

Name: Ng Shi Wei

Admission No.: A0185450E

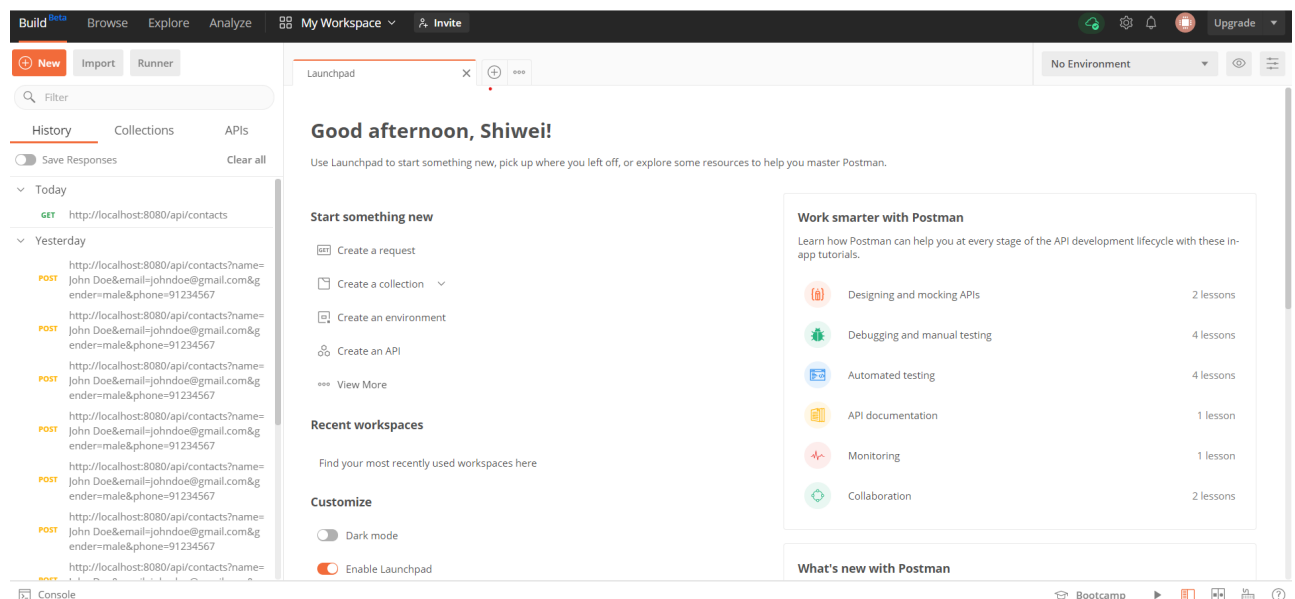
Instructions

Run API locally (B1)

1. Go to <https://github.com/shiweing/rest-api>
2. Clone the repository locally
3. Open the command prompt in the repository folder
4. Enter the command `node index.js`
5. Open a browser and go to <http://localhost:8080>
6. The page should state **Welcome to the world of Pokemons!**

Accessing API with Postman (B1)

1. Go to <https://www.postman.com/>
2. Sign up for an account and launch the workspace



GET API call

1. Select **GET** as the request type and enter <http://localhost:8080/api/pokemons> as url

2. Click **Send**

GET <http://localhost:8080/api/pokemons> **Send**

Params Authorization Headers Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (2) Test Results Status: 200 OK Time: 32 ms Size: 160 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "success",
3   "message": "Pokemons retrieved successfully",
4   "data": []
5 }
```

3. The request result will be shown

If an error appears like the following image

Response

Could not send request

CORS Error: The request has been blocked because of the CORS policy | [Use Postman Desktop Agent](#)

[Learn more about troubleshooting API requests](#)

download the chrome plugin to unblock CORS when testing at <https://chrome.google.com/webstore/detail/cors-unblock/lfhmikememgdcahcdlaciloanbchjino?hl=en>

POST API call

1. Select **POST** as the request type and enter `http://localhost:8080/api/pokemons` as url
2. Open the body tab and select **x-www-form-urlencoded**
3. Add the parameters as shown below
4. Click **Send**

5. A response indicating that the pokemon has been added will be returned.

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

- Method:** POST (highlighted with a red box)
- URL:** `http://localhost:8080/api/pokemons`
- Body:** The 'Body' tab is selected. The format is 'x-www-form-urlencoded'. A table contains the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Bulbasaur	
<input checked="" type="checkbox"/> id	001	
- Response:** The 'Body' tab shows the JSON response:

```
1 {
2   "message": "New pokemon added!",
3   "data": {
4     "_id": 1,
5     "name": "Bulbasaur",
6     "__v": 0
7   }
8 }
```
- Status:** 200 OK, Time: 104 ms, Size: 162 B

6. Run the GET request to see the new contact being returned

PUT API call

1. Run the GET request and copy the id of the pokemon that was just added.
2. Select **PUT** as the request type and enter `http://localhost:8080/api/pokemons/[id]` as url (where `[id]` is the id that was copied)
3. Change the value of the name parameter under body
4. Click **Send**
5. A response indicating that the contact has been updated will be returned.

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

- Method:** PUT (highlighted with a red box)
- URL:** `http://localhost:8080/api/pokemons/1` (the `1` is highlighted with a red box)
- Body:** The 'Body' tab is selected. The format is 'x-www-form-urlencoded'. A table contains the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Bulbasaurus	
- Response:** The 'Body' tab shows the JSON response:

```
1 {
2   "message": "Pokemon Info updated",
3   "data": {
4     "_id": 1,
5     "name": "Bulbasaurus",
6     "__v": 0
7   }
8 }
```
- Status:** 200 OK, Time: 94 ms, Size: 166 B

6. Run the GET request to see the contact has been updated

DELETE API call

1. Run the GET request and copy the id of the pokemon that was just added.
2. Select **DELETE** as the request type and enter `http://localhost:8080/api/pokemons/[id]` as url (where `[id]` is the id that was copied)
3. Click **Send**

4. A response indicating that the pokemon has been deleted will be returned.

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

- Method:** DELETE (highlighted with a red box)
- URL:** http://localhost:8080/api/pokemon/1 (the `/1` is highlighted with a red box)
- Send Button:** A blue button labeled "Send".
- Params:** A tab labeled "Params" with a green dot, indicating it is selected.
- Query Params:** A table with columns KEY, VALUE, and DESCRIPTION.

KEY	VALUE	DESCRIPTION
create_date		
Key	Value	Description
- Body:** A tab labeled "Body" with a red underline, indicating it is selected.
- Response:** A JSON object: `{ "status": "success", "message": "Pokemon deleted" }`. The response is displayed in a "Pretty" view.
- Status:** 200 OK, Time: 32 ms, Size: 475 B.

5. Run the GET request to see the contact has been deleted

Testing (B2)

- Test cases are written with mocha and chai-http.

Testing locally

- Run `npm run test` on a local copy of the application to run the tests locally.

Automated testing

- Travis is used as the CI tool to automate testing
- The command `mocha --exit` is added to `.travis.yml` to initialise the testing
- Below is the results of the travis build

```
Running RestHub on port 8080
/api/pokemons
  ✓ GET (45ms)
  ✓ POST

/api/pokemons/:id
  ✓ GET
  ✓ PUT
(node:5316) DeprecationWarning: collection.remove is deprecated. Use deleteOne, deleteMany, or bulkWrite instead.
{ status: 'success', message: 'Pokemon deleted' }
  ✓ DELETE

5 passing (171ms)

The command "mocha --exit" exited with 0.
```