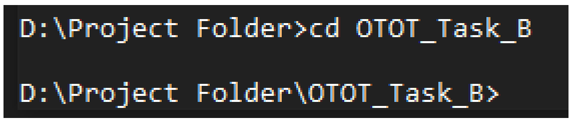
Student Name: Tan Boon Ji

Matriculation Number: A0200362Y

Link to Github repository:
https://github.com/tanboonji/OTOT_Task_B

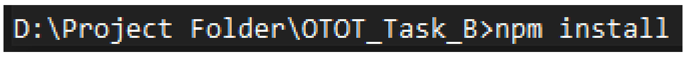
Instructions on how to run the API locally

1. On cmd, navigate your directory to the `OTOT_Task_B` folder.



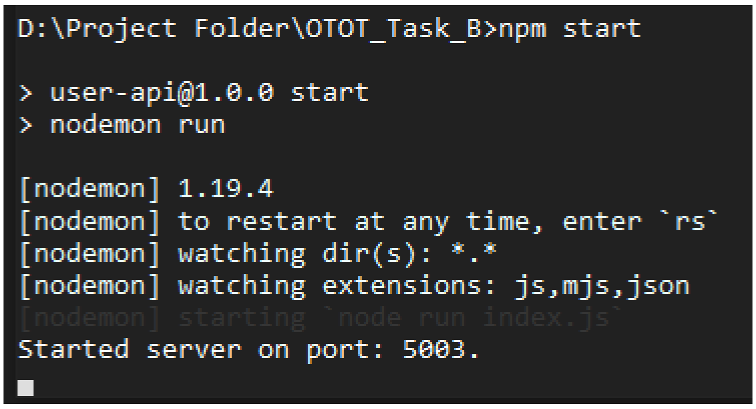
```
D:\Project Folder>cd OTOT_Task_B

D:\Project Folder\OTOT_Task_B>
```

2. Install the required packages by running the command `npm install`, then wait for the packages to be installed.



```
D:\Project Folder\OTOT_Task_B>npm install
```

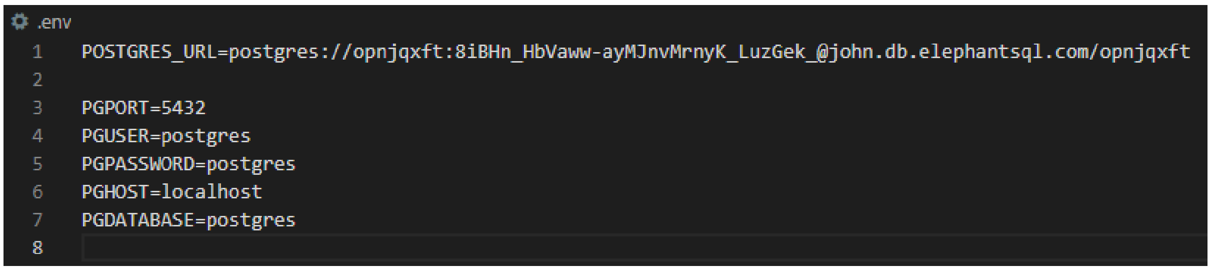
3. Start the server by running the command `npm start`.



```
D:\Project Folder\OTOT_Task_B>npm start

> user-api@1.0.0 start
> nodemon run

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node run index.js`
Started server on port: 5003.
```

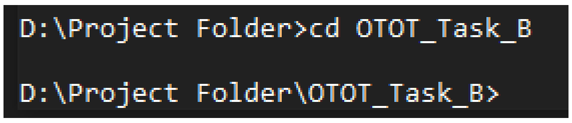
You should see `Started server on port: 5003.` once the server has successfully started.

4. The server retrieves data from an online database called ElephantSQL, therefore there should already be some initial data in the online database. The PostgreSQL URL for the online database is already provided in the `.env` file.



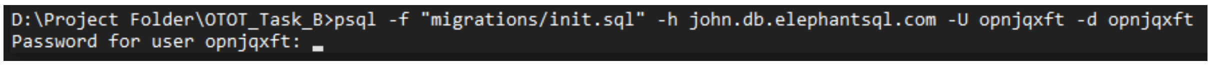
```
.env
POSTGRES_URL=postgres://opnjqxft:8iBHn_HbVaww-ayMJnvMrnyK_LuzGek_@john.db.elephantsql.com/opnjqxft

PGPORT=5432
PGUSER=postgres
PGPASSWORD=postgres
PGHOST=localhost
PGDATABASE=postgres
```

Note: Should there no data in the online database, you can run the following command in another cmd to reset the database. Similarly, your cmd needs to be navigated to the directory `OTOT_Task_B` folder first. (Ensure you have PostgreSQL installed on your computer to have the `psql` command in cmd)



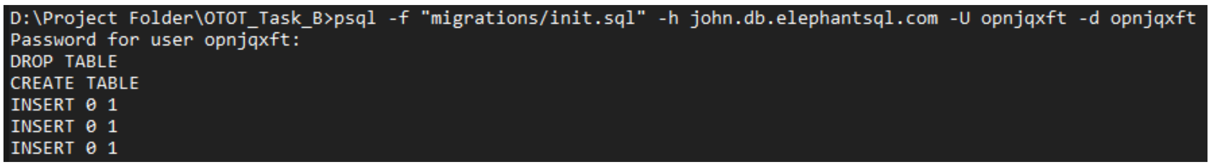
```
D:\Project Folder>cd OTOT_Task_B

D:\Project Folder\OTOT_Task_B>
```

`psql -f "migrations/init.sql" -h john.db.elephantsql.com -U opnjqxft -d opnjqxft`



```
D:\Project Folder\OTOT_Task_B>psql -f "migrations/init.sql" -h john.db.elephantsql.com -U opnjqxft -d opnjqxft
Password for user opnjqxft: _
```

You will be prompted to enter the password for user opnjqxft. The password is `8iBHn_HbVaww-ayMJnvMrnyK_LuzGek_`.

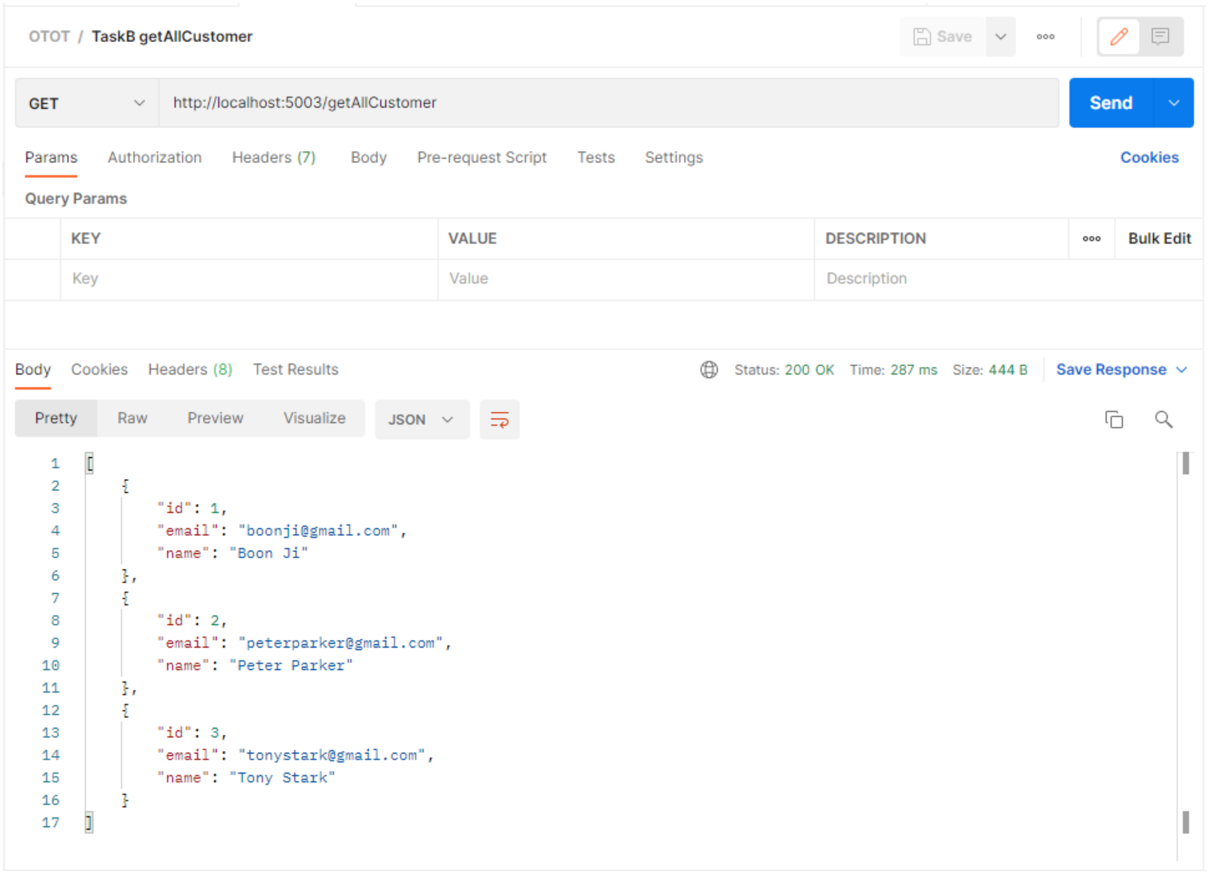


```
D:\Project Folder\OTOT_Task_B>psql -f "migrations/init.sql" -h john.db.elephantsql.com -U opnjqxft -d opnjqxft
Password for user opnjqxft:
DROP TABLE
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

5. Once the server has started, you can call the API endpoints locally.

<u>Screenshots of Postman calls to API locally</u>

1. `http://localhost:5003/getAllCustomer`

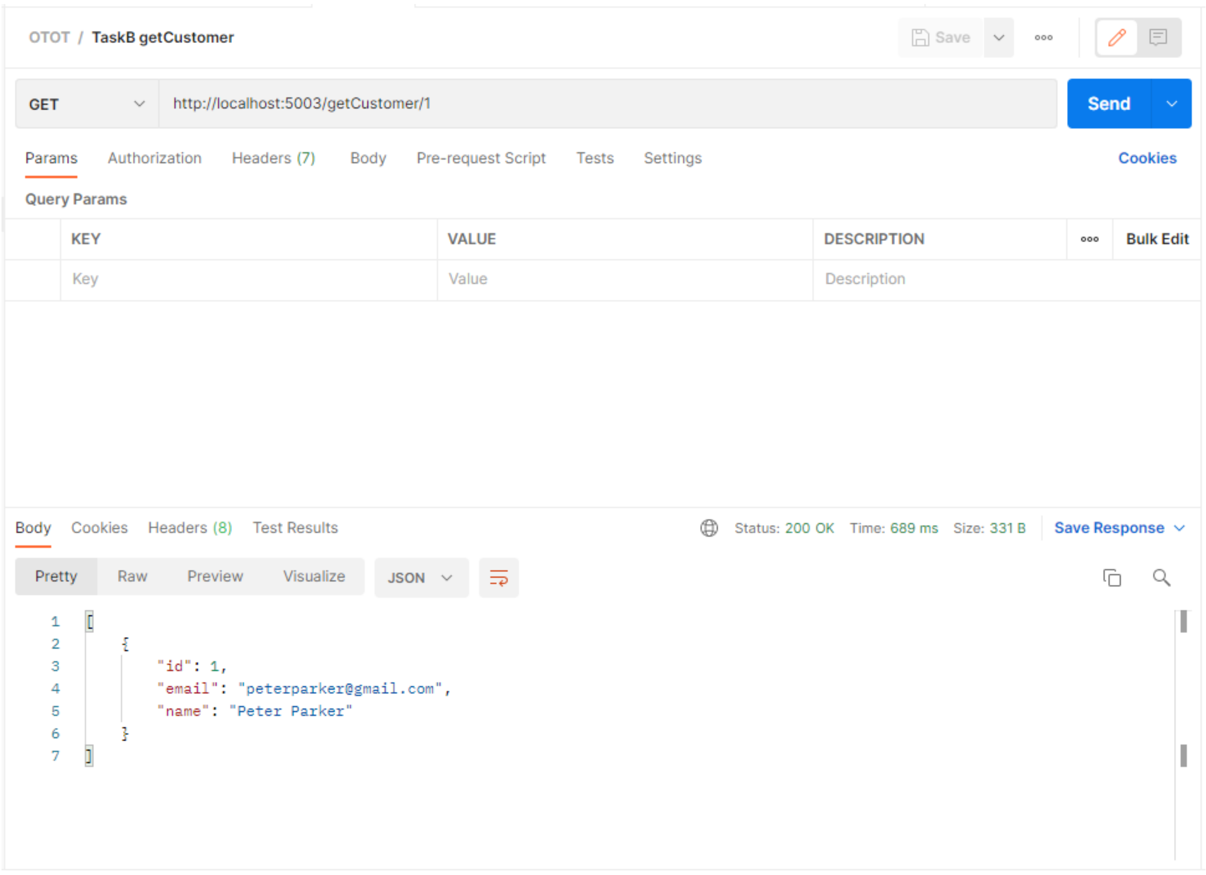


GET http://localhost:5003/getAllCustomer

Status: 200 OK Time: 287 ms Size: 444 B

```
[
    {
        "id": 1,
        "email": "boonji@gmail.com",
        "name": "Boon Ji"
    },
    {
        "id": 2,
        "email": "peterparker@gmail.com",
        "name": "Peter Parker"
    },
    {
        "id": 3,
        "email": "tonystark@gmail.com",
        "name": "Tony Stark"
    }
]
```

2. `http://localhost:5003/getCustomer/:id`



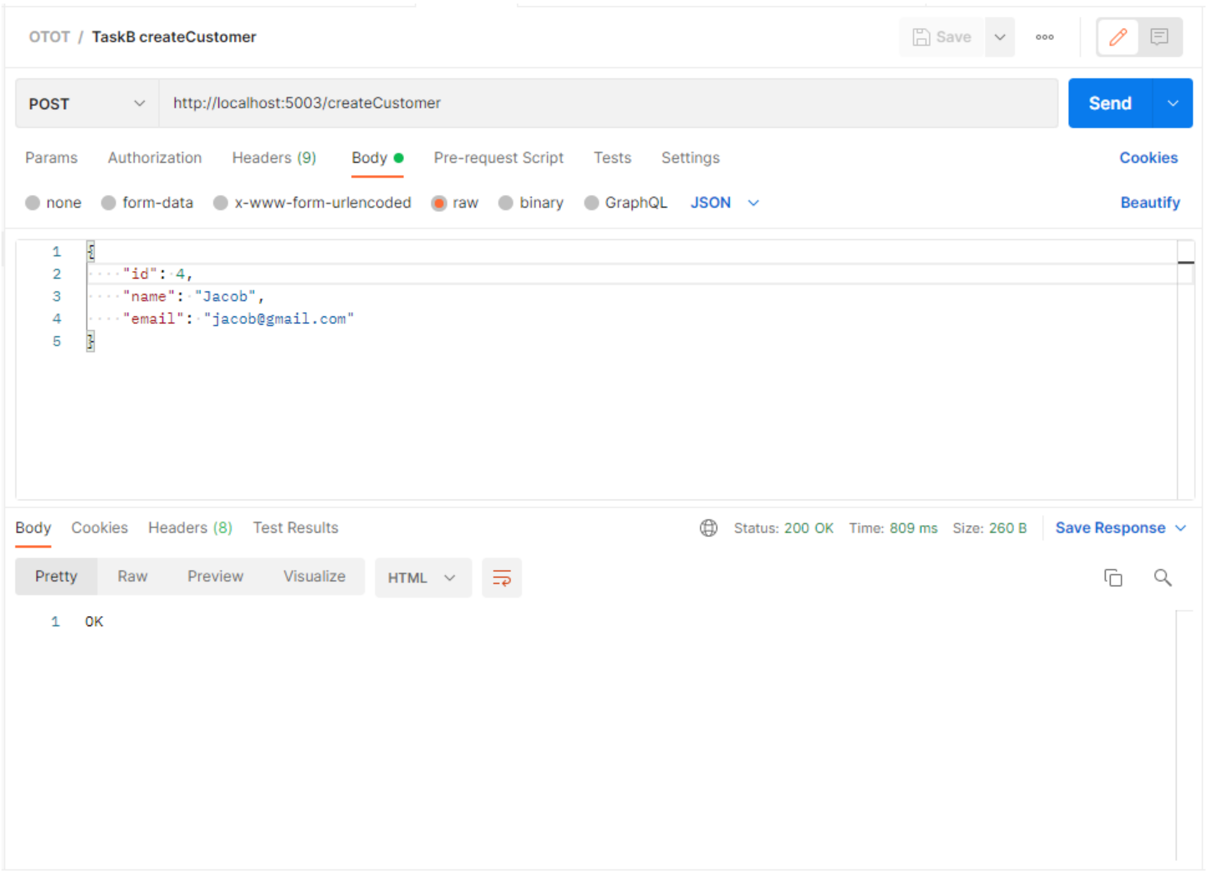
OTOT / TaskB getCustomer

GET http://localhost:5003/getCustomer/1

Status: 200 OK Time: 689 ms Size: 331 B

```
[
    {
        "id": 1,
        "email": "peterparker@gmail.com",
        "name": "Peter Parker"
    }
]
```

3. `http://localhost:5003/createCustomer`



OTOT / TaskB createCustomer

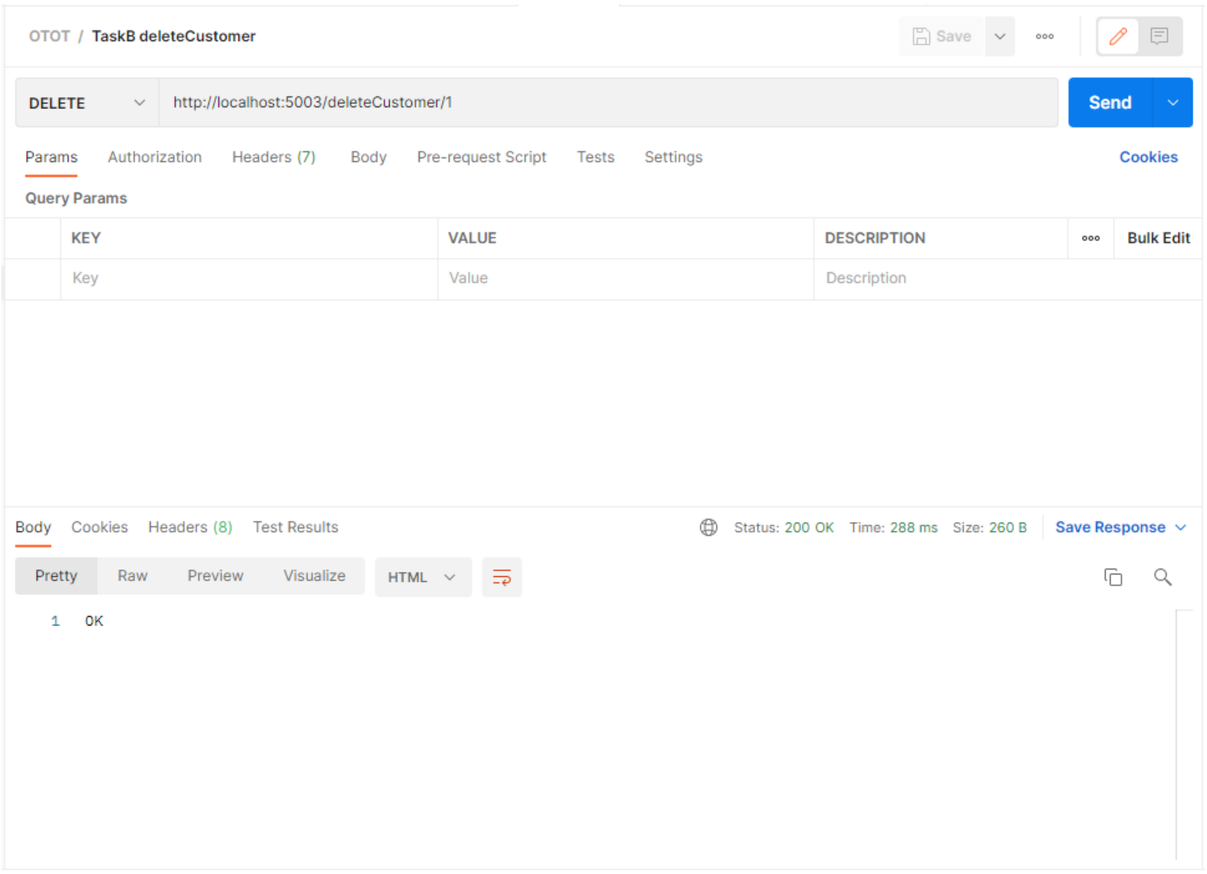
POST http://localhost:5003/createCustomer

```
{
    "id": 4,
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

Status: 200 OK Time: 809 ms Size: 260 B

```
OK
```

4. `http://localhost:5003/deleteCustomer/:id`


OTOT / TaskB deleteCustomer

DELETE http://localhost:5003/deleteCustomer/1

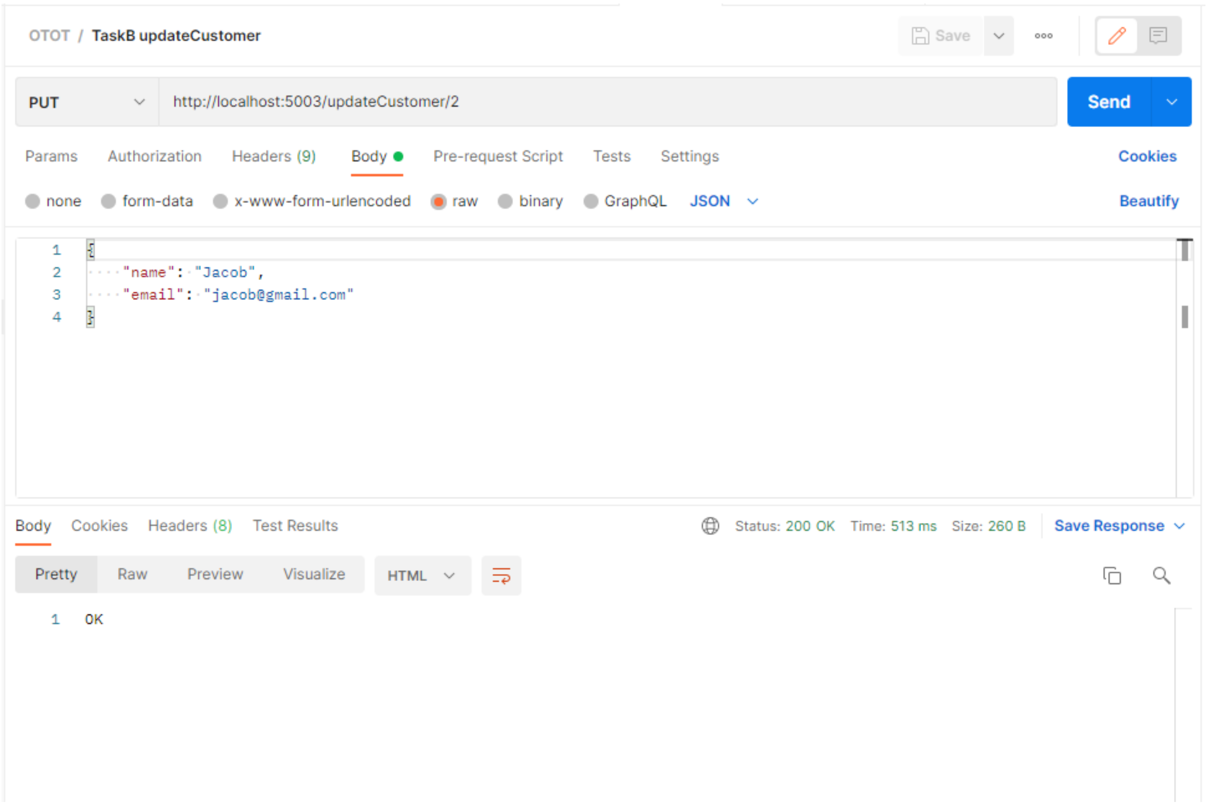
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 288 ms Size: 260 B

```
1  OK
```

5. `http://localhost:5003/updateCustomer/:id`


OTOT / TaskB updateCustomer

PUT http://localhost:5003/updateCustomer/2

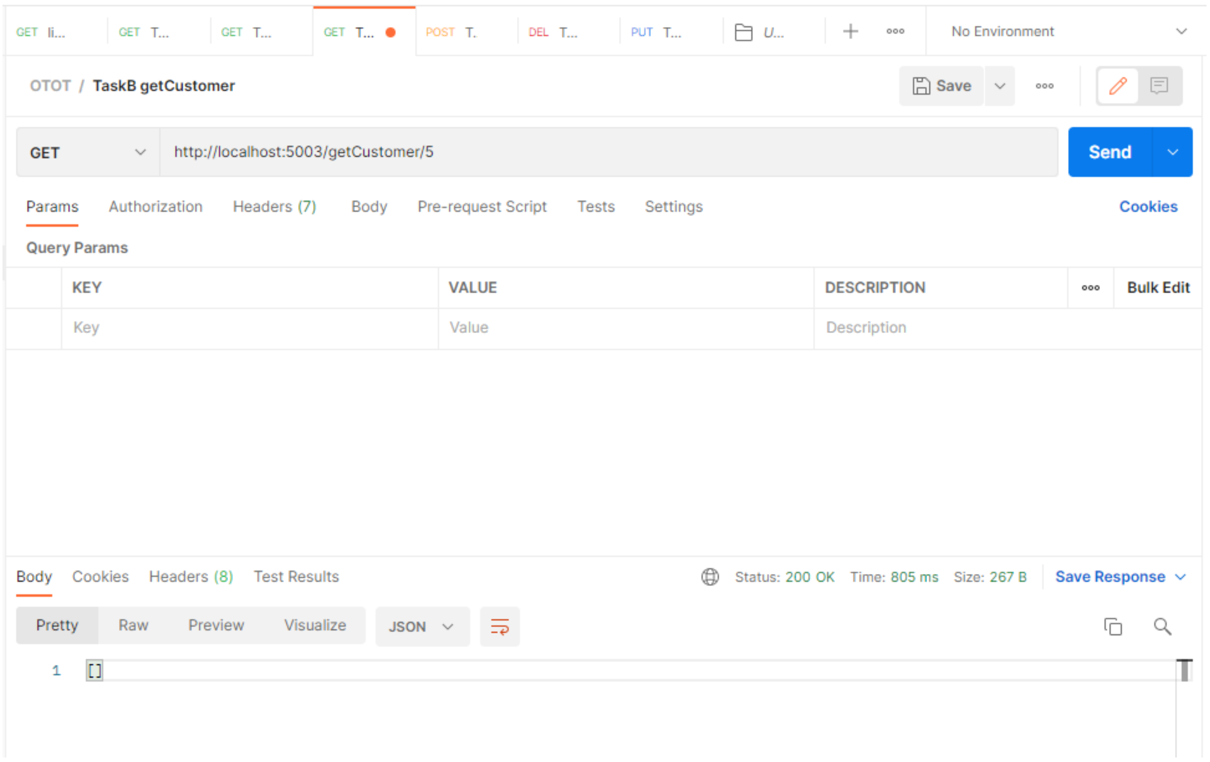
```
{
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

Status: 200 OK Time: 513 ms Size: 260 B

```
1  OK
```

Edge cases and error handling done and tested to prevent the deployed endpoint from crashing

1. (GET /getCustomer) Retrieving invalid customer id will return empty array.



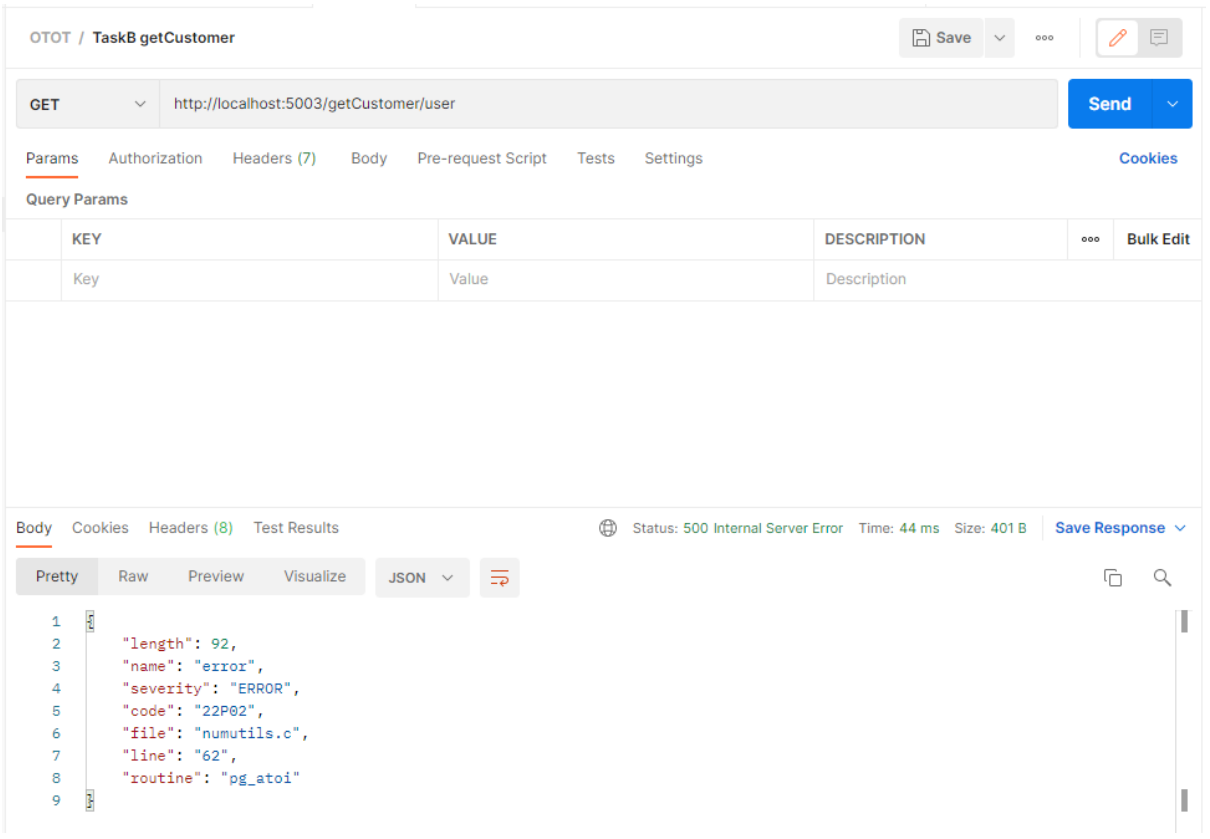
OTOT / TaskB getCustomer

GET http://localhost:5003/getCustomer/5

Status: 200 OK Time: 805 ms Size: 267 B

```
[]
```

2. (GET /getCustomer) Invalid customer id (not integer).



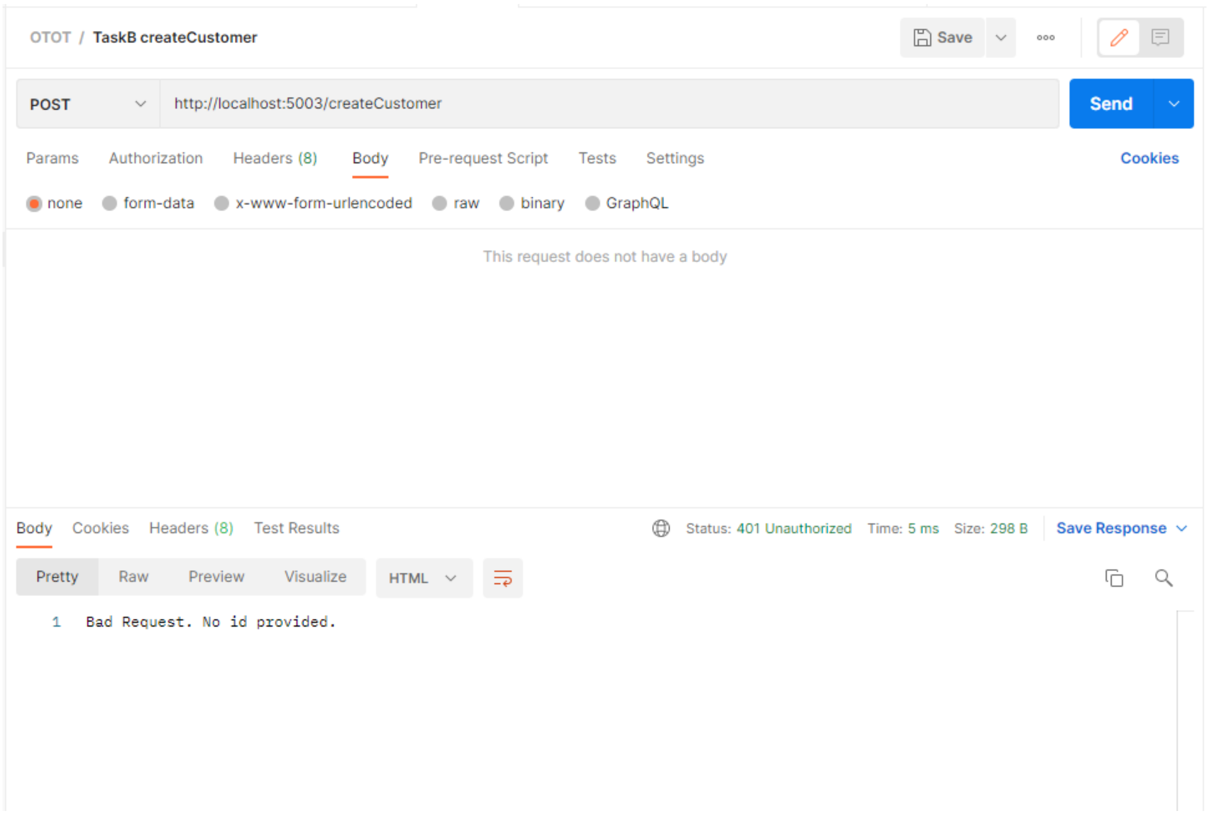
OTOT / TaskB getCustomer

GET http://localhost:5003/getCustomer/user

Status: 500 Internal Server Error Time: 44 ms Size: 401 B

```
{
    "length": 92,
    "name": "error",
    "severity": "ERROR",
    "code": "22P02",
    "file": "numutils.c",
    "line": "62",
    "routine": "pg_atoi"
}
```

3. (POST /createCustomer) Create customer without request body.



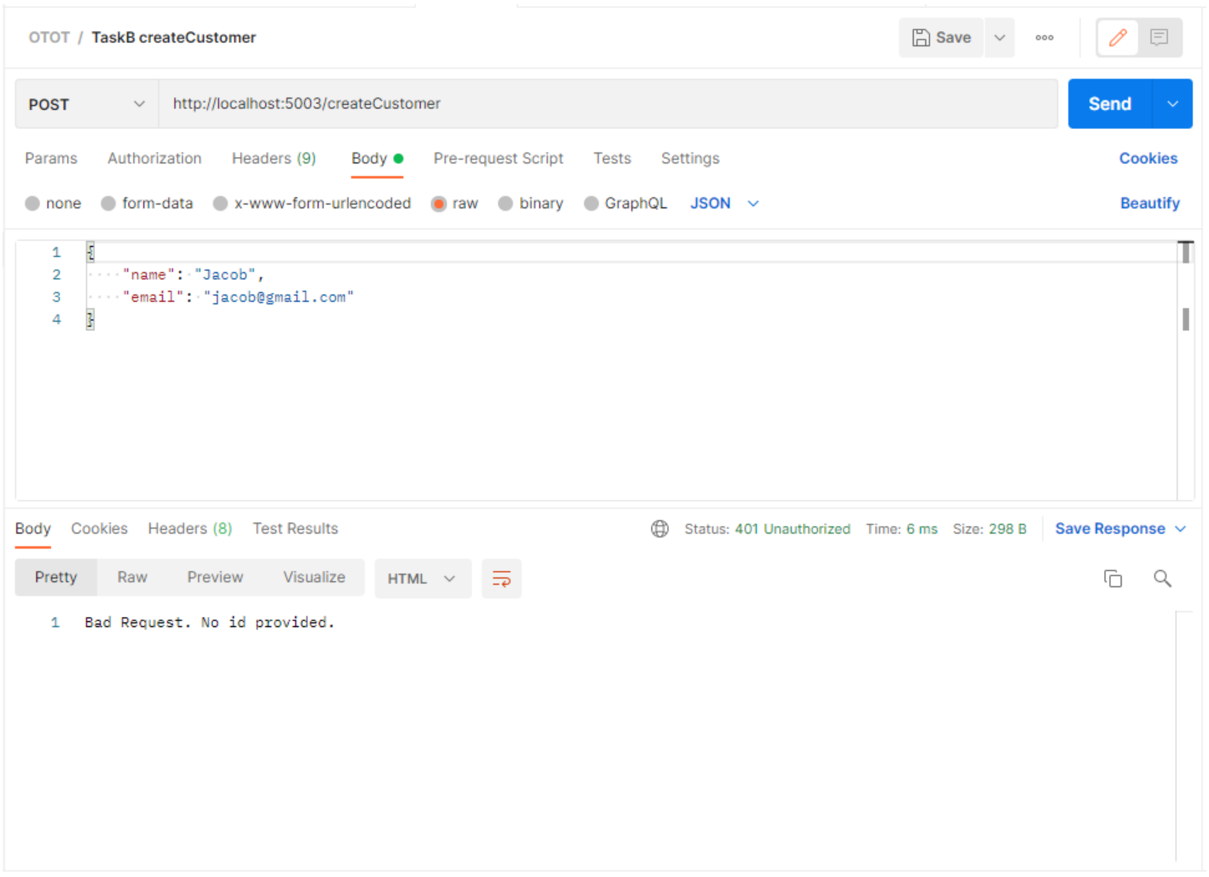
http://localhost:5003/createCustomer

This request does not have a body

Status: 401 Unauthorized Time: 5 ms Size: 298 B

```
Bad Request. No id provided.
```

4. (POST /createCustomer) Create customer without customer id.



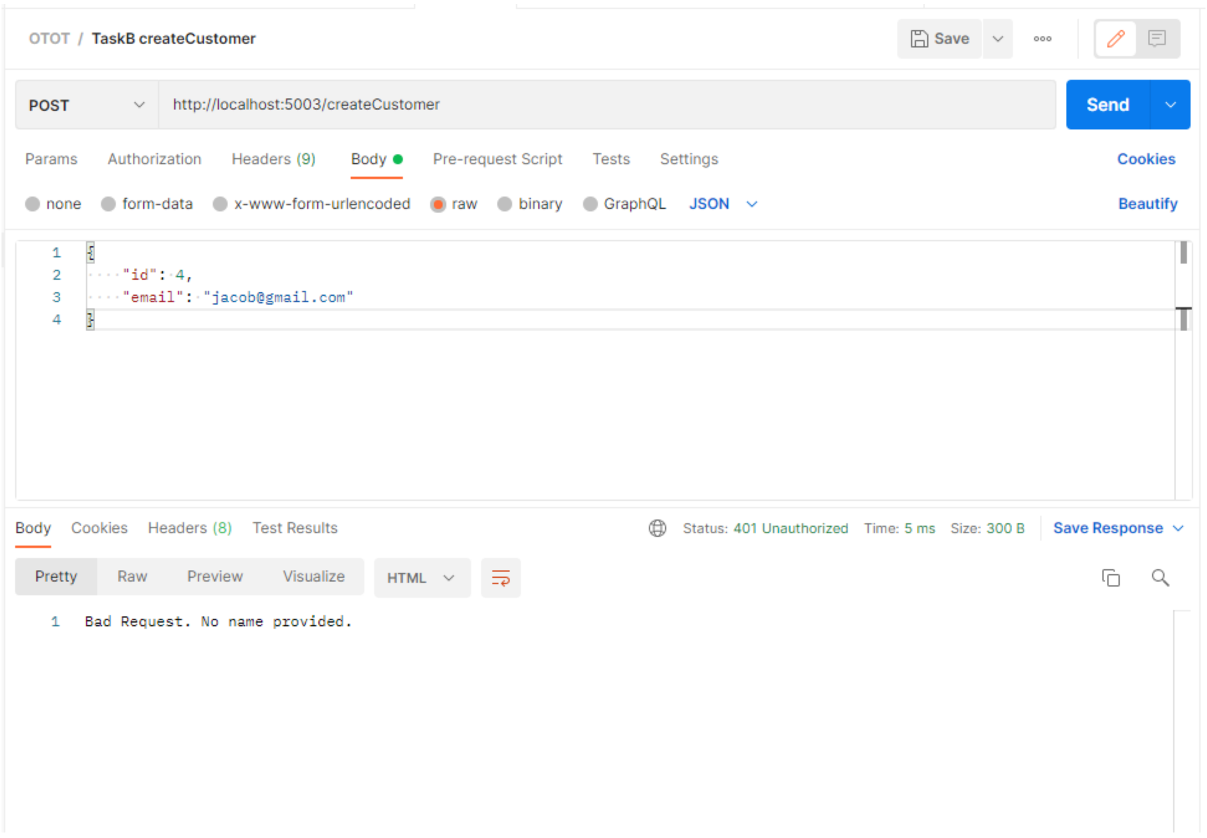
http://localhost:5003/createCustomer

```
{
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

Status: 401 Unauthorized Time: 6 ms Size: 298 B

```
Bad Request. No id provided.
```

5. (POST /createCustomer) Create customer without customer name.



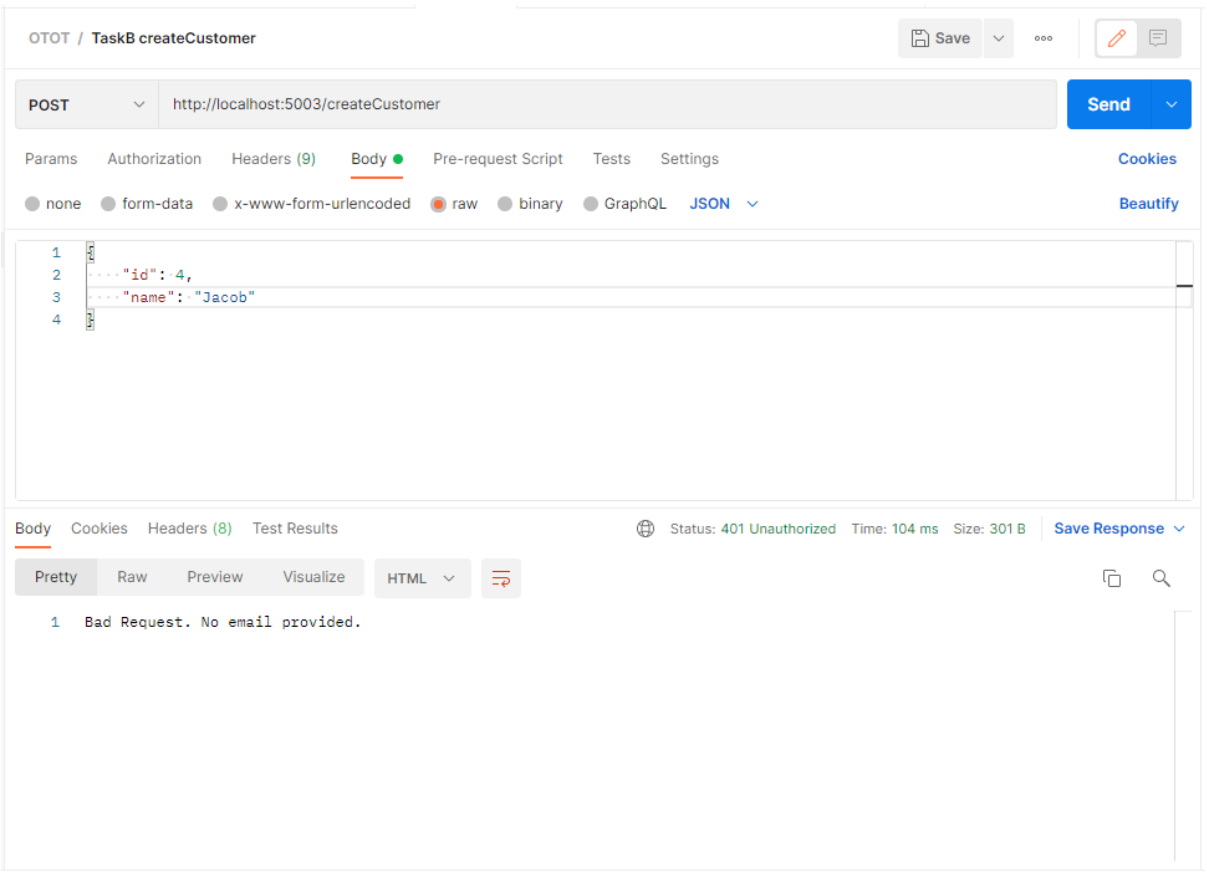
POST http://localhost:5003/createCustomer

```
{
    "id": 4,
    "email": "jacob@gmail.com"
}
```

Status: 401 Unauthorized Time: 5 ms Size: 300 B

```
1   Bad Request. No name provided.
```

6. (POST /createCustomer) Create customer without customer email.



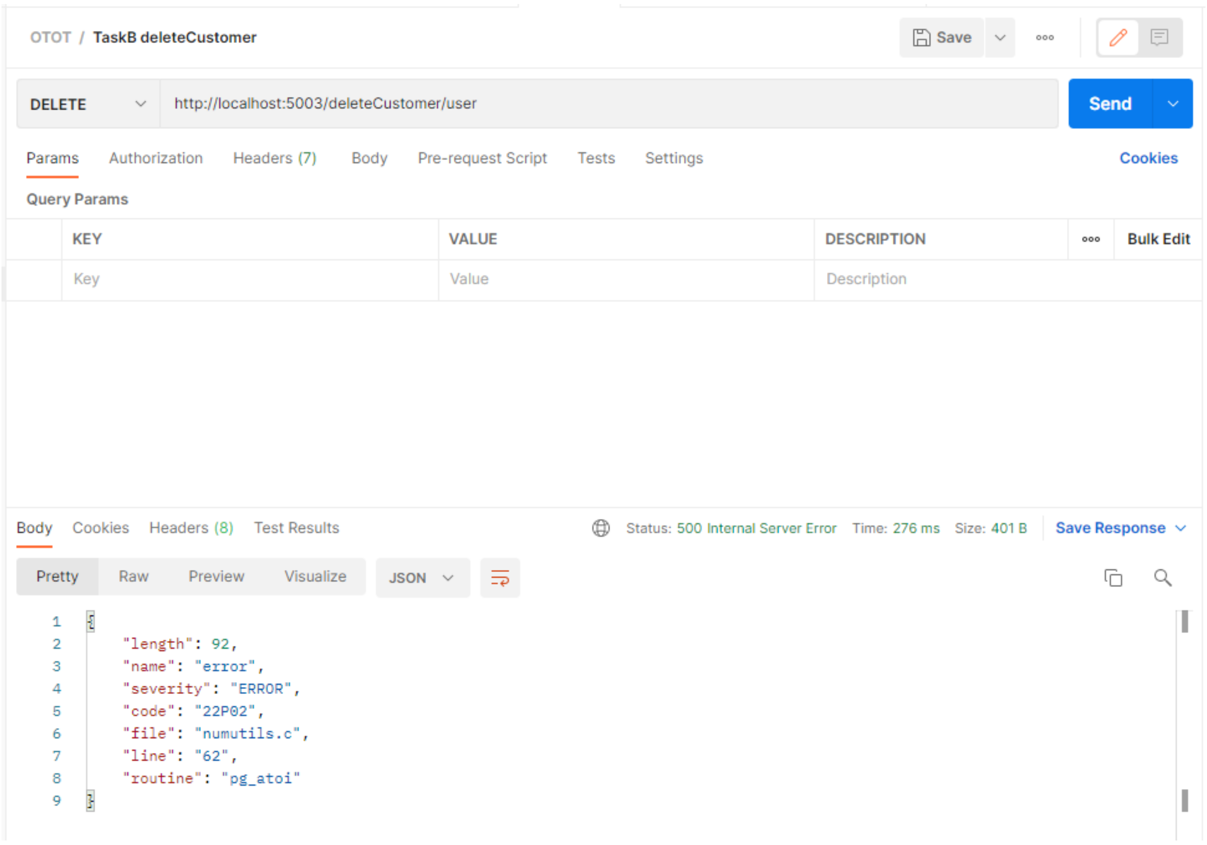
POST http://localhost:5003/createCustomer

```
{
    "id": 4,
    "name": "Jacob"
}
```

Status: 401 Unauthorized Time: 104 ms Size: 301 B

```
1   Bad Request. No email provided.
```

7. (DELETE /deleteCustomer) Invalid customer id (not integer).


OTOT / TaskB deleteCustomer

DELETE http://localhost:5003/deleteCustomer/user

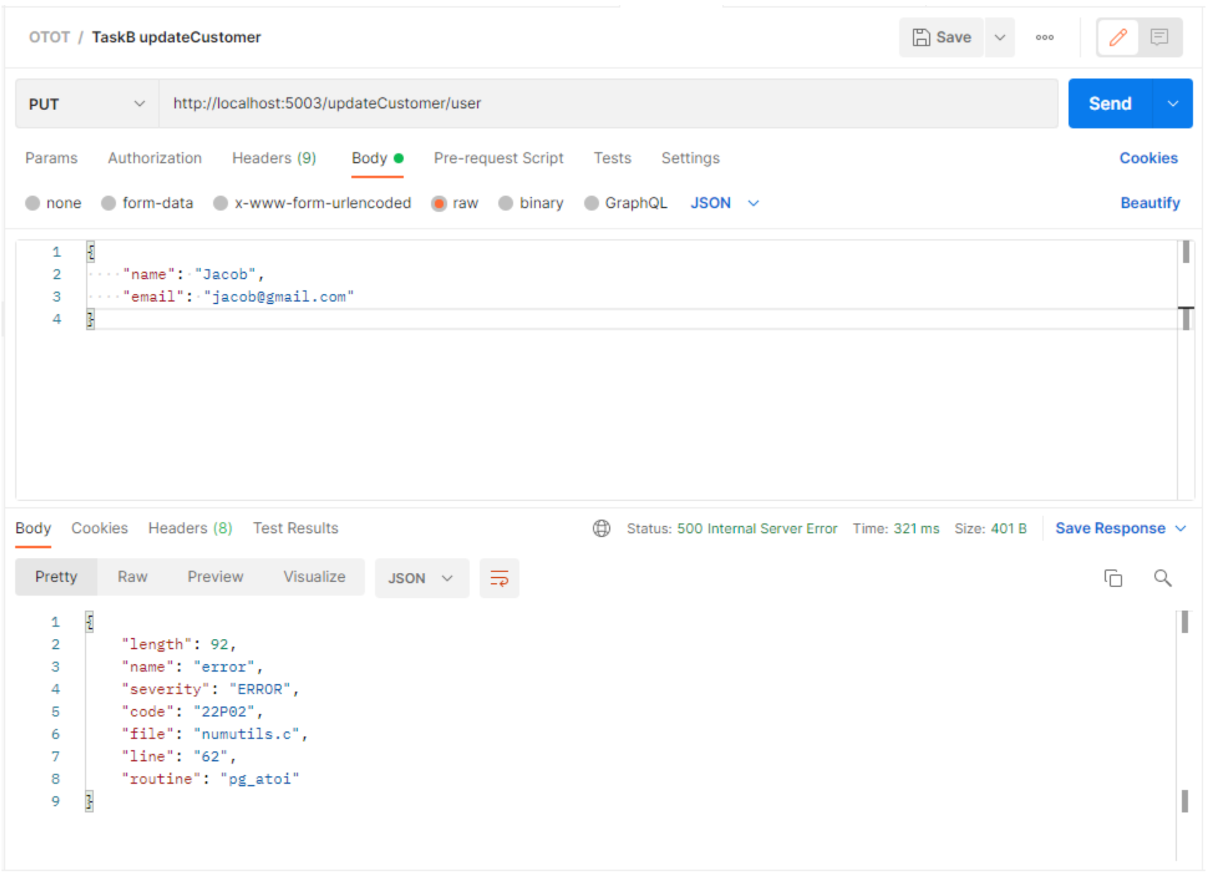
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 500 Internal Server Error Time: 276 ms Size: 401 B

```
{
    "length": 92,
    "name": "error",
    "severity": "ERROR",
    "code": "22P02",
    "file": "numutils.c",
    "line": "62",
    "routine": "pg_atoi"
}
```

8. (PUT /updateCustomer) Invalid customer id (not integer).


OTOT / TaskB updateCustomer

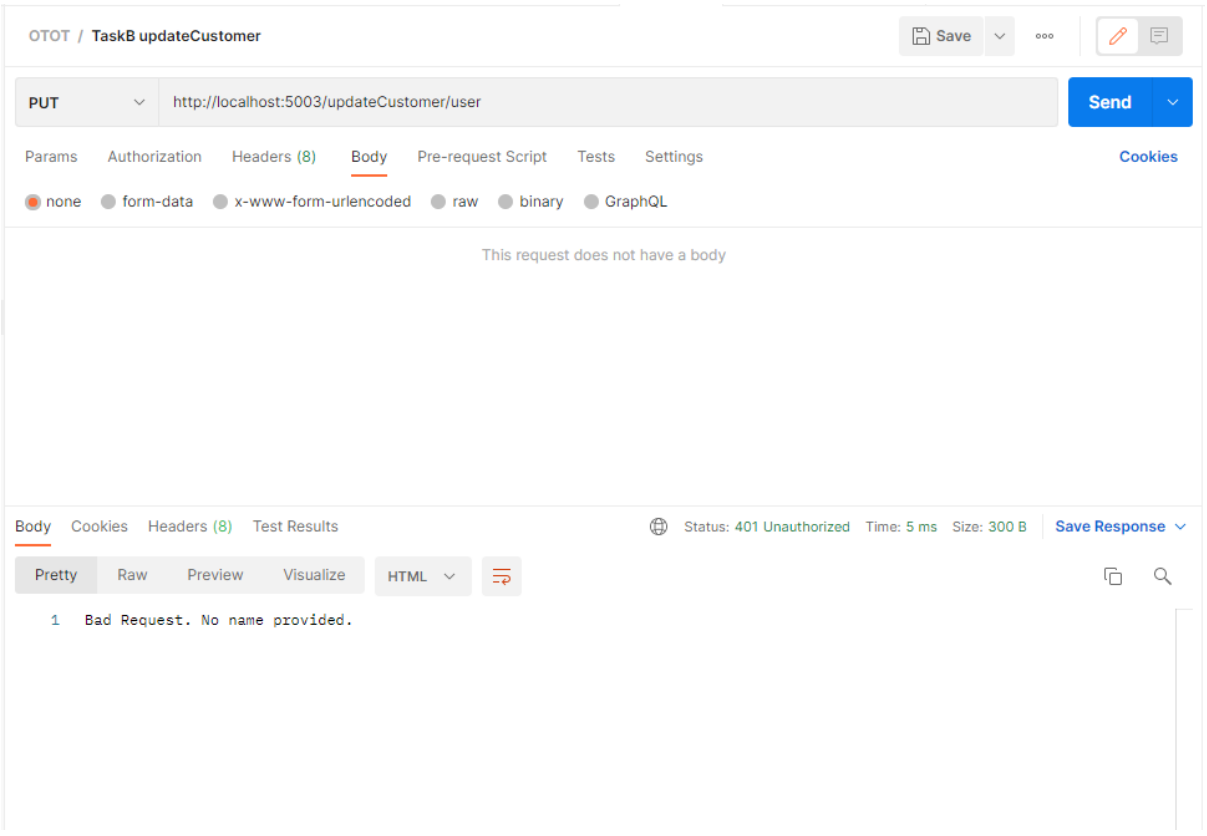
PUT http://localhost:5003/updateCustomer/user

```
{
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

Status: 500 Internal Server Error Time: 321 ms Size: 401 B

```
{
    "length": 92,
    "name": "error",
    "severity": "ERROR",
    "code": "22P02",
    "file": "numutils.c",
    "line": "62",
    "routine": "pg_atoi"
}
```

9. (PUT /updateCustomer) Update customer without request body.


OTOT / TaskB updateCustomer

PUT http://localhost:5003/updateCustomer/user

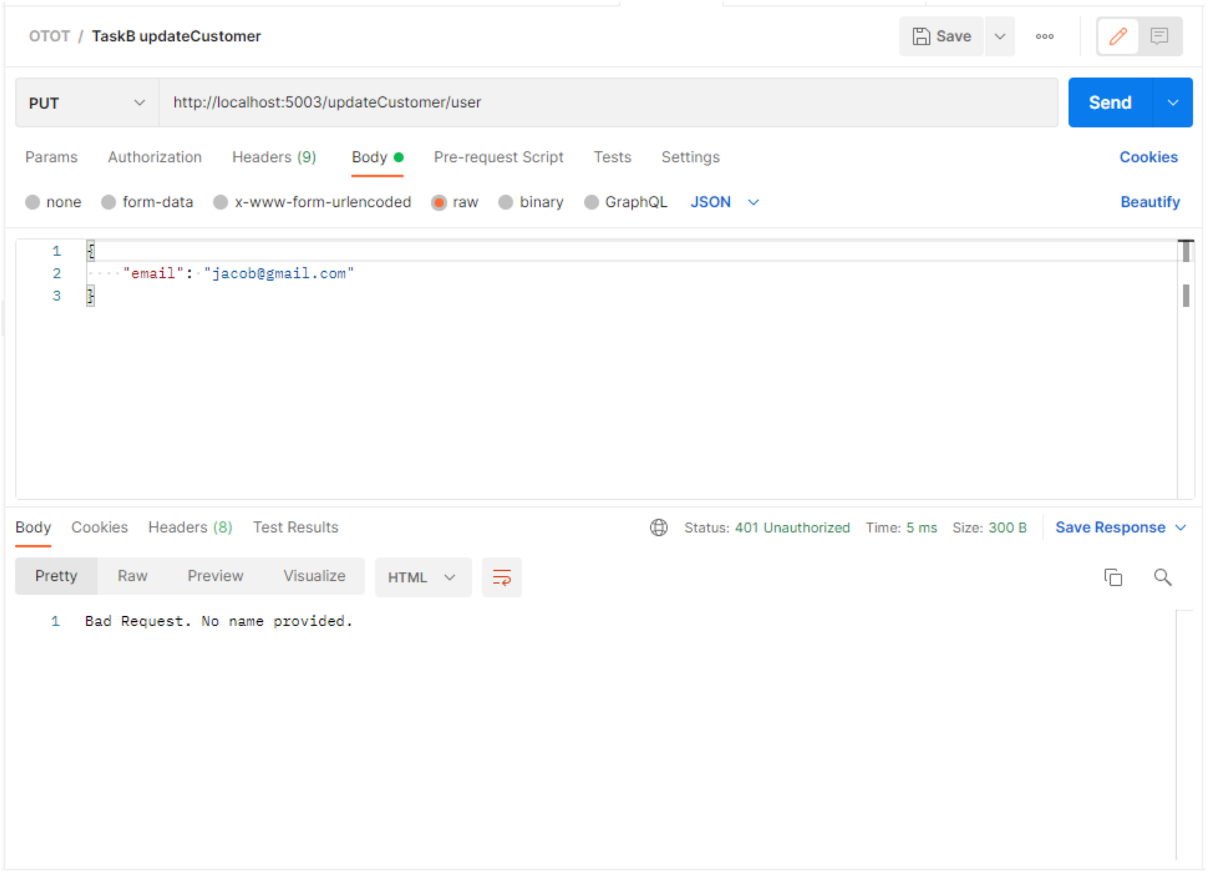
Body: none

This request does not have a body

Status: 401 Unauthorized Time: 5 ms Size: 300 B

```
Bad Request. No name provided.
```

10. (PUT /updateCustomer) Update customer without customer name.


OTOT / TaskB updateCustomer

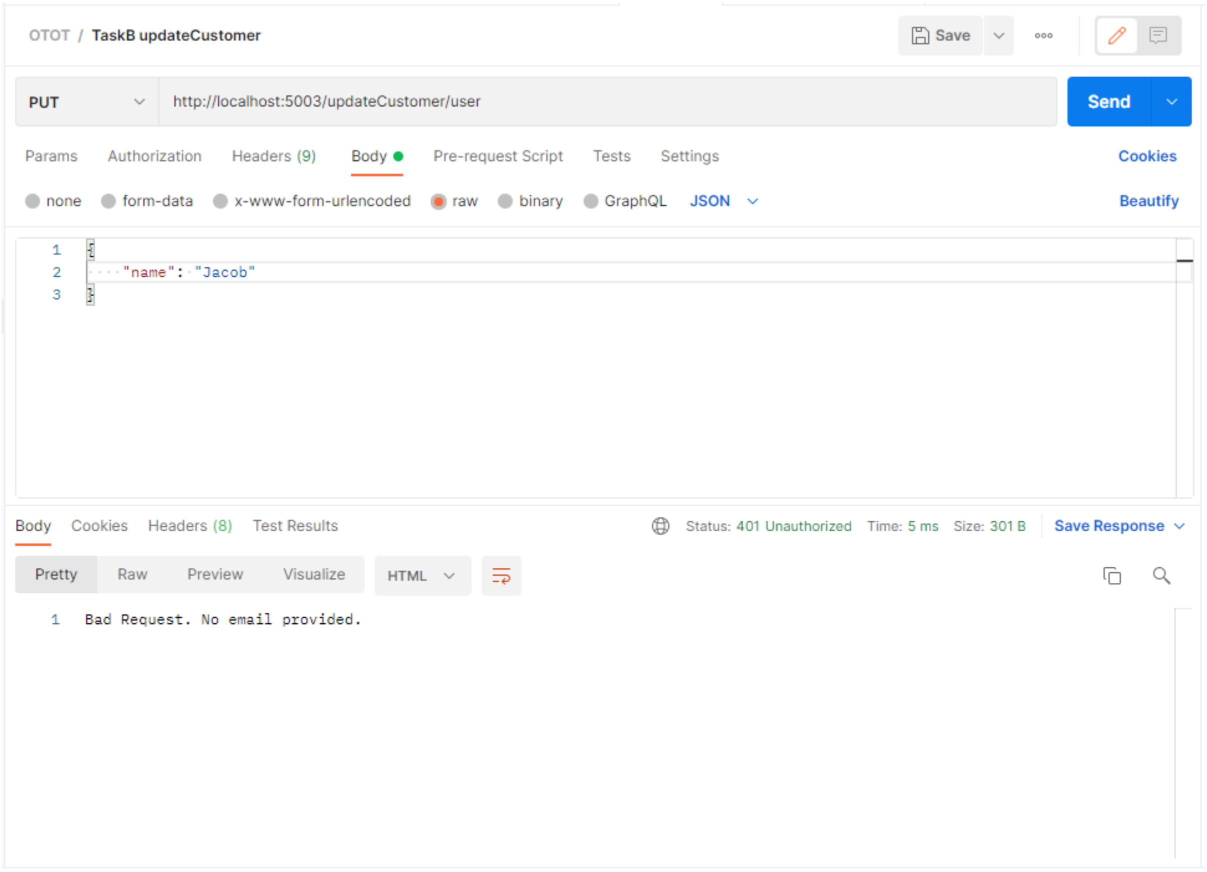
PUT http://localhost:5003/updateCustomer/user

```
{
    "email": "jacob@gmail.com"
}
```

Status: 401 Unauthorized Time: 5 ms Size: 300 B

```
Bad Request. No name provided.
```

11. (PUT /updateCustomer) Update customer without customer email.


OTOT / TaskB updateCustomer

PUT http://localhost:5003/updateCustomer/user

```
{
    "name": "Jacob"
}
```

Status: 401 Unauthorized Time: 5 ms Size: 301 B

```
Bad Request. No email provided.
```

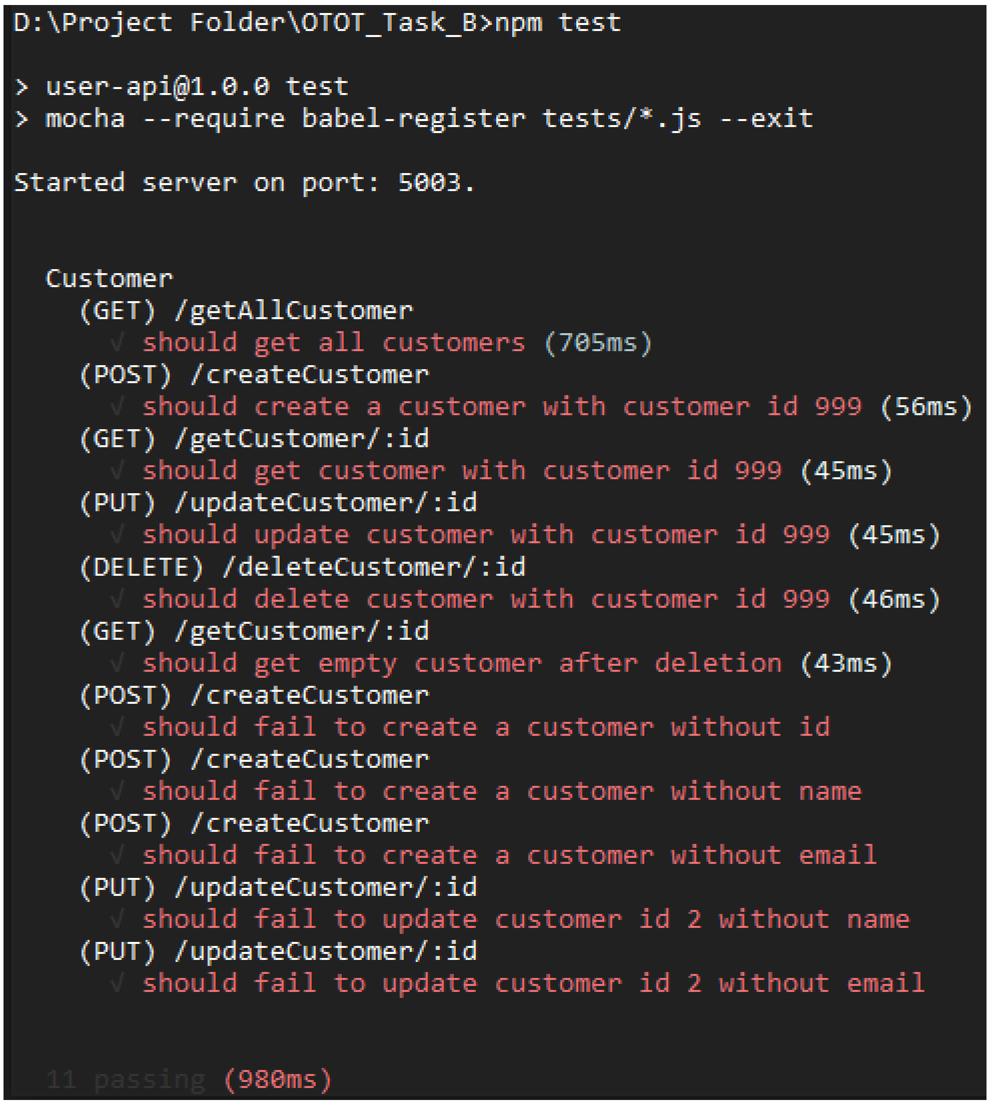
Run tests locally

1. On cmd, navigate your directory to the `OTOT_Task_B` folder.

```
D:\Project Folder>cd OTOT_Task_B

D:\Project Folder\OTOT_Task_B>
```

2. Run the command `npm test` to run the tests locally.

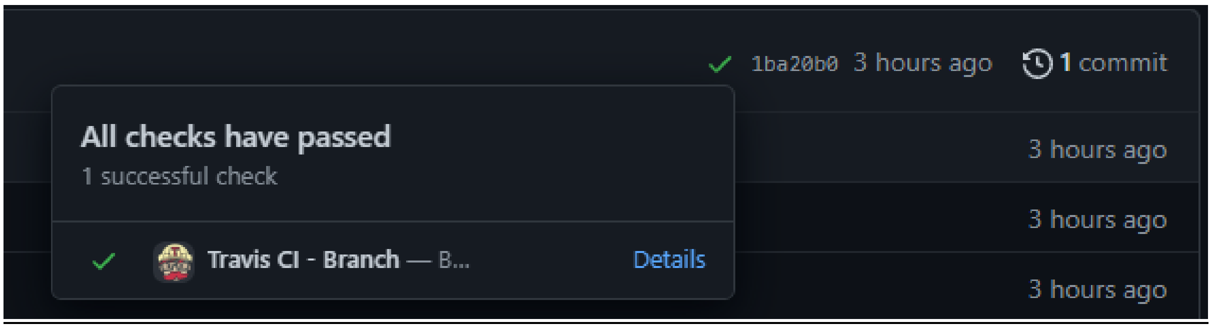
   (Note: Running the test locally will still connect you to the online database. If there is an existing customer with customer id 999 in the online database, the test will fail. In this scenario, you should either reset the database or delete the customer with id 999 before running the tests locally again.)

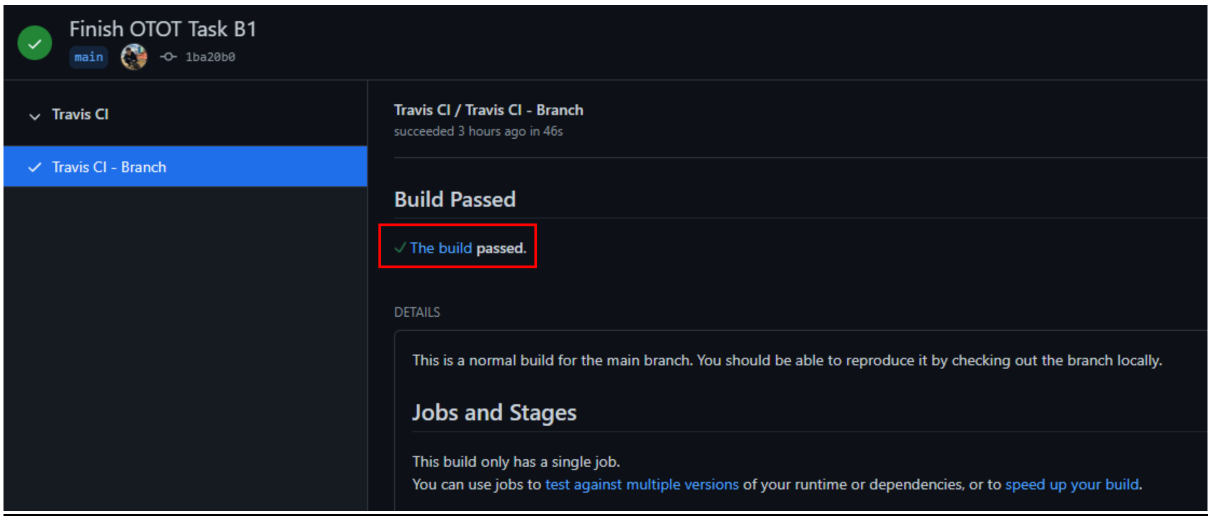


```
D:\Project Folder\OTOT_Task_B>npm test

> user-api@1.0.0 test
> mocha --require babel-register tests/*.js --exit

Started server on port: 5003.


  Customer
    (GET) /getAllCustomer
      √ should get all customers (705ms)
    (POST) /createCustomer
      √ should create a customer with customer id 999 (56ms)
    (GET) /getCustomer/:id
      √ should get customer with customer id 999 (45ms)
    (PUT) /updateCustomer/:id
      √ should update customer with customer id 999 (45ms)
    (DELETE) /deleteCustomer/:id
      √ should delete customer with customer id 999 (46ms)
    (GET) /getCustomer/:id
      √ should get empty customer after deletion (43ms)
    (POST) /createCustomer
      √ should fail to create a customer without id
    (POST) /createCustomer
      √ should fail to create a customer without name
    (POST) /createCustomer
      √ should fail to create a customer without email
    (PUT) /updateCustomer/:id
      √ should fail to update customer id 2 without name
    (PUT) /updateCustomer/:id
      √ should fail to update customer id 2 without email


  11 passing (980ms)
```

Running tests on Travis

Test on Travis is done automatically using Continuous Integration (CI) whenever a new commit is pushed into GitHub.




Finish OTOT Task B1
main 1ba20b0

Travis CI
Travis CI - Branch

Travis CI / Travis CI - Branch
succeeded 3 hours ago in 46s

Build Passed

✓ The build passed.

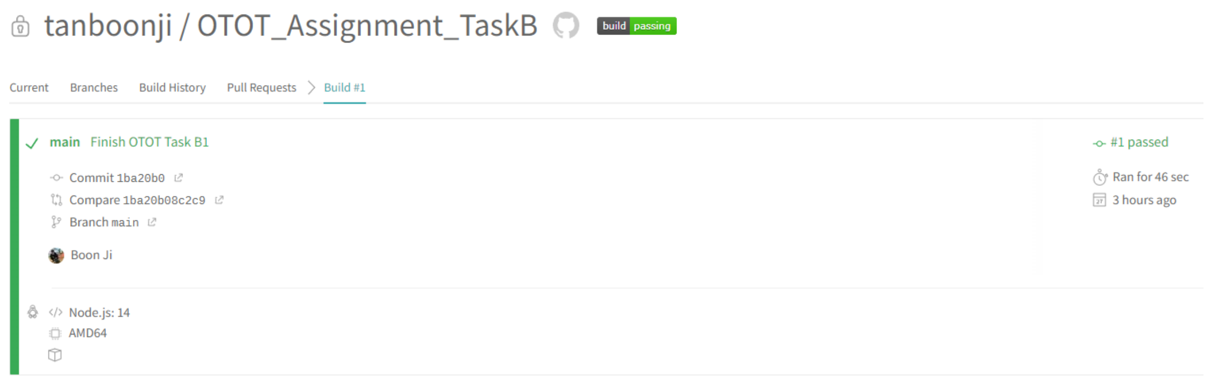
DETAILS

This is a normal build for the main branch. You should be able to reproduce it by checking out the branch locally.

Jobs and Stages

This build only has a single job.
You can use jobs to test against multiple versions of your runtime or dependencies, or to speed up your build.

You can click on `The build` to go to Travis to view the job logs where they will run `npm test`.

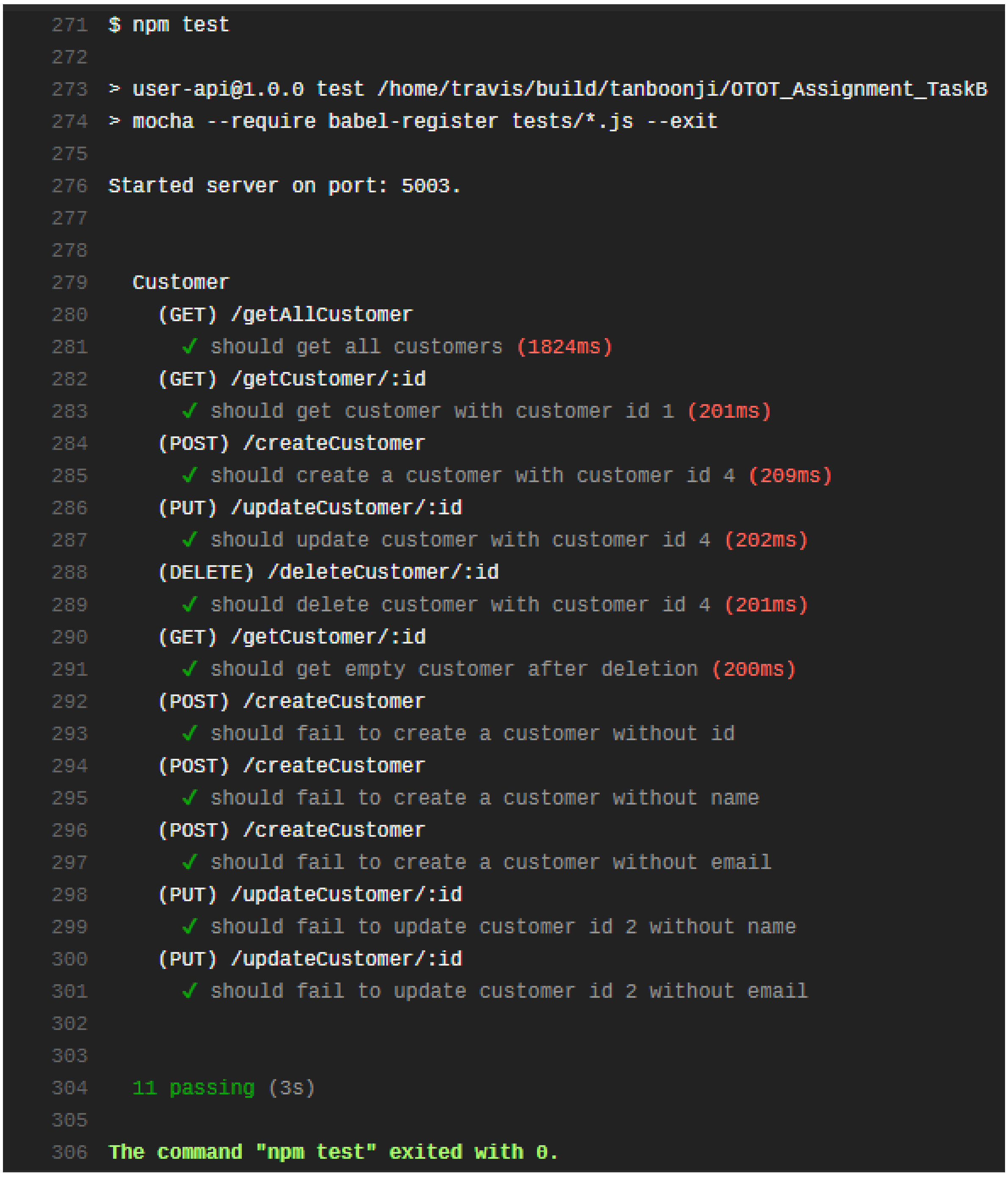

tanboonji / OTOT_Assignment_TaskB build passing

Current Branches Build History Pull Requests > Build #1

✓ main Finish OTOT Task B1
Commit 1ba20b0
Compare 1ba20b08c2c9
Branch main
Boon Ji

#1 passed
Ran for 46 sec
3 hours ago

Node.js: 14
AMD64

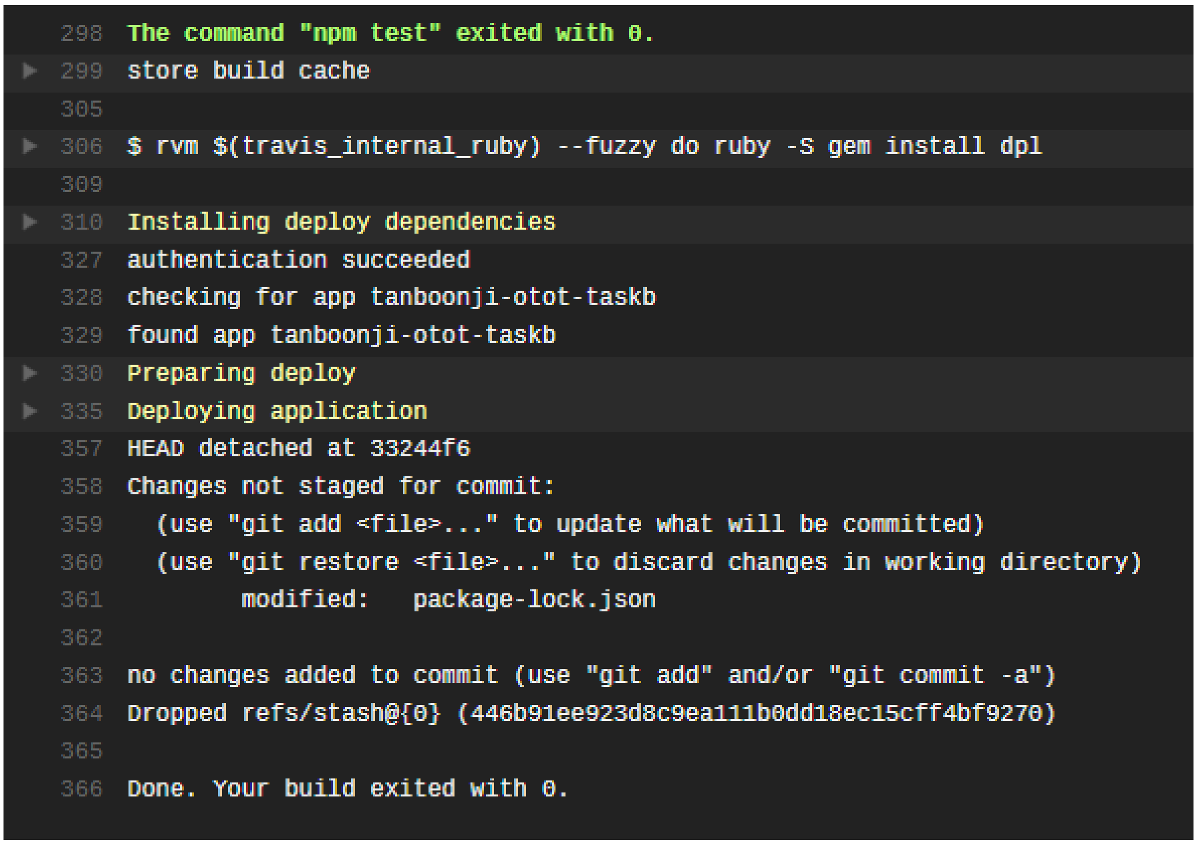


```
$ npm test

> user-api@1.0.0 test /home/travis/build/tanboonji/OTOT_Assignment_TaskB
> mocha --require babel-register tests/*.js --exit

Started server on port: 5003.


  Customer
    (GET) /getAllCustomer
      ✓ should get all customers (1824ms)
    (GET) /getCustomer/:id
      ✓ should get customer with customer id 1 (201ms)
    (POST) /createCustomer
      ✓ should create a customer with customer id 4 (209ms)
    (PUT) /updateCustomer/:id
      ✓ should update customer with customer id 4 (202ms)
    (DELETE) /deleteCustomer/:id
      ✓ should delete customer with customer id 4 (201ms)
    (GET) /getCustomer/:id
      ✓ should get empty customer after deletion (200ms)
    (POST) /createCustomer
      ✓ should fail to create a customer without id
    (POST) /createCustomer
      ✓ should fail to create a customer without name
    (POST) /createCustomer
      ✓ should fail to create a customer without email
    (PUT) /updateCustomer/:id
      ✓ should fail to update customer id 2 without name
    (PUT) /updateCustomer/:id
      ✓ should fail to update customer id 2 without email


  11 passing (3s)

The command "npm test" exited with 0.
```

<u>Continuous Deployment using Travis and Heroku</u>

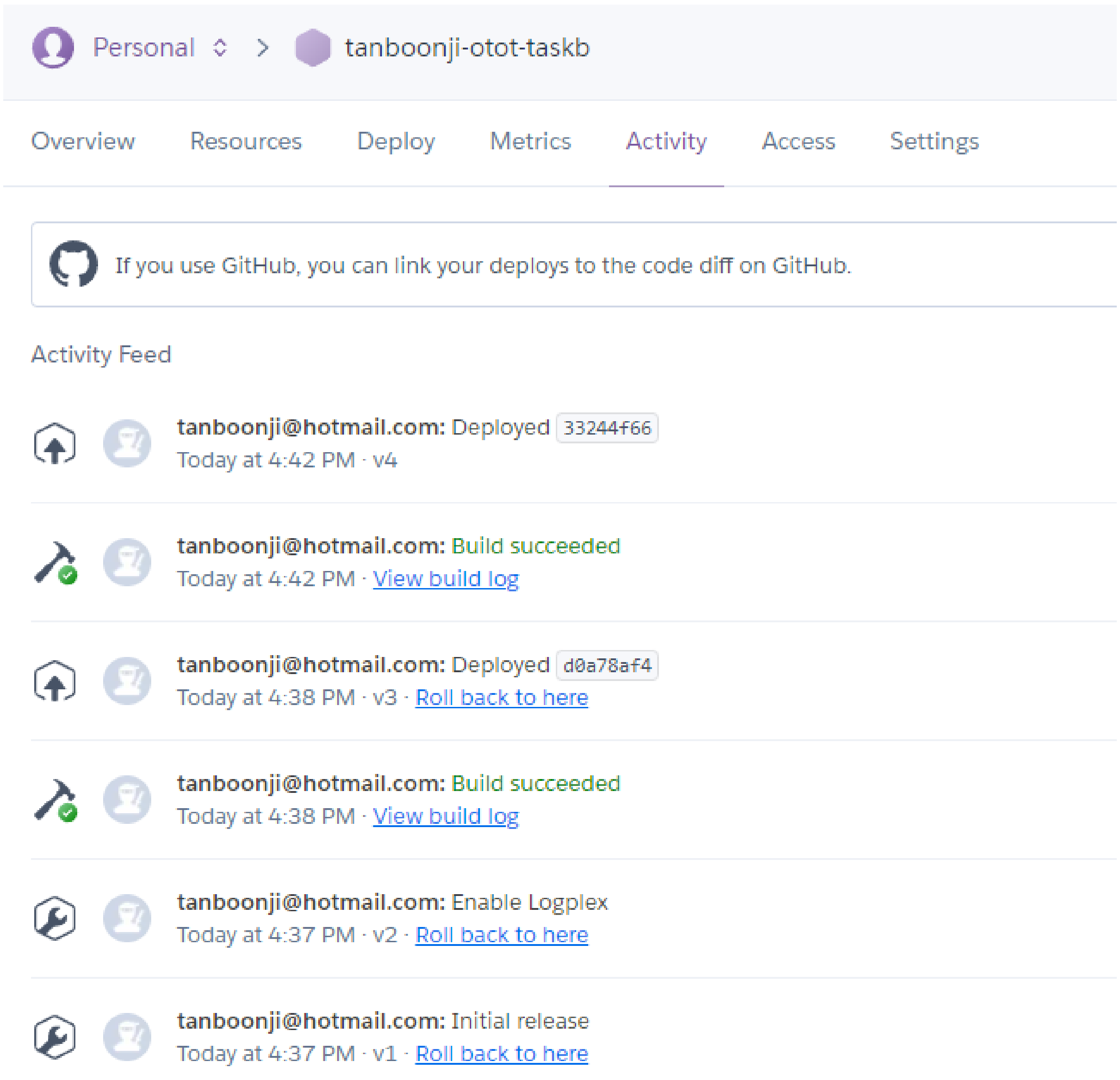
Like CI, Continuous Deployment (CD) is done on Travis with Heroku. Whenever a new commit is pushed into GitHub, after the tests in CI passes, the application is then deployed on Heroku.

Below are the logs on Travis showing the successful deployment to Heroku.



```
▶ The command "npm test" exited with 0.
▶ store build cache

▶ $ rvm $(travis_internal_ruby) --fuzzy do ruby -S gem install dpl

▶ Installing deploy dependencies
authentication succeeded
checking for app tanboonji-otot-taskb
found app tanboonji-otot-taskb
▶ Preparing deploy
▶ Deploying application
HEAD detached at 33244f6
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   package-lock.json

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (446b91ee923d8c9ea111b0dd18ec15cff4bf9270)

Done. Your build exited with 0.
```

Below is the deployment history on Heroku.



Personal > tanboonji-otot-taskb

Overview Resources Deploy Metrics Activity Access Settings

If you use GitHub, you can link your deploys to the code diff on GitHub.

Activity Feed

tanboonji@hotmail.com: Deployed `33244f66`
Today at 4:42 PM · v4

tanboonji@hotmail.com: Build succeeded
Today at 4:42 PM · View build log

tanboonji@hotmail.com: Deployed `d0a78af4`
Today at 4:38 PM · v3 · Roll back to here

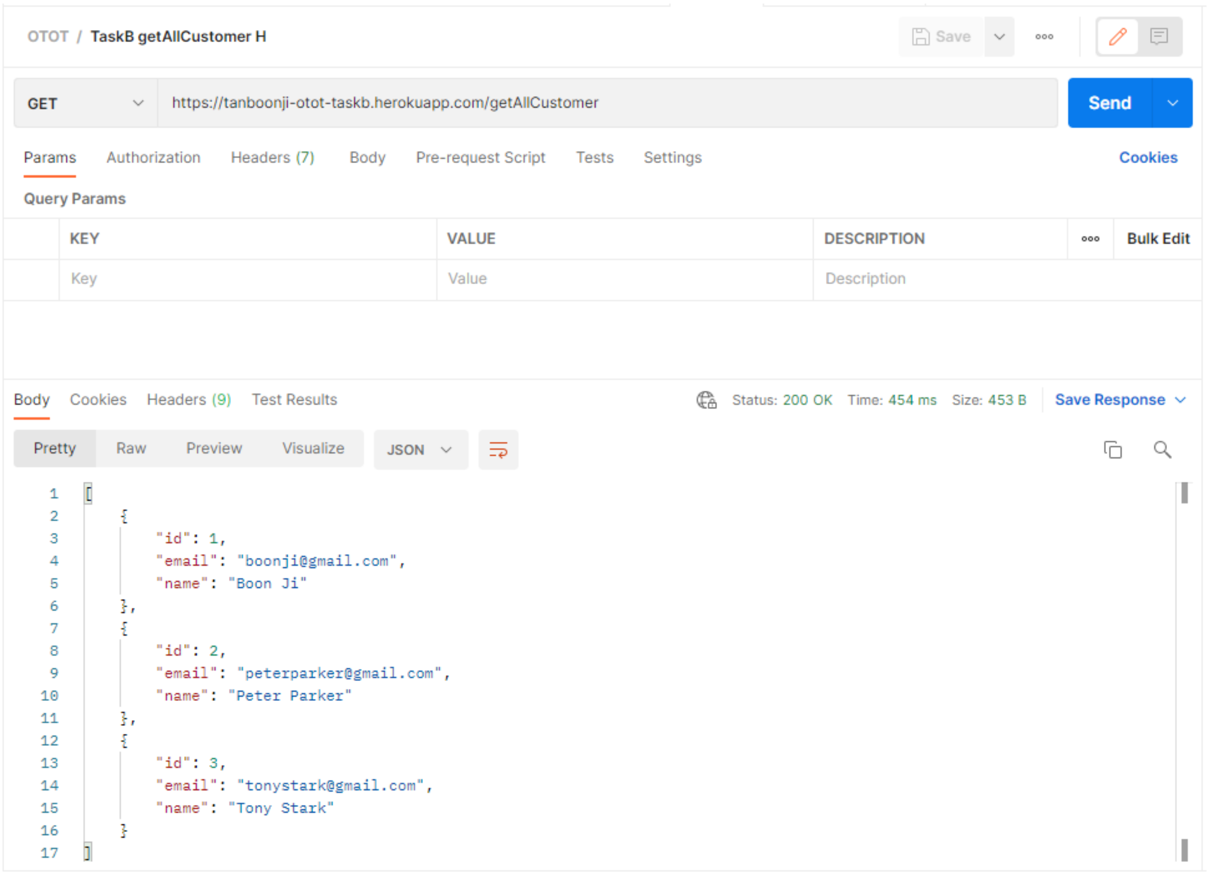
tanboonji@hotmail.com: Build succeeded
Today at 4:38 PM · View build log

tanboonji@hotmail.com: Enable Logplex
Today at 4:37 PM · v2 · Roll back to here

tanboonji@hotmail.com: Initial release
Today at 4:37 PM · v1 · Roll back to here

Screenshots of Postman calls to Deployed API

1. `https://tanboonji-otot-taskb.herokuapp.com/getAllCustomer`

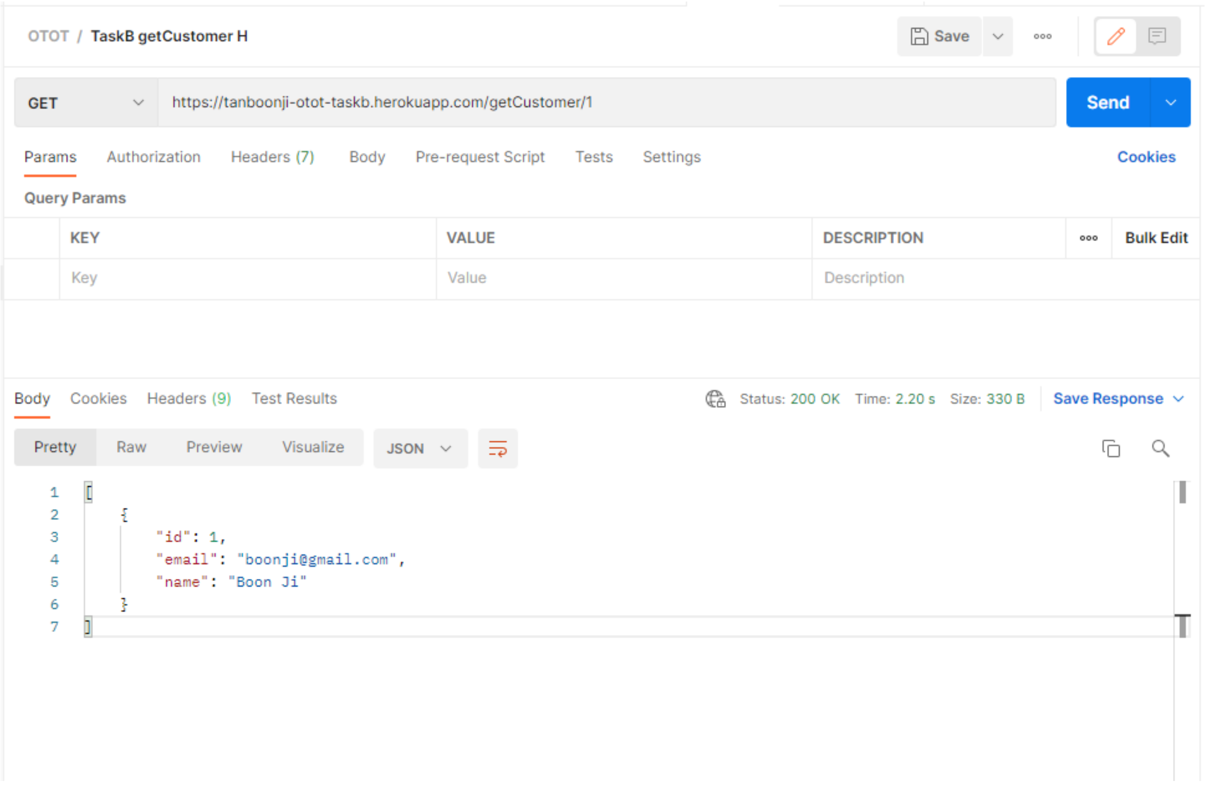


GET https://tanboonji-otot-taskb.herokuapp.com/getAllCustomer

Status: 200 OK Time: 454 ms Size: 453 B

```
[
    {
        "id": 1,
        "email": "boonji@gmail.com",
        "name": "Boon Ji"
    },
    {
        "id": 2,
        "email": "peterparker@gmail.com",
        "name": "Peter Parker"
    },
    {
        "id": 3,
        "email": "tonystark@gmail.com",
        "name": "Tony Stark"
    }
]
```

2. `https://tanboonji-otot-taskb.herokuapp.com/getCustomer/1`

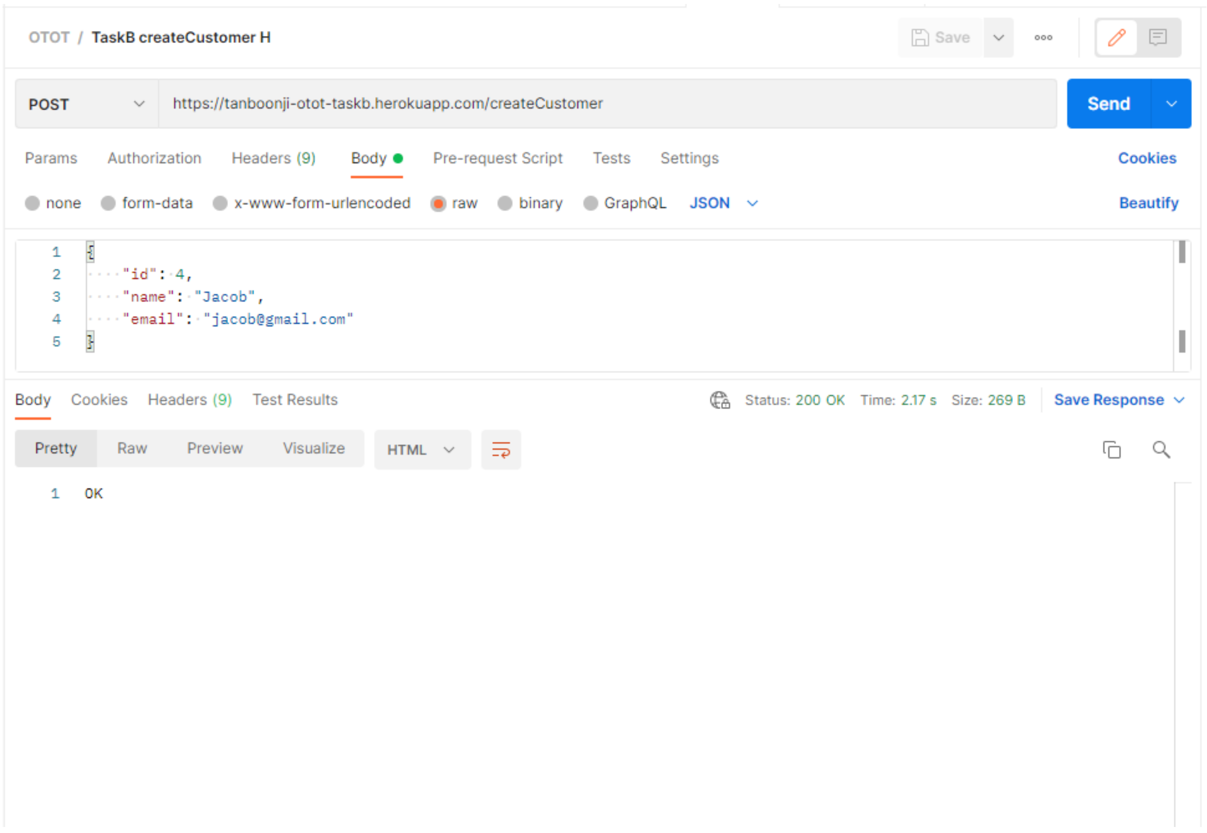


GET https://tanboonji-otot-taskb.herokuapp.com/getCustomer/1

Status: 200 OK Time: 2.20 s Size: 330 B

```
[
    {
        "id": 1,
        "email": "boonji@gmail.com",
        "name": "Boon Ji"
    }
]
```

3. `https://tanboonji-otot-taskb.herokuapp.com/createCustomer`



OTOT / TaskB createCustomer H

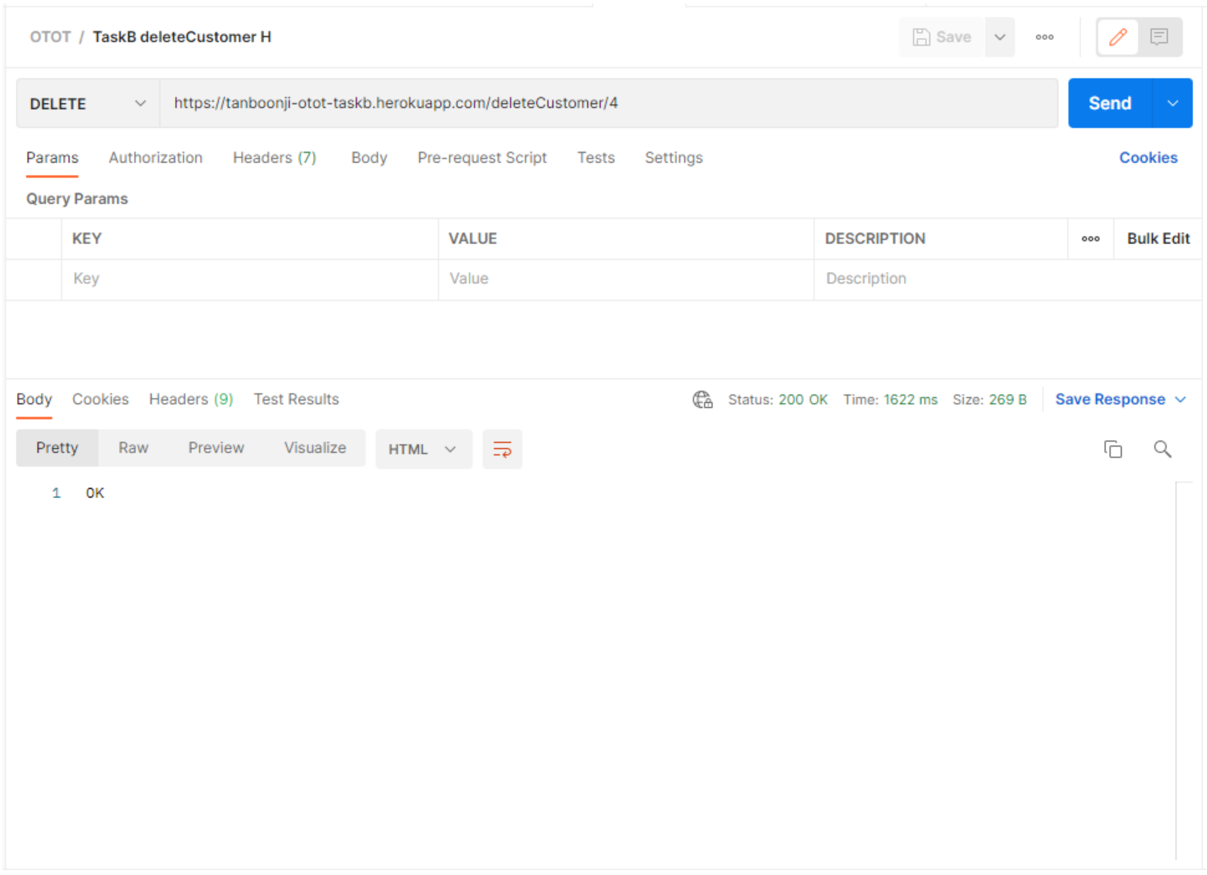
POST https://tanboonji-otot-taskb.herokuapp.com/createCustomer

```
{
    "id": 4,
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

Status: 200 OK Time: 2.17 s Size: 269 B

```
OK
```

4. `https://tanboonji-otot-taskb.herokuapp.com/deleteCustomer/4`



OTOT / TaskB deleteCustomer H

DELETE https://tanboonji-otot-taskb.herokuapp.com/deleteCustomer/4

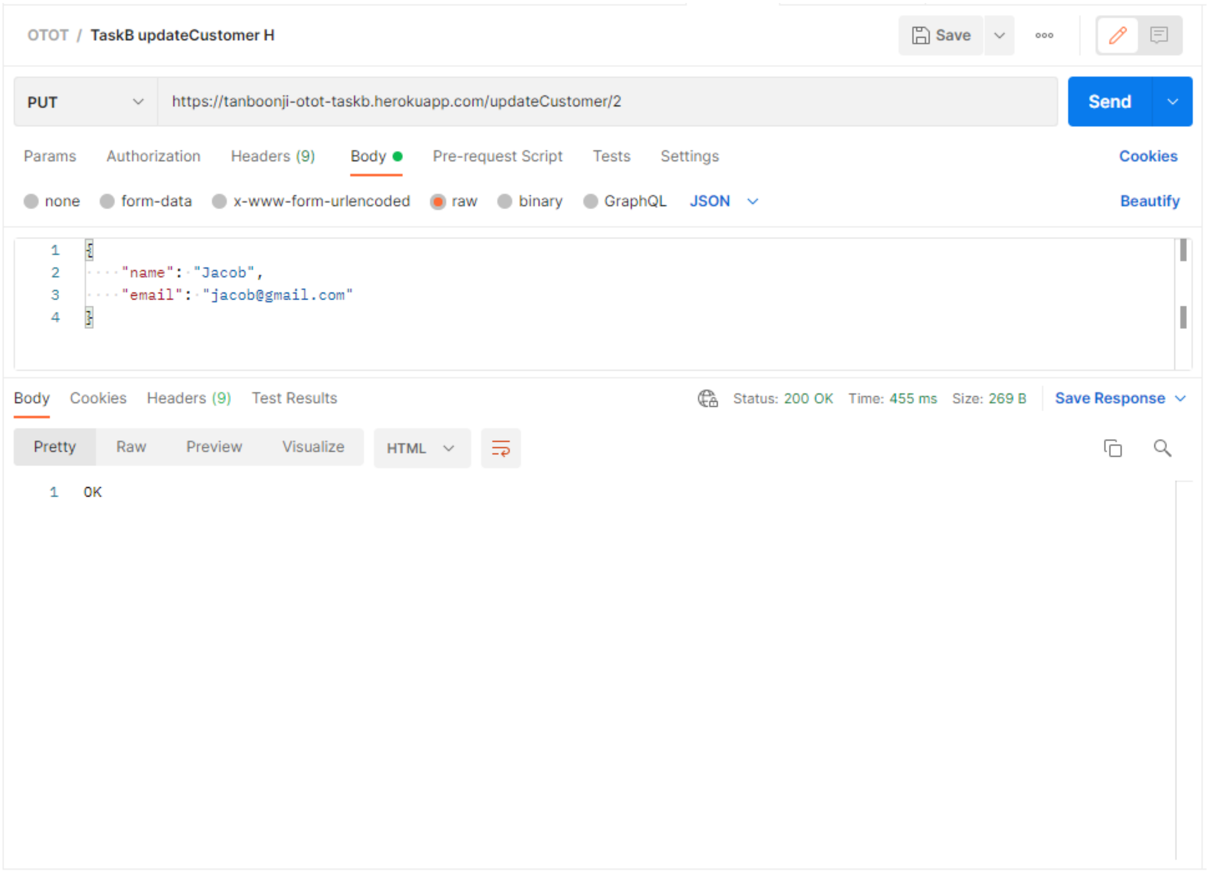
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 1622 ms Size: 269 B

```
OK
```

5. `https://tanboonji-otot-taskb.herokuapp.com/updateCustomer/2`


OTOT / TaskB updateCustomer H

PUT https://tanboonji-otot-taskb.herokuapp.com/updateCustomer/2

```
{
    "name": "Jacob",
    "email": "jacob@gmail.com"
}
```

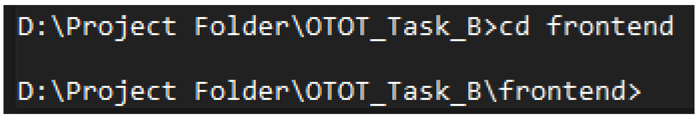
Status: 200 OK Time: 455 ms Size: 269 B

Pretty HTML

```
OK
```

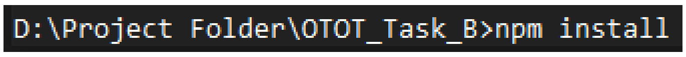
Set up frontend

1. On cmd, navigate your directory to the `OTOT_Task_B\frontend` folder.



```
D:\Project Folder\OTOT_Task_B>cd frontend

D:\Project Folder\OTOT_Task_B\frontend>
```

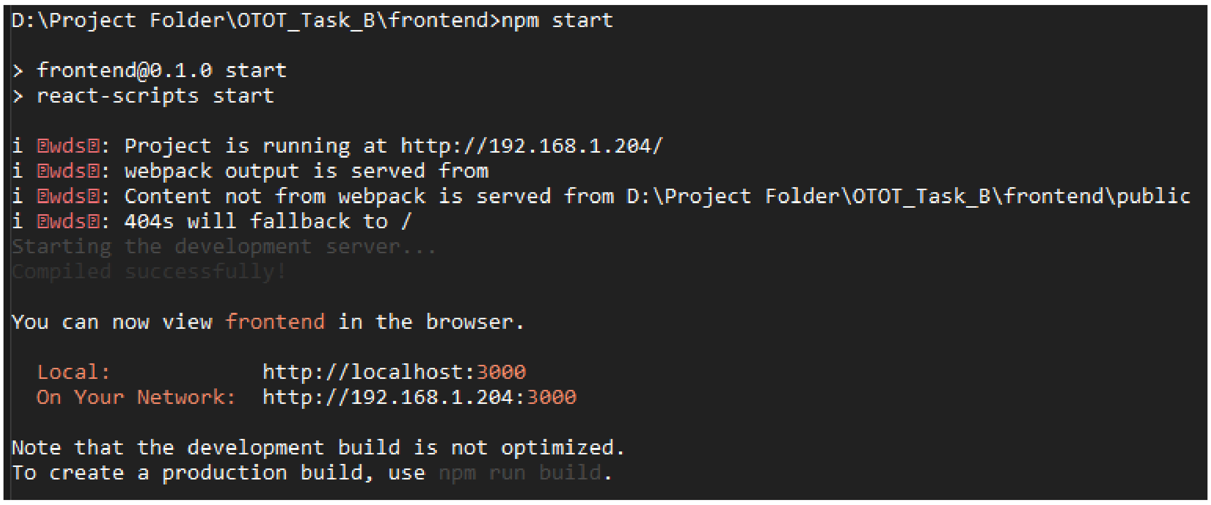
2. Install the required packages by running the command `npm install`, then wait for the packages to be installed.



```
D:\Project Folder\OTOT_Task_B>npm install
```

3. Start the frontend by running the command `npm start`.

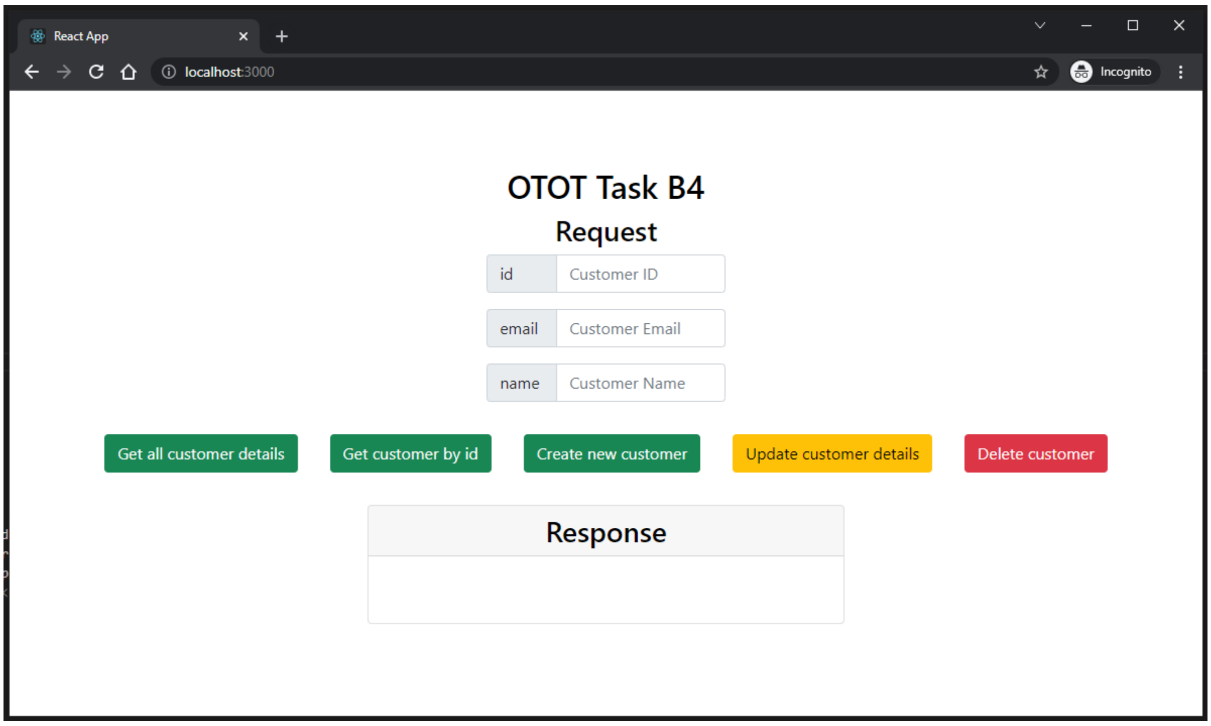
   You should see the following messages once the frontend has successfully started.



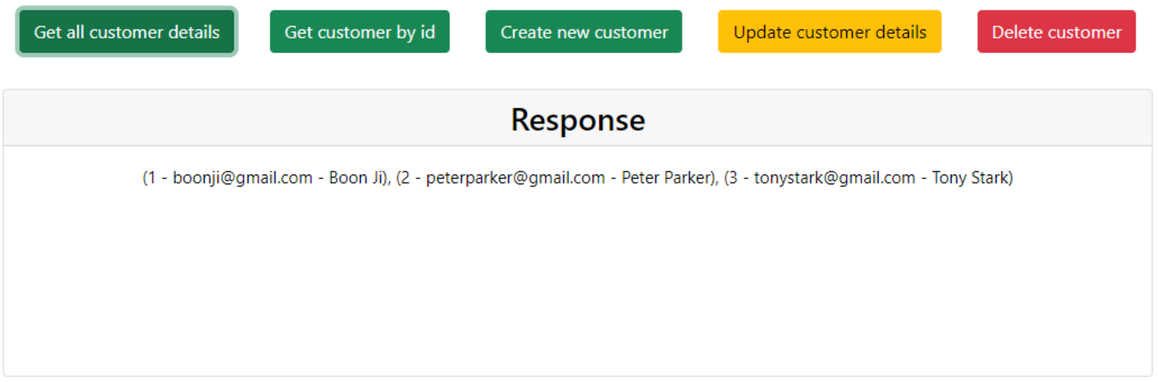
```
D:\Project Folder\OTOT_Task_B\frontend>npm start

> frontend@0.1.0 start
> react-scripts start

i ｢wds｣: Project is running at http://192.168.1.204/
i ｢wds｣: webpack output is served from
i ｢wds｣: Content not from webpack is served from D:\Project Folder\OTOT_Task_B\frontend\public
i ｢wds｣: 404s will fallback to /
Starting the development server...
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.204:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

4. Once the frontend has started, you can access it at `http://localhost:3000’.

   You can call the 5 APIs through the frontend to the deployed server on Heroku.

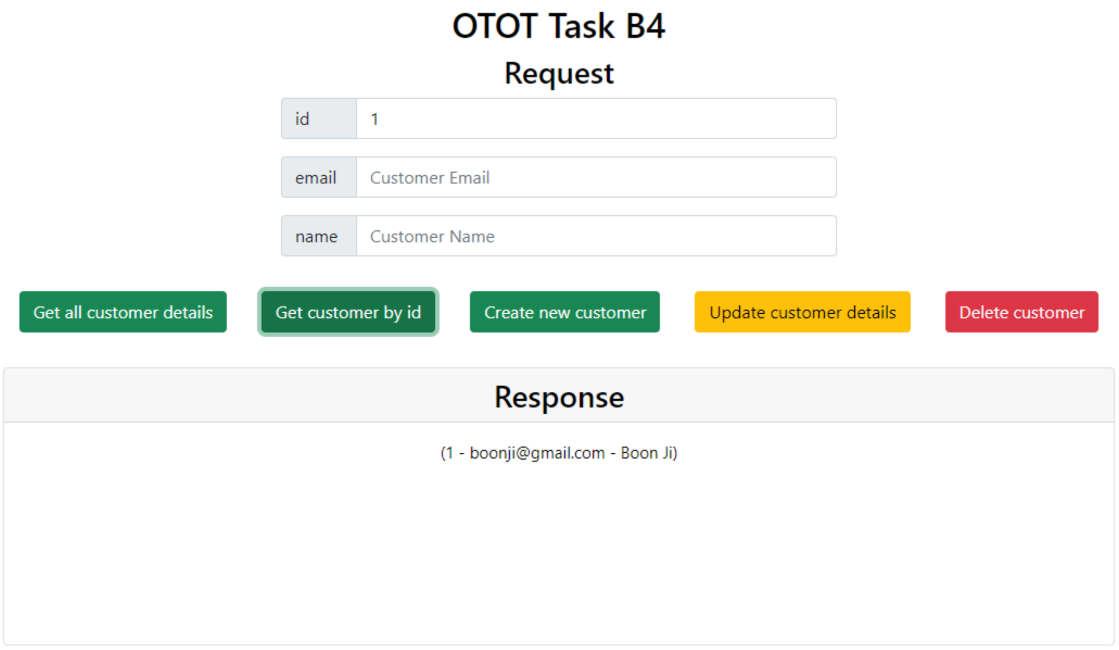


a. `Get all customer details`



(1 - boonji@gmail.com - Boon Ji), (2 - peterparker@gmail.com - Peter Parker), (3 - tonystark@gmail.com - Tony Stark)

b. `Get customer by id`


OTOT Task B4

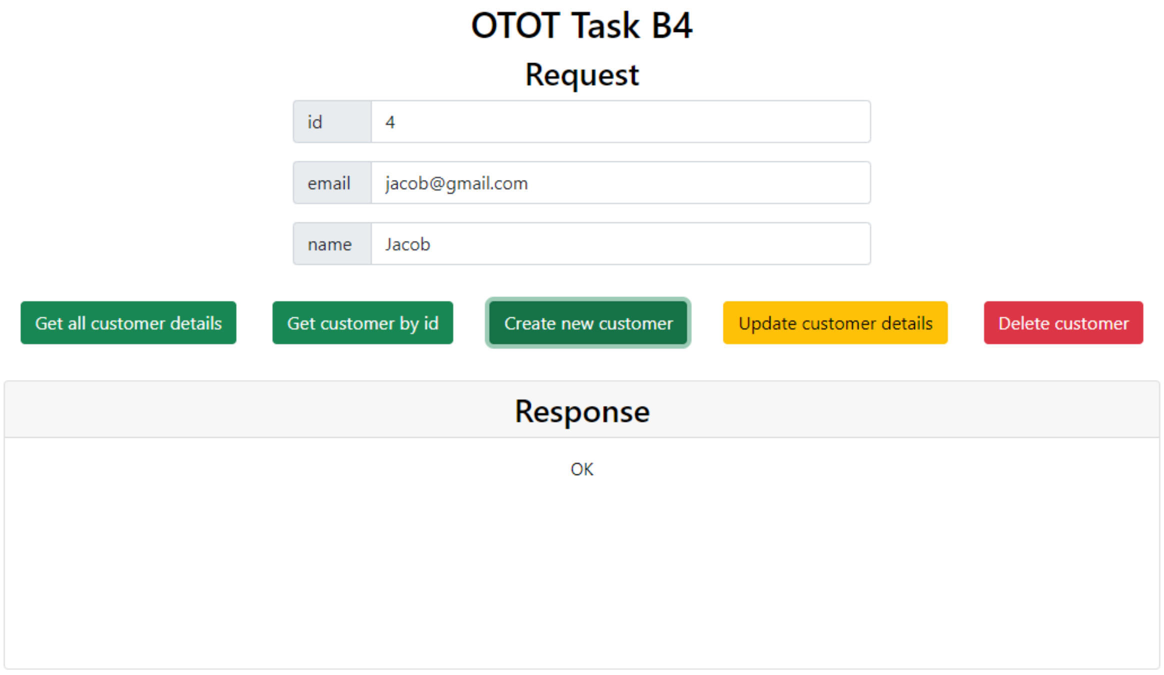
Request

id	1
email	Customer Email
name	Customer Name

Response

(1 - boonji@gmail.com - Boon Ji)

c. `Create new customer`


OTOT Task B4

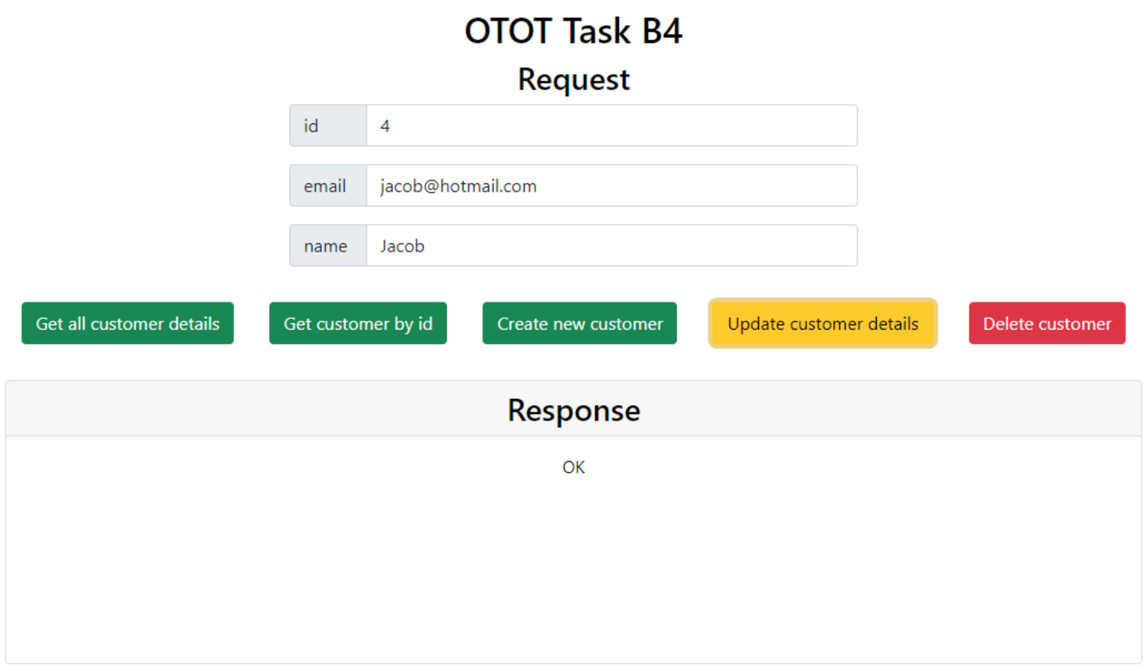
Request

id	4
email	jacob@gmail.com
name	Jacob

Response

OK

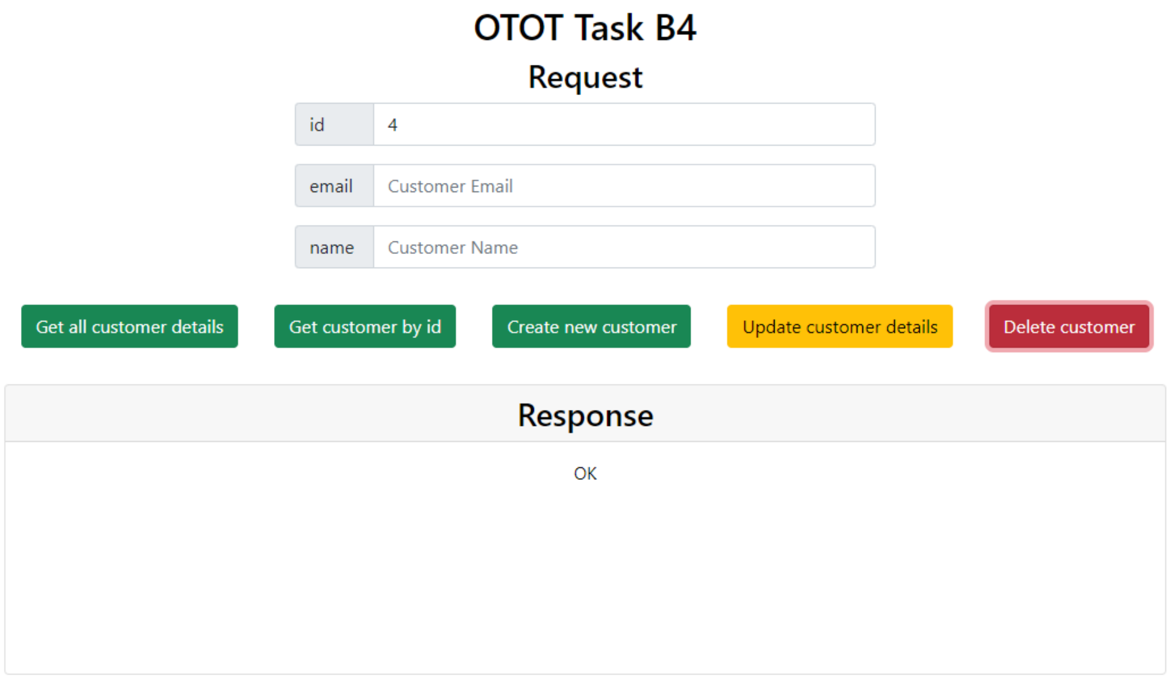
d. `Update customer details`



id 4
email jacob@hotmail.com
name Jacob

OK

e. `Delete customer`



id 4
email Customer Email
name Customer Name

OK