Installing Asura Tournament Manager

This is guide describes the process of installing Asura Tournament Manager on your own server. At the time of writing, you can reach our production server at https://asura.feal.no/.

Before installing

Before installing Asura Tournament Manager, you need the following:

- A suitable server to host the software. There are many ways to deploy the server, for example
    - Cloud hosting provider
    - Virtual Machine
    - Dedicated server
- A domain name to reach the server over the internet
- A MySQL database to store persistent data
    - This can be either on the same server as the application, or on a separate database server
- A Google account to sign in to the application
- A Google API key for authenticating users
    - Acquiring this is both simple and free, and is described in the Google API documentation
- [Recommended] A reverse proxy. Software proxies like Nginx will greatly improve the performance, reliability and security of the application.
    - The tournament manager does not handle SSL/HTTPS by itself
    - The server is susceptible to DoS attacks
    - Although the application logs requests to the console, they are not saved on disk
    - Setting up a reverse proxy is not required, but it is *strongly* recommended. Configuration is out of scope of this manual, but it can

be found in the Nginx documentation

Download the source

Before you can run the tournament system, you must download the source code. At the time of writing, it is available on NTNU-IDI Gitlab.

Download the entire repository to your machine:

```
git clone git@gitlab.stud.idi.ntnu.no:felixalb/dcst1008-2022-group1
```

Configure the client

The client is a web application that runs in the clients browser. After this step is finished, the entire build consists of a single html file, a few javascript files and some static assets.

Install dependencies

```
cd src/client
npm install
```

This step fetches all the required libraries into the node_modules folder.

Configure environment variables

```
cp dotenv-template .env
nano .env
```

Edit the `.env`-file containing a few simple settings for the client, using nano or any other editor. Only the two first lines have to be changed, and just requires the server URL.

Build the client

```
npm run build
```

This step will build the client, optimize it and package it into just a few files. The resulting files are placed in the `build` folder. You are now done with the client and everything else, like serving the build files, are handled by the server.

Configure the server

The server is a node application that both serves the client files and handles the database and tournament logic. The installation process is similar to the client.

Install dependencies

```
cd src/server
npm install
```

Configure environment variables

```
cp dotenv-template .env
nano .env
```

Edit the `.env`-file containing all the server options. This includes:

- The server URL
- The MySQL database credentials
- The Google API credentials
- A cookie secret. This can be any random string of text, as long as it's secret.
- The remaining options should be left as default

Initialize the database

To create the required tables, start with an empty database.

This can be done with a mysql cli utility:

```
mysql -h mysql.stud.ntnu.no -u felixalb_sysut -p felixalb_asura < .
```

Or by pasting the content of `src/server/management/initDB.sql` into another MySQL manager like PHPMyAdmin.

After initializing the empty tables, you will have to insert your own email address. Insert email=youraddress and isManager=1 into the `users` table to register your user. All other users can be added in the graphical user interface.

Check the list of requirements

Read the list of preparations above to see all the requirements. When you have configured everything, you are ready to start the server.

Start the server

When all the steps above have been completed, you can start the server with:

In `src/server`:

```
npm start
```

You should see a message like this:

```
> tournament-server@1.0.0 start
> node index.js

Listening on port 3000
```

Access logs, error messages and other useful messages will be printed to the screen. The database connection is handled automatically, and will reconnect if the connection is lost.

The server can be stopped by pressing Ctrl+C, as nodejs will handle terminate the process cleanly, freeing its resources.