Roommee Functionalities

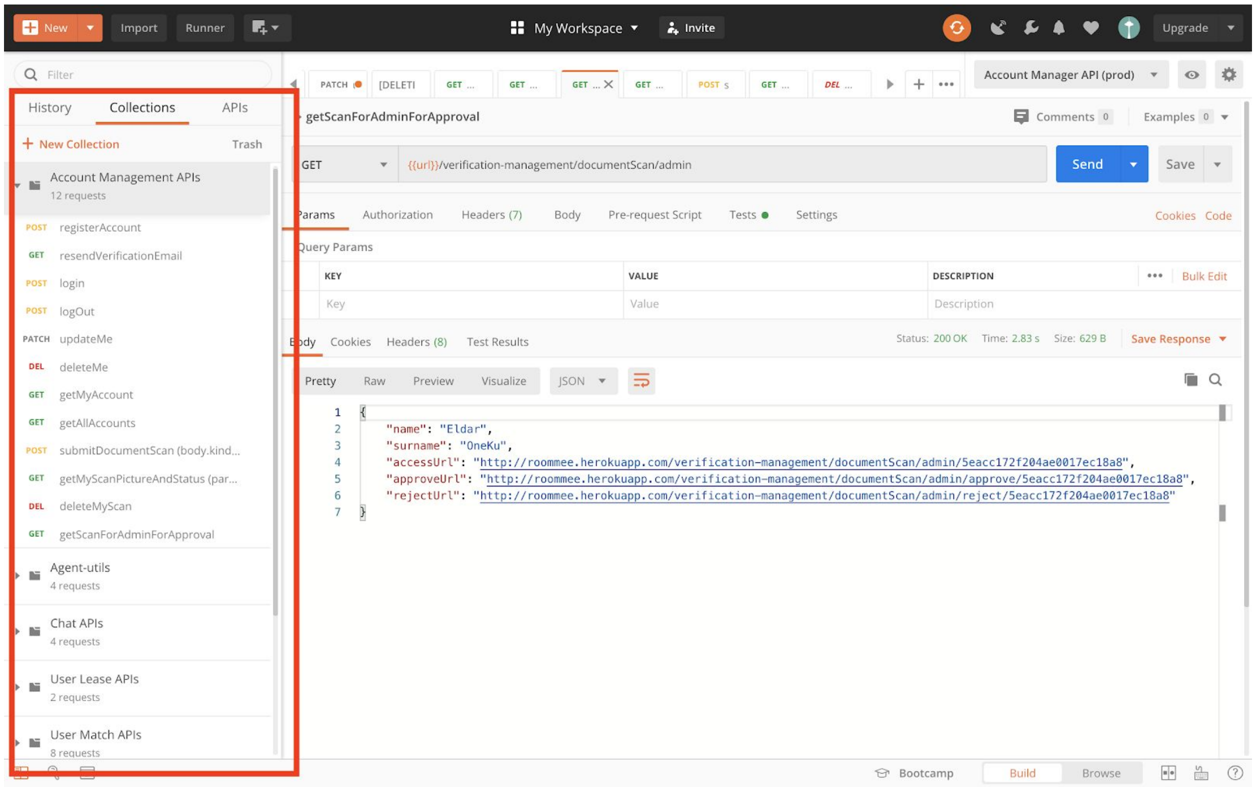
How to test in Postman or Chrome

Setting up Postman for testing

All endpoints requests have been set up in a shared Postman profile, so that we save time to the tutor of manually setting up a large number of routes.
Postman shared profile can be accessed with the following credentials via desktop app
login: ekurmakaev
password: assignment2!

Also we provide Mongo Atlas DB access in case it is necessary to check the DB
login: eldar.kurmakaev@gmail.com
password: GroupProject2!

All endpoint access configurations are within the Postman Collections tab on the left and are thematically organized in folders.



Account Management APIs
12 requests

POST	registerAccount
GET	resendVerificationEmail
POST	login
POST	logOut
PATCH	updateMe
DEL	deleteMe
GET	getMyAccount
GET	getAllAccounts
POST	submitDocumentScan (body.kind...
GET	getMyScanPictureAndStatus (par...
DEL	deleteMyScan
GET	getScanForAdminForApproval

Agent-utils
4 requests

Chat APIs
4 requests

User Lease APIs
2 requests

User Match APIs
8 requests

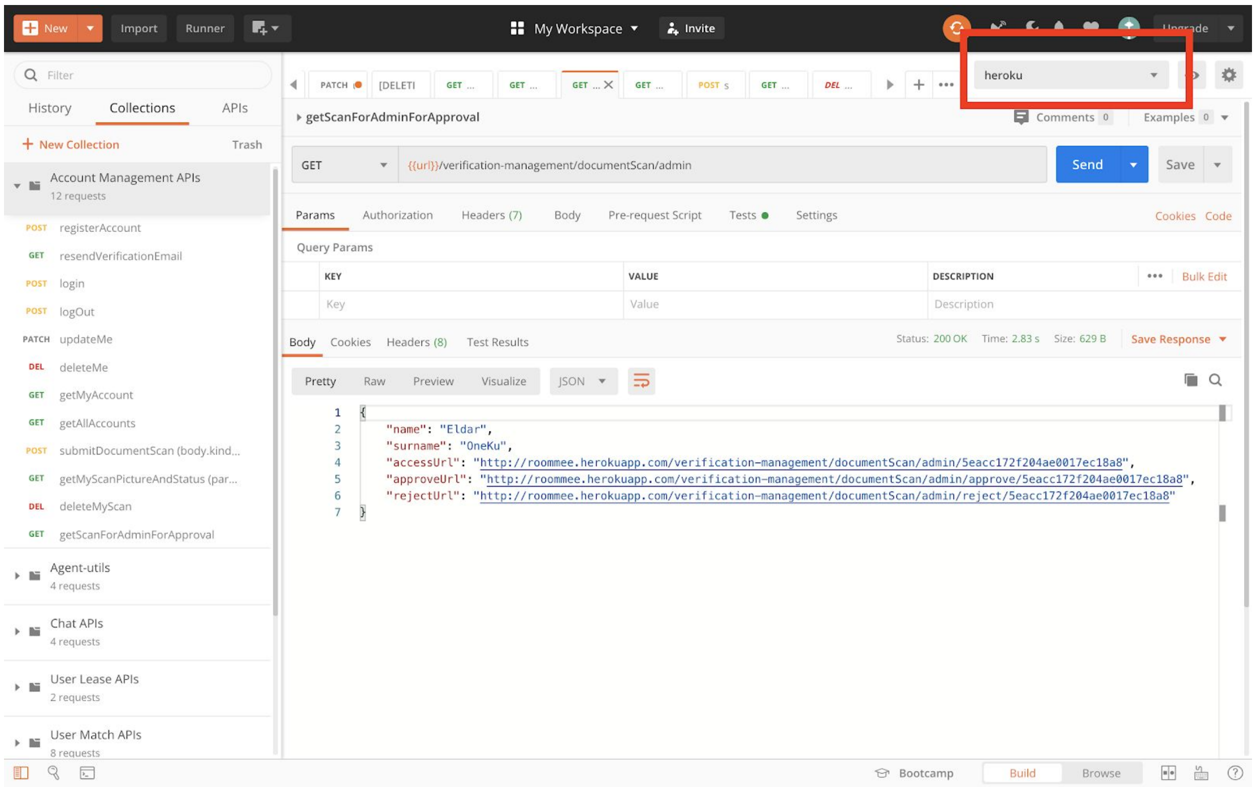
getScanForAdminForApproval

```
GET {{url}}/verification-management/documentScan/admin
```

Status: 200 OK Time: 2.83 s Size: 629 B

```
{
    "name": "Eldar",
    "surname": "OneKu",
    "accessUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/5eacc172f204ae0017ec18a8",
    "approveUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/approve/5eacc172f204ae0017ec18a8",
    "rejectUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/reject/5eacc172f204ae0017ec18a8"
}
```

Next it is important to check whether a correct {{url}} variable is set. Select “heroku” in the drop down list in the upper corner of the interface.

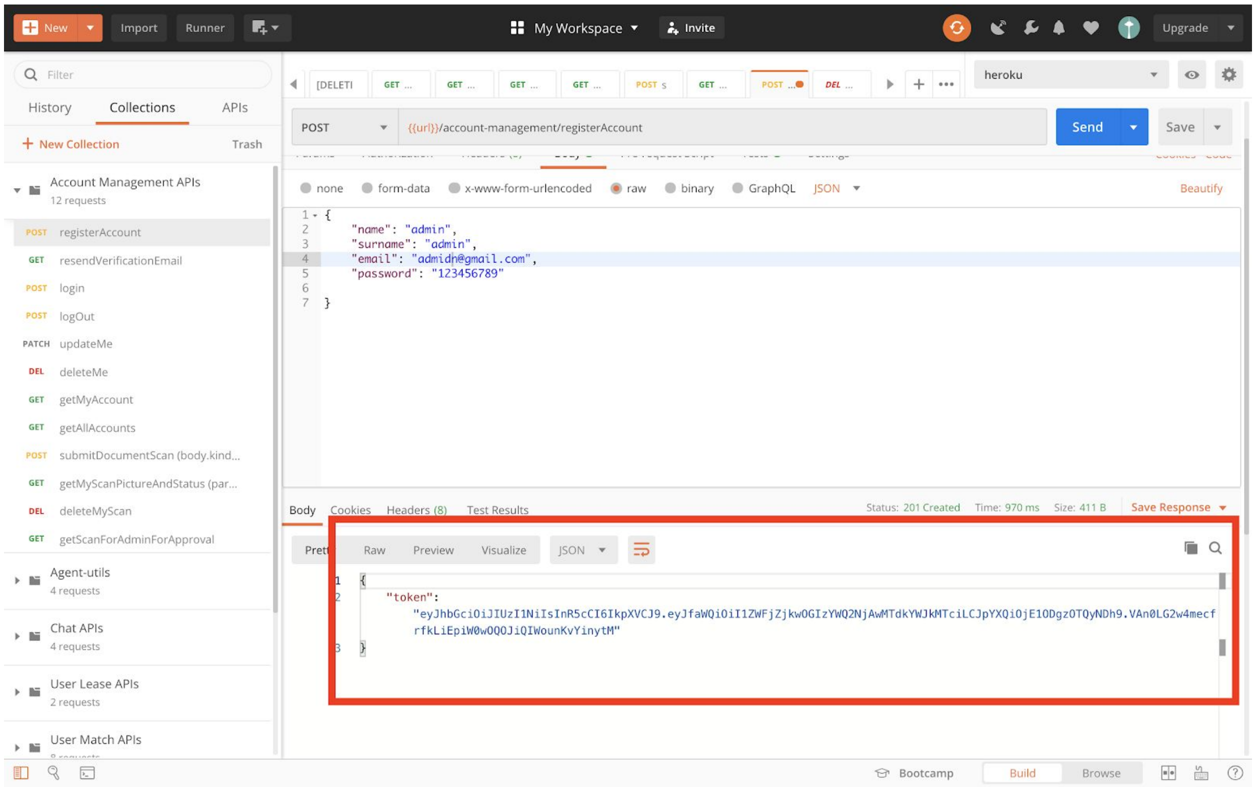

heroku
getScanForAdminForApproval
GET
{{url}}/verification-management/documentScan/admin
Send
Save
Status: 200 OK Time: 2.83 s Size: 629 B

```
{
    "name": "Eldar",
    "surname": "OneKu",
    "accessUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/5eacc172f204ae0017ec18a8",
    "approveUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/approve/5eacc172f204ae0017ec18a8",
    "rejectUrl": "http://roommee.herokuapp.com/verification-management/documentScan/admin/reject/5eacc172f204ae0017ec18a8"
}
```

Setting up browser for testing

Since not all functionality can be tested in Postman it will be necessary to use Chrome browsers for some endpoints. Because many of the endpoints are behind the authentication wall, it will be necessary to set up authentication tokens manually using Chrome extension. Please download and install the chrome extension Modheader to allow access with authorization.

Then, copy the token that is returned back after calling registration or login endpoints.



POST {{url}}/account-management/registerAccount

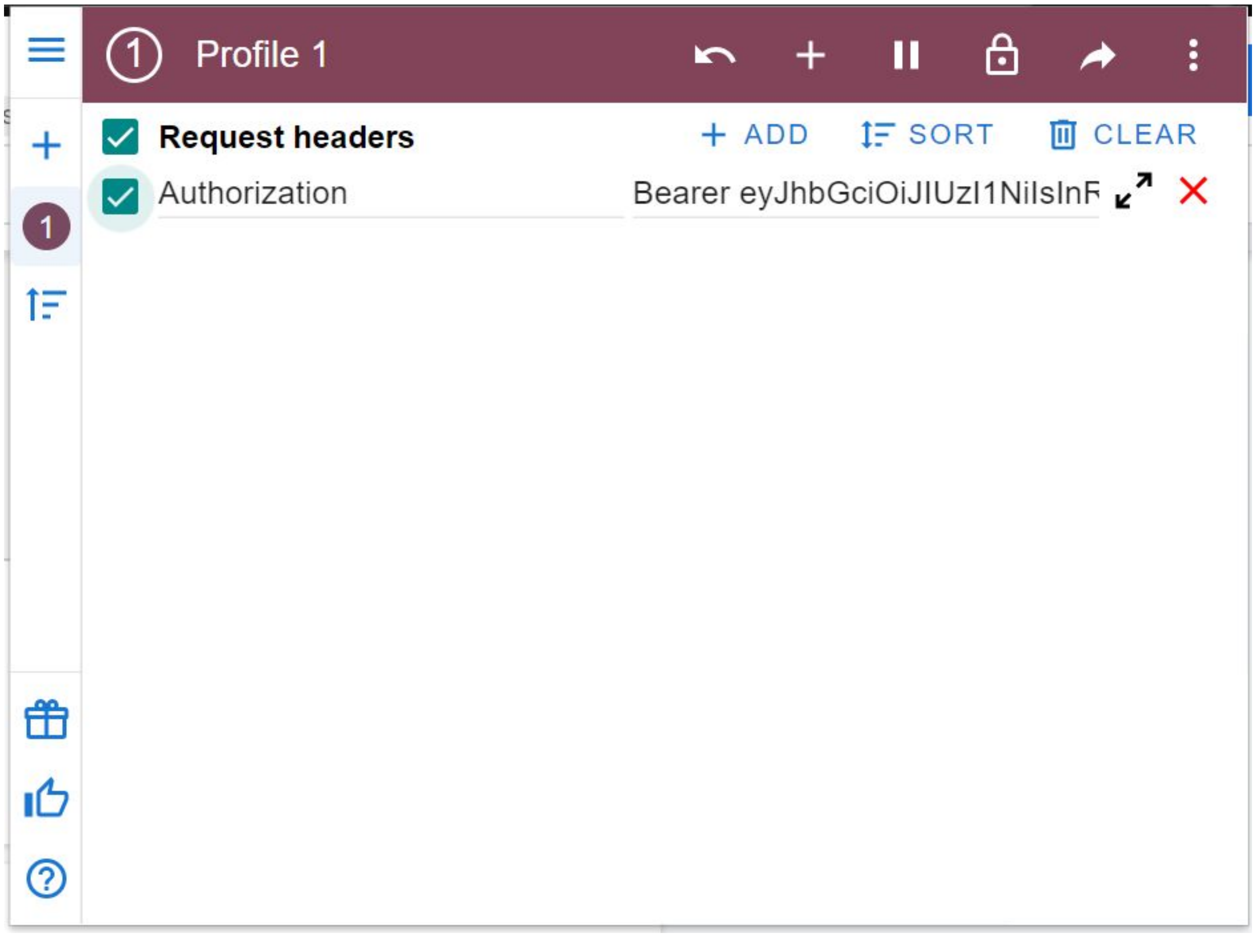
```
{
    "name": "admin",
    "surname": "admin",
    "email": "admidh@gmail.com",
    "password": "123456789"

}
```

Status: 201 Created Time: 970 ms Size: 411 B

```
{
    "token":
        "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1ZWFjZjkwOGIzYWQ2NjAwMTdkYWJkMTciLCJpYXQiOjE1ODgzOTQyNDh9.VAn0LG2w4mecf
        rfkLiEpiW0wOQOJiQIWounKvYinytM"
}
```

Once retrieved, insert the token into the modheader like in the following picture.



Request headers

Authorization | Bearer eyJhbGciOiJIUzI1NiIsInF

Indication of when to use Postman or Chrome for testing

Next to every endpoint there will be indication of whether it will be tested using Postman or Chrome

P: Please test the url in postman
B: Please test the url in browser

Core Functionality

1. Authentication & Verification

APIs:

- {{url}}/account-management/registerAccount **(POST) (P)**
  Registers user. Validates uniqueness of email, strength of password. Password is saved as hashed value, so if DB gets hacked passwords will not leak. Email to verify account is sent. Not verified email does not block a user from using the website, but the fact that the user did not verify his email can be shown to other users. Takes “name”, “surname”, “email”, “password” as input. Outputs authentication token and sends out validation email. In case email that is already in use is entered, the system responds with error in order to prevent duplication of email addresses.

- {{url}}/account-management/accounts/login **(POST) (P)**
  Authenticates user. Takes “email” and “password” as input.Outputs authentication token and user details (testing purpose).

- {{url}}/account-management/accounts/logout **(POST) (P)**
  Log out user. This takes an authentication token as input. Outputs nothing.

- {{url}}/account-management/accounts/me **(PATCH) (P)**
  Changes name/surname/password.
  Takes “name”, “surname”, “email”, “password” and authentication token as input.
  Outputs new profile details (testing purpose).

- {{url}}/account-management/accounts/me **(DELETE) (P)**
  Deletes own account. Takes authentication token as input. Outputs old profile (testing purpose)

- {{url}}/account-management/accounts **(GET) (P)**
  Gets information on profiles of other users. Takes authentication token as input. Outputs accounts of other users.

- {{url}}/account-management/accounts/me **(GET) (P)**
  Gets information on the user's account. Takes authentication token as input. Outputs account information.

- {{url}}/verification-management/documentScan/me/passportUpload **(POST) (P)**

User uploads two copies of the ID document (let's say passport as primary document and driving license as a secondary document) so the administration can compare the registration name to documents and keep a record of user identity.
Takes kindOfDocument with possible inputs of “Secondary” or “Primary”, second input is jpg or png picture also requires authentication token.

- {{url}}/verification-management/verifyEmail/sendAgain **(GET) (B)**
  Resends authentication email to user.
  Takes authentication token as input. Sends to the user with a unique link for confirmation. Accessing the endpoint via postman returns an error which does not prevent sending the email. If accessed via browser there is no problem.

- {{url}}/verification-management/documentScan/me/ **(GET) (P)**
  Users can check what picture he uploaded and what is it’s status.
  Takes authentication token, params : “Secondary” or “Primary” at the end of url after me/ as input. Outputs GET link to the picture (access via browser for authorized user) **(B)**, status of submitted picture (whether it has been approved by admin).

- {{url}}/verification-management/documentScan/me **(DEL) (P)**
  Deletes the submitted document. Takes authentication token as input. Also has a body parameter named “kindOfDocument” that can be equal to “Primary” or “Secondary” to delete the corresponding uploaded picture.

- {{url}}/verification-management/documentScan/admin **(GET) (P)**
  Allows the admin to see one of the documents in the pool of submitted documents by users and approve it or reject it. If called again another document will be available for approval.
  In order to complete, a user needs to be authenticated (logged in) as Admin with following credentials
  "email": "ekurmakaev@student.unimelb.edu.au",
  "password": "123456789"
  Input for the endpoint is an authentication token. Response consists of a GET link to PNG scan (can be viewed only by admin via Chrome) **(B)** and reference to two GET endpoints to approve and disapprove the document (can be accessed via browser **(B)** only by admin).

2. Matching

Note: **By this point, the user should have made the profile and questionnaire (please go to the complementary functionality). Current implementation only enables the user to have 2 roommee max (including the user), as more than 2 will be implemented only if there is more time. *It is recommended that the APIs are tested in this specific order. Also, using 2 window browsers at the same time would make it easier to test the matching part.***

APIs:

- *{{url}}/user-match* **(GET) (B)**
  Get the list of match users. The user could also fill in some preference in case if they want to make sudden changes on their filter 1 preference (for this one, use it in postman **(P)**).

- *{{url}}/user-match/sort-match* **(GET) (P)**
  After we get the match, the match users are listed in sorted order using euclidean distance calculated based on filter 2 data in the questionnaire. At this point, the user could play around with the sort option. Ex: sort based on the nightOwlRate, playsMusicRate, etc.

- *{{url}}/user-match/fill-status* **(POST) (B)**
  After we get the list of match users, we could fill in whether you like the user or not. If like, currently we specify it to ‘yes’ , and ‘no’ for rejection.

- *{{url}}/user-match/status* **(GET) (B)**
  The user could make some changes on the matching status, for example, from ‘yes’ to ‘no’, or vice-versa. The user could also chat with the match users if both users like each other.

  Note: To be able to chat with the match users, the current user needs to click ‘yes’. If the match user also clicked ‘yes’ to the current user, then the chat button will appear in both accounts. Otherwise, ‘pending’ will be shown to indicate that the other match user hasn’t liked the current user profile. This url is implementing the confirmation system.

- *{{url}}/user-match/click-match* **(POST) (P)**
  Click match is done after the chat, when both users decide to move to the next stage and become roommee. In order for both to become roommee, both users need to press the match button. If both have already pressed it, then the system would indicate that both are roommee. If one has not pressed the button, then same like chat, it will result in pending.

Note: This is the last chance that the user could freely cancel their roommate preference. For this deliverable, the tester should indicate the id of the match user and state it to yes (example in postman), as later the id of the match will be placed in the name of the input tag. This also implements the confirmation system.

- *{{url}}/user-match/remove-match* **(POST) (P)**
  This will remove or cancel the clicked match button before the other user clicked the match button.

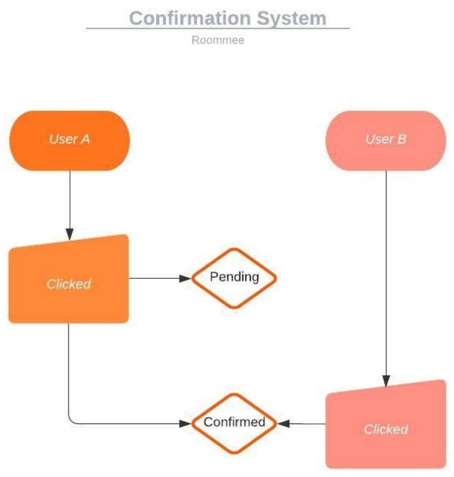
  Note: This button can’t be pressed again if the match user already pressed the button. This is not the removed roommee url, but this will only undo the user click match.

- *{{url}}/user-match/remove-roommee* **(POST) (P)**
  The system would enable the user to change roommee if and only if the other match user would also like to change roommee. They could discuss this in chat before clicking the remove roommee button.

  Note: This url also implements the confirmation system.

- *{{url}}/user-match/roommee* **(GET) (P)**
  The user could get his/her roommee profile.

  This is the schema of the confirmation flow for fill-status, click-match and remove roommee. Thus, it is recommended that testers should use 2 windows with 2 different users logged in to test the matching APIs



3. Chat

Note: To test the chat system, test all of these APIs using browser **(B)**.

APIs:

- *{{url}}/chats/createroom/:room* **(GET) (B)**
  A room will be created in the database and the user will be redirected to the room. Replace (:room) with the roomname that is the appropriate name.

- *{{url}}/chats/chatrooms* **(GET) (B)**
  List of chats the user has currently.

- *{{url}}/chats/chatrooms/:room* **(GET) (B)**
  The room that is created from /chats/createroom/:room

- *{{url}}/chats/delete/:room* **(GET) (B)**
  Delete a ongoing chatroom from the database for both users

4. Agent-utils

- *{{url}}/agent-management* **(GET) (B)**
  Display the list of properties including the informations regarding the agent

- *{{url}}/agent-management/chooseProperty/propertyId* **(POST) (B)**
  When a property in the list is chosen, it will be updated in the database

- *{{url}}/agent-management/setDate* **(POST) (B)**
  Set a start and end date for the lease

- *{{url}}/utils-management* **(GET) (B)**
  Display the list of utility companies

- *{{url}}/utils-management/chooseUtils/utilsId* **(POST) (B)**
  When a utility company is chosen, it will be updated in the database

Complementary Functionality

1. Profile

Note: **Do this before testing the matching APIs. Please test these APIs in browser (B)**

APIs:

- *{{url}}/user-profile/new* **(GET&POST) (B)**
  The GET router is only for redirect purposes. Create new profile after user sign up to roommee.

- *{{url}}/user-profile/* **(GET) (B)**
  Get the current user profile

- *{{url}}/user-profile/update* **(GET&POST) (B)**
  The GET router is only for redirect purposes. Update the current user profile

2. Questionnaire

Note: Note: **Do this before testing the matching APIs. Please test these APIs in browser (B)**

APIs:

- *{{url}}/user-questionaire/new* **(GET&POST) (B)**
  Create a new user questionnaire

- *{{url}}/user-questionaire* **(GET) (B)**
  Get the current user questionnaire

- *{{url}}/user-questionaire/update* **(GET&POST) (B)**
  Update the current user questionaire

3. Property

Note: This property APIs is for users to directly look at other roommee that listed their own property. Each user can only list 1 property max. The user with property will still appear in the match page, and will undergo the same match procedure in order to become a roommee.
Please use postman to test all of these APIs (P)

APIs:

- *{{url}}/user-property/listing* **(POST) (P)**
  This API is to list new property for the current user (can only list 1)

- *{{url}}/user-property/property* **(GET) (P)**
  Get the current user's property profile

- *{{url}}/user-property/match-property/:propertyId* **(GET) (P)**
  Get the other user's property profile. Use this for redirect in the match page and direct property page, but not in deliverable 2

- *{{url}}/user-property/update* **(POST) (P)**
  Update the current user's property profile

- *{{url}}/user-property/remove* **(DELETE) (P)**
  Delete or remove the current user's property profile

- *{{url}}/user-property/list-properties* **(GET) (P)**
  Get all lists of users with their property listed. This will give the current user another option to find a roommee.

- *{{url}}/user-property/search* **(GET) (P)**
  Get the property with specific criteria, for example, weekly rent between 300 to 400. This function ought to be implemented when the user already has a roommee after chat, or when the user decided to go through the second option process, which is to find users that listed their own property. Please look at the postman body section to see how this url can be implemented.

Note: If a match happens between users with a property, then they will skip the choosing property and utility stage and link directly to choosing the date start and end.

4. Lease

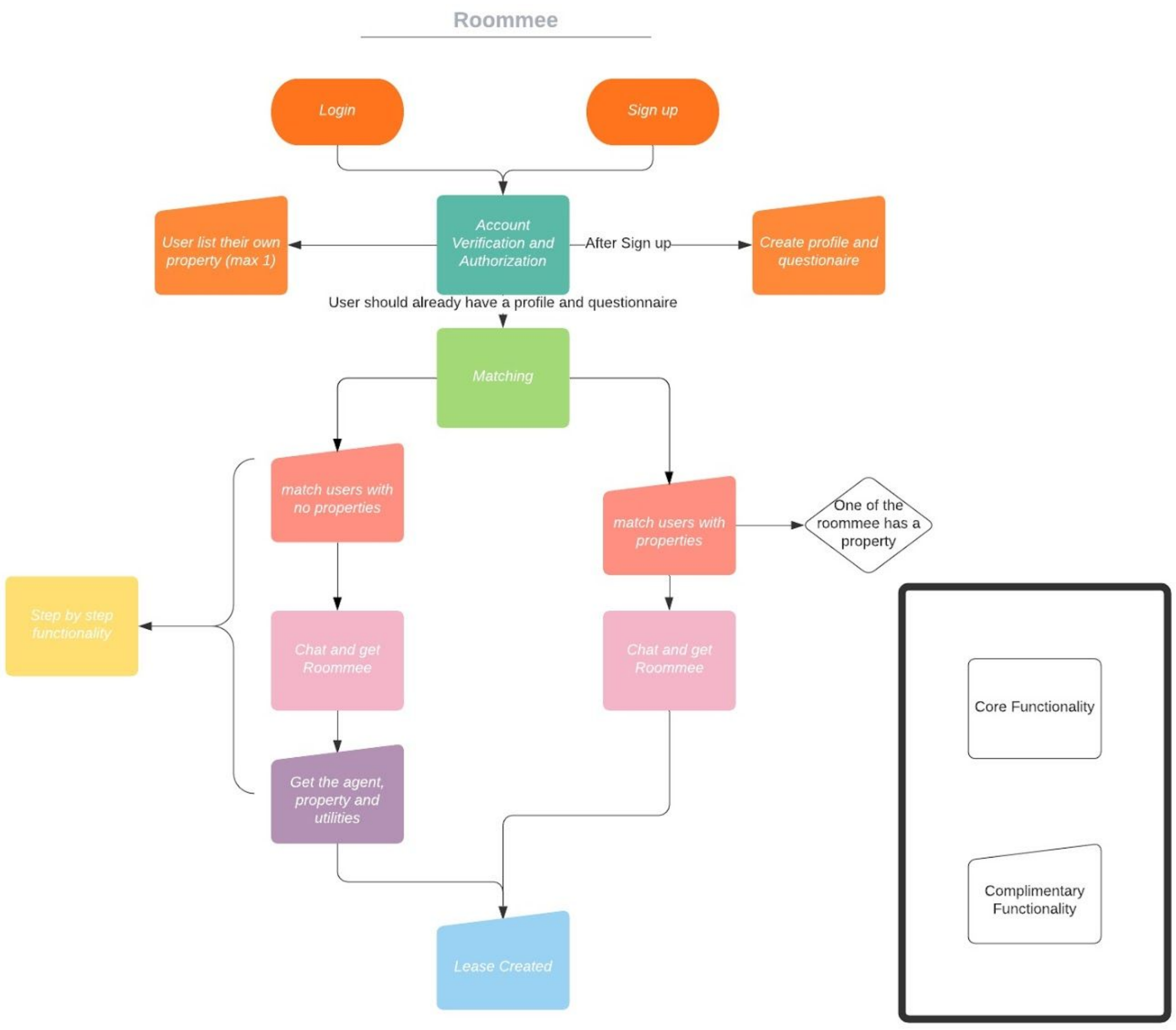
Note: Test these APIs when everything is finalised.

APIs:

- *{{url}}/user-lease/lease* **(GET) (B)**
  View the lease that has been created during the process

- *{{url}}/user-lease/enqEmail* (**POST) (P)**
  The current user could make a request for roommee to change or amend their lease if they think there is a mistake. The user would type their enquiries, and the system would send an email to the user, and also to the roommee email account (not yet created). To the user, the system would send an automated text saying that one of roommee's representative will contact them shortly through email. To the roommee email, the text will be the user's enquiries.

Website Flow:


Roommee
Login
Sign up
Account Verification and Authorization
User list their own property (max 1)
After Sign up
Create profile and questionaire
User should already have a profile and questionnaire
Matching
match users with no properties
match users with properties
One of the roommee has a property
Step by step functionality
Chat and get Roommee
Chat and get Roommee
Get the agent, property and utilities
Lease Created
Core Functionality
Complimentary Functionality