Project Report, M7011E: Design of Dynamic Web Systems

Isak Sundell, isasun-7@student.ltu.se
Enar Eriksson, enaeri-7@student.ltu.se

January 2021

1 Introduction

This project was made with the purpose to design a web system for the company Green Lean Electrics in the course about dynamic web systems, M7011E. The web system was made to fulfill three modules, simulator, prosumer and manager. Our solution to this is built with a simulator that contains the simulated world, for this simulator we have developed an API to make information available for front ends. The simulator also saves its data in a database to not lose information if its shut down. The last part of our system is a separate front end that only communicates with the API where customers can view information about their house and control some parameters and managers can moderate the simulated power plant and the users of the system.

2 Method

2.1 System Design

Our system consists of a react application for the front end and an express application for the back end. These are completely separate and the front end communicates with the back end through a restful API.

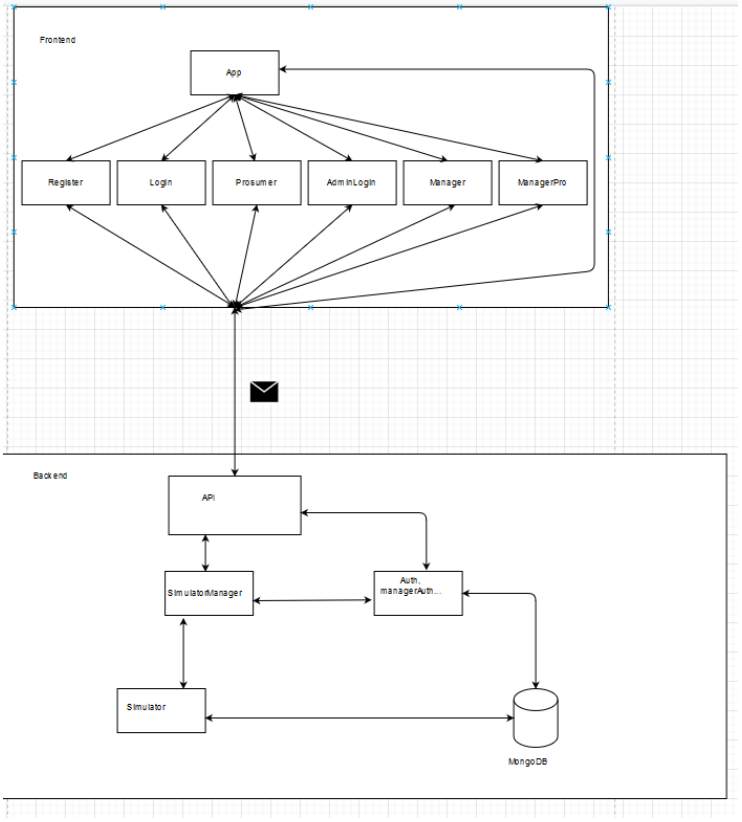

Figure 1: High level Overview of the system structure

2.2 design choices

We wanted to create a back end with an API for all of the required operations for which multiple front ends could be developed. Our front end application contains both the *Manager* and *Prosumer* systems.

2.2.1 Front end

For the Frontend we decided to use react.js, the main reason for this was that we had never used it but had always heard good stuff about it, after researching

some it also felt like it would be a good choice for this kind of system as its good at updating modules without refreshing the page.

For routing in the application we use browser-router, the app component is always rendered and is also where the router is placed, here all the components are loaded in and the router decide when to render them. In the app we also handle our special effect wind turbine, so its stays on in all the pages. We have three open routes and three protected, the three open routes are the *'/'*, *'/adminLogin'* and *'/reg'*. *'/'* is the login for the normal houses and prosumer houses where the *Frontlogin* module is rendered, both can also register them self at *'/reg'* where the *Reg* module is rendered. The *AdminLogin* module is rendered at *'/adminLogin'* and is only for the managers to login, they can not register. The protected routes first check that the props have been correctly pushed, and that only happens when someone is supposed to go trough that route. The protected routes are for the *'/admin'*, *'/ManagerPro'*, and the *'/pro'* page. On *'/pro'* the *Prosumer* module is rendered and the page is for normal houses and prosumer houses, but for non-prosumers the production is always zero so the battery ratios have no effect. On *'/admin'* the module *Manager* is rendered and this is the main page for the admin or manager. When the manager want to manage a user he will get directed to *'/ManagerPro'* where the *ManagerPro* moudle is rendered.
All connections from the front end to the backend are done with axios, an easy to use promise based http client. All data are transferred to and from the server with get, put, post and delete requests, except the pictures that are displayed from the source url.

2.2.2 Back end

The backend of our system is comprised on an express app. This application consists of our simulator, our API and some middleware. The simulator consists of three classes. The *Home* class represents a home of a prosumer or consumer. The *PowerPlant* class represents the coal power plant in our simulation.

The *Simulator* class holds all the PowerPlant and Home instances and is responsible for updating the state and reading from these. It will periodically update the state of the simulation including the state of all homes and the plant. This class contains a lot of get and set methods which are important for our API.

The *simulatorManager* module holds the instance of our Simulator and is accessed by our router. This module is responsible for fetching data and setting some values. The logic for these operations are handled mainly in our simulator.

Apart from that there is some additional middleware such as *auth* which authenticates users and managers, *managerAuth* which requires a manager to bypass. *credentials* and *getCredentials* updates and reads users from the database.

2.2.3 Database Schemas

Ther are 4 schemas in our mongo data base. *home* holds information about the homes in our simulation such as the buffer and battery ratios. *user* holds information about the registered users such as the name, username, password (hashed and salted) etc. *manager* is similar to *user* but it holds the information about a manager account. The reason for this was to make authentication easier. The only way to add a manager account is to do it manually on our VM. Finally *simData* holds information about the simulated environment such as the price, wind and power plant status and is updated periodically.

3 Results

3.1 Deployed Site

Figure 2: The header of the site, wind speed and a spinning wind turbine

This was simply something fun we did during the work, a wind turbine that spins at different speeds depending on the current wind speed, we also felt like the wind speed of the simulator could be open for everyone as its not sensitive information, the wind and the turbine also completes each other.

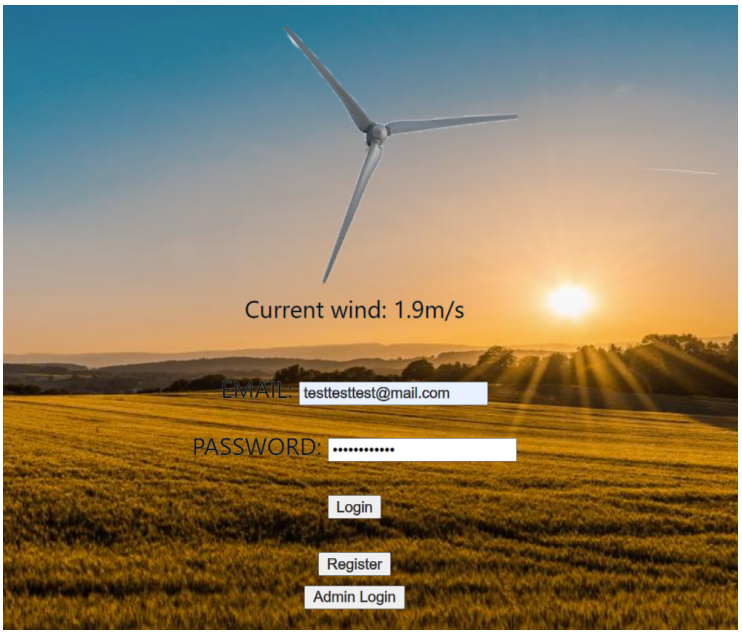

Figure 3: Login page for normal homes and prosumers

Figure 3 shows our login page, the page have 2 input parts, for the email and the password. The login button acts as a submit, and send a post request with the user to check if its correct, if the request is successful it will redirect you to the prosumer page.the Register button will redirect you to the register page and the admin login button to the adminlogin page.

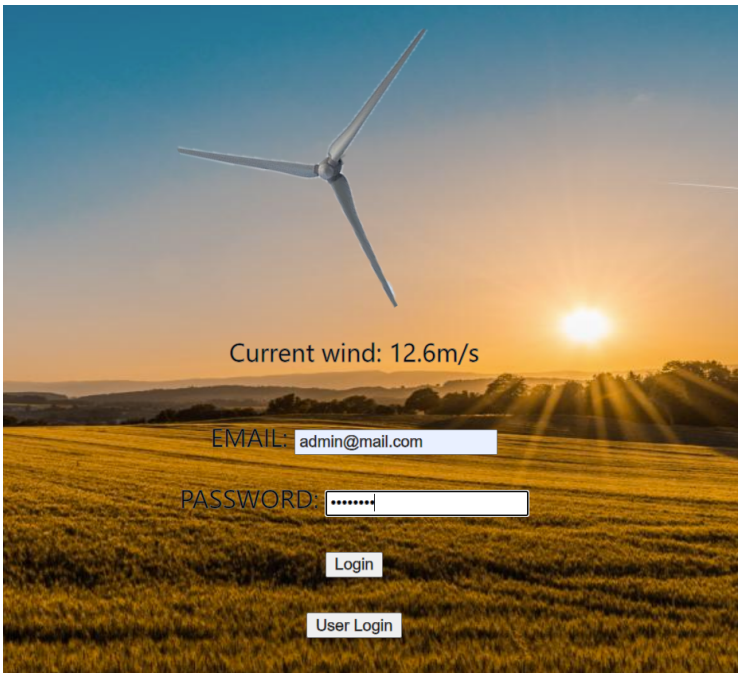

Figure 4: Login for the manager

Figure 4 shows the adminlogin page, this is the same as the normal login page, except the request will be sent to the adminlogin route to check if the user is an admin. The UserLogin button redirects to the normal login page.

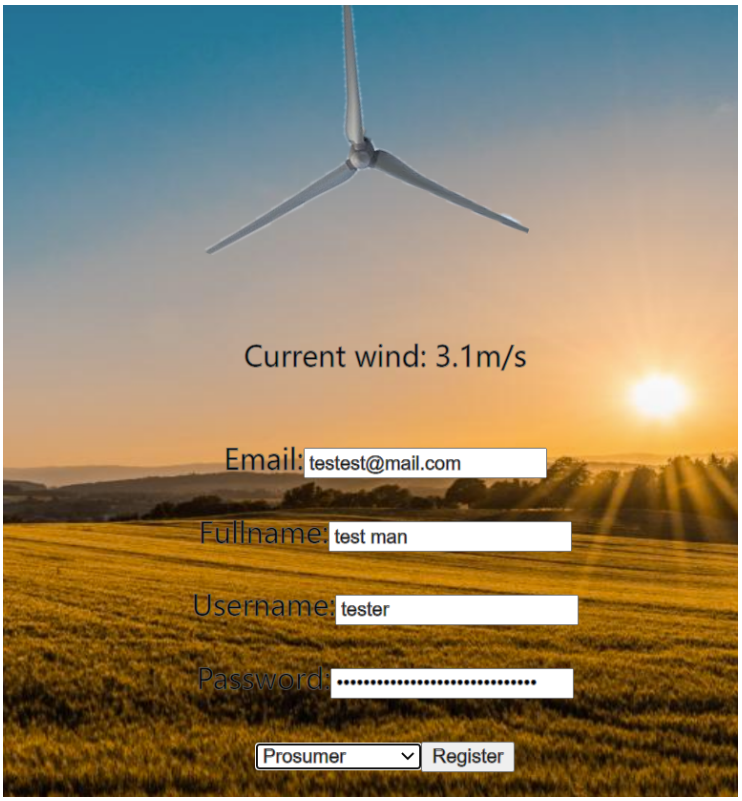

Figure 5: Register page for normal homes and prosumers

figure 5 shows the register page for normal home and prosumer users. It uses 4 free inputs and one 2 choice input, the email most be uniqe and an email, and the username most be uniqe for this post request to add a user, if the new user is succefully added, the user is also redirected to the prosumer page of the new user. The login button works as a redirect to the login page.

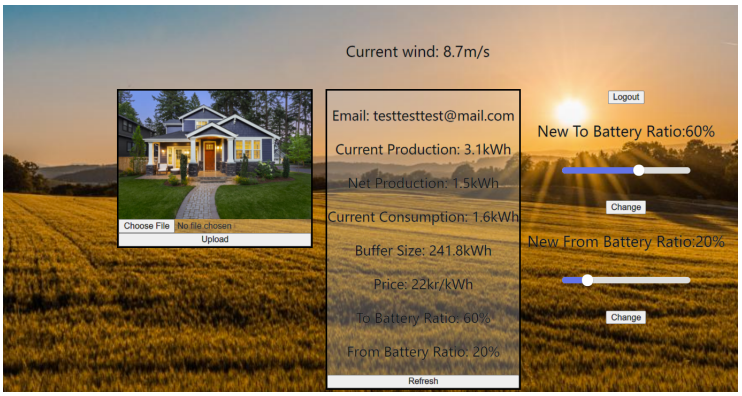

Figure 6: Prosummer and house page to manage and see their house

Figure 6 shows the prosumer and normal home page. This user is a prosumer. On the left the house picture is displayed(this can sometimes take a few sec to load in)you can easily upload a new picture there aswell by choosing a picture and pressing upload. The picture will be appended to a formdata that will then be sent with a post request to the backend. There the api handles it with the help of multer, and saves it to this user. If a picture is already uploaded it will be replaced. In the middle all the relevant data is displayed, this is refreshed with a get request once every 10 sedconds, there is also a refresh button if you dont want to wait. To the right we have the logout button that is a simple redirect to the login page, and the change batter ratios. The from and to battery ratios is changed with a put request to the api that happens when you press change, it will then take the value from the slider and update the resource.

Figure 7: The manager page with powerplant data and user list to acces all user data

Figure 7 shows the manager page, that you will be directed to after a successful admin login, to the left a list of all the users in the system can be seen, if you press any of the names, you will be directed to the manger pro page

where you can manage that user. Then you have the manager profile, where the picture can be changed if the manger wants to, under that blackouts will get displayed if they happen, this usually only happen when the power plant is shutdown, as it produces a lot of power. Then we have the info page where you can see the info for the manager, mostly some network status, price, modeled price, and also the power plant info. You can also change the to buffer ratio, toggle power plant status, and set price. There is also the simple logout button to take you back to the login page.

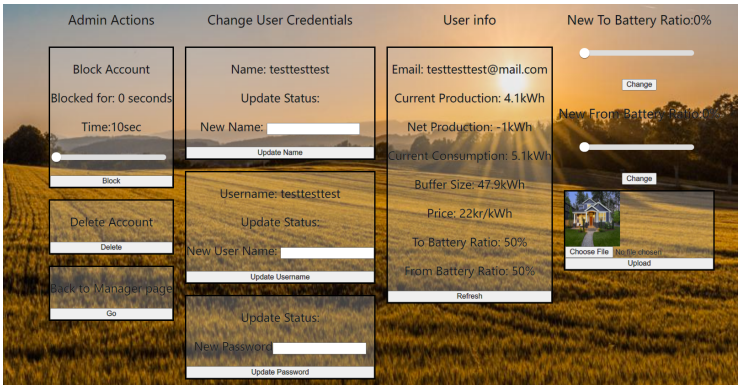

Figure 8: The manager page to manage a user, the page is dependent on the user pressed on user list

Figure 8 shows the manager prosumer page, where the manager can manage a user that it has pressed on the user list. To the right we first have the admin actions, block this user from selling, delete this user, and go back to admin page. The sell block will send a put request to force 100% to battery ratio for the amount of seconds on the slider. The delete button will send a delete request to remove the account and the house from the simulation, if the request is successful it will redirect you back to the admin page. The go back is a simple redirect to the admin page. After that we have the change credentials for the user. The admin can there change the name username and password of the user, all of these buttons will make a put request to the API that will update the resources. To the left of the page you can see the whole prosumer page in a compressed form and do all the actions that the prosumer themself can do on their own page.

3.2 API

The table below shows the mehtods available in our API through port 5000:

API			
URL	Method	Headers	Request Body
/price	GET	-	-
/price	PUT	adminemail, token	price
/wind	GET	-	-
/home	GET	email, token	-
/batteryRatio	PUT	email, token	batteryRatio
/consumeRatio	PUT	email, token	consumeRatio
/plantInfo	GET	adminemail, token	-
/plantRatio	PUT	adminemail, token	batteryRatio
/plantState	PUT	adminemail, token	-
/blackouts	GET	adminemail, token	-
/users	GET	adminemail, token	-
/users	DELETE	adminemail, token	email
/credentials	GET	adminemail, token, email	-
/credentials/username	PUT	adminemail, token	email, username
/credentials/fullname	PUT	adminemail, token	email, fullname
/credentials/password	PUT	adminemail, token	email, password
/sellBlock	PUT	adminemail, token	email
/adminCredentials	GET	adminemail, token	-
/user/login	POST	-	email, password
/user/register	POST	-	email, password, username, fullname
/pictures	POST	email, token, thistype	[formdata]
/pictures/:username	GET	-	-
/adminpictures	POST	adminemail, token, thistype	[formdata]
/adminpictures/:username	GET	-	-
/manager/login	POST	-	email, password

The required headers in the table indicate what sort of authentication is required. *email* and *token* of the user are required for normal user authentication. *adminemail* and *token* are required for manager authentication. Note that valid manager authentication will bypass normal authentication as well so a manager can access all of the API methods. Wind and price (not modeled) information as well as login and register functionality is publicly available.

3.3 Simulator

The Simulator simulates the wind speed and based on that calculates the production of all prosumers in the system. It also simulates the varying consump-

tion of all prosumers and consumers. The prosumers homes have a battery and can control the ratio of energy consumed from the battery and the ratio of energy sold to the marked during underproduction and overproduction respectively. Should the excess energy in the market not be enough the home will experience a blackout. The coal power plant is also simulated. When it is running it has a very high energy production and some of that energy goes to the buffer from which the users can draw energy even when the plant is offline. A modeled price is also calculated based on the excess energy and total consumption.

3.4 Advanced Features

The advanced features we have implemented from the assignments in canvas are the gauges for wind speed which is our wind turbine with rotation speed varying based on the wind, the sliders for setting the battery ratios and the ability to use the website on multiple devices simulatneously including on mobile phones where it works really well.

4 Discussion

4.1 Scalability

The current system is able to be used by a few number of users without any problems but it could not handle large ammounts of users very well. We do not use a load balancer of any kind so the bottle-neck would most likely be our back-end not being able to keep up with the ammount of requests. We have tried to reduce the number of requests sent by the front-end by bundling a lot of data together, sometimes more than required to reduce the overhead of many smaller requests. Due to a lack of time the frontend currently requests the pictures in an inefficient manner requesting the same image multiple times. Since pictures are pretty large the internet speed could also be a bottle-neck.

4.2 Security

The application uses json web tokens for authentication. Any information that is sent from the API that is tied to a specific user requires authentication. Manager authentication works in a similar way. The managers have access to all of the users data along with data that only managers can access.

In the current deployment there is no encryption which is a big security flaw. Anyone who intercepts the requests could get access to a valid token to gain access to and modify that users data. The solution to this is to use HTTPS which we managed to do using self signed certificates. However the self signed certificates were not trusted by web browsers. FireFox gave warnings but allowed the user to continue at their own risk. Google Chrome and Microsoft Edge refused to connect to our website however so we reverted back to HTTP in our current deployment.

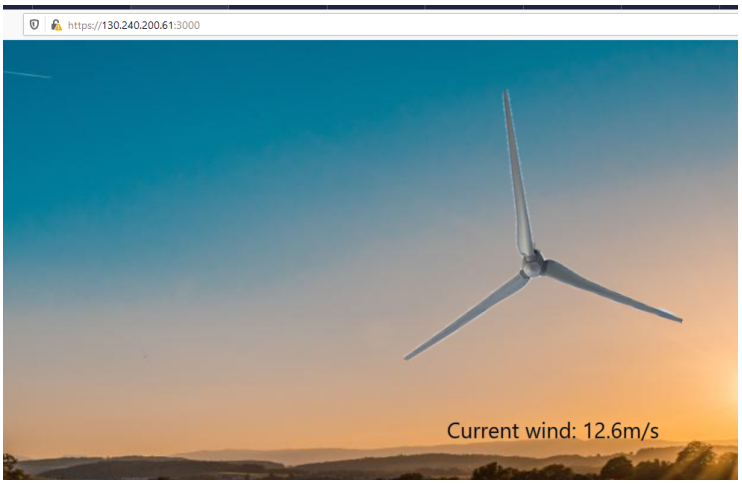

Figure 9: https connection

If we were to use https our handling of tokens should be secure. The tokens which are received on the front end are stored only in the memory of the application. They are not stored in localStorage, sessionStorage or even cookies and are not visible to the user. As a result of this the user needs to log in again after closing the site but it is a very secure solution as the token can not be accessed by other malicious applications.

Passwords for our users are hashed and salted using bcrypt before being put in the data base as a security meassure if someone would get access to the database.

4.3 Challenges

Initially it was challenging to plan what to work on and split up the work as we did not have a lot of web development experience. Once we got further into the project we started setting goals for tasks to accomplish at given dates and this increased productivity greatly.

There were many instances were there was a bug in the API and it was hard to pin point were it came from as it could be either in the front end or the back end. In those cases we had to verify the requests and responses to try to figure it out.

Due to using javascript there were a few bugs where misspelling a variable name would cause updates to non existing variables instead. These were sometimes hard to find since no warnings are issued.

4.4 Future work

To increase the security of our system the first thing which should be done is to use HTTPS. The way pictures are currently requested by the front end is not

very efficient as the same picture is requested multiple times potentially and there can be a delay of a few seconds to display the image.

If we had more time we also felt like we could have added some more advanced functionality like a chat function, that sounds like a really use full thing in a system like this, right now if the admin has something to say to a user regarding their house status or anything like that they would have to mail them, the same goes for the users, that would need to mail the admin. It would be better if you could just ask directly on the site and solve the problem where it originated. We could also have added more gauges to give more immersion and a better understanding of the what the numbers mean.

5 References

This was the base for our token authentication.
https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i

A video which got us started on our API.
https://www.youtube.com/watch?v=pKd0Rpw7O48

This was the 2 toutorials that helped us set up react at first.
https://www.youtube.com/watch?v=hQAHSlTtcmY&t=518s&ab_channel=WebDevSimplified
https://www.youtube.com/watch?v=_vsGmIvu4Rg&ab_channel=UmeshRamya

This was how we did protected routes even if its modifed to work for our site
https://dev.to/mychal/protected-routes-with-react-function-components-dh

6 Appendix

6.1 Time Report Analysis

After looking trough our time report we can see that we had a really slow start. We have reflected on why that happened and our conclusion was that it was hard to start. When first faced with a large project like this that will fill the course it is hard to know what we should do first and how we will reach the end. The more work we did the easier it was to understand what we should do next. Another part is that we at first tried making all the code together, this was inefficient as only one of us could really write at a time. We fixed this by splitting so one of us to mostly writes the front end code, and one of us to mostly writes the back end code. We mostly worked with both group members in discord so both always had an input and understood what was done on the two parts. After we started doing this we could easily make progress and there was a lot to do at all times and the hours started stacking up. The problem with our slow start was that we had to put in a lot of work later in the course and not much time could be put on more advanced functionality.

6.2 Contributions

Both Isak Sundell and Enar Eriksson did the simulator together with some split work and some pair programming after that Isak mostly focused on the API and the rest of the backend and Enar mostly on the react server as front end. Both also always discussed both the parts on how we should solve the diffrent problems, but the code it self was split up to be more efficent.

6.3 Grade

We believe we should at least get a 3 as all the basic functionality is implemented. Also we have implemented a little bit of advanced functionality but are unsure if it is enough for the grade 4. Since we have put in the same ammount of time and effort and designed the system together we should get the same grade.

6.4 Links

Github: `https://github.com/EnarEriksson/M7011E_course`
Time Report at github
URL to the website: `http://130.240.200.61:4000/`

To login as manager use email: admin@mail.com and password: admin123, navigate to admin login or go to /adminLogin

To login as one of our existing users use email: testtesttest@mail.com, password: testtesttest

To Register a prosumer or non prosumer go navigate to register or go to /reg