

# CS3219 Task B: CRUD Application Task

**Name:** Koh Vinleon

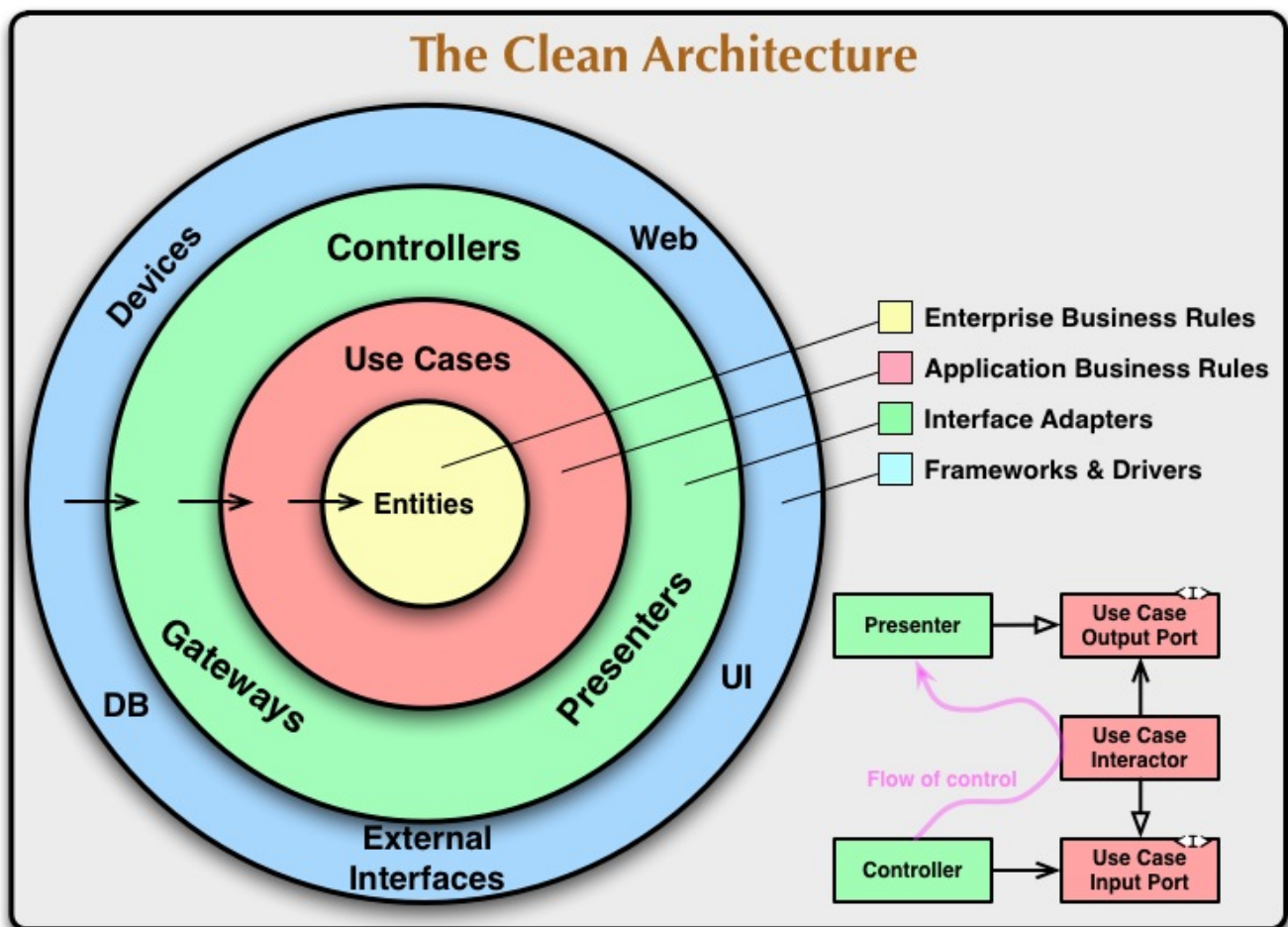
**Matric Number:** A0202155W

**GitHub Link:** <https://github.com/glatiuden/CS3219-OTOT-TaskB>

## Task B1: Implementing Backend

This is an attempt in building a (semi) Clean Architecture Node.js backend.

Clean Architecture



Read more at [Clean Coder Blog](#)

### Layer description:

- Entities: Contain enterprise business model/object
- Use Cases: Contain application business rules/logic
- Interface Adapter: Contains a set of adapters that convert data from entities/use-case layer to external dependencies such as DB or Web/HTTP
- Frameworks/Driver: Compose of frameworks and tools (DB, Web Frameworks)

### References

- [Using Clean Architecture for Microservice APIs in Node.js with MongoDB and Express](#)
- [Rules for clean code](#)
- [Node clean code architecture](#)
- [Application layer - use-cases](#)
- [Domain-driven Design articles](#)
- [Screaming architecture](#)
- [What is screaming architecture](#)
- [Clean architecture use-case structure](#)
- [Denormalize data](#)
- [Mongoose Database](#)
- [Expression documentation on API](#)
- [Bodyparser](#)
- [Winston-express for HTTP logging](#)
- [Winston for error logging](#)
- [Regex route express](#)

## Set Up

**Database Used:** Atlas MongoDB

**Libraries Used:** [Winston](#), [Nodemon](#), [Mongoose](#) and [Lodash](#)

Please ensure you are in the `/backend` folder (`cd backend`).

Please create a `.env` file in the backend directory with the following credentials.

```
MONGO_USERNAME="admin"
MONGO_PASSWORD="3YHYkUdqNUMykugo"
MONGO_DB="cs3219-otot-task-b"
```

## Install the necessary modules

```
npm install
```

## Start the server

```
npm run dev
```

## Design

- All the endpoints are structured in this format `{URL}/api/{COLLECTION_NAME}`.

### Note API

Method	Route	Description
POST	/api/note	Create a new note
GET	/api/note	Get all notes
GET	/api/note/:note_id	Get note by ID
PUT	/api/note	Update a note
DELETE	/api/note/:note_id	Soft delete a note
DELETE	/api/note/hard-delete/:note_id	Hard delete a note

- The results returned by the API must be `data`.
- For `GET`, there are two variants: one will get a specific record by `ID` while the other will get all the records from the database.
- For `DELETE`, there are two variants: one will perform a soft delete while the another will perform a hard delete.

### Error Resiliency

- The controllers which require parameters use a validator middleware to ensure the required parameters are in place. If there are missing parameters or invalid data, the response (error) code is `422`.
- If there is an error encountered during the execution of a query, such as a record not found or an internal error, the response (error) code will be `404`.

### Endpoint

- Localhost: <http://localhost:5000>
- Deployed Endpoint: <https://asia-southeast1-cs3219-otot-task-b-325509.cloudfunctions.net/cs3219-otot-task-b-dev-app>

▶ Run in Postman

Alternatively, you may want to import it to your workspace via the [JSON link](#) or download the Postman JSON file in the [Github Directory](#).

## Demonstration

### POST (CREATE)

- Method: **POST**
- Route: **/api/note**
- Description: Create new note
- Data Required (JSON): **title** (required), **description** (required)

### Success (200)

POST Create Note X + ... CS3219-TaskB

CS3219-TaskB / Localhost / Create Note

POST http://localhost:5000/api/note/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "title": "Hello",
3   "description": "Hello This Is A New Note"
4 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "_id": "6147ecd8bdf3052cc4aab5cc",
4     "title": "Hello",
5     "description": "Hello This Is A New Note",
6     "created_at": "2021-09-20T02:07:20.972Z",
7     "updated_at": "2021-09-20T02:07:20.972Z"
8   }
9 }
```

Status: 200 OK Time: 130 ms Size: 448 B Save Response

- For ease of demonstration and testing, the **note\_id** returned in the body will be saved as a variable in Postman's local environment to be used in the subsequent requests.

POST Create Note X + ... CS3219-TaskB

CS3219-TaskB / Localhost / Create Note

POST http://localhost:5000/api/note/ Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

```
1 var jsonData = JSON.parse(responseBody);
2 postman.setEnvironmentVariable("note_id", jsonData.data._id);
3
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

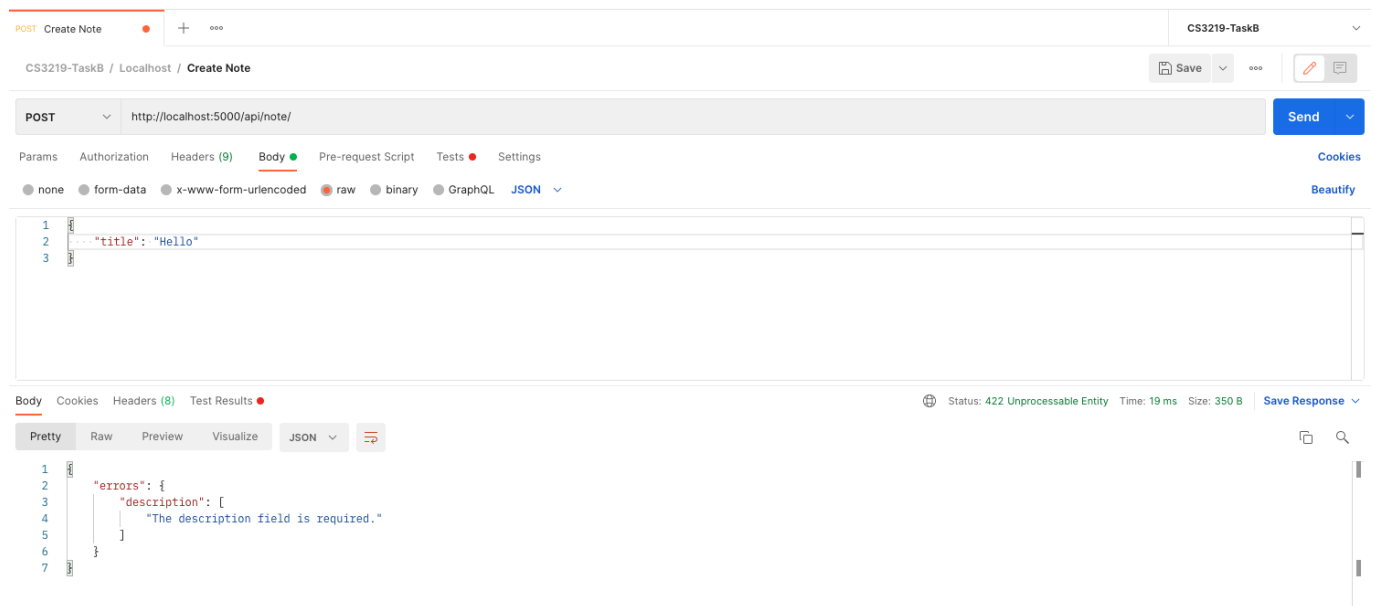
SNIPPETS

- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable

Response

## Error (422)

- Occurs due to missing data fields.



The screenshot shows a REST client interface for a POST request to `http://localhost:5000/api/note/`. The request body is a JSON object: `{ "title": "Hello" }`. The response status is `422 Unprocessable Entity` with a message: `"The description field is required."`.

```
POST http://localhost:5000/api/note/

{
  "title": "Hello"
}
```

Status: 422 Unprocessable Entity Time: 19 ms Size: 350 B Save Response

```
{
  "errors": {
    "description": [
      "The description field is required."
    ]
  }
}
```

- Optimally, there can be an additional Error `404` if a note with the same `title` and `description` already exists in the database. However, this is omitted as it does not fit the context of a "note" application and for ease of testing.

## GET (Retrieve)

- Method: **GET**
- Route: **/api/note**
- Description: Get all notes

## Success (200)

GET

http://localhost:5000/api/note

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

BodyCookiesHeaders (8)Test Results

PrettyRawPreviewVisualizeJSON

```
1  {
2    "data": [
3      {
4        "_id": "6147ecd8bdf3052cc4aab5cc",
5        "title": "Hello",
6        "description": "Hello This Is A New Note",
7        "created_at": "2021-09-20T02:07:20.972Z",
8        "updated_at": "2021-09-20T02:07:20.972Z"
9      },
10     {
11       "_id": "6146de62ac43d74496fa7daf",
12       "title": "Hello",
13       "description": "Hello This Is A New Note",
14       "created_at": "2021-09-19T06:53:22.172Z",
15       "updated_at": "2021-09-19T06:53:22.172Z"
16     }
17   ]
18 }
```

Status: 200 OKTime: 40 msSize: 623 BSave Response

- Optimally, it can be an additional Error **204** (no content) if no notes are in the collections. I believe it's a debate between 204 and returning 200 with an empty array.
- For this task, I have chosen to follow 200 with an empty array.

## GET (Retrieve By ID)

- Method: **GET**
- Route: **/api/note/:note\_id**
- Description: Get all notes

## Success (200)

GETGet Note

CS3219-TaskB

CS3219-TaskB / Localhost / Get Note

GET

http://localhost:5000/api/note/{(note\_id)}

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

BodyCookiesHeaders (8)Test Results

Status: 200 OKTime: 39 msSize: 448 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    "data": {
3      "_id": "6147ecd8bdf3052cc4aab5cc",
4      "title": "Hello",
5      "description": "Hello This Is A New Note",
6      "created_at": "2021-09-20T02:07:20.972Z",
7      "updated_at": "2021-09-20T02:07:20.972Z"
8    }
9  }
```

## Error (404)

- Valid parameter present but data is not found in the database

GET

http://localhost:5000/api/note/6147f0dff6e558b2a6b24fa8

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

BodyCookiesHeaders (8)Test Results

Status: 404 Not FoundTime: 69 msSize: 327 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    "errors": "Note 6147f0dff6e558b2a6b24fa8 not found."
3  }
```

## Error (422)

- Caused by invalid parameter (In this scenario, the `note_id` is not a valid ObjectId).

GET Get Note

CS3219-TaskB

CS3219-TaskB / Localhost / Get Note

GET

http://localhost:5000/api/note/123

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (8)

Test Results

Status: 422 Unprocessable Entity

Time: 12 ms

Size: 342 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "errors": {
3     "note_id": [
4       "The note id format is invalid."
5     ]
6   }
7 }
```



## PUT (Update)

- Method: **PUT**
- Route: **/api/note/**
- Description: (Partial) Update existing note
- Data Required (JSON): **\_id** (required), **title** (optional), **description** (optional)

## Success (200)

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

- URL:** http://localhost:5000/api/note/
- Method:** PUT
- Body (JSON):**

```
{  "id": "{{note_id}}",  "title": "Update my note",  "description": "This note is updated"}
```
- Status:** 200 OK
- Time:** 61 ms
- Size:** 453 B
- Response (JSON):**

```
{  "data": {    "id": "6147ecd8bdf3052cc4aab5cc",    "title": "Update my note",    "description": "This note is updated",    "created_at": "2021-09-20T02:07:20.972Z",    "updated_at": "2021-09-20T02:10:23.611Z"  }}
```

## Error (404)

- Valid parameter present but data is not found in the database

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

- URL:** http://localhost:5000/api/note/
- Method:** PUT
- Body (JSON):**

```
{  "id": "6147f0dff6e558b2a6b24fa8",  "title": "Update my note",  "description": "This note is updated"}
```
- Status:** 404 Not Found
- Time:** 71 ms
- Size:** 327 B
- Response (JSON):**

```
{  "errors": "Note 6147f0dff6e558b2a6b24fa8 not found."}
```

## Error (422)

- The data field `_id` is missing
- As we are updating a note, `_id` is required to know which record to update.

The screenshot shows the Postman interface for a PUT request to `http://localhost:5000/api/note/`. The request body is a JSON object: `{"title": "Update my note,", "description": "This note is updated"}`. The response is a 422 Unprocessable Entity error: `{"errors": {"_id": ["The id field is required."]}}`.



## Delete (Soft Delete)

- Method: **DELETE**
- Route: **/api/note/:note\_id**
- Description: Soft-delete an existing note

## Success (200)

▼ / Localhost / Positive Test Case / Delete Note

Save ⌵ ⋮

DELETE ⌵ http://localhost:5000/api/note/((note\_id))

Send ⌵

Params Authorization Headers (7) Body Pre-request Script Tests Settings


Cookies

Query Params

	KEY	VALUE	DESCRIPTION	⋮	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (8) Test Results

⊞ Status: 200 OK Time: 30 ms Size: 471 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 



```
1  {
2    "is_deleted": true,
3    "data": {
4      "_id": "6147ecd8bdf3052cc4aab5cc",
5      "title": "Update my note",
6      "description": "This note is updated",
7      "created_at": "2021-09-20T02:07:20.972Z",
8      "updated_at": "2021-09-20T02:10:23.611Z"
9    }
10 }
```

## Error (404)

- Valid parameter but data has already been soft-deleted

▼ / Localhost / Positive Test Case / Delete Note

Save ⌵ ⋮

DELETE ⌵ http://localhost:5000/api/note/((note\_id))

Send ⌵

Params Authorization Headers (7) Body Pre-request Script Tests Settings


Cookies

Query Params

	KEY	VALUE	DESCRIPTION	⋮	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (8) Test Results

⊞ Status: 404 Not Found Time: 18 ms Size: 347 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 

```
1  {
2    "errors": "Note 6147ecd8bdf3052cc4aab5cc has already been soft-deleted."
3  }
```

- Valid parameter present but data is not found in the database

DELETE ▼ http://localhost:5000/api/note/6147f0dff6e558b2a6b24fa8 Send ▼

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

---

Body Cookies Headers (8) Test Results Status: 404 Not Found Time: 39 ms Size: 327 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

```

1
2
3
"errors": "Note 6147f0dff6e558b2a6b24fa8 not found."

```

## Error (422)

- Caused by invalid parameter (In this scenario, the `note_id` is not a valid ObjectId).
- As we are soft deleting a note, `_id` is required to know which record to soft delete.

DELETE ▼ http://localhost:5000/api/note/123 Send ▼

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

---

Body Cookies Headers (8) Test Results Status: 422 Unprocessable Entity Time: 26 ms Size: 342 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

```

1
2
3
4
5
6
7
"errors": {
  "note_id": [
    "The note id format is invalid."
  ]
}

```

## Delete (Hard Delete)

- Method: **DELETE**
- Route: **/api/note/hard-delete/:note\_id**
- Description: Hard-delete an existing note

## Success (200)

Localhost / Positive Test Case / Hard Delete Note

Save

DELETE

http://localhost:5000/api/note/hard-delete/{(note\_id)}

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION		Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 38 ms

Size: 286 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"is\_deleted": true

## Error (404)

- Valid parameter present but data is not found in the database

Localhost / Positive Test Case / Hard Delete Note

Save

DELETE

http://localhost:5000/api/note/hard-delete/{(note\_id)}

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION		Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 404 Not Found

Time: 23 ms

Size: 327 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"errors": "Note 6147ecd8bdf3052cc4aab5cc not found."

## Error (422)

- Caused by invalid parameter (In this scenario, the `note_id` is not a valid ObjectId).
- As we are hard deleting a note, `_id` is required to know which record to hard delete.

DELETE

▼

http://localhost:5000/api/note/hard-delete/123

Send

▼

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 422 Unprocessable Entity

Time: 23 ms

Size: 342 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "errors": {
3      "note_id": [
4        "The note id format is invalid."
5      ]
6    }
7  }
```

## Task B2: Testing through Continuous Integration (CI)

**Test Frameworks:** Mocha & Chai

The tests will covers all the available requests **POST**, **GET** (get by ID & get all), **PUT** and **DELETE** (soft delete and hard delete), which is split into positive (200) and negative (404 and 422) test cases. This ensures that the API endpoint responses are accurate.

### Run the test locally

```
npm run test
```

### Running the test through CI

Code Snippet from **.travis.yml**

```
language: node_js
node_js:
  - node
services:
  - mongodb
install:
  - npm install
jobs:
  include:
    - stage: test
      script: npm test
```

Travis has been integrated into the repository. **npm run test** is executed whenever the codes are pushed into the repository, under the job stage **test**.

This is a screenshot of an example of the test.

```
$ npm install
$ npm test

> [secure]i@1.0.0 test
> env TS_NODE_COMPILER_OPTIONS='{ "module": "commonjs" }' mocha --exit --timeout 10000 -r ts-node/register './src/tests/*.ts'

Setting up database...
Listening on port 5000

Notes Postive Test Cases
POST /
Successfully connected to DB
verbose: Note created {"note_id":"6144cdbf17024daea00d608f","timestamp":"2021-09-17T17:17:54.819Z"}
http: HTTP POST /api/note 200 367ms {"metadata":{"timestamp":"2021-09-17T17:17:54.825Z"}}
✓ Should create a note (3705ms)
GET /
verbose: Notes retrieved {"notes_count":2005,"timestamp":"2021-09-17T17:17:56.384Z"}
http: HTTP GET /api/note/ 200 1557ms {"metadata":{"timestamp":"2021-09-17T17:17:56.396Z"}}
✓ Should fetch all notes (1566ms)
verbose: Note retrieved {"note_id":"6144cdbf17024daea00d608f","timestamp":"2021-09-17T17:17:56.634Z"}
http: HTTP GET /api/note/6144cdbf17024daea00d608f 200 229ms {"metadata":{"timestamp":"2021-09-17T17:17:56.641Z"}}
✓ Should fetch single note that was previously created (243ms)
PUT /
verbose: Note updated {"note_id":"6144cdbf17024daea00d608f","timestamp":"2021-09-17T17:17:57.331Z"}
http: HTTP PUT /api/note 200 689ms {"metadata":{"timestamp":"2021-09-17T17:17:57.337Z"}}
✓ Should update note that was previously created (693ms)
DELETE /
verbose: Note soft-deleted {"note_id":"6144cdbf17024daea00d608f","timestamp":"2021-09-17T17:17:57.790Z"}
http: HTTP DELETE /api/note/6144cdbf17024daea00d608f 200 457ms {"metadata":{"timestamp":"2021-09-17T17:17:57.800Z"}}
✓ Should soft delete note that was previously created (462ms)
verbose: Note hard-deleted {"note_id":"6144cdbf17024daea00d608f","timestamp":"2021-09-17T17:17:58.260Z"}
http: HTTP DELETE /api/note/hard-delete/6144cdbf17024daea00d608f 200 457ms {"metadata":{"timestamp":"2021-09-17T17:17:58.261Z"}}
✓ Should hard delete note that was previously created (459ms)

Notes Negative Test Cases
POST /
http: HTTP POST /api/note 422 1ms {"metadata":{"timestamp":"2021-09-17T17:17:58.266Z"}}
✓ Should create a note
GET /
http: HTTP GET /api/note/123 422 1ms {"metadata":{"timestamp":"2021-09-17T17:17:58.271Z"}}
✓ Should not fetch a note due to invalid ID
error: Note 613f4186abe8ac519b619877 not found. {"timestamp":"2021-09-17T17:17:58.502Z"}
http: HTTP GET /api/note/613f4186abe8ac519b619877 404 227ms {"metadata":{"timestamp":"2021-09-17T17:17:58.503Z"}}
✓ Should not fetch a note due to object not found (230ms)
PUT /
http: HTTP PUT /api/note 422 1ms {"metadata":{"timestamp":"2021-09-17T17:17:58.507Z"}}
✓ Should not update any note due to invalid ID
error: Note 613f4186abe8ac519b619877 not found. {"timestamp":"2021-09-17T17:17:58.737Z"}
http: HTTP PUT /api/note 404 226ms {"metadata":{"timestamp":"2021-09-17T17:17:58.738Z"}}
✓ Should not update any note due to object not found (230ms)
DELETE /
http: HTTP DELETE /api/note/123 422 0ms {"metadata":{"timestamp":"2021-09-17T17:17:58.749Z"}}
✓ Should not soft delete note due to invalid ID
error: Note 613f4186abe8ac519b619877 not found. {"timestamp":"2021-09-17T17:17:58.987Z"}
http: HTTP DELETE /api/note/613f4186abe8ac519b619877 404 227ms {"metadata":{"timestamp":"2021-09-17T17:17:58.988Z"}}
✓ Should not soft delete note due to object not found (232ms)
http: HTTP DELETE /api/note/hard-delete/123 422 1ms {"metadata":{"timestamp":"2021-09-17T17:17:58.994Z"}}
✓ Should not hard delete note due to invalid ID
error: Note 613f4186abe8ac519b619877 not found. {"timestamp":"2021-09-17T17:17:59.230Z"}
http: HTTP DELETE /api/note/hard-delete/613f4186abe8ac519b619877 404 226ms {"metadata":{"timestamp":"2021-09-17T17:17:59.231Z"}}
✓ Should not hard delete note due to invalid ID (231ms)

15 passing (8s)

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

## References

- <https://medium.com/@asciidev/testing-a-node-express-application-with-mocha-chai-9592d41c0083>
- <https://gist.github.com/cklanac/81a6f49fabb52b3c95dff397fe62c771>



## Task B3: Deployment through Continuous Deployment (CD)

### Serverless Service: Serverless Google Cloud Functions

This task is accomplished using the Serverless Framework via Google Cloud Functions.

A `serverless.yml` has been set up as a set of instructions to deploy to Google Cloud Functions.

We can either deploy locally or via CD in Travis.

### Deploying locally

```
npm run deploy
```

### Deploying through CD

Similar to Task B2, `npm run deploy` under the job stage `deploy` is executed whenever the codes are pushed into the repository after the `test` stage is completed.

Code snippet from `.travis.yml`

```
- stage: deploy
  install:
  - npm install -g serverless
  before_script:
  - openssl aes-256-cbc -K $encrypted_3c84dc6bbe_key -iv $encrypted_3c84dc6bbe_iv
    -in .env.enc -out .env -d
  - openssl aes-256-cbc -K $encrypted_4e8c5512ae30_key -iv $encrypted_4e8c5512ae30_iv
    -in serverless-key.json.enc -out serverless-key.json -d
  script:
  - npm run deploy
```

This is a screenshot of an example of continuous deployment.

```
$ npm install -g serverless
$ openssl aes-256-cbc -K $encrypted_3c84dcdc6bbe_key -iv $encrypted_3c84dcdc6bbe_iv -in .env.enc -out .env -d
$ openssl aes-256-cbc -K $encrypted_4e8c5512ae30_key -iv $encrypted_4e8c5512ae30_iv -in serverless-key.json.enc -out serverless-key.json -d
$ npm run deploy

> [secure]1@1.0.0 deploy
> sls deploy

Serverless: DOTENV: Loading environment variables from .env:
Serverless:   - MONGO_USERNAME
Serverless:   - MONGO_PASSWORD
Serverless:   - MONGO_DB
Serverless: Configuration warning at 'provider.region': should be equal to one of the allowed values [us-central1, us-east1, us-east4, europe-west1, europe-west2, asia-east2, asia-northeast1, asia-northeast2, us-west2, us-west3, us-west4, northamerica-northeast1, southamerica-east1, europe-west3, europe-west6, australia-southeast1, asia-south1, asia-southeast2, asia-northeast3]
Serverless:
Serverless: Learn more about configuration validation here: http://sls.io/configuration-validation
Serverless:
Serverless: Compiling with Typescript...
Serverless: Using local tsconfig.json
Serverless: Typescript compiled.
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Compiling function "app"...
Serverless: Uploading artifacts...
Serverless: Artifacts successfully uploaded...
Serverless: Updating deployment...
Serverless: Checking deployment update progress...
.....
Serverless: Done...
Service Information
service: [secure]
project: [secure]-325509
stage: dev
region: asia-southeast1

Deployed functions
app
  https://asia-southeast1-[secure]-325509.cloudfunctions.net/[secure]-dev-app

Serverless: Removing old artifacts...
Serverless: Deprecation warnings:

Starting with next major, Serverless will throw on configuration errors by default. Adapt to this behavior now by adding "configValidationMode: error" to service configuration
More Info: https://www.serverless.com/framework/docs/deprecations/#CONFIG_VALIDATION_MODE_DEFAULT

Support for "package.include" and "package.exclude" will be removed with next major release. Please use "package.patterns" instead
More Info: https://www.serverless.com/framework/docs/deprecations/#NEW_PACKAGE_PATTERNS

The command "npm run deploy" exited with 0.
store build cache

Done. Your build exited with 0.
```

The application is deployed to <https://asia-southeast1-cs3219-otot-task-b-325509.cloudfunctions.net/cs3219-otot-task-b-dev-app>.

## References

- <https://www.serverless.com/framework/docs/providers/google/guide>
- <https://blog.travis-ci.com/2019-05-30-setting-up-a-ci-cd-process-on-github>

## Task B4: Implement a frontend

- **Frontend Framework:** Next.js (React.js)
- **UI Framework:** Material-UI

This is an attempt in creating a frontend using Next.js. The web application supports the CRUD operations created in Task B1.

As part of the learning objectives, the codes are structured in an MVC folder structure, along with using React's useReducer as a store.

Please ensure you are in the **frontend** directory (**cd frontend**).

Please create a **.env** file with the following variables:

```
DB_HOST_URL = "https://asia-southeast1-cs3219-otot-task-b-325509.cloudfunctions.net/cs3219-otot-task-b-dev-app"
```

### Install the necessary modules

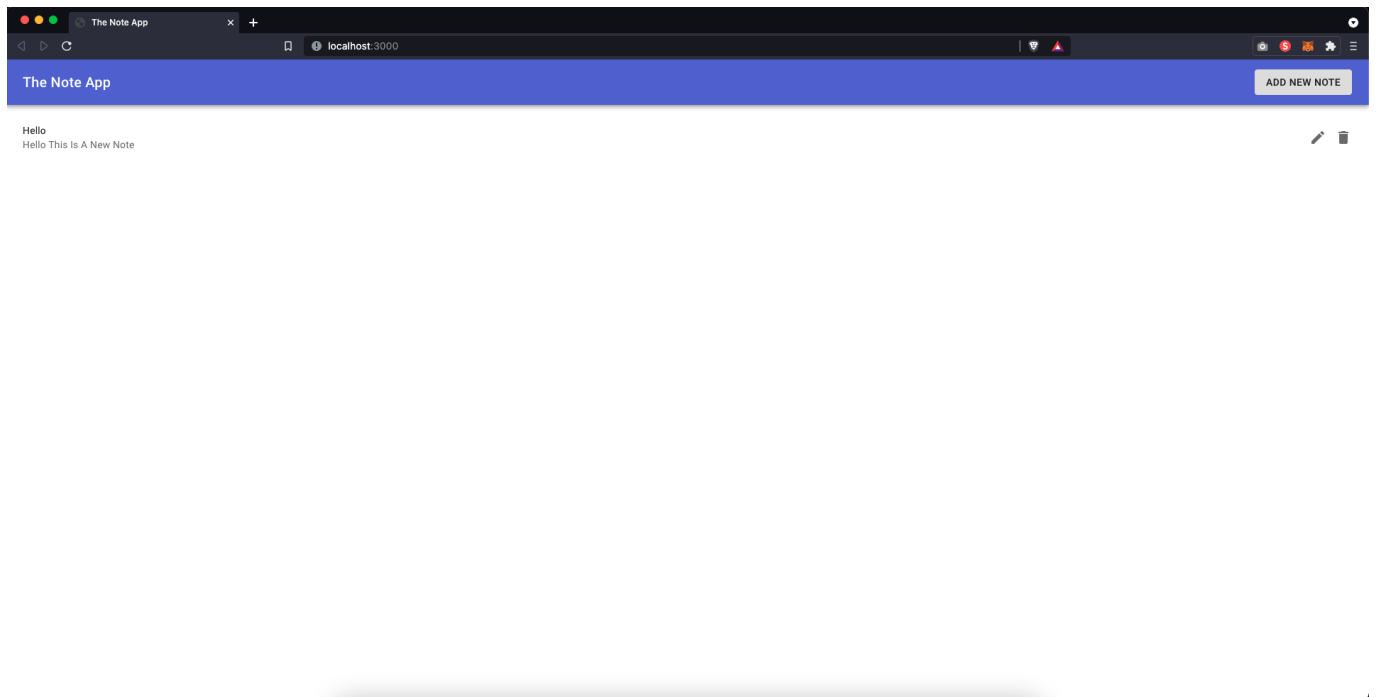
```
npm install
```

### Start the server

```
npm run dev
```

### Endpoint

- Localhost: <http://localhost:5000>
- Deployed Endpoint: <https://cs3219-otot-task-b-325509.as.r.appspot.com/>



## Learning Outcome

- Despite the learning objectives being to use an MVC in real-life frameworks, the newer frameworks are no longer based on the "MVC" structure.
- The closest thing to MVC was Redux, which goes from Reducer -> Store -> View.
- In this task, I tried to replicate as closely as possible by utilizing controllers (which will interact with the APIs) and models (to define the attributes) while using React's reducer and store functionality.