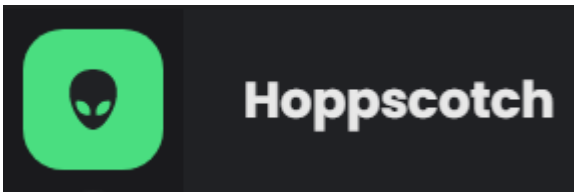
Black box testing:

Tool Used : Hoppscotch



We used hoppscotch as our API testing client, which provides response to given API requests. It is an open source tool for API developments and testing. So for black-box testing we used this tool as we can use it without knowing any source code. This tool provides a simple interface where you can make API requests directly to the server without needing any front end.

1. Register a new user

(POST : https://naivebakerr.herokuapp.com/user/register)

Fields:

- Name
  - all names are valid so 1 equivalent class containing all possible names
- Email
  - Valid email address
  - Invalid email address
- Password
  - Valid password has length in [8,16]
  - Password length less than 8 (Invalid)
  - Password length greater than 16 (Invalid)
  - Password length in between 8 and 16 (Valid)

So for equivalence classes,

Name has 1 class n1
Email has 2 class e1,e2
Password has 3 class p1,p2,p3

For equivalence partitioning, we will have total **6 test case** scenarios

Equivalence	Name	Email	Password	Verdict
n1 e1 p1	John	john@gmail.com	Length < 8	Invalid password
n1 e1 p2	Jenna	jenna@gmail.com	8 <= Length <= 16	Success
n1 e1 p3	James	james@gmail.com	Length > 16	Invalid password
n1 e2 p1	John	john@	Length < 8	Invalid password,email
n1 e2 p2	Jenna	gmail.com	8 <= Length <= 16	Invalid email
n1 e2 p3	James	james.com	Length > 16	Invalid password,email

2. User Login

(POST: https://naivebakerr.herokuapp.com/user/login)

Fields:

- Email
    - Correct email address
    - Incorrect email address
- Password
    - Correct password
    - Incorrect password

So for equivalence classes,
Email has 2 class e1,e2
Password has 2 class p1,p2

For equivalence partitioning, we will have total **4 test case** scenarios

Equivalence	Email	Password	Verdict
e1 p1	john@gmail.com	Correct password	Successful login
e1 p2	jenna@gmail.com	InCorrect password	Incorrect password
e2 p1	john@	Correct password	Invalid email
e2 p2	gmail.com	Incorrect password	Invalid email , Incorrect password

3. Fetch User details

(GET : https://naivebakerr.herokuapp.com/user/detail)

Users must be authorized to fetch user details hence there can be 2 possible scenarios for fetching user details.

If user is authorized then it will send user details else API will give error message showing “please login again”

4. Search For a Recipe

(POST : https://naivebakerr.herokuapp.com/query/search)

Available options:
Search by ingredients
Search by chef name
Search by meal type (breakfast, luch, dinner, snack)
Search by cuisine type (italian , american, indian, chinese etc.)
Search by preparation time (minutes)
Search by category (veg , non-veg , vegan)

Request Body	Response
Ingredients : [“flour” , “egg”] Meal type: dinner Cuisine type: american Category : Non-Veg	[Cake]
Meal type: dinner Cuisine type: Italian Category : Veg	[Pasta, Lasagna]
Meal type: dinner Cuisine type: american Category : Non-Veg Preparation time : 30	[Burger, Mashed potatoes, Hotdog]
Ingredients : [“rice” ,”paneer” , “onion”] Meal type: Lunch Cuisine type: indian Chef name : Sanjeev kapoor	[Hyderabadi Biryani, Fried Rice]

5. Fetch Another User's profile

(GET : https://naivebakerr.herokuapp.com/user/profile/id)

Any user can see another user's public profile hence only 1 equivalence class which contains a successful response.

6. Fetch All Recipe details

(GET : https://naivebakerr.herokuapp.com/recipe/all)

Any user can see all recipes hence only 1 equivalence class which contains a successful response having an array of all recipes.
(similarly user can fetch all chef names and all ingredients as well)

7. Upload a Recipe

(POST : https://naivebakerr.herokuapp.com/recipe/upload)

Users must be authorized to upload recipe details.
All necessary fields should be filled.

So we will have 2 equivalence class in total,
Authorization will have a1,a2
Necessary fields will have n1,n2

hence there can be 4 possible scenarios for uploading user details.

For equivalence partitioning, we will have total **4 test case** scenarios

Equivalence	Authorization	Necessary Fields	Verdict
a1 n1	Yes	Complete	Successful upload
a1 n2	Yes	Missing	All fields are not filled
a2 n1	No	Complete	Unauthorized user
a2 n2	No	Missing	Unauthorized user, All fields are not filled

8. Forgot Password

To reset forgotten password users should send requests for password change first which will send OTP in response.

Then the user will enter this OTP and new password and make another request for password reset, then password will be changed. User can only change password from the device which he requested OTP that is authenticated by token.

So we can have following scenarios,

1) OTP (valid / invalid)
2) Email address (valid/invalid)
3) New password (valid/invalid)

Here each field has 2 equivalence classes (valid/invalid) , so total of 8 test cases

OTP	Email	New Password	Verdict
Valid	Valid	Valid	**Success**
Valid	Valid	Invalid	Error
Valid	Invalid	Valid	Error
Valid	Invalid	Invalid	Error
Invalid	Valid	Valid	Error
Invalid	Valid	Invalid	Error
Invalid	Invalid	Valid	Error
Invalid	Invalid	Invalid	Error