# PROIECT-GESTIONAREA UNUI SPITAL

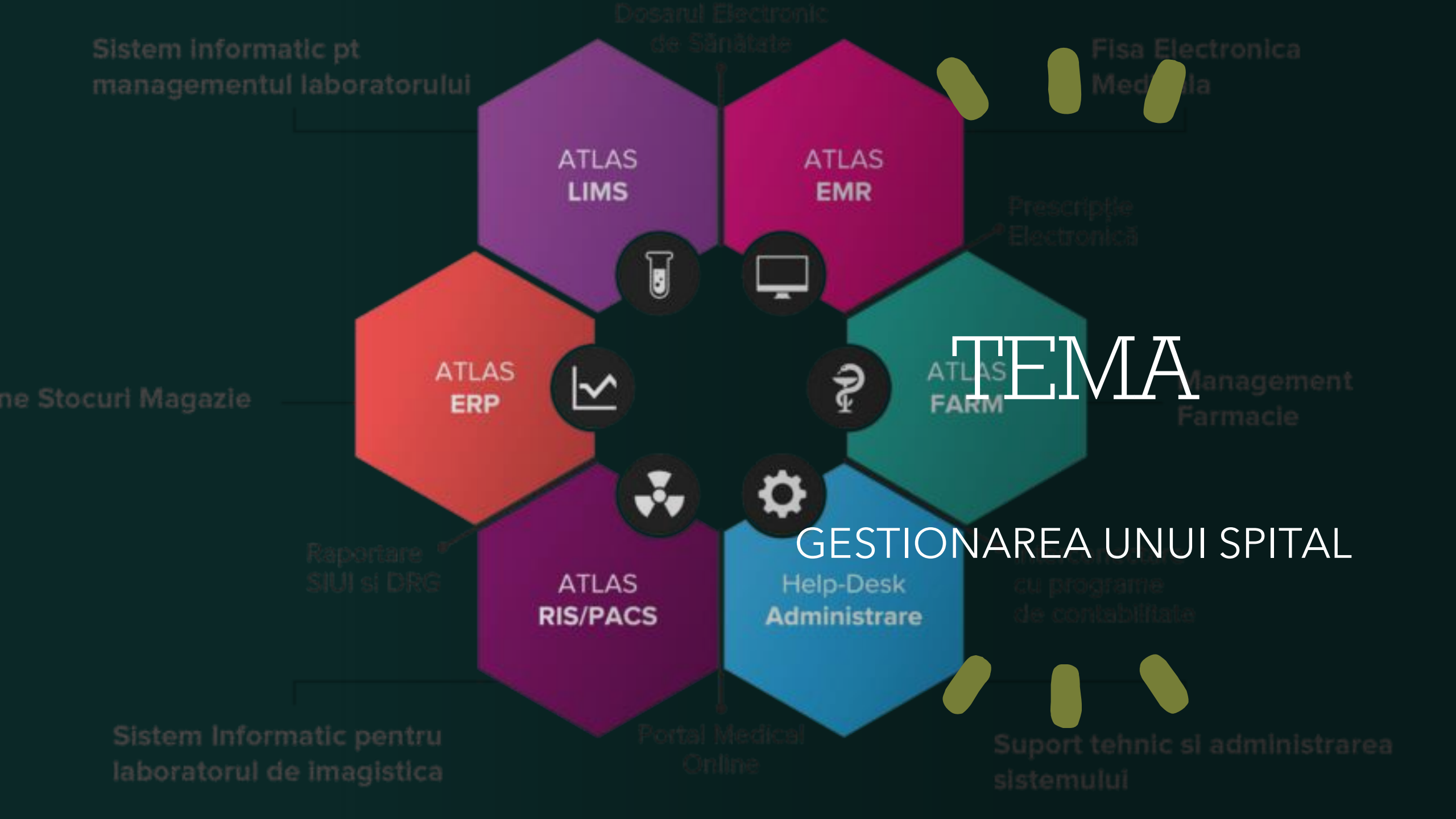

NUME: BALTOIU BIANCA NICOLETA

AN STUDIU: 1

GRUPA: 4LF531

- UNIVERSITATEA TRANSILVANIA BRASOV
- FACULTATEA: INGINERIE ELECTRICA SI STIINTA CALCULATOARELOR
- SPECIALIZARE: INGINERIE ELECTRICA SI CALCULATOARE(IN LIMBA ENGLEZA)


# SPECIFICATIA DE DEFINIRE A PROBLEMEI




# TEMA

## GESTIONAREA UNUI SPITAL

# CERINTE

- inregistrarea numarului de locuri in saloane pentru diferite domenii medicale
- inregistrarea pacientilor
- crearea listei de asteptare, se tine seama de urgenta
- operatii cu datele: afisare, cautare, editare
- realizarea unor statistici

# SPECIFICAREA FUNCTIONALA

- **1. Protecția cu Parolă**
- **Date de Intrare**: Introducerea parolei de către utilizator.
- **Date de Ieșire**: Mesaj de acces acordat dacă parola este corectă; mesaj de eroare dacă parola este incorectă.

# 2. Înregistrare Salon

- **Date de Intrare**: Domeniul salonului (string), capacitatea (integer).
- **Date de Ieșire**: Mesaj de confirmare a înregistrării salonului.

# 3. Înregistrare Pacient

- **Date de Intrare**: Numele pacientului (string), vârsta (integer), boala (string), urgența (integer).
- **Date de Ieșire**: Mesaj de confirmare a înregistrării pacientului.

# 4. Afișare Saloane

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Lista saloanelor cu detaliile lor.

# 5. Afișare Pacienți

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Lista pacienților cu detaliile lor.

# 6. Afișare Lista de Așteptare

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Lista pacienților din lista de așteptare, sortată după urgență.

# 7. Căutare Pacient

- **Date de Intrare**: Numele pacientului (string).
- **Date de Ieșire**: Detalii despre pacient dacă este găsit, altfel un mesaj de eroare.

# 8. Vizualizare Statistici

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Diverse statistici, cum ar fi numărul de saloane și pacienți.

# 9. Salvare Date

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Datele scrise într-un fișier numit **`pacienti.txt`**.

# 10. Ieșire din Program

- **Date de Intrare**: Niciuna.
- **Date de Ieșire**: Mesaj de ieșire.

# INTERFATA UTILIZATOR

**1. Autentificare**

**Dialog**: Utilizatorul introduce parola.

**2. Meniul Principal**

**Dialog**: Utilizatorul alege opțiunea dorită din meniu (1-9).

**3. Înregistrare Salon**

**Dialog**: Utilizatorul introduce domeniul și numărul de locuri pentru un salon.

**4. Înregistrare Pacient**

**Dialog**: Utilizatorul introduce detalii despre pacient: nume, vârstă, boală, urgență.

**5. Afișare Saloane**

**Dialog**: Se afișează lista de saloane cu detaliile lor.

**6. Afișare Pacienți**

**Dialog**: Se afișează lista de pacienți cu detaliile lor.

**7. Afișare Lista de Așteptare**

**Dialog**: Se afișează lista de așteptare a pacienților, sortată după urgență.

**8. Căutare Pacient**

**Dialog**: Utilizatorul introduce numele pacientului pentru căutare.

**9. Statistici**

**Dialog**: Se afișează statistici despre numărul de saloane și pacienți.

**10. Salvare Date**

**Dialog**: Datele pacienților sunt salvate cu succes într-un fișier text.

**11. Ieșire din Program**

**Dialog**: Programul se închide.

Aceste cerințe asigură o interfață simplă și intuitivă pentru utilizator, facilitând utilizarea eficientă a funcționalităților programului de gestionare a spitalului.

# DEZVOLTAREA PROIECTULUI

# OBIECTIVE GENERALE

OBIECTIVUL PROGRAMULUI ESTE DE A AJUTA LA ORGANIZAREA MAI EFICIENTA A UNUI SPITAL

# DESCRIEREA DATELOR PRELUCRATE DE PROGRAM

**DATE DE INTRARE**

**1. Autentificare**

**Date de Intrare**: Parola introdusă de utilizator.

**Formate Utilizate**: Șir de caractere (`char[]`).

**2. Înregistrare Salon**

**Date de Intrare**: Domeniul și numărul de locuri pentru un salon.

**Formate Utilizate**: Șir de caractere pentru domeniu (`string`) și număr întreg (`int`) pentru locuri.

**3. Înregistrare Pacient**

**Date de Intrare**: Numele, vârsta, boala și nivelul de urgență al pacientului.

**Formate Utilizate**: Șir de caractere pentru nume și boală (`string`), numere întregi pentru vârstă și urgență (`int`).

**4. Căutare Pacient**

**Date de Intrare**: Numele pacientului căutat.

**Formate Utilizate**: Șir de caractere (`string`).

**5. Salvare Date**

**Date de Intrare**: Nu există date de intrare pentru această acțiune.

**Formate Utilizate**: Nu este aplicabil.

**DATE DE IESIRE**

**1. Afișare Saloane**

**Date de Ieșire**: Lista de saloane cu detaliile lor.

**Formate Utilizate**: Șiruri de caractere (`string`) pentru domeniu și număr întreg (`int`) pentru locuri.

**2. Afișare Pacienți**

**Date de Ieșire**: Lista de pacienți cu detaliile lor.

**Formate Utilizate**: Șiruri de caractere (`string`) pentru nume și boală, numere întregi (`int`) pentru vârstă și urgență.

**3. Afișare Lista de Așteptare**

**Date de Ieșire**: Lista de așteptare a pacienților, sortată după urgență.

**Formate Utilizate**: Șiruri de caractere (`string`) pentru nume și boală, numere întregi (`int`) pentru vârstă și urgență.

**4. Statistici**

**Date de Ieșire**: Numărul de saloane și numărul de pacienți.

**Formate Utilizate**: Numere întregi (`int`).

# DESCRIEREA MODULELOR DE PROGRAM

**1. Autentificare**

**Funcție**: authentification

**Parametri**: Parola introdusă de utilizator.

**Valori Returnate**: Rezultatul autentificării (adevărat/fals).

**Algoritm Utilizat**: Compararea parolei introduse de utilizator cu parola predefinită.

**2. Înregistrare Salon**

**Funcție**: registerWard

**Parametri**: Domeniul și numărul de locuri pentru un salon.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Înregistrarea informațiilor despre salon într-o structură de date sau clasă și adăugarea acesteia într-o listă de saloane.

**3. Înregistrare Pacient**

**Funcție**: registerPatient

**Parametri**: Detalii despre pacient: nume, vârstă, boală, nivel de urgență.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Înregistrarea detaliilor pacientului într-o structură de date sau clasă și adăugarea acestuia într-o listă de pacienți.

**4. Căutare Pacient**

**Funcție**: searchPatient

**Parametri**: Numele pacientului căutat.

**Valori Returnate**: Detaliile pacientului căutat sau un mesaj de eroare dacă pacientul nu este găsit.

**Algoritm Utilizat**: Parcurgerea listei de pacienți și compararea numelor pentru găsirea pacientului căutat.

**5. Afișare Saloane**

**Funcție**: displayWards

**Parametri**: Nu necesită parametri.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Parcurgerea listei de saloane și afișarea detaliilor acestora.

**6. Afișare Pacienți**

**Funcție**: displayPatients

**Parametri**: Nu necesită parametri.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Parcurgerea listei de pacienți și afișarea detaliilor acestora.

**7. Afișare Lista de Așteptare**

**Funcție**: displayWaitingList

**Parametri**: Nu necesită parametri.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Parcurgerea listei de așteptare a pacienților și afișarea acestora, sortată după nivelul de urgență.

**8. Statistici**

**Funcție**: `statistics`

**Parametri**: Nu necesită parametri.

**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Calcularea și afișarea statisticilor generale despre spital, cum ar fi numărul de saloane și numărul de pacienți.

**9. Salvare Date**

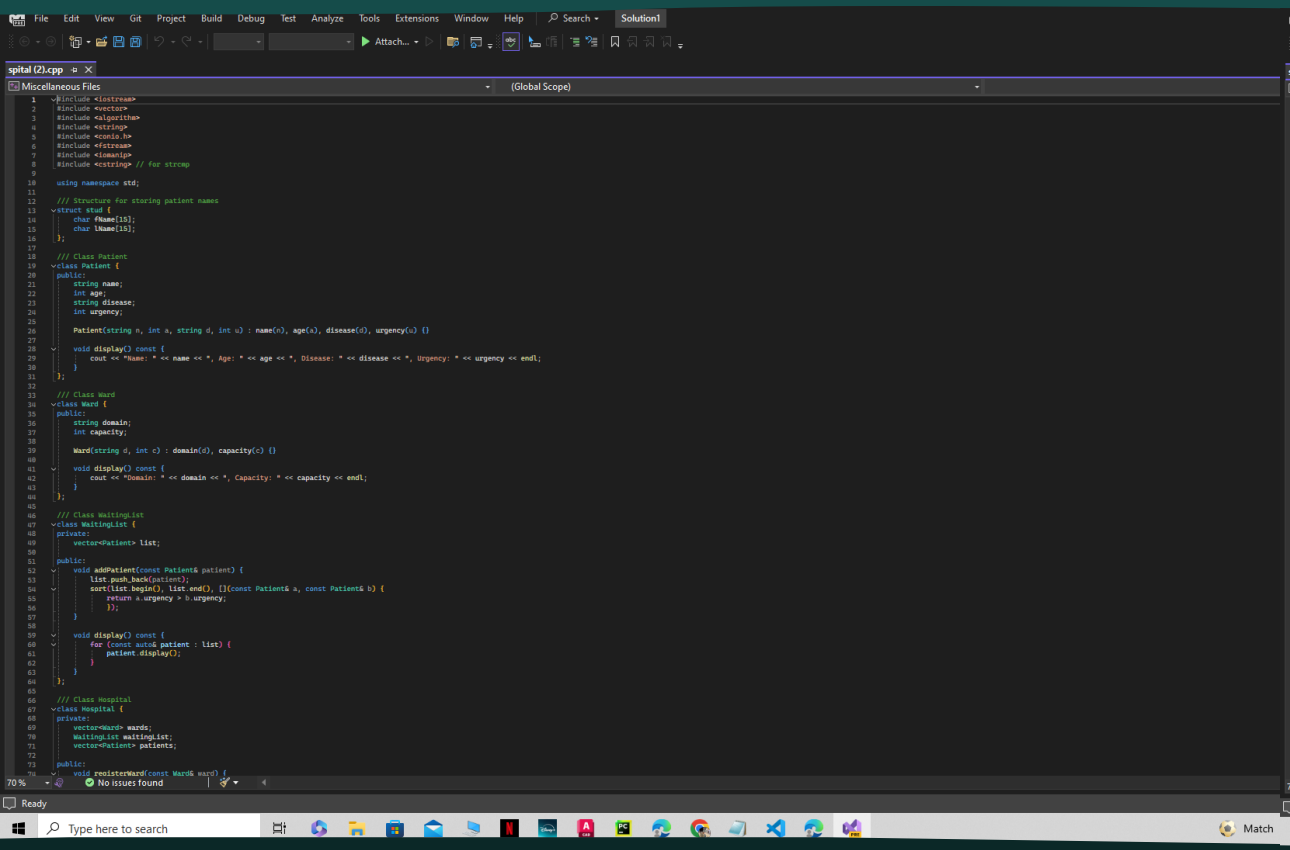
**Funcție**: `saveData`

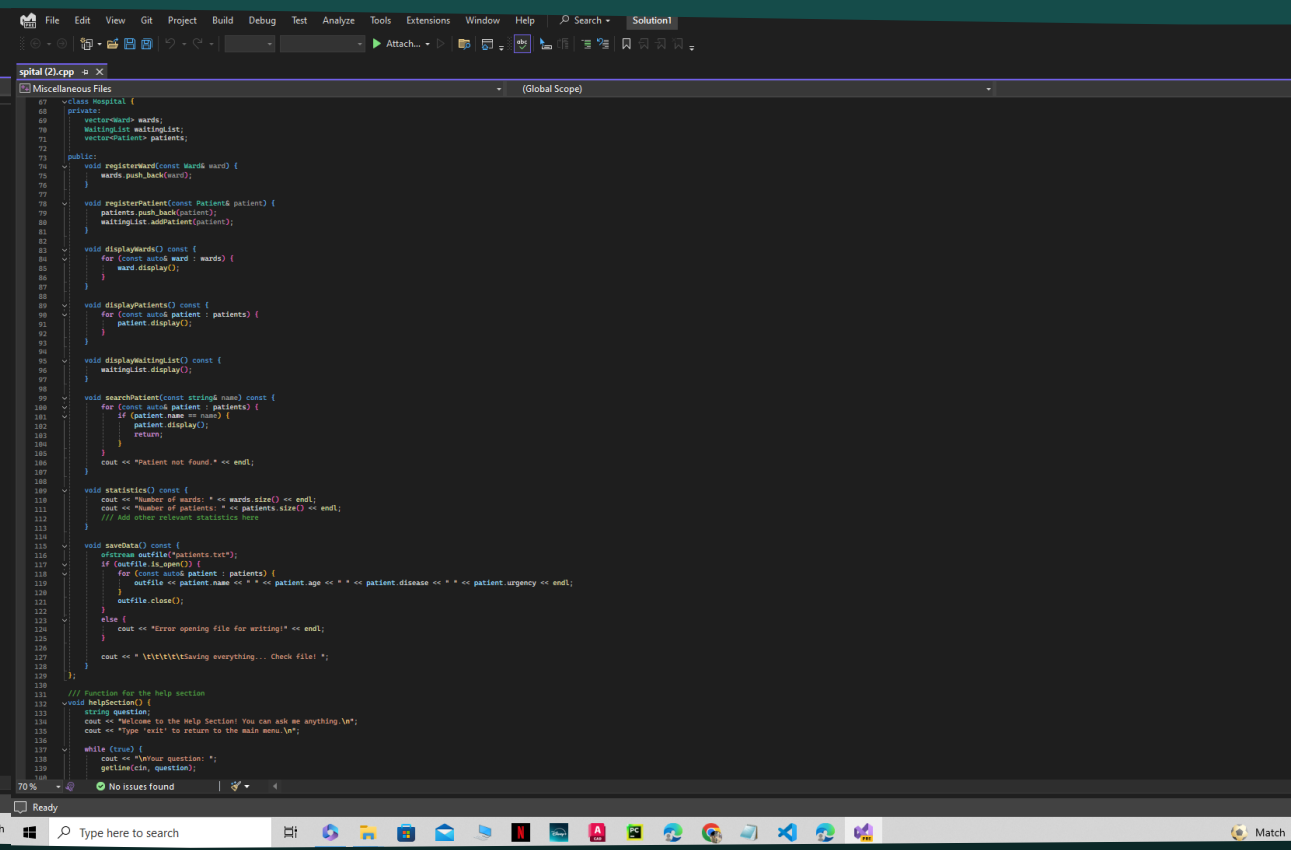
**Parametri**: Nu necesită parametri.

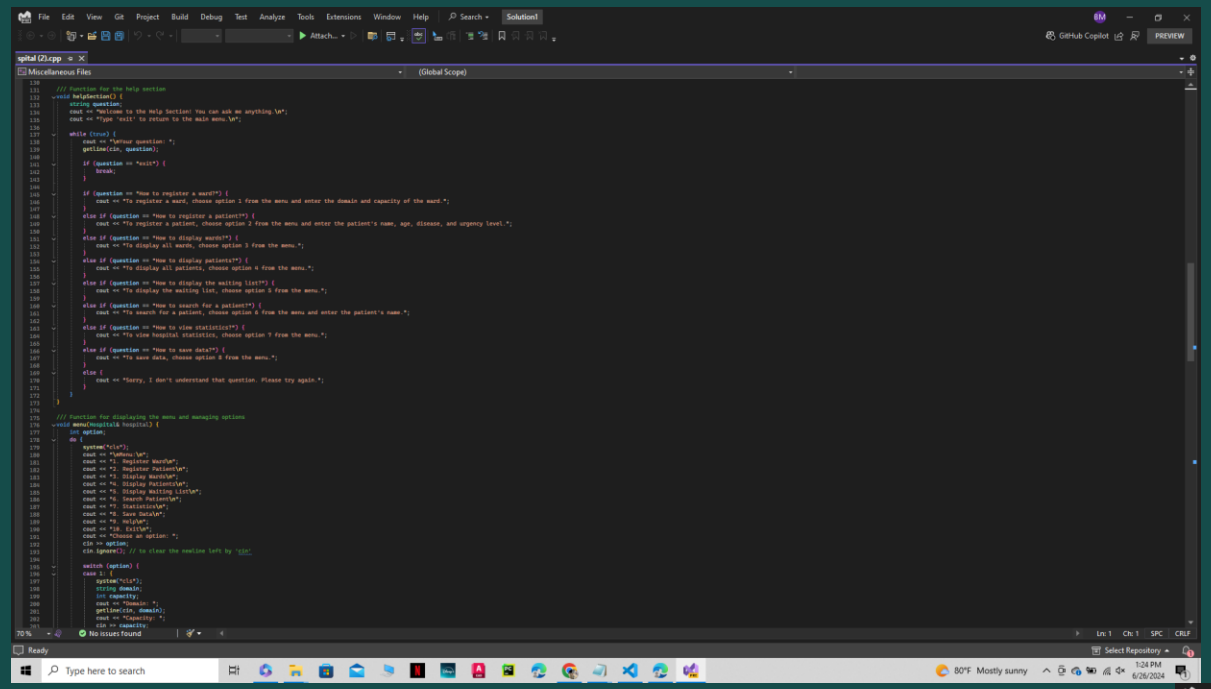
**Valori Returnate**: Nu există o valoare returnată.

**Algoritm Utilizat**: Salvarea datelor pacienților într-un fișier text.

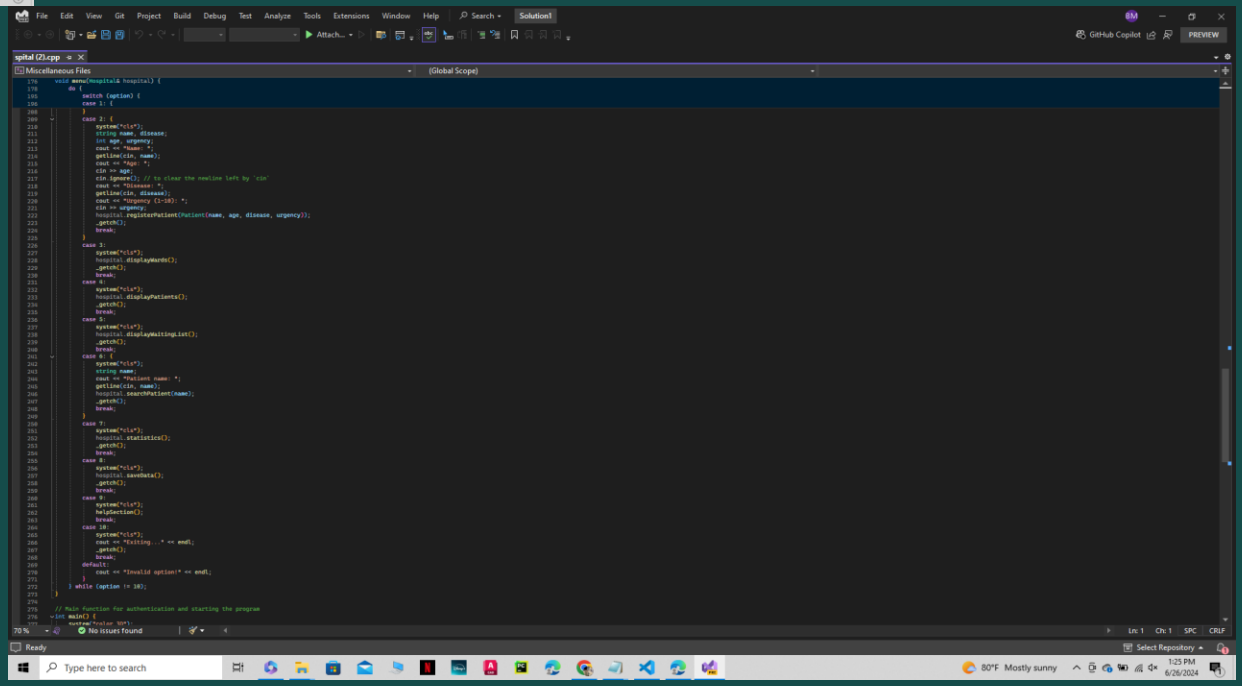
# CODUL COMPLET





```
/// Function for the help section
void helpSection() {
    string question;
    cout << "Welcome to the Help Section! You can ask me anything.\n";
    cout << "Type 'exit' to return to the main menu.\n";

    while (true) {
        cout << "\nYour question: ";
        getline(cin, question);

        if (question == "exit") {
            break;
        }

        if (question == "How to register a ward?") {
            cout << "To register a ward, choose option 1 from the menu and enter the domain and capacity of the ward.";
        }
        else if (question == "How to register a patient?") {
            cout << "To register a patient, choose option 2 from the menu and enter the patient's name, age, disease, and urgency level.";
        }
        else if (question == "How to display wards?") {
            cout << "To display all wards, choose option 3 from the menu.";
        }
        else if (question == "How to display patients?") {
            cout << "To display all patients, choose option 4 from the menu.";
        }
        else if (question == "How to display the waiting list?") {
            cout << "To display the waiting list, choose option 5 from the menu.";
        }
        else if (question == "How to search for a patient?") {
            cout << "To search for a patient, choose option 6 from the menu and enter the patient's name.";
        }
        else if (question == "How to view statistics?") {
            cout << "To view hospital statistics, choose option 7 from the menu.";
        }
        else if (question == "How to save data?") {
            cout << "To save data, choose option 8 from the menu.";
        }
        else {
            cout << "Sorry, I don't understand that question. Please try again.";
        }
    }
}

/// Function for displaying the menu and managing options
void menu(Hospital& hospital) {
    int option;
    do {
        system("cls");
        cout << "\nMenu:\n";
        cout << "1. Register Ward\n";
        cout << "2. Register Patient\n";
        cout << "3. Display Wards\n";
        cout << "4. Display Patients\n";
        cout << "5. Display Waiting List\n";
        cout << "6. Search Patient\n";
        cout << "7. Statistics\n";
        cout << "8. Save Data\n";
        cout << "9. Help\n";
        cout << "10. Exit\n";
        cout << "Choose an option: ";
        cin >> option;
        cin.ignore(); // to clear the newline left by 'cin'

        switch (option) {
        case 1: {
            system("cls");
            string domain;
            int capacity;
            cout << "Domain: ";
            getline(cin, domain);
            cout << "Capacity: ";
            cin >> capacity;
```



```
void menu(Hospital& hospital) {
    do {
        switch (option) {
        case 1: {
        }
        case 2: {
            system("cls");
            string name, disease;
            int age, urgency;
            cout << "Name: ";
            getline(cin, name);
            cout << "Age: ";
            cin >> age;
            cin.ignore(); // to clear the newline left by 'cin'
            cout << "Disease: ";
            getline(cin, disease);
            cout << "Urgency (1-10): ";
            cin >> urgency;
            hospital.registerPatient(Patient(name, age, disease, urgency));
            _getch();
            break;
        }
        case 3:
            system("cls");
            hospital.displayWards();
            _getch();
            break;
        case 4:
            system("cls");
            hospital.displayPatients();
            _getch();
            break;
        case 5:
            system("cls");
            hospital.displayWaitingList();
            _getch();
            break;
        case 6: {
            system("cls");
            string name;
            cout << "Patient name: ";
            getline(cin, name);
            hospital.searchPatient(name);
            _getch();
            break;
        }
        case 7:
            system("cls");
            hospital.statistics();
            _getch();
            break;
        case 8:
            system("cls");
            hospital.saveData();
            _getch();
            break;
        case 9:
            system("cls");
            helpSection();
            break;
        case 10:
            system("cls");
            cout << "Exiting..." << endl;
            _getch();
            break;
        default:
            cout << "Invalid option!" << endl;
        }
    } while (option != 10);
}

// Main function for authentication and starting the program
int main() {
```

```
...aneous Files                                    (Global Scope)
176   void menu(Hospital& hospital) {
178       do {
195           switch (option) {
262               helpSection();
263               break;
264           case 10:
265               system("cls");
266               cout << "Exiting..." << endl;
267               _getch();
268               break;
269           default:
270               cout << "Invalid option!" << endl;
271           }
272       } while (option != 10);
273   }
274
275   // Main function for authentication and starting the program
276   int main() {
277       system("color 3D");
278       char password[20], my_password[20] = "spital";
279       int i = 0;
280       char ch;
281
282       system("cls");
283       cout << "PASSWORD: ";
284
285       ch = _getch();
286       while (ch != 13) {
287           if (ch == 8) {
288               if (i > 0) {
289                   cout << "\b \b";
290                   i--;
291               }
292           }
293           else {
294               if (i < 19) {
295                   password[i] = ch;
296                   cout << "*";
297                   i++;
298               }
299           }
300           ch = _getch();
301       }
302       password[i] = '\0';
303
304       if (strcmp(password, my_password) != 0) { // verify password
305           cout << "\n\nIncorrect password !!!" << endl;
306           cout << "You typed: " << password << endl;
307           cout << "The correct password is: " << my_password << endl;
308           _getch();
309           return 1; // if the password is incorrect it will exit the program
310       }
311
312       cout << "\n\nThe password is correct so the program is executed!" << endl;
313       _getch();
314
315       Hospital hospital;
316       menu(hospital);
317
318       return 0;
319   }
320
```

# CONCLUZII

# CONCLUZII DIN REALIZAREA PROIECTULUI

- **Structurare și Organizare**: Proiectul a evidențiat importanța unei structuri bine definite și a unei organizații coerente a codului pentru gestionarea eficientă a datelor și funcționalităților.
- **Interfață Utilizator Simplificată**: O interfață simplificată, bazată pe meniuri, facilitează utilizarea și înțelegerea programului de către utilizatori.
- **Gestionarea Datelor**: Utilizarea claselor și a structurilor de date adecvate pentru gestionarea pacienților, saloanelor și a listei de așteptare demonstrează o abordare eficientă pentru manipularea și stocarea datelor.
- **Utilizarea Bibliotecilor Standard**: Folosirea bibliotecilor standard pentru operarea cu fișiere, manipularea șirurilor de caractere și alte operațiuni comune a redus complexitatea și a îmbunătățit fiabilitatea codului.

# PROPUNERI DE DEZVOLTARI ULTERIOARE

- **Interfață Grafică**: Implementarea unei interfețe grafice (GUI) ar putea îmbunătăți experiența utilizatorului prin furnizarea de elemente vizuale și interacțiuni mai intuitive.
- **Gestionarea Avansată a Pacienților**: Extinderea funcționalității pentru gestionarea istoricului medical al pacienților, programări pentru consultații sau intervenții medicale și alte detalii specifice.
- **Sisteme de Autentificare Avansate**: Implementarea unor sisteme de autentificare mai avansate, cum ar fi autentificarea cu două factori sau utilizarea unor tehnici de criptare pentru securitate sporită.
- **Optimizare Performanță**: Optimizarea algoritmilor și a structurilor de date pentru performanță sporită, în special în ceea ce privește operațiile de căutare și sortare.

# ELEMENTE DE LUAT IN CONSIDERARE LA REALIZAREA ALTOR PROIECTE

- **Planificare și Documentare**: O planificare și documentare adecvată a cerințelor și a structurii proiectului poate îmbunătăți procesul de dezvoltare și asigură că toți membrii echipei lucrează la un scop comun.
- **Testare și Debugging**: Testarea continuă a codului și debugarea sunt cruciale pentru identificarea și remedierea erorilor într-un stadiu incipient al dezvoltării.
- **Flexibilitate și Scalabilitate**: Proiectele ar trebui să fie construite cu o arhitectură flexibilă și scalabilă, pentru a permite adăugarea de noi funcționalități și extinderea în viitor.
- **Securitate**: Asigurarea securității datelor și implementarea unor măsuri adecvate de securitate sunt esențiale, mai ales în aplicații care gestionează informații sensibile, cum ar fi datele medicale.

# BIBLIOGRAFIE


- PRIMA POZA:https://www.google.com/imgres?q=unitbv%20logo%20png&imgurl=https%3A%2F%2Fasset.brandfetch.io%2FiddZOuWTRB%2Fidw8aDKkyL.jpeg&imgrefurl=https%3A%2F%2Fbrandfetch.com%2Funitbv.ro&docid=-z0RjjelkTtNLM&tbnid=fVavFwe0W8OPNM&vet=12ahUKEwiwxuickZ-GAxXK_rsIHYlWAjUQM3oECGgQAA..i&w=836&h=836&hcb=2&ved=2ahUKEwiwxuickZ-GAxXK_rsIHYlWAjUQM3oECGgQAA
- A DOUA POZA:https://www.google.com/imgres?q=GESTIONAREA%20UNUI%20SPITAL&imgurl=https%3A%2F%2Fgamait.ro%2Fwp-content%2Fuploads%2F2021%2F09%2FMED.png&imgrefurl=https%3A%2F%2Fgamait.ro%2Fatlas-med%2F&docid=q0qichdJ6cj2uM&tbnid=EXDQZo4OIqNsuM&vet=12ahUKEwi5-6SvkZ-GAxUUg_0HHQJWBnEQM3oECG8QAA..i&w=1024&h=493&hcb=2&ved=2ahUKEwi5-6SvkZ-GAxUUg_0HHQJWBnEQM3oECG8QAA