

Cloud Technologies 2024 - Assignment 2 (Reddit)

- Author: Tomáš Hladký
- Project available: <http://129.114.26.177>
- Project repository: <https://github.com/Tem12/ntnu-cloud-tech-reddit>

Reddit-like web [Science](#) [Books](#) [Space-Around-Us](#)

[My Profile](#) [Logout](#)

Books

Create post

< Previous

Showing 11 - 15 out of 15

> Next

Python for Dummies. You won't believe it, but this book saved my career

[MrBig](#)

26/05/2024, 20:49:15

156

Read People Like a Book. My favourite one this year.

[alexander](#)

26/05/2024, 20:49:15

56

Have you read Hobit?

[MrBig](#)

26/05/2024, 20:49:15

5

Harry Potter will always be my favourite

[alexander](#)

26/05/2024, 20:49:15

14

Project structure

```
ntnu-cloud-tech-reddit/
├── .github/
├── .vscode
├── api
│   ├── data-seed
│   ├── Dockerfile
│   ├── requirements.txt
│   └── ...
├── cache-service
│   ├── Dockerfile
│   ├── requirements.txt
│   └── ...
├── doc
├── frontend
│   ├── Dockerfile
│   └── ...
├── nginx
│   ├── Dockerfile
│   └── nginx.conf
├── .env
├── .gitignore
├── api_requests.postman_collection.json
├── deploy-start-script.sh
├── docker-compose.yaml
├── LICENSE
├── Makefile
└── README.md
```

- `.github`: Github action configuration
- `.vscode`: VSCode configuration
- `api`: ProcessingAPI and like batcher implementation. It can be build and run as a single service.
 - `data-seed`: Contains database and cache seeder files to add initial to databases
- `cache-service`: Caching service that sync data from Redis with PostgreSQL
- `doc`: Images for README.md
- `frontend`: Frontend service implementation
- `nginx`: Nginx for load-balancing
 - `nginx.conf`: Configuration file used for custom nginx image
- `.env`: Env variables that include connection strings to PostgreSQL, Redis and JWT token secret. See **Env configuration** for setup.
- `api_requests.postman_collection.json`: Postman collection to test all available API and cache requests

- `deploy-start-script.sh`: Collection of commands to setup whole infrastructure on new virtual machine. Tested on ubuntu 20.04 with python3
- `docker-compose.yaml`: Configuration of all containers
- `Makefile`: Set of commands to make project interaction easier
- `README.md`: This file

System description

System uses PostgreSQL and Redis databases. Frontend web is accessed via nginx load balancer that can balance between multiple frontend instances.

API service contains migration files created by sqlalchemy necessary to create tables in SQL database. This can be executed using `make init-db`.

After initialization database needs to have initial data (categories, users, posts). This can be executed using `make seed`, that also insert data in the caching service.

Like batching service synchronize likes from redis to postgresql every 5 minutes and it is a part of the API service.

Project also contains postman collection to test API requests using postman.

HTML coverage can be generated for `api` and `cache-service` by running `pytest` inside these directories.

User experience description

Users can browse available posts in categories or by choosing an individual user to see only their posts. There is always displayed maximum 10 posts, to load next page, user can click appropriate button to load more.

Each post can be liked by unregistered/registered user which only increment like count inside redis/postgres database. By default, frontend contains variable `CACHE_ENABLED` in `src/env.js` that is by default set on `True`. This indicates that like requests (get likes, send like) are requested to caching service that uses redis.

Users can login and register but it's not mandatory to use website. They can also browse their profiles but if user is logged in and is browsing their own profile, they will see additional section for user management. This section includes change of email, username, password and option to remove whole account.

When user is logged in or just registered, they will receive `httpOnly` cookie with JWT token for further authentication. This token has 5 days validity. If user access the web within this period, the token will be automatically re-generated by API service.

Website mapping:

- `http://{url}/` - Home webpage. User is asked to choose category or login/register if they are not logged in.
- `http://{url}/category/{id}` - Category page. Displays available posts in category created by users sorted by time in descending order.
- `http://{url}/user/{username}` - User's posts. Similar to category but posts are filtered to posts that are owned by the user.

`url` refers to webpage hostname (e.g., `localhost`, `129.114.26.177`).

Running project using services

- Prerequisites: `apt`, `python3`, `make`, `git`
- Run `deploy-start-script.sh`

Running project locally

- Prerequisites: `apt`, `python3`, `make`, `git`
- Install python dependencies: `pip install -r api/requirements.txt`
- Install `node.js` with version `>= 16`
- Create `.env` in root directory (see Env configuration)
- Install `node.js` dependencies for frontend:
 - `npm install -g yarn`
 - `npm install -g typescript`
 - `npm install -g serve`
 - `cd frontend && yarn install`
- Create database tables
 - `make init`
- Seed database and cache
 - `make seed`
- Build services
 - `make build`
- Run tests (optional)
 - `make test`
- Start services locally:
 - API: `cd api/src/assignment2api && python3 -m uvicorn main:app --port 8000`
 - Cache-service: `cd cache-service/src/assignment2cache && python3 -m uvicorn main:app --port 8001`
 - Frontend: `cd frontend && serve -s build`

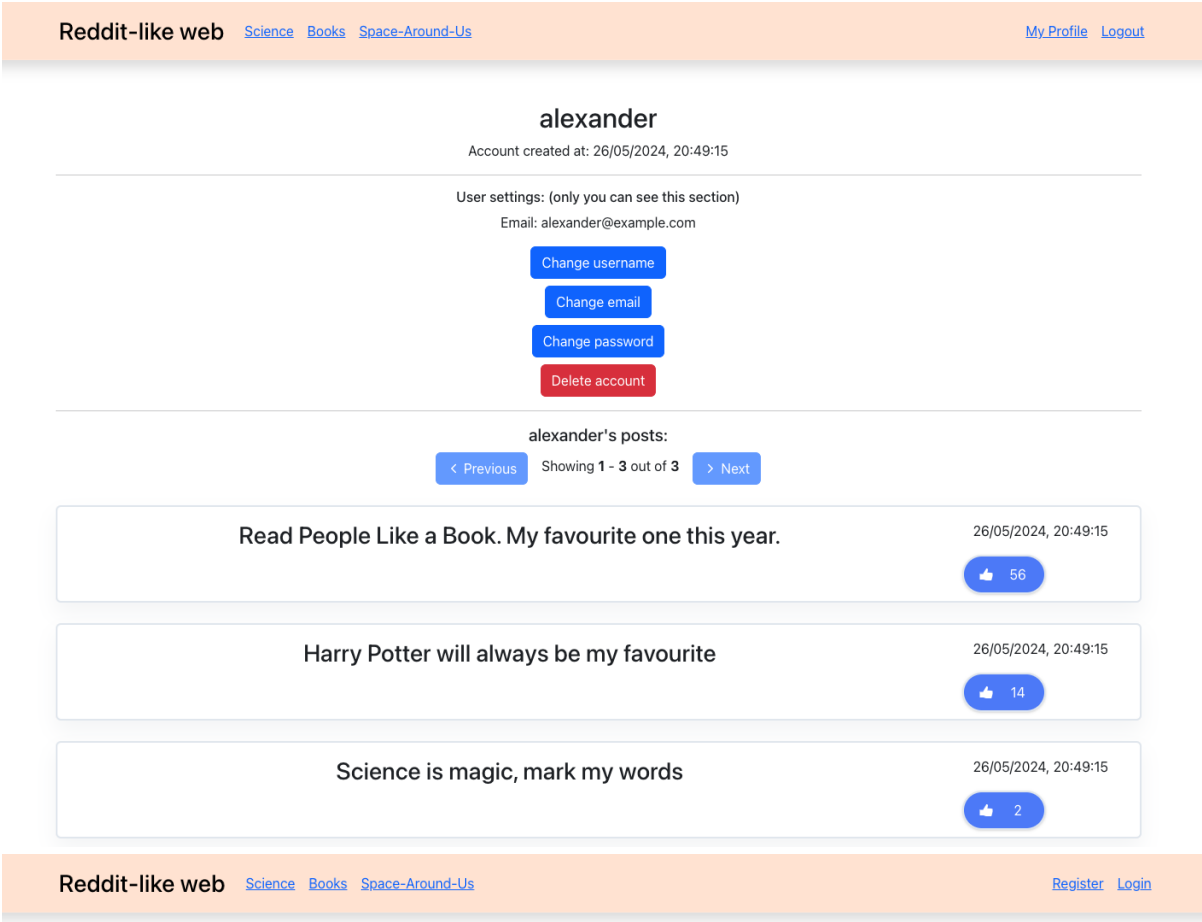
Env configuration

Before locally running services, `.env` file must be created in root directory. **This action is not required for containerized services.**

Env variables for local services:

- DB_URL=postgresql://postgres:postgres@localhost:5432/reddit: URL for connection to postgresQL
- JWT_SECRET=mysecretstring: Secret variable for JWT tokens
- REDIS_HOST=localhost: machine hostname where Redis is running

Project images



Register

Username

Username

Email

Email

Password

Password

Register

Notes

Part of the github action job fails because postgresql cannot be deployed. This container for pytest. However, it can be tested manually on local machine with generation of coverage reports. Build-images job pass successfully.