M7011E Project

Martin Bråthen - marbrt-6@student.ltu.se
Anton Dahlin - anodah-6@student.ltu.se

https://github.com/MartinBrathen/D7011M_assignment

January 18, 2020

1 Introduction

The task of this assignment was to create a web system for modelling the electricity production and consumption of a small scale power grid, to which a coal power plant and a number of households are connected. These households have the ability to produce power of their own using wind turbines. The household will be referred to as "prosumers" from now on, since the both produce and consume power.

A prosumer will use the electricity it produced to satisfy its consumption, it will then sell the remaining electricity to the grid. In the case of underproduction, it will consume all its produced power, and then buy from the grid to meet its needs.

In the case that all prosumers as a whole are consuming more than then they collectively produce, a power outage would occur. To prevent this from happening, large batteries have been installed at each prosumer. Instead of selling to the market in the case of overproduction, the prosumer can decide to store some or all of it's excess power in its battery. This is also the case for under production, it can chose how much to buy from the grid and how much to consume from the battery.

An outage can still occur if the prosumers have depleted their buffers, and are consuming more than they are producing. In such a case, the power plant has to start producing an appropriate amount of electricity. To control the power plant, a manager system was made, that has the ability to control the power output of the plant, control what percentage of the output that goes to its own buffer, set the price that prosumers buy and sell electricity for, which in turn influences the prosumers behavior.

The manager's goal is to avoid power outages, and also to use the power plant as little as possible, in an effort to reduce carbon emissions.

To help evaluate how this system would work, a simulator was made to model the production and consumption of the prosumers.

2 Design Choices

The simulator is deterministic and does not use a database. This is to make it as fast as possible. Both the weather and the consumption is determined by time which would make it possible to generate historical graphs without the use of a database. The simulator also uses sinus waves and noisemaps to make the values linear.

The server has a state which keeps track and updates the prosumers and the powerplant. If this system would have been in the real world then the buffers would have the information about how full they are, similarly the electricity on the grid would have been measured directly of the grid. Since this is a simulation we have to simulate all this data, this takes place in a loop. The faster this loop runs the more realistic the state becomes but takes more power from the server.

To update the content of the prosumer dashboard and the powerplant dashboard pages two methods are used, GET requests and sockets with events.

Values that we expect to continuously change e.g. weather and consumption are requested with GET requests. This also makes it possible for the client side to chose how often it should update its values.

Values that change on some event e.g. when a prosumer logs in, are transmitted over a socket. Only sending the data when it is updated minimises network traffic and computing power needed.

Making a computation on the client is preferred over the server. This allows the server to handle more users. A simple example is when the client wants the *net production*, the consumption and production is sufficient for the client to calculate net production for themselves. This is preferred over calculating the requested net production on the server.

3 Scalability Analysis

Since the server has a state that keeps track of prosumers' consumption, production, and power stored in their battery etc. the storage space, and processing time grows quite fast with an increasing number of prosumers.

An approach that would scale very well would be to keep the state stored in each client, then the server would only have to process the requests, but not have to keep an internal state. This approach would probably not lend itself very well to this application however, since it requires the clients to always be online in order to be part of the system.

4 Security Analysis

We are using Let's Encrypt to encrypt the traffic coming to and from out server, but we gave the user the ability to use regular HTTP instead if they would want.

passport.js was used to handle user authentication. It came with the benefit of automatically salting and hashing passwords in our MongoDB database.

To protect against denial of service attacks we used express-rate-limit, which blocks requests from a sender if they sent too many requests in a short time period.

We have not done enough testing to determine if the system is vulnerable to injection attacks.

5 Advanced Features

- In the simulator, the windspeed is determined by latitude and longitude. The closer two coordinates are to each other the more similar the windspeed will be.

- A simple page was added where the user can update its profile. The user can also delete its profile here, this removes all data related to the user from the system and logs the user out.
- It is possible to be logged in to the same account on multiple devices at the same time.

6 Discussion

6.1 Challenges

Working with so many systems at the same time, client-side, server-side, database, imported npm modules, etc., sometimes made it hard to find errors. It was important to isolate and debug individual parts when debugging.

6.2 Future Work

For the system to be ready for the market one would want to hire a UX designer and a graphics designer.

If the simulator is to be used for actual research and business decisions then the simulator should be improved to give results representing the real world.

It would be interesting for the prosumer to be able to view historical data, not important but a nice feature.

Some elements are not very friendly on smaller screens e.g. the table of users in the powerplant dashboard.

More safety measures could be implemented e.g. two-factor authentication and blocking of malicious IP's.

7 References

The server is built with nodejs. The project is dependent on a number of npm modules, these modules are in turn dependent on others, below is a list of the major modules with a short description.

- express is used for routing.
- ejs is the used HTML template engine.
- mongoose is used for the database.
- passport is for user authentication.
- socket.io for communication over sockets.

8 Appendix

8.1 Time Analysis

8.1.1 Anton

I spent 520 minutes on the initial simulator, most of this time was focused on the weather.

980 minutes was spent on the prosumer part where 340 min was on the dashboard, 220 on registration and login page and 310 was spent on the edit profile page.

1300 minutes was spent on the powerplant page and its functionality on the server.

In each of these three parts some time was always spent modifying old code.

I have not touched the live aws server which is something i regret because it is something i want to be comfortable working with but isn't.

8.1.2 Martin

480 minutes was spent on server related things, which was fun to learn. Most of the time spent was due to silly mistakes. This involved setting up AWS server, installion and configuration of node.js, Pm2, Nginx, MongoDB and Let's Encrypt!.

I have logged 1800 minutes on development tasks, but I did not do a good job with the time report, it is missing quite a few entries, and the ones that do exist are not very well detailed.

A lot of time was spent on fixing previous work, that was found to not be good enough to complete later tasks. We should have spent more time planning our work, which would have improved efficiency.

8.2 Work Distribution

Overall the work has been split evenly and the communication has been good. Both of us has worked at least a little with almost all parts of the project.

8.3 Grade

Not many advanced features have been implemented. The systems we have implemented have been thought out and well implemented. All of the basic functionality has been implemented. The application works as intended and does not suffer from performance issues.

We think we deserve at least a pass.

8.4 Github

https://github.com/MartinBrathen/D7011M_assignment
installation and deployment instructions in github README.