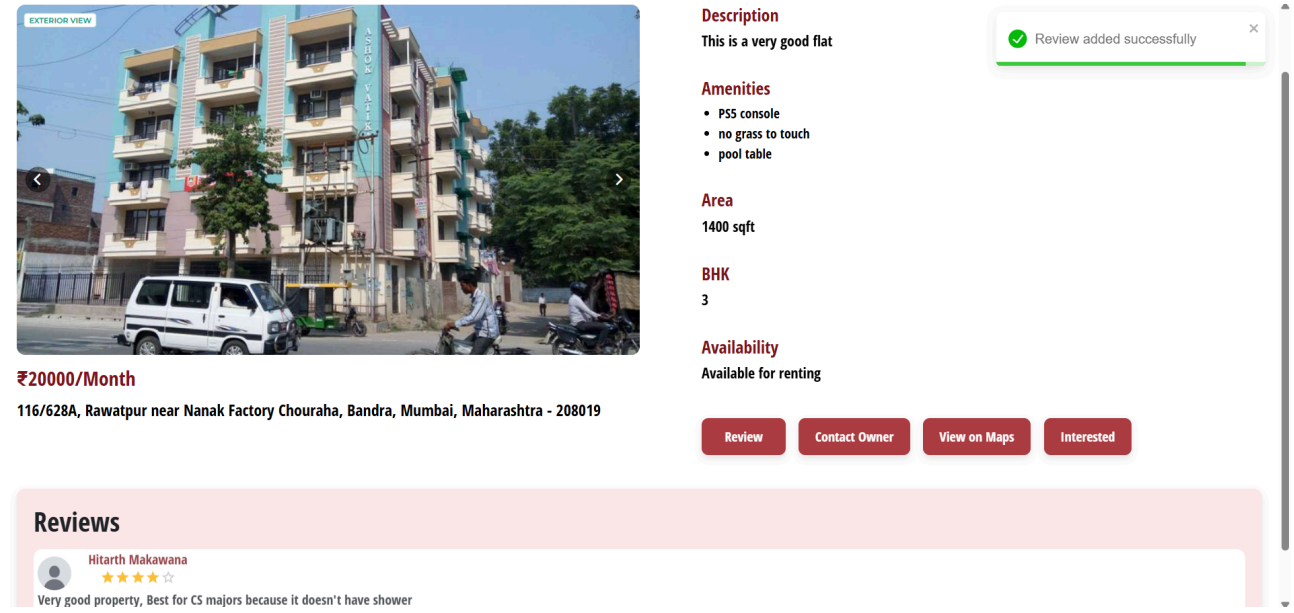
Test #2: Adding/Viewing reviews for properties



The user clicks on review button and is prompted with a form to submit reviews.



The tenant can review the property out of 5 stars and can write a review message and click the submit button to submit the review.



The review section of the property is updated and the current reviewer is prompted with a success message saying review submitted successfully.

```
_id: ObjectId('67ebc94c9a575e22057f043a')
city : "Mumbai"
town : "Bandra"
address : "116/628A, Rawatpur near Nanak Factory Chouraha, Bandra, Mumbai, Mahara…"
area : 1400
bhk : 3
description : "This is a very good flat"
▸ amenities : Array (3)
price : 20000
available : true
▸ Images : Array (4)
lat : 19.05952552667832
lng : 72.83009995687377
▾ reviews : Array (1)
  ▾ 0: Object
      reviewer : ObjectId('67ebc4919a575e22057f036f')
      reviewertype : "tenant"
      rating : 4
      comment : "Very good property, Best for CS majors because it doesn't have shower"
landlord : ObjectId('67ebc87b9a575e22057f041d')
createdAt : 2025-04-01T11:09:00.316+00:00
updatedAt : 2025-04-01T11:44:50.925+00:00
__v : 2
```


The MongoDB database is updated and the details of the reviewer and the review added by him/her is stored in the database.

Test Results: When the tenant clicks on review button under any of the flatmate/property profiles, then the tenant has to fill a review form where he/she has to rate the flatmate/property out of 5 stars and write a review on them. After submitting the form, the tenant is notified with the success message that the review submitted successfully.

12. Add Property

Module Details: Tested the Add Property Functionality for Landlord, to see if the details the Landlord adds in the frontend are correctly getting stored in the backend.
Test Owner: Aarsh Jain
Test Date: 01/04/2025

Add Property

Upload Photos *

Drag & Drop image uploading

Drag & Drop image here or **Browse**

Description

hola amigos!

BHK *

4

Area(sqft) *

2000

Rent(Per Month) *

10000

City *

Mumbai

Location *

Dahisar

Address *

Hall 12, D112, Dahisar

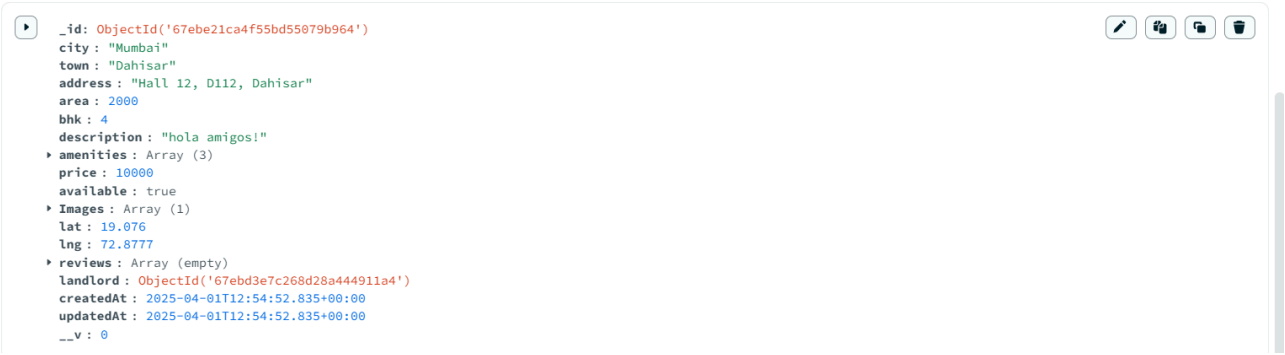
Amenities

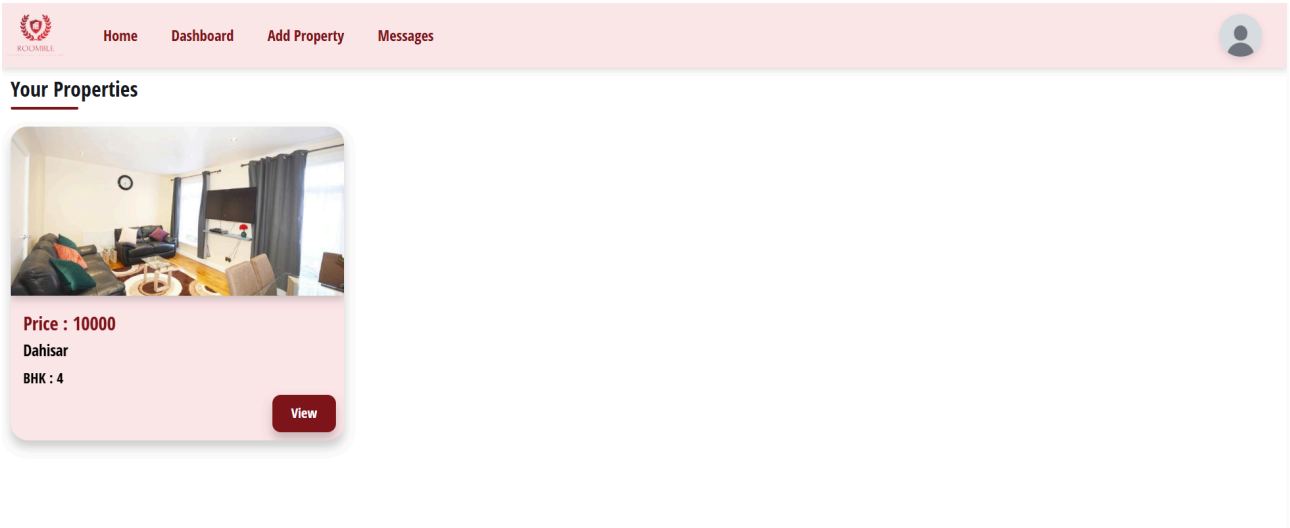
gym, pool, games

Pick the location on Maps

Submit

The landlord when clicks on Submit, the property should be successfully added to the backend.



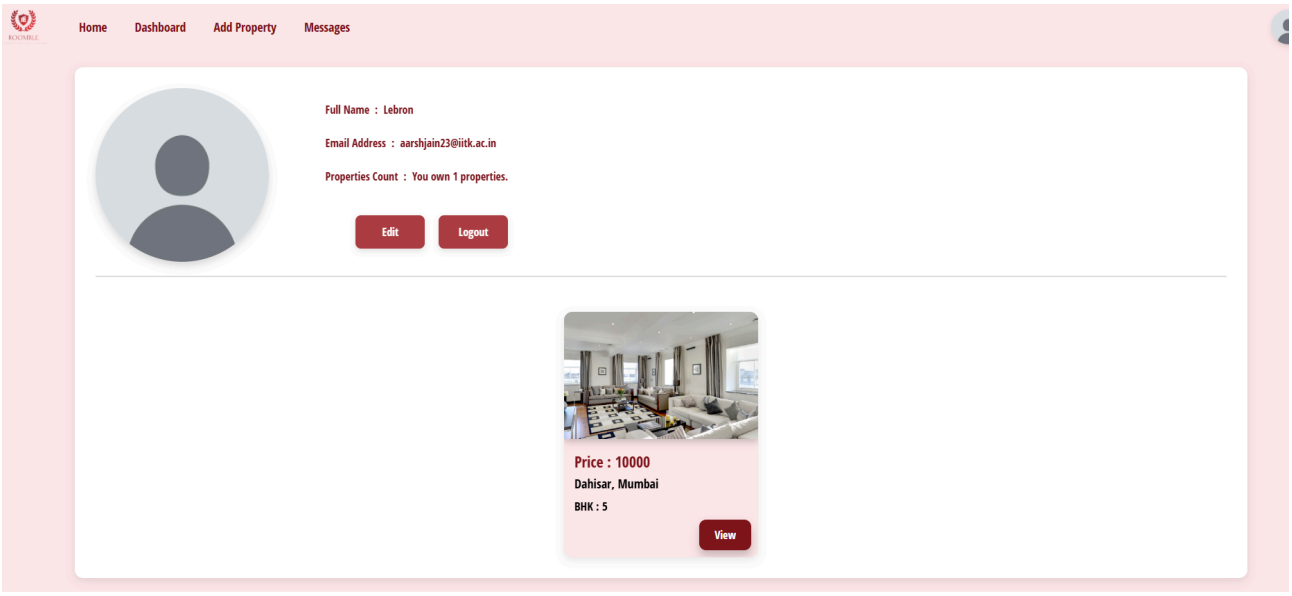


Test Results: The property gets stored successfully with the correct details, and gets updated in the user dashboard.

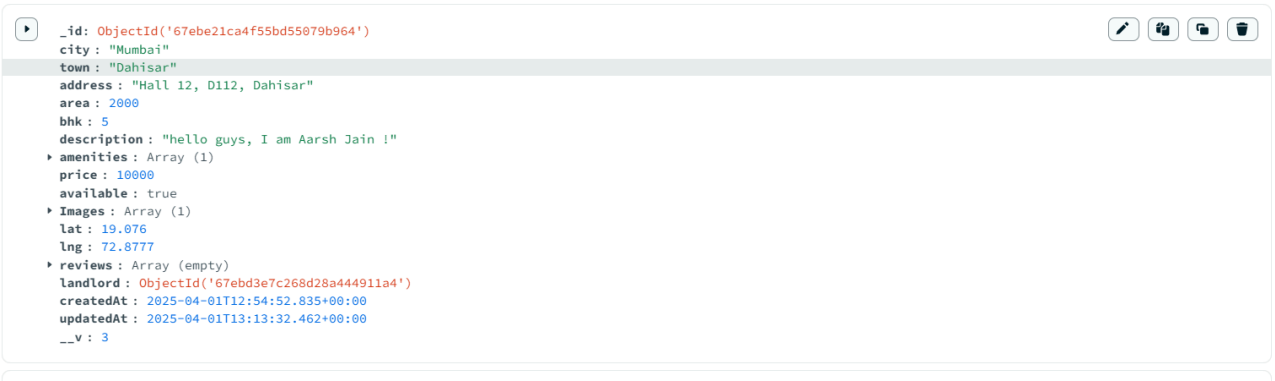
13. Edit Property

Module Details: Tested the Edit Property Functionality for Landlord, to see if the details the Landlord changes in the frontend are correctly getting updated in the backend.

Test Owner: Aarsh Jain
Test Date: 01/04/2025



The Landlord clicked on edit, got redirected to this page, and changed the following details. After clicking on submit, I got redirected to the profile page, where the pic and changed details are updated.

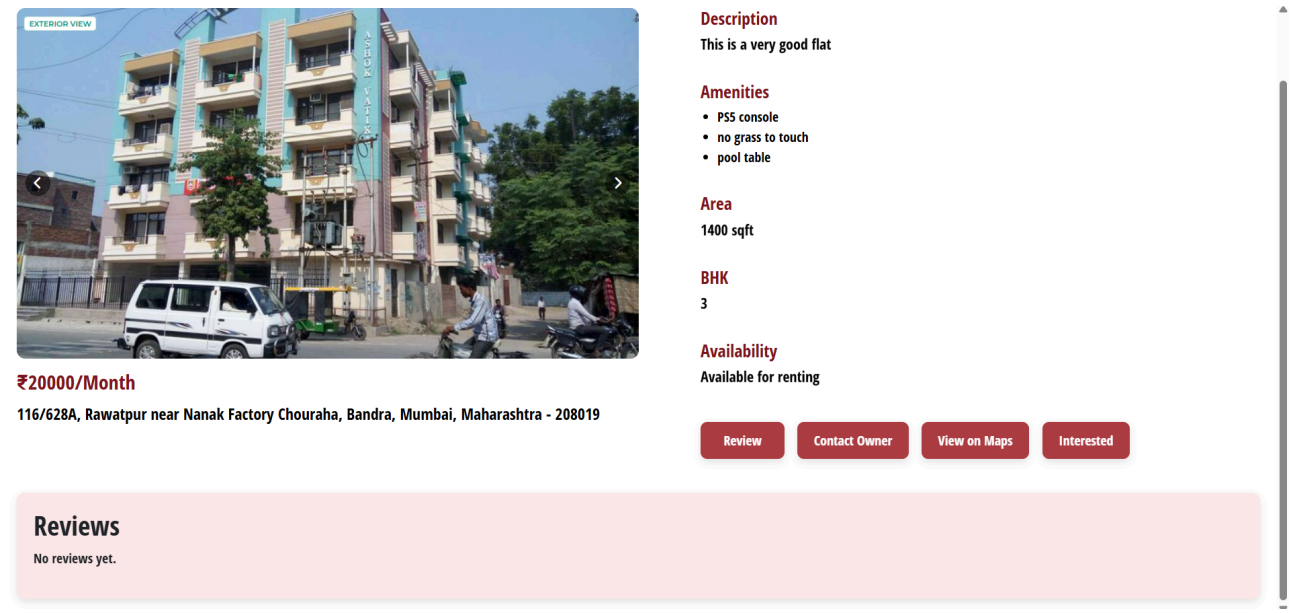


Test Results: The property details that were changed in the front end were edited and updated successfully in the backend.

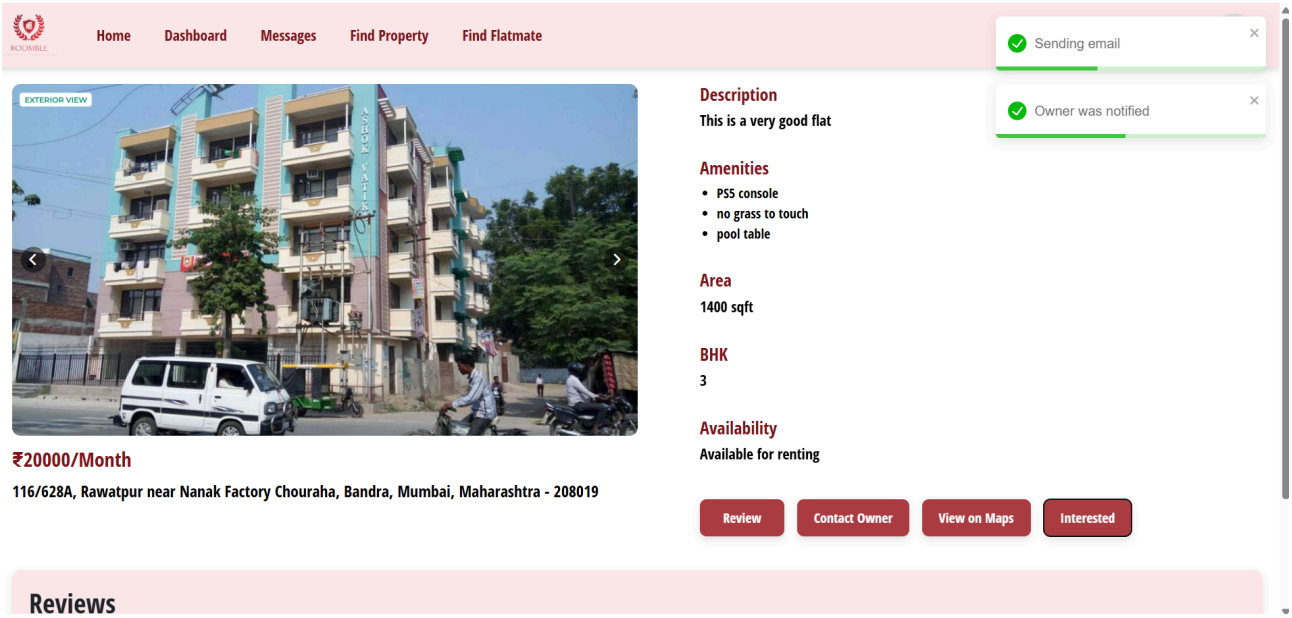
14. Mailing the landlord if the tenant clicks on “Interested” property

The tenant can show interest in a property by clicking on the interested button, which notifies the landlord via mail.

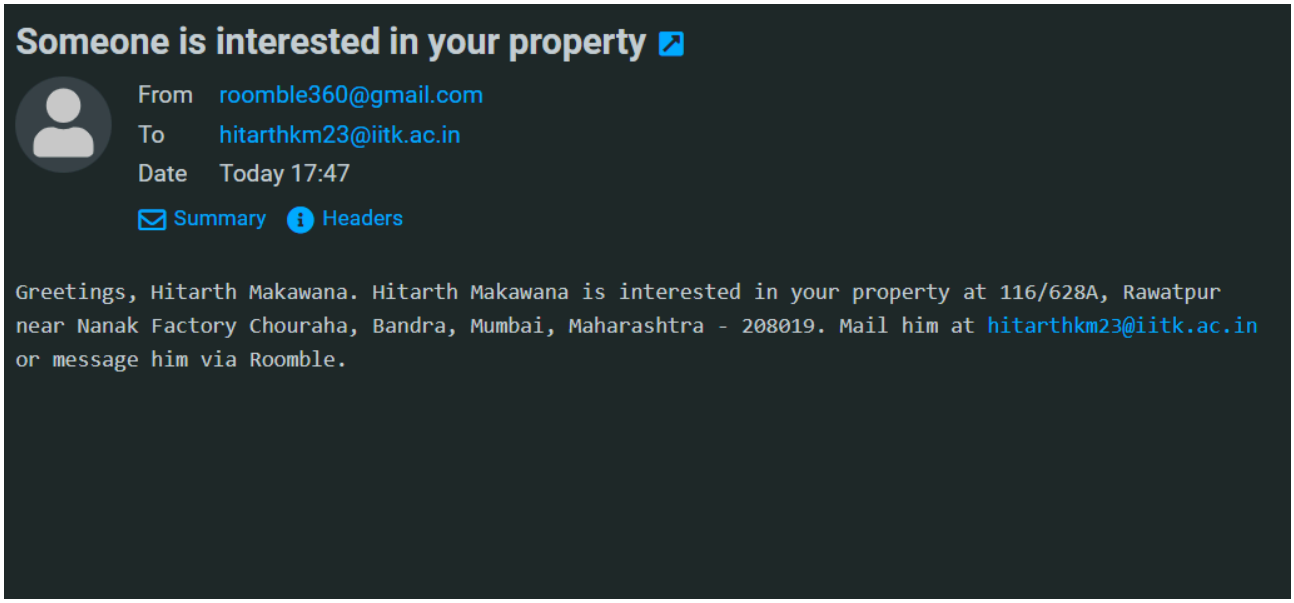
Module Details: This module contains tests for sending an email to the landlord whenever the tenant clicks on the interested button for the property he/she is interested in.
Test Owner: Hitarth Makawana
Test Date: 01/04/2025



The tenant clicks on the Interested button to notify the landlord that he/she is interested.



On clicking the interested button, the tenant is notified with the success message "sending email." When the email is sent to the landlord, it shows the success message "Owner was notified."



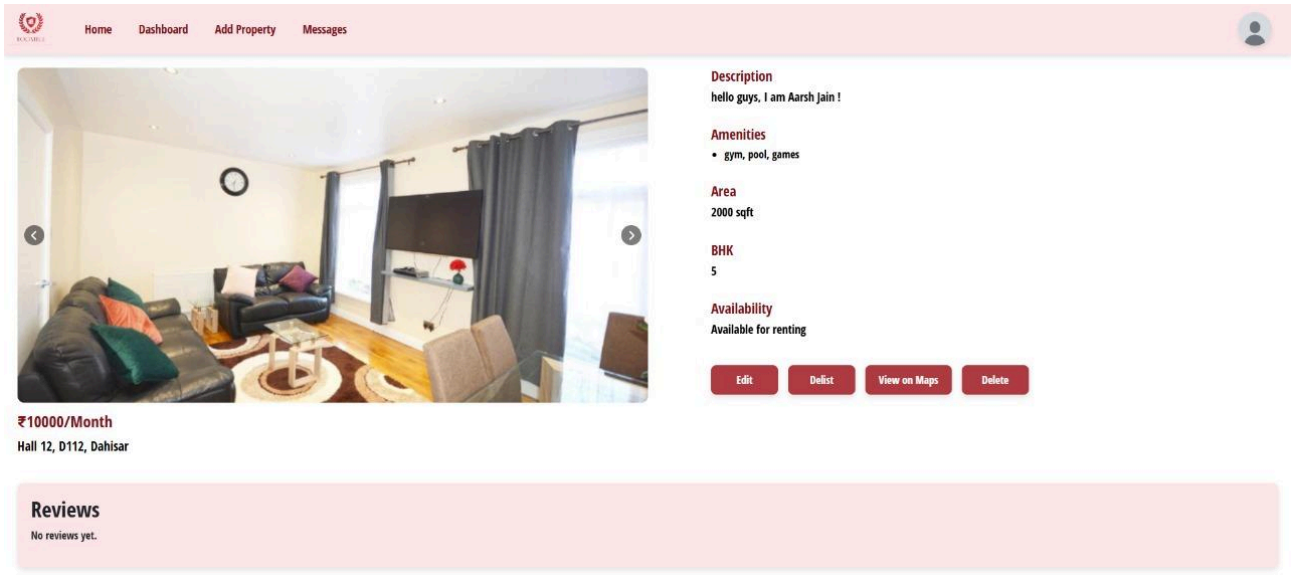
The landlord is notified via mail that someone is interested in his/her property. The mail includes the details regarding the tenant interested and the property address that the tenant is interested in.

Test Results: When the tenant clicks on the Interested button, the backend sends a success message of sending an email to the landlord who owns the property and notifies the tenant that the mail is sent. Additionally, the landlord is notified over his/her registered email about the details of the interested tenant and the interested property's address.

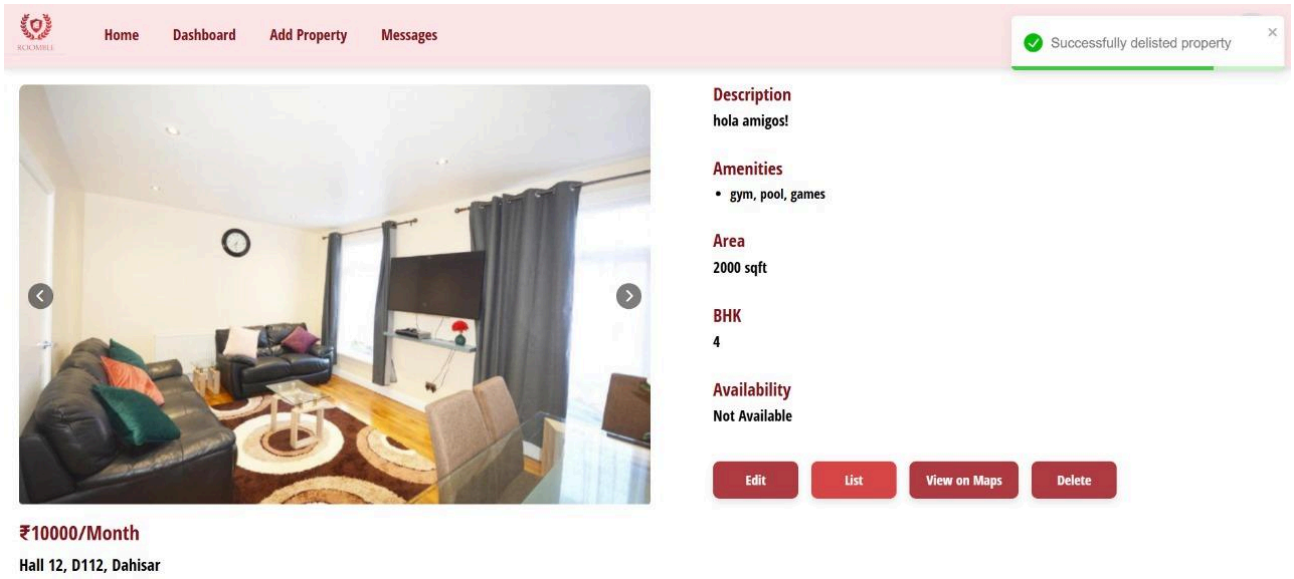
15. List/Delist Property

The landlord can list/delist his/her properties depending on whether or not a tenant is staying there. All other users can see the property as grey if it is delisted.

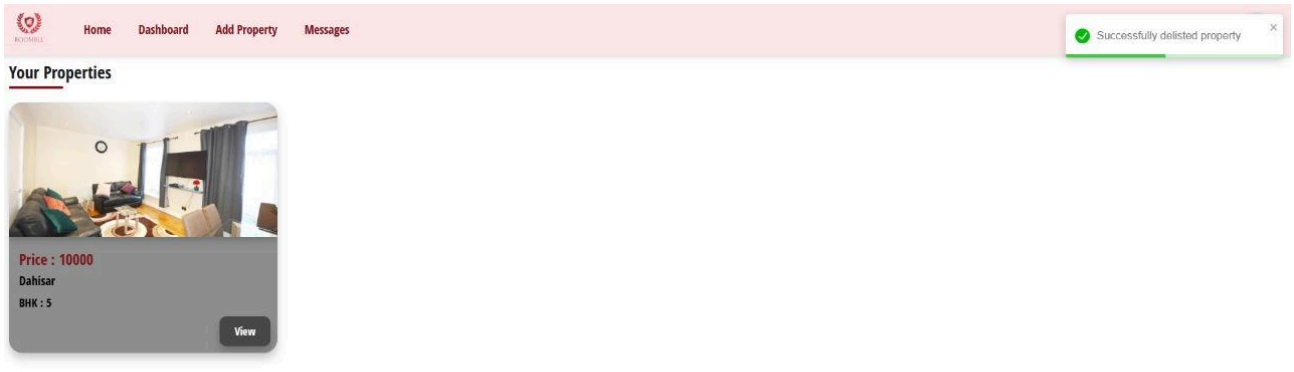
Module Details: This module contains tests for checking the list/delist feature used by the landlord to list/delist his properties, depending upon whether the landlord is satisfied with the property being occupied.
Test Owner: Hitarth Makawana
Test Date: 01/04/2025



The landlord clicks on delist button to delist his/her property



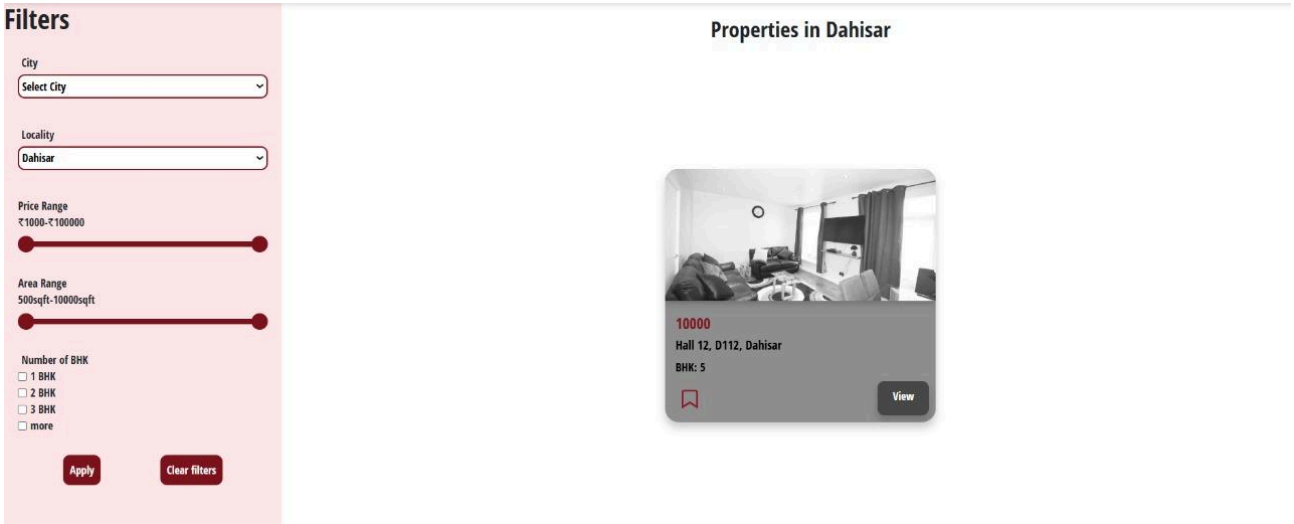
The landlord is notified that the property is delisted



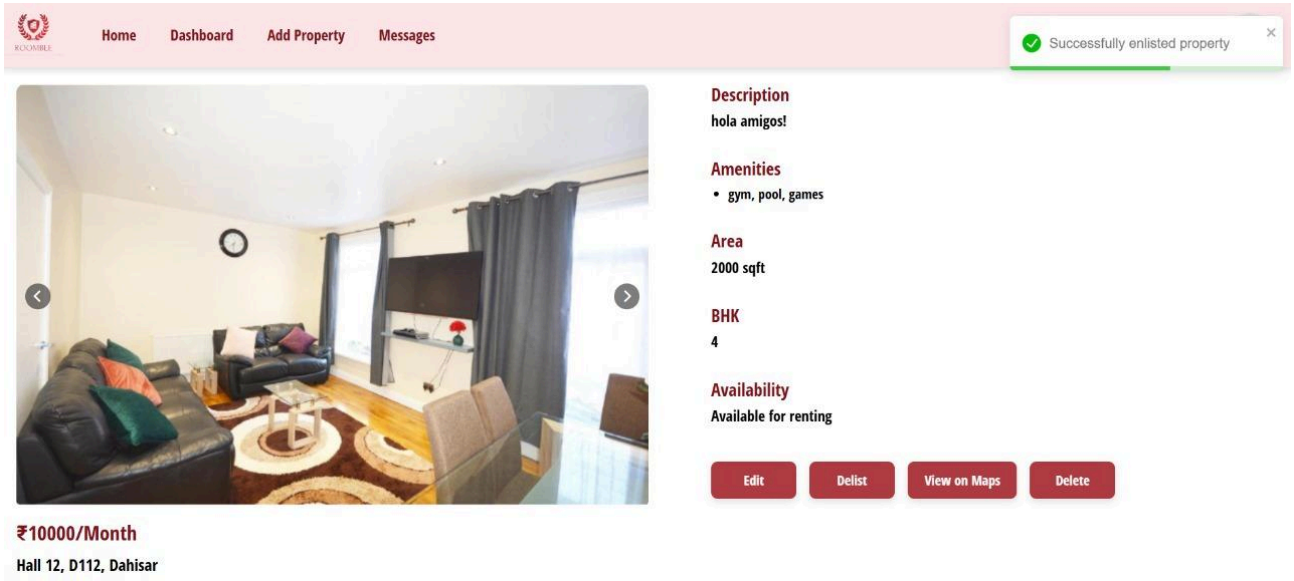
The property is displayed as grey on the landlord dashboard, after delisting.


```
_id: ObjectId('67ebe21ca4f55bd55079b964')
city : "Mumbai"
town : "Dahisar"
address : "Hall 12, D112, Dahisar"
area : 2000
bhk : 5
description : "hello guys, I am Aarsh Jain !"
amenities : Array (1)
  0: "gym, pool, games"
price : 10000
available : false
Images : Array (1)
  0: "http://localhost:3000/Pictures/property/67ebe21ca4f55bd55079b964/0.png"
lat : 19.076
lng : 72.8777
reviews : Array (empty)
landlord : ObjectId('67ebd3e7c268d28a444911a4')
createdAt : 2025-04-01T12:54:52.835+00:00
updatedAt : 2025-04-01T13:18:37.411+00:00
__v : 4
```

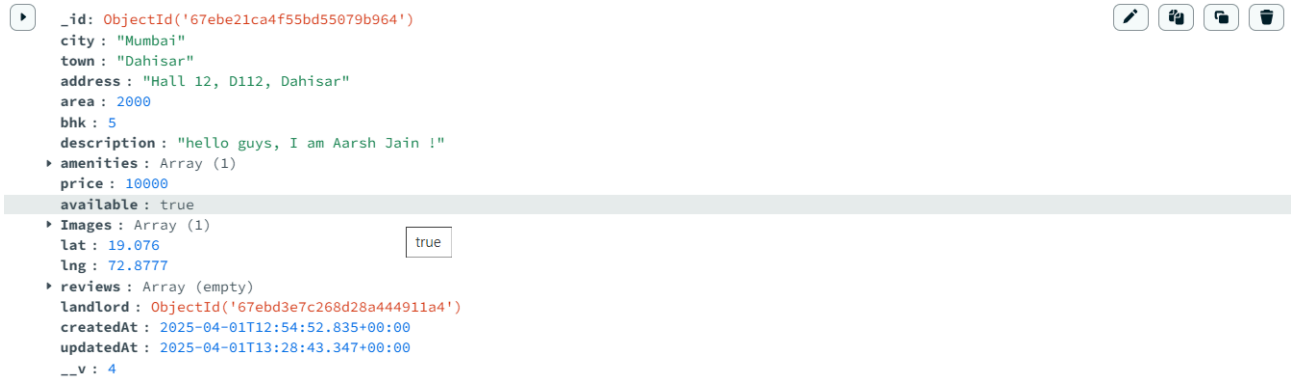

The MongoDB database also updates the available attribute as false after delisting.



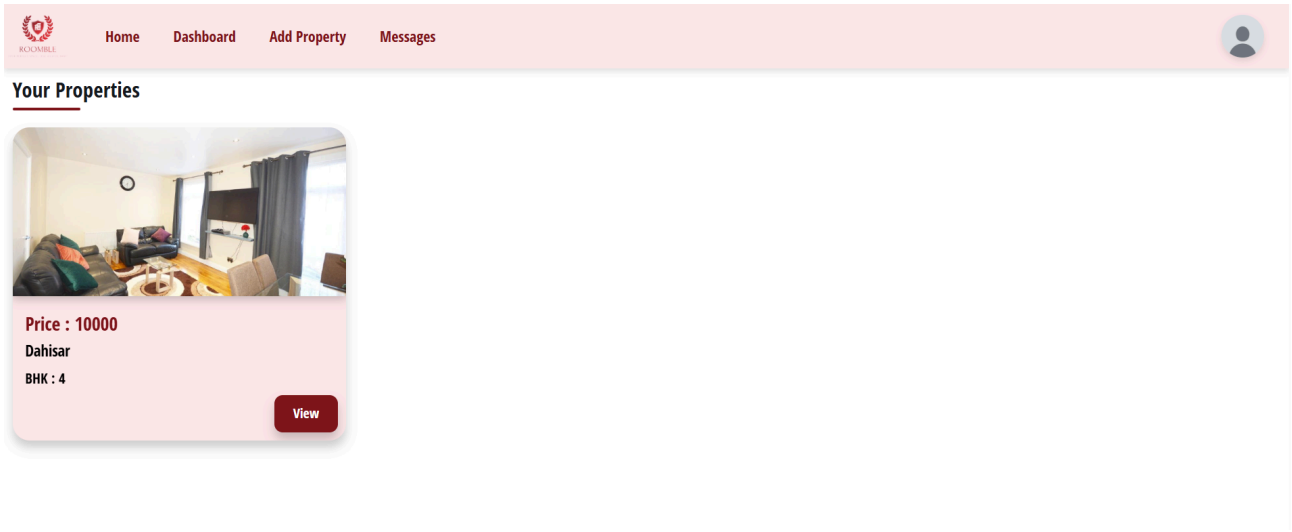
The property is displayed as grey to the tenant also when he/she searches for the property



The landlord can similarly click on the list button to list his property back, after which he will be notified that the property is enlisted successfully.



The MongoDB database also gets updated where the available field is true after enlisting the property.

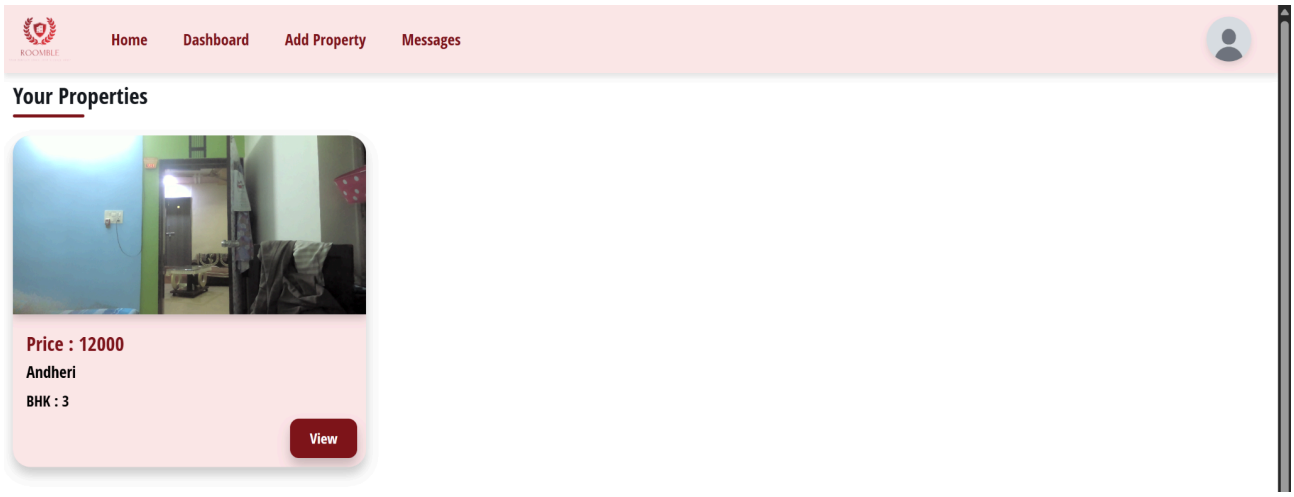


The property is visible again as it earlier was, before delisting to both the landlord and all other users.

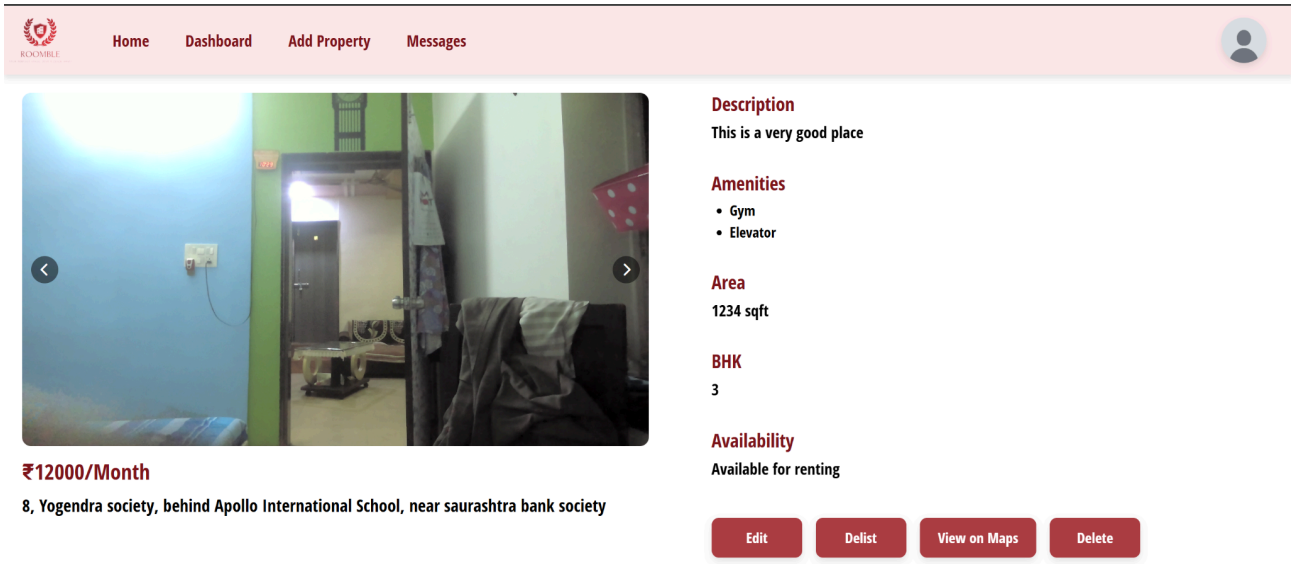
Test Results: When the landlord clicks on the delist button, the property is delisted, the availability of the property on the MongoDB database is set to false, and then the property is visible as grey to all users, even the landlord that owns it. Similarly, after clicking on the list button, the MongoDB database updates the availability field to true, and hence, all the users can see the property again as it was before delisting. The landlord is notified every time with a success message for enlisting/delisting his her/ property.

16. Delete Property

Module Details: We have implemented a delete property feature; the landlord, who will be the property owner, will get the option to delete it from the Database.



So, consider an initial version of the landlord dashboard, as we can see there is a property already added.



After going to this property page, I will get the button to delete it.

Clicking the delete button will remove the property from the database using the API. As we can see, the property is removed from the dashboard after deleting.

Test Owner: Surepally Pranaysriharsha
Test Date: 1/4/2025
Test Results: The delete functionality is working correctly.

4 System Testing

Functional Requirements

1. Requirement: A new user should be able to sign up by creating an account and providing the required details, such as name and email ID. If the user is a Tenant, they must also answer a few additional questions, including City, Locality, Gender, Smoking/Drinking preferences, Pets, Food preferences, and whether they are looking for flatmates. Finally, the user must set a valid password to complete the registration.

The sign-up functionality allows new users to create accounts by providing the required details as outlined above. To validate its effectiveness, we designed a comprehensive set of test cases covering a variety of scenarios. These tests were conducted by manually navigating to the sign-up page, entering both valid and invalid data into the respective fields, and observing the system's responses. We confirmed that users could successfully register by providing all necessary details and setting a valid password, resulting in the successful creation of their account. In addition, we tested edge cases such as incomplete or incorrect information, duplicate email addresses, and passwords that did not meet the required length constraints (minimum 6 and maximum 10 characters, inclusive) to ensure proper validation and error handling.

One notable case involved the use of the same email address to sign up as both a Tenant and a Landlord. While this is logically acceptable—since a Landlord could potentially also be a Tenant elsewhere—it introduces the risk of a user reviewing their own Landlord profile from their Tenant account. This scenario should be handled appropriately by restricting users from reviewing their own alternate user-type accounts.

Another issue was observed during the Tenant registration process. When a duplicate email ID is detected on the second page (where personal details are entered), the system displays the error message on the first page. Logically, the system should either prevent the user from progressing past the first page if the email already exists or display the message at the end of the second page after submission.

Aside from these issues, no major bugs or inconsistencies were found during testing, confirming that the sign-up functionality performs as intended and meets the defined requirements.

Test Owner: Bhukya Vaishnavi
Test Date: 04/04/2025
Test Results: The Sign-Up page functioned as expected. The first issue—where a user could review their own Landlord account—was resolved by ensuring that users cannot review a Landlord profile linked to their own email ID. The second issue, related to the incorrect placement of the duplicate email error message during Tenant registration, was addressed by updating the code to display the error on the appropriate sign-up page.
Additional Comments: NA

2. Requirement: The user should be able to sign in through his email ID and password if he is already a member.

The sign-in functionality was thoroughly tested to ensure its reliability and effectiveness. We developed a range of test cases covering various scenarios, including both valid and invalid inputs. These tests were performed by manually navigating to the sign-in page, entering different combinations of email IDs and passwords, and observing the system's responses.

For instance, we verified the successful sign-in process by using valid credentials and confirming that the user was correctly redirected to the Dashboard page. We also tested invalid scenarios such as incorrect email formats, unregistered user credentials, and wrong passwords. In each case, the system displayed appropriate error messages—such as "Wrong Password. Entry Denied" for incorrect passwords and "No User Exists" for non-registered users.

No bugs or issues were encountered during testing, confirming that the sign-in functionality works as intended and meets the specified requirements.
Test Owner: Bhukya Vaishnavi
Test Date: 04/04/2025
Test Results: The Sign-in page worked as expected. We tested it thoroughly. No bug was reported.
Additional Comments: NA

3. Requirement: There should be an option of ‘Forgot Password?’ in case the user forgets the password. Authentication shall be carried out through OTP verification on the email the user had provided earlier.

The "**Forgot Password**" functionality enables users to reset their password through OTP verification sent to their registered email address. To test this feature, we performed a series of scenarios including entering a valid email, receiving and verifying the OTP, and successfully setting a new password. Logging in with the new credentials confirmed that the process worked as intended.

We also validated error handling for edge cases such as entering an invalid or unregistered email, and incorrect OTPs—each of which triggered appropriate error messages and responses.

The "Forgot Password" functionality performed reliably and met all specified requirements.

Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The "**Forgot Password**" feature is now working correctly, and no bugs have been reported.
Additional Comments: NA

4. Requirement: Upon logging in, the **Tenant Dashboard** should display three main sections: **Messages**, **Find Property**, and **Find Flatmates**.
The dashboard should also allow tenants to view their **bookmarked properties and flatmates**, with the ability to switch between the two using a toggle/switch. In the **Messages tab**, users can view messages from other users and engage in real-time chat. In the **Find Property tab**, users can browse properties using various filters to refine their search. In the **Find Flatmates tab**, users can search for potential flatmates based on their preferences.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The Dashboard page worked as expected. We tested it thoroughly. No bug was reported.
Additional Comments: NA

5. Requirement: Upon logging in, the **Landlord Dashboard** should display two main sections: **Messages, Add Property**.
The dashboard should also allow landlords to view their **added properties**. In the **Messages tab**, users can view messages from other users and engage in real-time chat. In the **Add Property tab**, users can add properties for rent.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The Dashboard page worked as expected. We tested it thoroughly. No bug was reported.
Additional Comments: NA

6. Requirement: The dashboards shall also have the **Profile** option (where the user can change or update his other details except email).

The **Edit** (Profile Editing) and **Logout** options on the dashboard underwent different test to validate their functionality. Manual testing confirmed that the Profile Edit option enables users to efficiently update their details, excluding email, with changes accurately reflected in the system. Additionally, the Logout option securely logs users out of their accounts, terminating the current session and redirects them to the login page for re-authentication. No bugs were found.

Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The profile option, edit and logout worked as expected. No bug was reported.
Additional Comments: NA

7. Requirement: Prospective landlords will be able to add their properties with appropriate details.

Using the **Add Property** tab Landlords are able to upload their property details, for example, Photos, Description, BHK, Area(sqft), Rent(per Month), City, Location, Address, Amenities. They can also Pick the location of the property on Maps.

Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The Add property option worked as expected. No bug was reported.
Additional Comments: NA

8. Requirement: Landlords will have the option of editing their listed properties. Under the Dashboard, all the listed properties of the Landlord are shown. By clicking on the **View** button of one of them, Using the **Edit** tab Landlods are able to edit their property details, for example, Photos, Description, BHK, Area(sqft), Rent(per Month), City, Location, Address, Amenities.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The Edit property option worked as expected. No bug was reported.
Additional Comments: NA

9. Requirement: Landlords will also have the option of delisting/listing, delete their listed properties.

Under the Dashboard, all the listed properties of the Landlord are shown. By clicking on one of them, using **Delist** button, temporarily they can make the property Unavailable for renting and using **List** they can again make them available. Using **Delete** they can delete the property from database. Using **View on Maps** they can view the exact location of the property.

Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: All the options worked as expected. No bug was reported.
Additional Comments: NA

10. Requirement: Tenants will have the option of finding properties.

By clicking on the **Find Property** tab on the Dashboard, Tenants can see the filters available to search for a property of their interest. By choosing the filters City, Locality, Price Range and BHK, we can find the best suitable property of our interest.

After clicking on **View** to see a property, **View on Maps** option gives the option to visit the property on maps. If he/she is interested in renting it, clicking on **Interested** button will send an email to the own informing the name and email of the tenant who's interested into his property. By clicking on **Contact Owner**, a conversation through messages can be initiated between them. Also, a button to **Review** the property is also there.

No bugs or issues were encountered during testing, confirming that the functionality works as intended and meets the specified requirements.

Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: All the options worked as expected. No bug was reported.
Additional Comments: NA

11. Requirement: Tenants will have the option of finding flatmates.
By clicking on the **Find Flatmates** tab on the **Dashboard**, tenants can access filters to search for potential flatmates based on their preferences. Users can refine their search by selecting filters such as **City, Locality,** and **Preferences** (Gender, Smoking/Drinking habits, Pets, and Food preferences). The system then generates a list of tenants ranked from **highest to lowest similarity score** based on the selected criteria.

By clicking on a profile, one can **Review** the User(Tenant) or **Message** them directly.No bugs or issues were encountered during testing, confirming that the functionality works as expected and meets the specified requirements.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: All the options worked as expected. No bug was reported. Also we assured by testing that a Tenant who has '**No**' in "**Are you Seeking a Flatmate?**" option in the profile, won't be visible to the other Tenants looking for flatmates.
Additional Comments: NA

12. Requirement: Tenants will have the option of reviewing Landlords, Tenants and Properties.
As mentioned earlier, a Tenant have the privilege to **Review a Property** or a **Tenant (Potential Flatmate)**. Also, clicking on **Contact Owner** option within a property, we will be redirected to the profile of the **Landlord**. There will be an option to **Review** the Landlord.To review a **Property** or a **User**, you must submit a rating (on a scale of **0 to 5**) along with a text review.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: All the review options worked as expected. No bug was reported.
Additional Comments: NA

13. Requirement: Tenants will have the option to engage in **two-way messaging** with both **Landlords** and **other Tenants**.

As mentioned earlier, a **Tenant** can access the profiles of both "**potential flatmates** (other Tenants) and **Landlords** (owners of viewed properties). Clicking on the **Message** button in either case redirects the user to the **Messages** page, where a chat conversation is initiated between the two parties. They can then communicate to discuss the property or potential flatmate arrangements.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The chat feature worked as expected. No bug was reported.
Additional Comments: NA

14. Requirement: Tenants will have the option to "**bookmark"** properties of interest and **Tenants** (potential flatmates) of their choice.

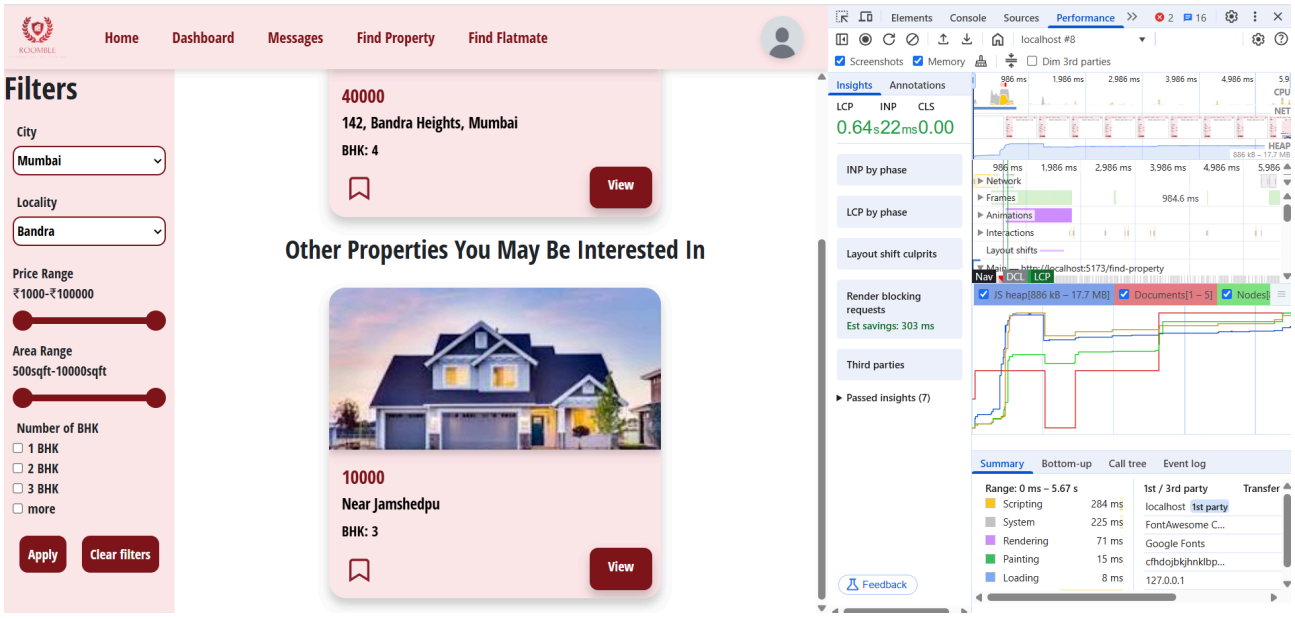
As mentioned earlier, a Tenant can browse through available **properties** in the **Find Property** tab and explore potential **flatmates** in the **Find Flatmates** tab. They can bookmark any property or flatmate they are interested in. These bookmarked selections will be reflected in the **Bookmarked Properties** and **Bookmarked Flatmates** sections on the Tenant's **Dashboard**.
Test Owner: Aritra Ambudh Dutta
Test Date: 04/04/2025
Test Results: The bookmark feature worked as expected. No bug was reported.
Additional Comments: NA

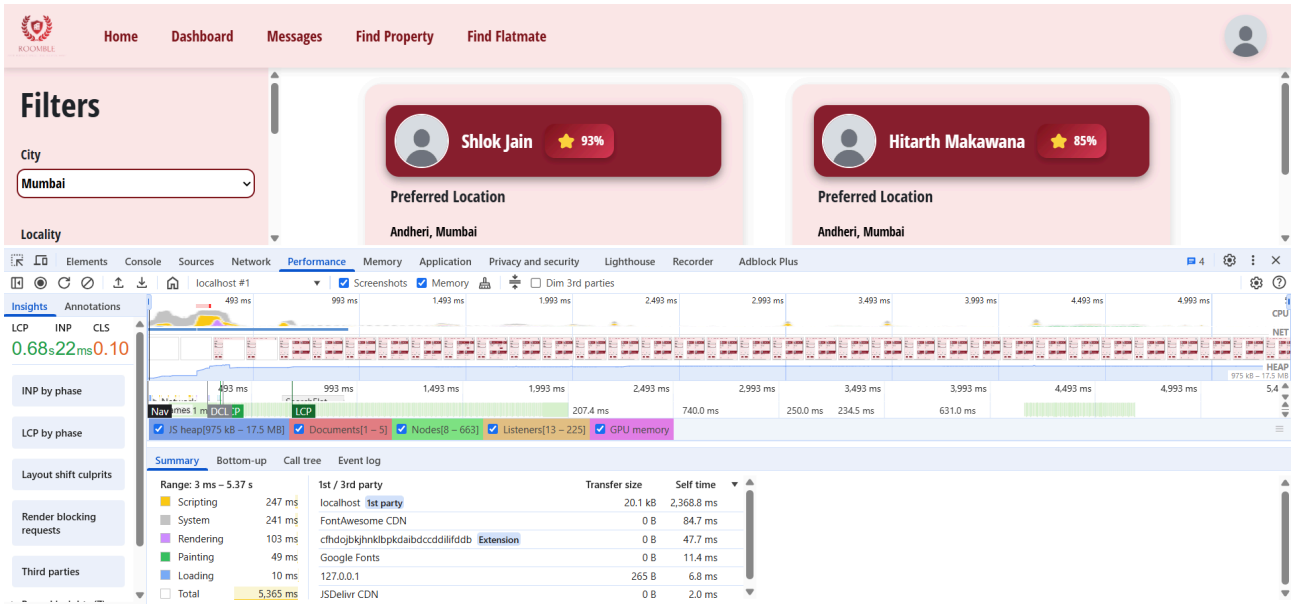
Non-Functional Requirements

1. Requirement: The software is expected to maintain 99% of the period of service, which does not include shut down due to maintenance breaks and software update windows.
Test owner: Aritra Ray

Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: We ran the full software and explored all its features several times with no disruption in the period of service whatsoever.

2. Requirement: The software should be able to respond to the search requests within 5 seconds with the best-matched search results with every search request by the user.
Test owner: Ronav Puri
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: We used ChromeDevTools to figure out the time latency of search. For searching properties, the mean time (approximate) required was 2.83s, while for searching flatmates, the mean time (approximate) required was 2.68s; both are much faster than 5 seconds. Moreover, always the best-matched results are returned without fail.





3. Requirement: Validates user credentials such as mobile numbers, emails, passwords, and roles (tenant/landlord).
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: User credentials are validated in each step in the backend using JWT authtokens.

4. Requirement: Cross-checks inputs with a securely stored database to ensure only authorized users gain access.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: During login, username and password are securely matched with the data in the database, which is securely stored with a secret MongoURI key. Additionally, passwords are hashed to ensure maximum security.

5. Requirement: Cross-checks inputs with a securely stored database to ensure only authorized users gain access.
Test owner: Ronav Puri
Test Date: 04/04/2025
Test Results: Test passed

Additional Comments: During login, username and password are securely matched with the data in the database, which is securely stored with a secret MongoURI key.

6. Requirement: Encrypts personal information using industry-standard protocols to ensure data is secure during storage and transmission.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: Passwords are hashed using bcrypt and stored securely in database.

7. Requirement: Restricts access to sensitive user data, making it invisible to admins and inaccessible over the server without proper verification.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: Passwords are hashed using bcrypt and stored securely in database, so even admins cannot access it.

8. Requirement: Facilitates landlord-tenant communication via a DM system, keeping contact details private and secure.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: Contact details like phone and email are not shared between the parties, hence privacy of both is preserved. The DM system acts like a secure channel of communication without any privacy concerns.

9. Requirement: Preserves data consistency during system downtime, enabling seamless recovery after maintenance.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: Data is stored in an external MongoDB database, hence system down time does not result in any data loss. Data can be recovered easily after connection is restored with the database.

10. Requirement: Ensures that updating or adding any data does not affect the already stored data.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: MongoDB is used for all database-related requirements, which supports the addition of deletion of data without any changes to the existing data.

11. Requirement: The software should have a user-friendly interface, which allows the users to navigate through desired functionalities without confusion so that the rental process proceeds with ease.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: The interface is fairly intuitive. Also all the features are briefly enumerated in the Home page, so users have no trouble in navigating through the software.

12. Requirement: The architecture, design, implementation, and documentation of the software must be such that they make the system reduce the maintenance overhead as much as possible and adding a particular minor feature should not take as more than 1 person week while fixing a security bug should be done within 2 person days along with updating the documentation.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: The software has been designed in an organized manner so that new features can be added and modifications can be done very easily within two working days. Various components of the software are well organised into folders and **comments** are added wherever required streamline future improvements.

13. Requirement: Since the user requests might vary seasonally hence the database availability might be enough to cater all the requests, i.e. ensure that the software can handle the load as and when the user and property database grows.

Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: The database has a maximum storage capacity of 512 MB while less than 1 MB is currently occupied. Hence the software can handle 500 times more load, which is reasonable.

14. Requirement: The MTTF (mean time to failure) shall be more than one week.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: The software was tested multiple times throughout the testing week period without a single failure.

15. Requirement: The software must be consistent in providing correct or best matched results according to the prompt filters and search requests.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: The same filters and search criteria for finding flatmates and properties were tried out multiple (10-12) times and consistent results were obtained each time.

16. Requirement: The software is portable and can run on most of the modern web browsers and wider screens.
Test owner: Aritra Ray
Test Date: 04/04/2025
Test Results: Test passed
Additional Comments: Node.js is used for designing the backend part of our software and React.js for designing the frontend part of our software, thus the website can run on most browsers. The software was tested on Chrome, Safari, Firefox, Edge and Opera and it ran successfully.

17. Requirement: The software is portable and can run on most of the modern web browsers and wider screens.
Test owner: Aritra Ray
Test Date: 04/04/2025

Test Results: Test passed
Additional Comments: Node.js is used for designing the backend part of our software and React.js for designing the frontend part of our software, thus the website can run on most browsers. The software was tested on Chrome, Safari, Firefox, Edge and Opera and it ran successfully.

5 Conclusion

- **How Effective and exhaustive was the testing?**

  Testing was done almost exhaustively. Each API underwent testing to ensure that they were modifying the database and returning the desired information when needed. We have taken care of all cases, thus ensuring complete branch coverage. The frontend was also tested and bugs were resolved. Validity of each input has been checked in all forms. Appropriate alerts have been shown in such cases. Edge cases were also tested and were confirmed to work as expected. For example, when the entered start date is ahead of the entered end date while viewing reports, an alert is displayed preventing the user from dealing with such invalid inputs.

  Also, the testing was quite effective. A developer tested a component that was not developed by him/her, thus eliminating the possibility of bias. The number of bugs we found and corrected was more than 50, indicating that the testing was effective.

- **Which components have not been tested adequately?**

  Our test coverage for non functional requirements outlined in the SRS document could be further expanded. Adequate performance tests for non-functional requirements like reliability, maintainability and ease of learning could not be performed.

- **What difficulties have you faced during testing?**

  The main difficulty faced during testing was ensuring non-functional requirements are satisfied, in system testing.

- **How could the testing process be improved?**

  The testing could have been improved by doing the unit testing or integration testing as automated rather than manual. The developer might subconsciously assume a few crucial things which may be not so obvious to

an end-user or someone who was not involved in the development of the software.

Appendix A - Group Log

Date	Timings	Duration	Discussion & Work done
29 Mar	13:00 - 14:00	1 hour	Meet to discuss the work distribution of the user manual and the testing document among sub-groups. Also initiated the code improvement process to align it with the testing process.
31 Mar	14:00 - 16:00	2 hours	Discussed the progress of the work along with the problems that were faced by the sub-groups.
3 Apr	20:00 - 23:00	3 hours	Finalised the User Manual and completed a large portion of the testing intended to be done
4 Apr	18:00 - 20:00	2 hours	Finalized the Testing Document after reviewing it and the User Manual thoroughly.