Final project report

MPSE2201 System design and engineering

Group 10

April 9, 2020

Abstract

Contents

List of Figures

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

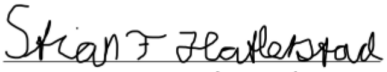
Terra Solaris Borealis

- 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

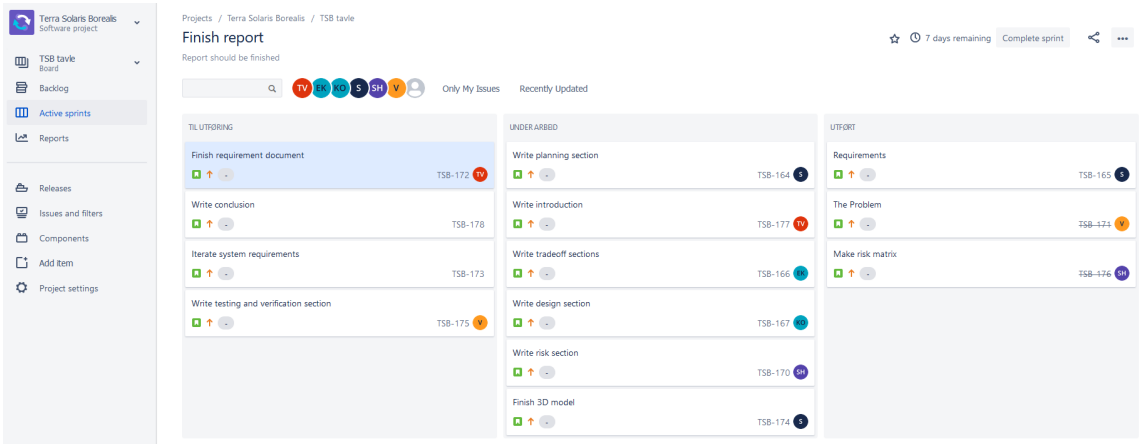


Figure 1: Jira screenshot

Microsoft Teams

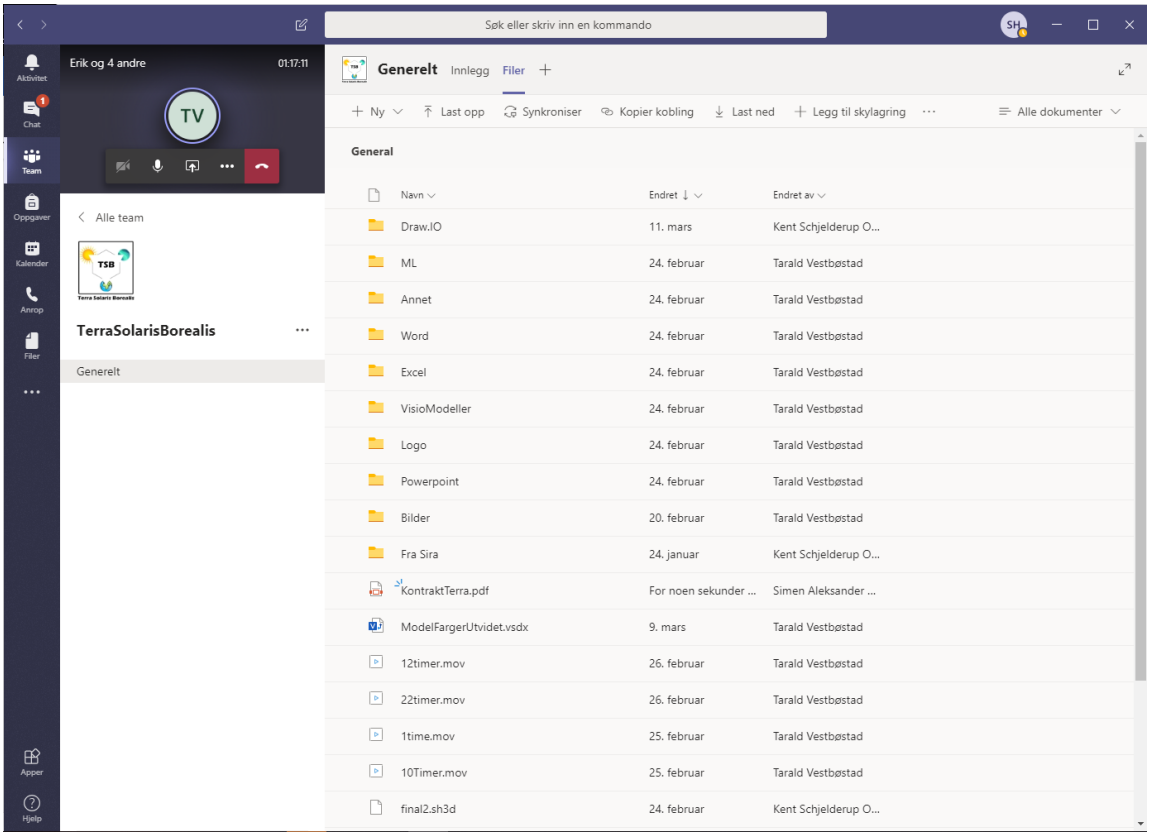


Figure 2: Microsoft Teams screenshot

Facebook Messenger

Microsoft Visio

Microsoft Word

Microsoft Excel

Adobe Illustrator

Gantt

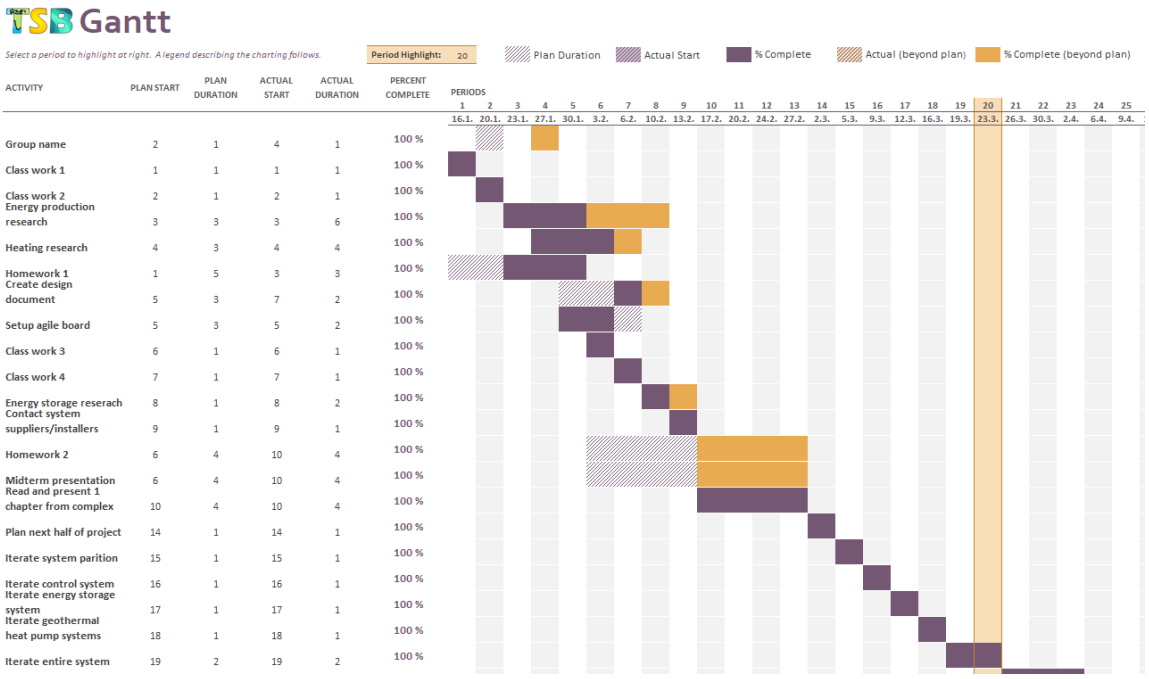


Figure 3: Gantt diagram screenshot

Draw.io

Sweet Home 3D

Solidworks

Programming Languages

Github

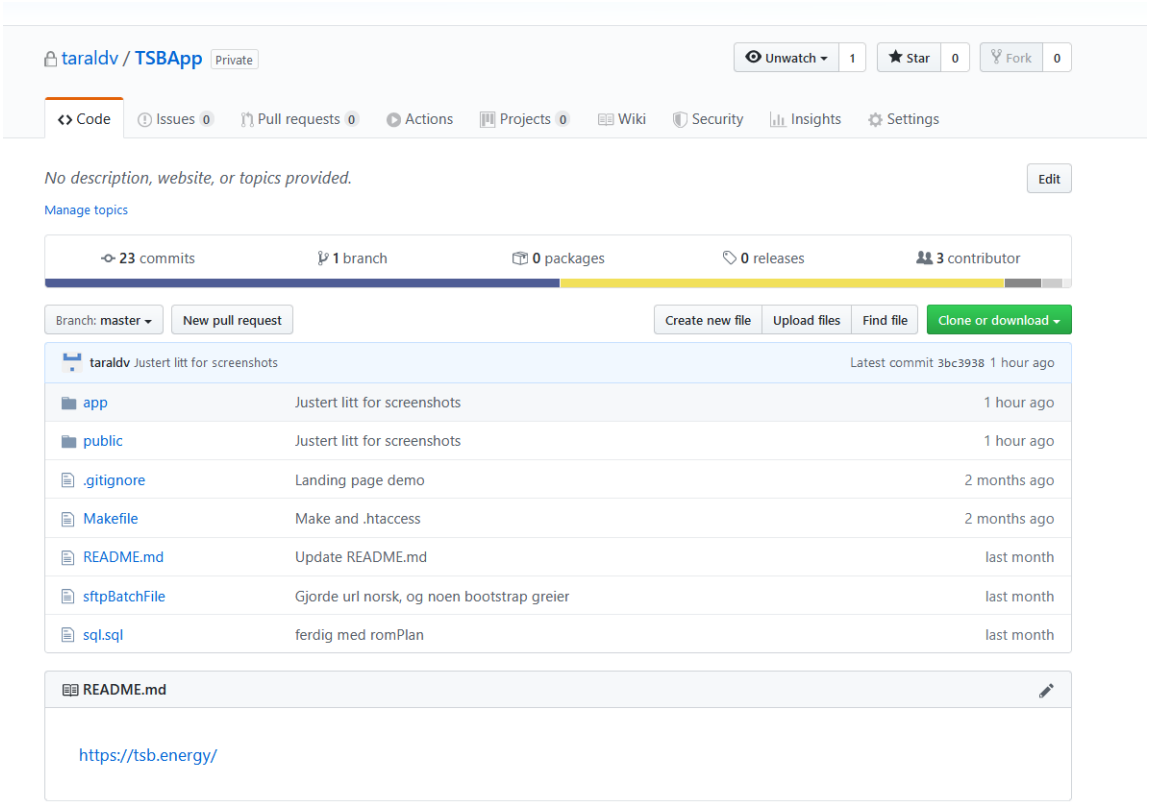


Figure 4: Github web-application repo screenshot

MySQL

MongoDB

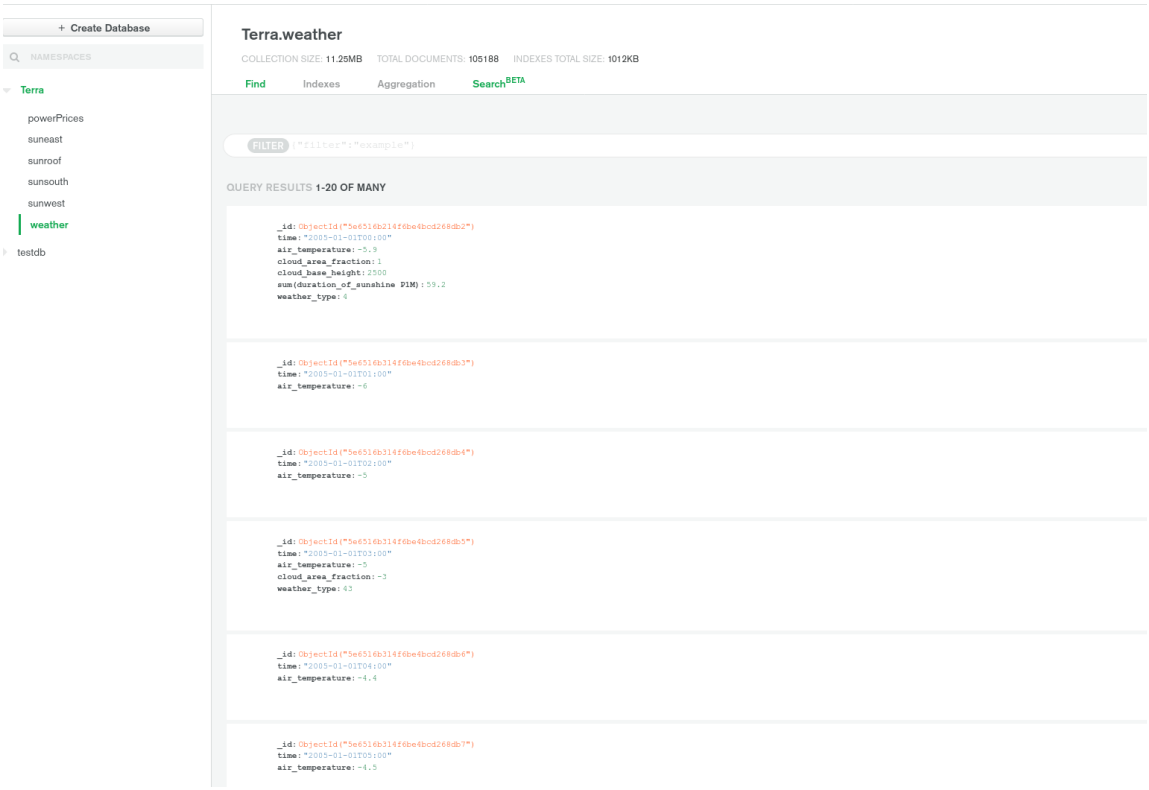


Figure 5: MongoDB screenshot

Internal Discussions

The Problem

Understanding the Problem

Context

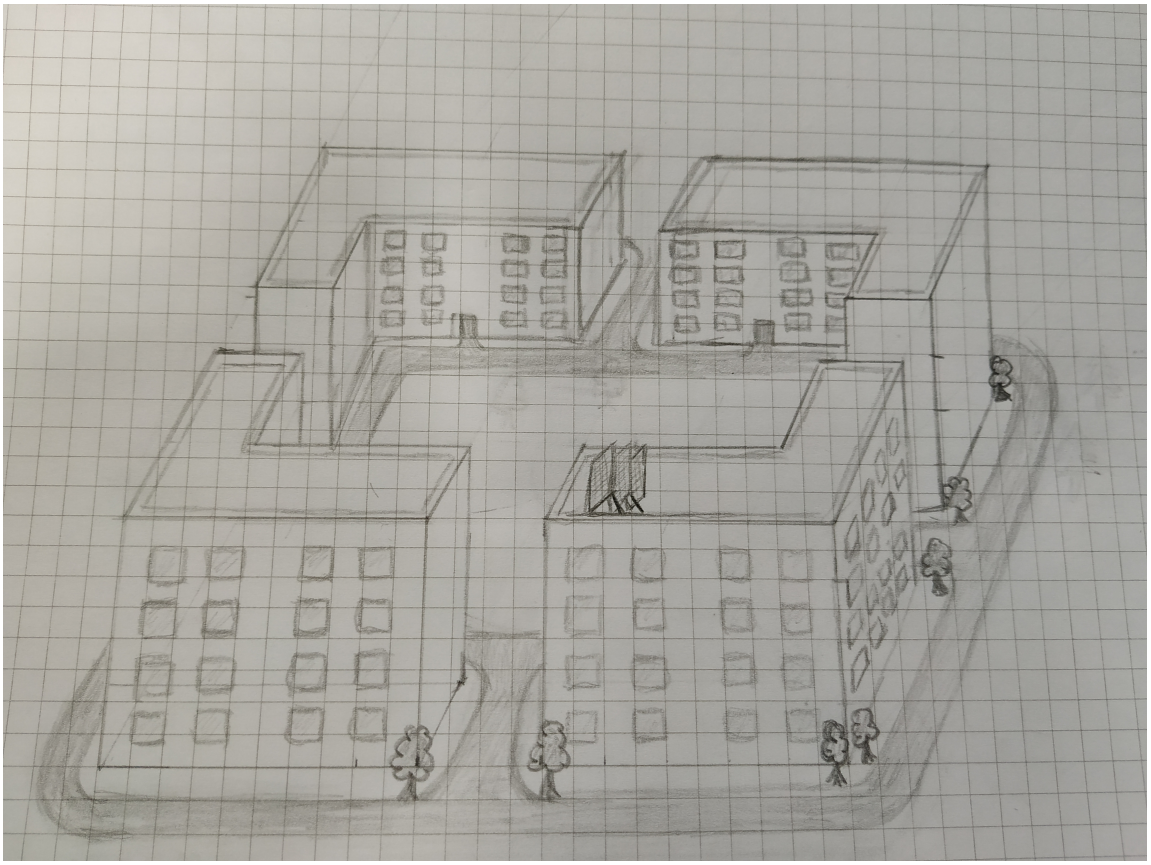


Figure 6: The initial sketch of the context

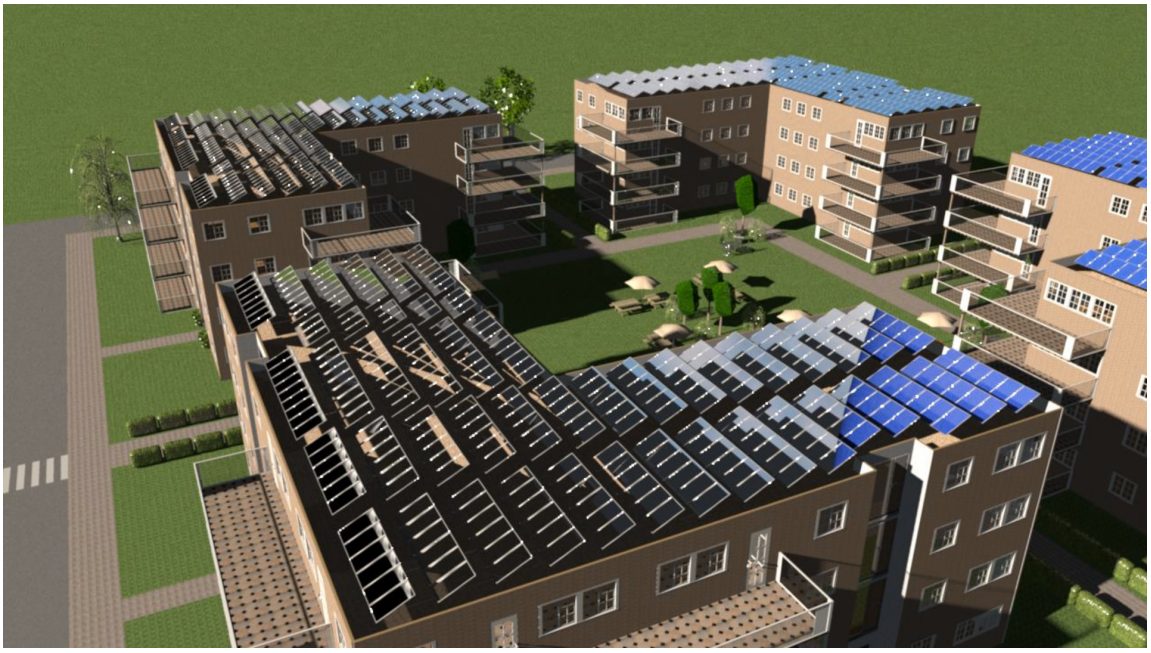


Figure 7: A 3D model of the building

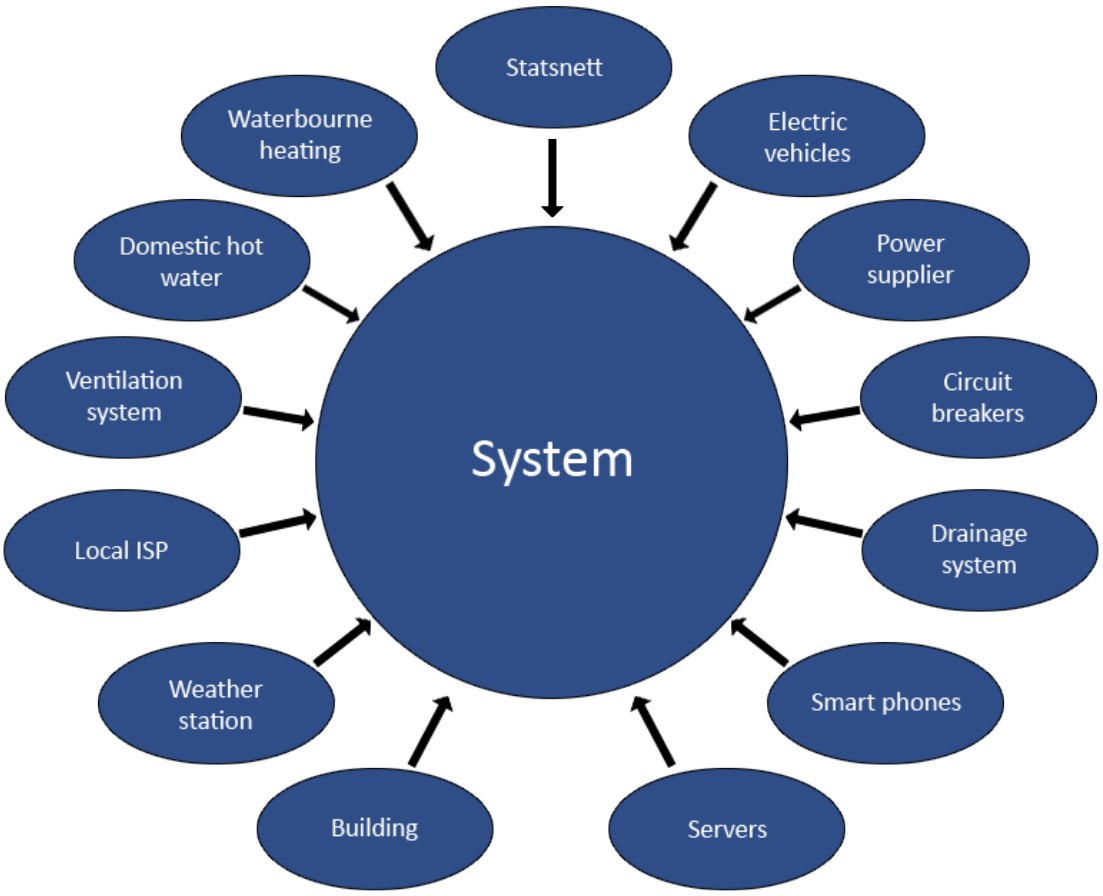


Figure 8: The man-made systems in our systems immediate context

Stakeholders

Requirements

User Requirements

Effectiveness Measures

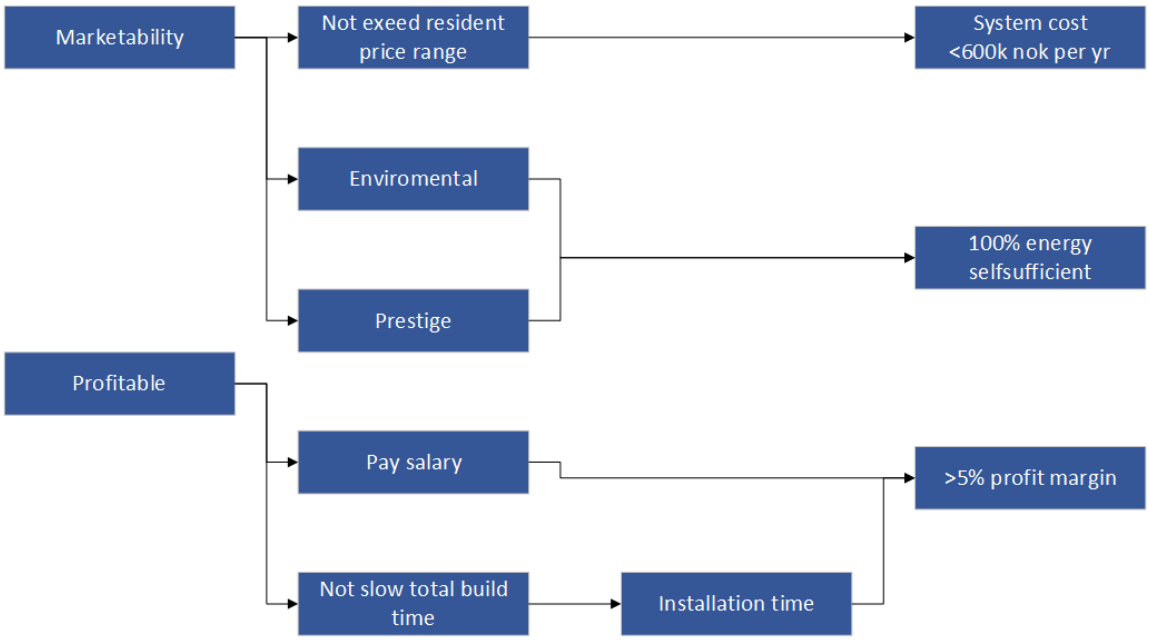


Figure 9: Key-driver-graph for OBOS

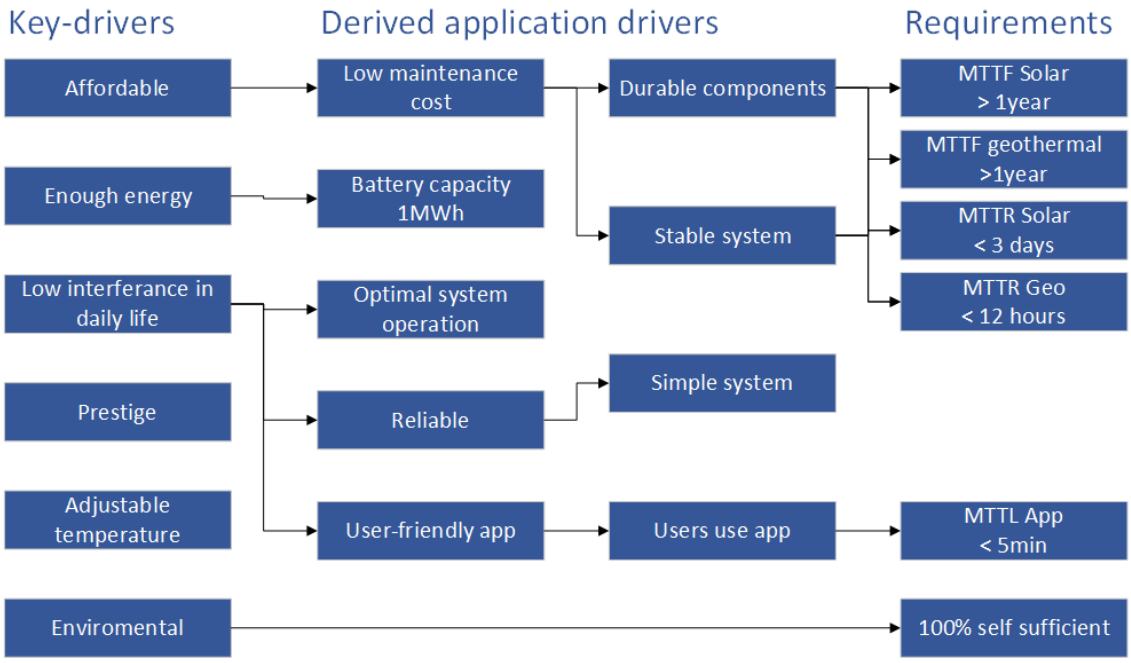


Figure 10: Key-driver-graph for the users of our system

System Requirements

Functional

Temporal Performance

Non-Temporal Performance Requirements

Interface Requirements

Design Requirements

Energy Consumption

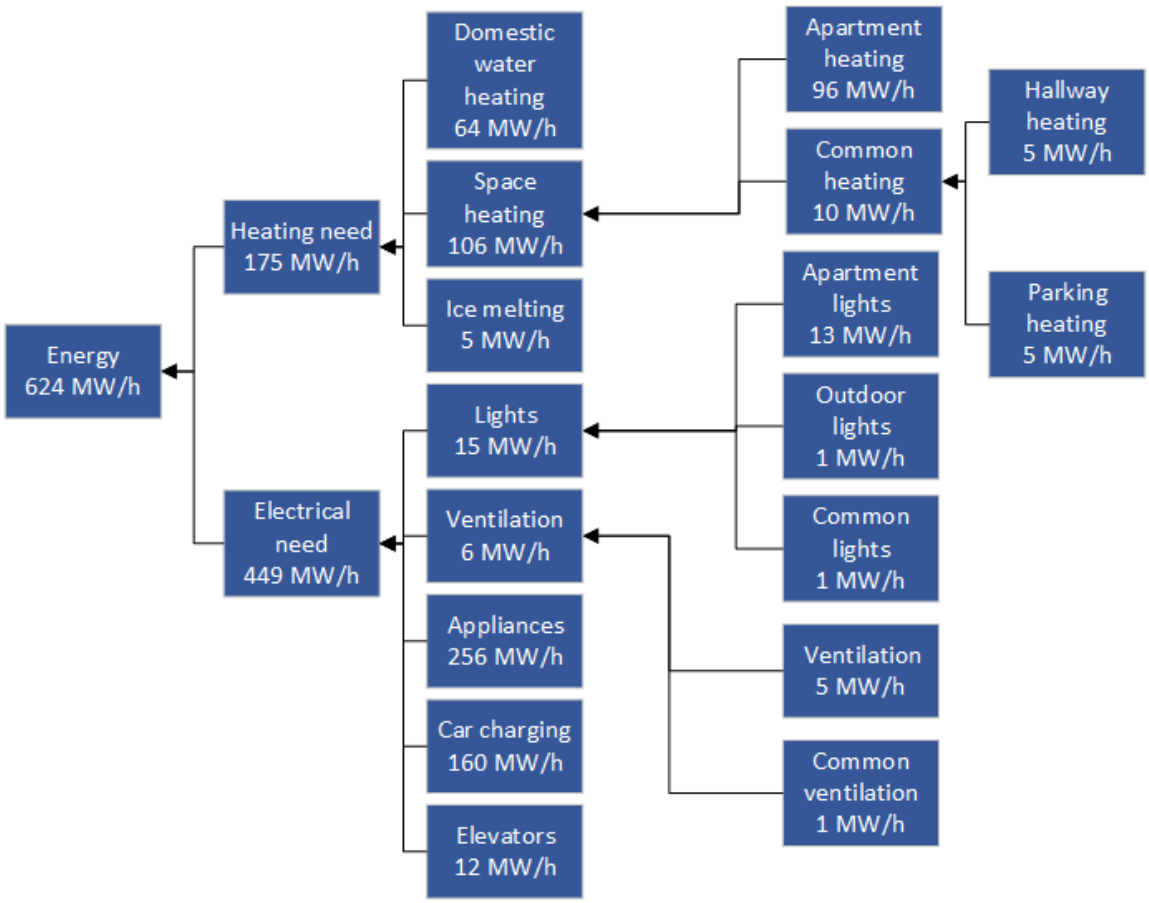


Figure 11: A technical budget displaying the energy needs of the building

System Architecting

Tier 0: The System in Its Context

Operation

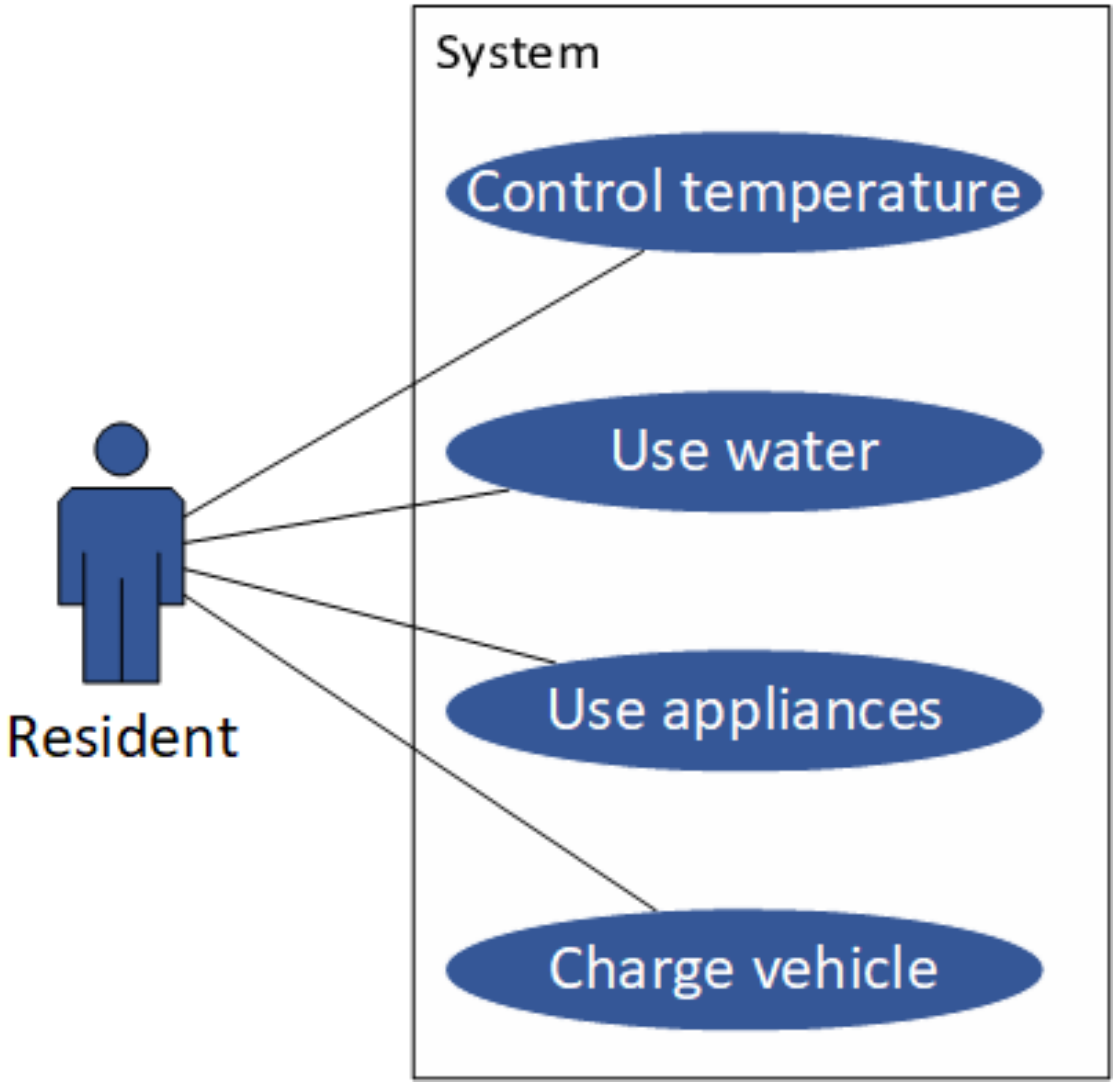


Figure 12: Use case diagram for the system

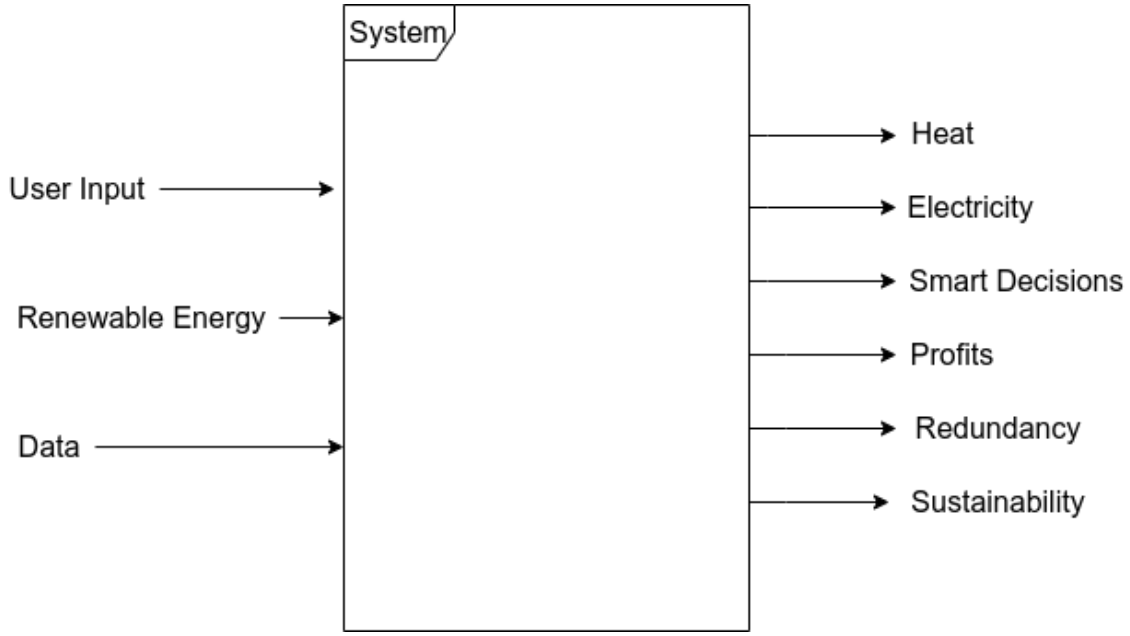


Figure 13: Black box of our system

Behavior

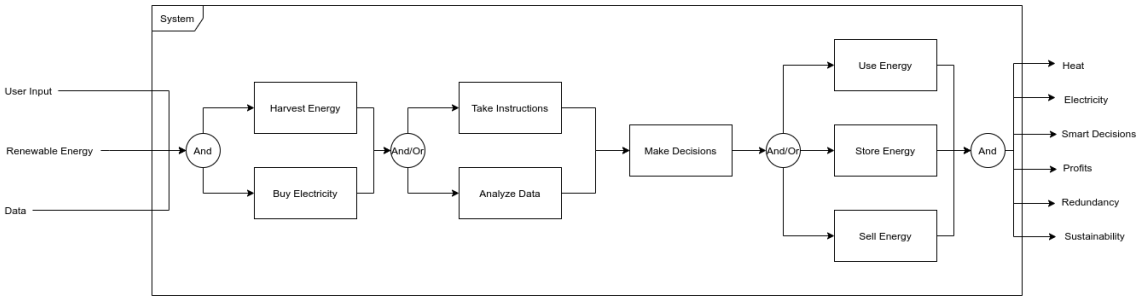


Figure 14: FFBD within our system

Structure

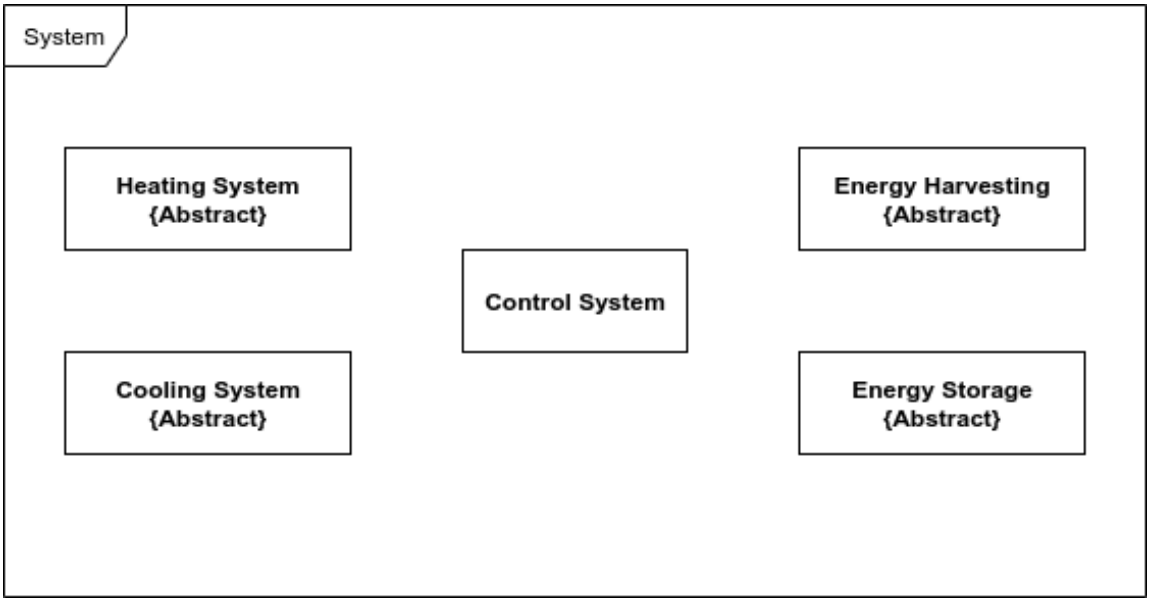


Figure 15: The system modeled as a UML class diagram, where all the classes are still abstract

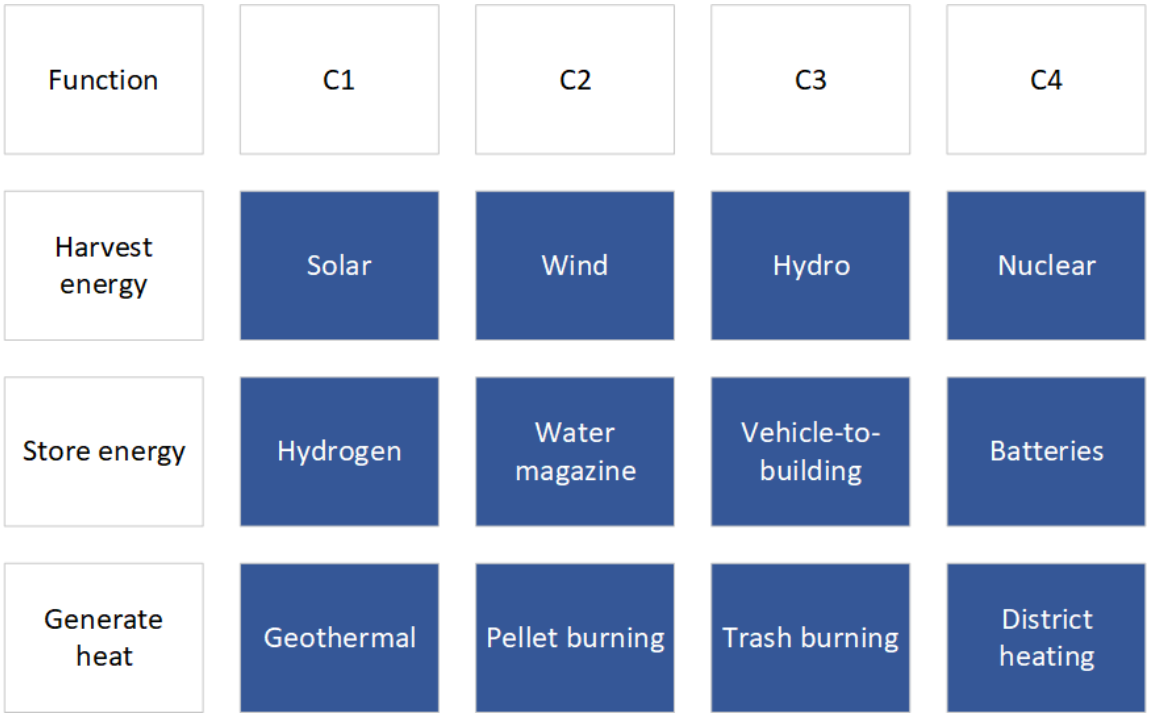


Figure 16: Morphological diagram, showing possible concepts

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 17: 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**

Figure 18: Pugh matrix for storing energy

Generating Heat

Critera	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 19: Pugh matrix for generating heat

Recommendations

Function	C1	C2	C3	C4
Harvest energy	Solar	Wind	Hydro	Nuclear
Store energy	Hydrogen	Water magazine	Vehicle-to-building	Batteries
Generate heat	Geothermal	Pellet burning	Trash burning	District heating

Figure 20: Morphological diagram, showing the recommended concept

Derived Requirements

Tier 1: The System

Operation

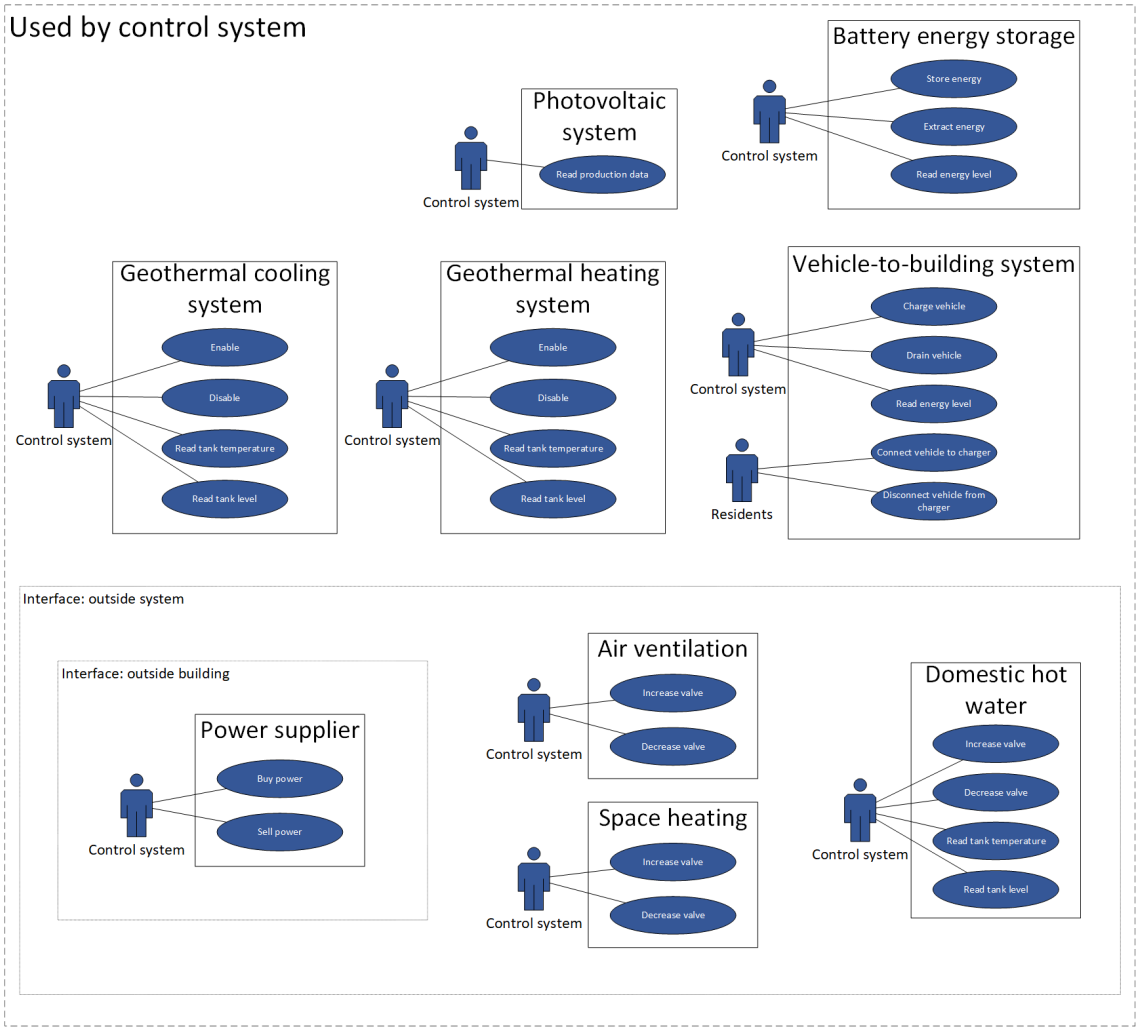


Figure 21: Use case diagram, between subsystems

Control System

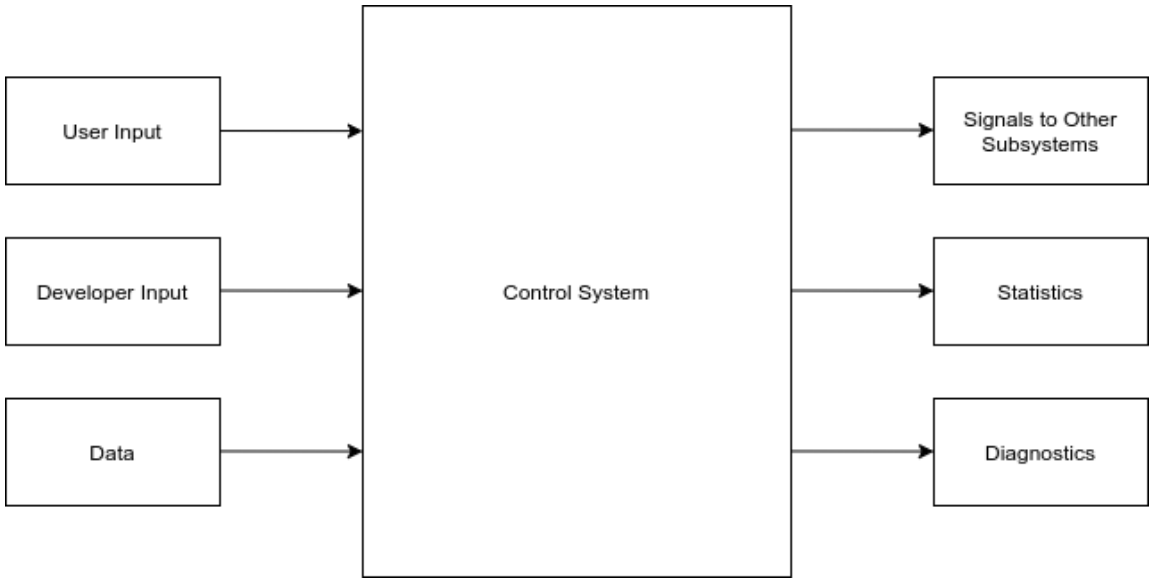


Figure 22: Black box of Control System

Photovoltaic System

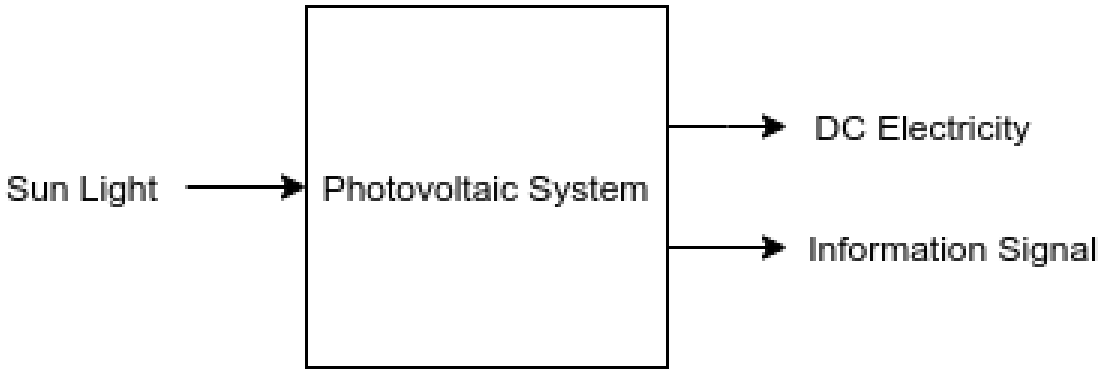


Figure 23: Black box of Photovoltaic System

Energy Storage system

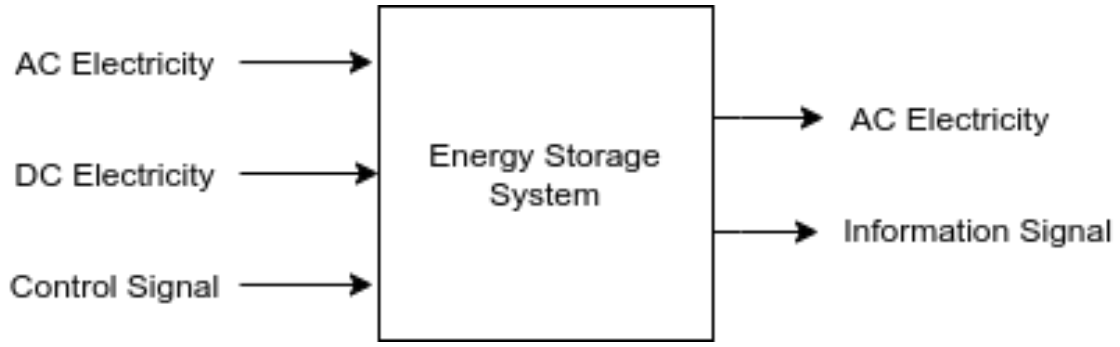


Figure 24: Black box of Energy Storage System

Geothermal Heating/Cooling

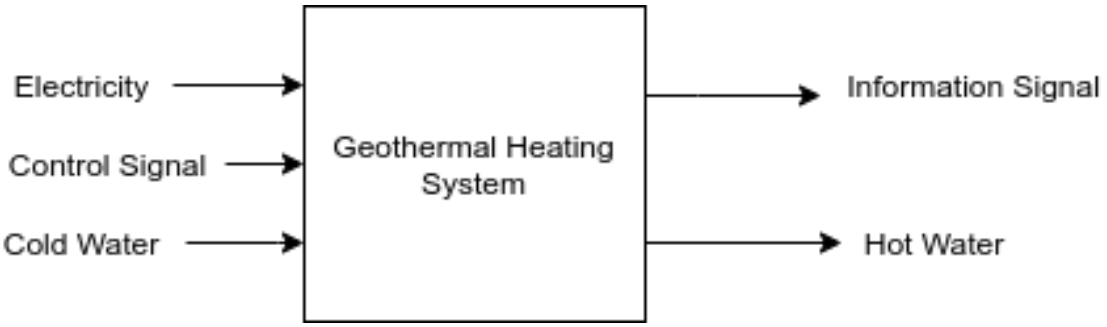


Figure 25: Black box of Geothermal Heating System

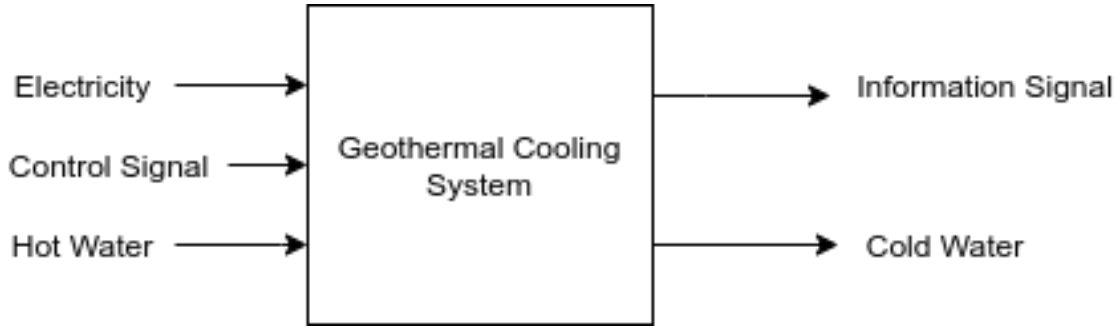


Figure 26: Black box of Geothermal Cooling System

Behavior

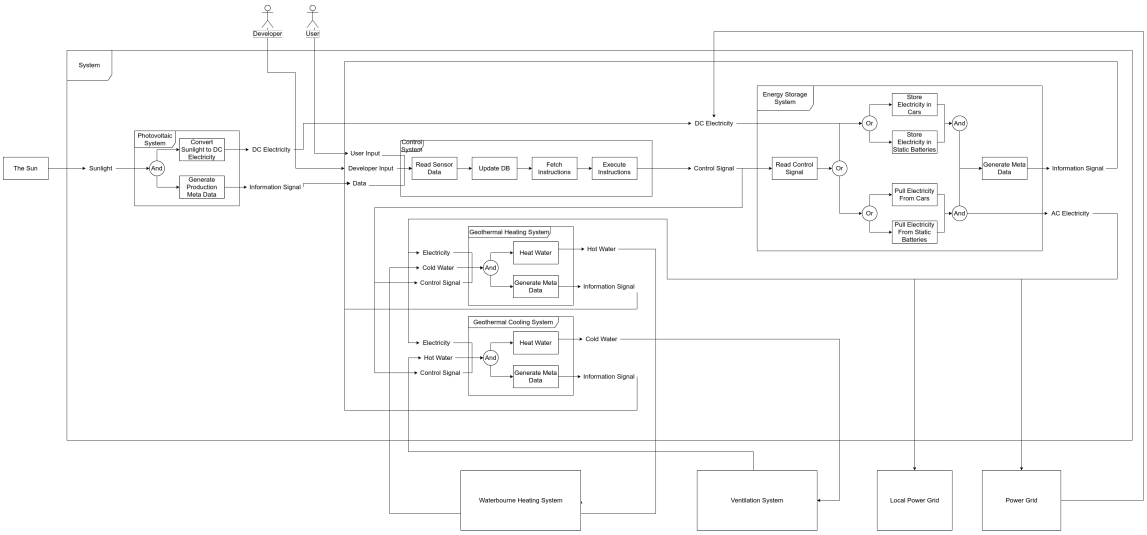


Figure 27: FFBD of the respective subsystems

Structure

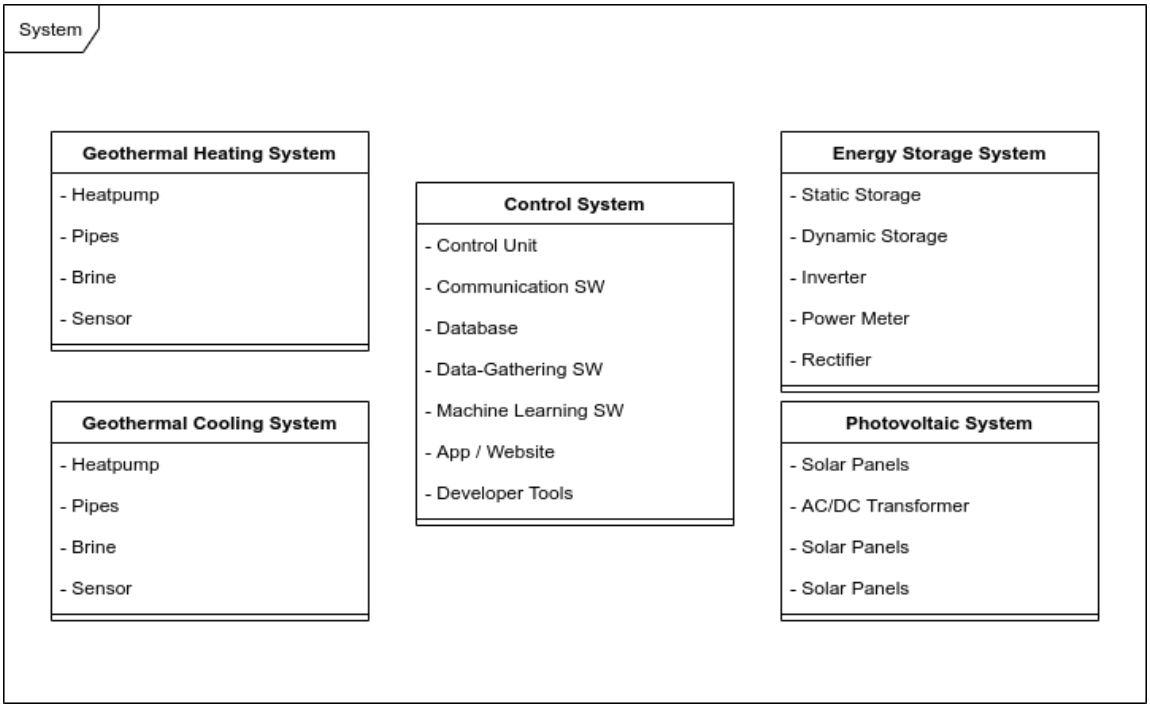


Figure 28: Class diagram showing the subsystems

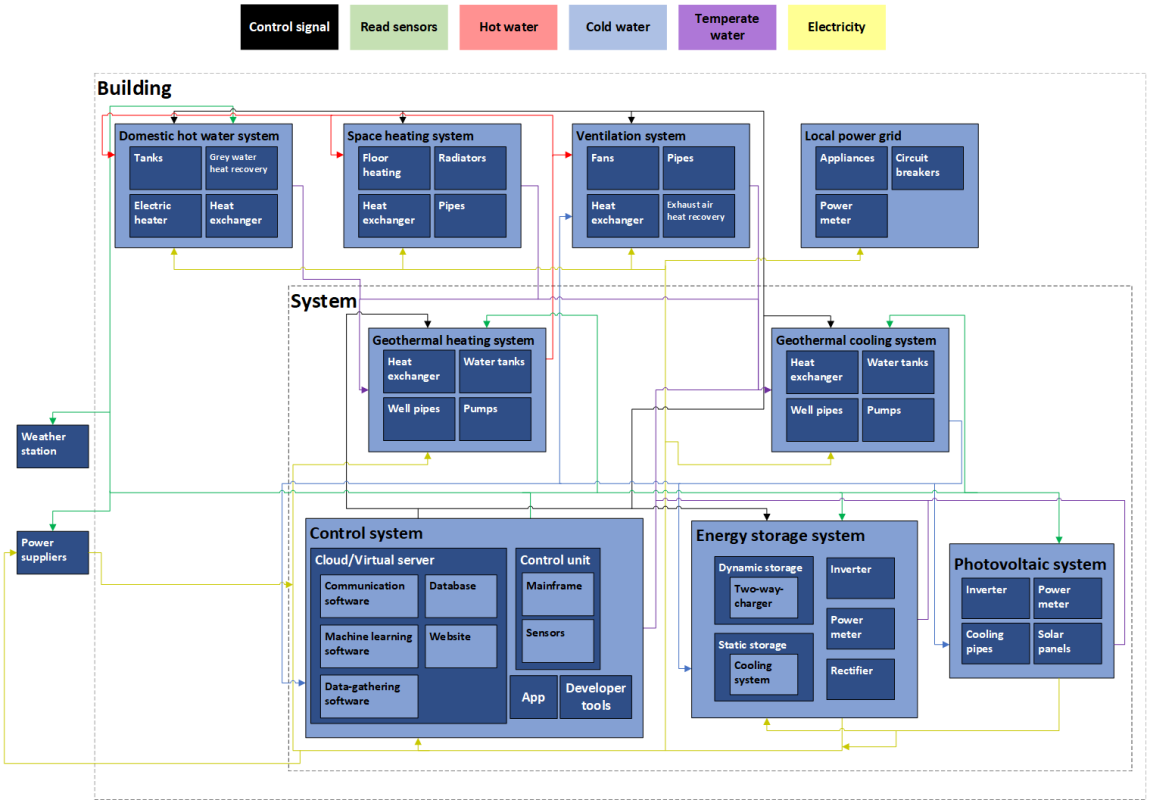


Figure 29: A more detailed structural model of our system and its context

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 30: Pugh matrix for Control System

Recommendations

Derived Requirements

Tier 2: Subsystems

Energy Storage System

Behavior

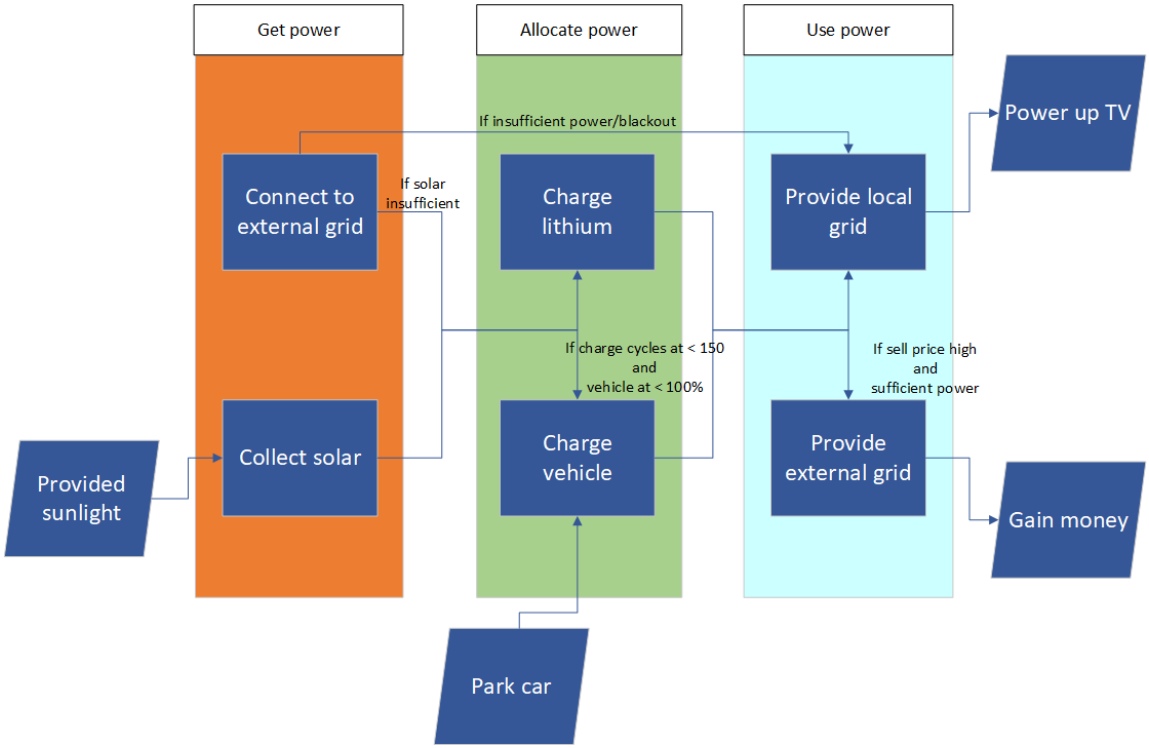


Figure 31: Behavior of the Energy Storage System

Structure

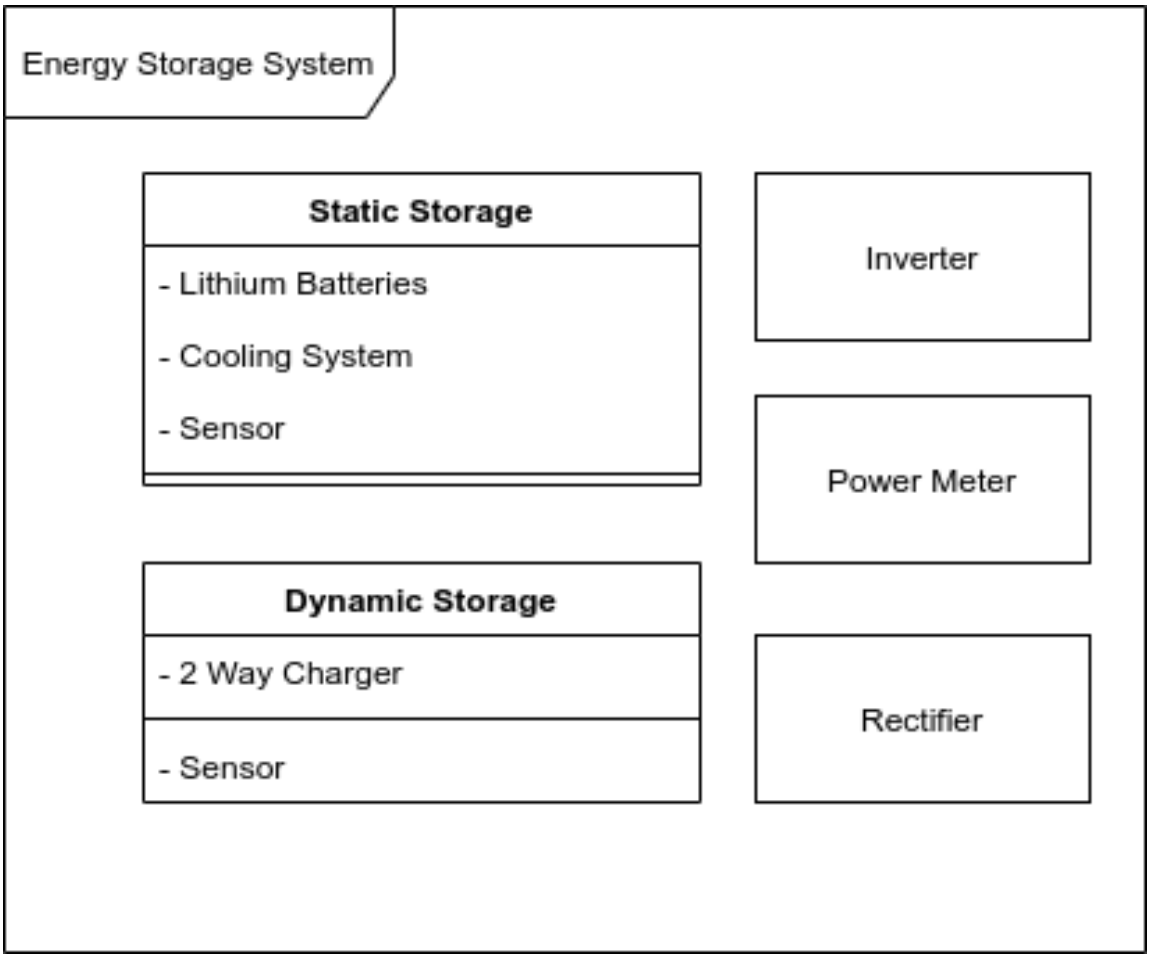


Figure 32: Class diagram of the Energy Storage System

Derived Requirements

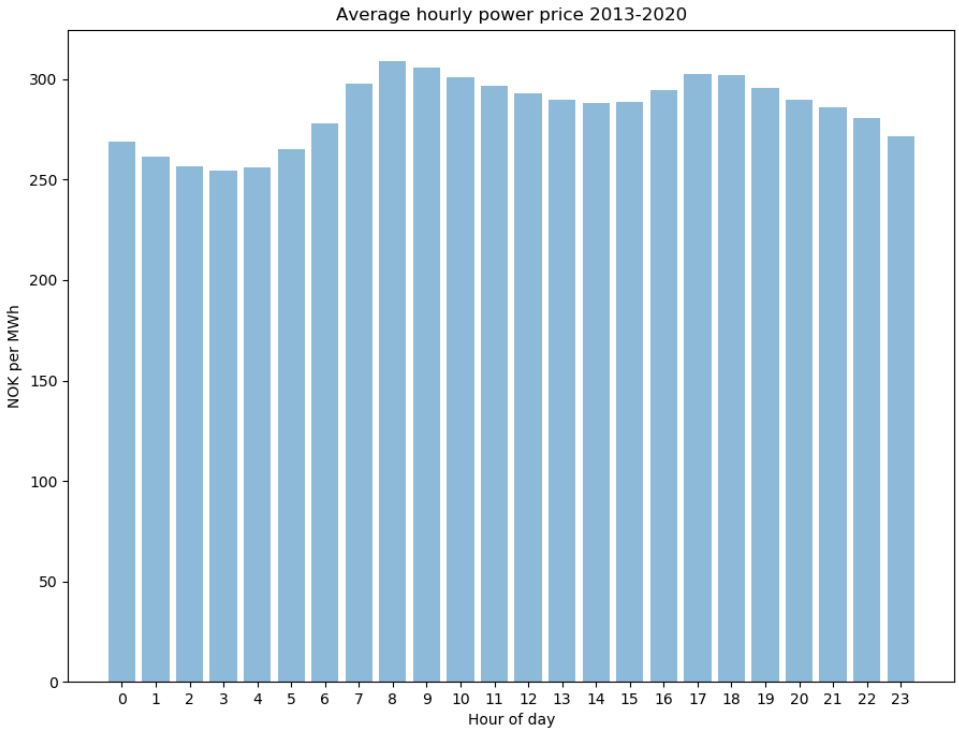


Figure 33: Average hourly power prices 2013-2020

Control System

Operation

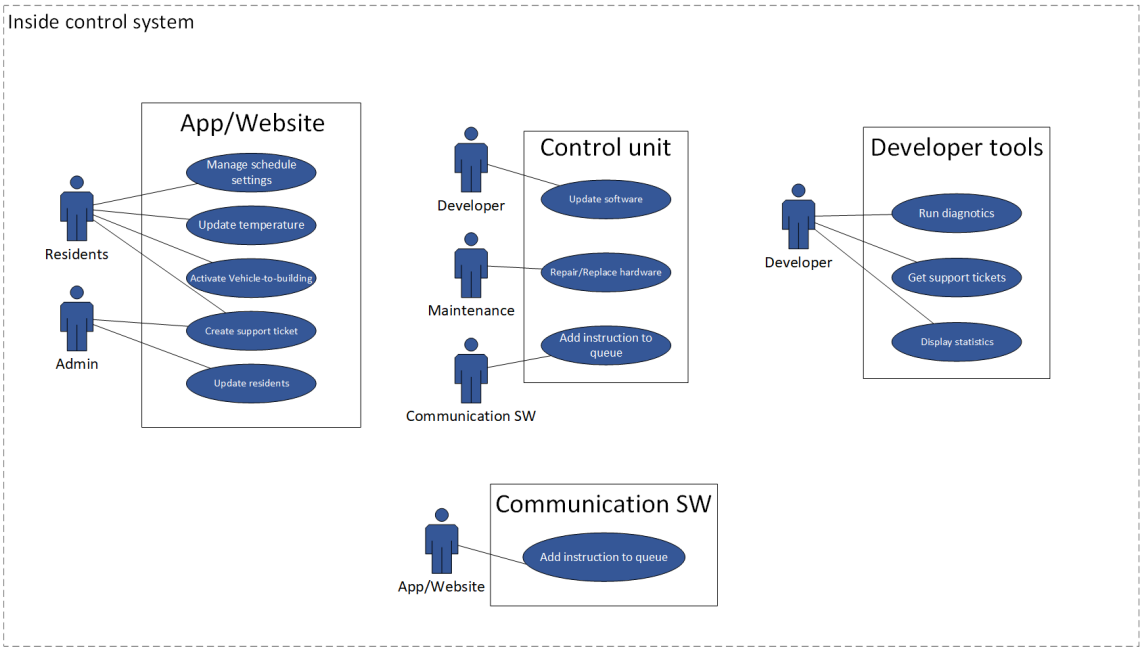


Figure 34: Use case diagram within the control system

Behavior

Control Unit

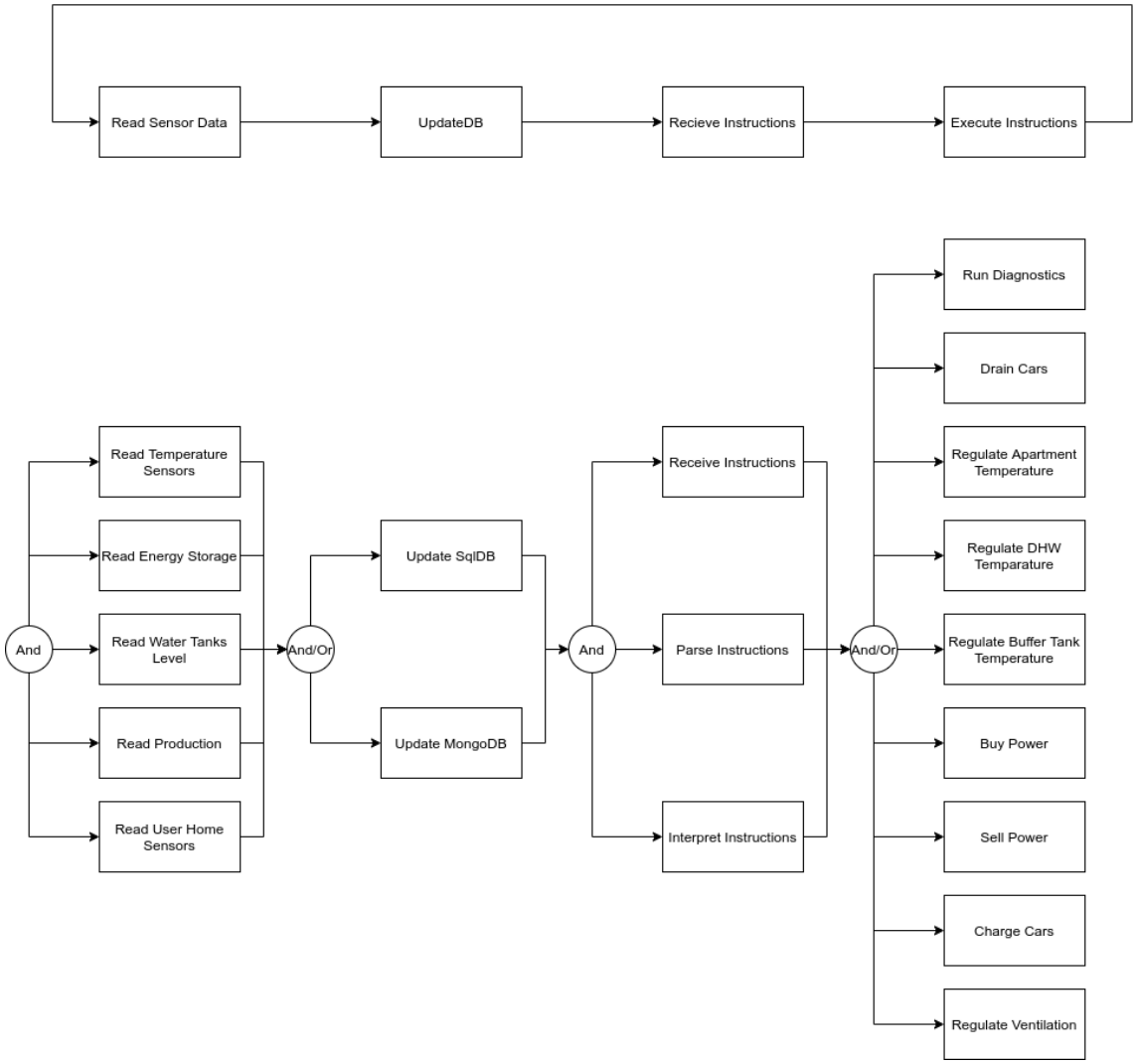


Figure 35: FFBD for the Control Unit

MCU (Main Communication Unit)

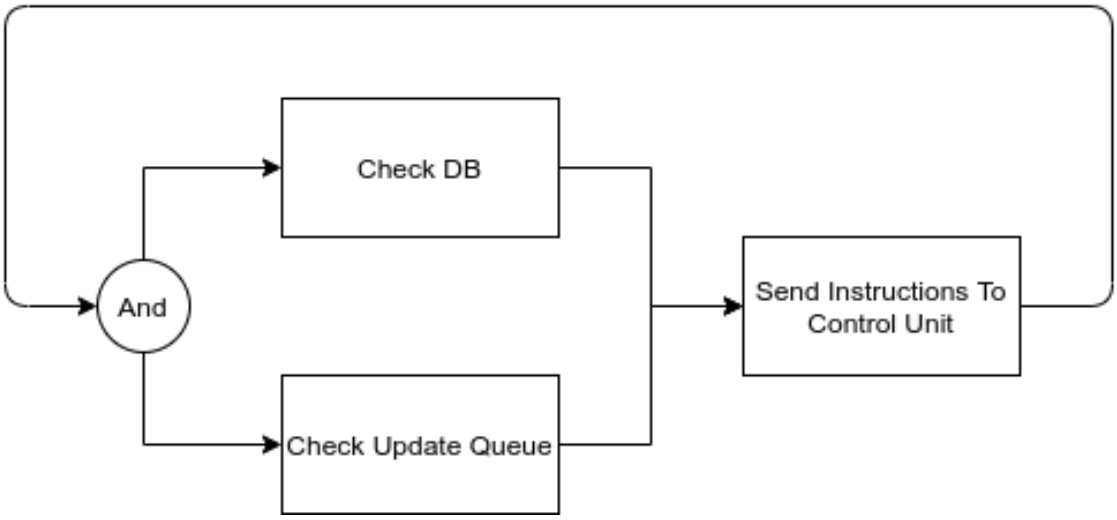


Figure 36: FFBD for the MCU

Databases

MySQL SQL *MongoDB*

Data-Gathering Software

Historical Data

Continuous Data

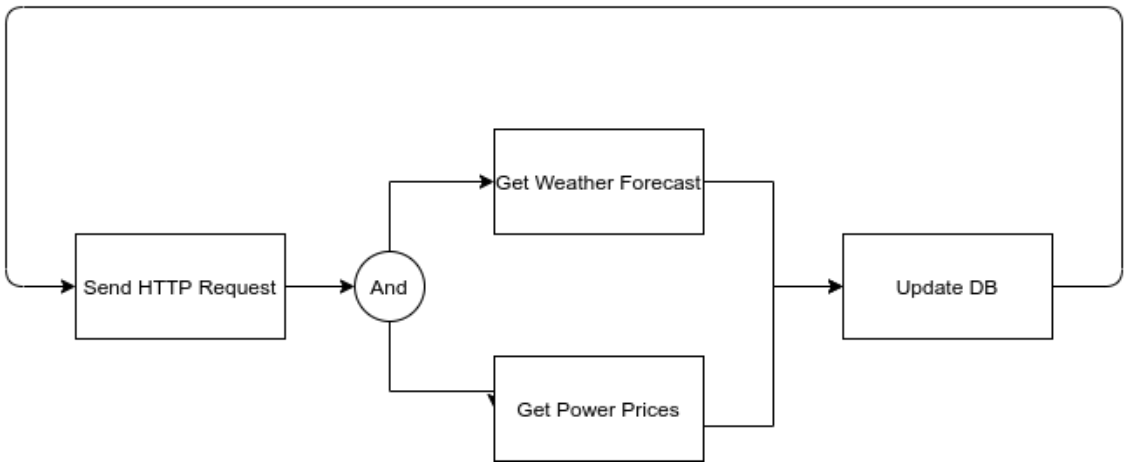


Figure 37: FFBD for the continuous data gathering software

Machine Learning Software

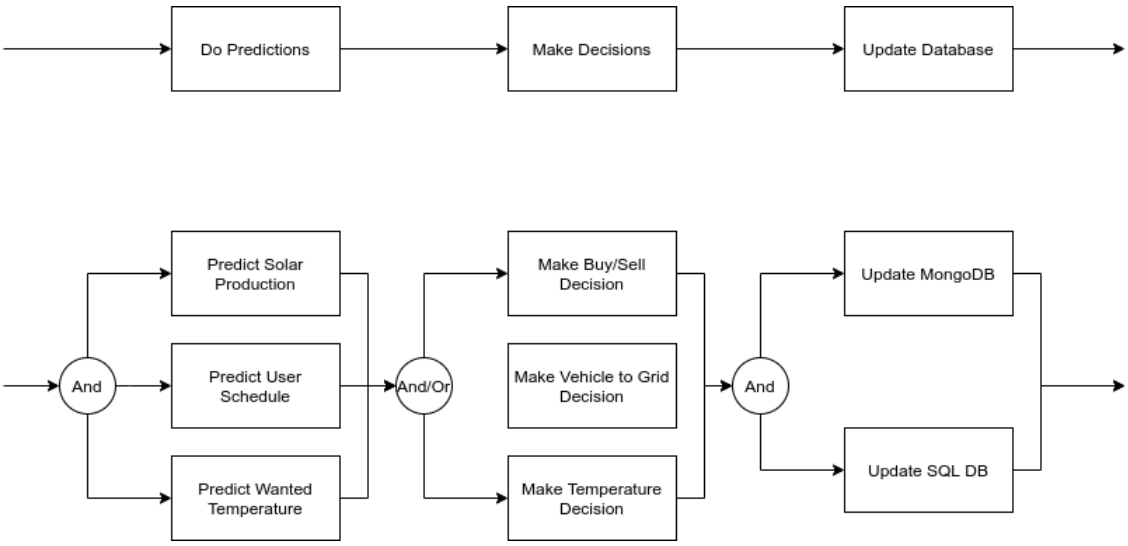


Figure 38: FFBD for the machine learning software

Application / Website

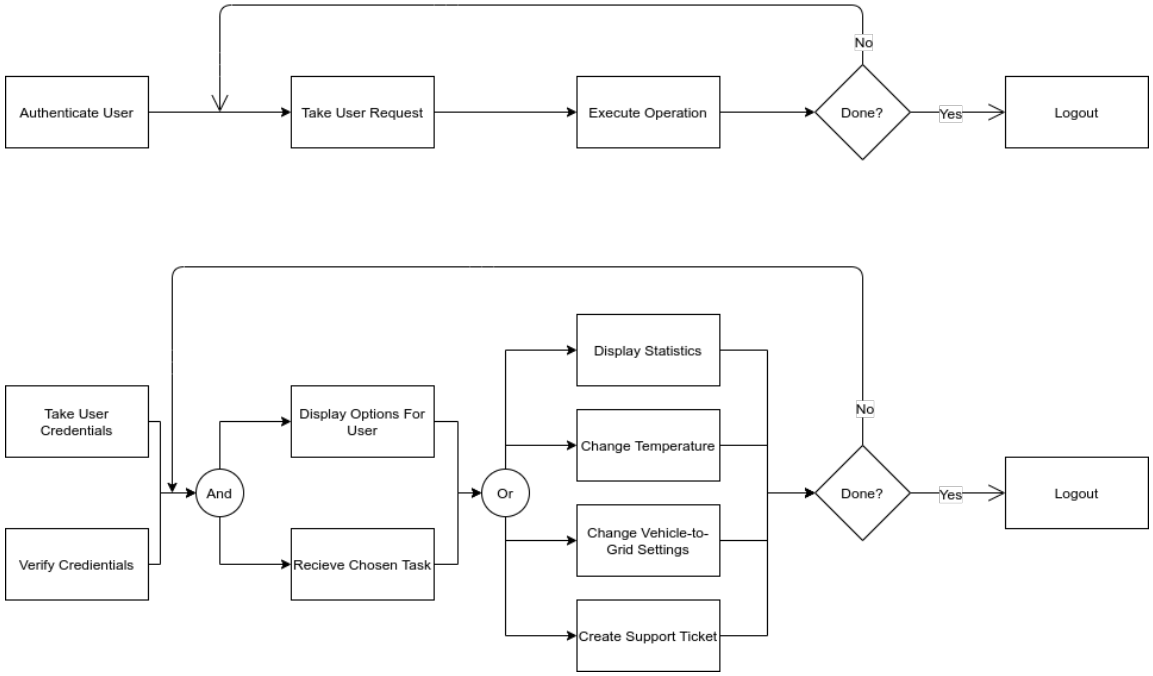


Figure 39: FFBD for the application/website

Developer Tools

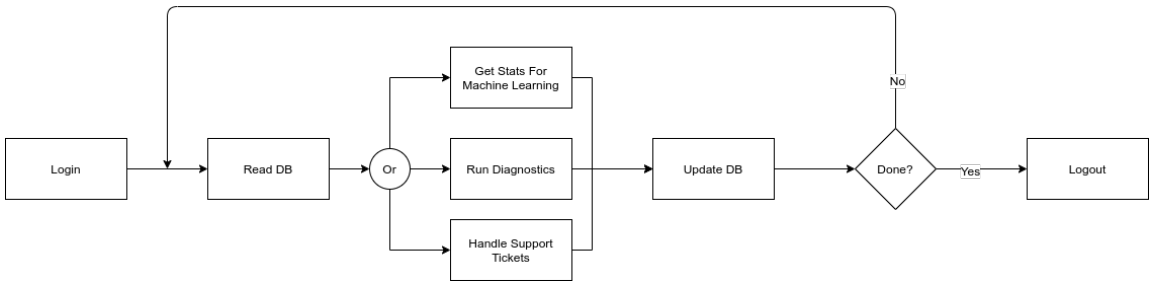


Figure 40: FFBD for the developer tools

Structure

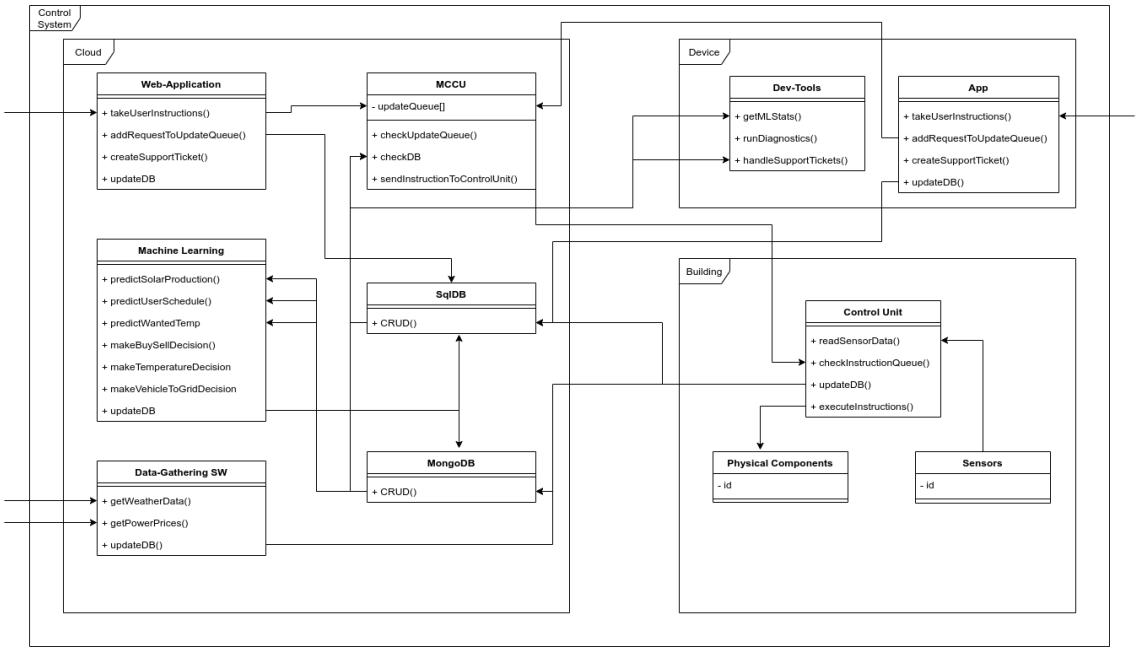


Figure 41: Class diagram for Control System

Derived Requirements

Build and Test Plan

Verification

Production

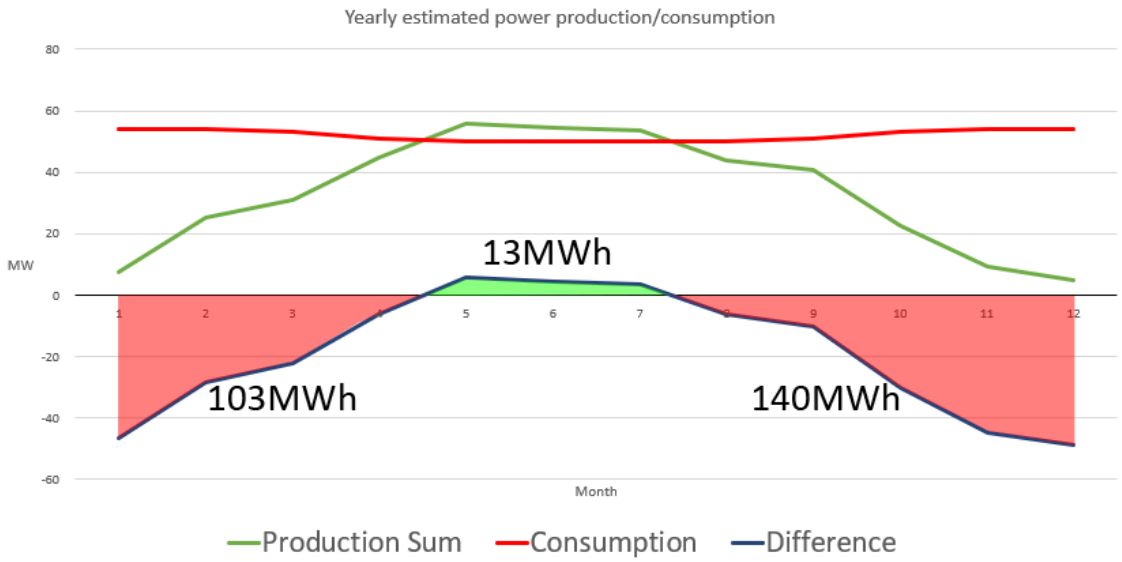


Figure 42: FINN PÅ EN GOD CAPTION TARALD

Capacity for Storage

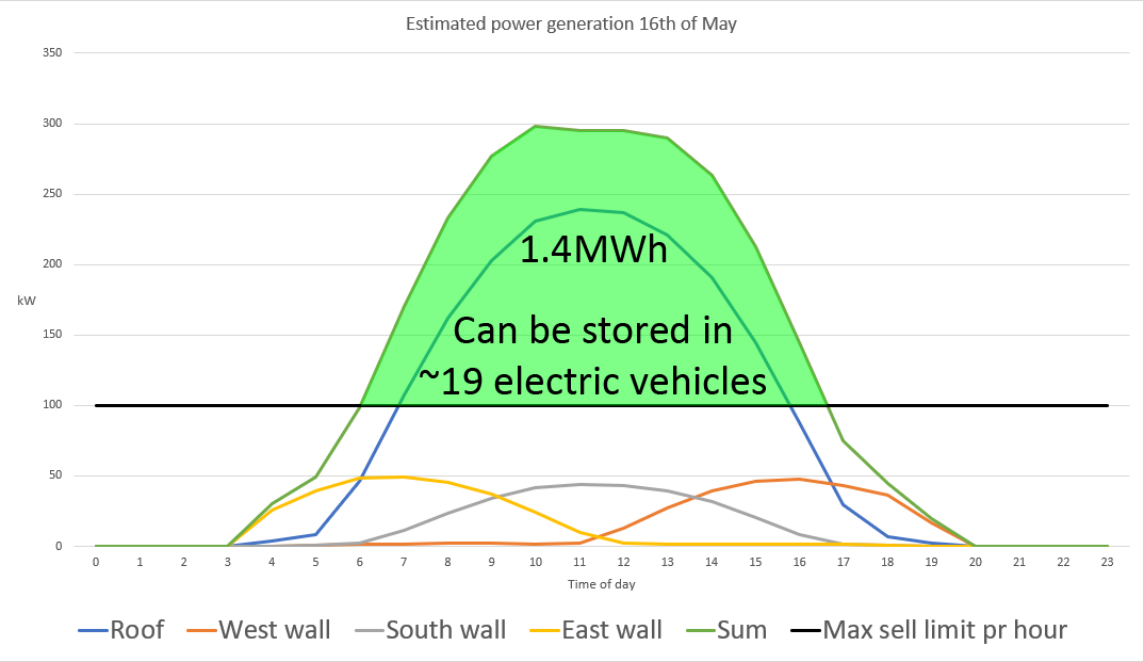


Figure 43: FINN PÅ EN GOD CAPTION TARALD

Prototypes / Proof-of-Concept

Machine Learning

Predicting Solar Panel Power Production

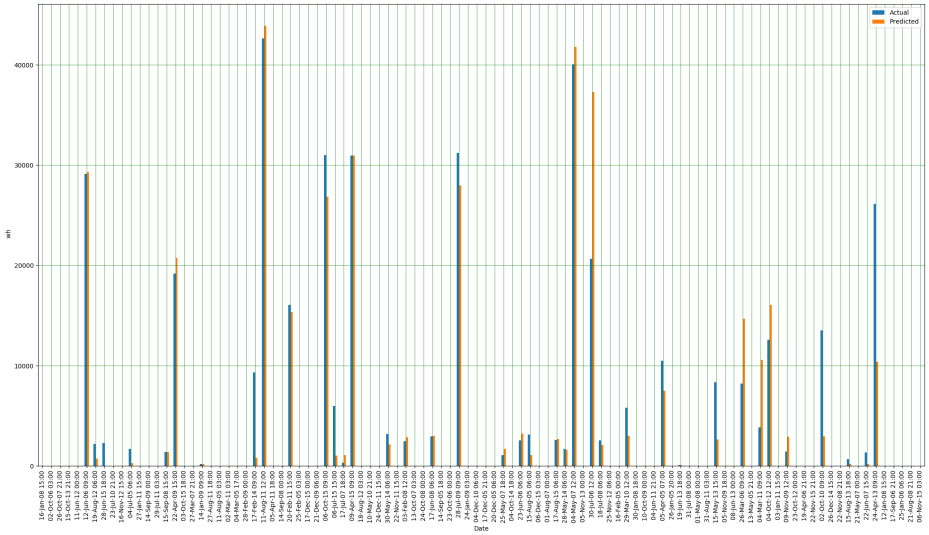


Figure 44: Machine learning prediction results for shuffled solar production

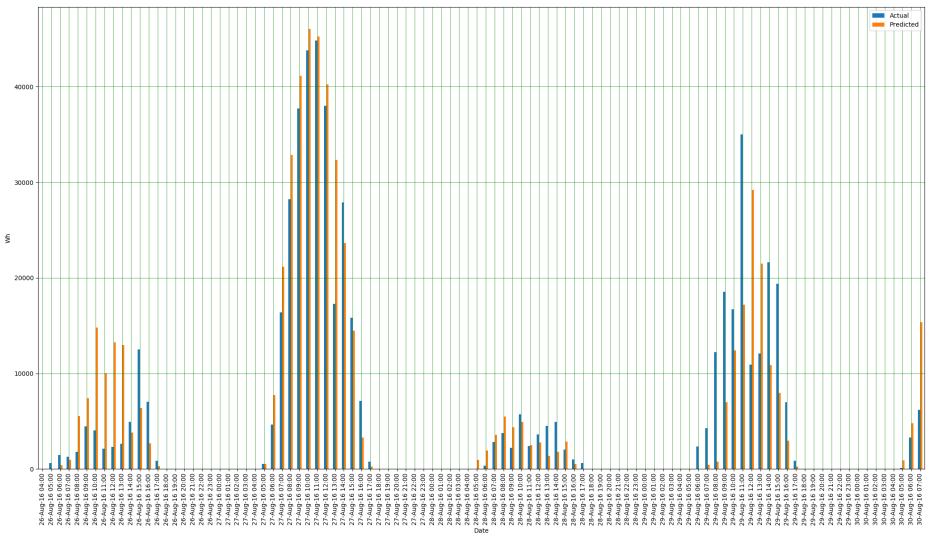


Figure 45: Machine learning prediction results for chronological solar production

Predicting Power Prices

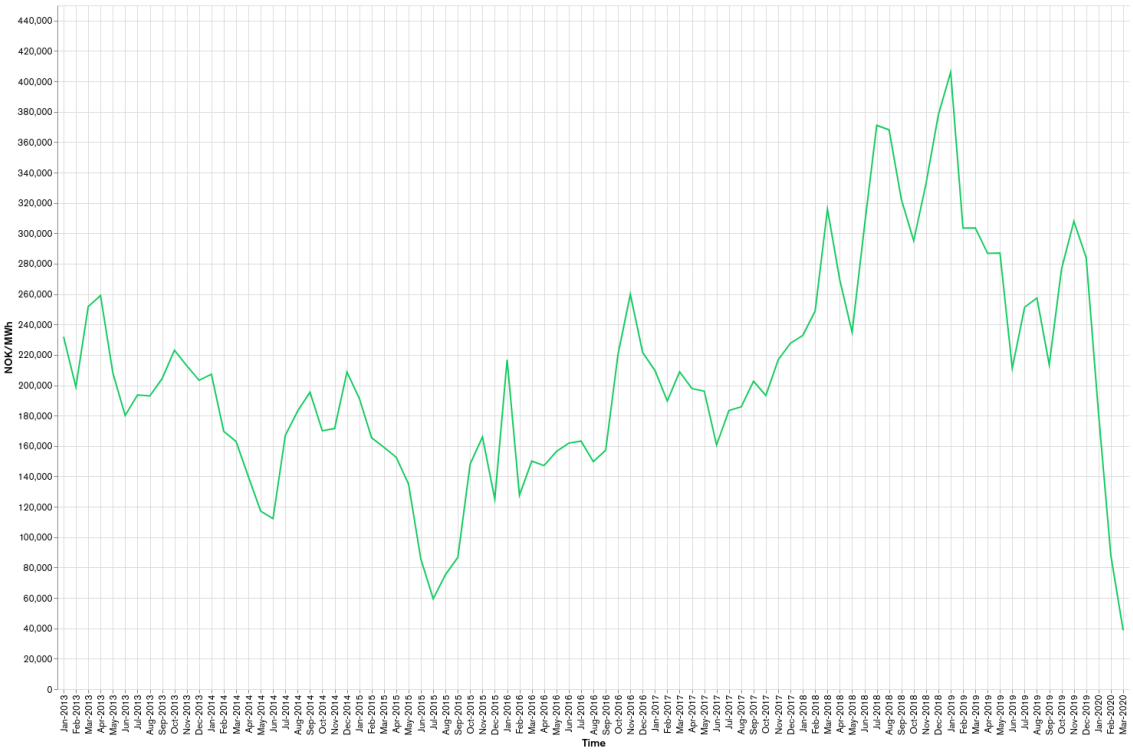


Figure 46: Power prices 2013-2020 in NOK/MWh

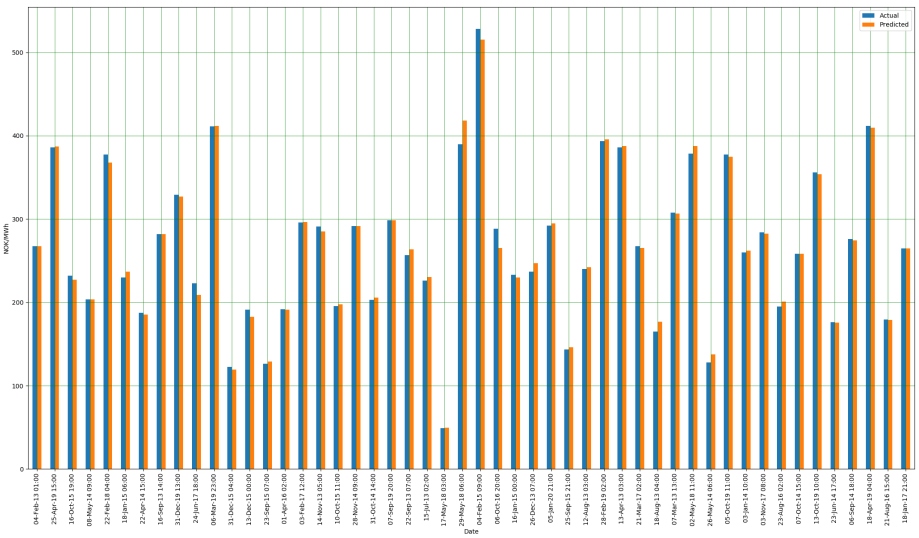


Figure 47: Machine learning prediction results for shuffled power prices

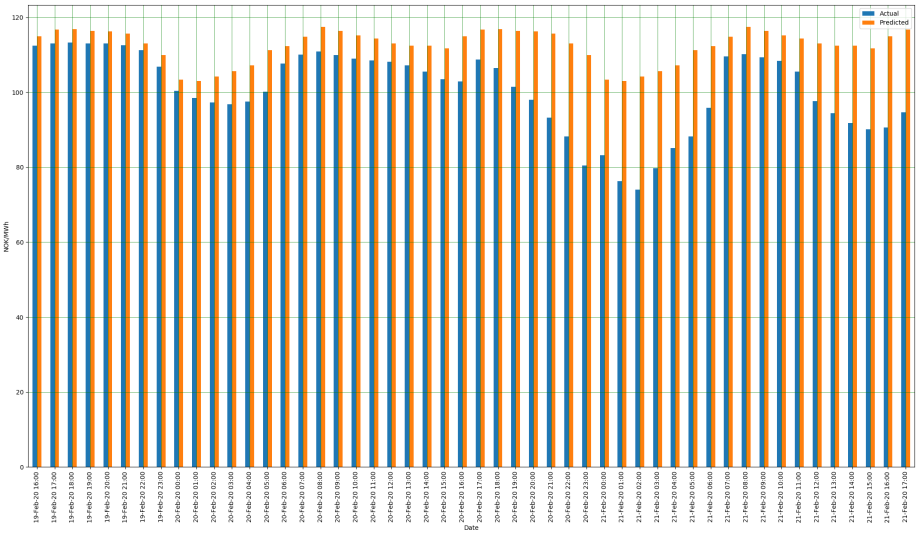


Figure 48: Machine learning prediction results for chronological power prices

Web-Application

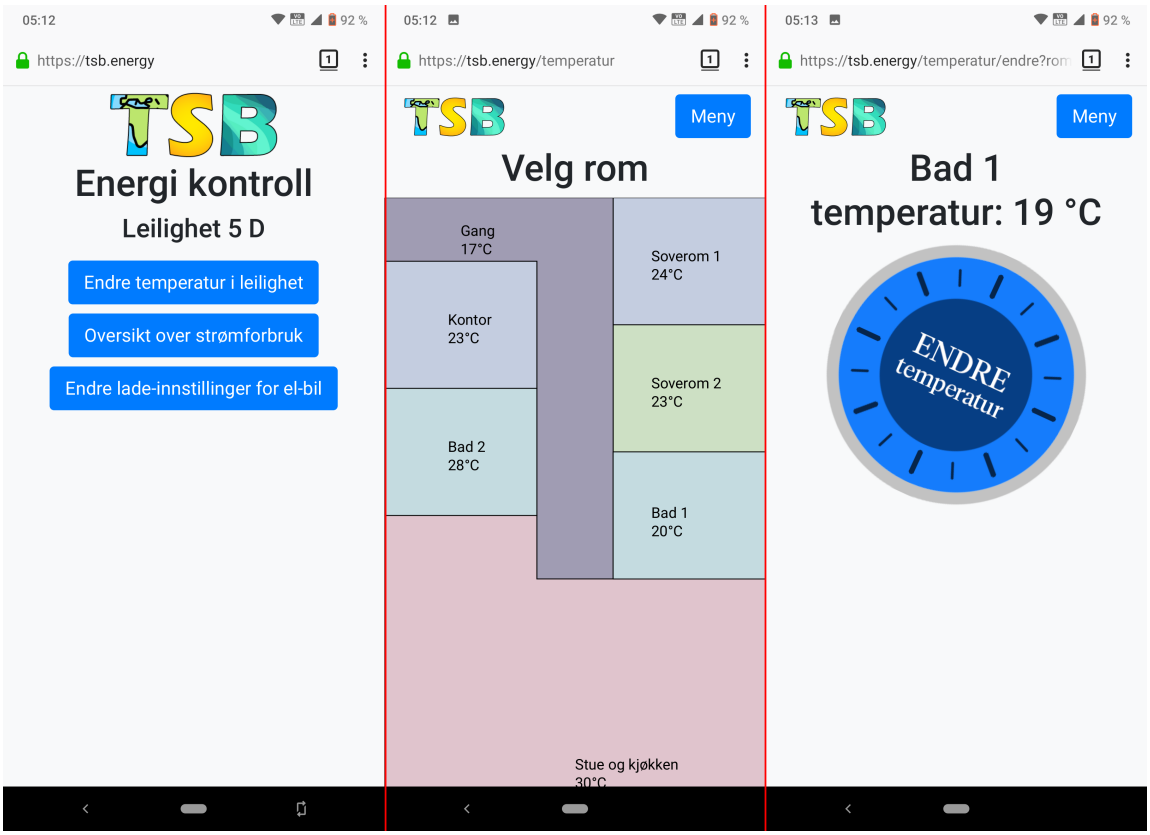


Figure 49: Screenshots of the website prototype

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 50: 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 51: 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 52: Budget for our company

Our Cash Flow

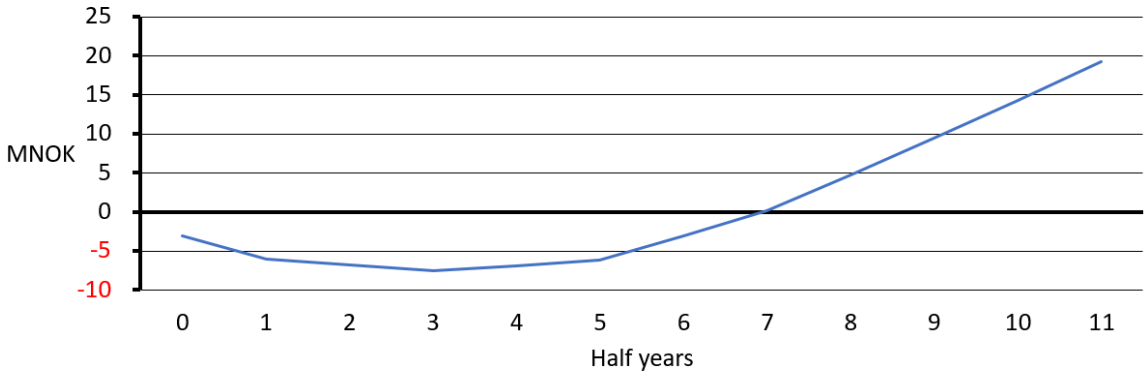


Figure 53: Cash flow graph for our company

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	-104k	-104k	-104k	-104k	-104k	-104k	-104k	-104k	-104k
Service-agreement	-50k	-50k	-50k	-50k	-50k	-50k	-50k	-50k	-50k	-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 54: Budget for customer

Residents Cash Flow

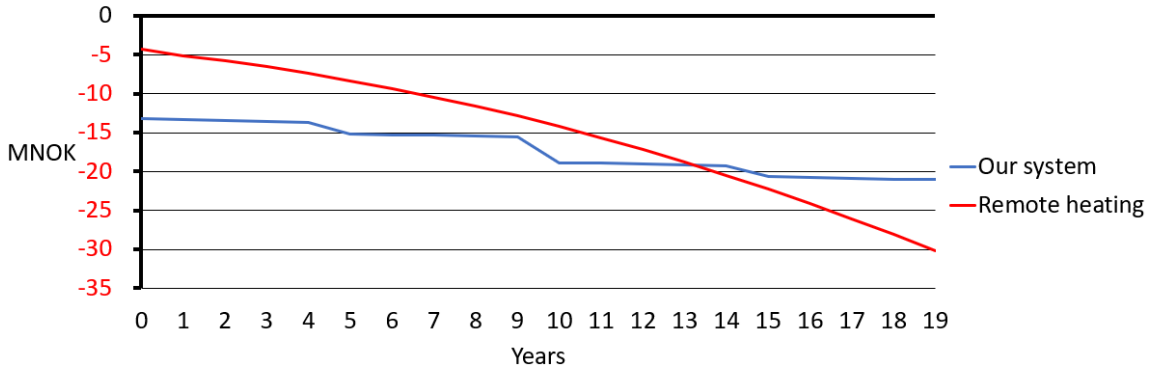


Figure 55: Cash flow graph for customer

Conclusion

References

[1] Nordpool Group ,. https://www.nordpoolgroup.com/. Accessed: March-20.

[2] Rooftop Wind Turbines: Are They Worthwhile? , engineering.com. https://www.engineering.com/ElectronicsDesign/ElectronicsDesignArticles/ArticleID/9556/Rooftop-Wind-Turbines-Are-They-Worthwhile.aspx. Accessed: March-20.

[3] Siragrunnen vindpark , multiconsult. https://www.multiconsult.no/prosjekter/siragrunnen-vindpark/. Accessed: March-20.

[4] Dette bruker du strømmen til , smartepenger.no. https://www.smartepenger.no/boligokonomi/2438-dette-bruker-du-strommen-til. Accessed: March-20.

[5] Elevator energy calculator, thyssenkrupp. https://www.thyssenkruppelevator.com/tools/energy-calculator/. Accessed: March-20.

[6] Fremtidens bolig, enova. https://www.enova.no/upload_images/42B0BD408ABB4271A1CDEABF9C402A6E.pdf. Accessed: March-20.

[7] Kjørelengder, ssb. https://www.ssb.no/transport-og-reiseliv/statistikker/klreg. Accessed: March-20.

[8] Krav til energieffektivitet TEK17 , direktoratet for byggkvalitet. https://dibk.no/byggereglene/byggteknisk-forskrift-tek17/14/14-2/. Accessed: March-20.

[9] Krav til passivhus NS3700 , tekna.no. https://www.tekna.no/fag-og-nettverk/bygg-og-anlegg/byggbloggen/krav-til-passivhus. Accessed: March-20.

[10] Norwegian household power consumption, ssb. https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/vi-bruker-mindre-strom-hjemme. Accessed: March-20.

[11] Photovoltaic Geographical Information System , european commision. https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html. Accessed: March-20.

[12] Samlet energibruk , nve. https://www.nve.no/energibruk-effektivisering-og-teknologier/energibruk/samlet-energibruk/?ref=mainmenu. Accessed: March-20.

[13] Strømforbruk for elbil, fjordkraft. https://www.fjordkraft.no/strom/stromforbruk/elbil/. Accessed: March-20.

[14] Strømforbruk og strømsparing, hafslund strøm. https://www.hafslundstrom.no/strom/privat/stromforbruk/2025. Accessed: March-20.

[15] Veiledning TEK17 , direktoratet for byggkvalitet. https://dibk.no/byggereglene/byggteknisk-forskrift-tek17/. Accessed: March-20.

Appendices

Figures

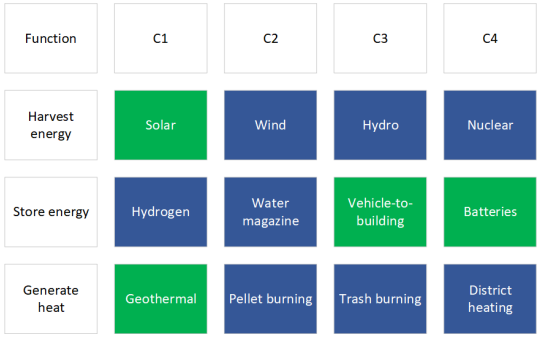


Figure 56: Use Case Diagram Example

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_iso_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://{0}:{1}@terrasolaris-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://{0}:{1}@terrasolaris-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://{0}:{1}@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
```