

Project Details

Task: B1 - B4
Done by: Tan Wei Jie (A0202017B)
Repo Link: https://github.com/tanweijie123/CS3219_Sandbox/tree/main/Task_B
Backend Link: <http://tanwj.link:8080/api/book>
Frontend Link: <http://tanwj.link:3000/>
Autodeployment Link: <https://asia-southeast1-cs3219b.cloudfunctions.net/api/api/book>

Note: All of the links above uses the same database at tanwj.link:27017

Overview

This project is about creating a password manager using React App for frontend, REST API for backend CRUD operations, and MongoDB for data storage.
Each log contains a website, id and password used for the website.

Task B1: A simple backend using REST API

/GET

When the backend is accessed, it will perform a `/GET` operation. This will list all the passwords currently stored in this application.

GEThttp://tanwj.link:8...

+

...

http://tanwj.link:8080/api/book

GET

http://tanwj.link:8080/api/book

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

| KEY | VALUE |
|-----|-------|
| Key | Value |

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

```
{
  "status": "success",
  "message": "Book retrieved successfully",
  "data": [
    {
      "_id": "614c9eca6d5ed7b8b83d2f89",
      "create_date": "2021-09-23T15:35:38.011Z",
      "name": "nusmail.com",
      "id": "e0415826@u.nus.edu",
      "pw": "BestPassw0rd",
      "__v": 0
    },
    {
      "_id": "614ea21a1189c357d45b277a",
      "create_date": "2021-09-25T04:14:18.902Z",
      "name": "google.com",
      "id": "tanwj",
      "pw": "GoodPassword"
```

/POST to add website

To add a website, perform a /POST operation.

POSThttp://tanwj.link:8080/api/book/

POST

http://tanwj.link:8080/api/book/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE |
|-------------------------------------|------|---------------|
| <input checked="" type="checkbox"/> | name | WritingReadMe |
| <input checked="" type="checkbox"/> | id | readme |
| <input checked="" type="checkbox"/> | pw | pass1234 |

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "New site created!",

3

"data": {

4

"_id": "6157d2edd51a4be915e437c2",

5

"create_date": "2021-10-02T03:33:01.510Z",

6

"name": "WritingReadMe",

7

"id": "readme",

8

"pw": "pass1234",

9

"__v": 0

10

}

11

}

/GET for individual item

To perform a /GET for individual items, append the website name to the url. Note that each website name is unique.

GEThttp://tanwj.link:8080/api/book/WritingReadMe

GET

http://tanwj.link:8080/api/book/WritingReadMe

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "Site details found...",

3

"data": [

4

{

5

"_id": "6157d2edd51a4be915e437c2",

6

"create_date": "2021-10-02T03:33:01.510Z",

7

"name": "WritingReadMe",

8

"id": "readme",

9

"pw": "pass1234",

10

"__v": 0

11

}

12

]

13

}

/PUT to edit a website

To edit a website, use `/PUT` with the website name appended to the url. For example, in this case, we changed the password from `pass1234` to `password1`.

PUThttp://tanwj.link:8080/api/book/WritingReadMe

PUThttp://tanwj.link:8080/api/book/WritingReadMe

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE |
|-------------------------------------|------|---------------|
| <input checked="" type="checkbox"/> | name | WritingReadMe |
| <input checked="" type="checkbox"/> | id | readme |
| <input checked="" type="checkbox"/> | pw | password1 |

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "message": "WritingReadMe Info updated",
3   "data": {
4     "_id": "6157d2edd51a4be915e437c2",
5     "create_date": "2021-10-02T03:33:01.510Z",
6     "name": "WritingReadMe",
7     "id": "readme",
8     "pw": "password1",
9     "__v": 0
10  }
11 }
```

/DELETE to delete a website

To delete a website, use `/DELETE` with the website name.

DELhttp://tanwj.link:8080/api/book/WritingReadMe

DELETEhttp://tanwj.link:8080/api/book/WritingReadMe

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "status": "success",
3   "message": "1 Site deleted",
4   "data": {
5     "deletedCount": 1
6   }
7 }
```

Some edge cases

- Trying to insert same website name.

In this system, website names are the "primary key". In this example, we try to insert another `nusmail.com` website, but it fails because the name already exists.

since the website already exist, it will reply with a message, "Site name already exist. Use another name for this site."

POSThttp://tanwj.link:8080/api/book/

POST

http://tanwj.link:8080/api/book/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE |
|-------------------------------------|------|-------------|
| <input checked="" type="checkbox"/> | name | nusmail.com |
| <input checked="" type="checkbox"/> | id | readme |
| <input checked="" type="checkbox"/> | pw | password1 |

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

Site name already exist. Use another name for this site.

2. Trying to access unavailable website.

In the `/DELETE to delete a website` section, we had deleted `WritingReadMe` website, so if we were to access it again, it will reply with a status 404 with a message "Site details not found..."

GEThttp://tanwj.link:8080/api/book/WritingReadMe

GET

http://tanwj.link:8080/api/book/WritingReadMe

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "Site details not found...",

3

"data": []

4

}

3. Trying to insert / update without full details.

In this system, we require website name, id and password to be filled. If there are empty fields, it will return a error message "name, id, pw are required."

POSThttp://tanwj.link:8080/api/book/

POST

http://tanwj.link:8080/api/book/

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE |
|-------------------------------------|------|-------------|
| <input checked="" type="checkbox"/> | name | newsite.com |
| <input checked="" type="checkbox"/> | id | |
| <input checked="" type="checkbox"/> | pw | pass_only |

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeHTML

1name, id, pw are required.

PUThttp://tanwj.link:8080/api/book/nusmail.com

PUT

http://tanwj.link:8080/api/book/nusmail.com

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

| | KEY | VALUE |
|-------------------------------------|------|-------------|
| <input checked="" type="checkbox"/> | name | nusmail.com |
| <input checked="" type="checkbox"/> | id | id_only |
| <input checked="" type="checkbox"/> | pw | |

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeHTML

1name, id, pw are required.

4. Trying to delete non-existent website.

Since website name are unique, there are only 1 or 0 entry with such website name. As such, when trying to delete a non-existent website, it will return success with deleteCount: 0. Note that this message is different from the `/DELETE to`

delete a website section.

DELhttp://tanwj.link:80...

+

...

http://tanwj.link:8080/api/book/WritingReadMe

DELETE

⌵

http://tanwj.link:8080/api/book/WritingReadMe

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON ⌵

⌵

↺

1

{

2

"status": "success",

3

"message": "0 Site deleted",

4

"data": {

5

"deletedCount": 0

6

}

7

}

Task B2: Testing using CI

In this step, we test the backend using Mocha/Chai and Github Actions as the automating tester.

build_and_test

succeeded 15 hours ago in 34s

> ✓ Set up job

> ✓ Run actions/checkout@v2

> ✓ Check Path

> ✓ Configure Node.js and run

> ✓ Start Docker for MongoDB

> ✓ Run npm ci

▼ ✓ Run npm test

1 ▶ Run npm test

8

9 > functions@ test /home/runner/work/CS3219_Sandbox/CS3219_Sandbox/Task_B/firebase/functions

10 > mocha --require @babel/register test.js --timeout 10000 --exit

11

12 {"severity":"WARNING","message":"Warning, FIREBASE_CONFIG and GCP_PROJECT environment variables are missing. Initializing firebase-admin will fail"}

13 Db connected successfully

14 Running RestHub on port 8080

15

16

17 Book

18 GET /api/book

19 ✓ should get all book record (3137ms)

20 POST /api/book

21 ✓ add new password record (509ms)

22 GET /api/book

23 ✓ get that added record (243ms)

24 PUT /api/book

25 ✓ update that added record (482ms)

26 DELETE /api/book

27 ✓ delete that added record (244ms)

28 GET /api/book

29 ✓ get that added record (246ms)

30

31

32 6 passing (5s)

33

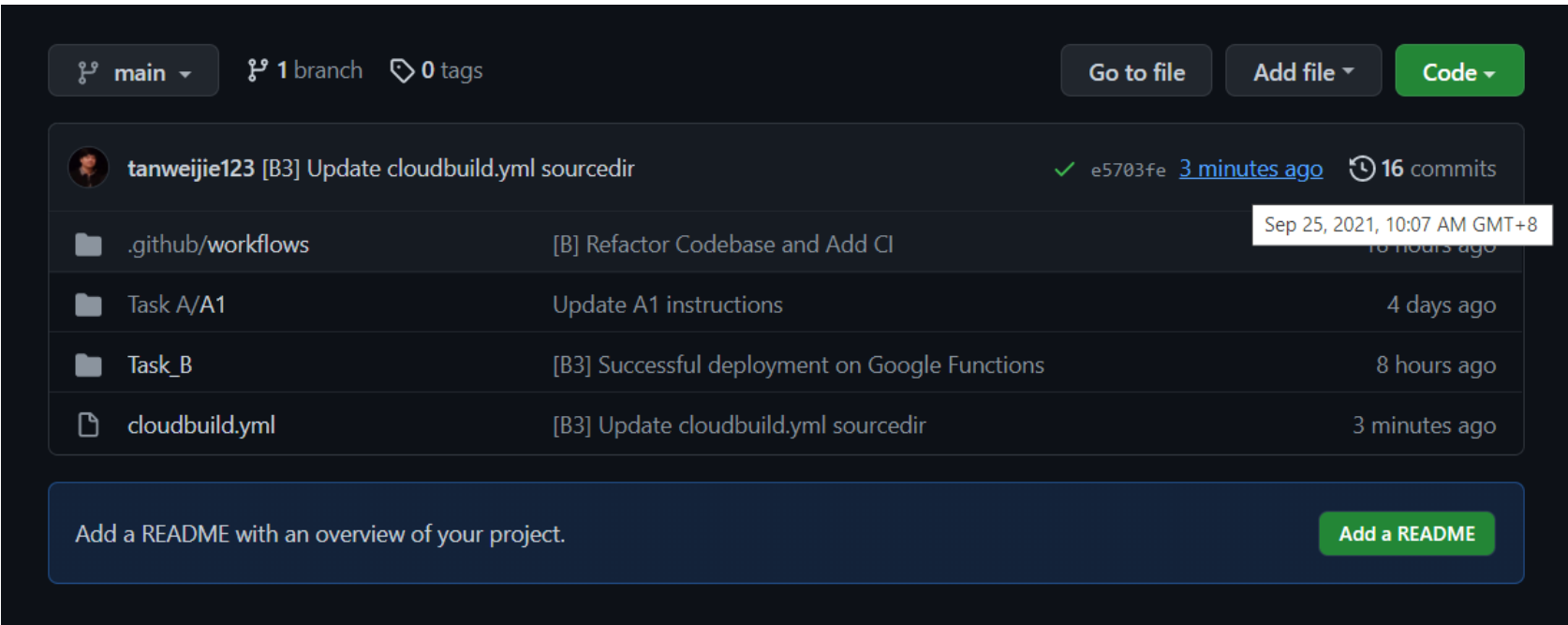
> ✓ Post Configure Node.js and run

> ✓ Post Run actions/checkout@v2

> ✓ Complete job

Task B3: Deployment through CD

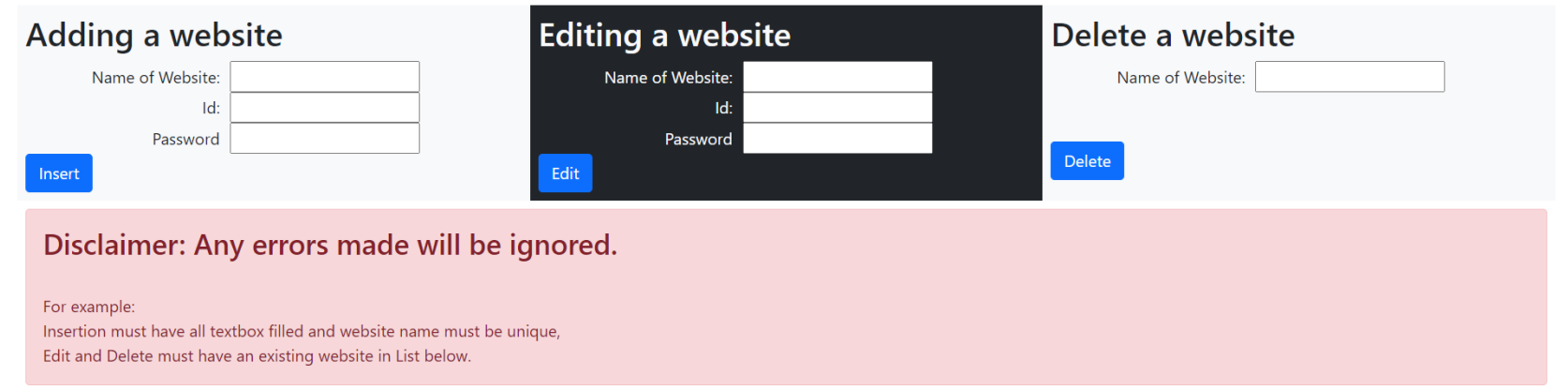
In this step, we use Google Functions as the serverless service and autodeploy our functions when commit. It will automatically deploy the function to `https://asia-southeast1-cs3219b.cloudfunctions.net/api/api/book`



| Google Cloud Platform cs3219b | | | | | | | | | | |
|-------------------------------|---|------|-----------------|---------|------------|------------------|-------------------|-----------------------|-----------------------|---------|
| Cloud Functions Functions | | | | | | | | | | |
| Filter functions | | | | | | | | | | |
| | | Name | Region | Trigger | Runtime | Memory allocated | Executed function | Last deployed | Authentication | Actions |
| <input type="checkbox"/> | ✓ | api | us-central1 | HTTP | Node.js 14 | 256 MB | api | 25 Sep 2021, 02:09:57 | Allow unauthenticated | ⋮ |
| <input type="checkbox"/> | ✓ | api | asia-southeast1 | HTTP | Node.js 14 | 256 MB | api | 25 Sep 2021, 10:10:30 | Allow unauthenticated | ⋮ |

Task B4: Frontend

In this step, we use React with Bootstrap to create a simple frontend to handle the request.



List of Password:

| | | | | |
|--|-------------------------------------|---------------------------------------|--------------------|-------------------|
| nusmail.com e0415826@u.nus.edu BestPassw0rd | google.com tanwj GoodPassworD | google2.com tanwj2 GoodPassworD | tester bb cc | test1 aa bb |
| FrontEndTester.sku.com.sg a0202017b passw0rd~!@# | | | | |

Instructions on how to run

In this section, we will go through through how to run the application given that you have downloaded from the [repo link](#)

Task B1

1. Run `npm install` to download the dependencies
2. Run `node index2` to start the server. *Note: index is used for Google Functions (serverless deployment), and index2 is used for local deployment.*
3. When you access [it](#), it will display an introduction message, and the main backend is at `/api/book`.



4. For demo of GET, POST, PUT, DELETE functions using Postman, refer to the overview section [here](#).

Task B2

1. To run the test locally, run `npm test` in the working directory. You should see this screen.

```
tanwj@tanwj-VirtualBox:~/Desktop/Git/CS3219_Sandbox/Task_B/firebase/functions$ npm test

> functions@ test /home/tanwj/Desktop/Git/CS3219_Sandbox/Task_B/firebase/functions
> mocha --require @babel/register test.js --timeout 10000 --exit

{"severity":"WARNING","message":"Warning, FIREBASE_CONFIG and GLOUD_PROJECT environment variables are missing. Initializing firebase-admin will fail"}
Db connected successfully
Running RestHub on port 8080

Book
  GET /api/book
    ✓ should get all book record (197ms)
  POST /api/book
    ✓ add new password record (96ms)
  GET /api/book
    ✓ get that added record
  PUT /api/book
    ✓ update that added record
  DELETE /api/book
    ✓ delete that added record
  GET /api/book
    ✓ get that added record

6 passing (376ms)

tanwj@tanwj-VirtualBox:~/Desktop/Git/CS3219_Sandbox/Task_B/firebase/functions$
```

2. If you insert this `.yaml` file into the `.github/workflows/` directory, Github Actions will automate the testing for you. The `.yaml` used can be found [here](#).

35 lines (26 sloc) | 742 Bytes

```
1  name: b_testing
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  env:
10   working-directory: ${github.workspace}/Task_B/firebase/functions
11
12  jobs:
13    build_and_test:
14      runs-on: ubuntu-latest
15
16      steps:
17        - uses: actions/checkout@v2
18
19        - name: Check Path
20          run: echo ${env.working-directory}
21
22        - name: Configure Node.js and run
23          uses: actions/setup-node@v2
24
25        - name: Start Docker for Mongodb
26          run: docker run -d -p 27017:27017 mongo
27
28        - run: npm ci
29          working-directory: ${env.working-directory}/
30
31        - run: npm test
32          working-directory: ${env.working-directory}/
33          env:
34            MONGODB_HOST: mongo
35            MONGODB_PORT: 27017
```

Task B3

For this step, it is simple to setup a continuous deployment to Google Functions.

1. Create a trigger on Google Functions.

Google Cloud Platform

cs3219b

Search products and resources

Cloud Build

Triggers

+ CREATE TRIGGER

CONNECT REPOSITORY

MANAGE REPOSITORIES

Dashboard

History

Triggers

Settings

Filter

Enter property name or value

| Name | Description | Repository | Event | Build configuration | Status | |
|----------------|-------------|-----------------------------|----------------|---------------------|---------|-----|
| autodeployment | - | tanweijie123/CS3219_Sandbox | Push to branch | Auto-detected | Enabled | RUN |

2. Add the `.yaml` file into your repo root so it will automatically trigger the deployment. The `.yaml` used can be found [here](#).

11 lines (11 sloc) | 223 Bytes

```
1  steps:
2    - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
3    args:
4      - gcloud
5      - functions
6      - deploy
7      - api
8      - --region=asia-southeast1
9      - --source=./Task_B/firebase/functions/
10     - --trigger-http
11     - --runtime=nodejs14
```

For both steps above, you can refer to the one-time setup instructions [here](#).

Task B4

- 1. Run `npm install` to download the dependencies
- 2. Run `npm start` to start the webpage.

Conclusion

You can test the project live using the links given in the [Project Details](#).

Resources

Resources that are used and referred to during the creation of this project.

| Desc | Link |
|---|---|
| Guide on how to create a simple REST API with express | https://medium.com/@dinyangetoh/how-to-build-simple-restful-api-with-nodejs-expressjs-and-mongodb-99348012925d |
| Google Function's Continuous Deployment Guide | https://cloud.google.com/build/docs/deploying-builds/deploy-functions |
| Creating a basic React App | https://www.datarmatics.com/reactjs/ |
| Video on React with REST API | https://youtu.be/qXvFaEkkZH8 |
| Making API calls in React App | https://www.kindsonthegenius.com/how-to-make-rest-api-calls-in-react-get-post-put-delete/ |
| React with Bootstrap | https://react-bootstrap.github.io/getting-started/introduction/ |