



Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

«Системний Інжиніринг»

Проект
Холодильник

Виконав:
Студент III курсу ФТІ
групи ФБ-81
Чалий Олексій

Перевірив:
Ткач В.М

Діаграми проекту Холодильник:

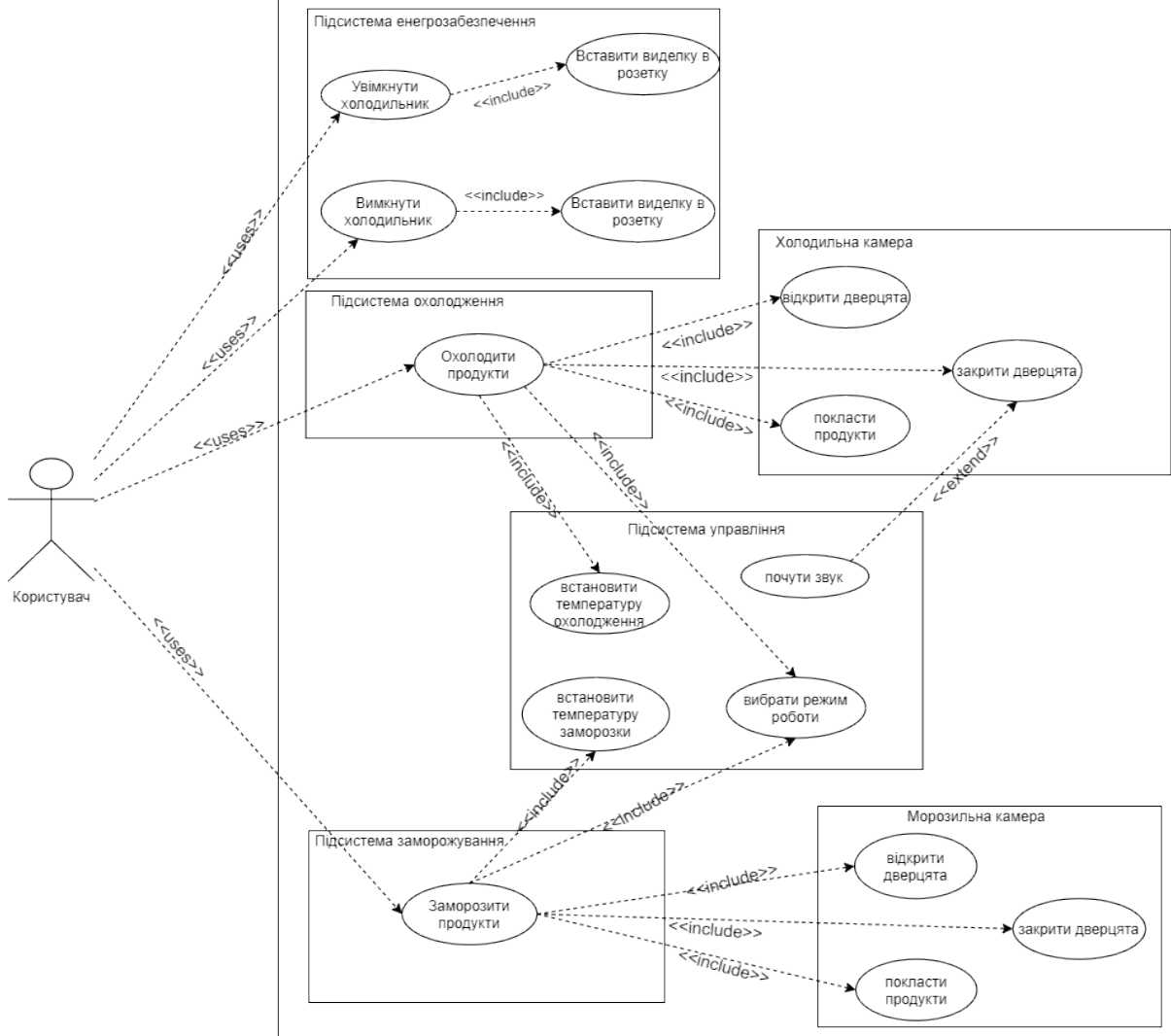
!Скрішоти проекта починаються на стр 18

Структурна діаграма

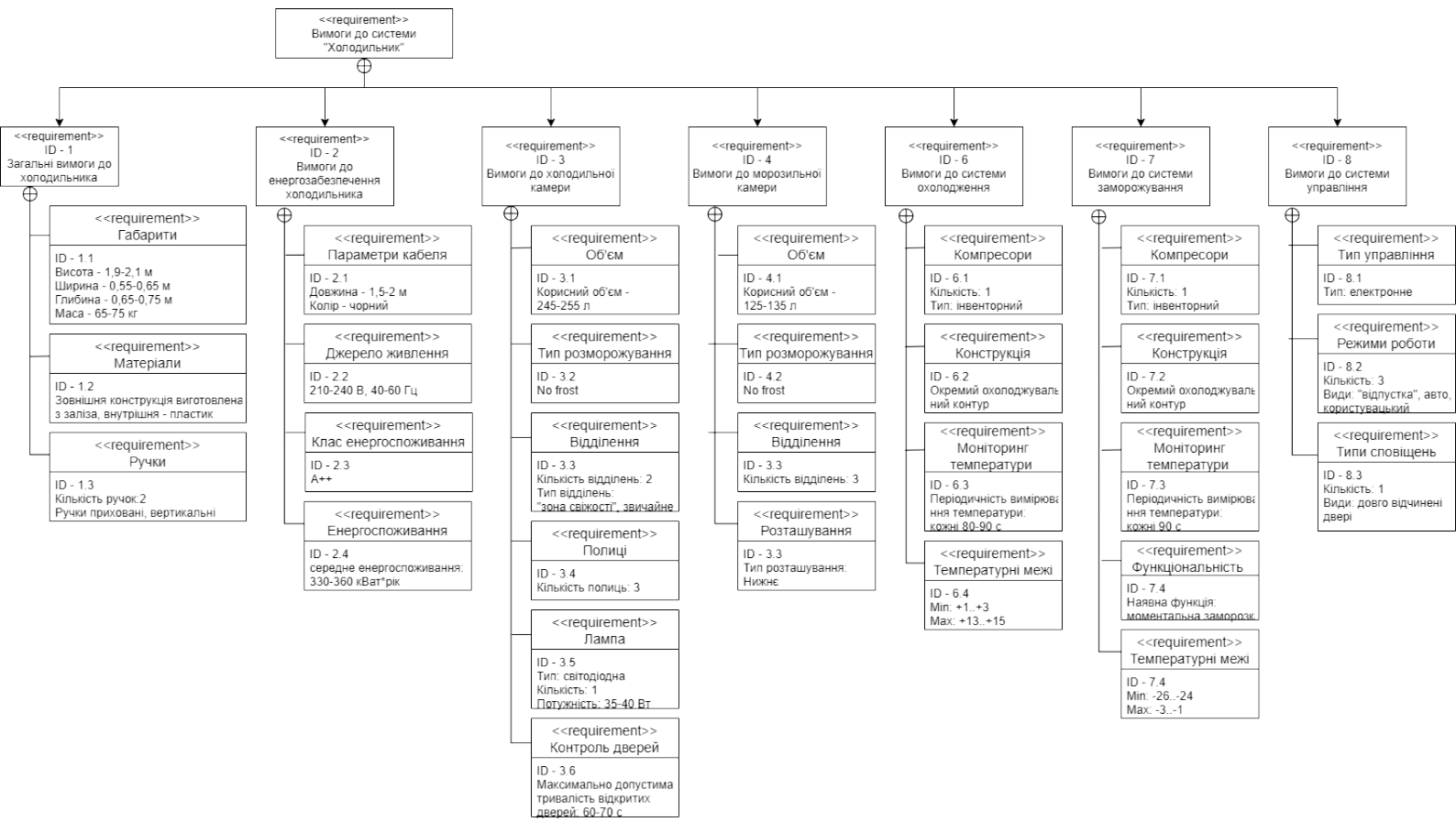


- use cases diagram

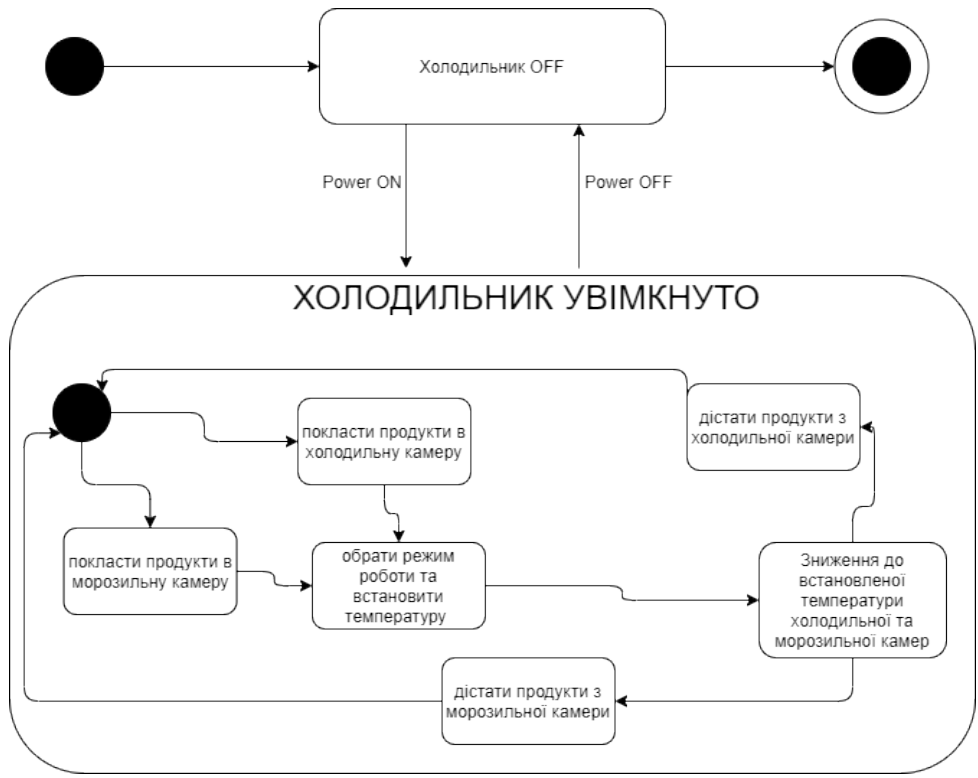
Холодильник



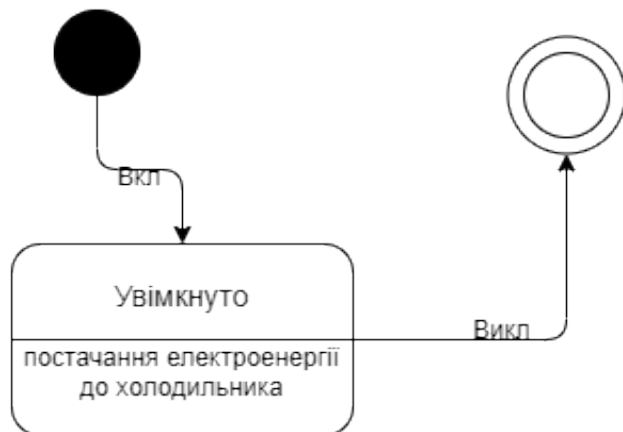
- requirements diagram



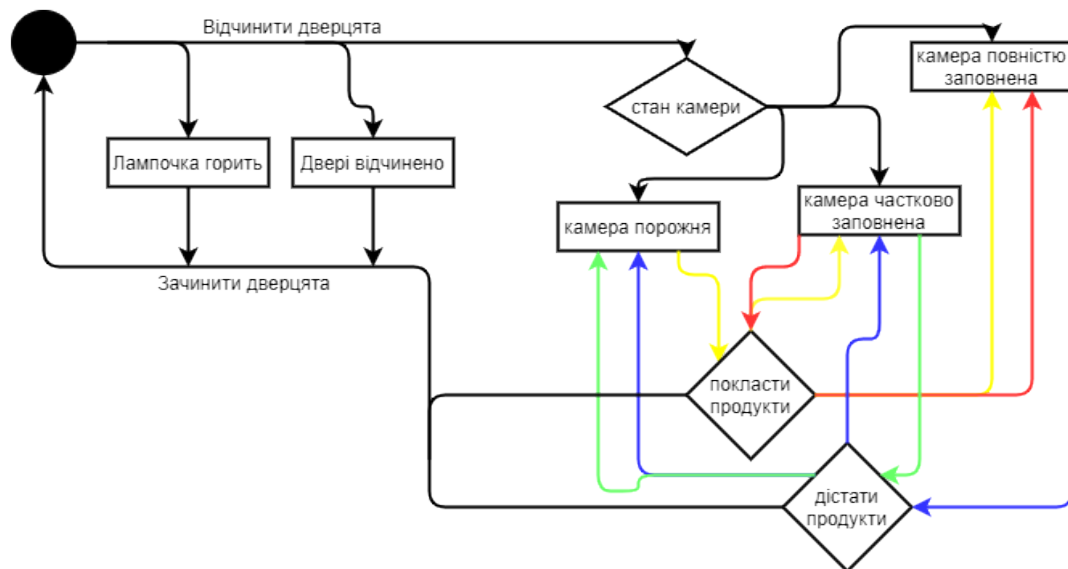
- state machine diagram



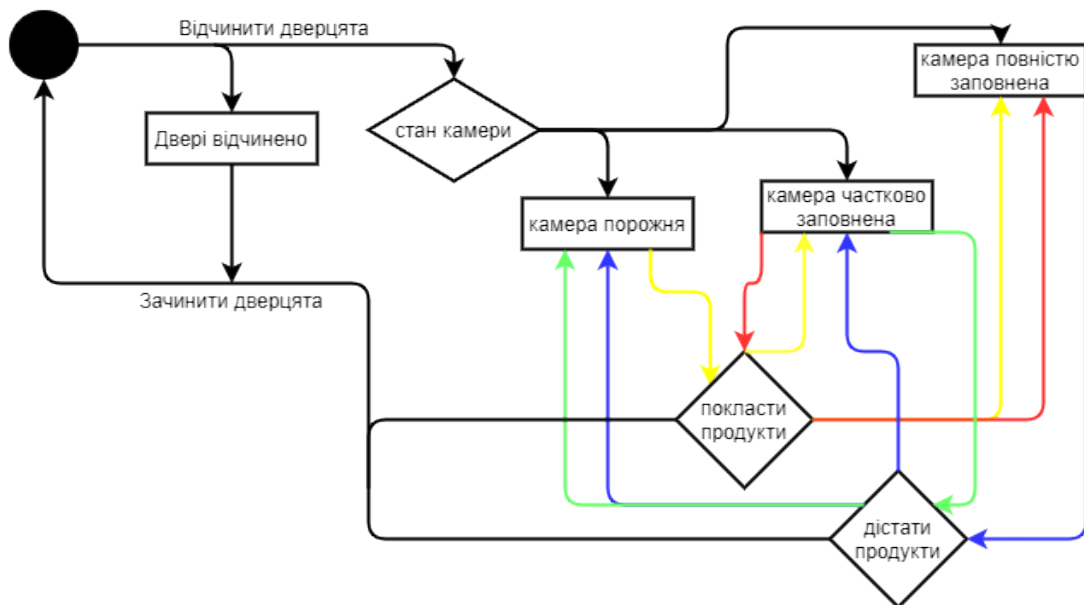
Підсистема енергозабезпечення



Підсистема холодильна камера



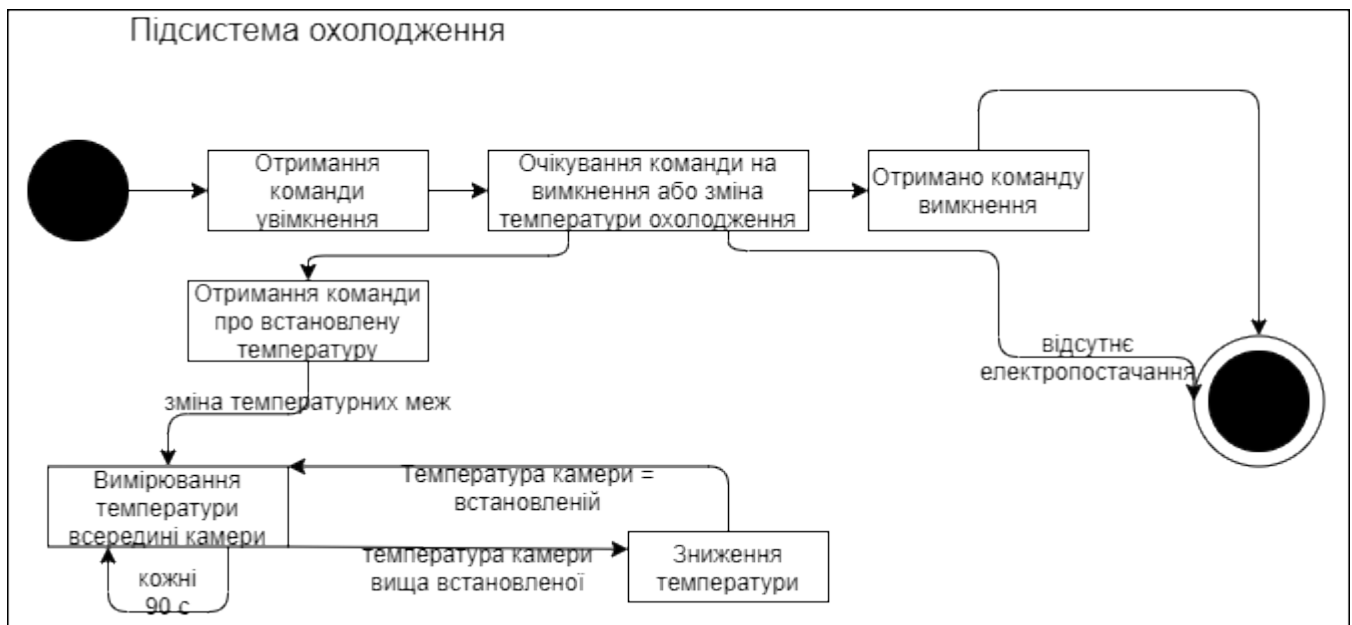
Підсистема морозильна камера



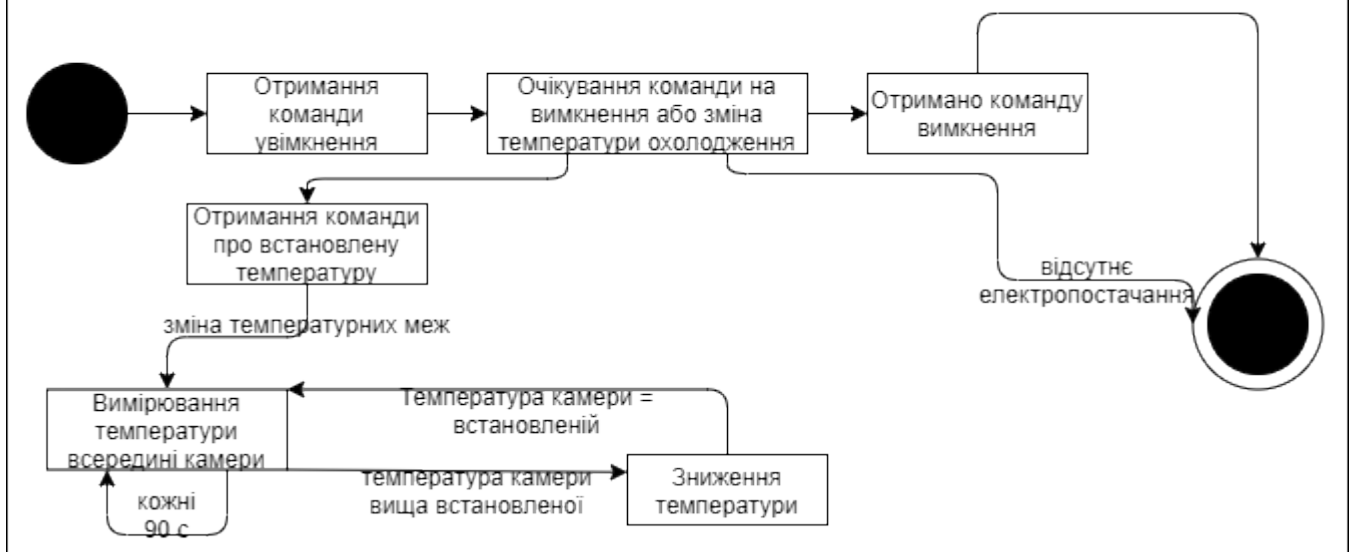
Підсистема управління



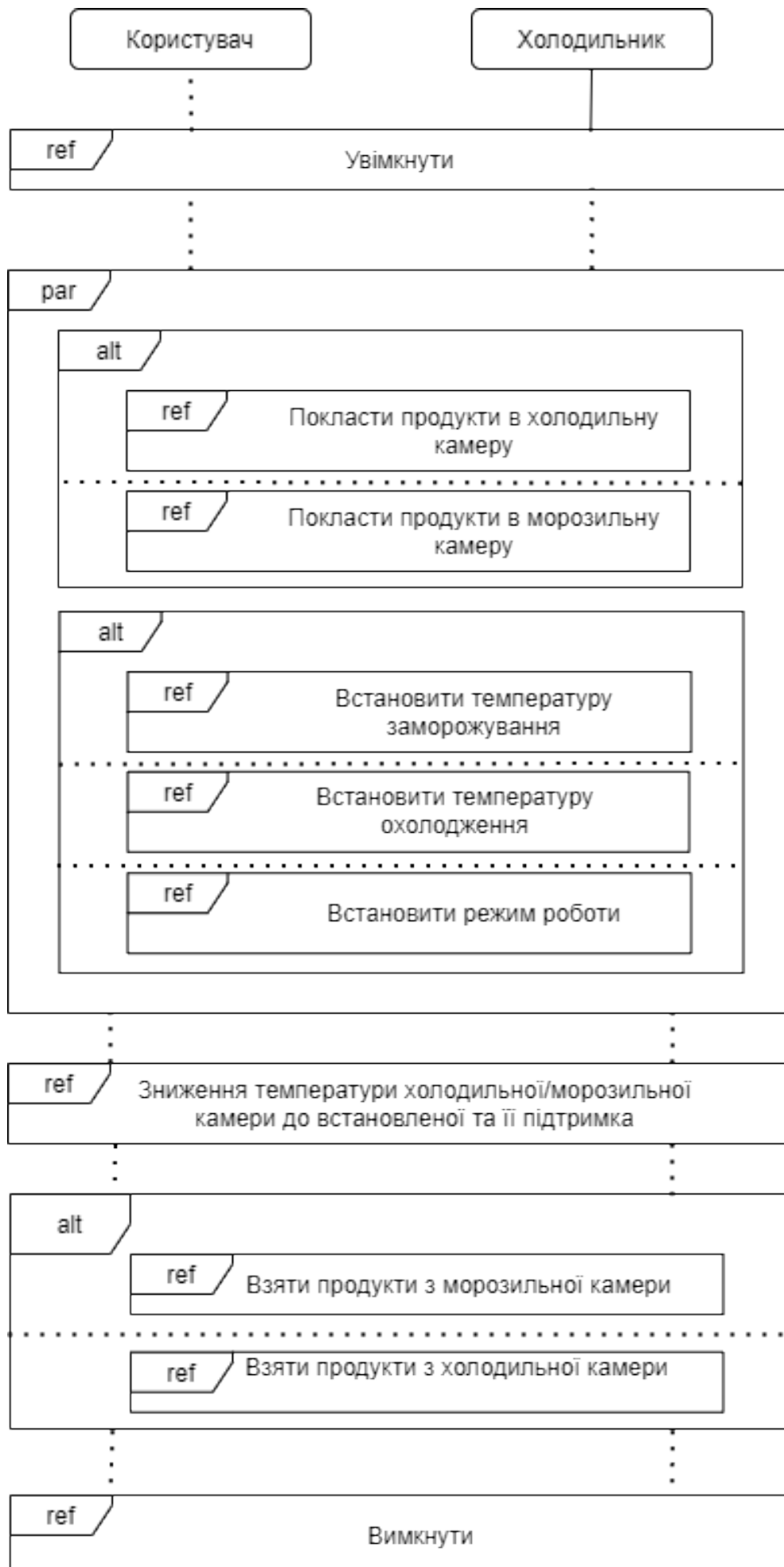
Підсистема охолодження

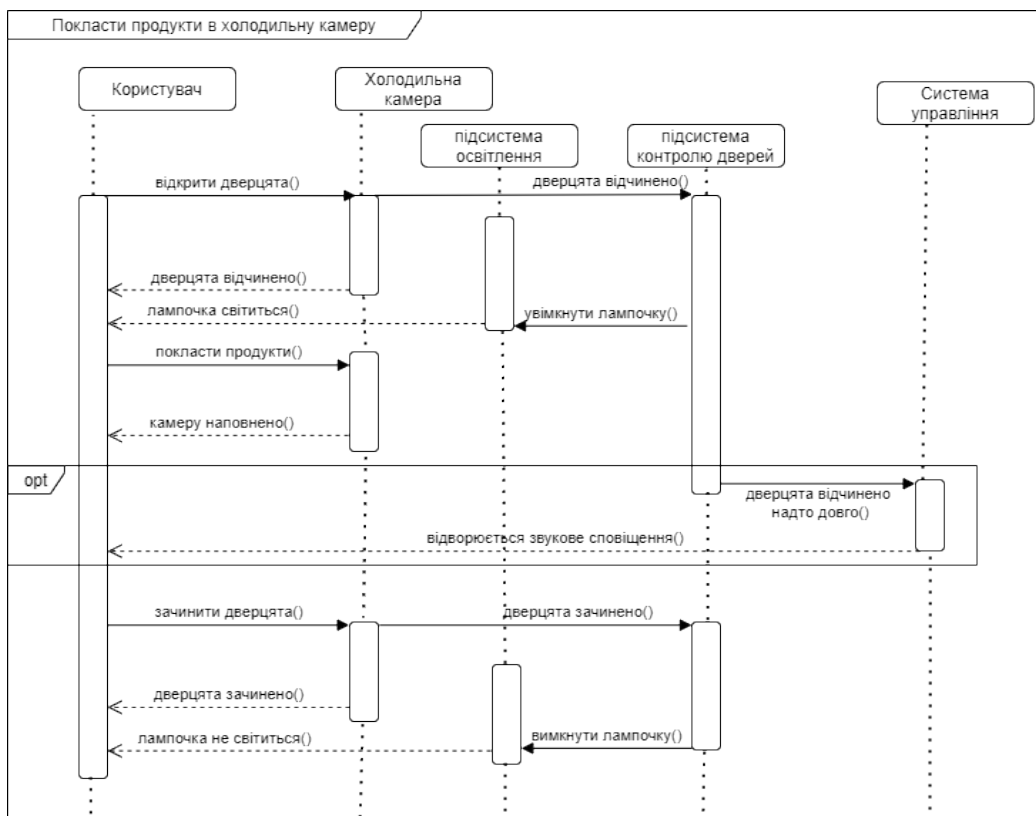
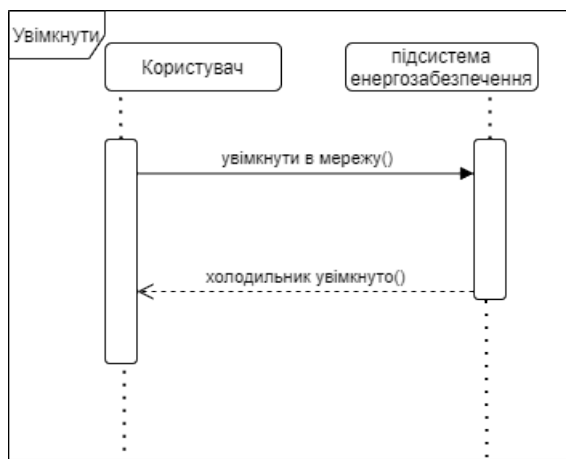


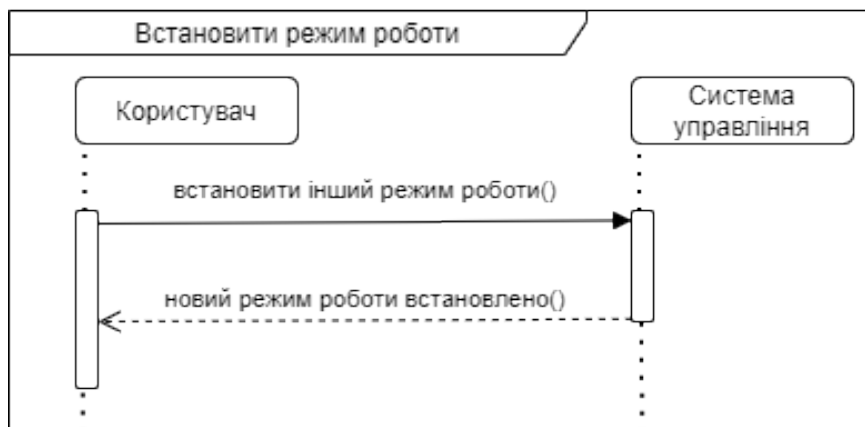
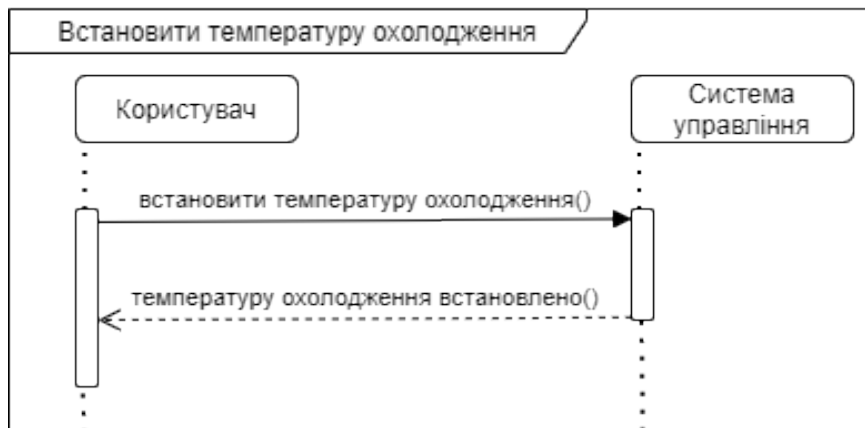
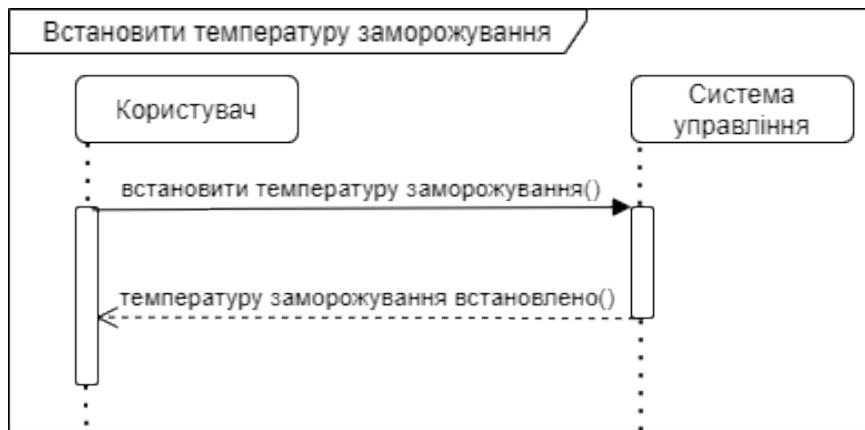
Підсистема заморожування



- sequences diagram



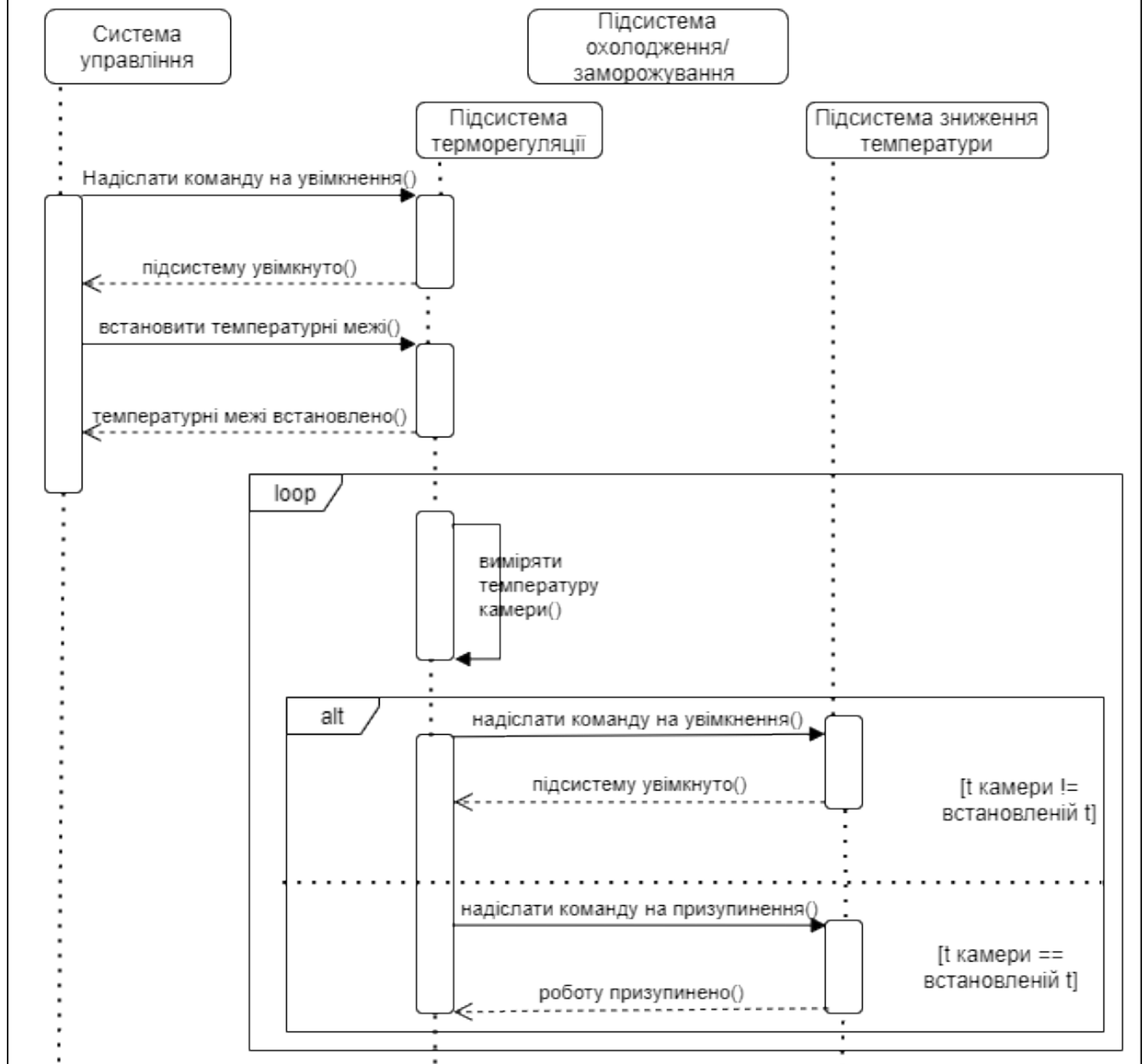


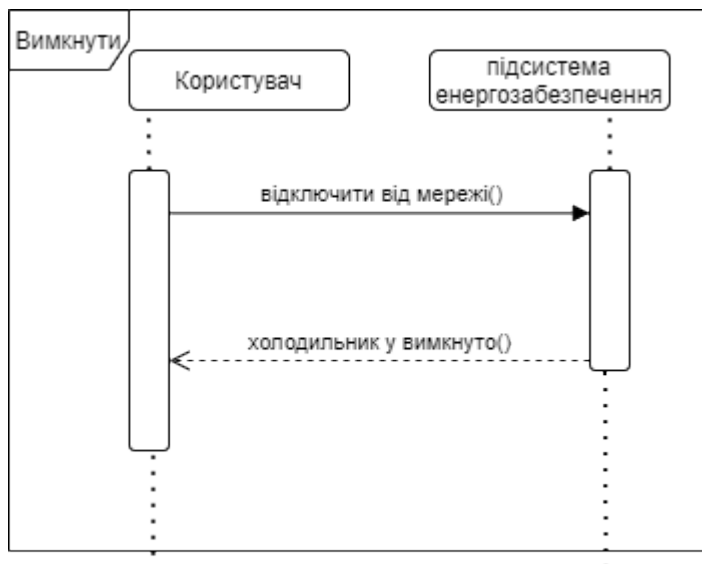
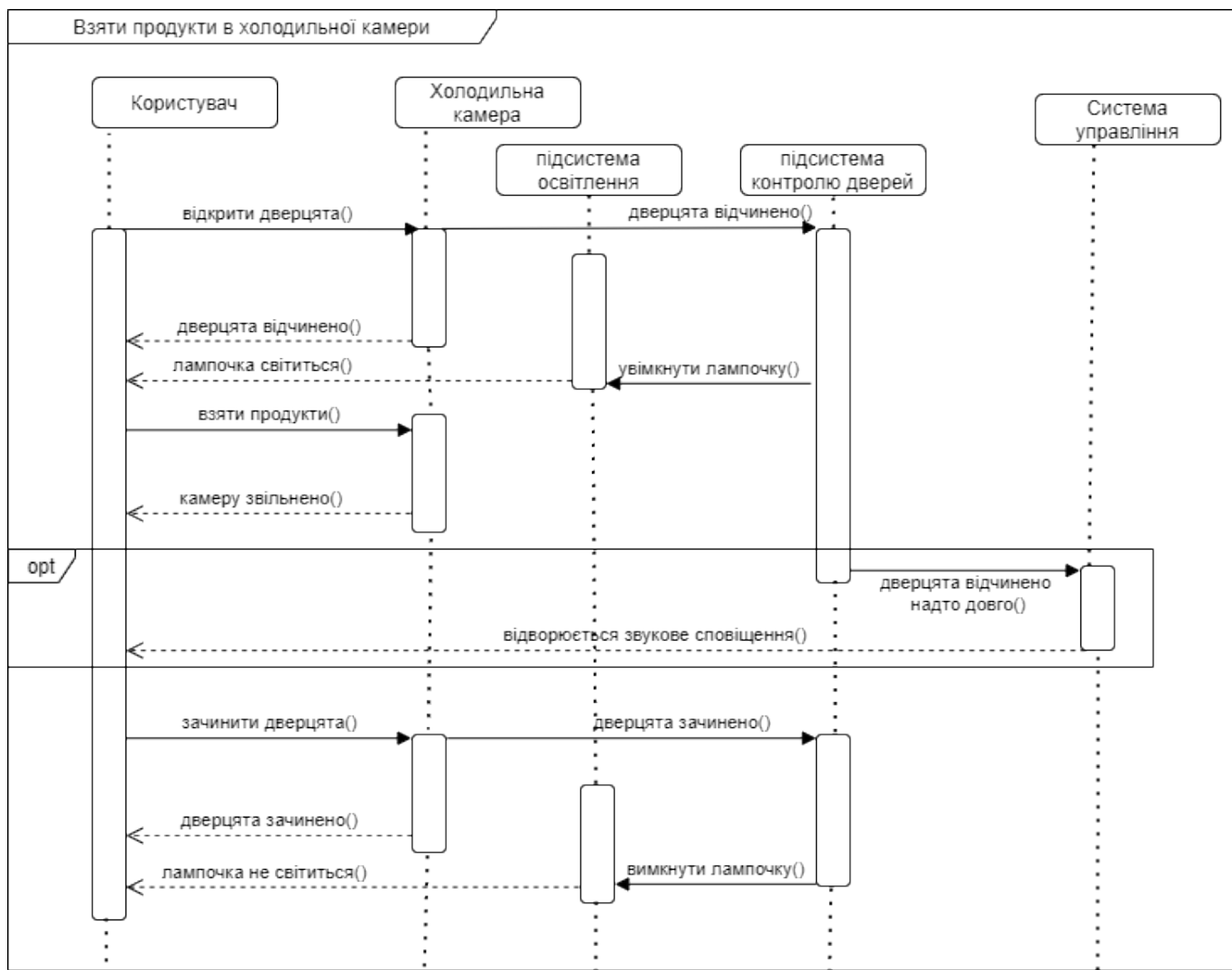


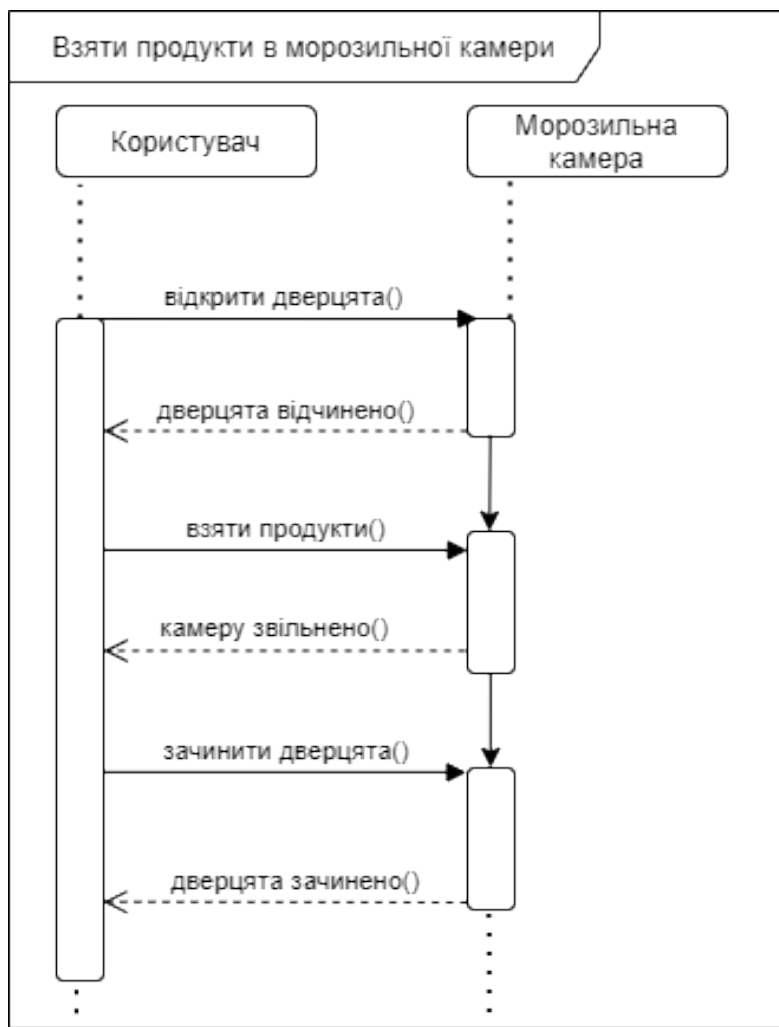
Покласти продукти в морозильну камеру



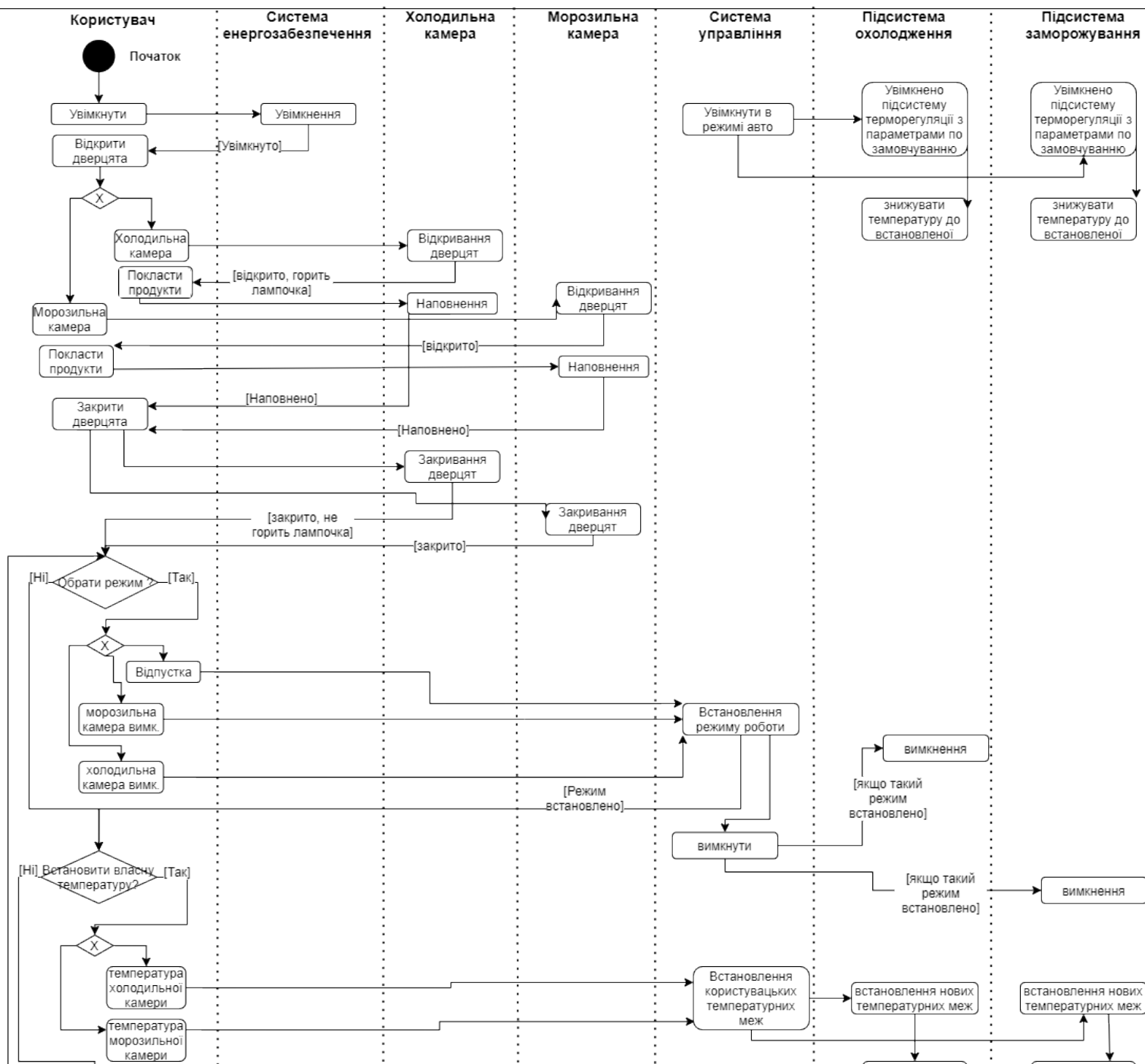
Зниження температури холодильної/морозильної камери до встановленої та підтримка її

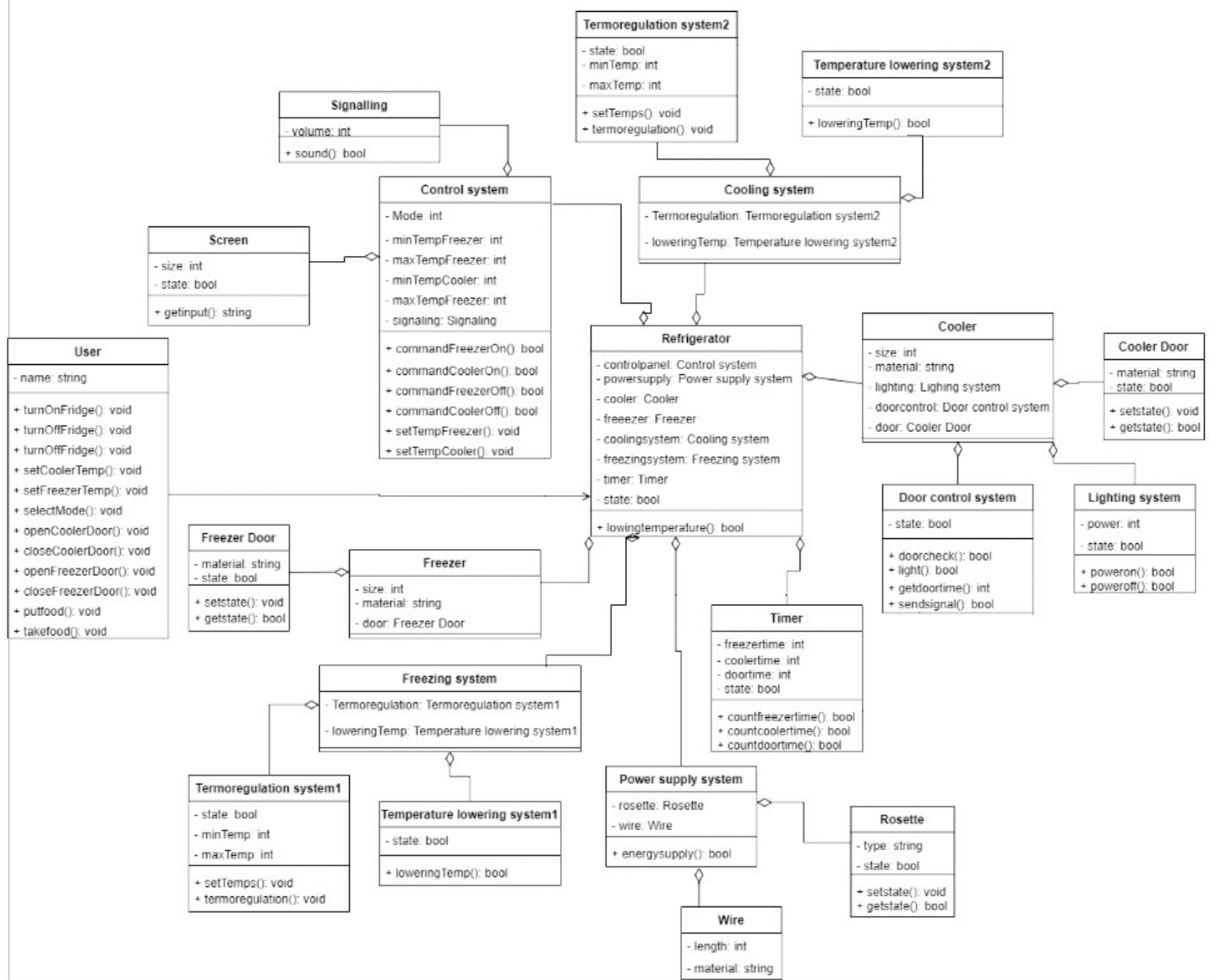






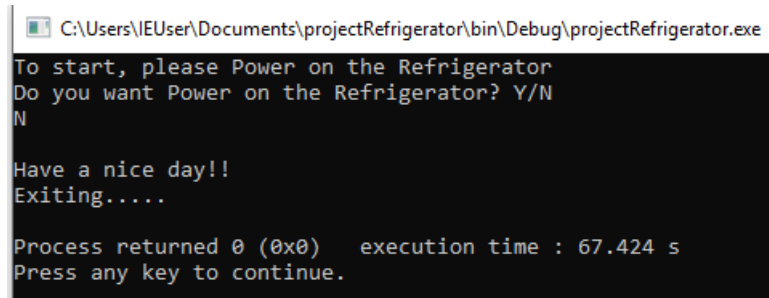
- activity diagram





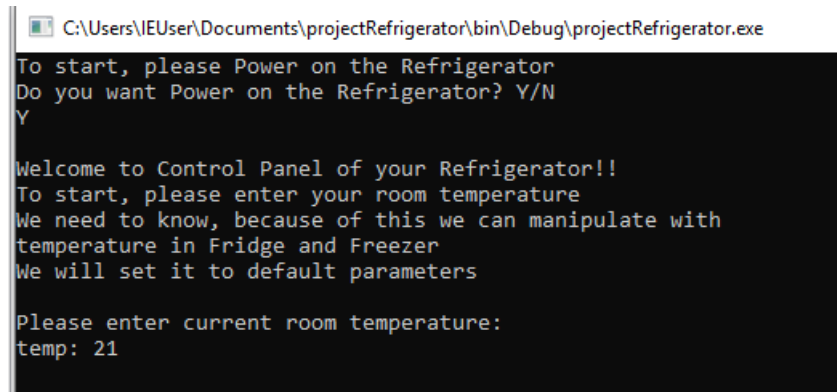
Скріншоти проекту Холодильник:

Початок програми, в користувача запитують чи бажає він включити холодильник. Якщо ні:



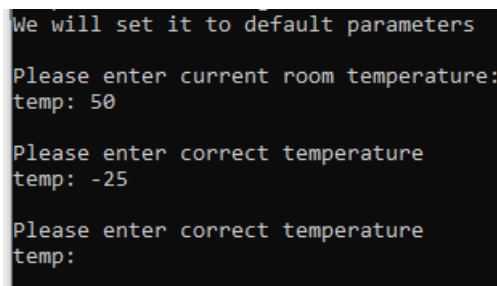
```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
To start, please Power on the Refrigerator
Do you want Power on the Refrigerator? Y/N
N
Have a nice day!!
Exiting.....
Process returned 0 (0x0)   execution time : 67.424 s
Press any key to continue.
```

Якщо так, то в користувача запитують температуру у кімнаті. Вводимо її

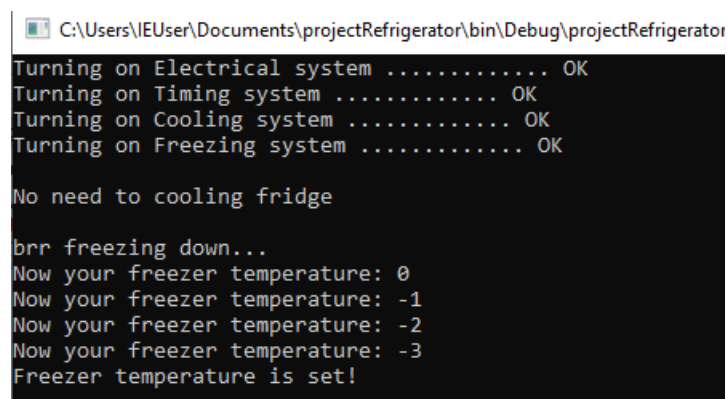


```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
To start, please Power on the Refrigerator
Do you want Power on the Refrigerator? Y/N
Y
Welcome to Control Panel of your Refrigerator!!
To start, please enter your room temperature
We need to know, because of this we can manipulate with
temperature in Fridge and Freezer
We will set it to default parameters
Please enter current room temperature:
temp: 21
```

Далі починається запуск основних систем та охолодження температури в холодильній та морозильній камері відповідно до значень за замовченням. Після врегулювання, користувача зустрічає інформаційна панель та варіанти дій: Варто зазначити, що критичні межі враховані, і в разі певних помилок виводиться відповідні повідомлення. У разі, якщо температура оточення менше за дефолтні значення, температура залишається такою як в оточенні.



```
We will set it to default parameters
Please enter current room temperature:
temp: 50
Please enter correct temperature
temp: -25
Please enter correct temperature
temp:
```



```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator
Turning on Electrical system ..... OK
Turning on Timing system ..... OK
Turning on Cooling system ..... OK
Turning on Freezing system ..... OK
No need to cooling fridge
brr freezing down...
Now your freezer temperature: 0
Now your freezer temperature: -1
Now your freezer temperature: -2
Now your freezer temperature: -3
Freezer temperature is set!
```

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Turning on Electrical system ..... OK
Turning on Timing system ..... OK
Turning on Cooling system ..... OK
Turning on Freezing system ..... OK

brn cooling down...
Now your fridge temperature: 21
Now your fridge temperature: 20
Now your fridge temperature: 19
Now your fridge temperature: 18
Now your fridge temperature: 17
Now your fridge temperature: 16
Now your fridge temperature: 15
Now your fridge temperature: 14
Now your fridge temperature: 13
Cooler temperature is set!

brn freezing down...
Now your freezer temperature: 21
Now your freezer temperature: 20
Now your freezer temperature: 19
Now your freezer temperature: 18
Now your freezer temperature: 17
Now your freezer temperature: 16
Now your freezer temperature: 15
Now your freezer temperature: 14
Now your freezer temperature: 13
Now your freezer temperature: 12
Now your freezer temperature: 11
Now your freezer temperature: 10
Now your freezer temperature: 9
Now your freezer temperature: 8
Now your freezer temperature: 7
Now your freezer temperature: 6
Now your freezer temperature: 5
Now your freezer temperature: 4
Now your freezer temperature: 3
Now your freezer temperature: 2
Now your freezer temperature: 1
Now your freezer temperature: 0
Now your freezer temperature: -1
Now your freezer temperature: -2
Now your freezer temperature: -3
Freezer temperature is set!
```

Головне меню:

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe

      REFRIGERATOR
    Made by Oleksii Chalyi FB-81

Here is information about Refrigerator:
Current fridge temperature: 13
Current freezer temperature: -3
Current fridge free space: 240
Current freezer free space: 130

Here is an action list of what you can do
1. Set fridge temperature
2. Set freezer temperature
3. Open fridge door
4. Open freezer door
5. Turn off the power and exit
```

1. Задаємо температуру холодильник. Нас вітає меню з можливістю задання температури для холодильника (cooler)

```
Выбрать C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Current Fridge temperature: 13
1. Enter Fridge temperature you want to set::
2. Go to ACTION LIST
```

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Current Fridge temperature: 12
1. Enter Fridge temperature you want to set:
2. Go to ACTION LIST
1
Setting Temperature: 7
brr cooling down...
current temp: 12
current temp: 11
current temp: 10
current temp: 9
current temp: 8
current temp: 7
```

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Current Fridge temperature: 7
1. Enter Fridge temperature you want to set:
2. Go to ACTION LIST
```

2. Задаємо температуру морозильнику. Нас вітає меню з можливістю задання температури для фризера.

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Current Freezer temperature: -4
1. Enter Freezer temperature you want to set:
2. Go to ACTION LIST
1
Setting Temperature: -13
brr freezing down...
current temp: -4
current temp: -5
current temp: -6
current temp: -7
current temp: -8
current temp: -9
current temp: -10
current temp: -11
current temp: -12
current temp: -13
```

Цікавий факт, згідно діаграм реалізована така штука як автоматична регульовка температури. Згідно з діаграмою вимог, кожні 90с проводиться регулювання температури. Користувача це сповіщає повідомлення.

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Current Fridge temperature: 7

1. Enter Fridge temperature you want to set:
2. Go to ACTION LIST
Your freezer and cooler temperature has been changed.
Returning it .....
```

Ясна річ, що в мені в інформаційному таблі, змінюється значення температура

```
Going to ACTION LIST...

      REFRIGERATOR
    Made by Oleksii Chalyi FB-81

Here is information about Refrigerator:
Current fridge temperature: 7
Current freezer temperature: -13
Current fridge free space: 240
Current freezer free space: 130

Here is an action list of what you can do
1. Set fridge temperature
2. Set freezer temperature
3. Open fridge door
4. Open freezer door
5. Turn off the power and exit
```

3. Відкриваємо холодильник. В нас загоряється лампочка та виводить інформацію про кількість місця. Для більше легкого розрахунку, було прийнято рішення, що 1 продукт займає 10 місця

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Fridge door is opened
Light: ON
Free space is: 240

1. Put some products
2. Take some products
3. Close the door
```

Якщо дверцята будуть занадто довго відкриті, буде звук. Згідно діаграм вимог, звук після 50 секунд:

```
Выбрать C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Fridge door is opened
Light: ON
Free space is: 240

1. Put some products
2. Take some products
3. Close the door
Beep Beep, Please close the door! BEEP
```

Кладемо в холодильник продукти:

Я обрав 23, і в мене залишилося 10 місця. Візьмемо продукти з холодильника:

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Fridge door is opened
Light: ON
Free space is: 10

1. Put some products
2. Take some products

3. Close the door
2
Enter the number of product you will take
12
freespace: 130
_
```

4. Відкриваємо морозильну камеру. Згідно діаграм вимог, в нас доступно 130 місця. Я поклав продукти на весь об'єм. Критичні значення враховані:

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Freezer door is opened
Free space is: 0

1. Put some products
2. Take some products

3. Close the door
2
Enter the number of product you will take: 14
you can't take so many objectss
```

Варто зазначити, що дані про об'єми морозильника і холодильника також оновлюються:

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
REFRIGERATOR
Made by Oleksii Chalyi FB-81

Here is information about Refrigerator:
Current fridge temperature: 7
Current freezer temperature: -13
Current fridge free space: 130
Current freezer free space: 0

Here is an action list of what you can do
1. Set fridge temperature
2. Set freezer temperature
3. Open fridge door
4. Open freezer door
5. Turn off the power and exit
```

5. Після завершення роботи з холодильником, вимикаємо його.

```
C:\Users\IEUser\Documents\projectRefrigerator\bin\Debug\projectRefrigerator.exe
Turning off Timing system ..... OK
Turning off Cooling system ..... OK
Turning off Freezing system ..... OK
Turning off Electrical system ..... OK

Have a nice day!!
Exiting.....

Process returned 0 (0x0)   execution time : 1707.865 s
Press any key to continue.
```

Висновок:

В результаті був створений проект Холодильник на основі усіх діаграм зазначених в цьому протоколі. Варто зазначити, що весь функціонал був дотриманий та реалізований. Значенні зазначені в діаграмі вимог також реалізовані в реальність. Однак, не можна сказати, що я не вносив деякі зміни. Насправді, я тільки розширив функціонал шляхом додавання температури середовища, оскільки необхідно було зробити автоматичну систему зниження температур, да і погодьтеся, виглядає ефектно:) .Відповідно задля розширення функціоналу було додано класи Environment та Point_to_Start. Також серед змін були додані функції до класу Refrigerator. В кінці кінців маємо консольний проект розроблений на C++ в CodeBlocks 20.03. Використовувалася віртуальна машина Windows10 на Oracle Virtual box 6.1

Код:

main.cpp

```
#include "Header.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{

    Menu();
    return 0;
}
```

Header.h

```
#pragma once
#include "windows.h"
#include <iostream>
#include <string>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <iterator>
#include <vector>
#include <windows.h>
#include <unistd.h>
#include <locale>
#include <chrono>
#include <ctime>
#include <thread>
#include <atomic>
using namespace std;

#define FRIDGE_CAP 240
#define DEF_FRIDGE_TEMP 13
#define FREEZER_CAP 130
#define DEF_FREEZER_TEMP -3

class CoolerDoor;
```

```

class Lightningsystem;
class Doorcontrolsystem;
class Cooler;
class FreezerDoor;
class Freezer;
class Signalling;
class Screen;
class ControlPanel;
class Wire;
class Rosette;
class PowerSupplySystem;
class TermoregulationSystem2;
class Temperatureloweringsystem2;
class CoolingSystem;
class Timer;
class Environment;
class Pont_of_Start;
class TermoregulationSystem1;
class Temperatureloweringsystem1;
class FreezingSystem;
class Refrigerator;
class User;

class CoolerDoor
{
public:
    CoolerDoor();
    void setstate();
    bool getstate();
private:
    string material;
    bool state;
};
CoolerDoor::CoolerDoor()
{
    state = false;
    material = "Zalizo";
}
bool CoolerDoor::getstate()
{
    cout << "State is get"<<endl;
    return state;
}
void CoolerDoor::setstate()
{
    if (state == true)
    {
        state = false;
        cout << "Cooler door is closed\n";
    }
    else
    {
        state = true;
        cout << "Cooler door is opened\n";
    }
}

class Lightningsystem

```



```

{
public:
    Lightningsystem();
    bool poweron();
    bool poweroff();
private:
    int power;
    bool state;
};

Lightningsystem::Lightningsystem()
{
    power = rand() % 30 + 210; // 210-240 V
    state = false;
}

bool Lightningsystem::poweron()
{
    cout << " Power is on" <<endl;
    return true;
}

bool Lightningsystem::poweroff()
{
    cout << " Power is off" <<endl;
    return true;
}

class Doorcontrolsystem
{
public:
    Doorcontrolsystem();
    bool doorcheck();
    bool light();
    int getdoortime();
    bool sendsignal();
private:
    bool state;
};

Doorcontrolsystem::Doorcontrolsystem()
{
    state = false;
}

bool Doorcontrolsystem::doorcheck()
{
    cout << "Door is ok"<<endl;
    return true;
}

bool Doorcontrolsystem::light()
{
    cout << "Light is on" <<endl;
    return true;
}

bool Doorcontrolsystem::sendsignal()

```

```

{
    cout << "Signal is sent"<<endl;
    return true;
}
int Doorcontrolsystem::getdoortime()
{
    cout << "Please close your door, it has already opened during:"<<rand()
    %70<<"seconds"<<endl;
    return rand()%70;
}

```

```

class Environment {
private:
    int temp = 0;
public:
    Environment() : temp(10) {};
    Environment(int t) : temp(t) {};
    int GetTemp() {
        return temp;
    }
};

```

```

class Pont_of_Start {
private:
    int freeSpace = 0;
    bool isDoorOpened = false;
    int currentTemp = 0;
    int wishedTemp = 0;
    bool isLightsOn = false;
    Environment* envir;
public:
    Pont_of_Start(int currentTemp) {
        this->currentTemp = currentTemp;
    };

    Pont_of_Start() {}

    int GetTemp() {
        return currentTemp;
    }

    void SetTemp(int t) {
        this->currentTemp = t;
    }

    void Pont_of_Start_(int currentTemp, Environment*
myEnvironmentTemperature) {
        this->currentTemp = myEnvironmentTemperature->GetTemp();
        this->wishedTemp = currentTemp;
        this->envir = myEnvironmentTemperature;
    }

    void SetVolume(int v) {
        this->freeSpace = v;
    }
}

```

```

}

int GetVolume() {
    return this->freeSpace;
}

bool DoorState() {
    return this->isDoorOpened;
}

void OpenDoor(bool open) {
    isDoorOpened = open;
}

void LightsOn(bool on) {
    isLightsOn = on;
}

bool LightsPowerState() {
    return this->isLightsOn;
}

int AddObject(int productNumber, int constMaxVolume) {
    int lim = 10 * productNumber;
    if (lim > constMaxVolume) {
        cout << "you cant place this many obj here\n";
        Sleep(2000);
        return -1;
    }
    else if (freeSpace < lim) {
        cout << "you cant place this many obj here\n";
        Sleep(2000);
        return -1;
    }
    else if (freeSpace >= 0 || freeSpace > lim) {
        freeSpace -= lim;
        cout << "freespace: " << freeSpace << endl;
    }
    else {
        return -1;
        cout << "error \n";
    }
    return freeSpace;
}

int RemObject(int productNumber, int constMaxVolume, int item_count) {
    int lim = 10 * productNumber;
    int over = freeSpace + lim;
    if (lim > constMaxVolume || over > constMaxVolume) {
        cout << "you can't take so many objectss\n";
        Sleep(2000);
        return -1;
    }
    else if (freeSpace < constMaxVolume) {
        freeSpace += lim;
        item_count += productNumber;
        cout << "freespace: " << freeSpace << endl;
    }
}

```

```

        else {
            return NULL;
            cout << "error \n";
        }
        return freeSpace;
    }

};

class Cooler : public Pont_of_Start {
private:
    int size = rand() % 10 + 130; // 130 - 135 L;
    string material = "Zalizo";
    Lightningsystem lightning;
    Doorcontrolsystem doorcontrol;
    CoolerDoor door;
public:
    Cooler() {}
    Cooler(int currentTemp, Environment* myEnvironmentTemperature) {
        Pont_of_Start_(currentTemp, myEnvironmentTemperature);
    }
};

class FreezerDoor
{
public:
    FreezerDoor();
    void setstate();
    bool getstate();
private:
    string material;
    bool state;
};

FreezerDoor::FreezerDoor()
{
    material = "zalizo";
    state = false;
}

bool FreezerDoor::getstate()
{
    cout << "State was get"<<endl;
    return state;
}

void FreezerDoor::setstate()
{
    if (state == true)
    {
        state = false;
        cout << "Door closed\n";
    }
    else
    {
        state = true;
        cout << "Door opened\n";
    }
}

class Freezer : public Pont_of_Start {

```

```

private:
    int size = rand() % 10 + 245; //245-255;
    string material = "Plastik";
    FreezerDoor door;
public:
    Freezer() {}
    Freezer(int currentTemp, Environment* myEnvironmentTemperature) {
        Pont_of_Start_(currentTemp, myEnvironmentTemperature);

    }
};

```

```

class Signalling
{
public:
    Signalling();
    bool sound();
private:
    int volume;
};

```

```

Signalling::Signalling()
{
    volume = 20;
}
bool Signalling::sound()
{
    cout << "Beep Beep Beep!"<<endl;
    return true;
}

```

```

class Screen
{
public:
    Screen();
    string getinput;
private:
    int size;
    bool state;
};
Screen::Screen()
{
    size = 10;
    // cout <<"Checking the vitality of screen"<<endl<<"Type something:\n";
    // cin >> getinput;
    // cout << "Ok, your text is:"<< getinput <<endl<<endl;
    state = false;
}

```

```

class ControlPanel {
protected:
    int Mode = 3;
    int minTempFreezer = rand() % 2 + (-26); //-24 - -26
    int maxTempFreezer = rand() % 2 + (-3); // -3 - -1
    int minTempCooler = rand() % 1 + 3; // 1 - 3
    int maxTempCooler = rand() % 2 + 13; // 13 - 15

```

```

    Signalling signalling;
    Screen screen;
        bool PowerOnStatus = false;
        string SystemName = "nosystemname";
public:
    string GetName() {
        return this->SystemName;
    }
    void SetPowerStatus(bool on) {
        this->PowerOnStatus = on;
    }
    bool GetPowerStatus() {
        return this->PowerOnStatus;
    }
    void TurnOn(string ControlPanelname, bool powerOnStatus) {
        if (powerOnStatus) {
            cout << "The " << ControlPanelname << " is already turned on\n";
        }
        else {
            SetPowerStatus(true);
            cout << "Turning on " << ControlPanelname << " ..... OK\n";
        }
    }
    void TurnOff(string ControlPanelname, bool powerOnStatus) {
        SetPowerStatus(false);
        cout << "Turning off " << ControlPanelname << " ..... OK\n";
    }
    void Working() {
        cout << this->SystemName << " working.....";
    }
};

```

```

class Wire
{
public:
    Wire();
private:
    int length;
    string material;
};
Wire::Wire()
{
    length = rand() % 1 + (1,5); // 1.5 - 2 m
    material = "coaxial";
}

```

```

class Rosette
{
public:
    Rosette();
    void setstate();
    bool getstate();
private:
    string type;
    bool state;
};
Rosette::Rosette()
{

```

```

        type = "A++";
        state = false;
    }
    bool Rosette::getstate()
    {
        return true;
    }
    void Rosette::setstate()
    {
        if (state == true)
        {
            state = false;
            cout << "Power is off\n";
        }
        else
        {
            state = true;
            cout << "Power is on\n";
        }
    }
}

class PowerSupplySystem : public ControlPanel {
private:
    Rosette rosette;
    Wire wire;
public:
    bool energysupply();
    PowerSupplySystem()
    {
        this->PowerOnStatus = false;
        this->SystemName = "Electrical system";
    }
};

bool PowerSupplySystem::energysupply()
{
    this->rosette.setstate();
    return true;
}

class TermoregulationSystem2
{
public:
    TermoregulationSystem2();
    void setTemps();
    void termoregulation();
private:
    bool state;
    int minTemp;
    int maxTemp;
};

TermoregulationSystem2::TermoregulationSystem2()
{
    state = false;
    minTemp = rand() % 1 + 3; // 1 - 3
    maxTemp = rand() % 2 + 13; // 13 - 15
}

void TermoregulationSystem2::setTemps()

```

```

{
cout << "Please, set min cooling temperature\n";
int a;
cin >> a;
if (a < rand() % 1 + 3)
{
    cout << "Error, your temp is less min that it can be\n";
}
else
{
    minTemp = a;
    cout << "Now your min temp is:" << minTemp << endl;
}
cout << "Please, set max cooling temperature\n";
int b;
cin >> b;
if (b > rand() % 2 + 13)
{
    cout << "Error, your temp is more max that it can be\n";
}
else
{
    maxTemp = b;
    cout << "Now your max temp is:" << maxTemp << endl;
}
}

void TermoregulationSystem2::termoregulation()
{
    cout << "Your min and max temperature now is:" << minTemp << maxTemp << endl;
    int a = rand() % 10; // random number of changing temperature
    maxTemp = maxTemp - a; //changing maxTemp during sleep
    minTemp = minTemp + a; //changing minTemp during sleep
    sleep(90);
    cout << "Now, your min and max temperature now is:" << minTemp << maxTemp << endl;
}

class Temperatureloweringsystem2
{
public:
    Temperatureloweringsystem2();
    bool loweringTemp();
private:
    bool state;
};

Temperatureloweringsystem2::Temperatureloweringsystem2()
{
    state = false;
}

bool Temperatureloweringsystem2::loweringTemp()
{
    cout << "Temperature is lowing" << endl;
    return true;
}

class CoolingSystem : public ControlPanel {
private:

```



```

TermoregulationSystem2 Termoregulation;
Temperatureloweringsystem2 loweringTemp;
public:
    CoolingSystem() {
        this->PowerOnStatus = false;
        this->SystemName = "Cooling system";
    }

    int SetTemp(bool poweredOn, Cooler &CoolerCamera, int t ) {
        if (poweredOn == false) {
            cout << "error\n";
        }
        else {
            int CurrentTemp = CoolerCamera.GetTemp();
            if (CurrentTemp == t) {
                cout << "Going to Control Panel\nentered value is a
current one\n";

                Sleep(300);
                return 0;
            }
            else if (CurrentTemp > t) {
                cout << "brr cooling down...\n";
                int newdefault1 = CurrentTemp - t;
                for (int i = 0; i <= newdefault1; i++) {
                    CoolerCamera.SetTemp(CurrentTemp--);
                    cout << "current temp: " <<
CoolerCamera.GetTemp() << endl;
                    Sleep(300);
                }
            }
            else if (CurrentTemp < t) {
                cout << "brr cooling up...\n";
                int newdefault2 = t - CurrentTemp;
                for (int i = 0; i <= newdefault2; i++) {
                    CoolerCamera.SetTemp(CurrentTemp++);
                    cout << "current temp: " <<
CoolerCamera.GetTemp() << endl;
                    Sleep(300);
                }
            }
            return CoolerCamera.GetTemp();
        }
    }

    int InitialCooling(bool poweredOn, Cooler &CoolerCamera, Environment*
myEnvironmentTemperature) {
        if (poweredOn==false) {
            cout << "error\n";
        }
        else {
            int EnteredTemp = myEnvironmentTemperature->GetTemp();
            int newdefault1 = EnteredTemp - DEF_FRIDGE_TEMP;
            CoolerCamera.SetTemp(EnteredTemp);
            if (DEF_FRIDGE_TEMP >= EnteredTemp) {
                cout << endl;
                cout << "No need to cooling fridge\n"<<endl;
                return 0;
            }
        }
    }

```

```

    }

    else {
        cout << endl << "brr cooling down... \n";
        for (int i = 0; i <= (newdefault1); i++) {
            CoolerCamera.SetTemp(EnteredTemp--);
            cout << "Now your fridge temperature: " <<
CoolerCamera.GetTemp() << endl;
            Sleep(200);
        }
        cout << "Cooler temperature is set!\n\n";
        cout << endl;
    }
}
}
};

```

```

class Timer : public ControlPanel
{
public:
    Timer();
    bool countfreezertime();
    bool countcoolertime();
    bool countdoortime();
    void setTimeout(auto function, int delay);
private:
    int freezertime;
    int coolertime;
    int doortime;
    bool state;
    atomic<bool> active{true};
};

Timer::Timer()
{
    state = false;
    freezertime = 90; //90 c
    coolertime = rand() % 10 + 80; //80 - 90 c
    doortime = rand() % 10 + 60; //60 - 70 c
    this->PowerOnStatus = false;
    this->SystemName = "Timing system";
    atomic<bool> active{true};
}

bool Timer::countfreezertime()
{
    if (freezertime >= 90)
        cout << "Freeze is ok. Checking Freeze Temperature"<<endl;
    return true;
}

bool Timer::countcoolertime()
{
    if (coolertime >= rand() % 10 + 80 )
        cout << "Cooler is ok. Checking Cooler Temperature"<<endl;
    return true;
}

```

```

bool Timer::countdoortime()
{
    if (doortime >= rand() % 10 + 60)
        cout << "Please close your door!"<<endl;
    return true;
}

void Timer::setTimeout(auto function, int delay) {
    active = true;
    thread t([=]() {
        if(!active.load()) return;
        this_thread::sleep_for(chrono::milliseconds(delay));
        if(!active.load()) return;
        function();
    });
    t.detach();
}

class TermoregulationSystem1
{
public:
    TermoregulationSystem1();
    void setTemps();
    void termoregulation();
private:
    bool state;
    int minTemp;
    int maxTemp;
};

TermoregulationSystem1::TermoregulationSystem1()
{
    state = false;
    minTemp = rand() % 2 + (-26); //-24 - -26
    maxTemp = rand() % 2 + (-3); // -3 - -1
}

void TermoregulationSystem1::setTemps()
{
    cout << "Please, set min freezing temperature\n";
    int a;
    cin >> a;
    if (a < rand() % 2 + (-26))
    {
        cout << "Error, your temp is more min that it can be\n";
    }
    else
    {
        minTemp = a;
        cout << "Now your min temp is:"<<minTemp<<endl;
    }
    cout << "Please, set max freezing temperature\n";
    int b;
    cin >> b;
    if (b > rand() % 2 + (-3))
    {
        cout << "Error, your temp is more max that it can be\n";
    }
}

```

```

else
{
maxTemp = b;
cout << "Now your max temp is:" << maxTemp << endl;
}
}
void TermoregulationSystem1::termoregulation()
{
cout << "Your min and max temperature now is:" << minTemp << maxTemp << endl;
int a = rand() % 10; // random number of changing temperature
maxTemp = maxTemp - a; //changing maxTemp during sleep
minTemp = minTemp + a; //changing minTemp during sleep
sleep(90);
cout << "Now, your min and max temperature now is:" << minTemp << maxTemp << endl;
}

```

```

class Temperatureloweringsystem1
{
public:
    Temperatureloweringsystem1();
    bool loweringTemp();
private:
    bool state;
};
Temperatureloweringsystem1::Temperatureloweringsystem1()
{
    state = false;
}
bool Temperatureloweringsystem1::loweringTemp()
{
    cout << "Temparute is lowing" << endl;
    return true;
}

```

```

class FreezingSystem : public ControlPanel {
private:
    TermoregulationSystem1 Termoregulation;
    Temperatureloweringsystem1 loweringTemp;
public:
    FreezingSystem() {
        this->PowerOnStatus = false;
        this->SystemName = "Freezing system";
    }
    int SetTemp(bool poweredOn, Freezer &FreezerCamera, int t) {
        if (poweredOn == false) {
            cout << "error\n";
        }
        else {
            int CurrentTemp = FreezerCamera.GetTemp();
            if (CurrentTemp == t) {
                cout << "Going to Control Panel\nentered value is a
current one\n";

                Sleep(300);
                return 0;
            }
            else if (CurrentTemp > t) {
                cout << "brr freezing down...\n";
                int newdefault1 = CurrentTemp - t;

```

```

        for (int i = 0; i <= newdefault1; i++) {
            FreezerCamera.SetTemp(CurrentTemp--);
            cout << "current temp: " <<
FreezerCamera.GetTemp() << endl;
            Sleep(300);
        }
    }
    else if (CurrentTemp < t) {
        cout << "brr freezing up...\n";
        int newdefault2 = t - CurrentTemp;
        for (int i = 0; i <= newdefault2; i++) {
            FreezerCamera.SetTemp(CurrentTemp++);
            cout << "current temp: " <<
FreezerCamera.GetTemp() << endl;
            Sleep(300);
        }
    }
    return FreezerCamera.GetTemp();
}

}

int InitialFreezing(bool poweredOn, Freezer &FreezerCamera, Environment*
myEnvironmentTemperature) {
    if (poweredOn==false) {
        cout << "error\n";
    }
    else {
        int EnteredTemp = myEnvironmentTemperature->GetTemp();
        int newdefault2 = EnteredTemp + abs(DEF_FREEZER_TEMP);
        FreezerCamera.SetTemp(EnteredTemp);
        if (DEF_FREEZER_TEMP >= EnteredTemp) {
            cout << endl;
            cout << "No need to freezing down\n";
            return 0;
        }
        else {
            cout << "brr freezing down... \n";
            EnteredTemp = myEnvironmentTemperature->GetTemp();
            for (int i = 0; i <= (newdefault2); i++) {
                FreezerCamera.SetTemp(EnteredTemp--);
                cout << "Now your freezer temperature: " <<
FreezerCamera.GetTemp() << endl;
                Sleep(200);
            }
            cout << "Freezer temperature is set!\n\n";
            cout << endl;
        }
    }
}

};

```

```

class Refrigerator {
private:
    string name = "null";
    int weight, length, width, height = 0;
public:

```

```

bool PowerOnStatus = false;
PowerSupplySystem powersystem;
CoolingSystem coolsys;
FreezingSystem freezsys;
Cooler CoolerCamera;
Freezer FreezerCamera;
Timer timer;
Environment* envir;
int c_item_count, f_item_count = 0;

```

```

Refrigerator(string name, int weight, int length, int width, int height, int
cooler_volume, int freezer_volume, Environment* myEnvironmentTemperature) {
    this->name = name;
    this->weight = weight;
    this->length = length;
    this->width = width;
    this->height = height;
    this->envir = myEnvironmentTemperature;
    CoolerCamera.SetVolume(cooler_volume);
    FreezerCamera.SetVolume(freezer_volume);
}

```

```

Refrigerator() {}

```

```

string GetName() {
    return name;
}

```

```

void SetName(string name) {
    this->name = name;
}

```

```

int GetWeight() {
    return weight;
}

```

```

void SetWeight(int weight) {
    this->weight = weight;
}

```

```

int GetLength() {
    return length;
}

```

```

void SetLength(int length) {
    this->length = length;
}

```

```

int GetWidth() {
    return width;
}

```

```

void get_temp_test() {
    int i = envir->GetTemp();
    cout << "Set temperature: " << i << endl;
}

```

```

void SetWidth(int width) {

```

```

        this->width = width;
    }

    int GetHeight() {
        return height;
    }

    void SetHeight(int height) {
        this->height = height;
    }

    int GetFridgeVol() {
        return CoolerCamera.GetVolume();
    }

    void SetFridgeVol(int cooler_volume) {
        CoolerCamera.SetVolume(cooler_volume);
    }

    int GetFreezerVol() {
        return FreezerCamera.GetVolume();
    }

    void SetFreezerVol(int freezer_vol) {
        FreezerCamera.SetVolume(freezer_vol);
    }

    void SetTimeout(auto function, int delay)
    {
timer.setTimeout(function, delay);
    }

    int Put_food_to_Cooler(int num) {
        CoolerCamera.AddObject(num, FRIDGE_CAP);
        c_item_count += num;
        return CoolerCamera.GetVolume();
    }

    int Put_food_to_Freezer(int num) {
        FreezerCamera.AddObject(num, FREEZER_CAP);
        f_item_count += num;
        return FreezerCamera.GetVolume();
    }

    int Take_food_from_Cooler(int productNumber) {
        CoolerCamera.RemObject(productNumber, FRIDGE_CAP,
c_item_count);
        c_item_count -= productNumber;
        return CoolerCamera.GetVolume();
    }

    int Take_food_from_Freezer(int productNumber) {
        FreezerCamera.RemObject(productNumber, FREEZER_CAP,
f_item_count);
        f_item_count -= productNumber;
        return FreezerCamera.GetVolume();
    }

    void OpenFridgeDoor() {
        CoolerCamera.OpenDoor(true);
    }

```

```

        CoolerCamera.LightsOn(true);

        cout << "Fridge door is opened" << endl;
        cout << "Light: ON" << endl;
        cout << "Free space is: " << CoolerCamera.GetVolume() << endl;

    }

    void OpenFreezerDoor() {
        FreezerCamera.OpenDoor(true);
        FreezerCamera.LightsOn(true);
        cout << "Freezer door is opened" << endl;
        cout << "Free space is: " << FreezerCamera.GetVolume() << endl;
    }

    bool GetPowerStatus() {
        return this->PowerOnStatus;
    }

    void SetPowerStatus(bool on) {
        this->PowerOnStatus = on;
    }

    void StartSystems() {
        powersystem.TurnOn(powersystem.GetName(),
powersystem.GetPowerStatus());
        Sleep(300);
        timer.TurnOn(timer.GetName(), timer.GetPowerStatus());
        Sleep(300);
        coolsys.TurnOn(coolsys.GetName(), coolsys.GetPowerStatus());
        Sleep(300);
        freezsys.TurnOn(freezsys.GetName(), freezsys.GetPowerStatus());
        Sleep(300);
        coolsys.InitialCooling(coolsys.GetPowerStatus(), CoolerCamera, envir);
        freezsys.InitialFreezing(freezsys.GetPowerStatus(), FreezerCamera,
envir);
    }

    void StopSystems() {
        timer.TurnOff(timer.GetName(), timer.GetPowerStatus());
        Sleep(300);
        coolsys.TurnOff(coolsys.GetName(), coolsys.GetPowerStatus());
        Sleep(300);
        freezsys.TurnOff(freezsys.GetName(), freezsys.GetPowerStatus());
        Sleep(300);
        powersystem.TurnOff(powersystem.GetName(),
powersystem.GetPowerStatus());
        Sleep(300);
    }

};

class User
{
public:
    User();
    void turnOnFridge(Refrigerator* rf);
    void turnOffFridge(Refrigerator* rf);

```



```

    void setCoolerTemp(Refrigerator* rf);
    void setFreezerTemp(Refrigerator* rf);
    void selectMode(Refrigerator* rf);
    void openCoolerDoor(Refrigerator* rf);
    void closeCoolerDoor(Refrigerator* rf);
    void openFreezerDoor(Refrigerator* rf);
    void closeFreezerDoor(Refrigerator* rf);
    void putfood(Refrigerator* rf);
    void takefood(Refrigerator* rf);
private:
    string name;
};
User::User()
{
    name = "Aleksey";
}
void User::turnOnFridge(Refrigerator* rf)
{
    rf->StartSystems();
}
void User::turnOffFridge(Refrigerator* rf)
{
    rf->StopSystems();
}
void User::setCoolerTemp(Refrigerator* rf)
{
    rf->get_temp_test();
}
void User::setFreezerTemp(Refrigerator* rf)
{
    rf->get_temp_test();
}
void User::selectMode(Refrigerator* rf)
{
    // rf->SetMode();
}
void User::closeCoolerDoor(Refrigerator* rf)
{
    rf->OpenFridgeDoor();
}
void User::openCoolerDoor(Refrigerator* rf)
{
    rf->OpenFridgeDoor();
}
void User::openFreezerDoor(Refrigerator* rf)
{
    rf->OpenFreezerDoor();
}
void User::closeFreezerDoor(Refrigerator* rf)
{
    rf->OpenFridgeDoor();
}
void User::putfood(Refrigerator* rf)
{
    // rf->Put_food_to_Freezer();
    // rf->Put_food_to_Freezer();
}

```

```

void User::takefood(Refrigerator* rf)
{
//  rf-> Take_food_from_Cooler();
//  rf ->Take_food_from_Freezer();
}

```

```

void Menu() {
    User us;
    Environment room_temperature(10);
    int main_menu_choice = 1;
    int choice = 1;
    int user_temp = 0;
    int wanted_temperature = 0;
    Refrigerator Gorenje("Gorenje", 75, 75, 65, 201, 240, 130, &room_temperature);
    cout << "To start, please Power on the Refrigerator\n";
    char ch;
    cout << "Do you want Power on the Refrigerator? Y/N\n";
    while (1) {
        cin >> ch;
        if (ch == 'y' or ch == 'Y') {
            cout << "\nWelcome to Control Panel of your Refrigerator!!\n";
            cout << "To start, please enter your room temperature\n";
            cout << "We need to know, because of this we can manipulate with\n";
            cout << "ntemperature in Fridge and Freezer\n";
            cout << "We will set it to default parameters\n\n";
            cout << "Please enter current room temperature: \n";
            while (1) {
                cout << "temp: ";
                cin >> user_temp;
                cout << endl;
                if (user_temp > -10 && user_temp < 45) {
                    room_temperature = Environment(user_temp);
                    cout << "Ok, your the temperature in your room is: ";
                    cout << user_temp << endl << endl;
                    Sleep(500);
                    break;
                }
                else {
                    cout << "Please enter correct temperature\n";
                    Sleep(500);
                }
            }
            system("cls");
            Gorenje.SetPowerStatus(true);
            Gorenje.StartSystems();
            break;
        }
        else if (ch == 'n' or ch == 'N') {
            cout << "\nHave a nice day!!\n";
            cout << "Exiting.....\n";
            Sleep(500);
            exit(0);
        }
    }

    while (choice != 0) {
        cout << endl;
        cout << endl;
        menu_ptr:
    }
}

```

```

        cout << "          REFRIGERATOR\n    Made by Oleksii Chalyi FB-81";
        cout << endl;
        cout << endl;
        cout << "Here is information about Refrigerator:"<<endl;
        cout << "Current fridge temperature: " <<
Gorenje.CoolerCamera.GetTemp() << endl;
        cout << "Current freezer temperature: " <<
Gorenje.FreezerCamera.GetTemp() << endl;
        cout << "Current fridge free space: "
<<Gorenje.CoolerCamera.GetVolume()<<endl;
        cout << "Current freezer free space: "
<<Gorenje.FreezerCamera.GetVolume()<<endl;
        cout << endl;
        cout << "Here is an action list of what you can do"<<endl;
        cout << "1. Set fridge temperature\n";
        cout << "2. Set freezer temperature\n";
        cout << "3. Open fridge door \n";
        cout << "4. Open freezer door \n";
        cout << "5. Turn off the power and exit\n";
        cout << endl;
        Gorenje.SetTimeout([&]() {
            cout << "Your freezer and cooler temperature has been changed. \nReturning it
....."<< endl;
            }, 90000); //90 c
            cin >> choice;
            cout << "You choose: " << choice << endl;
            switch (choice) {
            case 5:
                system("cls");
                Gorenje.StopSystems();
                cout << "\nHave a nice day!!\n";
                cout << "Exiting.....\n";
                exit(0);
                break;

            case 1:
                while (main_menu_choice == 1) {
                    system("cls");
                    cout << "Current Fridge temperature: " <<
Gorenje.CoolerCamera.GetTemp() << endl;
                    cout << endl;
                    cout << "1. Enter Fridge temperature you want to set: \n";
                    cout << "2. Go to ACTION LIST\n";
                    cin >> choice;
                    switch (choice) {
                    case 1:
                        cout << "\nSetting Temperature: ";
                        cin >> wanted_temperature;
                        while ((wanted_temperature <= 1) ||
(wanted_temperature >= 13)) {
                            cout << "Something going wrong. Try to set
Temperature between 1 and 13\n";
                            cin >> wanted_temperature;
                        }
                        Gorenje.coolsys.SetTemp(true,
Gorenje.CoolerCamera, wanted_temperature);
                        break;
                    case 2:
                        cout << "Going to ACTION LIST...\n";

```

```

        Sleep(100);
        cout << endl;
        goto menu_ptr;
    }
}

case 2:
    while (main_menu_choice == 1) {
        system("cls");
        cout << "Current Freezer temperature: " <<
Gorenje.FreezerCamera.GetTemp() << endl;
        cout << endl;
        cout << "1. Enter Freezer temperature you want to set: \n";
        cout << "2. Go to ACTION LIST\n";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "\nSetting Temperature: ";
                cin >> wanted_temperature;
                while ((wanted_temperature <= -26) ||
(wanted_temperature >= -3)) {
                    cout << "Something going wrong. Try to set
Temperature between -26 and -3\n";
                    cin >> wanted_temperature;
                }
                Gorenje.freezsys.SetTemp(true,
Gorenje.FreezerCamera, wanted_temperature);
                break;
            case 2:
                cout << "Going to ACTION LIST...\n";
                Sleep(100);
                cout << endl;
                goto menu_ptr;
            }
        }

case 3:
    while (main_menu_choice == 1) {
        system("cls");
        Gorenje.OpenFridgeDoor();
        cout << endl;
        cout << endl;
        cout << "1. Put some products \n";
        cout << "2. Take some products \n";
        cout << endl;
        cout << "3. Close the door\n";
        int number = 0;
        Gorenje.SetTimeout([&]() {
cout << "Beep Beep, Please close the door! BEEP" << endl;
}, 70000); //70 c
        cin >> choice;
        switch (choice) {
            case 1:
                if (Gorenje.CoolerCamera.GetVolume() == 0) {
                    cout << "Fridge is full\n";
                    break;
                }

```

```

        cout << "Enter the number of product you will put\n";
n";

        cin >> number;
        if (Gorenje.CoolerCamera.GetVolume() > 0) {
            Gorenje.Put_food_to_Cooler(number);
            Sleep(2000);
            Gorenje.CoolerCamera.GetVolume();
            break;
        }
    case 2:
        if (Gorenje.CoolerCamera.GetVolume() ==
FRIDGE_CAP) {

            cout << "Fridge is empty\n";
            Sleep(2000);
            break;
        }
        else if ((Gorenje.CoolerCamera.GetVolume() >= 0)
&& (Gorenje.CoolerCamera.GetVolume() <= 240)) {
            cout << "Enter the number of product you
will take\n";

            cin >> number;
            Gorenje.Take_food_from_Cooler(number);
            Gorenje.CoolerCamera.GetVolume();
            Sleep(2000);
        }
        break;
    case 3:
        Gorenje.CoolerCamera.OpenDoor(false);
        cout << "Door is closed\n";
        Sleep(2000);
        system("cls");
        goto menu_ptr;
    }
}

case 4:
    while (main_menu_choice == 1) {

        system("cls");
        Gorenje.OpenFreezerDoor();
        cout << endl;
        cout << endl;
        cout << "1. Put some products \n";
        cout << "2. Take some products \n";
        cout << endl;
        cout << "3. Close the door\n";
        int number = 0;
        cin >> choice;
        switch (choice) {
        case 1:
            if (Gorenje.FreezerCamera.GetVolume() == 0) {
                cout << "Freezer is full\n";
                break;
            }
            cout << "Enter the number of product you will put:
";

            cin >> number;
            if (Gorenje.FreezerCamera.GetVolume() > 0) {

```

```

        Gorenje.Put_food_to_Freezer(number);
        Sleep(2000);
        Gorenje.FreezerCamera.GetVolume();
        break;
    }
case 2:
    if (Gorenje.FreezerCamera.GetVolume() ==
FREEZER_CAP) {
        cout << "Freezer is empty\n";
        Sleep(2000);
        break;
    }
    else if ((Gorenje.FreezerCamera.GetVolume() >= 0)
&& (Gorenje.FreezerCamera.GetVolume() <= 130)) {
        cout << "Enter the number of product you
will take: ";

        cin >> number;
        Gorenje.Take_food_from_Freezer(number);
        Gorenje.FreezerCamera.GetVolume();
        Sleep(2000);
    }
    break;
case 3:
    Gorenje.FreezerCamera.OpenDoor(false);
    cout << "Door is closed\n";
    Sleep(2000);
    system("cls");
    goto menu_ptr;
    }
    }
    }
}
}

```