M7011E, Design of Dynamic Web Systems Report

Martin Eklund*
Kevin Träff†

Teacher comments:

January 18, 2020

*ekumar-6@student.ltu.se
†kevtrf-6@student.ltu.se

Contents

1 Introduction

In this course we were given an assignment to develop web system for controlling the production and consumption of electricity on a smaller scaled market. The limits were that clientside and serverside code should be written in javascript and that Node.js were to be used as webserver. Given free choice of database we considered learning a new DB but ended up using MySql as we thought this would be the best fit for our system.
A group decision was made that we would aim to understand what we build and that we reasonably would not be able to develop much of the advanced features in the given time-frame.
Our sources of information (the ones we picked most information and influence from)can be found in the References (7) further down in the document.

2 Design choices

One of the first and biggest impact choices made was that we were going to use a REST API. The motivation for choosing RESTful API is that the functionality REST gives is enough to solve our problems and as we did not have any experience with APIs we found information about REST more easily (Information about REST was given in 'Related literature' in canvas).

The choice of using Express was easy as we were required to use Node.Js and Express is a great way of handling routes and requests to and from from db/client.

Not all choices are big but the decision to use Nodemon to watch the server files and restart the web-server when changes were saved was a real time saver.

The choice of MySql database was made because we wanted to focus on learning other things and we both are familiar with MySql. NoSql could have been a good choice to as it works well with the Restful API and the information stored.

3 Scalability analysis

The choice of going with MySql could if the project scales to where there is a lot of data and even new sorts of data be a bottleneck. If up-scaling was on the horizon a NoSql database would make more sense as it is more easily scalable and the DB design is more flexible making remodeling for new data easier.

We have not implemented websockets and websockets could prove useful as data might be required to be delivered fast to the user and websockets (socket.io or websockets) solve this. Websockets would improve on latency as well and could fix problems that would arise with many simultaneous users such as long wait time for certain information. especially if a client is on a low speed network connection.

Using caching more would make the system more scalable as it would improve latency, server load problems and free up network capacity. Our Get requests are automatically cached but not our Post requests, and as we have more Post than Get requests further caching would be beneficial.

4 Security analysis

The first and main step towards a secure website is that we hash our passwords using Bcrypt. We also salt the passwords in order to make it more secure. Salting adds a random bits to the password before hashing. This makes it so that even if two users have the same password, the hash will not look identical. If we would not salt our hashes there is a big risk that if someone gained access to a hashed password (not salted) the original passwords could be compromised using dictionary tables.

In our tests we have not succeeded with SQL injection but it would in theory be possible to do. We don't take any additional steps to prevent it beyond that the MySql-Node library we use take steps to prevent injection.

Our authentication has security vulnerabilities such as

- Allowing weak passwords and generic ones (password123 etc)
- Could be vulnerable to brute force attacks
- No multi-factor authentication

Sessions are handled in express and randomizes session ID upon new login. Further investigation if session and tokens are secure would be recommended to make sure there are no, but we not found any obvious vulnerabilities linked to sessions and tokens in our testing/analyze.

5 Advanced features

Our goal was to at first reach the basic requirements. In the end we did not have time to implement many advanced features. We did implement a fancy background video for the login page. The web pages are somewhat responsive but we have not put any major focus into it. We implemented sliders for controlling the market/battery ratios. There are also some advanced features that could be implemented easily as we already have the required features for them. Among those possible features are having a page to update credentials for the prosumers as we already have it as a feature for the manager. We could also easily implement breakdowns and downtime on the prosumers as we already have the ability to shut down and enable production for all prosumers but it is only implemented for the manager. Historical data could also be implemented as we have a timestamp for every simulator tick, it's only a question of storing it in a seperate table and displaying it to the users.

6 Challenges

The choice of using REST API gave us one major question/problem related to criterias given in one on the assignments. How would the system know when someone was logged in? As REST is stateless and information is given if the user asks and has the right token for that specific information. This happens every time a user wants information through the API and therefore we had to come up with a solution that in theory made it possible to see users that are online. We handled this through Express Sessions, which might have

made our system not being stateless and therefor making our API RESTful like.

We also struggled greatly with CORS-policys as they broke our session handling and without CORS our ajax requests could not be sent without running our web browser in disabled web security mode.

7 Future work

Future work should initially be on further security and more tests on possible security flaws. As the advanced features was not implemented these could also be seen as future work.

Two features from the basic requirements was not met, user being able to upload a picture and to see which users are online. We have made the ground work for solving the problem with seeing online users but did not get it to work in practice.

References

[1] Academind's youtube videos about Rest API, Node.js and express. Available at: https://www.youtube.com/channel/UCSJbGtTlrDamitDGPUV9-w (Accessed: throughout the course).

[2] W3schools guides Available at: https://www.w3schools.com/ (Accessed: throughout the course).

[3] RESTful API, Available at: https://restapitutorial.com/ Accessed: throughout the course).

[4] Postman, available at: https://www.guru99.com/postman-tutorial.html Accessed: throughout the course).

[5] Node.js Package Manager(Normal distribution). Available at: https://www.npmjs.com/package/distributions-normal Accessed: throughout the course).

[6] Guru99's guide in Node.js, Available at: https://www.guru99.com/node-js-tutorial.html Accessed: throughout the course).

8 APPENDIX

8.1 Martin

Looking at my time report i spent most of my time in the beginning on the API and setting up the project. The way i worked was that i spent a lot of time researching, learning and understanding how to solve our problems ahead. I might have misunderstood the time report as this time is not included and only active coding/working on the project was written in the time report. Me and Kevin divided the API and simulator in the beginning and later on we picked up what was next on the agenda and worked on that. Kevin and I put in similar work over the Christmas break and this was a big waste on developing time, we chose to go with Kevin's work as he had progressed further and it worked better. Although unnecessary work was spent the positive part was that both learned from it.

Active work is roughly about 60 hours and the double is easily put on learning/testing/laborating since this does not come easy for me. I would approximate that i have put 120-140 hours in to this course and I'm happy with the amount i have learned but could have put more time in coding. The time report has been a great tool in order to keep track of where you actually put effective time. It was also interesting how different me and Kevin learn and work where i like to sit down and learn then put in active work and Kevin does them both simultaneously.

If I would grade our project i would give it a grade: 3,5. We have missed some parts but learned so much and proved both in our work and meetings etc that we understand what we build.

8.2 Kevin

The project started out a bit slow in the beginning where I should have tried to put in more time than I did. For the majority of the project I was creating the simulator and had major problems with it. Due to how I at first designed it and also for a long time misunderstood some core concepts the simulator didn't progress much and a lot of time was wasted trying to make a bad design work. I ended up having to scrap large parts of it and

then rewrite it. Due to how long it took to get the simulator working the API and frontend only started seeing any real development in early january. Due to the approaching deadline many longs days were spent trying to rush together the project, while I made some great progress I'm not too satisfied with the quality of my work. One large struggle towards the end was handling CORS-policys.

The total amount of logged hours at the deadline was 151 hours of active work, which is a bit shy of the 200 hours that were supposed to be put in. Those 50 missing hours would have proven useful towards the end.

If I were to grade myself based on my effort put in and how much I've learned since the course started I would give myself a 4. I feel like I went from knowing almost nothing to essentially becoming a fullstack developer. If I had to grade myself by the quality of my work I would say it's a 3, as I stated earlier I'm not truly satisfied with the end result.