SnapSense

Project 8: Monitoring Diabetic Foot Ulcers

Developer Setup Guide

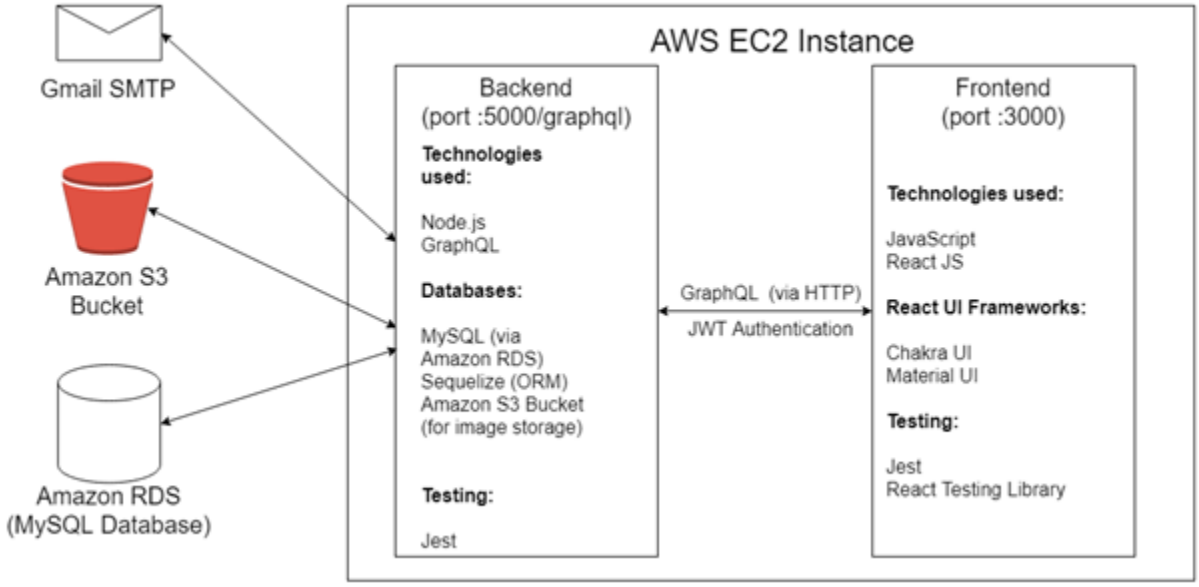
5CCS2SEG Software Engineering Group Project Major Group Project 2020/2021

Ali Zaini, Mohammad Albinhassan, Ayan Ahmad, Sonia Baileys, Mahsum Kocabey, Era Mullahasani Dula, Soyoung Park, Benedict Udall and Alisher Zhaken

Developed for the Co-founders of SnapSense AI:Shreya Chawla, Qingwei (Olivia) Zhang, Lorenzo Bernasconi

Contents

Brief Application Overview



Installations & Setup

There are many tools, languages, frameworks and services used in this application and it is suggested you install and setup before beginning development of this project.

- **Visual Studio Code** is the suggested IDE. You can download it here
    - There are some useful extensions you should install including Bracket Pair Colorizer, GraphQL, ESLint, npm, Prettier and others.
- **Nodejs** is required for this project. You can download it here
- **Git** is required for this project. You can download it here
- **MySQL** is required for this project
    - Ubuntu Installation/Setup Guide
    - MacOS Installation/Setup Guide
    - Windows 10 Installation/Setup Guide
    - You can also find many more guides and videos online
- A **Gmail** account, with access for less secure apps turned on, is required for sending transaction/scheduled emails to users. Guide
- **AWS** is required for deploying this project to enable features such as: Image Uploading and Database Access
    - You'll need an AWS account with up-to-date billing information (you shouldn't have to pay as you can use free tier for up to 12 months). Guide
    - You'll need to create an **S3 Bucket** with public read privileges. This is where images are uploaded and accessed. Guide
    - You'll need an **EC2** instance. This is essentially a remote server running Ubuntu. Guide
        - The EC2 instance is what is used to deploy/host the project. This could be deployed different, but this guide covers a method to deploy it on AWS as that is what the client has requested
    - You'll want to install the following packages:
        - npm - Guide
        - nginx - Guide
        - git - Guide
        - pm2 - Guide
    - You'll need an **RDS Database Cluster** on AWS with a Public Endpoint. This is basically the MySQL database that will be used in the deployed/production version. Guide
- (Optional) There are a few browser extensions that make frontend development easier. We suggest the following (you can find them for other browsers too):
    - Apollo Client Devtools Chrome
    - React Developer Tools Chrome
- (Optional) Setup and use WSL for development. This lets you use Linux/Ubuntu and develop on Windows
    - Windows WSL Installation Guide
    - VSCode Guide

Once all the required tools and services are installed and set up, you can begin with configuring the project on your machine, which is covered in the next section.

Configuration

IMPORTANT: Never commit and push the .env or config.json files. These should only exist locally.

The project README goes over the configurations required, but is highlighted here again.
You will need TWO '.env' files. One in the root of the project and another inside the frontend folder. You will also need ONE file in the backend called `config.json` which should be inside a folder called `config`.

The `.env` inside the frontend should simply look like this:

```
REACT_APP_FRONTEND_URL_PREFIX = http://localhost:3000
REACT_APP_BACKEND_URL = http://localhost:5000/graphql
```

The `.env` inside the root of the project should look something like the following. You'll need to change the values based on your AWS account. You can find out how to get these details here.

```
DATABASE_CONNECTION = 'AWS RDS IP address e.g. 123.123.123.123'
DATABASE_USERNAME = `AWS RDS MySQL username'
DATABASE_PASSWORD = 'AWS RDS MySQL password'
DATABASE_SCHEMA = 'snapsense or whatever you called it'

AWS_ACCESS_KEY_ID = 'e.g. AKIAIOSFODNN7EXAMPLE'
AWS_SECRET_ACCESS_KEY = `e.g. wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
AWS_REGION = 'e.g. ap-south-1'
AWS_S3_BUCKET = 'e.g. snapsensebucket'

MAIL_AUTH_EMAIL = 'e.g. snapsense@gmail.com'
MAIL_AUTH_PW = 'e.g. gmailpassword123'

ACCESS_TOKEN_SECRET_KEY = 'e.g. random stuff fsdlkfsdlfjsd'

FRONTEND_URL = http://localhost:3000
BACKEND_URL = http://localhost:5000/graphql
```

The config file should be at /backend/config/config.json and needs the following information. It contains the MySQL details for the development, test and production databases. Replace the details with your own MySQL credentials.

```
{
    "development": {
      "username": "dev",
      "password": "dev",
      "database": "database_development",
      "host": "127.0.0.1",
      "dialect": "mysql"
    },
    "test": {
      "username": "test",
      "password": "test_password",
      "database": "database_test",
      "host": "127.0.0.1",
      "dialect": "mysql"
    },
    "production": {
      "username": "dev_user",
      "password": "dev_user",
      "database": "snapsense",
      "host": "AWS RDS IP address e.g. 123.123.123.123",
      "dialect": "mysql"
    }
  }
```

Setup/Test/Run the Application

Finally, the fun part. To make it as easy as possible, there are basically 3 simple commands you can run to try out the application:

- From the root, run `npm run setup` to set up the frontend/backend (this will install all the relevant modules and set up the databases and seed them.)
- (Optional) From the root, run `npm run test` to run the tests for the application (this will run the backend and then the frontend tests)
- From the root, run `npm run start` to start the frontend and backend.
    - It is suggested you start them separately, but this works if you're in a hurry.
    - You can access the frontend from `http://localhost:3000` and the backend from `http://localhost:5000/graphql`

It is however suggested when developing that you run the frontend and backend in separate terminal windows. You can find more about the commands available to you by opening the `package.json` files in the frontend, backend and root of the project.

Default Access Credentials

We seed the database with some basic accounts and data so you can try out the application more easily. Here are the details:

- **Patient** - Email: patient1@gmail.com / patient2@gmail.com
- **Doctor** - Email: doctor1@nhs.net / doctor2@nhs.net
- **Admin** - Email: admin1@gmail.com / admin2@gmail.com
- **Superadmin** - Email: snapsense@gmail.com

The password for each of these accounts is simply ***Password123***

Deploying

Some prerequisites before deploying the project:

- An Amazon AWS Account with up-to-date billing information.
- An S3 Bucket with Public Read privileges. Support
- An EC2 Instance (Instructions for Ubuntu Instance) Support
- An RDS Database cluster with public endpoint access. Support
- All services must be in the same region
- EC2 with `PM2`, `NGINX`, `NODE`, `NPM`, `GIT` installed. Support

Deployment Procedure

1. Setup RDS credentials
2. Store Database credentials and API endpoint in `config/config.json`
3. Make sure the root `.env` file contains the correct **AWS** configuration details
4. Login to the EC2 and `git clone` the repo `snapsense`
5. `cd` into the folder (ex: snapsense)
6. run `cd frontend && npm i`
7. run `npm run build`
8. run `cd ../ && cd backend && npm i`
9. run `cd /etc/nginx/sites-available`
10. run `sudo nano default`

11. Paste the following nginx configurations in the file:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    client_max_body_size 100M;

    root /home/ubuntu/snapsense_production;

    index index.html index.htm index.nginx-debian.html;

    server_name _;
    location / {
        try_files $uri /index.html;
    }

    location /graphql {
        client_max_body_size 100M;
        proxy_pass http://localhost:5000/graphql/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

12. Save the file
13. run `cd /home/ubuntu/`
14. run `mkdir snapsense_production`
15. run `cd /home/ubuntu/snapsense/frontend`
16. run `cp -r build/* /home/ubuntu/snapsense_production/`
17. run `git stash`
18. run `systemctl sudo restart nginx`
19. run `cd /home/ubuntu/snapsense/backend/`
20. run `pm2 start server.js --name server`
21. The App should now be running on the EC2 public URL

(Optional) Continuous Deployment / Integration

You could even go further and set up CI/CD for your application. For continuous deployment we are using GitHub actions to connect to the AWS EC2 instance on any changes to our master branch and then we perform a series of commands on the server itself.

In order to setup CD from GitHub:

- Add the AWS_REGION, ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY , INSTANCE_ID secrets to GitHub Secrets for your repository.
- On your server, be sure to have setup ssh keys for your git account.
- Go to the Actions tab and set up a new Workflow
- Name the new file `deploy.yml` and paste the following:

```
on:
  push:
    branches: [master]

jobs:
  start:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: AWS SSM Send-Command
        uses: peterkimzz/aws-ssm-send-command@master
        id: ssm
        with:
          aws-region: ${{ secrets.AWS_REGION }}
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          instance-ids: ${{ secrets.INSTANCE_ID }}

          working-directory: /home/ubuntu/snapsense
          command: /bin/sh ./deploy.sh
          comment: Done Gracefully

      # Catch SSM outputs
      - name: Get the outputs
        run: echo "The Command id is ${{ steps.ssm.outputs.command-id }}"
```

- Add a new file named `deploy.sh` in the root of the repository and add the following commands:

```
#!/bin/sh
eval ssh-agent -s
ssh-add /home/ubuntu/keys/id_snap_gmail
sudo systemctl restart nginx
pm2 stop server
git pull
cd backend
npm install
cd ../ && cd frontend && npm I
npm run build
rm -rf /home/ubuntu/snapsense_production
mkdir /home/ubuntu/snapsense_production
cp -r build/* /home/ubuntu/snapsense_production/
git stash
pm2 restart server
```

- Continuous integration should now be set up.