SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Data Structure -MINI PROJECT

Topic-Job application

Name-Ishaan wawa

Roll number- F019

Abstract

The purpose of the job application system is to make it easier for admin and job searchers (users) to post jobs, apply for jobs, and view application. Admin will handle job applications, evaluate applicants, and decide who gets hired thanks to the system. Users can look through open vacancies, submit applications, and monitor the progress of their applications. This leads to reduction of manual labour and the maintenance of an organized workflow; this technology improves the hiring process efficiency.

The Job Application System is a console-based application built using the C++ programming language. It provides a basic digital platform for employers (admins) and job seekers (users) to interact in a structured and organized manner. By offering essential functionalities such as job posting, application tracking, and decision management, the system eliminates the need for traditional paper-based recruitment methods.

Admins can securely log into the system, add job listings, edit or remove them, and monitor applications submitted by users. They have the authority to accept or reject applications based on predefined criteria. This ensures that the hiring process is efficiently monitored and controlled from a central point.

On the user side, individuals can register, log in, view the list of available job opportunities, and apply to positions that match their interests. Users are also able to monitor the status of their applications and receive updates on whether their application was accepted or rejected.

The system provides a reliable and secure authentication process for both users and admins, promoting privacy and data security. While the project is implemented in a basic console format, it offers significant insight into structured programming and system design and can be expanded further with graphical interfaces or database integration.

In essence, this Job Application System aims to serve small institutions or academic purposes by demonstrating how software can streamline recruitment, improve communication, and reduce manual effort, all while serving as a foundation for more advanced systems.

Introduction

Objective

The fundamental goal of this Job Application System is to enable a smooth connection between job seekers and companies by offering an organized and efficient platform for job applications. The system guarantees that job listings are properly managed, applications are quickly handled, and responses are successfully communicated.

Problem Statement

Many traditional job application processes require paperwork, manual tracking, and ineffective communication between applicants and recruiters. These methods frequently result in lost applications, delayed responses, and misunderstanding among candidates. The suggested solution intends to digitize and automate this process, resulting in improved organization, transparency, and usability for both businesses and job searchers.

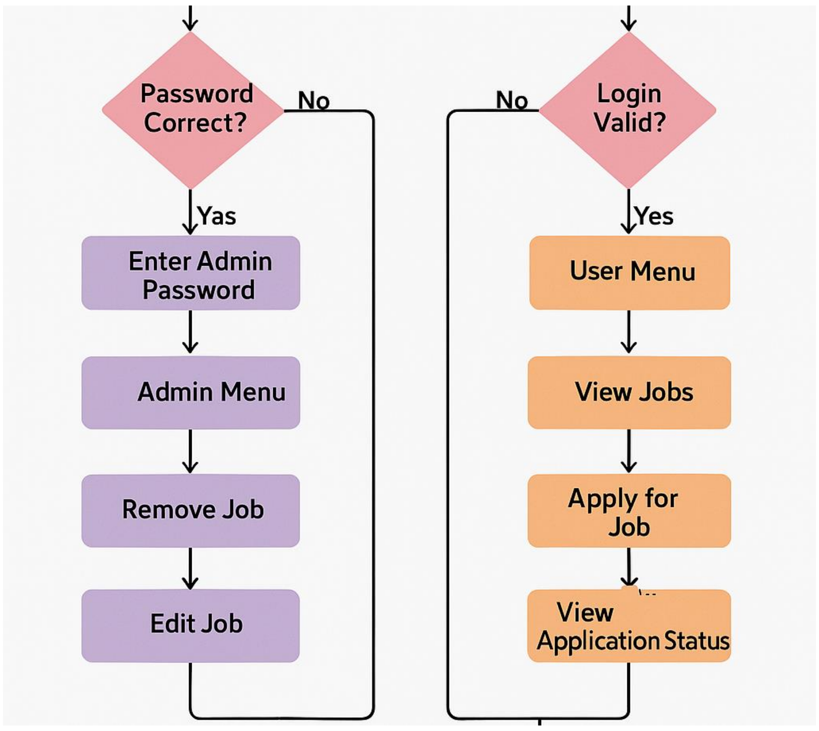
Scope

This project is intended to benefit small and medium-sized businesses looking for an efficient solution to manage job applications.

The system will contain:

1. Administrators can post jobs.
2. Users can submit their job applications.
3. Application monitoring and response management.
4. Secure login for both users and administrators.
5. A navigation interface that is well-organized and easy to use.

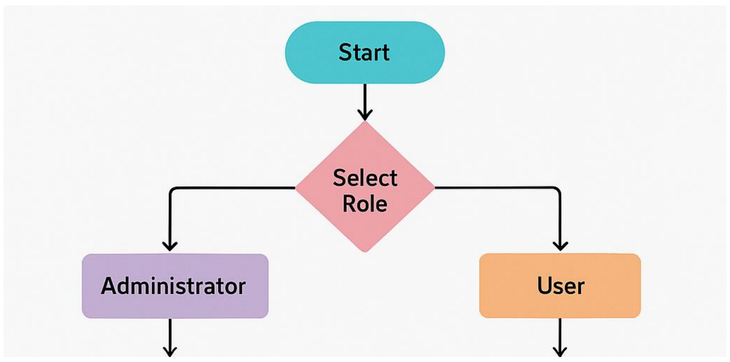
Methodology



Detailed Explanation of Features

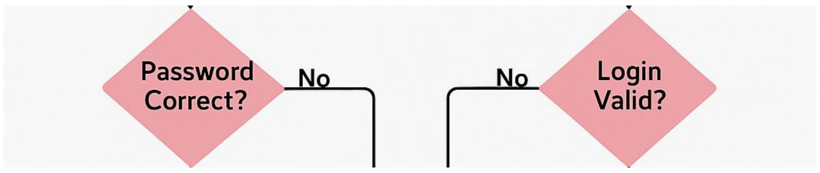
1. Select Role (User/Admin):

    - At the start of the program, the user is prompted to choose whether they are an Admin or a User. Based on the role selection, different functionalities become accessible.



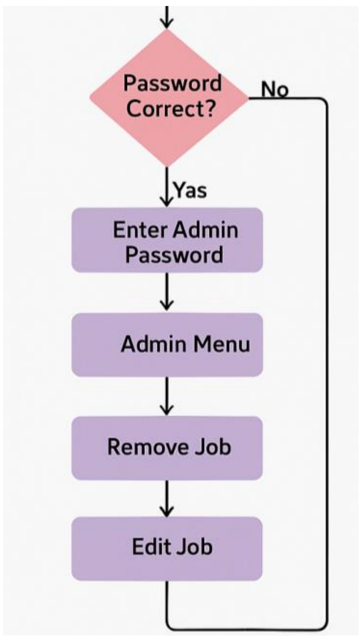
2. Authentication:

    - Admin: Enters a pre-set password to access job management features.
    - User: Either registers or logs in with a username-password combination.



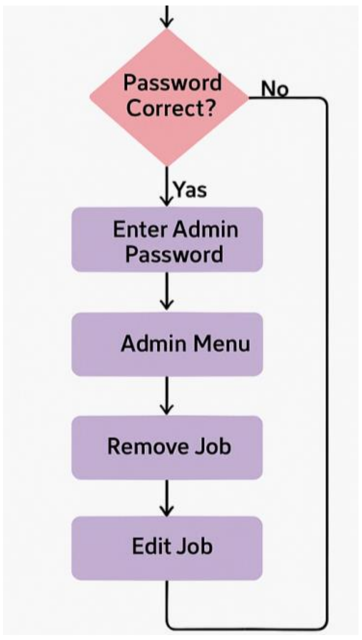
3. Job Management (Admin):

    - Admin can add jobs with details like job title and description.
    - Admin can edit or update job details if there is any change in requirement.
    - Admin can delete job listings that are no longer available.
    - The list of all active job postings is viewable for reference and updates.



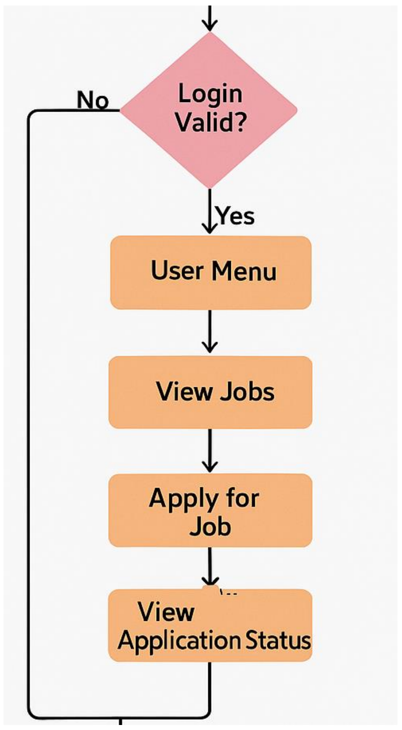
4. Application Management (Admin):

- Admin can view the applications submitted by users.
- Each application includes the applicant's name and the job applied for.
- Admin can mark an application as accepted or rejected.



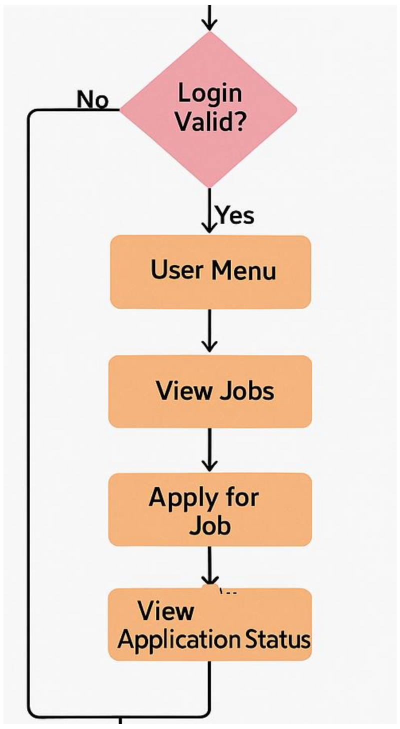
5. Job Browsing and Application (User):

- After login, users see the list of currently available jobs.
- Users can apply to specific jobs by entering the job ID.



6. Application Status (User):

- Once applied, the application is visible to the admin for review.
- Users can later view whether their application was accepted or rejected.



Technologies Used

- Programming language: C++
- Data storage method (arrays in this case)
- Future improvements (e.g., using databases for scalability)

C++ Concepts Used

1. **Functions**:
   - The code is modularized using functions to perform tasks such as login, job listing, applying for jobs, etc. This helps improve code organization and reusability.
2. **Arrays**:
   - Arrays are used to store job details, user credentials, and application statuses. Arrays allow fixed-size, indexed storage of multiple elements.
3. **Loops (for, while)**:
   - Loops are used to iterate over arrays for displaying job listings, checking login credentials, and managing application lists.
4. **Conditional Statements (if-else, switch-case)**:

- Used to make decisions based on user input (e.g., whether login is successful, which menu option is selected).

5. **String Handling (char arrays)**:
    - Usernames, passwords, and job titles are stored and compared using C-style strings (character arrays).
6. **Input/Output (cin, cout)**:
    - Standard input and output functions are used for interacting with the user via the console.
7. **Boolean Flags**:
    - Flags are used to track login status and check conditions (e.g., whether a user is registered or not).
8. **Goto Statement (optional)**:
    - May be used in basic navigation or redirection back to menu in simpler versions of the program (though generally discouraged in modern C++)

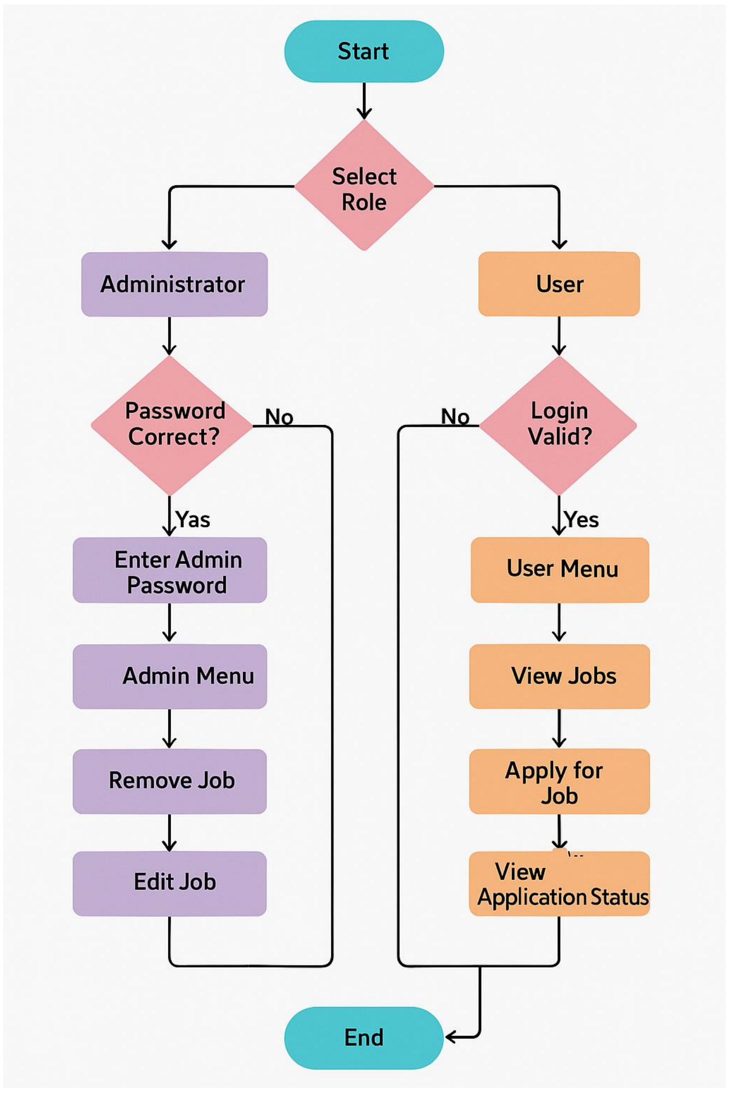
Challenges Faced

1. Managing job deletions while keeping the list structured.

2. Creating a strong and effective login system for users and administrators.

3. Using structured programming to efficiently track applications.

Implementation Details



Explanation

Admin Features and Explanation

1. **Admin Login**: Admin is required to enter a secure password to access the admin dashboard.
2. **Add Job**: Allows the admin to enter job title, description, and job ID to post new job opportunities.
3. **Edit Job**: Enables the admin to modify the details of any existing job.
4. **Remove Job**: Admin can remove a job from the listing by specifying its job ID.
5. **View Jobs**: Lists all current job openings added by the admin.
6. **View Applied Jobs**: Displays all applications submitted by users.
7. **Accept/Reject Applications**: Admin reviews the applications and decides whether to accept or reject them.

User Features and Explanation

1. **User Registration**: New users can create an account with a username and password.
2. **User Login**: Registered users log in using their credentials to access the user panel.
3. **View Jobs**: Displays the list of jobs posted by the admin.
4. **Apply for Jobs**: Users can select specific job IDs and apply.
5. **View Application Status**: Users can check if their applications have been accepted or rejected.

Complete Code Explanation (Overview)

1. **Data Structures**:
    - Arrays store user data, job data, and application status.
2. **Role Selection**:
    - At the start, the user selects between Admin or User.
3. **Authentication**:
    - Admin has a fixed password, and users register and log in using stored credentials.
4. **Functional Segregation**:
    - Different functions handle adding jobs, editing jobs, applying to jobs, and managing applications.

5. **Flow**:
    - Admin adds job → User applies → Admin views and decides → User checks application result.

Conclusion and Future Enhancement

- Database integration refers to the permanent storage of user data and job applications.
- To improve user experience.
- Email notifications are used to keep people up to date on the status of their applications.
- It allows consumers to rapidly identify appropriate job opportunities.
- This feature allows you to provide more information with your job application.

References

1. Books & Online Courses

    Bjarne Stroustrup, The C++ Programming Language

    Robert Lafore, Object-Oriented Programming in C++

    GeeksforGeeks, C++ Programming Concepts'

    TutorialsPoint, C++ Job Management Systems

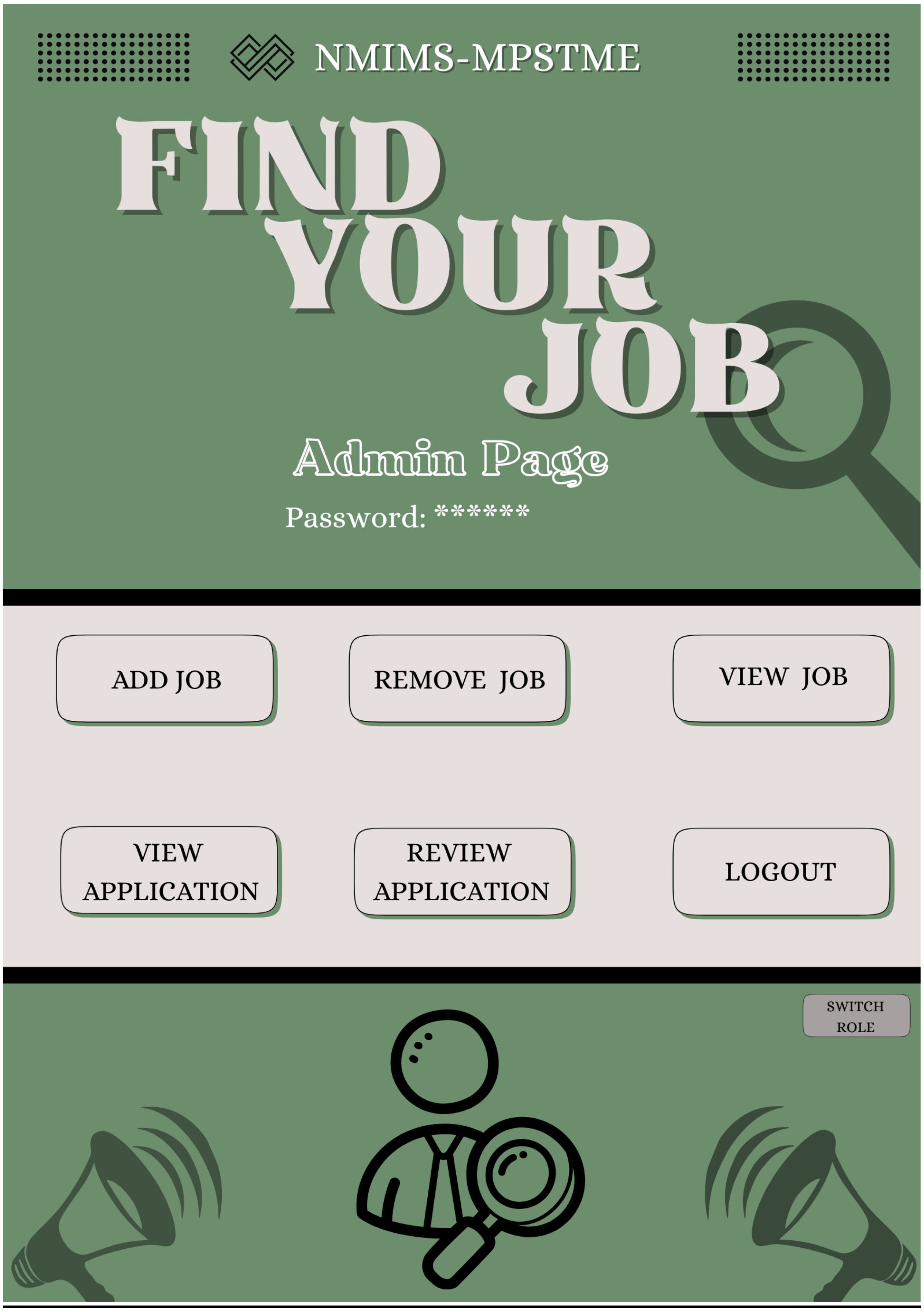
2. Project Examples & Tutorials:

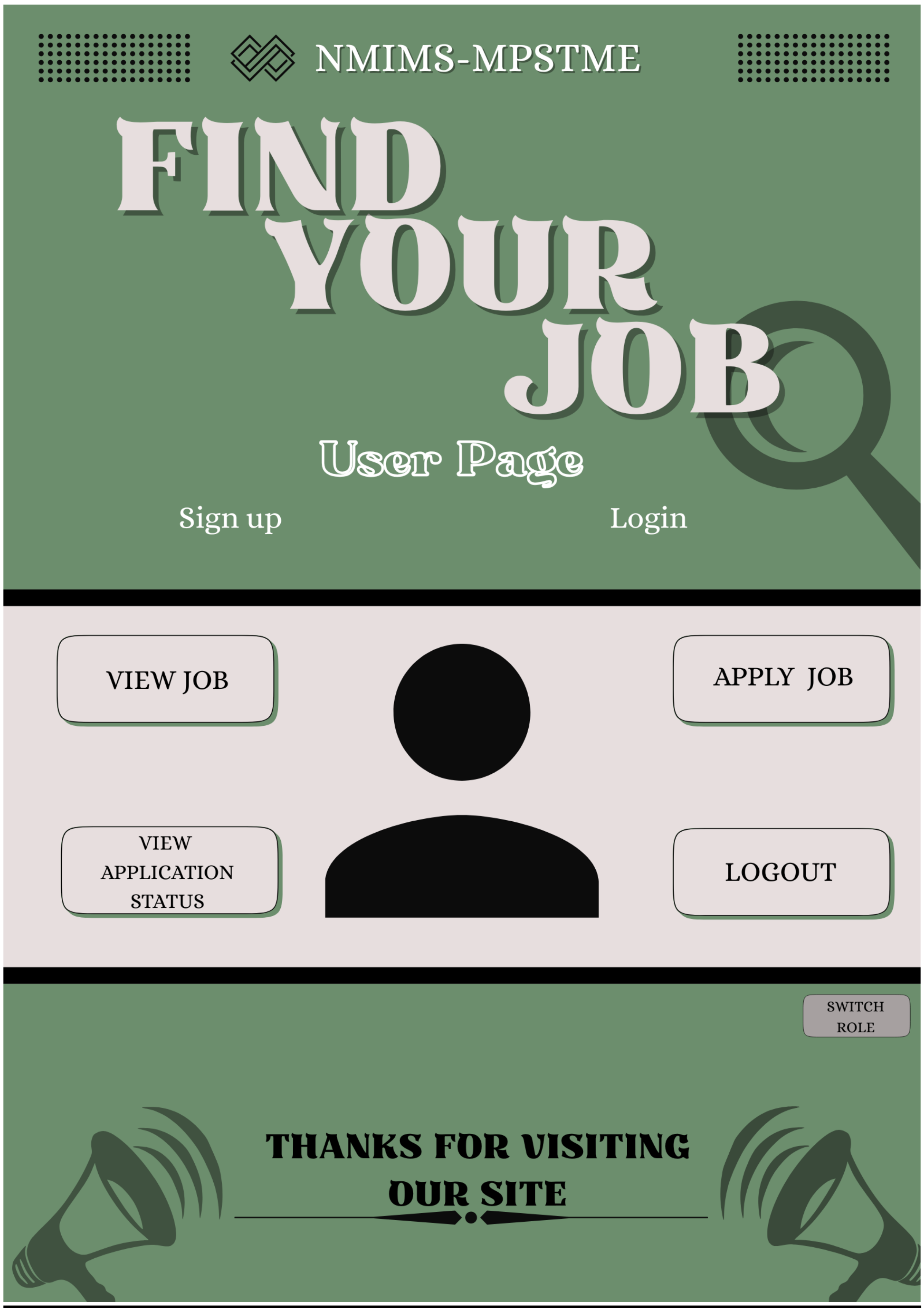
    GitHub repositories on C++ Job Management Systems

    YouTube tutorials on Building Job Portals with C++

    Research papers on Job Recruitment Systems and Algorithms

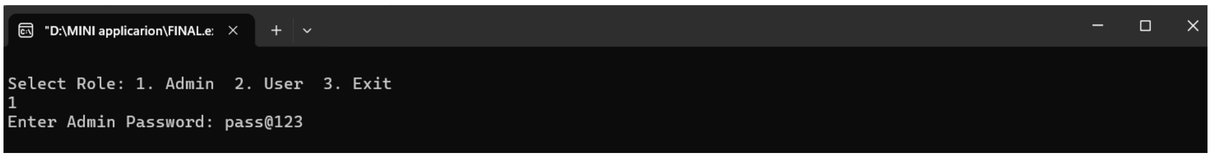
Poster:


NMIMS-MPSTME
FIND YOUR JOB
Admin Page
Password: ******
ADD JOB
REMOVE JOB
VIEW JOB
VIEW APPLICATION
REVIEW APPLICATION
LOGOUT
SWITCH ROLE


NMIMS-MPSTME
FIND YOUR JOB
User Page
Sign up
Login
VIEW JOB
APPLY JOB
VIEW APPLICATION STATUS
LOGOUT
SWITCH ROLE
THANKS FOR VISITING OUR SITE

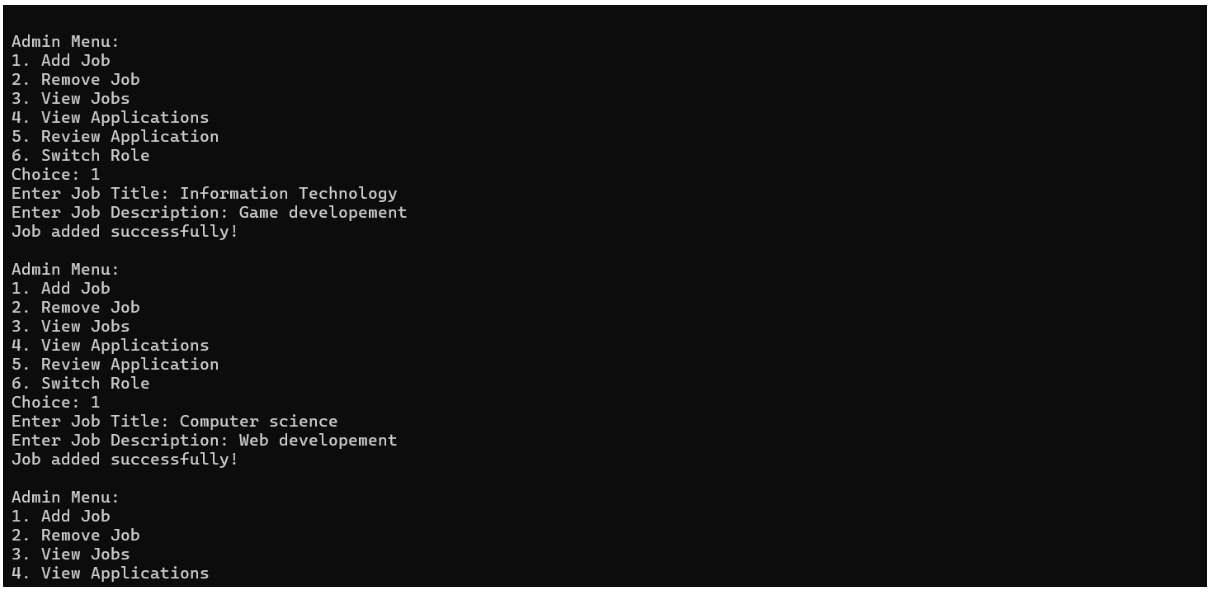
Output

1)Admin login system

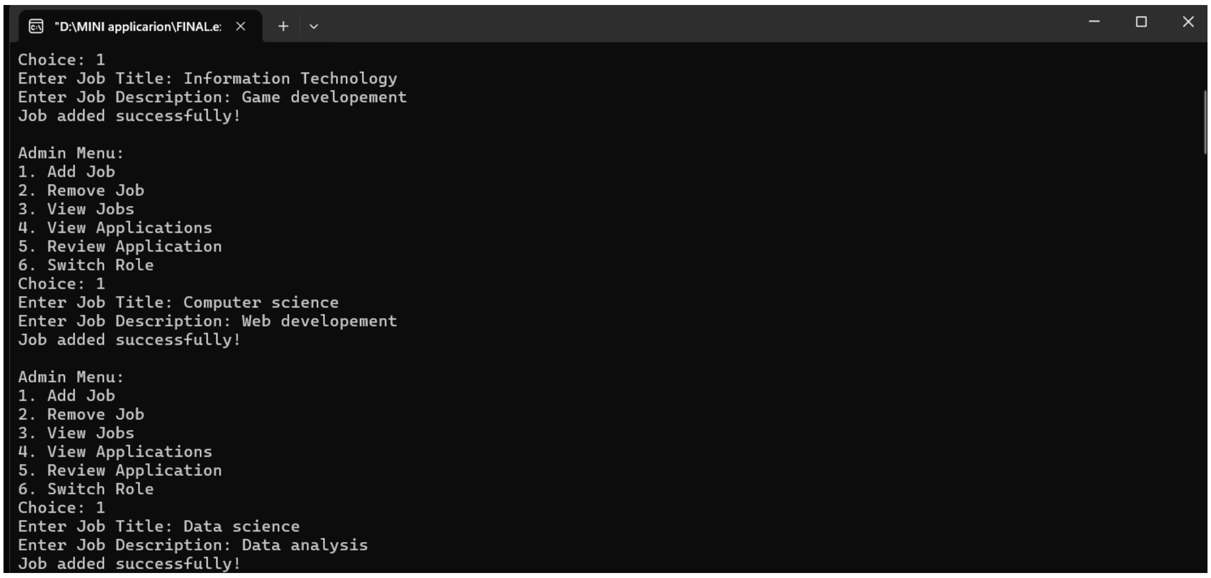


```
Select Role: 1. Admin  2. User  3. Exit
1
Enter Admin Password: pass@123
```

2)Add 2-3 jobs to your app

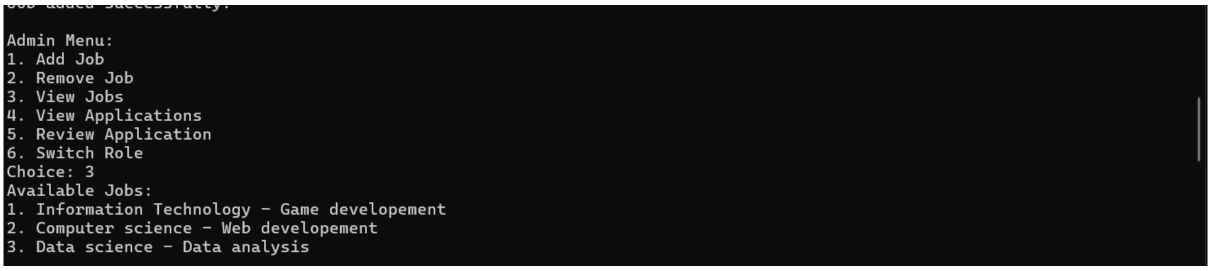


```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 1
Enter Job Title: Information Technology
Enter Job Description: Game developement
Job added successfully!

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 1
Enter Job Title: Computer science
Enter Job Description: Web developement
Job added successfully!

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
```

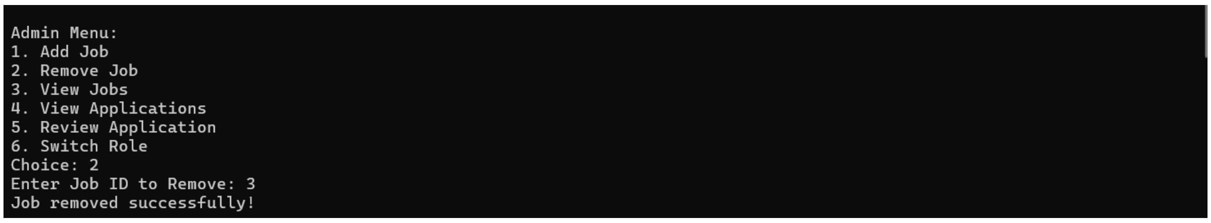


```
Choice: 1
Enter Job Title: Information Technology
Enter Job Description: Game developement
Job added successfully!

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 1
Enter Job Title: Computer science
Enter Job Description: Web developement
Job added successfully!

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 1
Enter Job Title: Data science
Enter Job Description: Data analysis
Job added successfully!
```

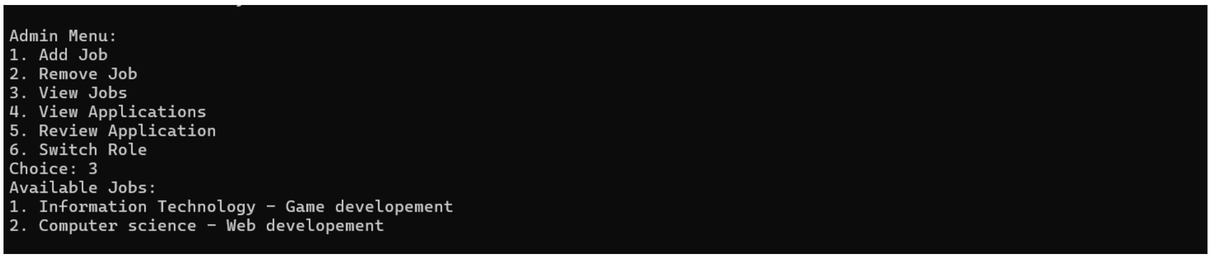
3)To view number of jobs available


```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 3
Available Jobs:
1. Information Technology - Game developement
2. Computer science - Web developement
3. Data science - Data analysis
```

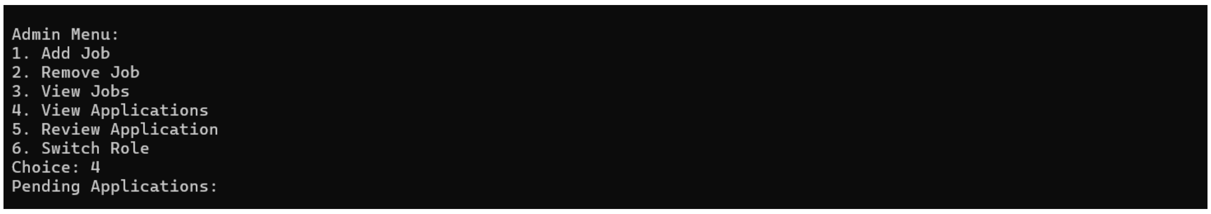
4)To remove the job you want


```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 2
Enter Job ID to Remove: 3
Job removed successfully!
```

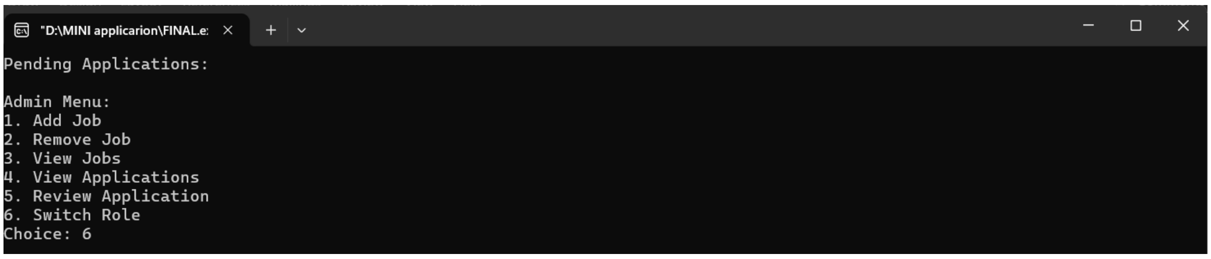
5)Check if it is removed(proof)


```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 3
Available Jobs:
1. Information Technology - Game developement
2. Computer science - Web developement
```

6)To view applications. Since you have not applied for any job there will be no applicants

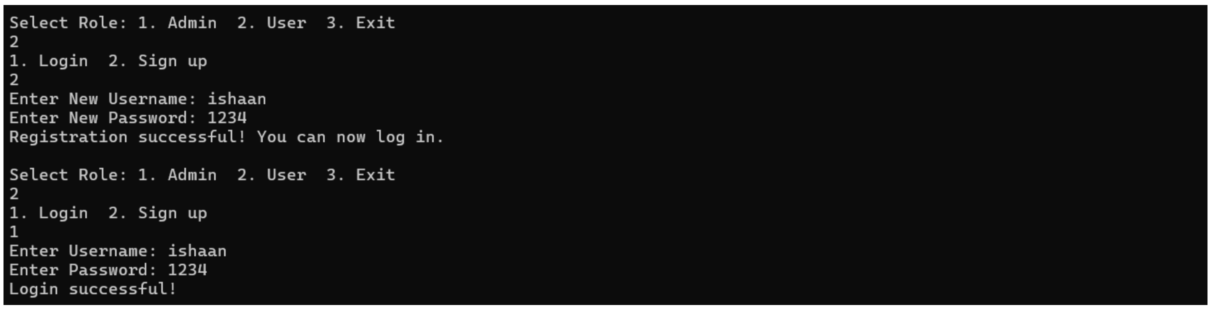

```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 4
Pending Applications:
```

7)Switch roles



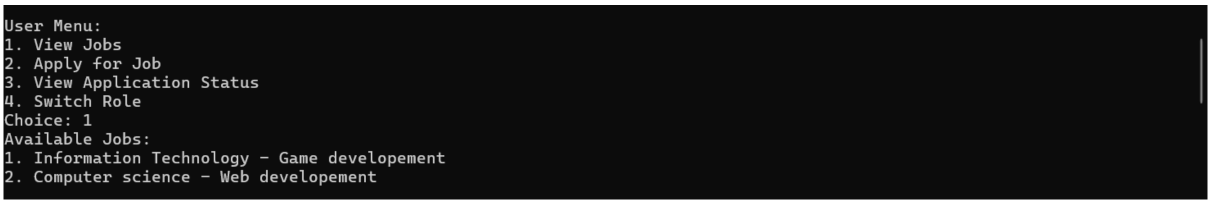
```
Pending Applications:

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 6
```

8)Login system for User



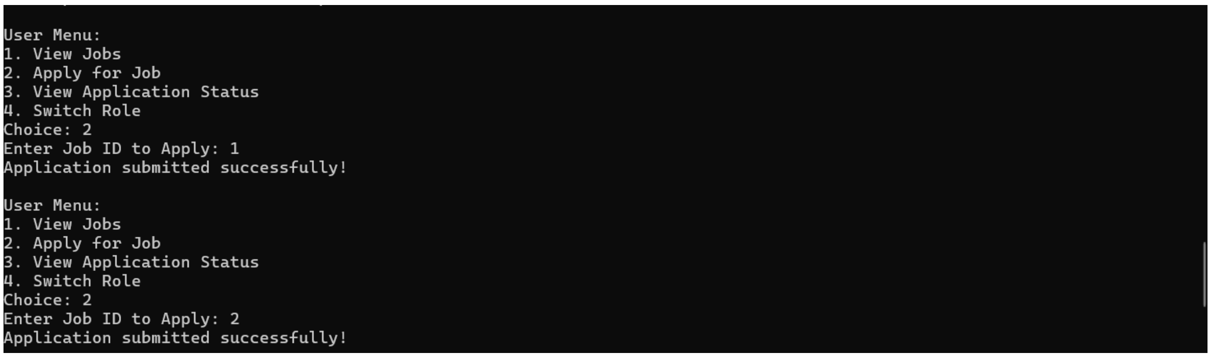
```
Select Role: 1. Admin  2. User  3. Exit
2
1. Login  2. Sign up
2
Enter New Username: ishaan
Enter New Password: 1234
Registration successful! You can now log in.

Select Role: 1. Admin  2. User  3. Exit
2
1. Login  2. Sign up
1
Enter Username: ishaan
Enter Password: 1234
Login successful!
```

9)View jobs as a job seeker



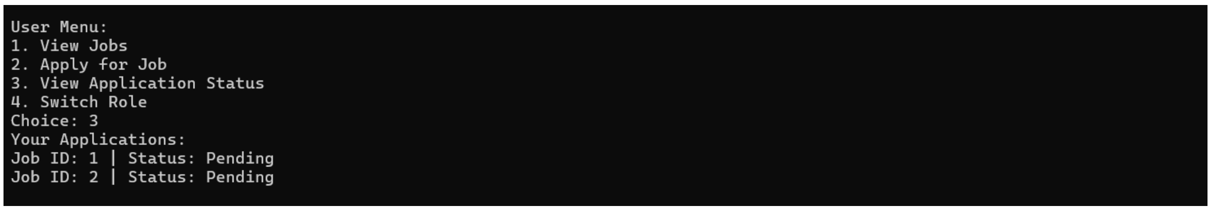
```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 1
Available Jobs:
1. Information Technology - Game developement
2. Computer science - Web developement
```

10)Apply for a job



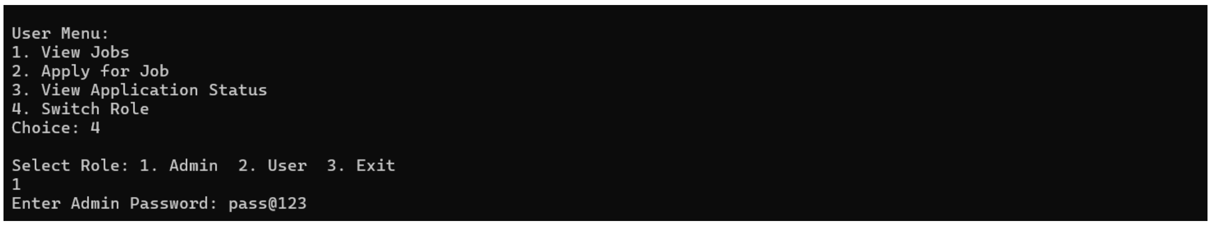
```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 2
Enter Job ID to Apply: 1
Application submitted successfully!

User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 2
Enter Job ID to Apply: 2
Application submitted successfully!
```

11)Check the status



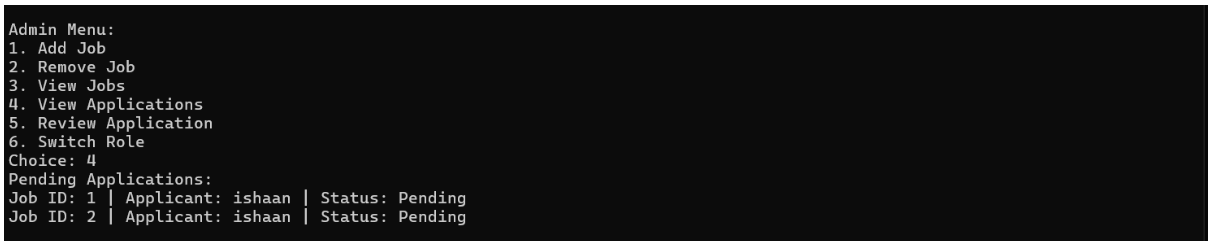
```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 3
Your Applications:
Job ID: 1 | Status: Pending
Job ID: 2 | Status: Pending
```

12)Switch role to admin to hire the job seeker



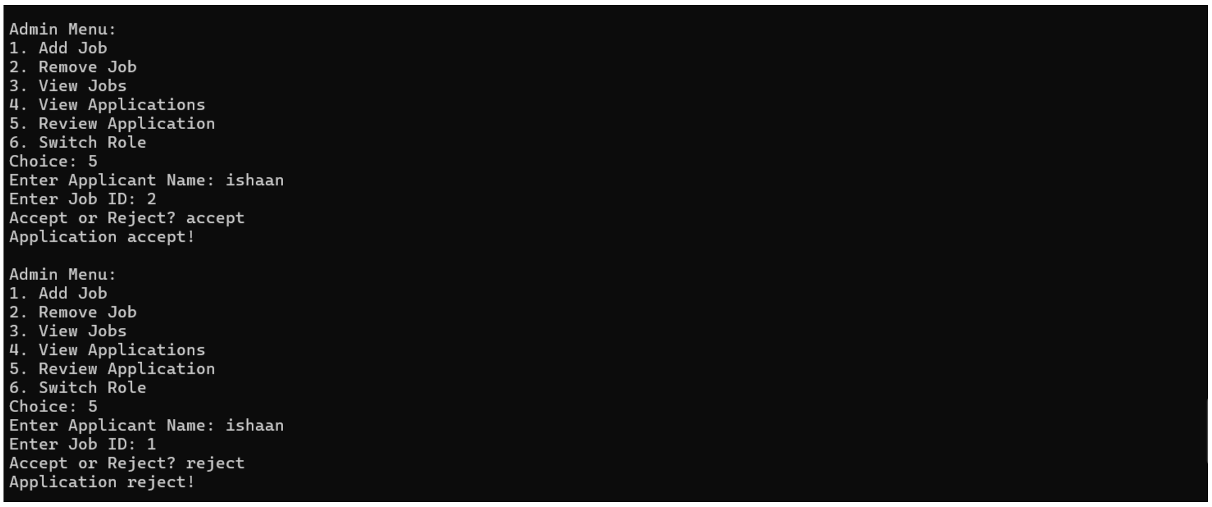
```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 4

Select Role: 1. Admin  2. User  3. Exit
1
Enter Admin Password: pass@123
```

13)View pending applications



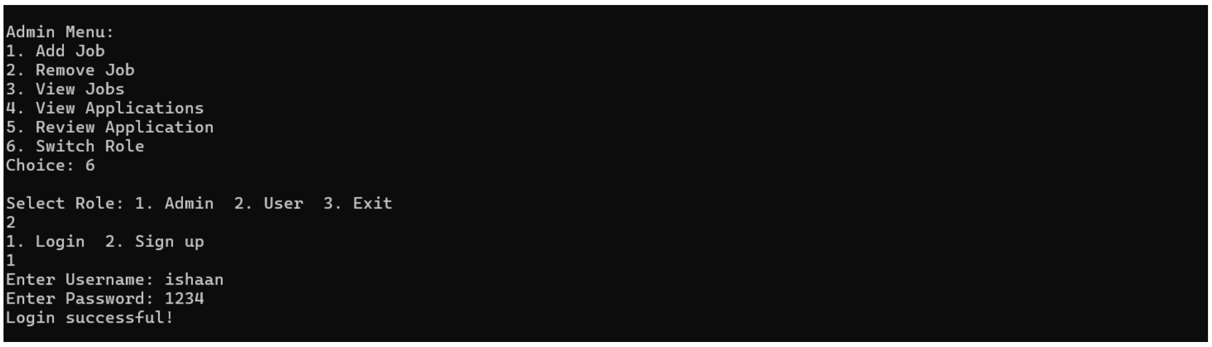
```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 4
Pending Applications:
Job ID: 1 | Applicant: ishaan | Status: Pending
Job ID: 2 | Applicant: ishaan | Status: Pending
```

14)Make your decision accept or reject



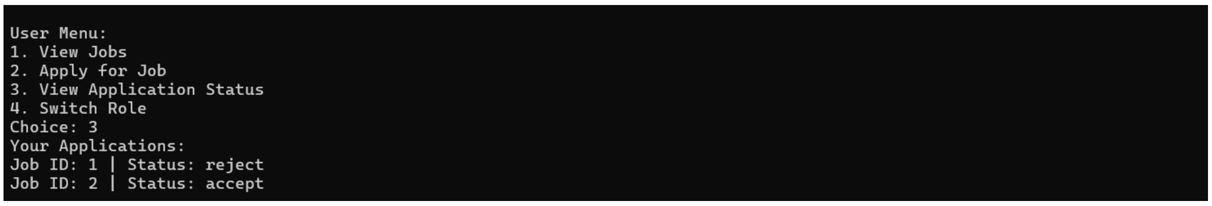
```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 5
Enter Applicant Name: ishaan
Enter Job ID: 2
Accept or Reject? accept
Application accept!

Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 5
Enter Applicant Name: ishaan
Enter Job ID: 1
Accept or Reject? reject
Application reject!
```

15)Switch role to user to find out your review



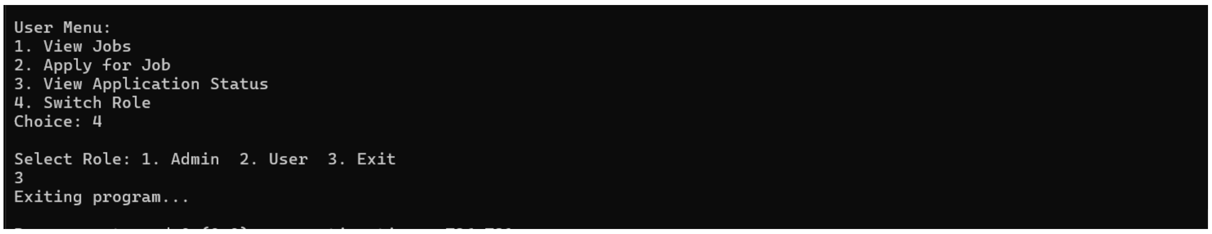
```
Admin Menu:
1. Add Job
2. Remove Job
3. View Jobs
4. View Applications
5. Review Application
6. Switch Role
Choice: 6

Select Role: 1. Admin  2. User  3. Exit
2
1. Login  2. Sign up
1
Enter Username: ishaan
Enter Password: 1234
Login successful!
```

16)View your application status



```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 3
Your Applications:
Job ID: 1 | Status: reject
Job ID: 2 | Status: accept
```

17)Exit



```
User Menu:
1. View Jobs
2. Apply for Job
3. View Application Status
4. Switch Role
Choice: 4

Select Role: 1. Admin  2. User  3. Exit
3
Exiting program...
```

CODE:

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

const int MAX_JOBS = 100;
const int MAX_APPLICATIONS = 100;
const int MAX_USERS = 100;
const string ADMIN_PASSWORD = "pass@123";

struct Job {
    int id;
    string title;
    string description;
};

struct Application {
    int jobId;
    string userName;
    string status;
};

struct User {
    string userName;
    string password;
};

Job jobs[MAX_JOBS];
Application applications[MAX_APPLICATIONS];
User users[MAX_USERS];
int jobCount = 0;
int applicationCount = 0;
int userCount = 0;

void addJob(string title, string description) {
    if (jobCount < MAX_JOBS) {
        jobs[jobCount] = {jobCount + 1, title, description};
        jobCount++;
        cout << "Job added successfully!\n";
    } else {
        cout << "Job list is full!\n";
    }
}

void removeJob(int jobId) {
    if (jobId < 1 || jobId > jobCount) {
        cout << "Invalid Job ID!\n";
        return;
    }
    jobs[jobId - 1] = jobs[jobCount - 1];
    jobCount--;
    cout << "Job removed successfully!\n";
}
```

```
void viewJobs() {
    cout << "Available Jobs:\n";
    for (int i = 0; i < jobCount; i++) {
        cout << jobs[i].id << ". " << jobs[i].title << " - " << jobs[i].description << "\n";
    }
}

void applyJob(int jobId, string userName) {
    if (jobId < 1 || jobId > jobCount) {
        cout << "Invalid Job ID!\n";
        return;
    }
    applications[applicationCount++] = {jobId, userName, "Pending"};
    cout << "Application submitted successfully!\n";
}

void viewApplications() {
    cout << "Pending Applications:\n";
    for (int i = 0; i < applicationCount; i++) {
        if (applications[i].status == "Pending") {
            cout << "Job ID: " << applications[i].jobId << " | Applicant: " <<
applications[i].userName << " | Status: " << applications[i].status << "\n";
        }
    }
}

void reviewApplication(string userName, int jobId, string decision) {
    for (int i = 0; i < applicationCount; i++) {
        if (applications[i].jobId == jobId && applications[i].userName == userName) {
            applications[i].status = decision;
            cout << "Application " << decision << "!\n";
            return;
        }
    }
    cout << "Application not found!\n";
}

void viewUserApplications(string userName) {
    cout << "Your Applications:\n";
    for (int i = 0; i < applicationCount; i++) {
        if (applications[i].userName == userName) {
            cout << "Job ID: " << applications[i].jobId << " | Status: " <<
applications[i].status << "\n";
        }
    }
}

bool userLogin(string &userName) {
    string password;
    cout << "Enter Username: ";
    cin >> userName;
    cout << "Enter Password: ";
    cin >> password;

    for (int i = 0; i < userCount; i++) {
        if (users[i].userName == userName && users[i].password == password) {
            cout << "Login successful!\n";
            return true;
        }
    }
    cout << "Invalid credentials!\n";
    return false;
```

```
}

void userRegister() {
    if (userCount >= MAX_USERS) {
        cout << "User limit reached!\n";
        return;
    }
    string userName, password;
    cout << "Enter New Username: ";
    cin >> userName;
    cout << "Enter New Password: ";
    cin >> password;
    users[userCount++] = {userName, password};
    cout << "Registration successful! You can now log in.\n";
}

int main() {
    int choice, role, jobId;
    string title, description, userName, decision, password;

    while (true) {
        cout << "\nSelect Role: 1. Admin  2. User  3. Exit\n";
        cin >> role;

        if (role == 1) {
            cout << "Enter Admin Password: ";
            cin >> password;
            if (password != ADMIN_PASSWORD) {
                cout << "Incorrect password!\n";
                continue;
            }
            while (true) {
                cout << "\nAdmin Menu:\n1. Add Job\n2. Remove Job\n3. View Jobs\n4. View Applications\n5. Review Application\n6. Switch Role\nChoice: ";
                cin >> choice;

                switch (choice) {
                    case 1:
                        cout << "Enter Job Title: ";
                        cin.ignore();
                        getline(cin, title);
                        cout << "Enter Job Description: ";
                        getline(cin, description);
                        addJob(title, description);
                        break;
                    case 2:
                        cout << "Enter Job ID to Remove: ";
                        cin >> jobId;
                        removeJob(jobId);
                        break;
                    case 3:
                        viewJobs();
                        break;
                    case 4:
                        viewApplications();
                        break;
                    case 5:
                        cout << "Enter Applicant Name: ";
                        cin >> userName;
                        cout << "Enter Job ID: ";
                        cin >> jobId;
```

```
                    cout << "Accept or Reject? ";
                    cin >> decision;
                    reviewApplication(userName, jobId, decision);
                    break;
                case 6:
                    goto role_selection;
                default:
                    cout << "Invalid choice!\n";
            }
        }
    } else if (role == 2) {
        int userChoice;
        cout << "1. Login  2. Sign up\n";
        cin >> userChoice;
        if (userChoice == 2) {
            userRegister();
            continue;
        }
        if (!userLogin(userName)) continue;

        while (true) {
            cout << "\nUser Menu:\n1. View Jobs\n2. Apply for Job\n3. View
Application Status\n4. Switch Role\nChoice: ";
            cin >> choice;

            switch (choice) {
                case 1:
                    viewJobs();
                    break;
                case 2:
                    cout << "Enter Job ID to Apply: ";
                    cin >> jobId;
                    applyJob(jobId, userName);
                    break;
                case 3:
                    viewUserApplications(userName);
                    break;
                case 4:
                    goto role_selection;
                default:
                    cout << "Invalid choice!\n";
            }
        }
    } else if (role == 3) {
        cout << "Exiting program...\n";
        break;
    } else {
        cout << "Invalid role selection!\n";
    }

    role_selection:;
  }
  return 0;
}
```

****