Build and Deployment Instruction for Project Team-Mejia

Backend:

The Dockerfile used for building backend Prisma application is defined within backend folder. For database, the docker image for database are directly extended based on the original version postgres:13, which was described in the docker-compose.yml file along with all the relevant configuration for environment set-up and connections between backend and database.

Building and Running Docker image with docker-compose:

```bash
# Navigate into backend folder
$ cd backend

# Building the docker images for both prisma application and database
docker-compose build

# Running docker images
docker-compose up

# Checking the containers are running
docker ps

# Stop the docker process
docker-compose down
```

Building and Running docker image with docker:

```bash
# Build docker images in current directory
docker build -t getting-started .

# Run docker image in background
docker run -d -p 80:80 getting-started
# -d for detach, run container in background
# -p pushlish-list, publish a container's port(s) to the host

docker stop [container_id …] # stop running
docker rm  [container_id … ]  # Remove one or more containers
docker rmi [image_id]            # Remove one or more container images
```

Deployment by running docker containers with EC2 virtual machines:

- First, have some understanding about EC2 instance, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html?icmpid=docs_ec2_console
- Instantiate an EC2 ubuntu instance in AWS, by following this tutorial, https://medium.com/@KerrySheldon/ec2-exercise-1-1-host-a-static-webpage-9732b91c78ef
- Login to the EC2 instance with your username and DNS (or public IP), e.g., `ssh -vvv -i id_rsa ubuntu@ec2-54-161-150-189.compute-1.amazonaws.com`
- Copy the backend.zip file into the EC2 instance, unzip it, run docker build, and docker run
- Open a browser with the given DNS for your EC2 instance, and you should see the application is running

Frontend:

Before building the frontend, you should run `npm install` inside the frontend folder to install all the necessary node modules.

We have the script for building and deploying the application inside package.json.

To build the application, simply run `npm run build`. This is equivalent to `gatsby build`, and gatsby will build the application and generate a folder called public that contains all the generated files for the frontend application.

To test out the built website before deployment, run `gatsby serve`. The website will be served at localhost:9000. You can check if there's any error or bug before shipping it to production.

To deploy the website, run `npm run deploy`, the deploy script will build the website and push the public folder to gh-pages branch of your repo for deployment. We used a module called gh-pages to achieve this. Everytime you run `npm run deploy`, the built product will be automatically pushed to the gh-pages branch. In your github repo, make sure you are serving the website using the gh-pages branch.

Note:

Currently, our frontend is served in one of our own repos because our project repo doesn't have github pages set up. In the end, we either need a designated repo or set up github pages in the project repo for serving our frontend.