Name: Ng Shi Wei

Admission No.: A0185450E

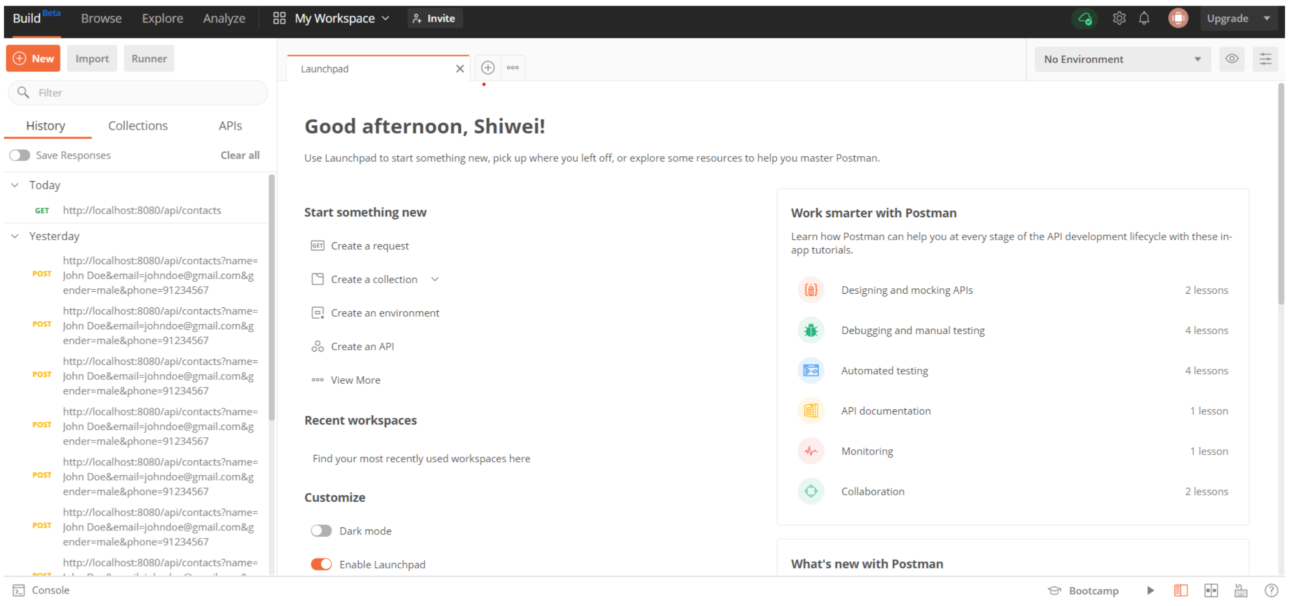
Instructions

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 & B3)

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



Today
GET http://localhost:8080/api/contacts
Yesterday
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
POST http://localhost:8080/api/contacts?name=John Doe&email=johndoe@gmail.com&gender=male&phone=91234567
http://localhost:8080/api/contacts?name=

Good afternoon, Shiwei!

Use Launchpad to start something new, pick up where you left off, or explore some resources to help you master Postman.

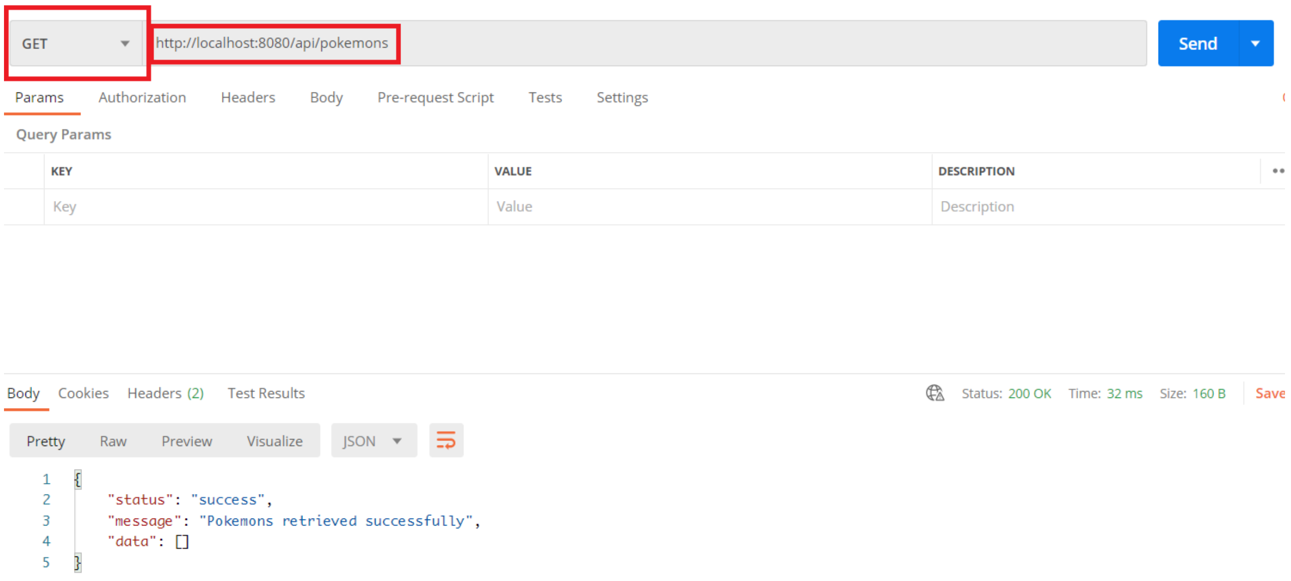
Find your most recently used workspaces here

Learn how Postman can help you at every stage of the API development lifecycle with these in-app tutorials.

GET API call

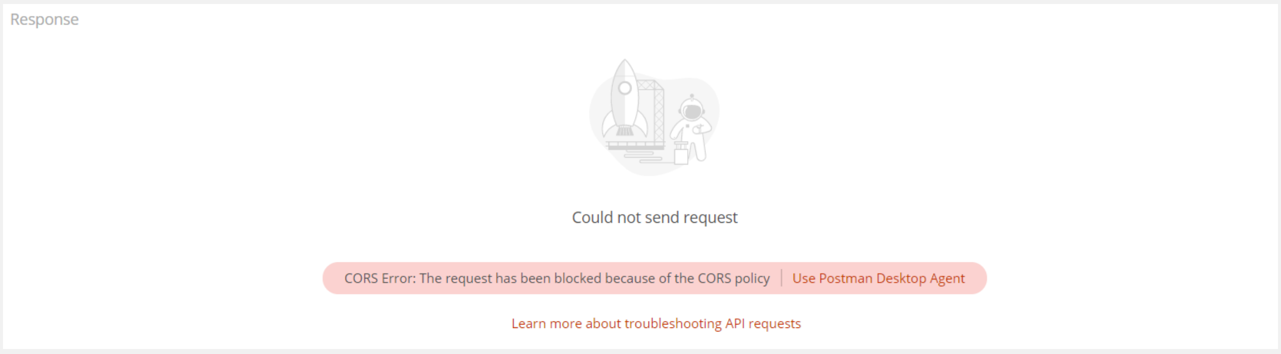
1. Select **GET** as the request type and enter `http://localhost:8080/api/pokemons` as url

2. Click **Send**



```
{
    "status": "success",
    "message": "Pokemons retrieved successfully",
    "data": []
}
```

3. The request result will be shown

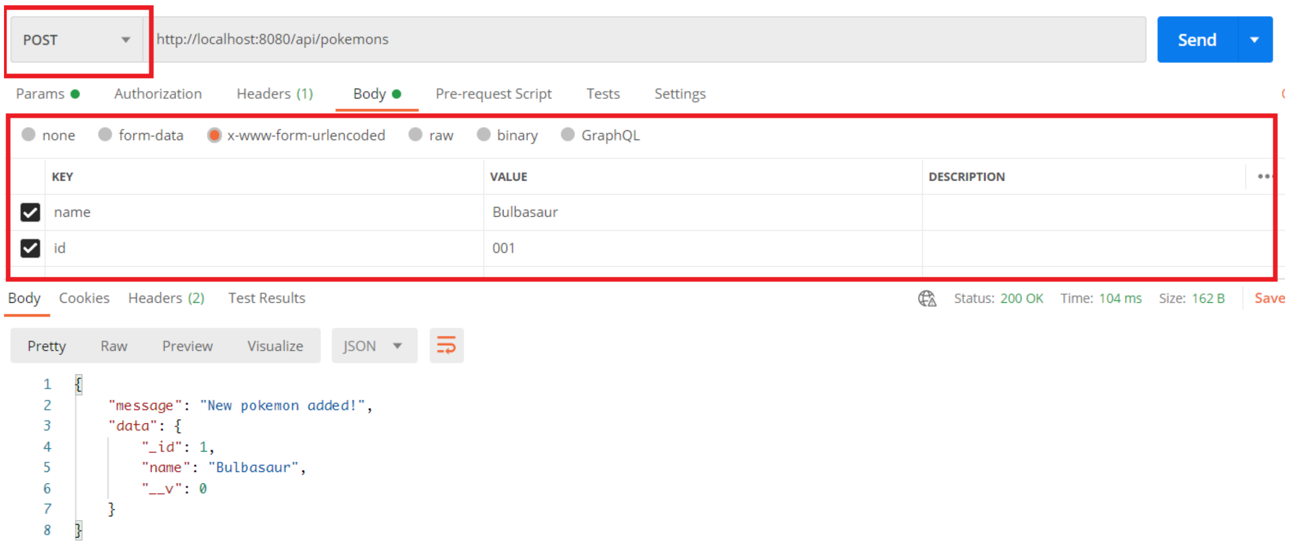
> If an error appears like the following image
>
> 
>

Response
> Use Postman Desktop Agent
> Learn more about troubleshooting API requests
>
> Could not send request
>
> CORS Error: The request has been blocked because of the CORS policy
>
> download the chrome plugin to unblock CORS when testing at https://chrome.google.com/webstore/detail/cors-unblock/lfhmikememgdcahcdlaciloancbhjino?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.



POST http://localhost:8080/api/pokemons

x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
✓ name	Bulbasaur	
✓ id	001	

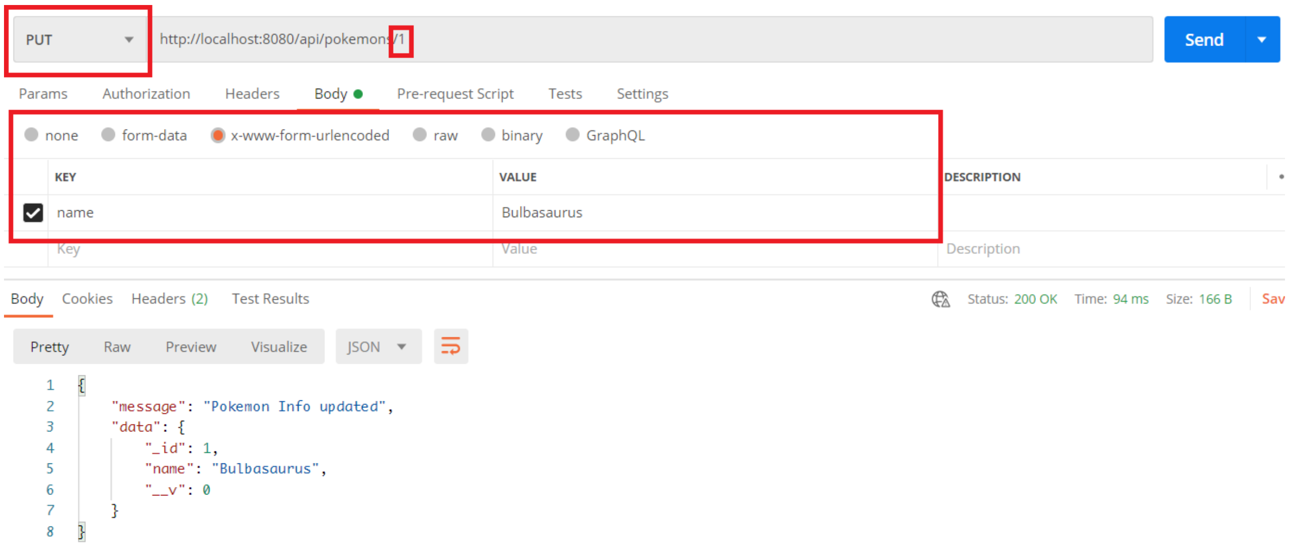
Status: 200 OK Time: 104 ms Size: 162 B

```
{
    "message": "New pokemon added!",
    "data": {
        "_id": 1,
        "name": "Bulbasaur",
        "__v": 0
    }
}
```

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.



PUT http://localhost:8080/api/pokemons/1

x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
✓ name	Bulbasaurus	

Status: 200 OK Time: 94 ms Size: 166 B

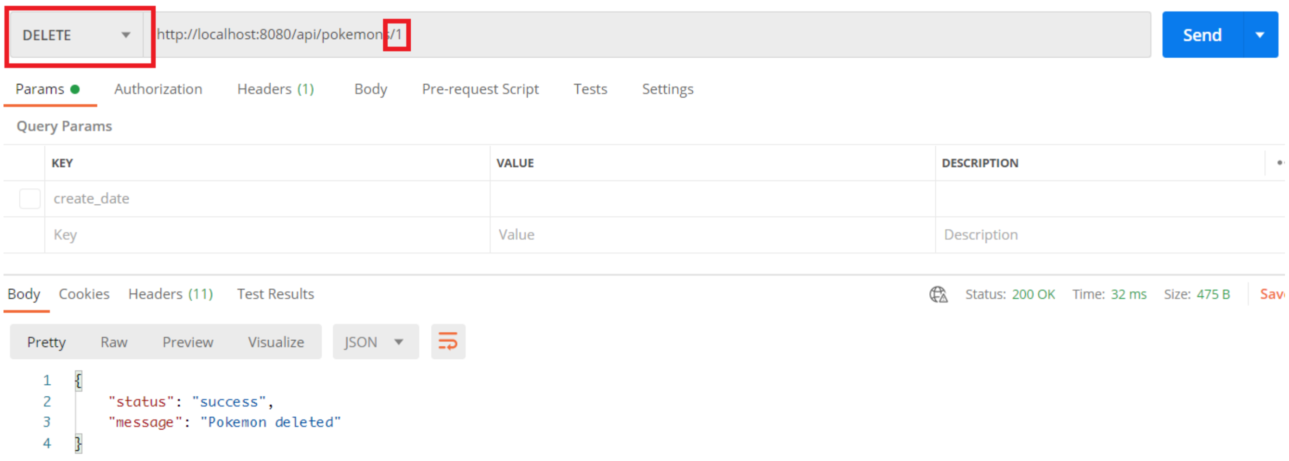
```
{
    "message": "Pokemon Info updated",
    "data": {
        "_id": 1,
        "name": "Bulbasaurus",
        "__v": 0
    }
}
```

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.



```
{
    "status": "success",
    "message": "Pokemon deleted"
}
```

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

> ⚠ Replace `http://localhost:8080` with `https://pokemon-rest-app.herokuapp.com/` to test the deployed endpoint

Testing (B2)

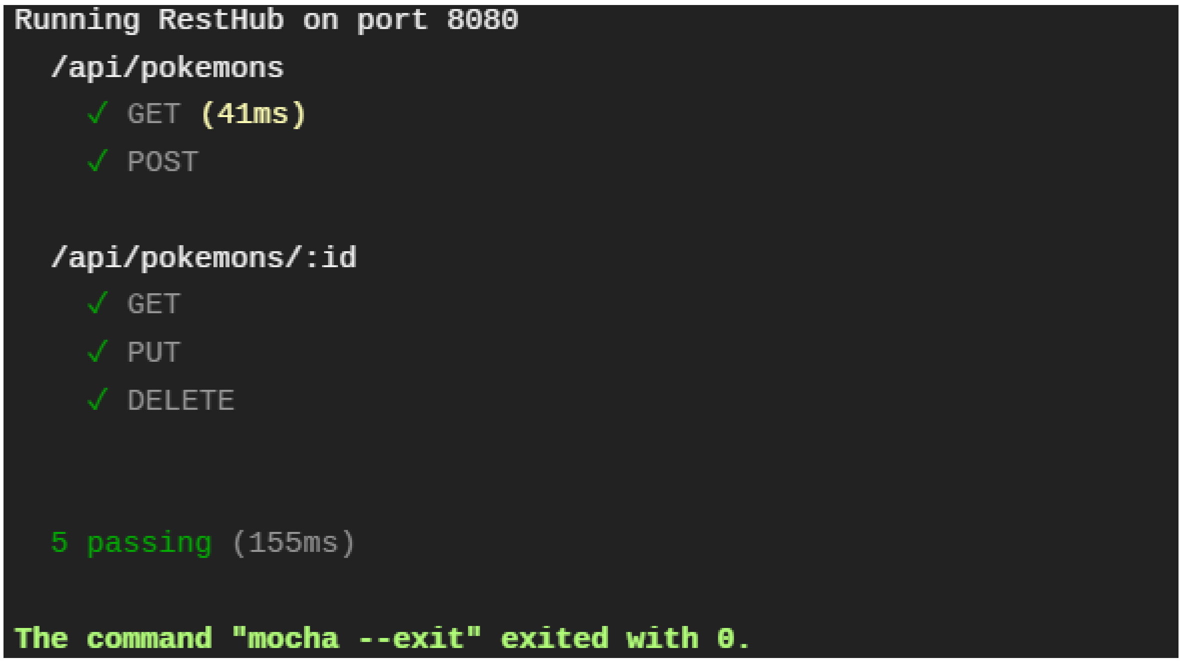
- Test cases are written with mocha and chai-http.

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 (41ms)
    ✓ POST

  /api/pokemons/:id
    ✓ GET
    ✓ PUT
    ✓ DELETE



  5 passing (155ms)

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

Continuus deployment (B3)

- Heroku was chosen as the cloud service for deployment.

.travis.yml

- The follwoing was appended to `.travs.yml`

```
deployment:
    provider: heroku
    api-key:
        secure: <encrypted-api-key>
    app: <heroku-app-name>
    on:
        repo: <repo-path>
```

- To get the api key, install travis and heroku command line clients
- Run `heroku auth:token` to get the api key.
- Copy the api key and run `travis encrypt <api-key> --add deploy.api_key`
- The encrypted api-key will be generated

MongoDB

- To utilise mongodb on Heroku, MongoDB Atlas is required.
- Create an account on MongoDB Atlas, and create a new project
- Select `Build a cluster` on the new project
  - Select a Cloud Provider
  - Select a Region
  - Enter a name for the cluster
- After the cluster is created, click on `Database Access` in the left menu
- Select `Add a new user`
  - Enter a username and password
  - Set the `User Privileges` to `Read and write to any database`
  - Save the settings
- Click on `Network Access` in the left menu
- Click on `Add IP Address`
  - `Allow Access from Anywhere` is selected for this app, but for an IP address should be specified for a secure deployment
- Click on `Clusters`
- Click `Connect > Connect Your Application`
  - A connection string will be displayed, copy the string
- Go to the Heroku dashboard for the application, under `Settings`
  - Add a config variable `MONGODB_URI: <connection string>`
  - Replace `<password>` in the connection string with the password set for the created DB user.