Team 26's Back-end Go Server

This was initially started as a rewrite of the previous (node.js) back-end since I anticipated we would need to write some back-end code for some extra features. As expected, we did, and the back-end now holds the code for three parts:

- wellbeing visualisation, with Google Maps
- data/message passing between mobile users (i.e. for user wellbeing sharing and sending step goals)
- add friend page, to serve a deeplink

Deployment Guide & Usage

You need an installation of `go` (use your package manager, or otherwise). You can verify you have it with `go version`.

Follow these steps on the Linode server:

- Clone or pull the repo into your home directory.
- `cd` into `team26-nudgeme-backend` and `go build` in that project directory.
- Start with `sudo systemdctl start nudgeme` or restart the systemd service with `sudo systemctl restart nudgeme`, *if it is not the first time running.*

Of course, this can be done on any server, but *you'd need to set up your own* ***`nudgeme.service`*** *file and SQL database connection.* An example `nudgeme.service` file is given below. For the SQL database, see the backend database schema. If you wish to change the server which hosts the SQL database, please modify the `ADDRESS` variable in `main.go`.

You can modify the nudgeme.service just like any systemd service, e.g. if you want to change where it searches for the binary. The service config file is located in `/lib/systemd/system/nudgeme.service`.

Current domain that links to the server: `https://comp0016.cyberchris.xyz/`. This is retrieved from the environment variable in the `nudgeme.service` file.

Example `nudgeme.service` file

This is the service file currently in use on the Linode server, except in the actual file I use the actual password instead of '$PASSWORD' as below.

Also note the domain name variable, this would need to be changed with a new domain name.

```
[Unit]
Description=back-end for NudgeMe
Documentation=https://github.com/thevirtuoso1973/team26-nudgeme-backend
```

```
After=network.target

[Service]
WorkingDirectory=/home/ct/team26-nudgeme-backend
Type=simple
User=root
ExecStart=/home/ct/team26-nudgeme-backend/nudgeme
Restart=on-failure
Environment="SQL_PASSWORD=$PASSWORD"
Environment="DOMAIN_NAME=comp0016.cyberchris.xyz"

[Install]
WantedBy=multi-user.target
```

API Docs

See the `WellbeingRecord` struct in `models.go` for the latest. Fields that are marked `omitempty` are intended to be optional in the future, e.g. weeklySteps.

Wellbeing Data & Steps for Map

Endpoint: *. . . /add-wellbeing-record*

- postCode: string e.g. TW6
- wellbeingScore: integer
- weeklySteps: integer
- errorRate: integer, this is abs(score-userScore), where score is our estimate of their score
- supportCode: String
- date_sent: string, 'yyyy-MM-dd'

User Wellbeing Sharing

. . . /user check if user exists.

Request example:

```
{
"identifier": "abc1337",
"password": "battery horse staple"
}
```

Response example:

```
{
"success": true,
"exists": true,
}
```

. . . /user/new add new user

Request example:

```
{
"identifier": "abc1337",
"password": "battery horse staple"
}
```

Response example:

```
{
"success": true,
}
```

. . . /user/message get unread 'messages' for user

Request example:

```
{
"identifier": "abc1337",
"password": "battery horse staple"
}
```

Response example:

```
[
{"identifier_from":"blahblah", "data":"123"},
{"identifier_from":"blahblah2", "data":"1234"},
...
]
```

. . . /user/message/new send message/data to a user

Request example:

```
{
"identifier_from": "abc1337",
"password": "battery horse staple",
"identifier_to": "bobby420",
"data": "UXVvdGggdGhlIFJhdmVuIOKAnE5ldmVybW9yZS7igJ0="
}
```

Response example:

```
{
"success": true,
}
```

P2P nudging

Nothing special on the back-end, uses the same structure as wellbeing sharing. It's up to the client to define the different spec.

Only difference is that it is using a different table.

. . . /user/nudge See . . . /user/message section.

. . . /user/nudge/new See . . . /user/message/new section.

UML Diagram of Database Interface

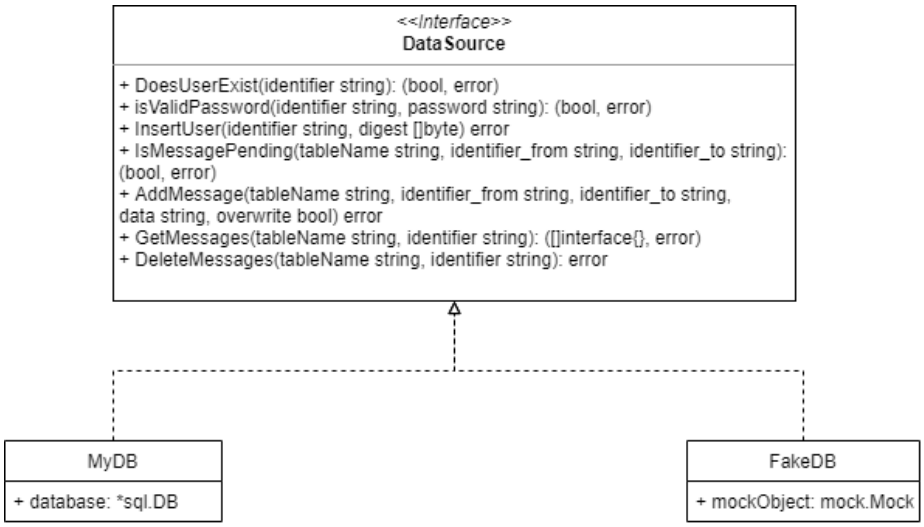

Figure 1: image

ER Diagram/Schema of Back-end Database

Performance

We used GTmetrix to test the performance of the visualisation.

Figure 2: image

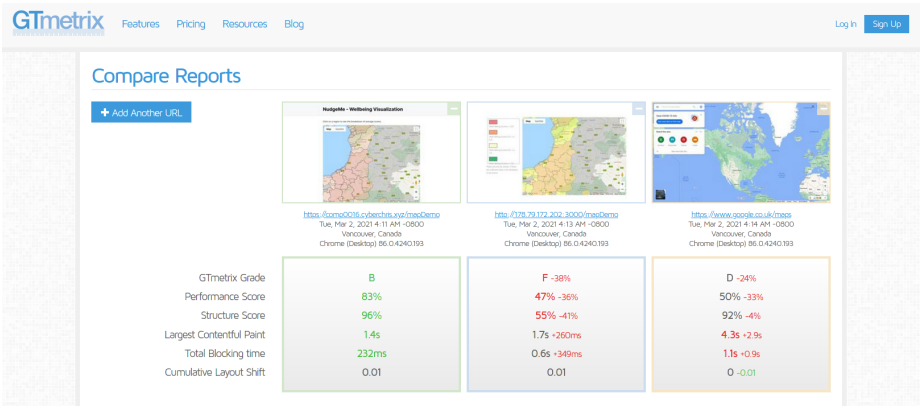

Figure 3: image