

4IT Erasmus+Project 2024-1-RO01-KA121-VET-000214264
Beneficiary Colegiul National Gheorghe Vranceanu, Bacau, Romania

Sending organization: Colegiul National ”Roman-Vodă” Roman, Romania

MOBILITY PORTFOLIO
Paphos, Cyprus

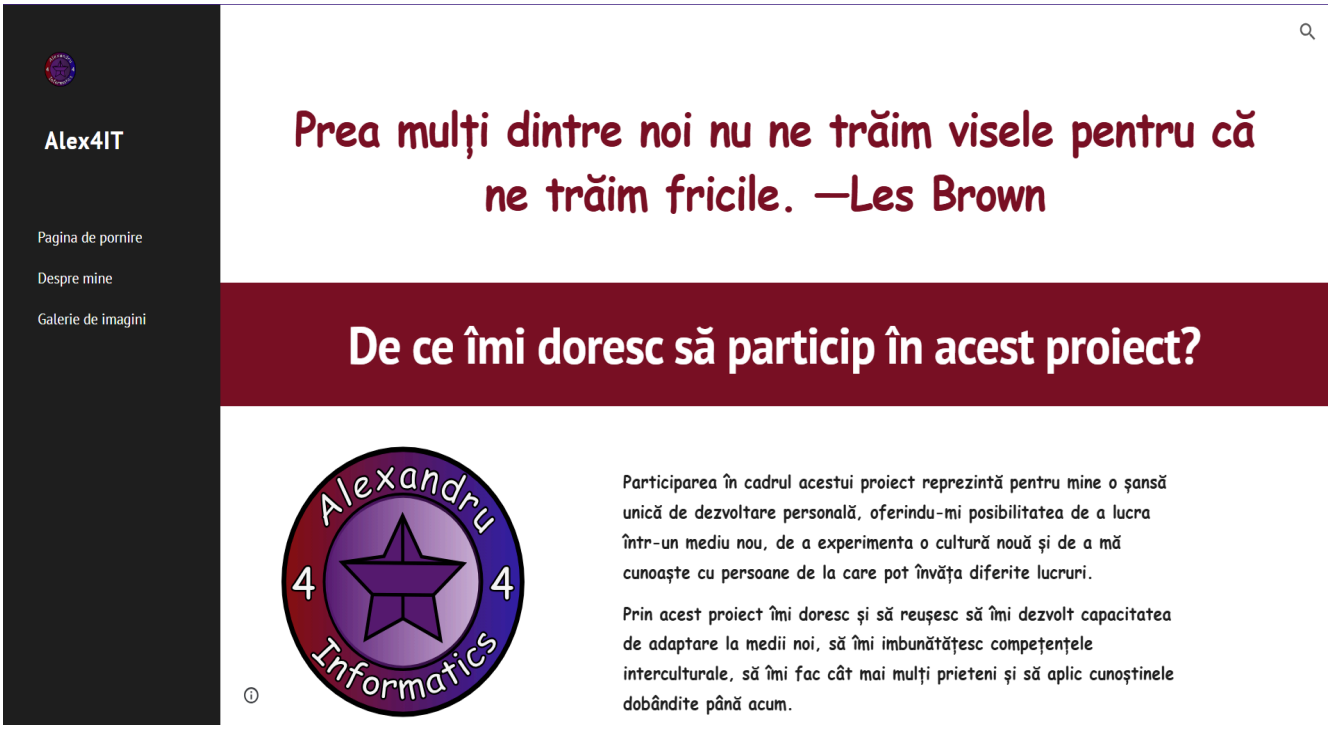
Mobility period:
PAPHOS, CYPRUS, 13.04.2025-27.04.2025 – practice period 14.04.2025-25.04.2025

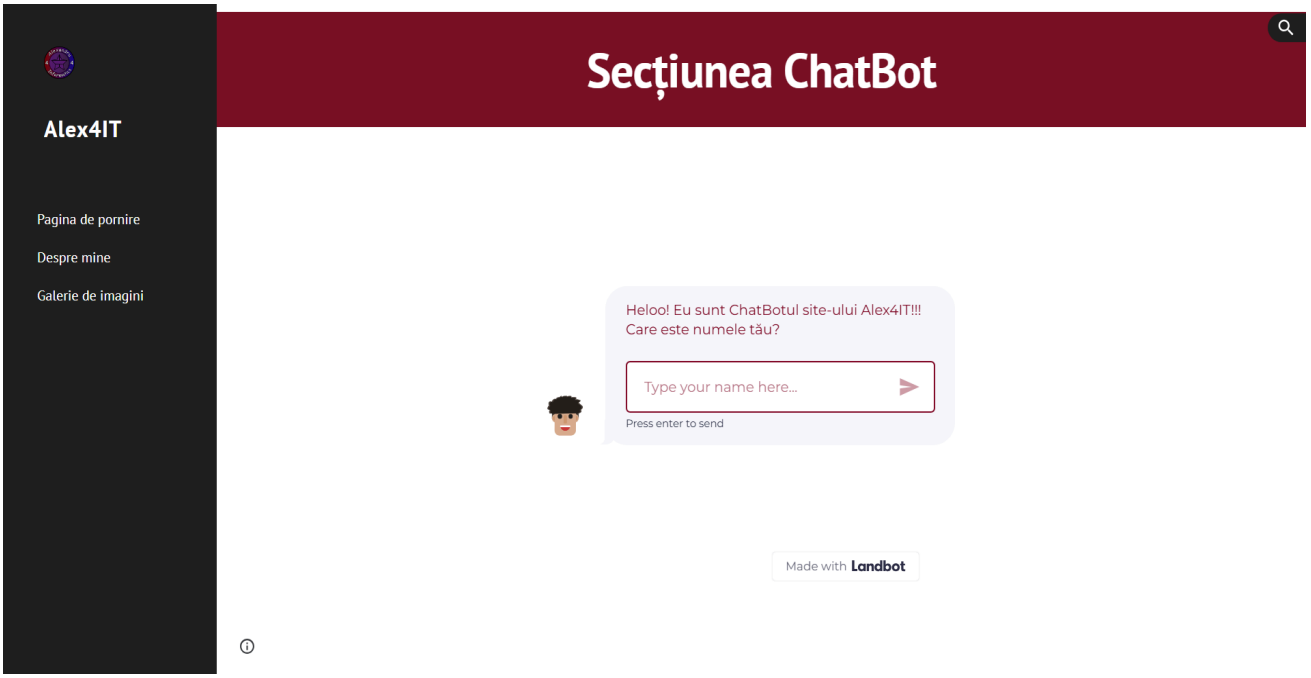
Receiving

Practice tutor: Lazaros Lazarou

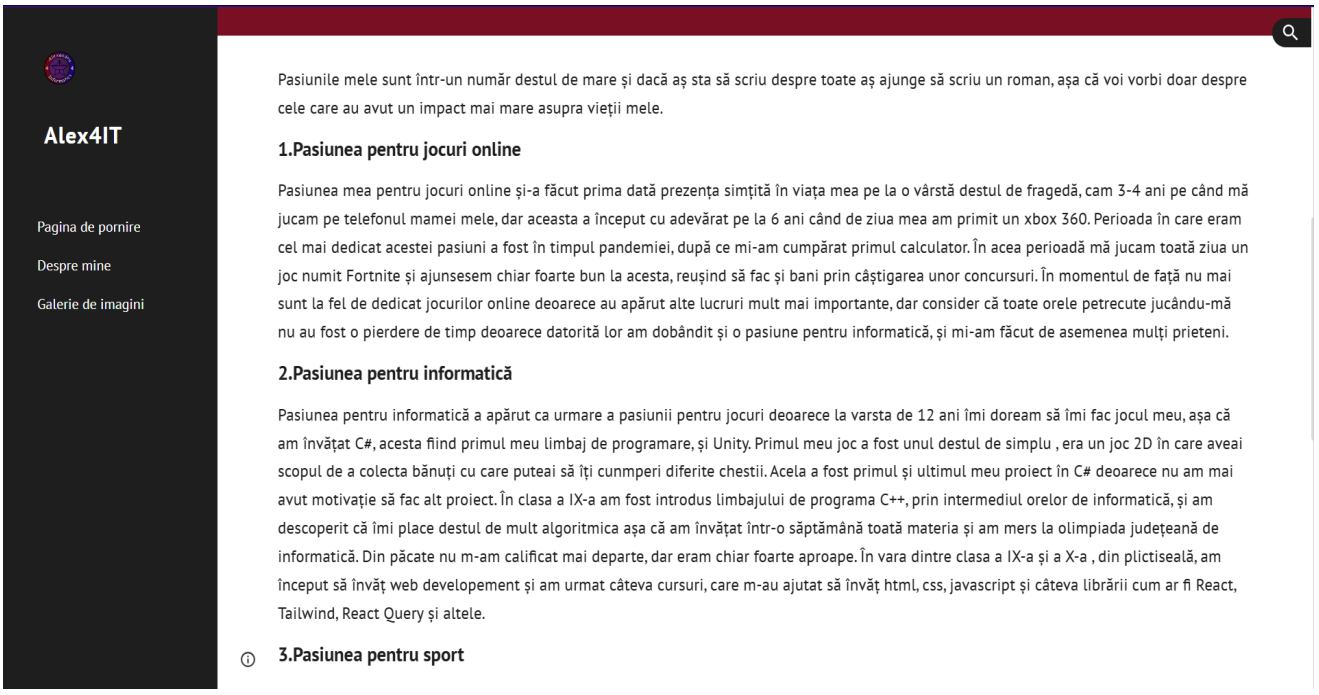
Participant: ENEA ALEXANDRU-PETRONEL

DISCLAIMER: The European Commission’s support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.


Alex4IT
Pagina de pornire
Despre mine
Galerie de imagini
Prea mulți dintre noi nu ne trăim visele pentru că ne trăim fricile. —Les Brown
De ce îmi doresc să particip în acest proiect?
Alexandru
4
4
Informatics
Participarea în cadrul acestui proiect reprezintă pentru mine o șansă unică de dezvoltare personală, oferindu-mi posibilitatea de a lucra într-un mediu nou, de a experimenta o cultură nouă și de a mă cunoaște cu persoane de la care pot învăța diferite lucruri.
Prin acest proiect îmi doresc și să reușesc să îmi dezvolt capacitatea de adaptare la medii noi, să îmi imbunătățesc competențele interculturale, să îmi fac cât mai mulți prieteni și să aplic cunoștinele dobândite până acum.


Alex4IT
Pagina de pornire
Despre mine
Galerie de imagini
Secțiunea ChatBot
Heloo! Eu sunt ChatBotul site-ului Alex4IT!!!
Care este numele tău?
Type your name here...
Press enter to send
Made with Landbot


Alex4IT
Pagina de pornire
Despre mine
Galerie de imagini
Despre mine
Cine sunt eu?
Mă numesc Enea Alexandru, am 17 ani și sunt elev în clasa a XI-a B la Colegiul Național "Roman-Vodă". Sunt o persoană inteligentă, deoarece învăț foarte rapid, prietenoasă, având mereu scopul de a fi prieten cu toată lumea, calmă, harnică, empatică, perseverentă, deoarece atunci când îmi setez un scop nu mă las până nu îl îndeplinesc, și flexibilă fiind capabill să mă adaptez la diferite situații destul de ușor.
Pasiunile mele


Alex4IT
Pagina de pornire
Despre mine
Galerie de imagini
Pasiunile mele sunt într-un număr destul de mare și dacă aș sta să scriu despre toate aș ajunge să scriu un roman, așa că voi vorbi doar despre cele care au avut un impact mai mare asupra vieții mele.
1.Pasiunea pentru jocuri online
Pasiunea mea pentru jocuri online și-a făcut prima dată prezența simțită în viața mea pe la o vârstă destul de fragedă, cam 3-4 ani pe când mă jucam pe telefonul mamei mele, dar aceasta a început cu adevărat pe la 6 ani când de ziua mea am primit un xbox 360. Perioada în care eram cel mai dedicat acestei pasiuni a fost în timpul pandemiei, după ce mi-am cumpărat primul calculator. În acea perioadă mă jucam toată ziua un joc numit Fortnite și ajunsesem chiar foarte bun la acesta, reușind să fac și bani prin câștigarea unor concursuri. În momentul de față nu mai sunt la fel de dedicat jocurilor online deoarece au apărut alte lucruri mult mai importante, dar consider că toate orele petrecute jucându-mă nu au fost o pierdere de timp deoarece datorită lor am dobândit și o pasiune pentru informatică, și mi-am făcut de asemenea mulți prieteni.
2.Pasiunea pentru informatică
Pasiunea pentru informatică a apărut ca urmare a pasiunii pentru jocuri deoarece la varsta de 12 ani îmi doream să îmi fac jocul meu, așa că am învățat C#, acesta fiind primul meu limbaj de programare, și Unity. Primul meu joc a fost unul destul de simplu , era un joc 2D în care aveai scopul de a colecta bănuți cu care puteai să îți cunmperi diferite chestii. Acela a fost primul și ultimul meu proiect în C# deoarece nu am mai avut motivație să fac alt proiect. În clasa a IX-a am fost introdus limbajului de programa C++, prin intermediul orelor de informatică, și am descoperit că îmi place destul de mult algoritmica așa că am învățat într-o săptămână toată materia și am mers la olimpiada județeană de informatică. Din păcate nu m-am calificat mai departe, dar eram chiar foarte aproape. În vara dintre clasa a IX-a și a X-a , din plictiseală, am început să învăț web developement și am urmat câteva cursuri, care m-au ajutat să învăț html, css, javascript și câteva librării cum ar fi React, Tailwind, React Query și altele.
3.Pasiunea pentru sport

Alex4IT

Pagina de pornire

Despre mine

Galerie de imagini

3.Pasiunea pentru sport

Pasiunea pentru sport a început de la 7 ani când am început să practic Qwan Ki Do, în cadrul clubului sportiv Bao Roman. După 2 ani acest sport a început să nu mă m-ai atragă, așa că am decis că nu mai are rost să continui. Între anii 2017-2019 am jucat football la echipa școlii, unde am câștigat și câteva premii. După ce a început pandemia, am început să duc un stil de viață nesănătos, aveam o dietă foarte proastă și stăteam toată ziua în casă fără să fac mișcare, așa că într-o zi am decis să îmi cumpăr un abonament la sală, și acest lucru mi-a schimbat extraordinar de mult viața. Cu timpul mi-am reparat și dieta și așa, ușor, ușor am ajuns să am un stil de viață mult mai bun. Această schimbare pe care am făcut-o, o recomand oricărei persoane deoarece îți îmbunătățești atât sănătatea fizică cât și sănătatea mentală.

Aspirațiile mele

Aspirațiile mele în viață sunt în primul rând să iau bacul cu o notă mare , apoi să îmi continui studiile în cadrul Universității Babeș-Bolyai unde vreau să fac și masteratul. Domeniul în care aș dori să lucrez în viitor este cybersecurity deoarece este o ramură a informaticii pe care o consider frumoasă și interesantă.

Colegiul Național "Roman-Vodă"

Str. Mihai Eminescu Nr. 4, Loc. Roman, jud.Neamt

Alex4IT

Pagina de pornire

Despre mine

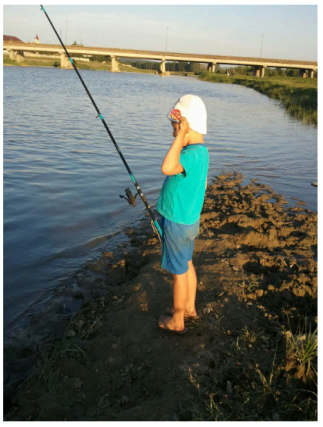
Galerie de imagini

Imagini Interactive



Peisaje generate cu Ai

Aceste imagini au fost generate cu ajutorul Chat Gpt-ului.



Poze cu mine din copilărie

Aceste poze sunt speciale pentru mine, deoarece reprezintă niște amintiri plăcute din viața mea.



Poze Random

Aceste poze sunt niște poze realizate de mine, alese în mod aleatoriu

Alex4IT

- Pagina de pornire
- Despre mine
- Galerie de imagini

Explicația Logo-ului



Această imagine, care este prezentă și pe prima pagină, este logo-ul acestui site, creat cu ajutorul aplicației Inkscape, și este alcătuit din mai multe componente cu diferite semnificații.

Cuvântrul "Alexandru", care este și prenumele meu, este un cuvânt de origine greacă format din trei părți "alexein", "aner", "anderos", care au următoarele semnificații: a oferi o mână de ajutor, bărbat luptător și războinic. Purtătorii acestui nume sunt doritori de glorie și vor mereu să câștige, acestea fiind niște calități care mă descriu și pe mine.

Cifra 4 reprezintă cifra mea norocoasă.

Cuvântul "Informatics" se referă la pasiune mea pentru informatică, care este descrisă pe pagina Despre mine.

Steaua din mijloc semnifică speranța, și faptul că nu trebuie să te dai bătut niciodată. Aceasta are culoarea mov deoarece reprezintă amestecul a celor 2 culori preferate ale mele, roșu și albastru.

Am decis să folosec forma prescurtată a acestor cuvinte, și datorită acestui lucru a rezultat numele "Alex4IT", care are scopul de a pune în lumină și proiectul datorită căruia am creat acest site.

Colegiul Național "Roman-Vodă"

Str. Mihai Eminescu Nr. 4, Loc. Roman, jud.Neamt

```cpp
#include <iostream>
using namespace std;

int main(){
    char *a, *temp;
    int arraysize=0,n=0,tmpN=0,input;//We initialize our vaiables;
    cout<<"Hello there, how many asci codes do you want to add to the array??"<<'\n';
    cin>>arraysize;//We input the array size from the user
    a=new char[arraysize];//We initialize our dynamic array
    do{
        cin>>input;
        if(input<65||input>90)
        cout<<"Wrong input"<<endl;
        else{
            a[n]=(char)input;//We typecast our int into a char;
            n++;//We increment our index
            if(n>=arraysize){
                cout<<"Array is out of space!! Do you want to add more elements(0 for no/
                cin>>tmpN;//We input how many more elements we want to add to our array
                arraysize+=tmpN;//We increase the size
                temp = new char[arraysize];
                for(int i=0;i<n;++i)
                    temp[i]=a[i];//We copy our old array into the new array
                delete a;//We release the space from the old array
                a=temp;//We make a so it point to temp

            }
        }
    }while(n<arraysize);

    for(int i=0;i<arraysize;++i)
        cout<<a[i]<<' '; // We output the elements from the array
      delete a;//We free our memory
      return 0;
}
```

```cpp
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

string searchPhoneUsingName(string nameForsearch,string *name, int arraySize,
string *tel);
string searchNameUsingPhone(string telForsearch,string *tel, int arraySize,
string *name);
int main(){
    int arraySize=0;
    cout<<"How many persons you want to add?"<<'\n';
    cin>>arraySize;
    string *tel = new string[arraySize];
    string *name = new string[arraySize];
    for(int i=0;i<arraySize;i++){
        cout<<"Enter a name and a phone number: ";
        cin>>name[i]>>tel[i];
    }
    for(int i =0;i<arraySize;i++){
        cout<<"Name: "<<name[i]<<'\n'<<"Tel: "<<tel[i]<<'\n';}
    int option=0;
    do{
        cout<<"Option 0 : exit"<<'\n';
        cout<<"Option 1 : Search for a phone number using a name"<<'\n';
        cout<<"Option 2 : Search for a name using a phone number"<<'\n';
        cin>>option;
        switch (option)
        {
        case 0:
            break;
        case 1:
        {
            string nameForSearch;
            cin>>nameForSearch;
            string telResulted = searchPhoneUsingName(nameForSearch,name,arraySize,tel)
            cout<<telResulted<<'\n';
            break;
        }
        case 2:
        {
```

```
            string telForSearch;
            cin>>telForSearch;
            string nameResulted=searchNameUsingPhone(telForSearch,tel,arraySize,name);
            cout<<nameResulted<<'\n';
            break;
        }
        default:
            break;
        }
    }while(option!=0);
    delete tel;
    delete name;
    return 0;

}

int GetMatchScore(string nameForSearch,string nameToMatch){
    int searchlength = nameForSearch.length();
    int matchlength = nameToMatch.length();

    int **matrix = new int*[searchlength+1];
    for(int i = 0;i<searchlength+1;i++)
        matrix[i] = new int[matchlength+1];

    for(int i = 0;i<searchlength+1;i++)
        matrix[i][0] = 0;
    for(int i = 0;i<matchlength+1;i++)
        matrix[0][i]=0;
    for(int i=1;i<searchlength+1;i++)
        for(int j=1;j<searchlength+1;j++)
            if(nameForSearch[i-1]==nameToMatch[j-1])
                matrix[i][j]=matrix[i-1][j-1]+1;
            else if(matrix[i-1][j]>matrix[i][j-1])
                matrix[i][j]=matrix[i-1][j];
            else matrix[i][j]=matrix[i][j-1];
    return matrix[searchlength][matchlength];
}
```

```
string searchPhoneUsingName(string nameForSearch,string *name, int arraySize,
string *tel){
    int scoreBestMatch = -1;
    int indexBestMatch = -1;
    for(int i=0;i<arraySize;i++){
        int score = GetMatchScore(nameForSearch,name[i]);
        if(score>scoreBestMatch){
            scoreBestMatch = score;
            indexBestMatch=i;
        }
    }
    return tel[indexBestMatch];
}

string searchNameUsingPhone(string telForSearch,string *tel, int arraySize,
  string *name){
    int scoreBestMatch = -1;
    int indexBestMatch = -1;
    for(int i=0;i<arraySize;i++){
        int score = GetMatchScore(telForSearch,tel[i]);
        if(score>scoreBestMatch){
            scoreBestMatch = score;
            indexBestMatch=i;
        }
    }

    return name[indexBestMatch];
}
```

```cpp
#include <iostream>
#include <string>
using namespace std;

class car{
    private:
    string brand;
    string model;
    string plateNumber;
    string ownerName;
    public:
    void setBrand(string s){
        this->brand = s;
    }
    void setModel(string s){
        this->model = s;
    }
    void setPlateNumber(string s){
        this->plateNumber = s;
    }
    void setOwnerName(string s){
        this->ownerName = s;
    }
    string getBrand(){
        return this->brand;
    }
    string getModel(){
       return  this->model;
    }
    string getPlateNumber(){
        return this->plateNumber;
    }
    string getOwnerName(){
        return this->ownerName;
    }
};

int main(){
    int SIZE;
    string s;
    cout<<"How many cars do you want to register?"<<'\n';
    cin>>SIZE;
```

```
    cin.get();
    car *cars = new car[SIZE];
    for(int i=0;i<SIZE;++i){
        cout<<"Enter car "<<i+1<<" brand: \n";
        getline(cin,s);
        cars[i].setBrand(s);
        cout<<"Enter car "<<i+1<<"model\n";
        getline(cin,s);
        cars[i].setModel(s);
        cout<<"Enter car "<<i+1<<"owner\n";
        getline(cin,s);
        cars[i].setOwnerName(s);
        cout<<"Enter car "<<i+1<<"plate number\n";
        getline(cin,s);
        cars[i].setPlateNumber(s);
    }
    for(int i=0;i<SIZE;++i){
        cout<<"Car"<<i+1<<": \n";
        cout<<"Brand: "<<cars[i].getBrand()<<'\n';
        cout<<"Model: "<<cars[i].getModel()<<'\n';
        cout<<"Owner name: "<<cars[i].getOwnerName()<<'\n';
        cout<<"Plate number: "<<cars[i].getPlateNumber()<<'\n';
    }

    delete[] cars;
}
```

```cpp
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
class product {
    private:
        string brand;
        string type;
        double price;
        int stockRemaining;
        string barCode;

    public:


        string getBrand() const {
            return this->brand;
        }

        string getType() const {
            return this->type;
        }

        double getPrice() const {
            return this->price;
        }

        int getStockRemaining() const {
            return this->stockRemaining;
        }

        string getBarCode() const {
            return this->barCode;
        }

        void setBrand(string b) {
            this->brand = b;
        }

        void setType(string t) {
            this->type = t;
```

```
        }

        void setPrice(double p) {
            this->price = p;
        }

        void setStockRemaining(int s) {
            this->stockRemaining = s;
        }

        void setBarCode(string bc) {
            this->barCode = bc;
        }
};



void searchByName(string nameForSearch, product* products, int arraySize) {
    bool found = false;
    for (int i = 0; i < arraySize; i++) {
        if (products[i].getBrand() == nameForSearch) { // Exact match
            cout << "Product found: " << endl;
            cout << "Brand: " << products[i].getBrand() << endl;
            cout << "Type: " << products[i].getType() << endl;
            cout << "Price: $" << products[i].getPrice() << endl;
            cout << "Stock Remaining: " << products[i].getStockRemaining() << endl;
            cout << "Bar Code: " << products[i].getBarCode() << endl;
            found = true;
        }
    }
    if (!found) {
        cout << "No product found with the name: " << nameForSearch << endl;
    }
}

void searchByType(string typeForSearch, product* products, int arraySize) {
    bool found = false;
    for (int i = 0; i < arraySize; i++) {
        if (products[i].getType() == typeForSearch) { // Exact match
            cout << "Product found: " << endl;
            cout << "Brand: " << products[i].getBrand() << endl;
            cout << "Type: " << products[i].getType() << endl;
            cout << "Price: $" << products[i].getPrice() << endl;
```

```
            cout << "Stock Remaining: " << products[i].getStockRemaining() << endl;
            cout << "Bar Code: " << products[i].getBarCode() << endl;
            found = true;
        }
    }
    if (!found) {
        cout << "No product found with the type: " << typeForSearch << endl;
    }
}

int main(){
    int SIZE = 0;
    cout << "Enter the number of products: ";
    cin >> SIZE;
    cin.get();
    product* products = new product[SIZE];
    for(int i=0;i<SIZE;++i){
        cout<<"Enter the brand of product"<<i+1<<": ";
        string s;
        getline(cin,s);
        products[i].setBrand(s);
        cout<<"Enter the type of product"<<i+1<<": ";
        string t;
        getline(cin,t);
        cout<<"Enter the price of product"<<i+1<<": ";
        double p;
        cin>>p;
        cin.get();
        products[i].setPrice(p);
        cout<<"Enter the stock remaining of product"<<i+1<<": ";
        int sR;
        cin>>sR;
        cin.get();
        products[i].setStockRemaining(sR);
        cout<<"Enter the bar code of product"<<i+1<<": ";
        string bC;
        getline(cin,bC);
        products[i].setBarCode(bC);

    }
    int option = -1;
    product *productsThatMatch = new product[SIZE];
    int numberofproductsthatmatch;
```

```
    do
    {
        cout<<"Enter 0 to exit"<<endl;
        cout<<"Enter 1 to search by name"<<endl;
        cout<<"Enter 2 to search by type"<<endl;
        cin>>option;
        cin.get();
        switch (option)
        {
        case 1:
        {
            cout << "Enter the name of the product to search: ";
            string nameForSearch;
            getline(cin, nameForSearch);
            searchByName(nameForSearch, products, SIZE);
            break;
        }
        case 2:{
            cout << "Enter the type of the product to search: ";
            string typeForSearch;
            getline(cin, typeForSearch);
            searchByType(typeForSearch, products, SIZE);
            break;

        }
        default:
            break;
        }
    } while (option!=0);

      delete[] products;
      delete[] productsThatMatch;

}
```

```cpp
#include <iostream>
#include <string>
using namespace std;

class car{
    private:
        string brand;
        string model;
        string plateNumber;
        string ownerName;
    public:
        car() : brand(""), model(""), plateNumber(""), ownerName(""){};
        void setBrand(string s){this->brand = s;}
        void setModel(string s){this->model = s;}
        void setPlateNumber(string s){this->plateNumber = s;}
        void setOwnerName(string s){ this->ownerName = s;}
        string getBrand(){return this->brand; }
        string getModel(){ return  this->model;}
        string getPlateNumber(){return this->plateNumber; }
        string getOwnerName(){return this->ownerName;}
};
void enterData(int &SIZE, car*& cars, int n){
    car *tempCars = new car[SIZE+n];
    for(int i=0;i<SIZE;++i)
        tempCars[i] = cars[i];
    string s;
    for(int i=SIZE;i<SIZE+n;++i){
        cout<<"Enter car "<<i+1<<" brand: \n";
        getline(cin,s);
        tempCars[i].setBrand(s);
        cout<<"Enter car "<<i+1<<"model\n";
        getline(cin,s);
        tempCars[i].setModel(s);
        cout<<"Enter car "<<i+1<<"owner\n";
        getline(cin,s);
        tempCars[i].setOwnerName(s);
        cout<<"Enter car "<<i+1<<"plate number\n";
        getline(cin,s);
        tempCars[i].setPlateNumber(s);
    }
    delete[] cars;
    SIZE+=n;
```

```
    cars = tempCars;
}

void displayData(int SIZE,car* cars){
    for(int i=0;i<SIZE;++i){
        cout<<"Car"<<i+1<<": \n";
        cout<<"Brand: "<<cars[i].getBrand()<<'\n';
        cout<<"Model: "<<cars[i].getModel()<<'\n';
        cout<<"Owner name: "<<cars[i].getOwnerName()<<'\n';
        cout<<"Plate number: "<<cars[i].getPlateNumber()<<'\n';
    }

}

void searchByBrand(int SIZE, car* cars, string brand) {
    bool found = false;
    for (int i = 0; i < SIZE; ++i) {
        if (cars[i].getBrand() == brand) {
            cout << "Car " << i + 1 << ":\n";
            cout << "Brand: " << cars[i].getBrand() << '\n';
            cout << "Model: " << cars[i].getModel() << '\n';
            cout << "Owner name: " << cars[i].getOwnerName() << '\n';
            cout << "Plate number: " << cars[i].getPlateNumber() << '\n';
            found = true;
        }
    }
    if (!found) {
        cout << "No cars found with brand: " << brand << '\n';
    }
}

void searchByOwner(int SIZE, car* cars, string owner) {
    bool found = false;
    for (int i = 0; i < SIZE; ++i) {
        if (cars[i].getOwnerName() == owner) {
            cout << "Car " << i + 1 << ":\n";
            cout << "Brand: " << cars[i].getBrand() << '\n';
            cout << "Model: " << cars[i].getModel() << '\n';
            cout << "Owner name: " << cars[i].getOwnerName() << '\n';
            cout << "Plate number: " << cars[i].getPlateNumber() << '\n';
            found = true;
        }
```

```
    }
    if (!found) {
        cout << "No cars found with owner: " << owner << '\n';
    }
}

void searchByPlateNumber(int SIZE, car* cars, string plateNumber) {
    bool found = false;
    for (int i = 0; i < SIZE; ++i) {
        if (cars[i].getPlateNumber() == plateNumber) {
            cout << "Car " << i + 1 << ":\n";
            cout << "Brand: " << cars[i].getBrand() << '\n';
            cout << "Model: " << cars[i].getModel() << '\n';
            cout << "Owner name: " << cars[i].getOwnerName() << '\n';
            cout << "Plate number: " << cars[i].getPlateNumber() << '\n';
            found = true;
        }
    }
    if (!found) {
        cout << "No cars found with plate number: " << plateNumber << '\n';
    }
}



int main(){
    int SIZE=0,option=-1;
    car* cars = new car[SIZE];
    do
    {
        cout<<"1.Enter data"<<'\n';
        cout<<"2.Display data"<<'\n';
        cout<<"3.Search by brand"<<'\n';
        cout<<"4.Search by owner"<<'\n';
        cout<<"5.Search by plate number"<<'\n';
        cin>>option;

        cin.get();
        switch (option)
        {
        case 1:
        {
```

```
            int n;
            cout<<"How many cars do you want to enter?'\n";
            cin>>n;
            cin.get();
            enterData(SIZE,cars,n);
            break;
        }
        case 2:
        {
            displayData(SIZE,cars);
            break;
        }
        case 3:
        {
            string brand;
            cout << "Enter the brand to search for:\n";
            getline(cin, brand);
            searchByBrand(SIZE, cars, brand);
            break;
        }
        case 4:
        {
            string owner;
            cout << "Enter the owner name to search for:\n";
            getline(cin, owner);
            searchByOwner(SIZE, cars, owner);
            break;
        }
        case 5:
        {
            string plateNumber;
            cout << "Enter the plate number to search for:\n";
            getline(cin, plateNumber);
            searchByPlateNumber(SIZE, cars, plateNumber);
            break;
        }
        default:
        {
            cout<<"Introdu o valoare corecta!!";
            break;
        }
        }
    } while (option!=0);
    delete[] cars;
    return 0;

}
```

```cpp
#include <iostream>
#include <string>
using namespace std;

class product {
private:
    string brand;
    string type;
    double price;
    int stockRemaining;
    string barCode;

public:
    product() : brand(""), type(""), price(0.0), stockRemaining(0), barCode("") {}

    void setBrand(string b) { this->brand = b; }
    void setType(string t) { this->type = t; }
    void setPrice(double p) { this->price = p; }
    void setStockRemaining(int s) { this->stockRemaining = s; }
    void setBarCode(string bc) { this->barCode = bc; }

    string getBrand() { return this->brand; }
    string getType() { return this->type; }
    double getPrice() { return this->price; }
    int getStockRemaining() { return this->stockRemaining; }
    string getBarCode() { return this->barCode; }
};

void enterData(int& SIZE, product*& products, int n) {
    product* temp = new product[SIZE + n];
    for (int i = 0; i < SIZE; ++i) {
        temp[i] = products[i];
    }

    string s;
    for (int i = SIZE; i < SIZE + n; ++i) {
        cout << "Enter product " << i + 1 << " brand:\n";
        getline(cin, s);
        temp[i].setBrand(s);

        cout << "Enter product " << i + 1 << " type:\n";
        getline(cin, s);
        temp[i].setType(s);
```

```
        cout << "Enter product " << i + 1 << " price:\n";
        double price;
        cin >> price;
        temp[i].setPrice(price);
        cin.get();

        cout << "Enter product " << i + 1 << " stock remaining:\n";
        int stock;
        cin >> stock;
        temp[i].setStockRemaining(stock);
        cin.get();

        cout << "Enter product " << i + 1 << " bar code:\n";
        getline(cin, s);
        temp[i].setBarCode(s);
    }

    delete[] products;
    SIZE += n;
    products = temp;
}

void displayData(int SIZE, product* products) {
    for (int i = 0; i < SIZE; ++i) {
        cout << "Product " << i + 1 << ":\n";
        cout << "Brand: " << products[i].getBrand() << '\n';
        cout << "Type: " << products[i].getType() << '\n';
        cout << "Price: $" << products[i].getPrice() << '\n';
        cout << "Stock Remaining: " << products[i].getStockRemaining() << '\n';
        cout << "Bar Code: " << products[i].getBarCode() << "\n\n";
    }
}

void searchByBrand(int SIZE, product* products, string brand) {
    int found = 0;
    for (int i = 0; i < SIZE; ++i) {
        if (products[i].getBrand() == brand) {
            cout << "Product " << i + 1 << ":\n";
            cout << "Brand: " << products[i].getBrand() << '\n';
            cout << "Type: " << products[i].getType() << '\n';
            cout << "Price: $" << products[i].getPrice() << '\n';
            cout << "Stock Remaining: " << products[i].getStockRemaining() << '\n';
            cout << "Bar Code: " << products[i].getBarCode() << "\n\n";
            found = 1;
```

```
        }
    }
    if (!found) {
        cout << "No product found with brand: " << brand << '\n';
    }
}

void searchByType(int SIZE, product* products, string type) {
    int found = 0;
    for (int i = 0; i < SIZE; ++i) {
        if (products[i].getType() == type) {
            cout << "Product " << i + 1 << ":\n";
            cout << "Brand: " << products[i].getBrand() << '\n';
            cout << "Type: " << products[i].getType() << '\n';
            cout << "Price: $" << products[i].getPrice() << '\n';
            cout << "Stock Remaining: " << products[i].getStockRemaining() << '\n';
            cout << "Bar Code: " << products[i].getBarCode() << "\n\n";
            found = true;
        }
    }
    if (!found) {
        cout << "No product found with type: " << type << '\n';
    }
}

int main() {
    int SIZE = 0;
    int option = -1;
    product* products = new product[SIZE];

    do {
        cout << "\nMENU\n";
        cout << "0. Exit\n";
        cout << "1. Enter product data\n";
        cout << "2. Display all products\n";
        cout << "3. Search by brand\n";
        cout << "4. Search by type\n";
        cout << "Choose an option: ";
        cin >> option;
        cin.get();

        switch (option) {
```

```
        case 1: {
            int n;
            cout << "How many products do you want to enter?\n";
            cin >> n;
            cin.get();
            enterData(SIZE, products, n);
            break;
        }
        case 2: {
            displayData(SIZE, products);
            break;
        }
        case 3: {
            string brand;
            cout << "Enter brand to search: ";
            getline(cin, brand);
            searchByBrand(SIZE, products, brand);
            break;
        }
        case 4: {
            string type;
            cout << "Enter type to search: ";
            getline(cin, type);
            searchByType(SIZE, products, type);
            break;
        }
        case 0: {
            cout << "Exiting...\n";
            break;
        }
        default:
            cout << "Invalid option. Try again.\n";
        }
    } while (option != 0);

    delete[] products;
    return 0;
}
```

```
product* enterData(int& SIZE, product* products, int n) {
    product* temp = new product[SIZE + n];
    for (int i = 0; i < SIZE; ++i) {
        temp[i] = products[i];
    }
    string s;
    for (int i = SIZE; i < SIZE + n; ++i) {
        cout << "Enter product " << i + 1 << " brand:\n";
        getline(cin, s);
        temp[i].setBrand(s);

        cout << "Enter product " << i + 1 << " type:\n";
        getline(cin, s);
        temp[i].setType(s);

        cout << "Enter product " << i + 1 << " price:\n";
        double price;
        cin >> price;
        temp[i].setPrice(price);
        cin.get();

        cout << "Enter product " << i + 1 << " stock remaining:\n";
        int stock;
        cin >> stock;
        temp[i].setStockRemaining(stock);
        cin.get();

        cout << "Enter product " << i + 1 << " bar code:\n";
        getline(cin, s);
        temp[i].setBarCode(s);
    }
    SIZE += n;
    return temp;
}
```

```
product* enterData(int& SIZE, product* products, int n) {
    product* temp = new product[SIZE + n];
    for (int i = 0; i < SIZE; ++i) {
        temp[i] = products[i];
    }
    string s;
    for (int i = SIZE; i < SIZE + n; ++i) {
        cout << "Enter product " << i + 1 << " brand:\n";
        getline(cin, s);
        temp[i].setBrand(s);

        cout << "Enter product " << i + 1 << " type:\n";
        getline(cin, s);
        temp[i].setType(s);

        cout << "Enter product " << i + 1 << " price:\n";
        double price;
        cin >> price;
        temp[i].setPrice(price);
        cin.get();

        cout << "Enter product " << i + 1 << " stock remaining:\n";
        int stock;
        cin >> stock;
        temp[i].setStockRemaining(stock);
        cin.get();

        cout << "Enter product " << i + 1 << " bar code:\n";
        getline(cin, s);
        temp[i].setBarCode(s);
    }
    SIZE += n;
    return temp;
}
```

```cpp
#include <iostream>
using namespace std;

// Definirea clasei `car`
class car {
    private:
        string brand; // Marca mașinii
        string model; // Modelul mașinii
    public:
        // Constructor implicit care inițializează valorile cu "EMPTY"
        car() {
            brand = "EMPTY";
            model = "EMPTY";
        }

        // Setează marca mașinii
        void setBrand(string B) {
            brand = B;
        }

        // Setează modelul mașinii
        void setModel(string M) {
            model = M;
        }

        // Returnează marca mașinii
        string getBrand() {
            return brand;
        }

        // Returnează modelul mașinii
        string getModel() {
            return model;
        }
};

// Declarații ale funcțiilor utilizate în program
car *readData(car *arr, int &arrSize, int &nextPos);
void displayData(car *arr, int nextPos);
void searchByBrand(car *arr, int nextPos, const string &brand);

int main() {
    int ARRAYSIZE = 4; // Dimensiunea inițială a array-ului
    int nextAvailPos = 0; // Următoarea poziție disponibilă în array
    car *carsArray; // Pointer către array-ul de obiecte `car`
```

```
    carsArray = new car[ARRAYSIZE]; // Alocare dinamică a array-ului
    int choice = -1; // Alegerea utilizatorului din meniu

    do {
        // Afișarea meniului
        cout << "Please Choose from the menu below: " << endl;
        cout << "1. Enter Data" << endl;
        cout << "2. Display Data" << endl;
        cout << "3. Search By Brand" << endl;
        cout << "0. Exit" << endl;
        cin >> choice;

        switch (choice) {
            case 1:
                // Citirea datelor de la utilizator
                carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
                break;
            case 2:
                // Afișarea datelor din array
                displayData(carsArray, nextAvailPos);
                break;
            case 3: {
                // Căutarea mașinilor după marcă
                string brandToSearch;
                cout << "Enter the brand to search: ";
                cin >> brandToSearch;
                searchByBrand(carsArray, nextAvailPos, brandToSearch);
                break;
            }
            case 0:
                // Ieșirea din program
                break;
            default:
                // Mesaj pentru alegere invalidă
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 0);

    delete[] carsArray; // Eliberarea memoriei alocate dinamic
    return 0;
}

// Funcția `readData` pentru citirea datelor de la utilizator
car *readData(car *arr, int &arrSize, int &nextPos) {
    int numcars = 0; // Numărul de mașini pe care utilizatorul dorește să le introducă
    int newSize = 0; // Noua dimensiune a array-ului, dacă este necesară redimensionarea
    string tstring; // String temporar pentru citirea datelor de la tastatură
```

```
    car *tmp; // Pointer temporar pentru redimensionarea array-ului

    cout << "How many cars are you entering: ";
    cin >> numcars; // Citirea numărului de mașini

    for (int i = 0; i < numcars; i++) { // Repetarea procesului pentru fiecare mașină
        if (nextPos >= arrSize) { // Verificarea dacă array-ul este plin
            newSize = arrSize + numcars; // Calcularea noii dimensiuni
            tmp = new car[newSize]; // Alocarea unui array mai mare
            for (int j = 0; j < arrSize; j++) { // Copierea datelor din array-ul vechi
                tmp[j] = arr[j];
            }
            arrSize = newSize; // Actualizarea dimensiunii array-ului

            delete[] arr; // Eliberarea memoriei vechi
            arr = tmp; // Actualizarea pointer-ului array-ului
        }

        // Citirea mărcii și modelului pentru mașina curentă
        cout << "Enter BRAND for car[" << nextPos << "]";
        cin >> tstring;
        arr[nextPos].setBrand(tstring);
        cout << "Enter MODEL for car[" << nextPos << "]";
        cin >> tstring;
        arr[nextPos].setModel(tstring);
        nextPos++; // Incrementarea poziției disponibile
    }
    return arr; // Returnarea pointer-ului către array
}

// Funcția `displayData` pentru afișarea datelor din array
void displayData(car *arr, int nextPos) {
    if (nextPos == 0) { // Verificarea dacă array-ul este gol
        cout << "No cars in the database." << endl;
        return;
    }
    for (int i = 0; i < nextPos; i++) { // Iterarea prin array și afișarea fiecărei mașin
        cout << "Car[" << i << "] - Brand: " << arr[i].getBrand()
             << ", Model: " << arr[i].getModel() << endl;
    }
}

// Funcția `searchByBrand` pentru căutarea mașinilor după marcă
void searchByBrand(car *arr, int nextPos, const string &brand) {
    bool found = false; // Flag pentru a verifica dacă s-a găsit o mașină
    for (int i = 0; i < nextPos; i++) { // Iterarea prin array
        if (arr[i].getBrand() == brand) { // Compararea mărcii
            cout << "Found Car[" << i << "] - Brand: " << arr[i].getBrand()
                 << ", Model: " << arr[i].getModel() << endl;
            found = true; // Setarea flag-ului dacă s-a găsit o mașină
        }
    }
    if (!found) { // Dacă nu s-a găsit nicio mașină
        cout << "No cars found with the brand: " << brand << endl;
    }
}
```

```cpp
#include <iostream>
using namespace std;


//CREATE THE CLASS
class car{
    private:
        string brand;
        string model;
    public:

        car(){
            brand="EMPTY";
            model="EMPTY";

        }
        void setBrand(string B){
            brand=B;
        }
        void setModel(string M){
            model=M;
        }
        string getBrand(){
            return brand;
        }
        string getModel(){
            return model;
        }
};

car *readData(car *, int & ,int &);
void displayData(car*, int);
int searchByBrand(car*,int);
void editData(car* arr, int index);

int main() {
    int ARRAYSIZE = 4;
    int nextAvailPos = 0;
    car* carsArray = new car[ARRAYSIZE]; // Allocate memory for the array
    int choice = -1;
    int searchIndex = -1;

    do {
        cout << "Please Choose from the menu below: " << endl;
        cout << "1. Enter Data" << endl;
        cout << "2. Display Data" << endl;
        cout << "3. Search By Brand" << endl;
        cout << "4. Edit Data" << endl;
        cout << "0. Exit" << endl;
        cin >> choice;

        if (cin.fail()) { // Validate input
            cin.clear(); // Clear the error flag
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Discard invalid input
            cout << "Invalid input. Please enter a number between 0 and 4." << endl;
            continue;
```

```
            continue;
        }

        switch (choice) {
            case 1:
                carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
                break;
            case 2:
                displayData(carsArray, nextAvailPos);
                break;
            case 3:
                searchIndex = searchByBrand(carsArray, nextAvailPos);
                break;
            case 4:
                searchIndex = searchByBrand(carsArray, nextAvailPos);
                if (searchIndex != -1) {
                    editData(carsArray, searchIndex);
                }
                break;
            case 0:
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 0);

    delete[] carsArray; // Free dynamically allocated memory
    return 0;
}

int searchByBrand(car* arr, int arrSize) {
    string tstr; // Temporary string to read data from keyboard
    int index = -1; // Declare and initialize index to -1
    cout << "Enter the brand you are looking for: ";
    cin >> tstr;
    for (int i = 0; i < arrSize; i++) {
        if (arr[i].getBrand() == tstr) {
            cout << "Index found at position: " << i << endl;
            index = i;
            break; // Exit the loop once the brand is found
        }
    }

    if (index == -1) {
        cout << "Brand not found." << endl;
    }

    return index;
}

//void displayData()
//the function receives as arguments
//a car object array and the size of the array.
//It displays all data that are saved on the car array
void displayData(car *arr, int arrSize){
```

```
    for (int i=0;i<arrSize;i++){
        cout<<"----- "<<"Car "<<i+1<<" -----"<<endl;
        cout<<arr[i].getBrand()<<endl;
        cout<<arr[i].getModel()<<endl;
    }

}
car *readData(car *arr, int &arrSize,int &nextPos){
    int numcars=0; //how many cars the user is going to input
    int newSize=0; //the new size of the array which is calculated by adding arra
    string tstring; //temporary string to read keyboard input
    car *tmp; //temporary car object to copy everything from the car array and gr

    cout<<"How many cars are you entering: ";
    cin>>numcars; //read from the keyboard how many cars the user needs to input
    for (int i=0;i<numcars;i++){//repeat the reading process 'numCars' times

        if( nextPos>=arrSize){ //first check if the array is full
           newSize=arrSize+numcars;//if you got here the array is full calculate
           tmp=new car[newSize]; //allocate temporary space for the new bigger ar
           for (int j=0;j<arrSize;j++){//copy everything from the small array to
                tmp[j]=arr[j];
            }
            arrSize=newSize; //save the newsize into arrSize (update the arrSize)

            delete []arr; //release the memory from the old array back to the OS
            arr=tmp; //set the pointer of the array to point to the new allocated

        }
        cout<<"Enter BRAND for car["<<nextPos<<"]";
        cin>>tstring;
        arr[nextPos].setBrand(tstring);
        cout<<"Enter MODEL for car["<<nextPos<<"]";
        cin>>tstring;
        arr[nextPos].setModel(tstring);
        nextPos++;

    }
    return arr;
}
void editData(car* arr, int index) {
    if (index < 0) {
        cout << "Invalid index. No data to edit." << endl;
        return;
    }

    cout << "Current data at index " << index << ":" << endl;
    cout << "Brand: " << arr[index].getBrand() << endl;
    cout << "Model: " << arr[index].getModel() << endl;

    string newBrand, newModel;
    cout << "Enter new BRAND: ";
    cin >> newBrand;
    cout << "Enter new MODEL: ";
    cin >> newModel;
```

```
    arr[index].setBrand(newBrand);
    arr[index].setModel(newModel);

    cout << "Data updated successfully!" << endl;
}
```

```
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);  // delay in between reads for stability
}
```


CUM SĂ-ȚI GĂSEȘTI
UN LOC DE MUNCĂ
Ciuștea Ioana-Alexandra
Enea Alexandru-Petronel
Grigoraș Darius-George
Laic Andrei-Ștefan
Murariu Cosmin-Andrei
Pascal Andrei
Jobs
EMPLOYMENT / HELP WANTED
EMPLOYMENT


DIGITAL
SKILLS
Importanța competențelor digitale în piața muncii
Enea Alexandru-Petronel



Enea Alexandru