

Final project report
MPSE2201 System design and engineering

Group 10

April 9, 2020



Abstract

Contents

Abstract	2
List of Figures	7
Abbreviations	10
Introduction	11
Assignment	11
Report	11
Our Systems Engineering Process	11
Planning	11
Contract of Operations	11
Complications due to COVID-19	13
Tools	13
Internal Discussions	18
The Problem	18
Understanding the Problem	18
Context	18
Stakeholders	20
Requirements	20
User Requirements	20
Effectiveness Measures	20
System Requirements	22
Functional	22

Temporal Performance	22
Non-Temporal Performance Requirements	22
Interface Requirements	22
Design Requirements	22
Energy Consumption	22
System Architecting	23
Tier 0: The System in Its Context	23
Operation	23
Behavior	24
Structure	25
Harvesting Energy	26
Storing Energy	26
Generating Heat	27
Trade-Off	27
Harvesting Energy	27
Storing Energy	28
Generating Heat	29
Recommendations	29
Derived Requirements	30
Tier 1: The System	30
Operation	30
Control System	31
Photovoltaic System	31
Energy Storage system	32
Geothermal Heating/Cooling	32

Behavior	33
Structure	34
Trade-Off	36
Control System	36
Recommendations	37
Derived Requirements	37
Tier 2: Subsystems	37
Energy Storage System	37
Behavior	37
Structure	38
Derived Requirements	39
Control System	40
Operation	40
Behavior	40
Structure	45
Derived Requirements	46
Build and Test Plan	46
Verification	46
Production	46
Capacity for Storage	47
Prototypes / Proof-of-Concept	48
Machine Learning	48
Predicting Solar Panel Power Production	48
Predicting Power Prices	50

Web-Application	52
Risk	53
Economics	54
Our Budget	54
Our Cash Flow	54
Residents Budget	55
Residents Cash Flow	55
Conclusion	55
References	56
Appendices	57
Figures	57
Code	57
Historical Weather Web-Crawler	57
MongoDB Filler	57
Historical Hourly Avg. Power Price	61
Machine Learning	62
Solar Power Production	62
Machine Learning, Power Price Prediction	64
Web-Application	66

List of Figures

1	Jira screenshot	13
2	Microsoft Teams screenshot	14
3	Gantt diagram screenshot	15
4	Github web-application repo screenshot	16
5	MongoDB screenshot	17
6	A 3D model of the building	18
7	The man-made systems in our systems immediate context	19
8	Key-driver-graph for OBOS	20
9	Key-driver-graph for the users of our system	21
10	A technical budget displaying the energy needs of the building	22
11	Use case diagram for the system	23
12	Black box of our system	24
13	FFBD within our system	24
14	The system modeled as a UML class diagram, where all the classes are still abstract	25
15	Morphological diagram, showing possible concepts	26
16	Pugh matrix for harvesting energy	27
17	Pugh matrix for storing energy	28
18	Pugh matrix for generating heat	29
19	Morphological diagram, showing the recommended concept	29
20	Use case diagram, between subsystems	30
21	Black box of Control System	31
22	Black box of Photovoltaic System	31
23	Black box of Energy Storage System	32
24	Black box of Geothermal Heating System	32

25	Black box of Geothermal Cooling System	33
26	FFBD of the respective subsystems	33
27	Class diagram showing the subsystems	34
28	A more detailed structural model of our system and its context	35
29	Pugh matrix for Control System	36
30	Behavior of the Energy Storage System	37
31	Class diagram of the Energy Storage System	38
32	Average hourly power prices 2013-2020	39
33	Use case diagram within the control system	40
34	FFBD for the Control Unit	41
35	FFBD for the MCU	42
36	FFBD for the continuous data gathering software	43
37	FFBD for the machine learning software	43
38	FFBD for the application/website	44
39	FFBD for the developer tools	44
40	Class diagram for Control System	45
41	FINN PÅ EN GOD CAPTION TARALD	46
42	FINN PÅ EN GOD CAPTION TARALD	47
43	Machine learning prediction results for shuffled solar production	48
44	Machine learning prediction results for chronological solar production	49
45	Power prices 2013-2020 in NOK/MWh	50
46	Machine learning prediction results for shuffled power prices	51
47	Machine learning prediction results for chronological power prices	51
48	Screenshots of the website prototype	52
49	Risk matrix for development	53

50	Risk matrix for system operation	53
51	Budget for our company	54
52	Cash flow graph for our company	54
53	Budget for customer	55
54	Cash flow graph for customer	55
55	Use Case Diagram Example	57

Abbreviations

COVID-19 Corona Virus Disease 2019

FFBD Functional Flow Block Diagram

MCU Main Communication Unit

SQL Structured Query Language

UML Unified Modeling Language

V2B Vehicle-to-Building

Introduction

Assignment

Report

Our Systems Engineering Process

Planning

Contract of Operations

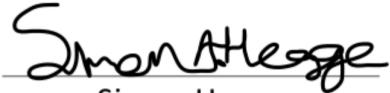
Contract of Operations

Terra Solaris Borealis

- The team members are obligated to attend all group meetings.
- If a team member is unable to attend a meeting, he shall inform the systems engineer in advance.
- All assigned work shall be submitted before its deadline.
- In case of disagreement, the systems engineer shall call a vote.
- In the case of a tie, the systems engineer has the final say.
- Every opinion and idea have the right to be expressed freely.
- All members of the team shall act with respect towards the others



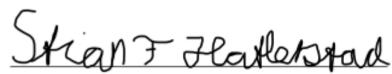
Tarald Vestbøstad



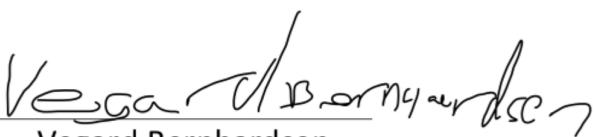
Simen Hegge



Erik Kvalvik



Stian Hatlestad



Vegard Bernhardsen



Kent Odde

Complications due to COVID-19

Tools

Jira

The screenshot shows a Jira board titled "Finish report". The board has three columns: "TIL UTRÖRING", "UNDER ARBBD", and "UTFÖRT".

TIL UTRÖRING	UNDER ARBBD	UTFÖRT
Finish requirement document TSB-172 (TV)	Write planning section TSB-164 (1)	Requirements TSB-165 (1)
Write conclusion TSB-178	Write introduction TSB-177 (1)	The Problem TSB-171 (1)
Iterate system requirements TSB-173	Write tradeoff sections TSB-166 (1)	Make risk matrix TSB-176 (1)
Write testing and verification section TSB-175 (1)	Write design section TSB-167 (1)	
	Write risk section TSB-170 (1)	
	Finish 3D model TSB-174 (1)	

The left sidebar shows project navigation: Terra Solaris Borealis Software project, TSB tavle Board, Backlog, Active sprints, Reports, Releases, Issues and filters, Components, Add item, and Project settings.

Figure 1: Jira screenshot

Microsoft Teams

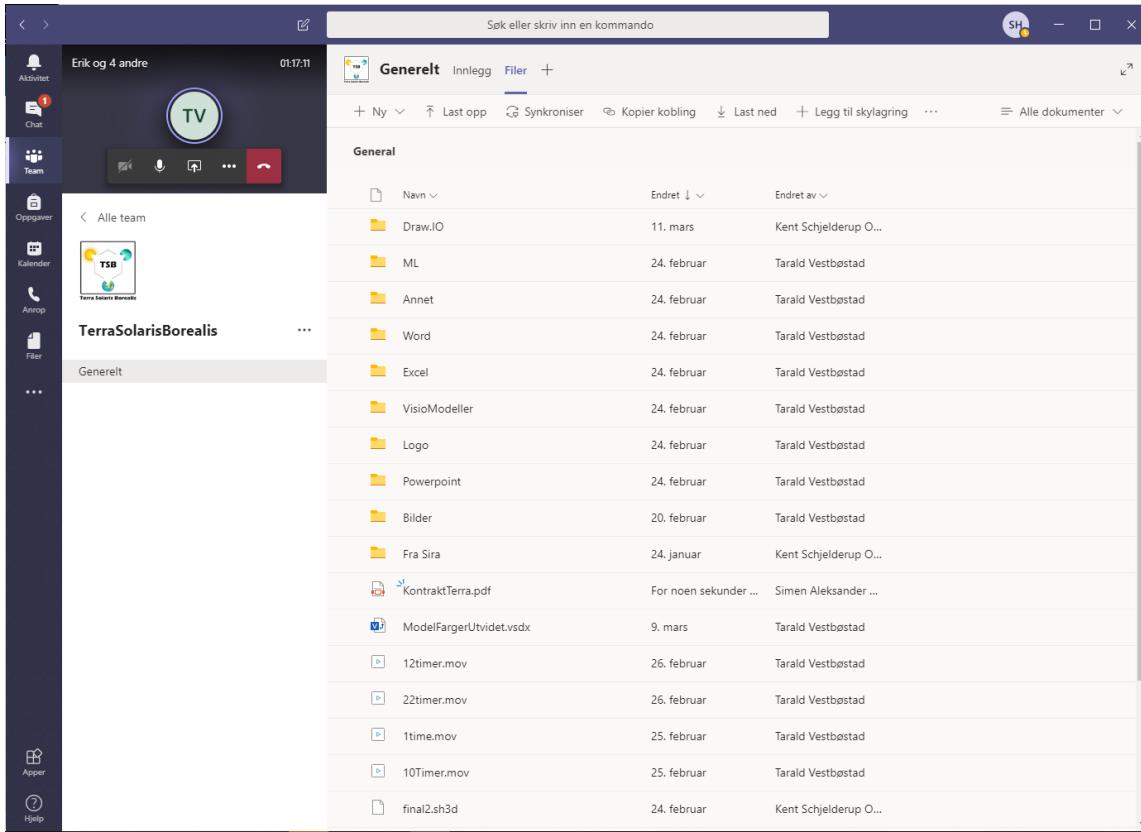


Figure 2: Microsoft Teams screenshot

Facebook Messenger

Microsoft Visio

Microsoft Word

Microsoft Excel

Adobe Illustrator

Gantt

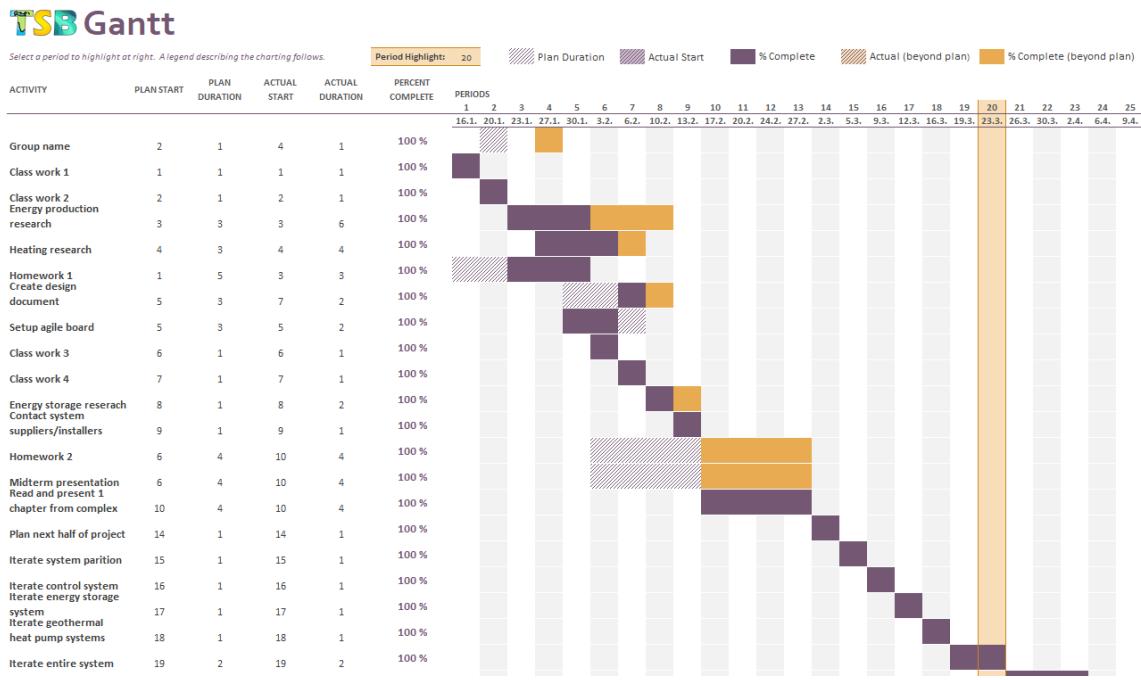


Figure 3: Gantt diagram screenshot

Draw.io

Sweet Home 3D

Solidworks

Programming Languages

Github

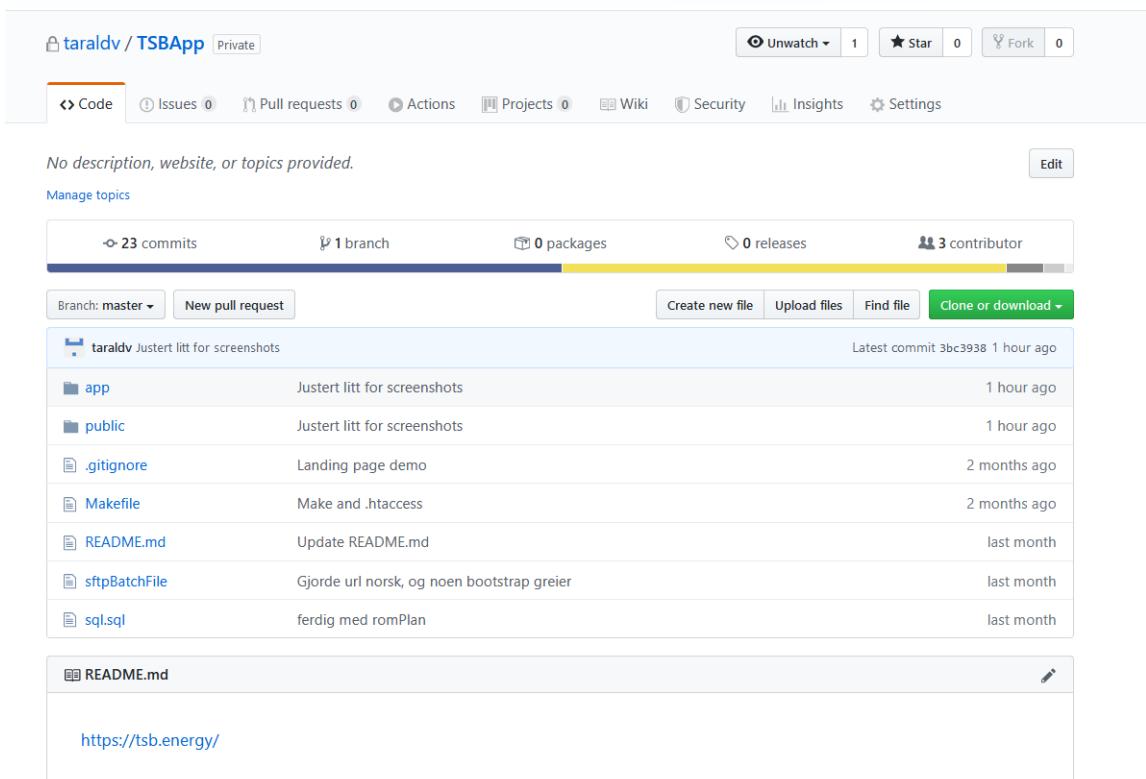


Figure 4: Github web-application repo screenshot

MySQL

MongoDB

Terra.weather

COLLECTION SIZE: 11.25MB TOTAL DOCUMENTS: 105188 INDEXES TOTAL SIZE: 1012KB

Find Indexes Aggregation Search^{BETA}

FILTER {"filter": "example"}

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("5e6516b314f6be4bcd268db2")
time: "2005-01-01T00:00"
air_temperature: -0.3
cloud_area_fraction: 1
cloud_low_weight: 2100
sun(duration_of_sunshine P1M): 59.2
weather_type: 5

_id: ObjectId("5e6516b314f6be4bcd268db3")
time: "2005-01-01T01:00"
air_temperature: 6

_id: ObjectId("5e6516b314f6be4bcd268db4")
time: "2005-01-01T02:00"
air_temperature: 15

_id: ObjectId("5e6516b314f6be4bcd268db5")
time: "2005-01-01T03:00"
air_temperature: -5
cloud_area_fraction: -3
weather_type: 43

_id: ObjectId("5e6516b314f6be4bcd268db6")
time: "2005-01-01T04:00"
air_temperature: -1.4

_id: ObjectId("5e6516b314f6be4bcd268db7")
time: "2005-01-01T05:00"
air_temperature: -4.5
```

Figure 5: MongoDB screenshot

Internal Discussions

The Problem

Understanding the Problem

Context



Figure 6: A 3D model of the building

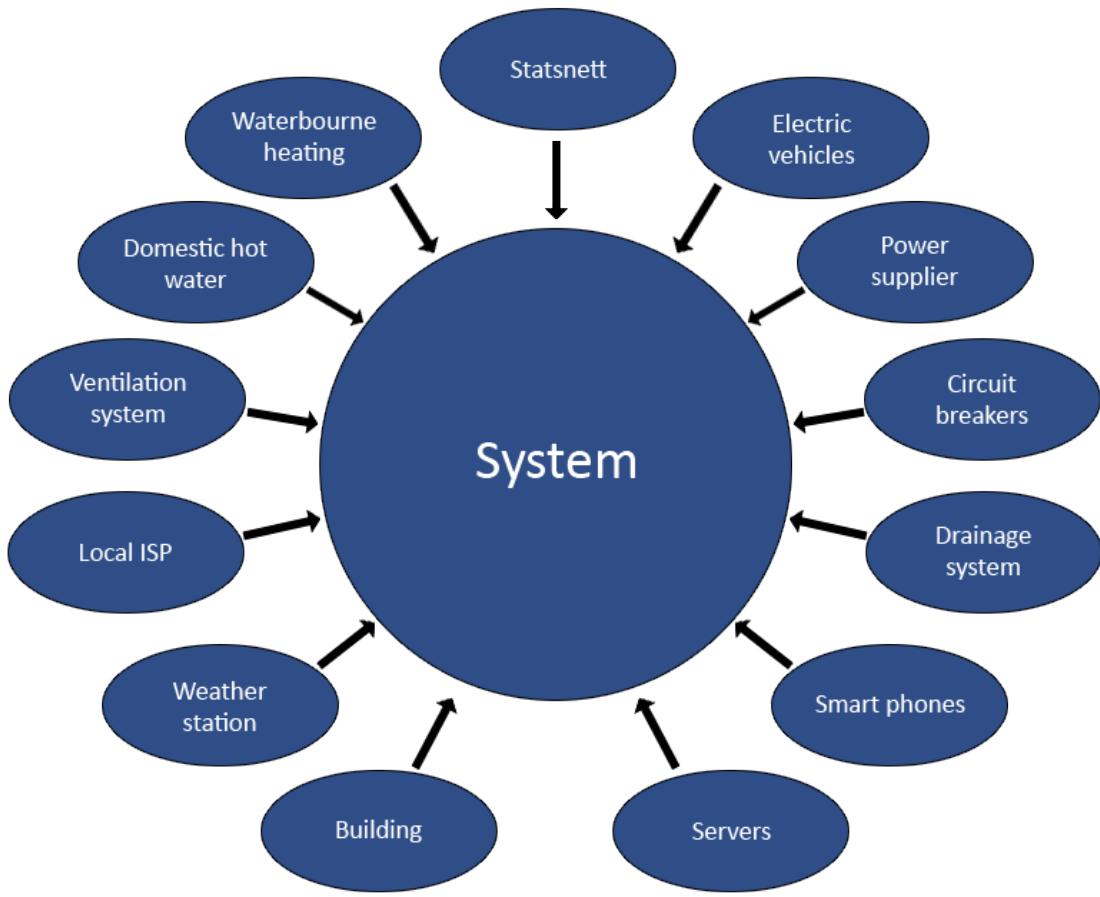


Figure 7: The man-made systems in our systems immediate context

Stakeholders

Requirements

User Requirements

Effectiveness Measures

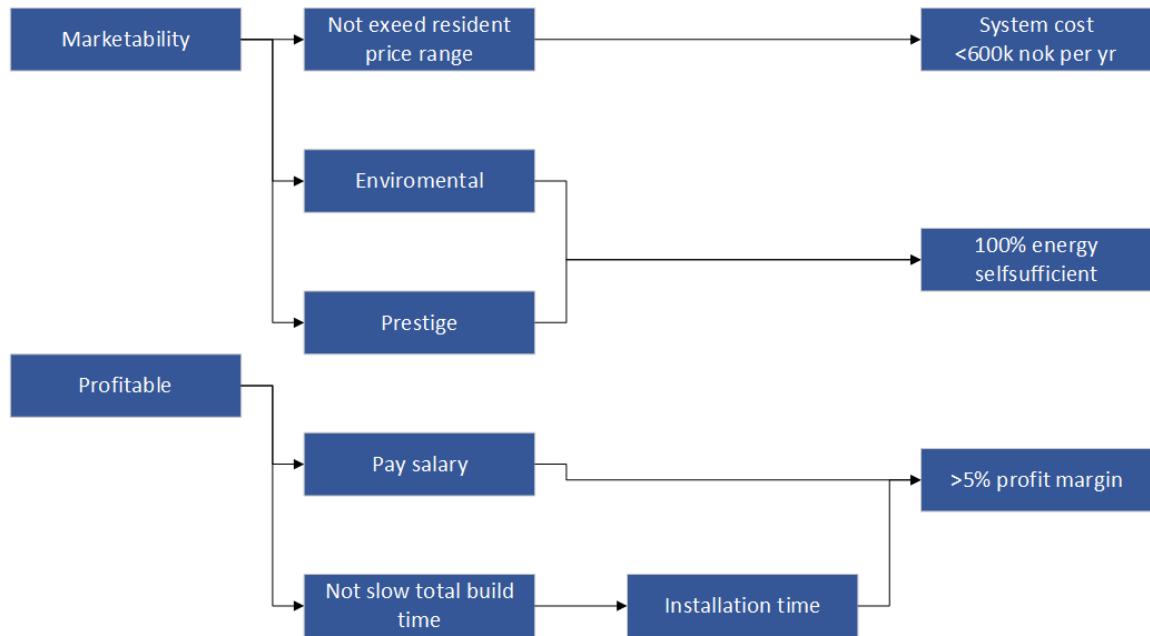


Figure 8: Key-driver-graph for OBOS

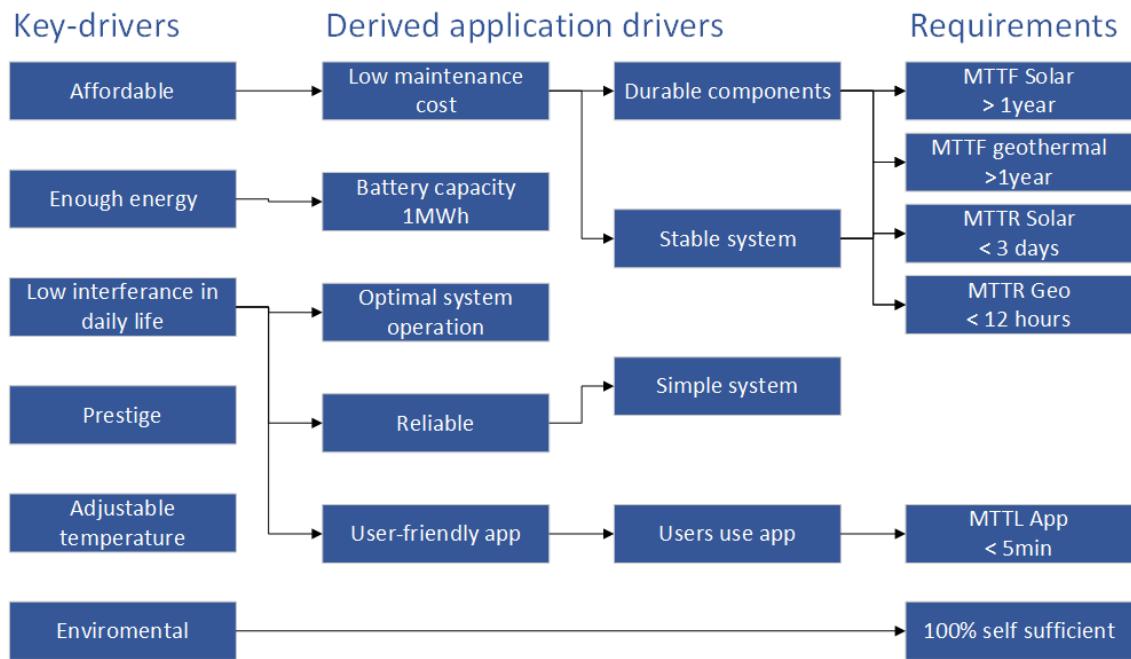


Figure 9: Key-driver-graph for the users of our system

System Requirements

Functional

Temporal Performance

Non-Temporal Performance Requirements

Interface Requirements

Design Requirements

Energy Consumption

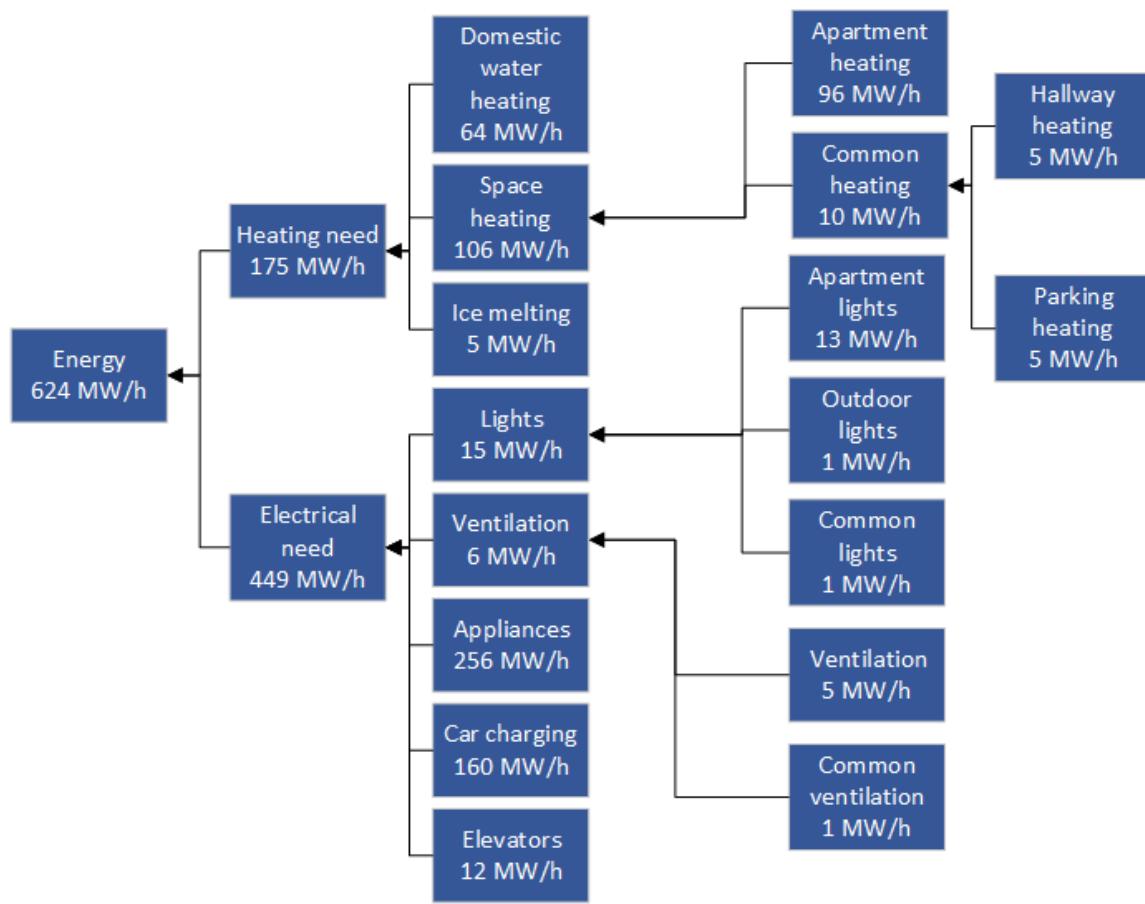


Figure 10: A technical budget displaying the energy needs of the building

System Architecting

Tier 0: The System in Its Context

Operation

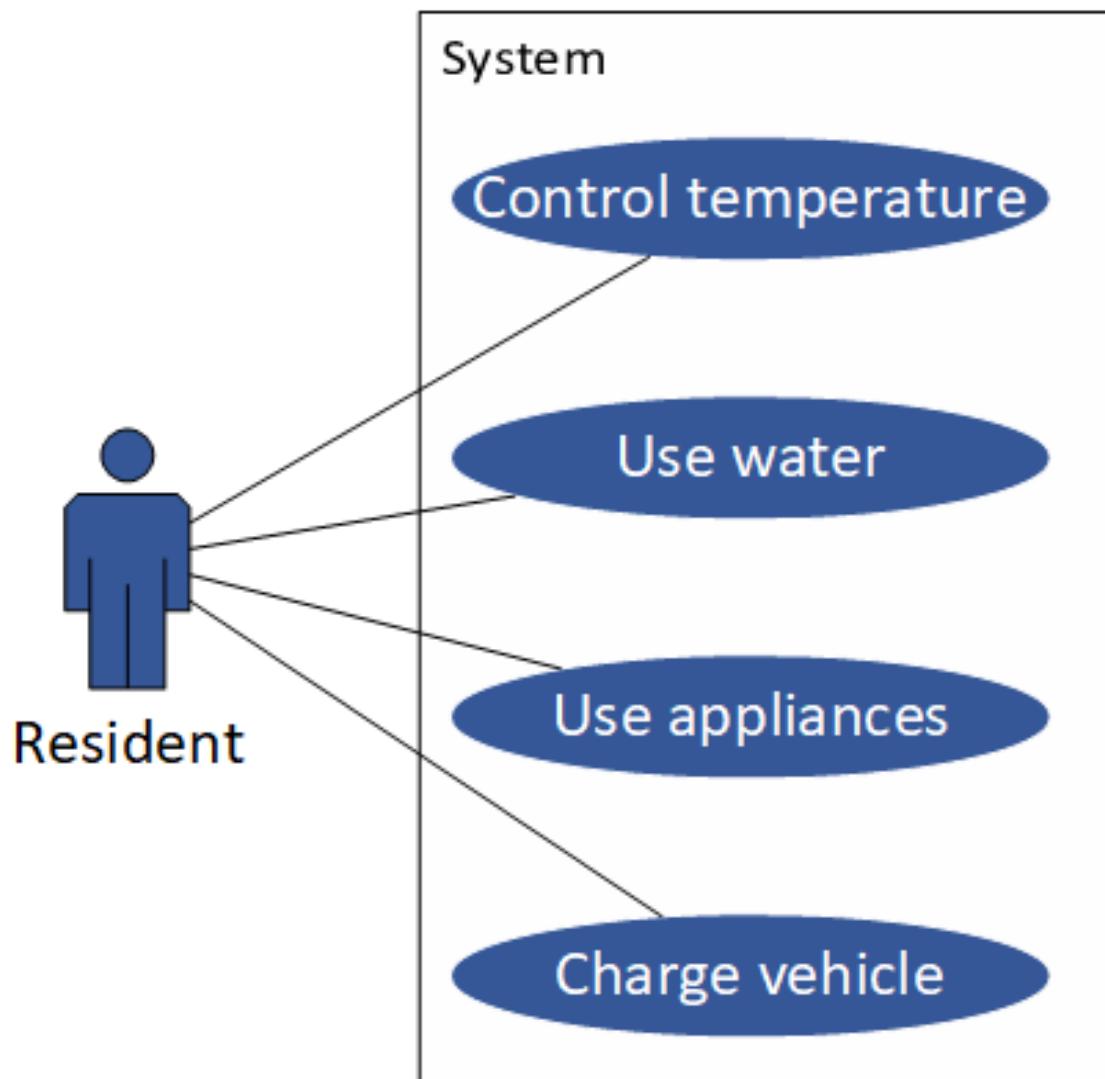


Figure 11: Use case diagram for the system

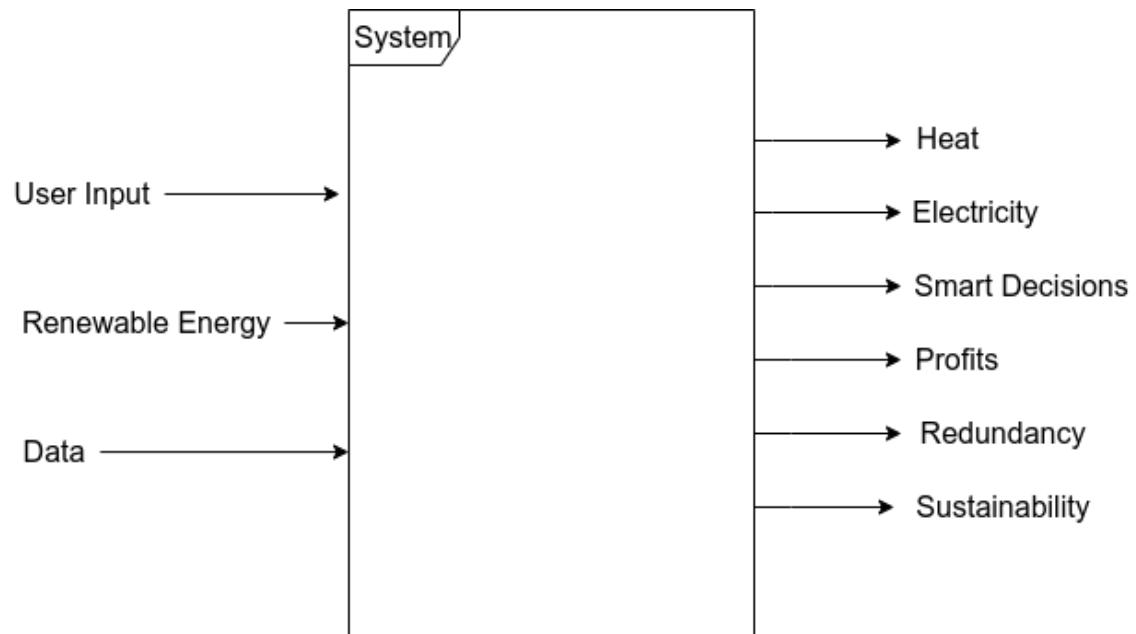


Figure 12: Black box of our system

Behavior

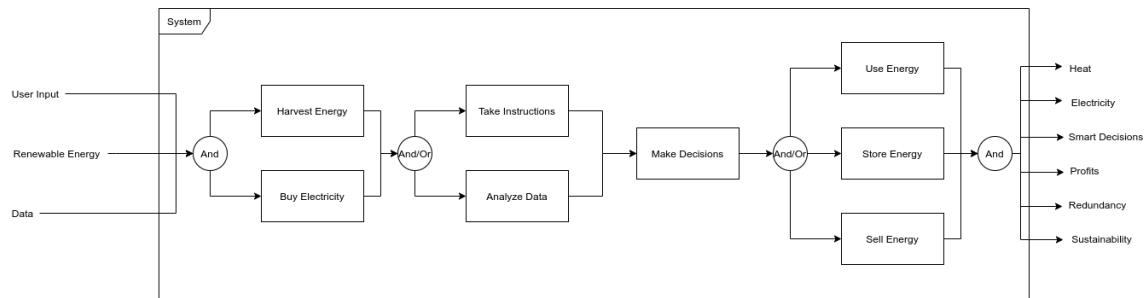


Figure 13: FFBD within our system

Structure

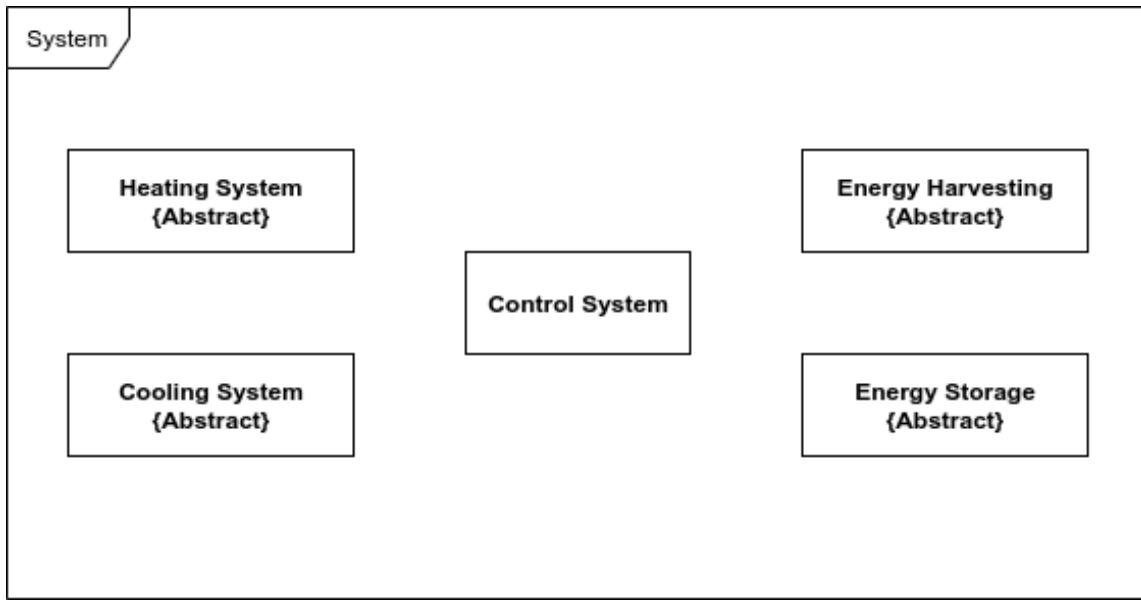


Figure 14: The system modeled as a UML class diagram, where all the classes are still abstract

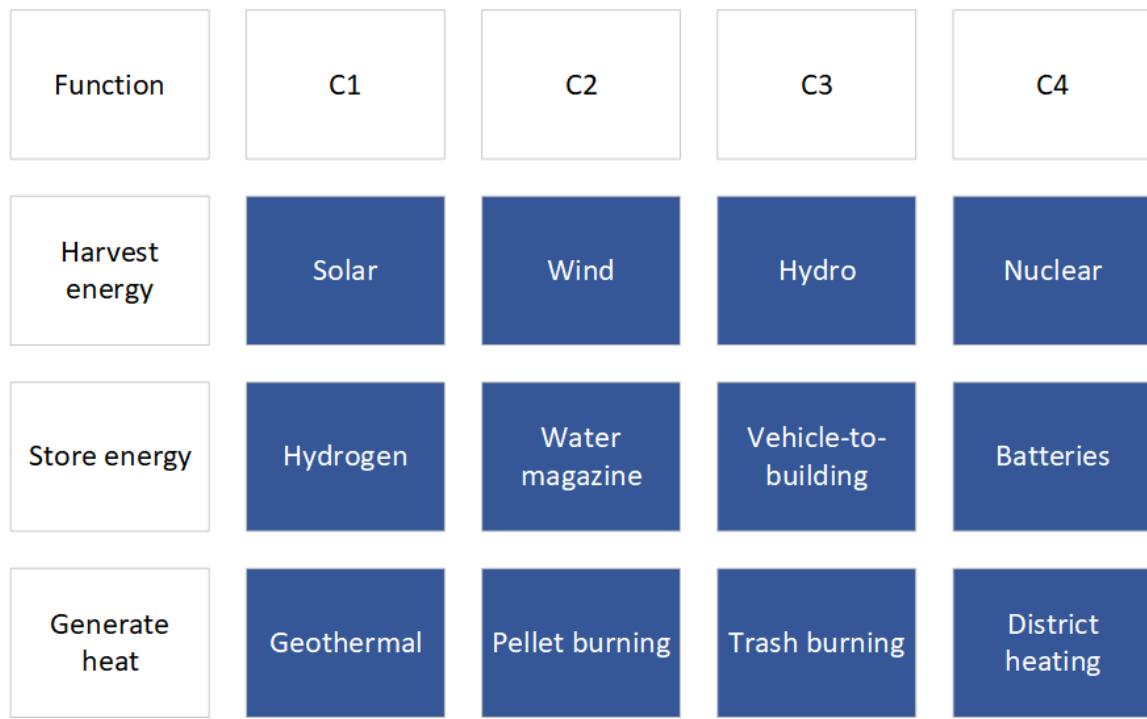


Figure 15: Morphological diagram, showing possible concepts

Harvesting Energy

Solar

Hydro

Wind

Wind

Nuclear

Storing Energy

Batteries **Vehicle-to-Building (V2B)** **Hydrogen** **Water Magazine**

Generating Heat

Burning Trash

Geothermal

Burning Pellets

District Heating

Trade-Off

Harvesting Energy

Criteria	Solar	Nuclear	Wind	Hydro	Thermal-Generator
Sustainability	5	3	5	3	2
Cost	5	1	4	3	2
Difficulty	5	1	5	3	3
System complexity	5	1	5	3	3
Lifetime	3	5	3	3	3
Improvability	1	2	1	2	2
Support	4	1	2	4	3
Outsourcing	4	2	4	4	3
Controversy	5	1	3	5	3
Availability	5	1	5	1	3
Safety	5	5	2	5	2
Efficiency	3	5	3	3	4
SUM	50	28	42	39	33

Figure 16: Pugh matrix for harvesting energy

Storing Energy

Criteria	V2B	Hydrogen	Only batteries
Sustainability	2	5	3
Cost	4	1	3
Difficulty	2	1	5
System complexity	2	3	4
Lifetime	2	4	2
Tenant capacity	2	3	3
Location	3	5	5
Independency	3	2	4
Improvability	4	5	3
Support	1	2	5
Outsourcing	5	1	1
Parking	5	1	1
Park	4	5	5
Availability	4	3	5
Strength	5	3	4
Safety	5	2	5
Uniqueness	5	4	1
Solar cell area	5	2	2
Energy need	2	3	3
Efficiency	5	2	5
SUM	70	57	69

Generating Heat

Criteria	Waste-to-energy	Geothermal heat pump	Pellet burning	Electricity
Heat generation / price	4	5	3	1
Size	2	3	4	5
CO2 emissions	2	4	3	5
Electricity/heat ratio	4	5	4	1
Renewable	3	5	3	4
Cooling synergy	1	5	1	1
Safety	1	4	1	4
Lifetime	3	3	3	5
SUM	20	34	22	26

Figure 18: Pugh matrix for generating heat

Recommendations

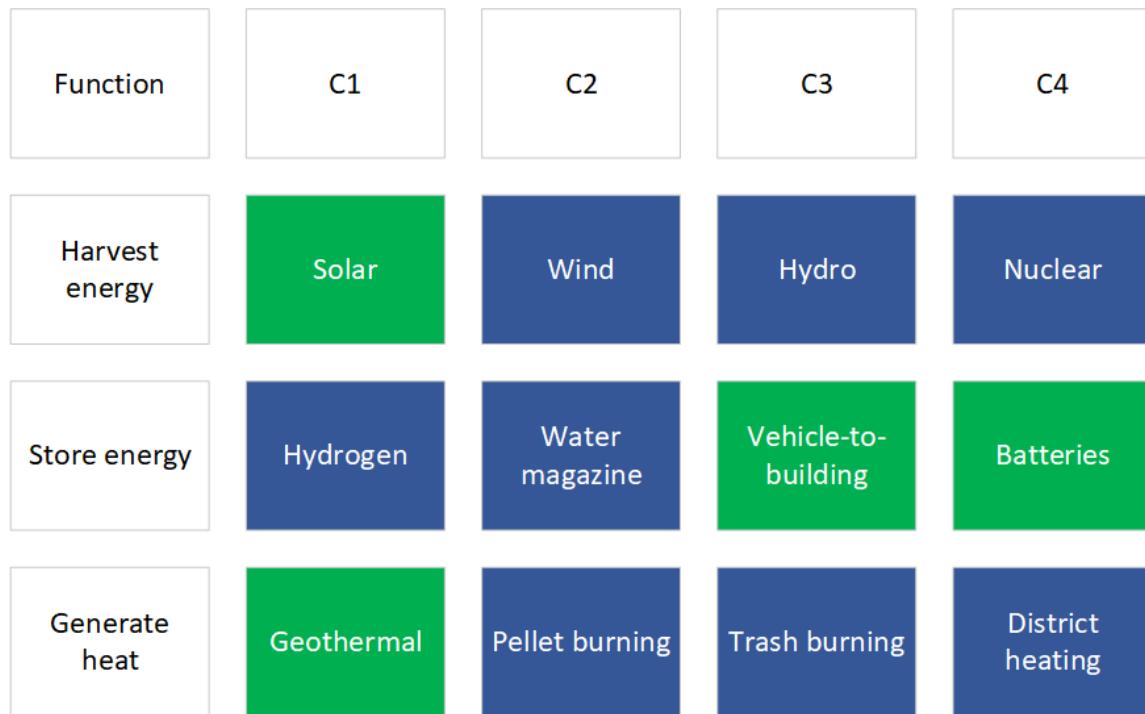


Figure 19: Morphological diagram, showing the recommended concept

Derived Requirements

Tier 1: The System

Operation

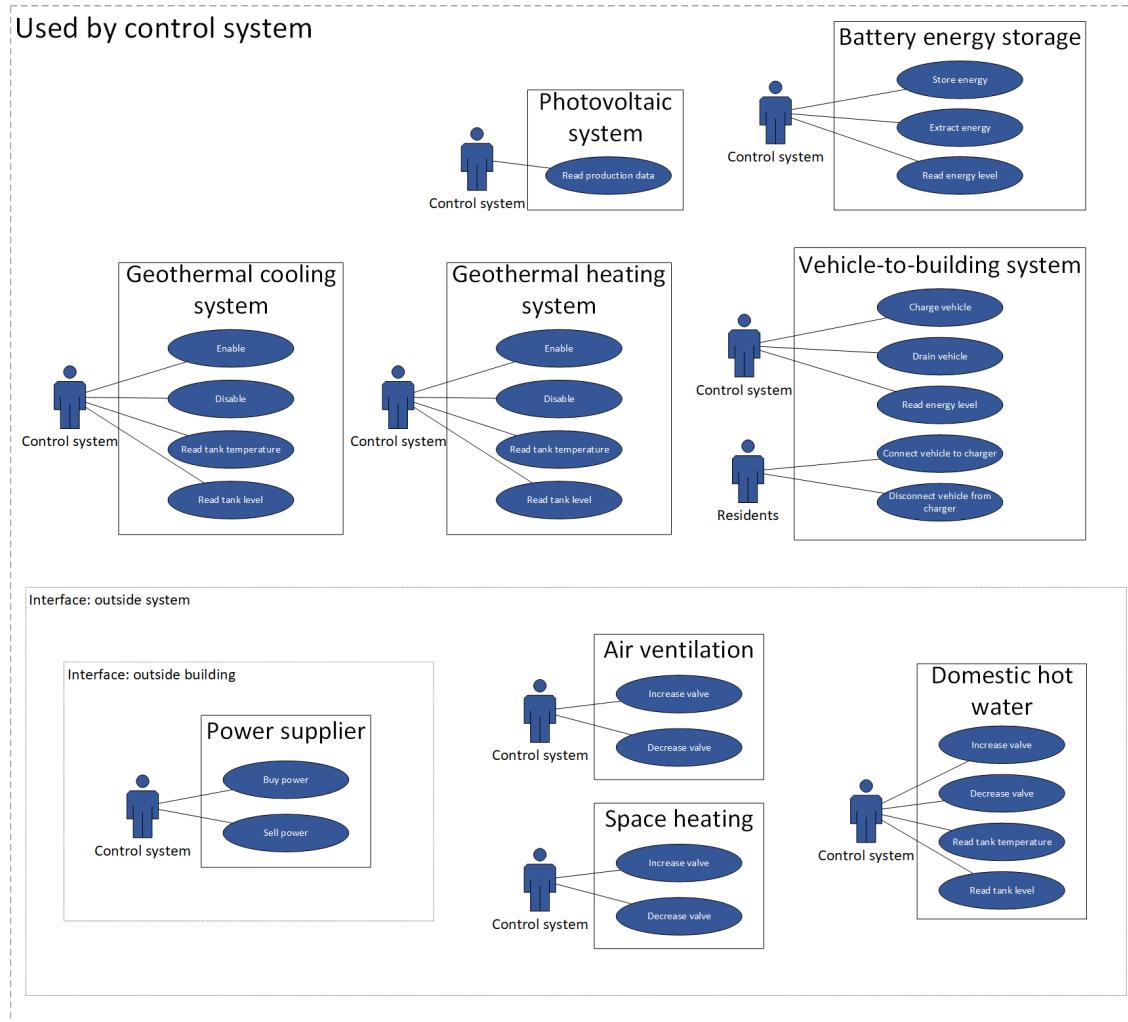


Figure 20: Use case diagram, between subsystems

Control System

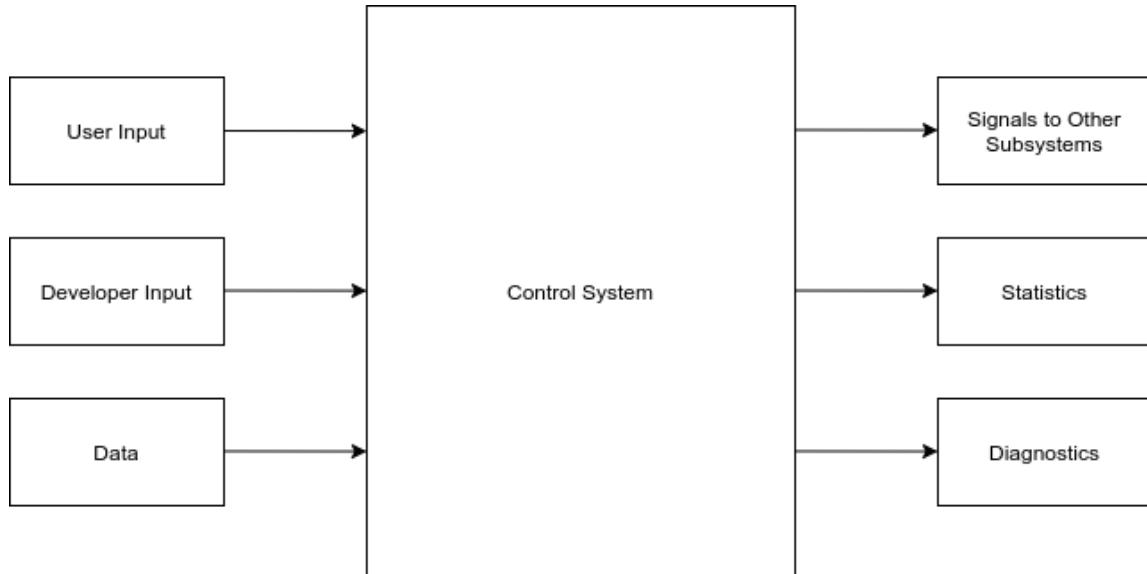


Figure 21: Black box of Control System

Photovoltaic System

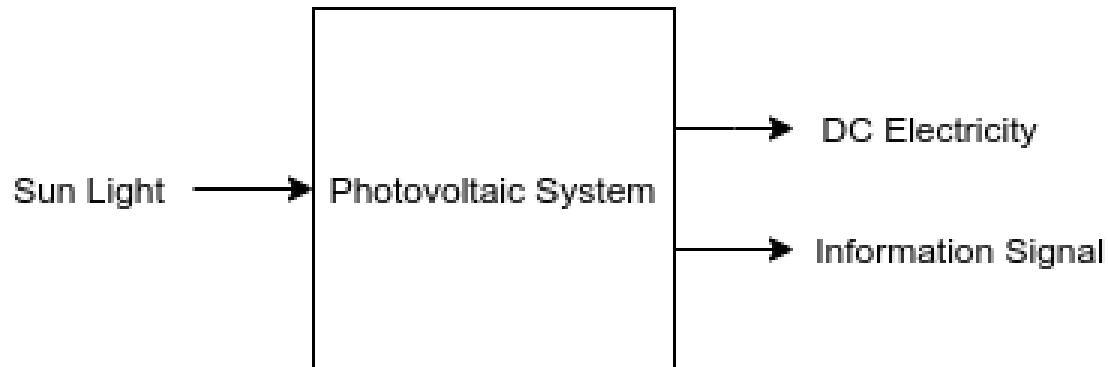


Figure 22: Black box of Photovoltaic System

Energy Storage system

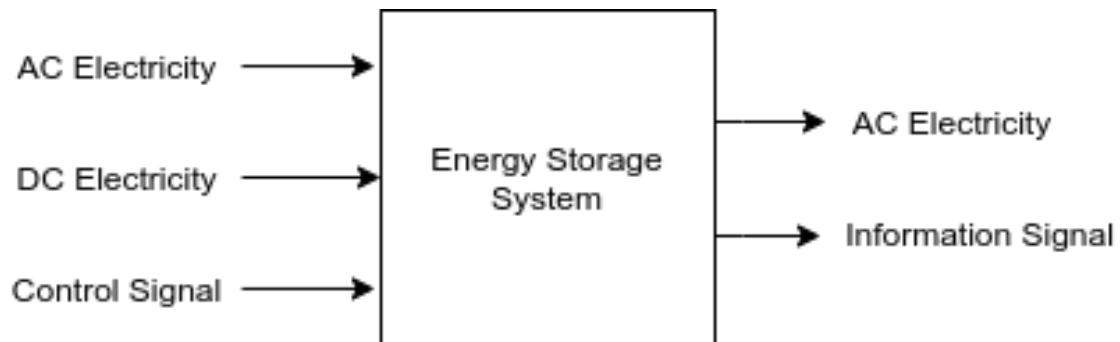


Figure 23: Black box of Energy Storage System

Geothermal Heating/Cooling

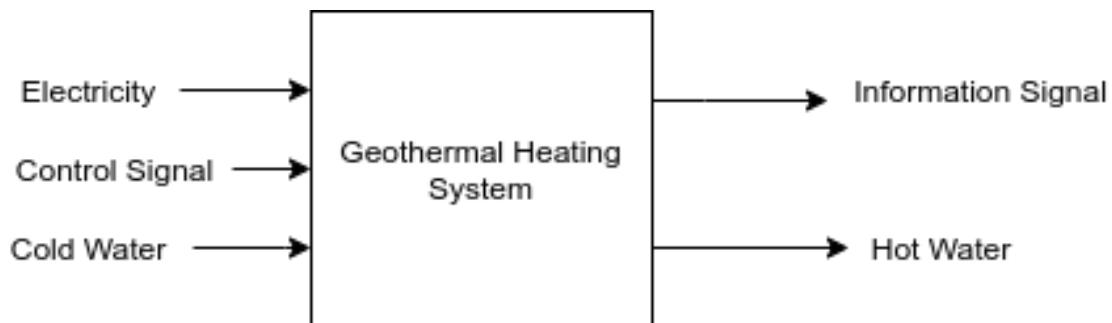


Figure 24: Black box of Geothermal Heating System

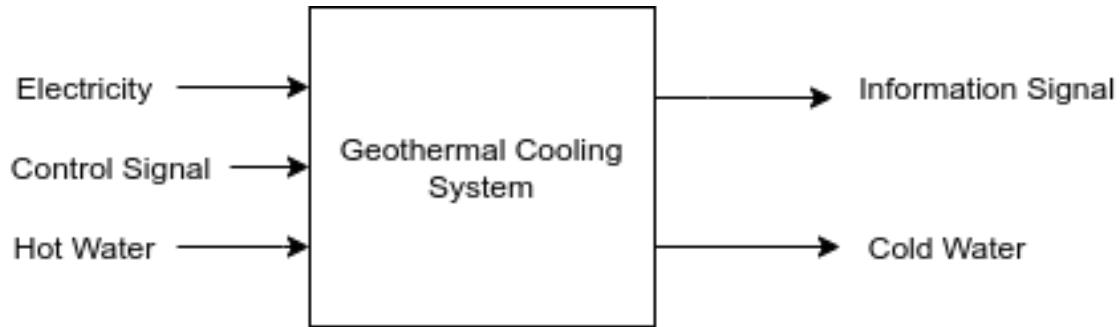


Figure 25: Black box of Geothermal Cooling System

Behavior

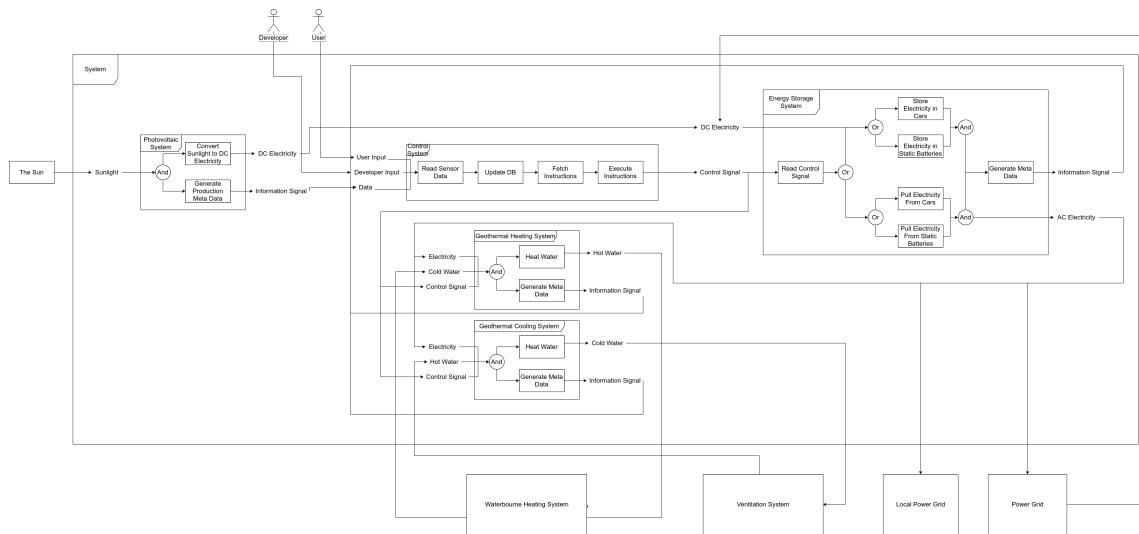


Figure 26: FFBD of the respective subsystems

Structure

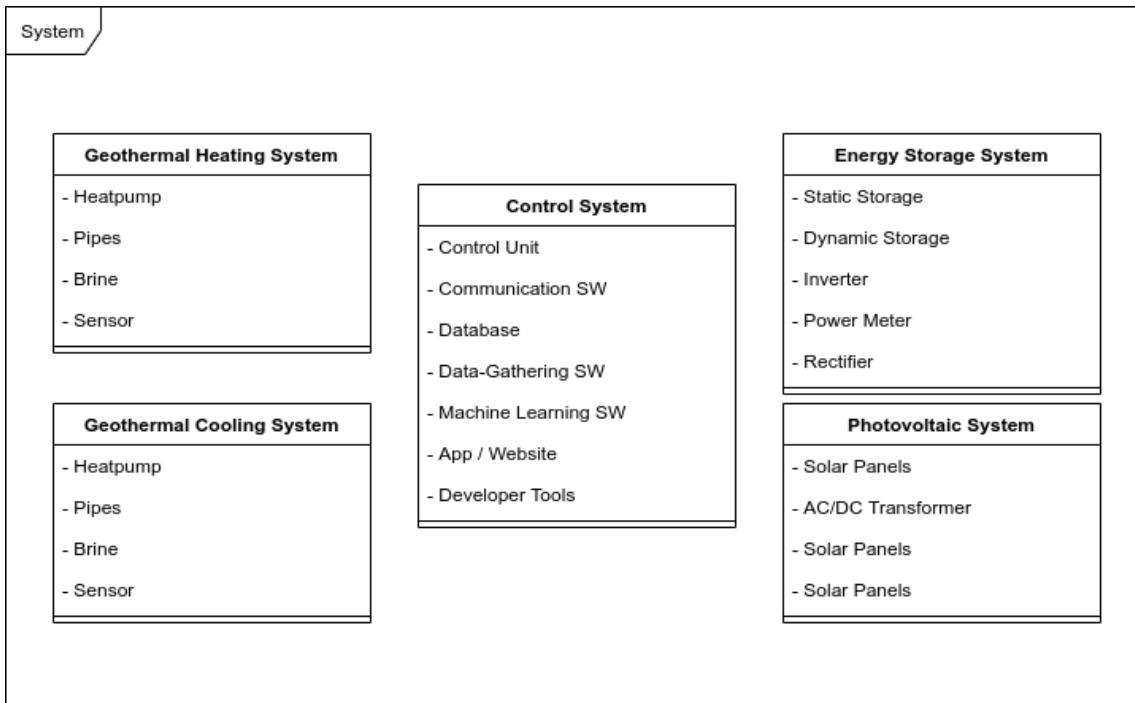


Figure 27: Class diagram showing the subsystems

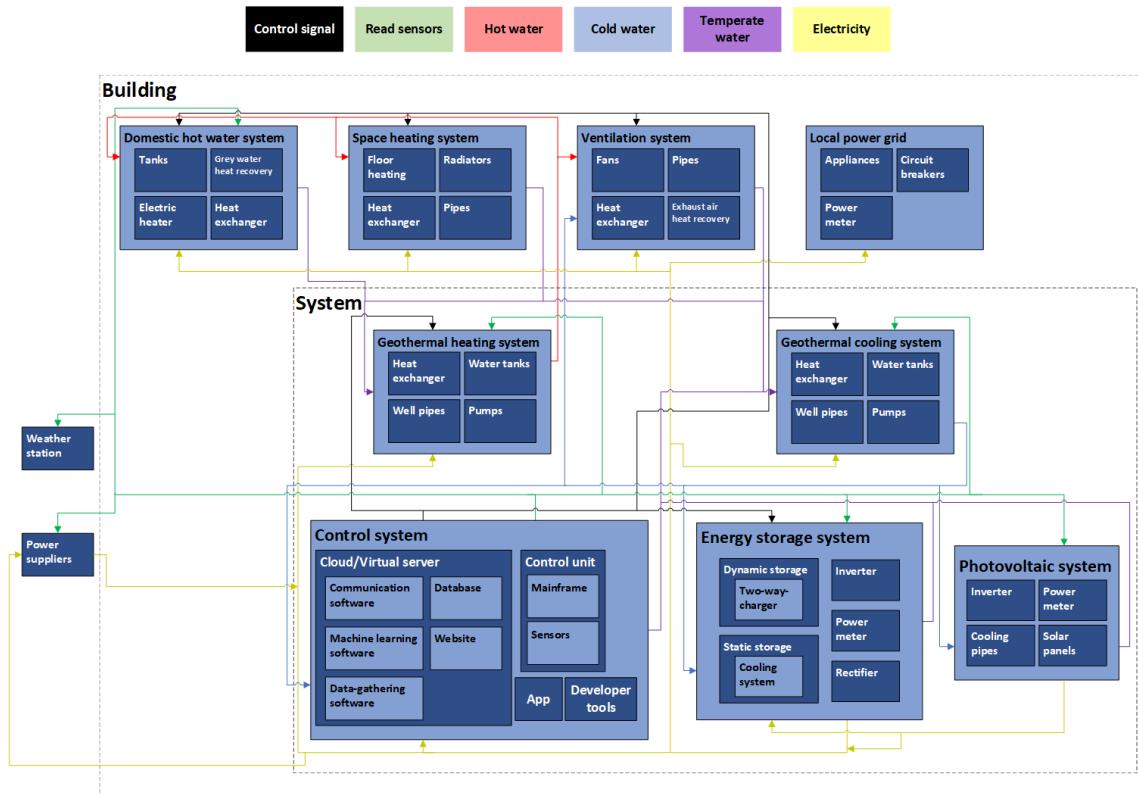


Figure 28: A more detailed structural model of our system and its context

Trade-Off

Control System

Criteria	Common cloud server	Individual building server
Response	2	5
Computing power / price	3	4
Modularity	5	3
Simplicity	4	3
Updating software	5	2
Hardware maintenance	5	1
Storage / price	5	3
Physical access	1	5
Safety	4	5
SUM	34	31

Figure 29: Pugh matrix for Control System

Recommendations

Derived Requirements

Tier 2: Subsystems

Energy Storage System

Behavior

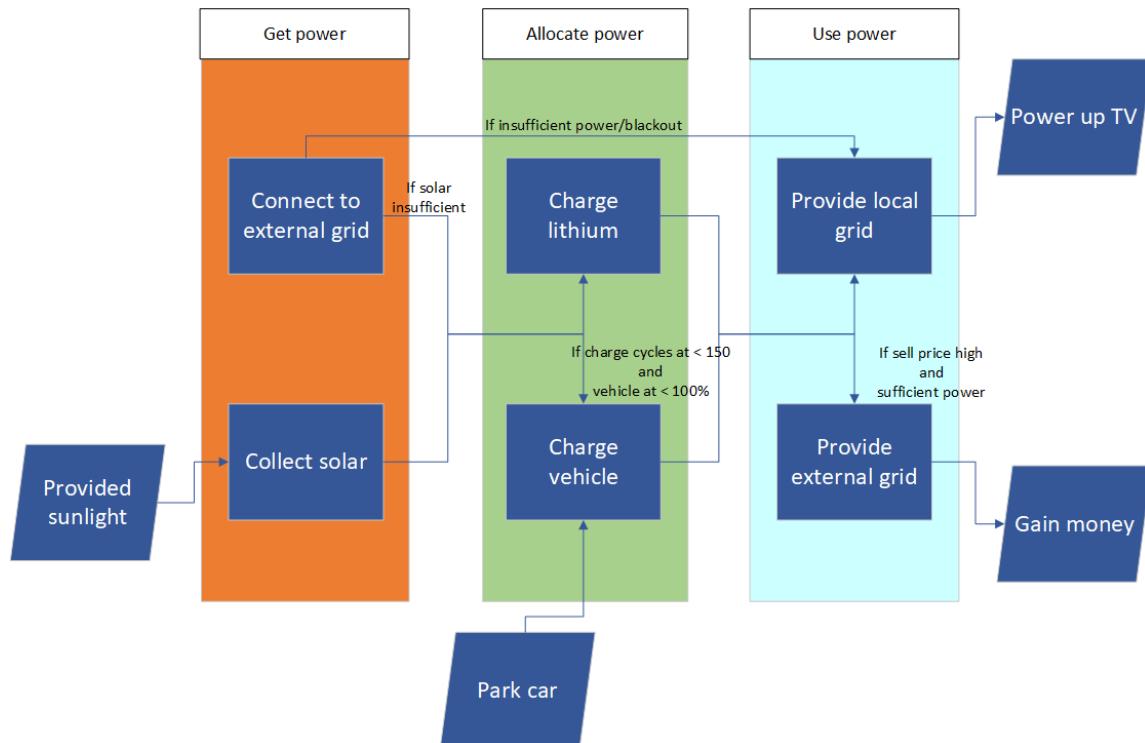


Figure 30: Behavior of the Energy Storage System

Structure

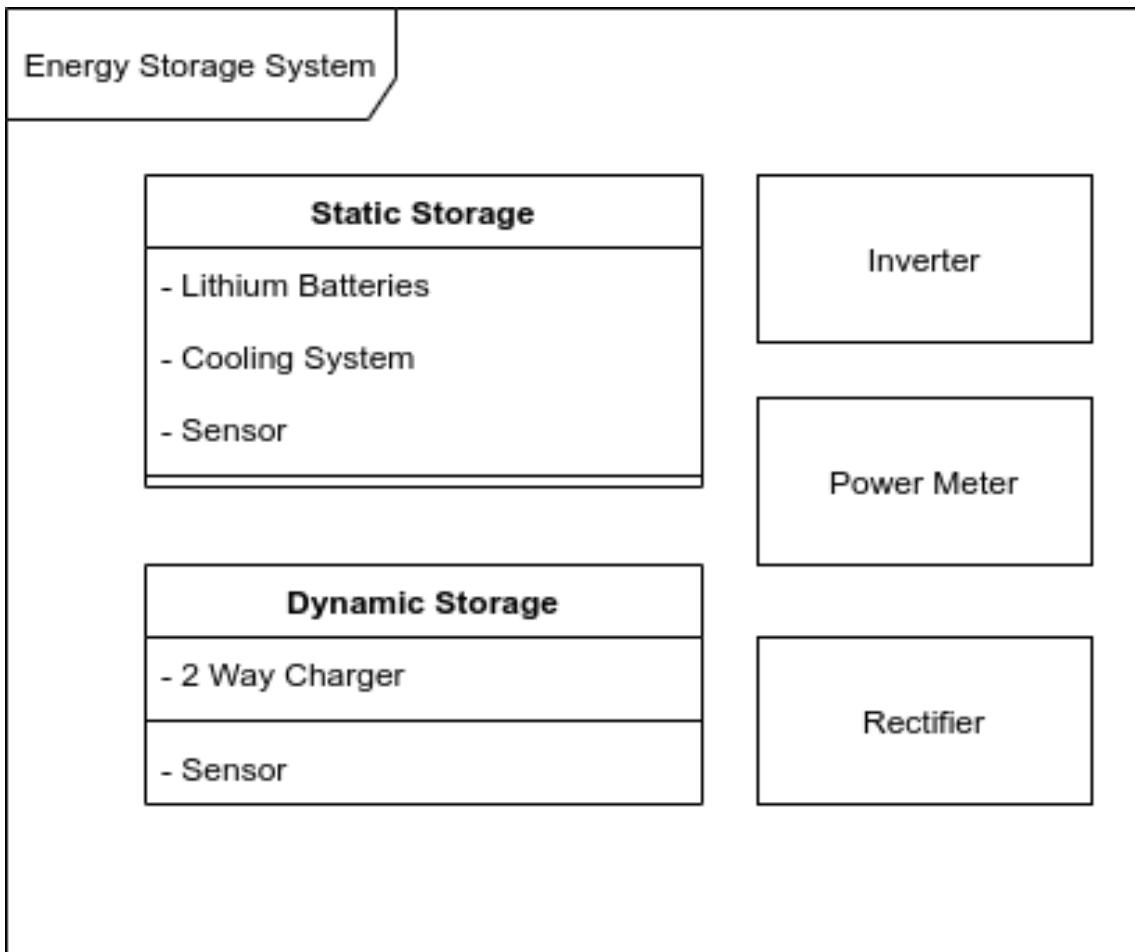


Figure 31: Class diagram of the Energy Storage System

Derived Requirements

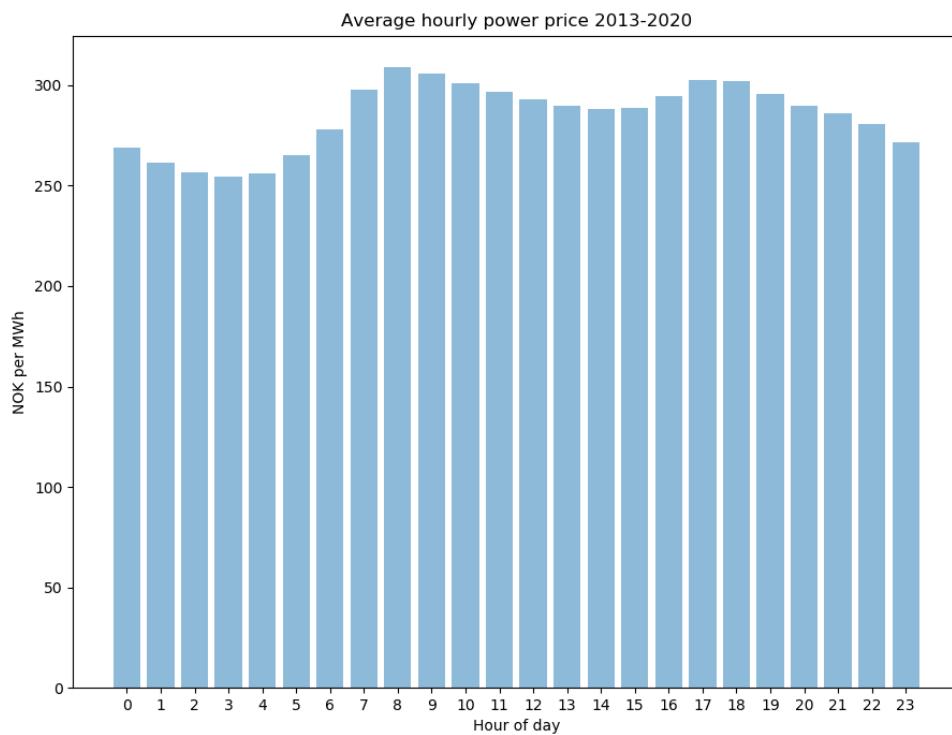


Figure 32: Average hourly power prices 2013-2020

Control System

Operation

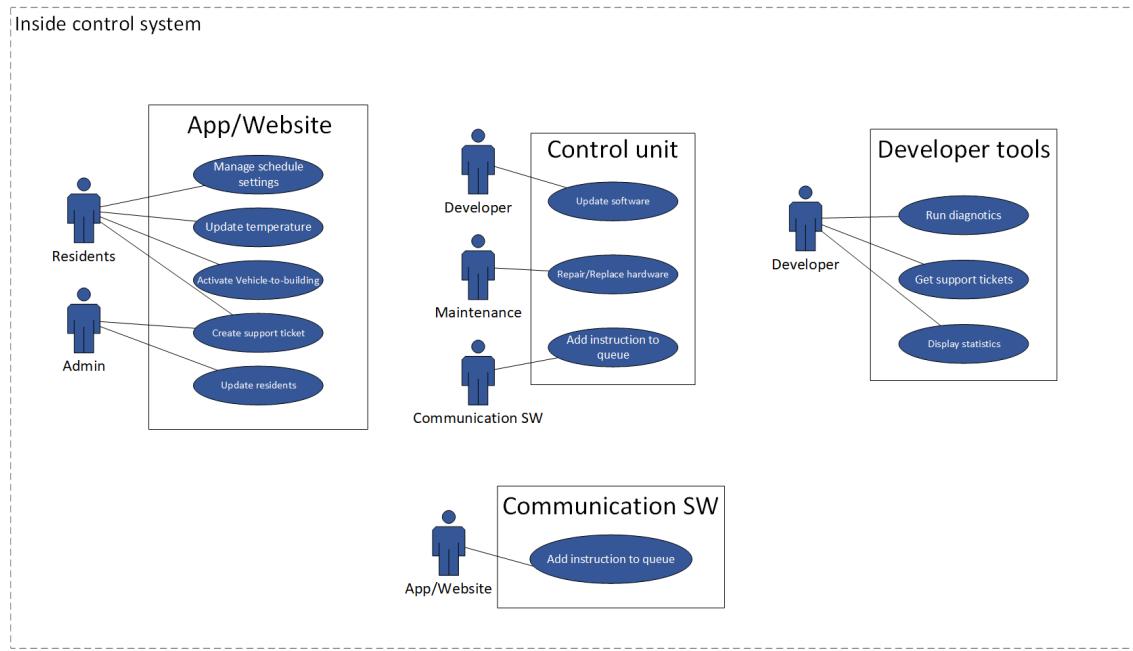


Figure 33: Use case diagram within the control system

Behavior

Control Unit

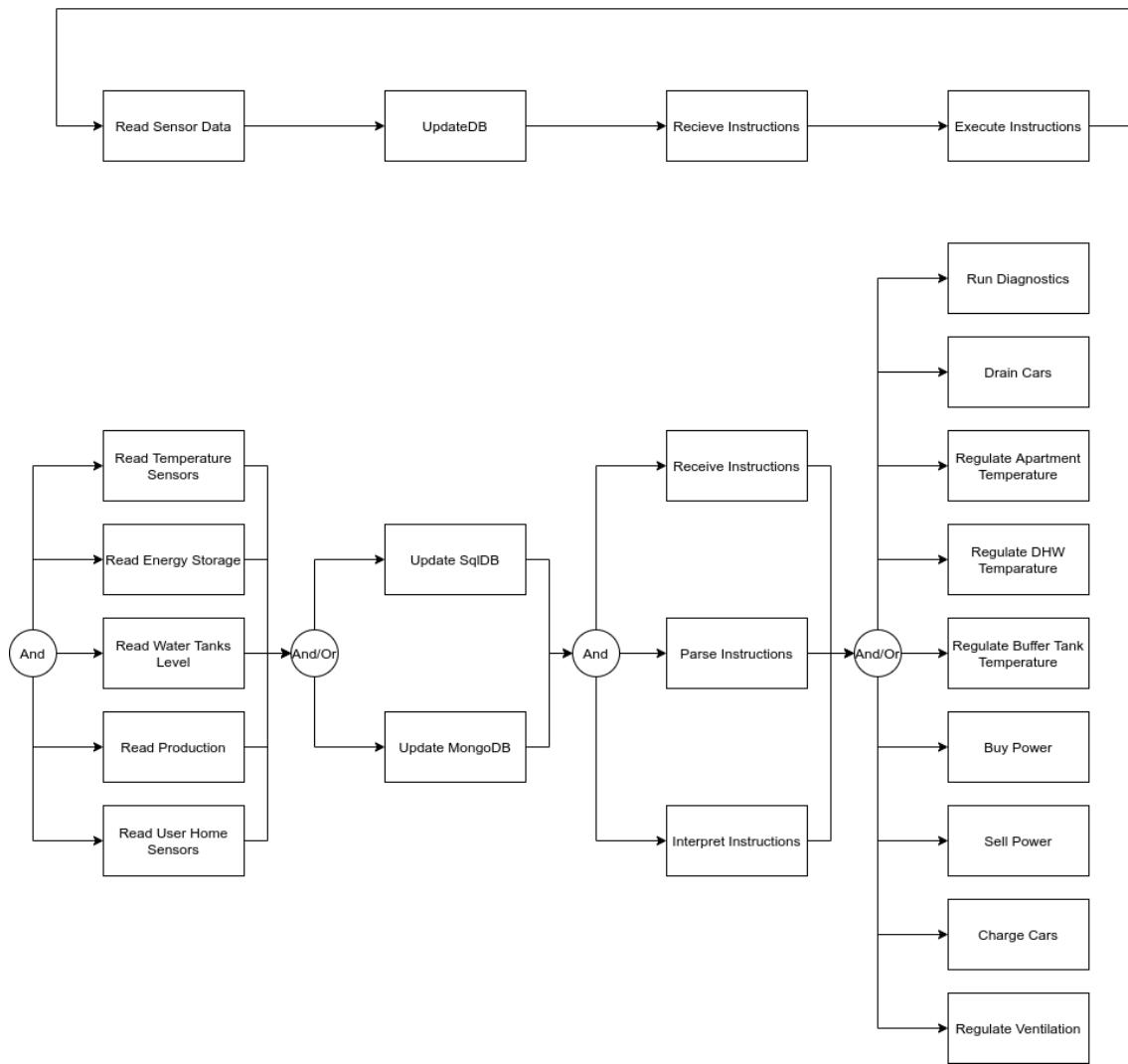


Figure 34: FFBD for the Control Unit

MCU (Main Communication Unit)

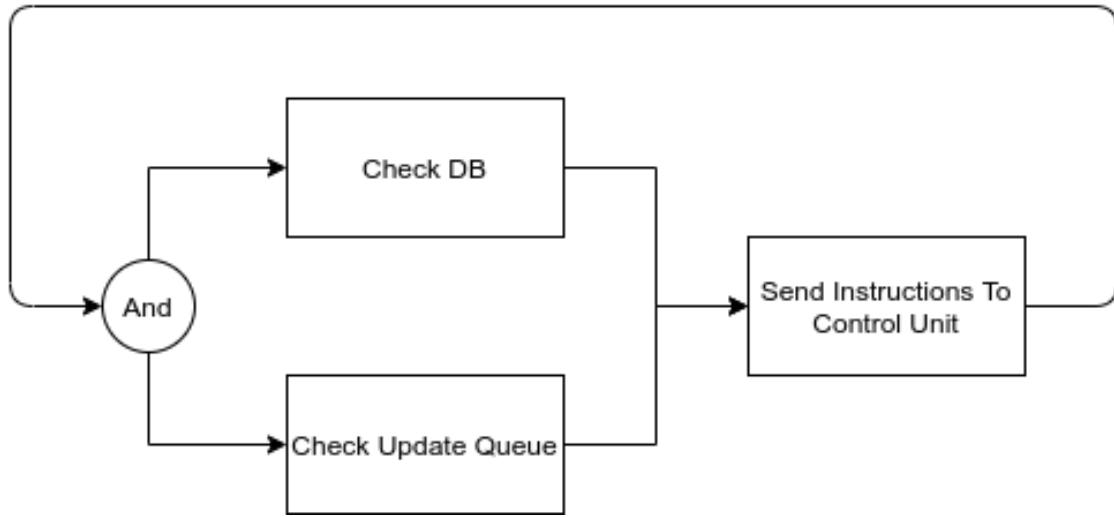


Figure 35: FFBD for the MCU

Databases

MySQL SQL MongoDB

Data-Gathering Software

Historical Data

Continuous Data

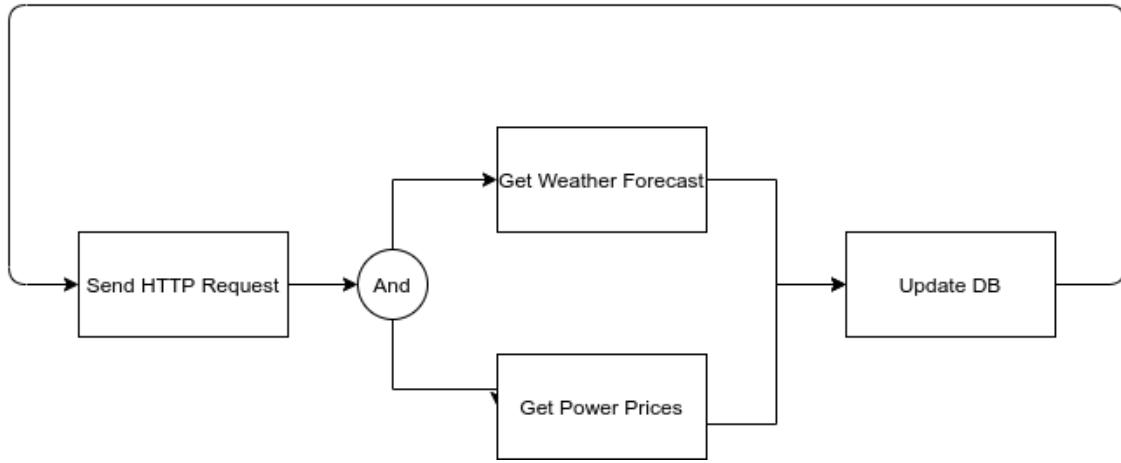


Figure 36: FFBD for the continuous data gathering software

Machine Learning Software

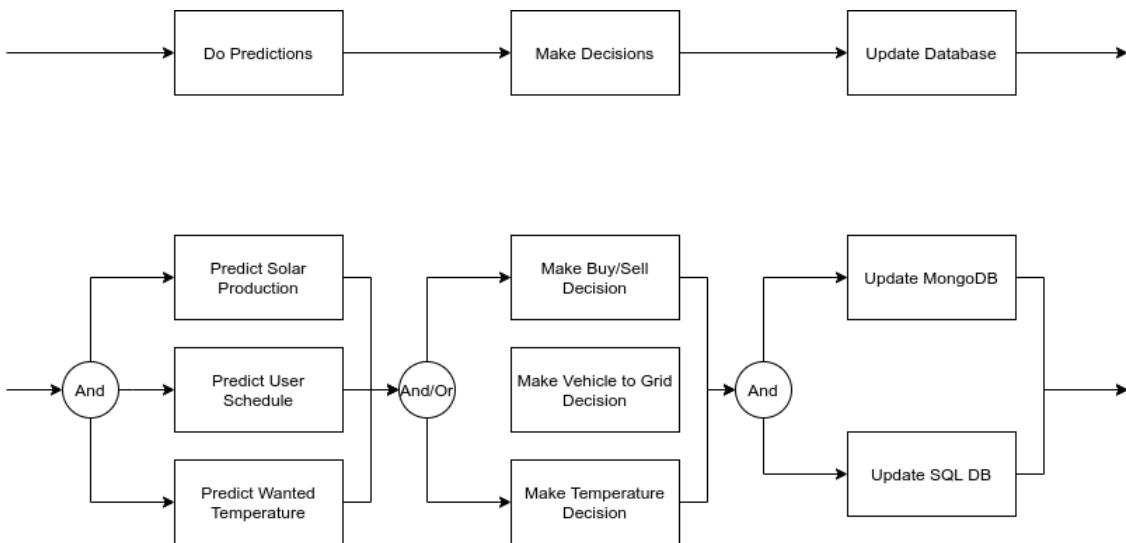


Figure 37: FFBD for the machine learning software

Application / Website

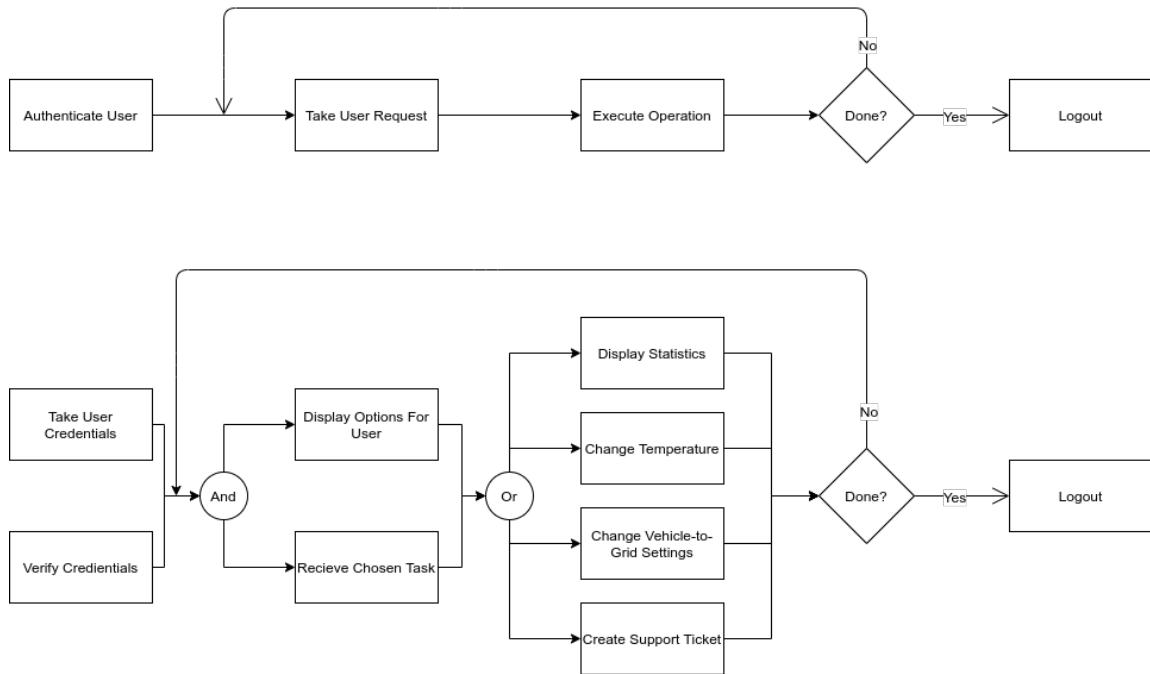


Figure 38: FFBD for the application/website

Developer Tools

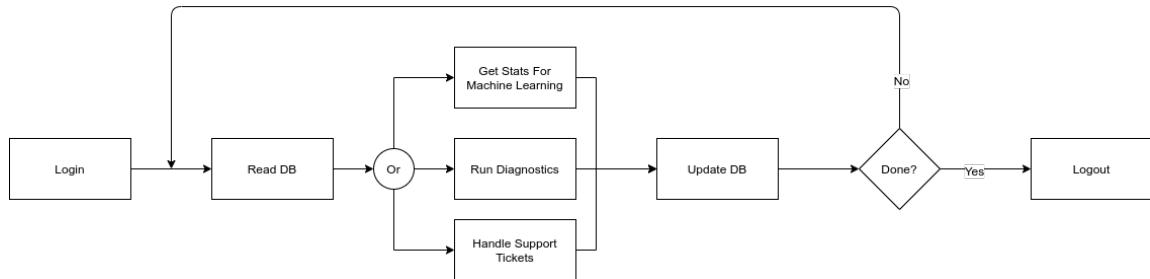


Figure 39: FFBD for the developer tools

Structure

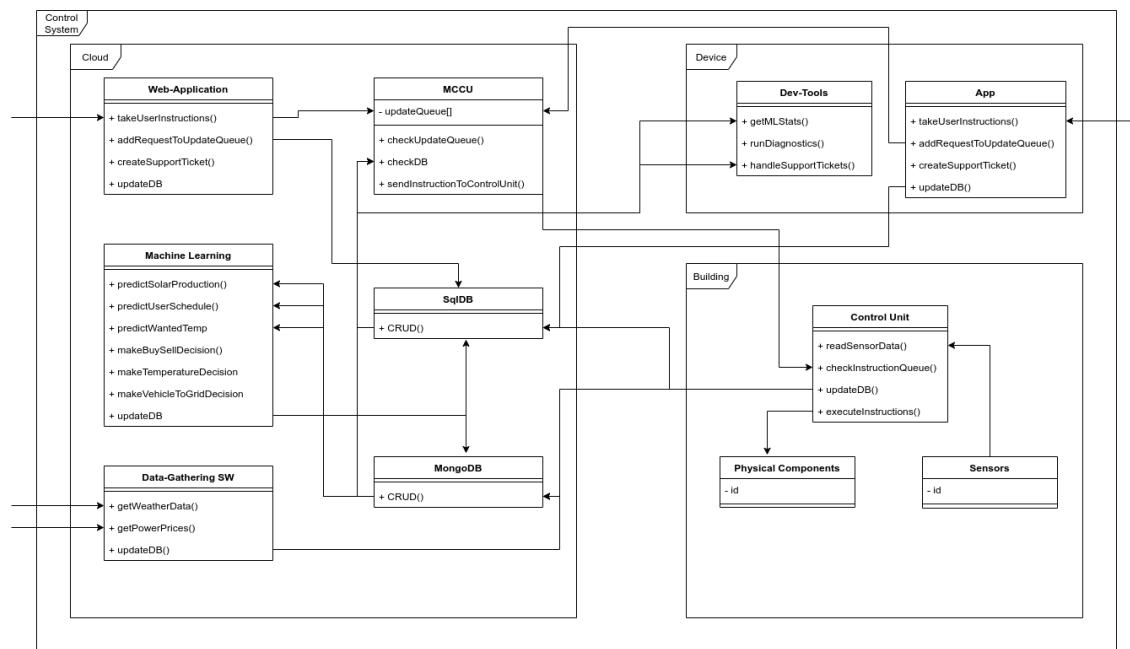


Figure 40: Class diagram for Control System

Derived Requirements

Build and Test Plan

Verification

Production

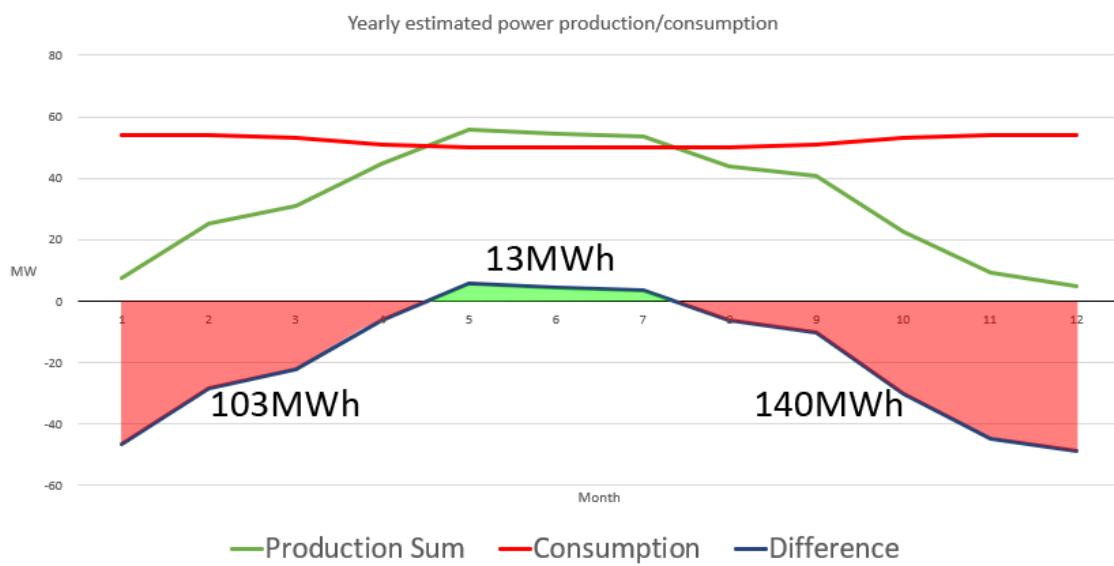


Figure 41: FINN PÅ EN GOD CAPTION TARALD

Capacity for Storage

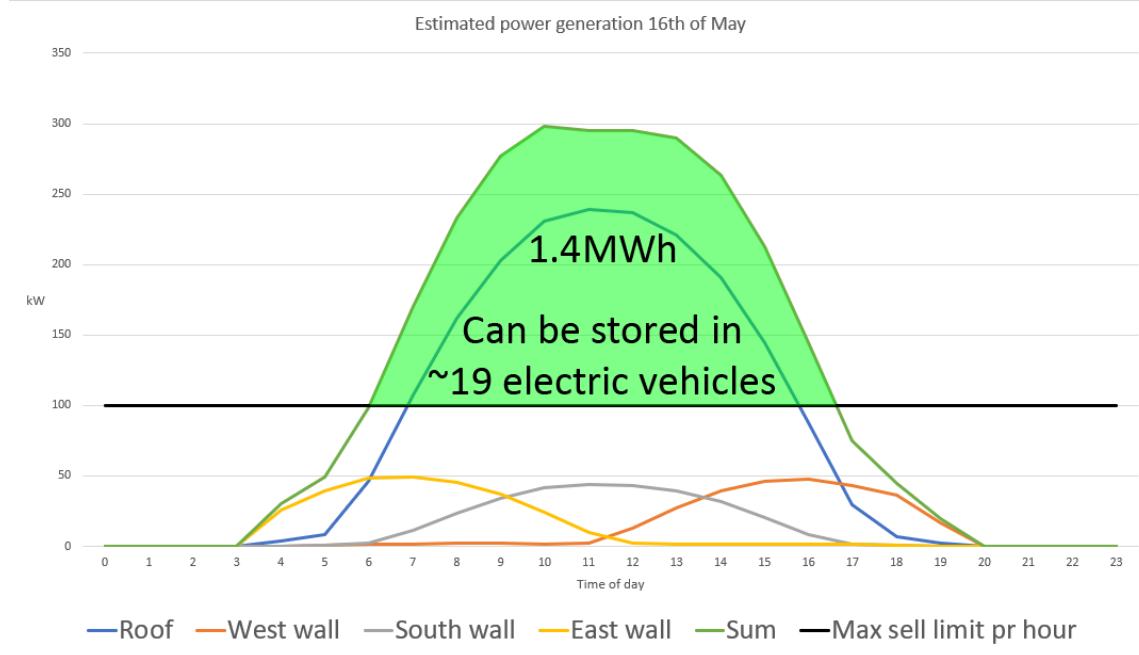


Figure 42: FINN PÅ EN GOD CAPTION TARALD

Prototypes / Proof-of-Concept

Machine Learning

Predicting Solar Panel Power Production

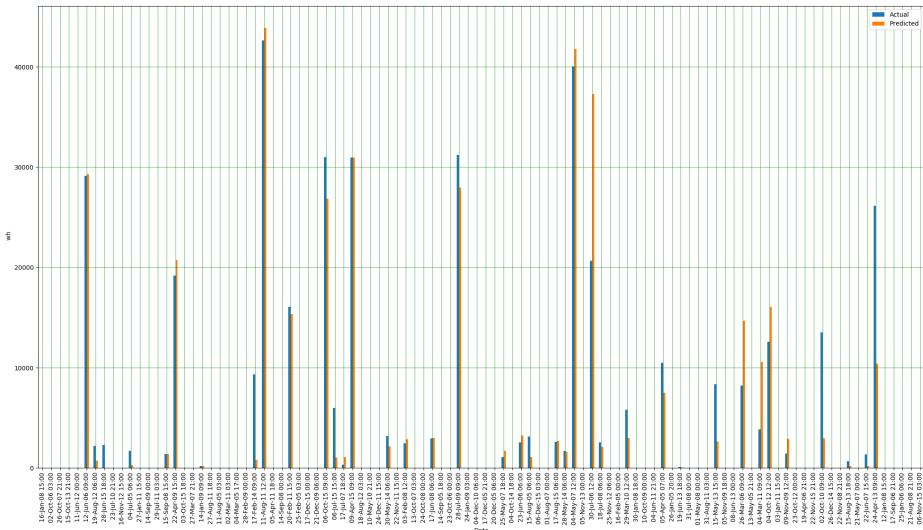


Figure 43: Machine learning prediction results for shuffled solar production

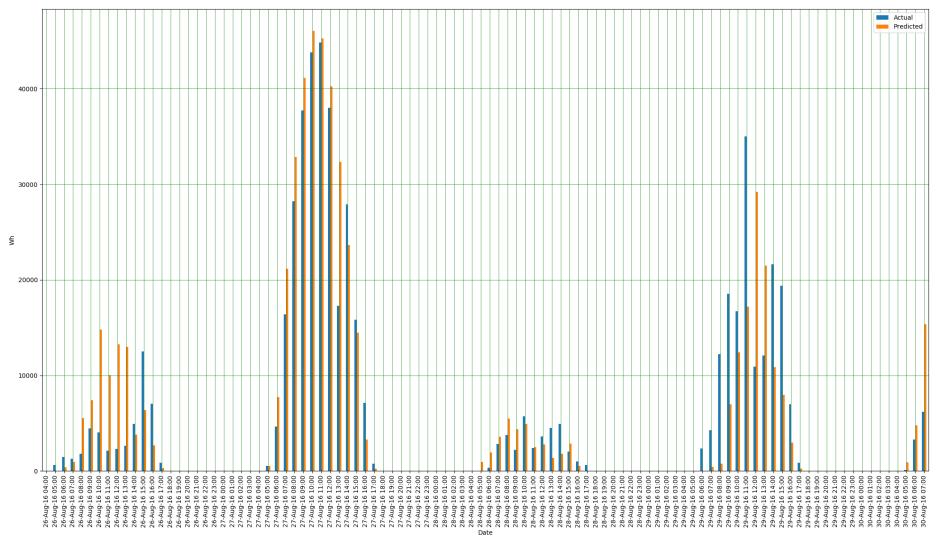


Figure 44: Machine learning prediction results for chronological solar production

Predicting Power Prices

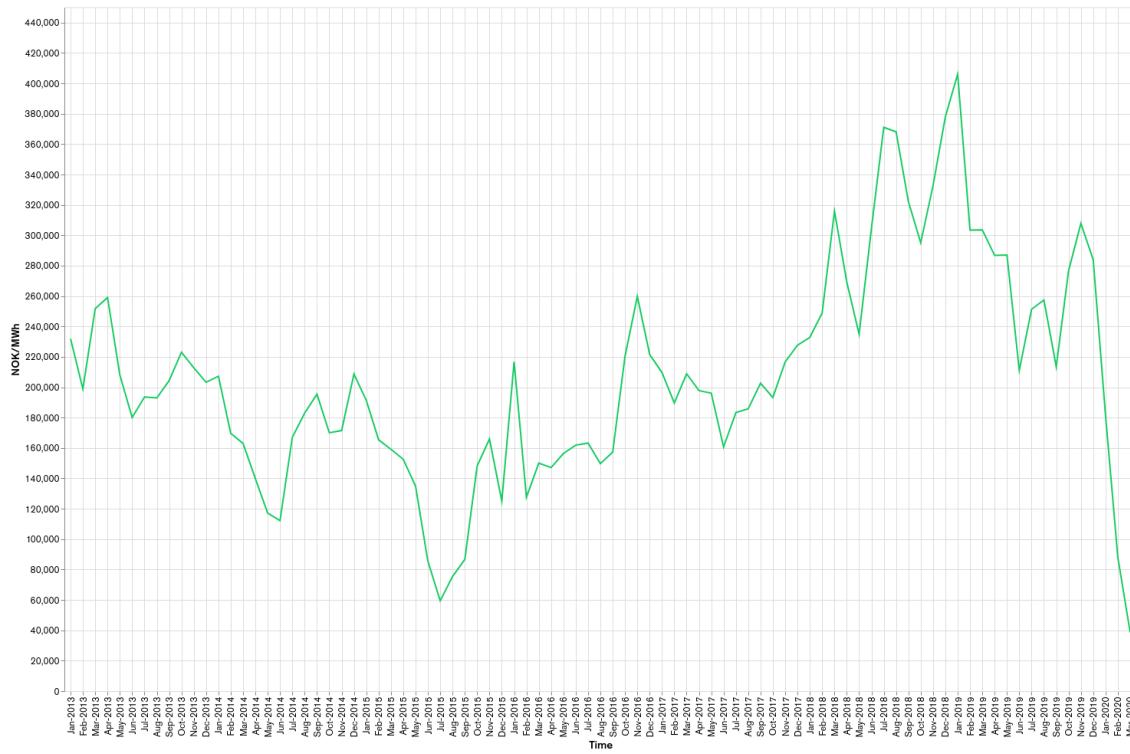


Figure 45: Power prices 2013-2020 in NOK/MWh

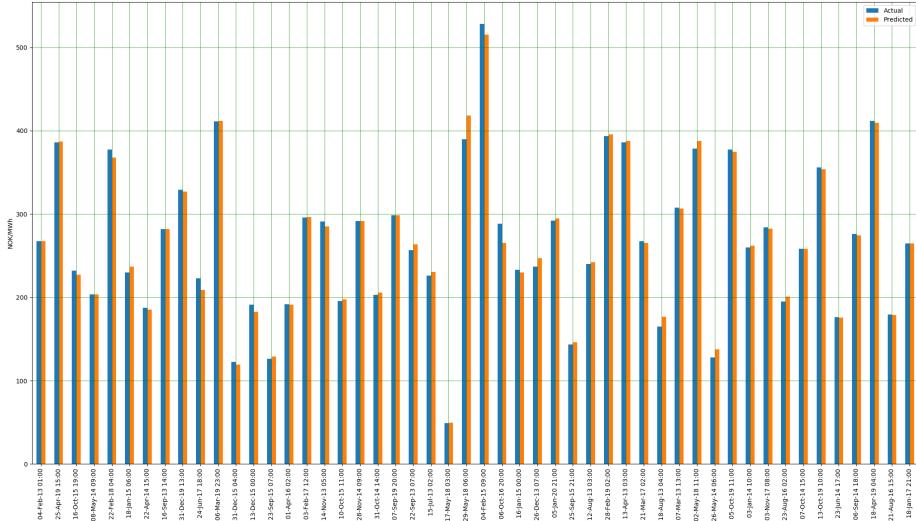


Figure 46: Machine learning prediction results for shuffled power prices

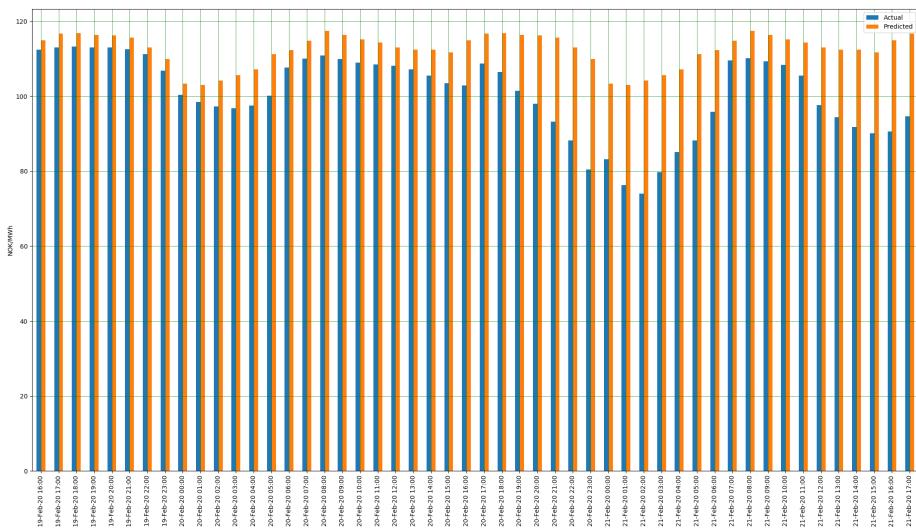


Figure 47: Machine learning prediction results for chronological power prices

Web-Application

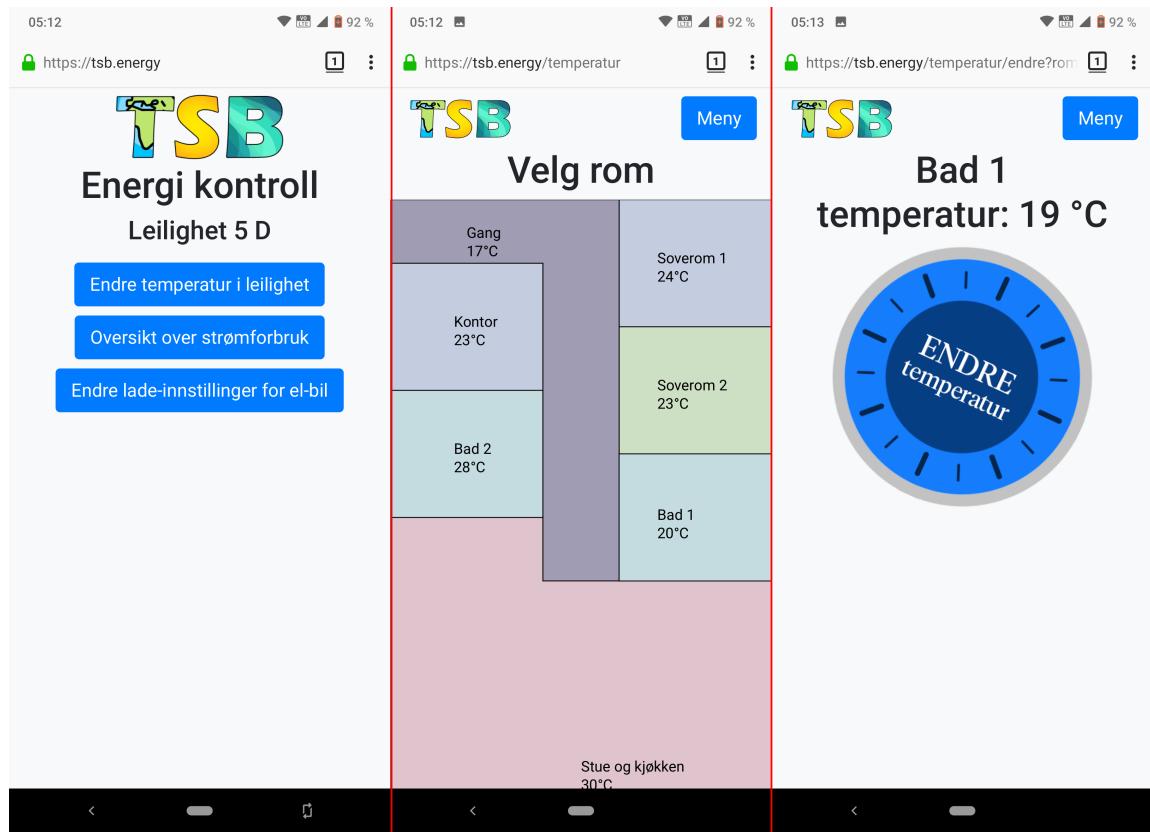


Figure 48: Screenshots of the website prototype

Risk

RISK	AREAS AFFECTED	SEVERITY	LIKELIHOOD	RISK IMPACT	RECOMMENDED ACTION(S)
Covid-19 pandemic	Developers health	LOW	HIGH	MEDIUM	Home office, and no physical meetings
Project goes over budget and delayed	Investor retention	MEDIUM	LOW	MEDIUM	Use proper system engineering method
Low sales	Developers salary	HIGH	LOW	MEDIUM	Customer surveys, make sure our product is wanted before starting development
Construction/installation delays	Customer satisfaction	MEDIUM	LOW	MEDIUM	Promise economic incentive in the contracts, to finish on time

Figure 49: Risk matrix for development

RISK	AREAS AFFECTED	SEVERITY	LIKELIHOOD	RISK IMPACT	RECOMMENDED ACTION(S)
Internet loss	Residents can not control temperature	LOW	MEDIUM	MEDIUM	Control unit has default values to fall back on for temperatures
Sensor failure	Temperature adjustment accuracy	LOW	MEDIUM	MEDIUM	Control unit try to hit set temperature by guessing at correct energy usage from historic data
Software bugs	Control system	LOW	MEDIUM	MEDIUM	Clean code, documentation, testing
Security breach	Control system	HIGH	LOW	MEDIUM	Encrypt or hash personal information

Figure 50: Risk matrix for system operation

Economics

Our Budget

	Year 1		Year 2		Year 3		Year 4		Year 5		Year 6	
	Half 1	Half 2	Half 1	Half 2	Half 1	Half 2	Half 1	Half 2	Half 1	Half 2	Half 1	Half 2
Investments	3 050k	3 050k	0	0	0	0	0	0	0	0	0	0
Sales												
Volume	0	0	1	1	2	2	3	3	4	4	4	4
Cumulative	0	0	1	2	4	6	9	12	16	20	24	28
Income												
Unit sales	0	0	10 000k	10 000k	20 000k	20 000k	30 000k	30 000k	40 000k	40 000k	40 000k	40 000k
Unit service	0	0	50k	100k	200k	300k	450k	600k	800k	1 000k	1 200k	1 400k
Expenses												
Salary	-3 000k	-3 000k	-3 000k	-3 000k	-4 000k	-4 000k	-4 000k	-4 000k	-5 000k	-5 000k	-5 000k	-5 000k
Other	-50k	-50k	-50k	-50k	-100k	-100k	-100k	-100k	-120k	-120k	-120k	-120k
Service												
Labour	0	0	-20k	-40k	-80k	-120k	-180k	-240k	-320k	-400k	-480k	-560k
Sales												
Material	0	0	-7 590k	-7 590k	-15 180k	-15 180k	-22 770k	-22 770k	-30 360k	-30 360k	-30 360k	-30 360k
Labour	0	0	-100k	-100k	-200k	-200k	-300k	-300k	-400k	-400k	-400k	-400k
Profit												
Half year	-3 050k	-3 050k	-710k	-680k	640k	700k	3 100k	3 190k	4 600k	4 720k	4 840k	4 960k
Cumulative	-3 050k	-6 100k	-6 810k	-7 490k	-6 850k	-6 150k	-3 050k	140k	4 740k	9 460k	14 300k	19 260k

Figure 51: Budget for our company

Our Cash Flow

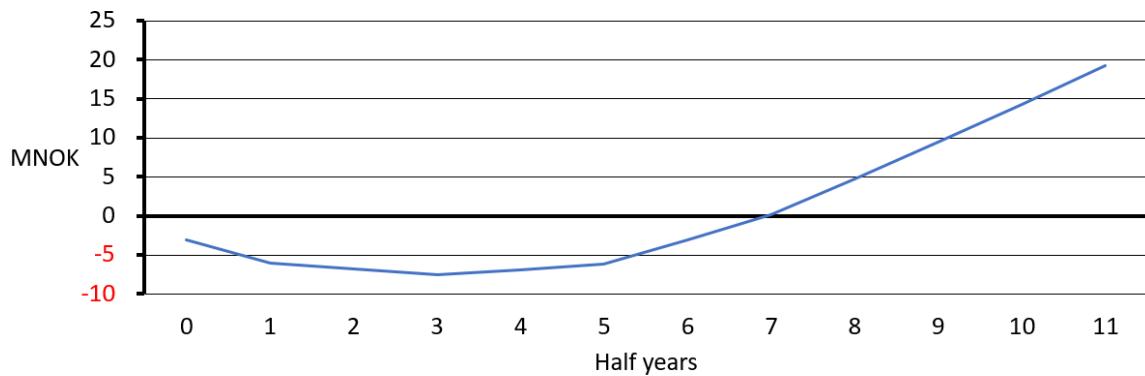


Figure 52: Cash flow graph for our company

Residents Budget

	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
System purchase	-13 000k	0	0	0	0	-1 350k	0	0	0	0
Expense without system										
Power	-253k	-296k	-338k	-380k	-422k	-465k	-507k	-549k	-591k	-634k
System expenses	-4 000k	0	0	0	0	0	0	0	0	0
Cumulative	-4 253k	-5 098k	-5 774k	-6 534k	-7 379k	-8 308k	-9 322k	-10 420k	-11 603k	-12 870k
Costs with system										
Clean solar panels	-104k									
Service-agreement	-50k									
Income with system										
Surplus power sale	18k	20k	23k	26k	29k	32k	35k	38k	41k	44k
Profit										
Year	-13 136k	-134k	-131k	-128k	-125k	-1 472k	-119k	-116k	-113k	-110k
Cumulative	-13 136k	-13 270k	-13 401k	-13 528k	-13 653k	-15 125k	-15 244k	-15 360k	-15 473k	-15 583k

Figure 53: Budget for customer

Residents Cash Flow

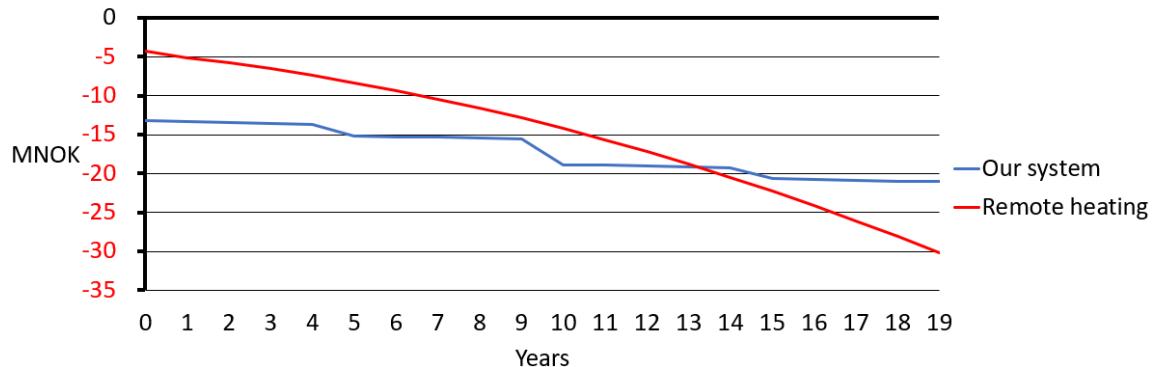


Figure 54: Cash flow graph for customer

Conclusion

References

- [1] SSB Norwegian household power consumption, ssb. <https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/vi-bruker-mindre-strom-hjemme>. Accessed: March-25.

Appendices

Figures



Figure 55: Use Case Diagram Example

Code

Historical Weather Web-Crawler

```
%LIM INN KODEN DIN HER TARALD
```

MongoDB Filler

```
#include <iostream>
#include <fstream>
#include <string>
#include <bsoncxx/builder/stream/document.hpp>
#include <bsoncxx/json.hpp>
#include <mongocxx/client.hpp>
#include <mongocxx/instance.hpp>
#include "boost/property_tree/ptree.hpp"
#include "boost/property_tree/json_parser.hpp"
#include "boost/date_time posix_time/posix_time.hpp"
#include <boost/foreach.hpp>
#include <thread>
#include <vector>
#include <functional>
#include <algorithm>
#include "power.h"
#include <boost/algorithm/string/replace.hpp>
```

```

using boost::property_tree::ptree;
using boost::posix_time::ptime;
using boost::posix_time::from_iso_string;

bool fillSunWall(std::string, mongocxx::client&);
bool fillWeather(mongocxx::client&);
bool parseSun();
bool fillPowerPrices(mongocxx::client&);

std::string powerPath = "data/PowerData/El";
std::string powerExt = ".csv";

std::string weatherPath = "data/WeatherData/data2.json";
std::string sunPath = "data/SolarData/";
std::string jsonExt = ".json";

int main(int, char**)
{
    mongocxx::instance inst{};
    mongocxx::client conn{
        mongocxx::uri{
            "mongodb+srv://kent:<Password>@terrasolaris-robmp.azure.
            mongodb.net/test?retryWrites=true&w=majority"}};

    fillSunWall("roof", conn);
    fillSunWall("south", conn);
    fillSunWall("west", conn);
    fillSunWall("east", conn);
    fillWeather(conn);
    fillPowerPrices(conn);

    return 0;
}

bool fillSunWall(std::string direction, mongocxx::client& conn)
{
    for(int i = 1; i <= 3; i++)
    {
        std::cout << "Document: " << direction + std::to_string(i) << "\n";
        std::ifstream ist(sunPath + direction + std::to_string(i) + jsonExt);
        ptree pt;
        read_json(ist, pt);

        pt = pt.get_child("outputs").get_child("hourly");

        for (auto & array_element : pt) {

            std::string time = array_element.second.get<std::string>("time");

```

```

        time.replace(8, 1, "T");
        ptime t = from_iso_string(time);
        double p_val = array_element.second.get<double>("P");

        bsoncxx::builder::stream::document document{};
        auto builder = bsoncxx::builder::stream::document{};
        bsoncxx::document::value doc_value = builder
            << "time" << to_extended_string(t)
            << "p" << p_val
            << bsoncxx::builder::stream::finalize;
        auto collection = conn["Terra"]["sun" + direction];
        collection.insert_one(doc_value.view());
    }
}

return true;
}

void fillData(boost::property_tree::ptree &pt, std::string time, mongocxx::client& conn)
{
    bsoncxx::builder::stream::document doc{};
    time = time.substr(0, 16);
    doc << "time" << time;

    for (auto & array_element : pt) {

        std::string elementId = array_element.second.get<std::string>("elementId");
        double value = array_element.second.get<double>("value");
        doc << elementId << array_element.second.get<double>("value");
    }
}

auto collection = conn["Terra"]["weather"];
collection.insert_one(doc.view());
}

void fillDoc(boost::property_tree::ptree &pt, std::string time, mongocxx::client& conn)
{
    if(pt.empty())
    {
        return;
    }
    else
    {
        for (boost::property_tree::ptree::iterator pos = pt.begin(); pos != pt.end();)
        {
            if(pos->first == "referenceTime")
            {
                time = pos->second.data();
            }
            else if(pos->first == "observations")

```

```

        {
            fillData(pos->second, time, conn);
        }
        fillDoc(pos->second, time, conn);
        ++pos;
    }
}

bool fillWeather(mongocxx::client& conn)
{
    std::ifstream ist(weatherPath);
    ptree pt;
    read_json(ist, pt);
    bsoncxx::builder::stream::document doc{};
    std::vector<std::thread*> threads;

    fillDoc(pt, "", conn);

    return true;
}

bool fillPowerPrices(mongocxx::client& conn)
{
    const int dateIndex = 0;
    const int hourIndex = 1;
    const int osloIndex = 10;

    for(int year = 2013; year <= 2020; year++)
    {
        std::cout << "Year: " << year << "\n";
        std::ifstream file;
        file.open(powerPath + std::to_string(year) + powerExt);
        auto table = readCSV(file);
        for(int i = 3; i < table.size(); i++)
        {
            std::string date = table[i][dateIndex];
            std::string hour = table[i][hourIndex];
            std::string price = (table[i][osloIndex]);
            boost::replace_all(price, ",", ".");
            if(price.empty())
            {
                continue;
            }
            double priceNum = std::stod(price);
            std::string time = date.substr(6, 4) + "-" + date.substr(3, 2) + "-" +
                date.substr(0, 2) + "T" + hour.substr(0, 2) + ":00:00";
        }
    }
}

```

```

bsoncxx::builder::stream::document document{};
auto builder = bsoncxx::builder::stream::document{};
bsoncxx::document::value doc_value = builder
    << "time" << time
    << "price" << priceNum
    << bsoncxx::builder::stream::finalize;
auto collection = conn["Terra"]["powerPrices"];
collection.insert_one(doc_value.view());
}

}

return true;
}

```

Historical Hourly Avg. Power Price

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
import datetime as dt
import pymongo
from mongoConfigFile import *

hourPrices = [0] * 24
hourPoints = [0] * 24
hours = list(range(0, 24))

clientUrl =
    ("mongodb+srv://{}:{}@terralsolaris-robmp.azure.mongodb.net/test?retryWrites=true&w=majority".
     format(mongoUsername, mongoPassword))
client = pymongo.MongoClient(clientUrl)
db = client["Terra"]

def main():
    fillData()
    drawPower()

def fillData():
    print("Fyller priser..")
    collection = db["powerPrices"]
    #cursor = collection.find({"time": "2005-01-01T00:00:00"})
    cursor = collection.find({})
    for document in cursor:

```

```

date = pd.to_datetime(document["time"])
hourPrices[int(date.hour)] += document["price"]
hourPoints[int(date.hour)] += 1

def drawPower():
    print("drawing")
    X = []
    for i in range(0, len(hours)):
        X.append(hourPrices[i] / hourPoints[i])

    y = hours

    objects = hours
    y_pos = np.arange(len(hours))
    performance = X

    plt.bar(y_pos, performance, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('NOK per MWh')
    plt.xlabel('Hour of day')
    plt.title('Average hourly power price 2013-2020')

    plt.show()

main()

```

Machine Learning

Solar Power Production

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
import datetime as dt
import pymongo
from mongoConfigFile import *

power = []
dates = []
weather = {}
data = {'Year': [], 'Month': [], 'Day': [], 'Hour': [], 'Temp': [],
'CloudAreaFraction': [], 'CloudBaseHeight': [], 'WeatherType': [], 'Power': []}

```

```

clientUrl =
    ("mongodb+srv://{}:{}@terralsolaris-robmp.azure.mongodb.net/test?retryWrites=true&w=majority".
     format(mongoUsername, mongoPassword))
client = pymongo.MongoClient(clientUrl)
db = client["Terra"]

def main():
    fillData()
    fillWeather()
    dataset = fillPanda()
    betterPredictWithWeather(dataset)

def fillData():
    print("Fyller sol..")
    collection = db["sunsouth"]
    cursor = collection.find({})
    for document in cursor:
        dates.append(document["time"])
        power.append(document["p"])

def fillWeather():
    print("Filling weather..")
    collection = db['weather']
    cursor = collection.find({})
    for document in cursor:
        arr = []
        arr.append(document.get('air_temperature'))
        arr.append(document.get('cloud_area_fraction'))
        arr.append(document.get('cloud_base_height'))
        arr.append(document.get('weather_type'))
        weather[document['time'] + ':00'] = arr

def fillPanda():
    print('Fyller Panda..')
    db = client["Terra"]
    collection = db["weather"]

    for date in dates:
        l = weather.get(date)
        date = pd.to_datetime(date)
        data['Year'].append(int(date.year))
        data['Month'].append(int(date.month))
        data['Day'].append(int(date.day))
        data['Hour'].append(int(date.hour))
        default = None
        data['Temp'].append(l[0] if l is not None and 0 < len(l) else default)
        data['CloudAreaFraction'].append(l[1] if l is not None and 1 < len(l) else default)
        data['CloudBaseHeight'].append(l[2] if l is not None and 2 < len(l) else default)
        data['WeatherType'].append(l[3] if l is not None and 3 < len(l) else default)

```

```

data['Power'] = power
df = pd.DataFrame(data=data)
df.fillna(df.mean(), inplace=True)
df = df.reset_index()
return df

def betterPredictWithWeather(dataset):
    X = dataset[['Year', 'Month', 'Day', 'Hour', 'Temp', 'CloudAreaFraction',
                 'CloudBaseHeight', 'WeatherType']].values
    y = dataset['Power'].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01, shuffle=False)
    regressor = DecisionTreeRegressor(random_state = 0)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)

    print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))

    X_dates = []
    for date in X_test:
        X_dates.append(dt.datetime(int(date[0]), int(date[1]), int(date[2]), int(date[3])))

    df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}, index = X_dates)

    df1 = df.head(100)
    ax = df1.plot(kind='bar', figsize=(200,100))
    plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
    plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

    x_labels = df1.index.strftime('%d-%b-%y %H:00')
    ax.set_xticklabels(x_labels)
    ax.set_xlabel('Date')
    ax.set_ylabel('Wh')
    plt.show()

main()

```

Machine Learning, Power Price Prediction

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics

```

```

import datetime as dt
import pymongo
from mongoConfigFile import *

powerPrices = []
dates = []
data = {'Year': [], 'Month': [], 'Day': [], 'Hour': [], 'PowerPrice': []}
clientUrl =
    ("mongodb+srv://{}:{}@terrasolaris-robmp.azure.mongodb.net/test?retryWrites=true&w=majority".
     format(mongoUsername, mongoPassword))
client = pymongo.MongoClient(clientUrl)
db = client["Terra"]

def main():
    fillData()
    dataset = fillPanda()
    betterPredict(dataset)

def fillData():
    print("Fyller priser..")
    collection = db["powerPrices"]
    #cursor = collection.find({"time": "2005-01-01T00:00:00"})
    cursor = collection.find({})
    for document in cursor:
        dates.append(document["time"])
        powerPrices.append(document["price"])

def fillPanda():
    print('Fyller Panda..')

    for date in dates:
        date = pd.to_datetime(date)
        data['Year'].append(int(date.year))
        data['Month'].append(int(date.month))
        data['Day'].append(int(date.day))
        data['Hour'].append(int(date.hour))

    data['PowerPrice'] = powerPrices
    df = pd.DataFrame(data=data)
    df.fillna(value=pd.np.nan, inplace=True)
    df = df.reset_index()
    return df

def betterPredict(dataset):
    X = dataset[['Year', 'Month', 'Day', 'Hour']].values
    y = dataset['PowerPrice'].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01,
                                                       shuffle=False) #, random_state=0)

```

```

regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

X_dates = []
for date in X_test:
    X_dates.append(dt.datetime(date[0], date[1], date[2], date[3]))

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}, index = X_dates)

df1 = df.head(50)
ax = df1.plot(kind='bar', figsize=(200,100))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

x_labels = df1.index.strftime('%d-%b-%y %H:00')
ax.set_xticklabels(x_labels)
ax.set_xlabel('Date')
ax.set_ylabel('NOK/MWh')
plt.show()

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))

main()

```

Web-Application

%LIM INN KODEN DIN HER TARALD
