

CS3219 OTOT Task E

- **Name:** Ryan Aidan
- **Matric. Number:** A0218327E
- **Repo Link:** <https://github.com/aidanaden/otot-e>

Tech stack used

- [NextJS](#) (frontend + backend API routes via nextjs' serverless functions)
- [TailwindCSS](#) (frontend styling)
- [Prisma](#) (database ORM)
- MySQL (database used)
- [tRPC](#) (used to set up API routes with end-to-end typesafety)

Task E: Backend cache

Requirements

1. MySQL (SQL database used)
2. NodeJS >= v14.19.2
3. [Postman](#) is used to (manually) test CRUD for the API.

Local Deployment

1. Install docker by clicking [here](#)
2. Install pnpm by clicking [here](#)
3. Copy the `.env-example` file to `.env` and paste the following:

```
# Prisma
DATABASE_URL=mysql://root:root@localhost:3369/mysql

# Next Auth
NEXTAUTH_SECRET=secret

# Calendarific
CALENDARIFIC_API_KEY=00e9be3d9730c3d5504e5c4794788bc428cdeea5

# Redis
REDIS_HOST=localhost
REDIS_PORT=6380
```

4. Run the following commands to deploy the app locally

```
# install packages
pnpm install

# start local mysql db, deploy schema, build source files
pnpm build:local
```

```
# run locally
pnpm start
```

Verify successful deployment

Run the following commands to verify successful local deployment

```
# query the /api/activity endpoint
curl http://localhost:3000/api/activity

# expected output
[]
```

Visit `http://localhost:3000` to confirm successful frontend deployment

Testing via Postman

Import the Postman collection via [this link](#).

Explanation of HTTP requests:

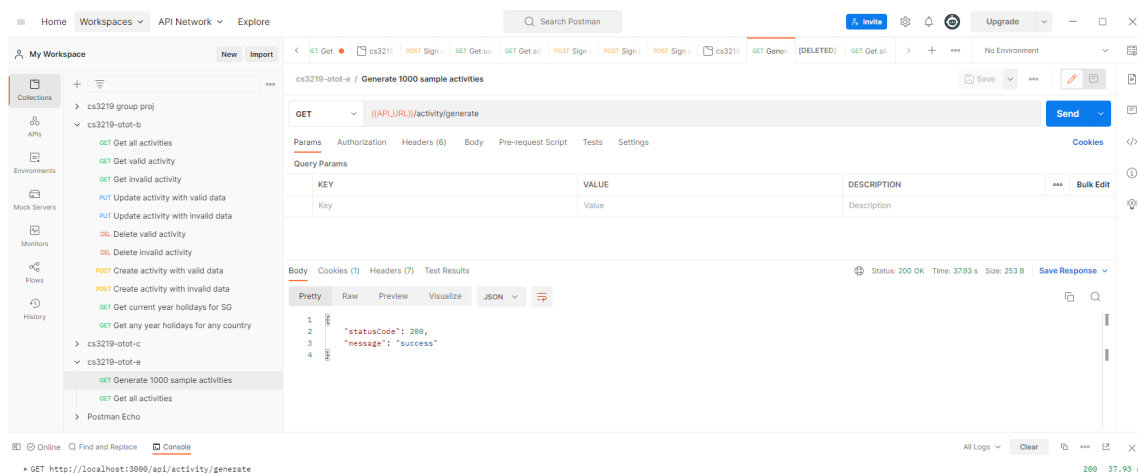
- `activity` : get all activities in database
- `activity/generate` : generate 1000 test activities

Note:

- `params` and `body` would need to be updated manually, where they are necessary.

Screenshots

Generate 1000 activities for testing



First fetch 1000 activities

Postman interface showing a REST client request for `GET http://localhost:3000/api/activity`. The response is a JSON array of activities, including details like `id`, `createdAt`, `updatedAt`, `name`, `location`, `categoryId`, and `category`.

```
1 {
2   "id": "c19mzdou000u08tdwyj507",
3   "createdAt": "2022-10-24T18:29:05.383Z",
4   "updatedAt": "2022-10-24T18:29:05.383Z",
5   "name": "generated name",
6   "location": "generated location",
7   "categoryId": "c19mzdou000u08tdwyj507",
8   "category": {
9     "id": "c19mzdou000u08tdwyj507",
10    "createdAt": "2022-10-24T18:29:05.383Z",
11    "updatedAt": "2022-10-24T18:29:05.383Z",
12    "name": "generated category"
13  }
14 }
15 ]
```

Console logs show the response status and time:

```
• GET http://localhost:3000/api/activity 200 37.93 s
• GET http://localhost:3000/api/activity 200 788 ms
```

Subsequent fetches

Postman interface showing subsequent REST client requests for `GET http://localhost:3000/api/activity`. The response is a JSON array of activities, including details like `id`, `createdAt`, `updatedAt`, `name`, `location`, `categoryId`, and `category`.

```
1 {
2   "id": "c19mzdou000u08tdwyj507",
3   "createdAt": "2022-10-24T18:29:05.383Z",
4   "updatedAt": "2022-10-24T18:29:05.383Z",
5   "name": "generated name",
6   "location": "generated location",
7   "categoryId": "c19mzdou000u08tdwyj507",
8   "category": {
9     "id": "c19mzdou000u08tdwyj507",
10    "createdAt": "2022-10-24T18:29:05.383Z",
11    "updatedAt": "2022-10-24T18:29:05.383Z",
12    "name": "generated category"
13  }
14 }
15 ]
```

Console logs show the response status and time:

```
• GET http://localhost:3000/api/activity/generate 200 37.93 s
• GET http://localhost:3000/api/activity 200 788 ms
• GET http://localhost:3000/api/activity 200 132 ms
• GET http://localhost:3000/api/activity 200 30 ms
• GET http://localhost:3000/api/activity 200 30 ms
```