OOP PROJECT REPORT

(Submitted to Ms. Shazia Paras)

NAIKI

GROUP MEMBERS:

NAME		NU_ID
1. Subhan Asif	---->	22k-4316
2. Ahmed	---->	22k-4384

INTRODUCTION

What is NAIKI?

NAIKI is a platform that aims to connect donors with people who are in need of financial assistance for medical bills, education fees, job finding, and loan requirements.

Background:

The idea for NAIKI came from the problems and challenges that people face when trying to find financial assistance for their needs. Many people struggle to find help and support when they need it the most, and this is where NAIKI comes in. We wanted to create a platform that can help alleviate some of the financial burdens that people face and provide them with a way to seek help and support in a transparent and efficient manner.

Objective:

The main objective of NAIKI is to connect donors with those who need financial assistance in a transparent and efficient manner. We want to empower people who are struggling financially by providing them with a platform to seek help and support. We also aim to create a community of donors who are willing to help those in need on a regular basis. NAIKI is not just a platform for financial assistance, but it is also a platform that aims to create a sense of community and support among its users.

METHODOLOGY

We have used the following components:

- **Inheritance**
- **Polymorphism**
- **Abstraction**
- **Encapsulation**
- **Dynamic Memory**
- **Down casting**

FEATURES

1: LOGIN/SIGNUP WITH ALL THE DETAILS REQUIRED

2: THERE WILL BE 3 ROLES WITH IN THE PROJECT

a. ADMIN

b. DONOR

c. NEEDY

3: NEEDY CAN BE A STUDENT/PATIENT/JOBLESS PERSON WHO WILL POST ACCORDING TO HIS NEED

WITH ALL THE DOCUMENTS THAT ARE REQUIRED.

4: DONOR WHO WILL SEE THE VERIFIED DOCUMENTS AND WILL DONATE THE NEEDY PERSON

ACCORDING TO HIS ABILITY OF DONATING.

5: WE CAN ALSO COLLABORATE WITH COMPANIES THAT WILL BE INTERESTED TO ORGANISE

SEMINARS/HACKATHONS FOR STUDENTS AND OUR APP SYSTEM WILL HELP THEM.

6: USERS CAN UPDATE ACCOUNT INFORMATION.

FUNCTIONALITIES CODE:

- **1: Sign Up**

CODE:

```
void signUp()
    {
        std::cout << "\033[2J\033[1;1H";
        std::cout << "     ********** SIGN UP ***********";
        std::cout << "\nEnter first name: ";
        fflush(stdin);
        std::cin >> uD.fName;
        check_string(&uD.fName);
        std::cout << "\nEnter last name: ";
        fflush(stdin);
        std::cin >> uD.lName;
        check_string(&uD.lName);
        while (1)
        {
            int found = 0;
            while (1)
            {
                std::string tempEmail;
                std::cout << "\nEnter email: ";
                fflush(stdin);
                std::cin >> tempEmail;
                uD.email = vemail(tempEmail);
                if (uD.email != "")
                {
                    break;
                }
                else
                {
                    continue;
                }
            }

            std::ifstream infile("users.txt");
            std::string line;
            while (std::getline(infile, line))
            {
```

```
                if (line.substr(0, uD.email.length()) == uD.email)
                {
                    found = 1;
                    std::cout << "This email already exists. Please try again
with a different email." << std::endl;
                    break;
                }
            }

            if (found == 0)
            {
                break;
            }
        }
        std::cout << "\nEnter phone no: ";
        fflush(stdin);
        std::cin >> uD.phoneNo;
        checkNumber(&uD.phoneNo);
        phoneNoValidation(&uD.phoneNo);
        std::string dateInput;
        std::cout << "\nEnter date of birth: \n";
        std::cout << "day: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        uD.dOB.day = stoi(dateInput);
        std::cout << "\nmonth: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        uD.dOB.month = stoi(dateInput);
        std::cout << "\nyear: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        uD.dOB.year = stoi(dateInput);
        validateDate(uD.dOB);
        std::cout << "\nEnter your role(N/n for needy, D/n for donor --> other
keywords are not accepted! )";
        while (1)
        {
            fflush(stdin);
```

```
            std::cin >> uD.role;
            if (uD.role == "n" || uD.role == "N" || uD.role == "d" || uD.role ==
"D")
            {
                break;
            }
            else
            {
                std::cout << "\nWrong role! \nEnter your role again: ";
                continue;
            }
        }
        while (1)
        {
            std::string tempPass;
            std::cout << "\nEnter password: ";
            fflush(stdin);
            std::cin >> tempPass;
            uD.password = passwordFormatCheck(tempPass);
            if (uD.password != "")
            {
                break;
            }
            else
            {
                continue;
            }
        }

        std::cout << "\nConfirm password: ";
        std::string confirmPassword;
        while (1)
        {
            fflush(stdin);
            std::cin >> confirmPassword;
            if (uD.password == confirmPassword)
            {
                srand(time(NULL));
                uD.uId = 10 * rand();
                std::string filePath = std::to_string(uD.uId) + ".txt";
                std::ofstream Write(filePath, std::ios::binary);
                if (Write.is_open())
```

```
            {
                Write << uD.uId << "\n"
                      << uD.fName << "\n"
                      << uD.lName << "\n"
                      << uD.email << "\n"
                      << uD.password << "\n"
                      << uD.phoneNo << "\n"
                      << uD.role << "\n"
                      << uD.dOB.day << "\n"
                      << uD.dOB.month << "\n"
                      << uD.dOB.year << "\n"
                      << false << "\n";
                Write.close();
            }
            FILE *filePtr = fopen("noOfUsers.txt", "r");
            int noOfUsers;
            fscanf(filePtr, "%d", &noOfUsers, sizeof(int));
            fclose(filePtr);
            filePtr = fopen("noOfUsers.txt", "w");
            fprintf(filePtr, "%d", noOfUsers + 1);
            fclose(filePtr);
            std::ofstream outfile("users.txt", std::ios::app);
            outfile << uD.email << " " << filePath << std::endl;
            outfile.close();
            std::cout << "Registration successful!" << std::endl;
            break;
        }
        else
        {
            std::cout << "\nPlease enter correct password!Enter again: ";
        }
    }
};
```

2: Log In:

CODE:

```
void logIn()
    {
        std::ifstream Read("currentlyLogin.txt");
```

```
        if (Read.is_open())
        {
            system("cls");
            std::string filePath;
            Read >> filePath;
            Read.close();
            Read.open(filePath);
            userData u;
            verification v;
            Read >> u.uId >> u.fName >> u.lName >> u.email >> u.password >>
u.phoneNo >> u.role >> u.dOB.day >> u.dOB.month >> u.dOB.year >> v.verify >>
v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo >> u.amountWithdrawOrDonate
>> u.amountInAccount;
            Read.close();
            identifyingRole(u, v);
        }
        else
        {
            system("cls");
            std::cout << "\n     ********LOGIN  ********";
            std::cout << "\nGo to main Screen: \n1: Yes \n2: No \nEnter your
choice: ";
            int backChoice;
            while (1)
            {
                fflush(stdin);
                std::cin >> backChoice;
                if (backChoice == 1)
                {
                    return;
                }
                else if (backChoice == 2)
                {
                    break;
                }
                else
                {
                    std::cout << "\nInvalid! Enter again: ";
                }
            }
            std::string filePath;
            std::string email, password;
```

```
            std::cout << "\nEnter your email: ";
            fflush(stdin);
            std::cin >> email;
            std::ifstream infile("users.txt");
            std::string line;
            bool userFound = false;
            while (getline(infile, line))
            {
                if (line.substr(0, email.length()) == email)
                {
                    userFound = true;
                    filePath = line.substr(email.length() + 1);
                    break;
                }
            }
            infile.close();
            if (!userFound)
            {
                std::cout << "This account does not exist. Please try again with
a valid a account email" << std::endl;
                return;
            }
            std::cout << "\nEnter your password: ";
            fflush(stdin);
            std::cin >> password;
            FILE *filePtr = fopen(filePath.c_str(), "rb");
            std::ifstream Read(filePath, std::ios::binary);
            if (!Read.is_open())
            {
                std::cout << "\nSome Error Occured!";
            }
            else
            {

                userData u;
                verification v;
                Read >> u.uId >> u.fName >> u.lName >> u.email >> u.password >>
u.phoneNo >> u.role >> u.dOB.day >> u.dOB.month >> u.dOB.year >> v.verify;
                if (!Read.eof())
                {
                    Read >> v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo
>> u.amountWithdrawOrDonate;
```

```
                    if (!Read.eof())
                    {
                        Read >> u.amountInAccount;
                    }
                }
                Read.close();
                if (u.password == password)
                {
                    std::cout << "\nSuccessfully login!";
                    std::ofstream Write("currentlyLogin.txt");
                    std::string filePath = std::to_string(u.uId) + ".txt";
                    Write << filePath;
                    Write.close();
                    std::cout << "\nPress any key to continue....";
                    getch();
                    identifyingRole(u, v);
                }
                else
                {
                    std::cout << "\nWrong Password!";
                }
            }
        }
        logIn();
    }
```

3: Profile Verification:

CODE:

DONOR:

```
virtual void profile()
    {
        User::profile();
        int logOutChoice;
        logOut(uD.uId, &logOutChoice);
        if (logOutChoice == 1)
        {
            return;
        }
```

```
        std::cout << "\nSelect any one of the following donor account types: \n1: Company \n2: Individual";
        std::cout << "\nC/c for company and I/i for individual\nEnter your choice: ";
        while (1)
        {
            fflush(stdin);
            std::cin >> v.accountType;
            if (v.accountType == "I" || v.accountType == "i")
            {
                std::cout << "\nEnter cnic Number: ";
                fflush(stdin);
                std::cin >> v.cnicORcompanyNo;
                verifyCnic(&v.cnicORcompanyNo);
                break;
            }
            else if (v.accountType == "C" || v.accountType == "c")
            {
                std::cout << "Enter company license Number:";
                fflush(stdin);
                std::cin >> v.cnicORcompanyNo;
                verifyLicense(&v.cnicORcompanyNo);
                break;
            }
            else
            {
                std::cout << "\nInvalid! Enter again: ";
                continue;
            }
        }
        std::cout << "\nEnter bank account number: ";
        fflush(stdin);
        std::cin >> v.bankAccountNo;
        verifyBankAccount(&v.bankAccountNo);
        v.verify = true;
        writingData(uD, v);
        std::cout << "\nPress any key to continue....";
        getch();
    }
```

NEEDY:

```
  virtual void profile()
    {
        User::profile();
        int logOutChoice;
        logOut(uD.uId, &logOutChoice);
        if (logOutChoice == 1)
        {
            return;
        }
        std::cout << "\nSelect any one of the following donor account types:
\n1:Student \n2:Loanee\n3:Unemployeed \n4: Patient";
        std::cout << "\ns/S for student, L/l for loanee, U/u for unemployeed and
P/p for patient";
        std::cout << "\nEnter your choice: ";
        while (1)
        {
            fflush(stdin);
            std::cin >> v.accountType;
            if (v.accountType == "s" || v.accountType == "S" || v.accountType ==
"L" || v.accountType == "l" || v.accountType == "U" || v.accountType == "u" ||
                v.accountType == "P" || v.accountType == "p")
            {
                break;
            }
            else
            {
                std::cout << "\nWrong keyword\nEnter again: ";
            }
        }

        std::cout << "\nEnter cnic Number: ";
        fflush(stdin);
        std::cin >> v.cnicORcompanyNo;
        checkNumber(&v.cnicORcompanyNo);
        verifyCnic(&v.cnicORcompanyNo);
        std::cout << "\nEnter bank account number: ";
        fflush(stdin);
        std::cin >> v.bankAccountNo;
        checkNumber(&v.bankAccountNo);
        verifyBankAccount(&v.bankAccountNo);
        v.verify = true;
        writingData(uD, v);
```

```
        std::cout << "\nPress any key to continue......";
        getch();
    }
```

Edit Profile:

CODE:

```
  void editProfile(userData &u, verification &v)
    {
        std::cout << "\nWhat do you want to edit \n1: first name \n2: last name
\n3: email \n4: password \n5: phone number"
                  << "\n6: date of birth \n7: bank account number\n8: Exit";
        int choice;
        std::cout << "\nEnter your choice: ";
        while (1)
        {
            fflush(stdin);
            std::cin >> choice;
            if (choice >= 1 && choice <= 8)
            {
                break;
            }
            else
            {
                std::cout << "\nInvalid Choice! Enter again: ";
                continue;
            }
        }
        if (choice == 1)
        {
            std::cout << "\nEnter new first name: ";
            fflush(stdin);
            std::cin >> u.fName;
            check_string(&u.fName);
        }
        else if (choice == 2)
        {
            std::cout << "\nEnter new last name: ";
            fflush(stdin);
            std::cin >> u.lName;
```

```
            check_string(&u.lName);
        }
        else if (choice == 3)
        {
            std::ifstream Read("users.txt");
            std::ofstream Write("temp.txt");
            std::string line;
            while (std::getline(Read, line))
            {
                std::string fileEmail = line.substr(0, u.email.length());
                if (fileEmail == u.email)
                {
                    std::cout << "\nEnter new email: ";
                    fflush(stdin);
                    std::cin >> u.email;
                    line = u.email + " " + std::to_string(u.uId) + ".txt";
                }
                Write << line << "\n";
            }
            Read.close();
            Write.close();
            std::remove("users.txt");
            std::rename("temp.txt", "users.txt");
        }
        else if (choice == 4)
        {
            std::cout << "\nEnter old password: ";
            std::string oldPassword, newPassword;
            fflush(stdin);
            std::cin >> oldPassword;
            if (oldPassword == u.password)
            {
                std::cout << "\nEnter new password: ";
                while (1)
                {
                    fflush(stdin);
                    std::cin >> newPassword;
                    u.password = passwordFormatCheck(newPassword);
                    if (u.password == "")
                    {
                        std::cout << "\nSorry password format not right! \nEnter
again: ";
```

```
                }
                else
                {
                    std::cout << "\nPassword has been reset!";
                    break;
                }
            }
        }
    }
    else if (choice == 5)
    {
        std::cout << "\nEnter new phone number: ";
        fflush(stdin);
        std::cin >> u.phoneNo;
        checkNumber(&u.phoneNo);
        phoneNoValidation(&u.phoneNo);
    }
    else if (choice == 6)
    {
        std::string dateInput;
        std::cout << "\nEnter new date of birth: ";
        std::cout << "day: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        u.dOB.day = stoi(dateInput);
        std::cout << "\nmonth: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        u.dOB.month = stoi(dateInput);
        std::cout << "\nyear: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        u.dOB.year = stoi(dateInput);
        validateDate(u.dOB);
    }
    else if (choice == 7)
    {
        std::cout << "\nEnter new bank account no: ";
        fflush(stdin);
```

```
            std::cin >> v.bankAccountNo;
            checkNumber(&v.bankAccountNo);
            verifyBankAccount(&v.bankAccountNo);
        }
        else if (choice == 8)
        {
            return;
        }
        else
        {
            std::cout << "\nInvalid Choice! Enter again: ";
        }
        std::string filePath = std::to_string(u.uId) + ".txt";
        std::ofstream Write(filePath);
        Write << u.uId << "\n"
              << u.fName << "\n"
              << u.lName << "\n"
              << u.email << "\n"
              << u.password << "\n"
              << u.phoneNo << "\n"
              << u.role << "\n"
              << u.dOB.day << "\n"
              << u.dOB.month << "\n"
              << u.dOB.year << "\n"
              << v.verify << "\n"
              << v.cnicORcompanyNo << "\n"
              << v.accountType << "\n"
              << v.bankAccountNo;
        if (uD.amountInAccount != -1)
        {
            Write << "\n"
                  << uD.amountInAccount;
        }
        if (uD.amountWithdrawOrDonate != -1)
        {
            Write << "\n"
                  << uD.amountWithdrawOrDonate;
        }
        Write.close();
    }
```

Creating a Post:

CODE:

Student:

```
void createPost()
    {
        std::cout << "\nWhat is your institute: \n1:College \n2:University\nEnter
your choice: ";
        int choice;
        while (1)
        {
            fflush(stdin);
            std::cin >> choice;
            if (choice >= 1 && choice <= 2)
            {
                break;
            }
            else
            {
                std::cout << "\nInvalid Type!\nEnter again: ";
            }
        }
        std::cout << "\nEnter institute name: ";
        fflush(stdin);
        std::cin >> p.instituteName;
        check_string(&p.instituteName);
        std::cout << "\nEnter amount needed: ";
        fflush(stdin);
        std::string getInput;
        std::cin >> getInput;
        checkNumber(&getInput);
        p.amount = stoi(getInput);
        if (choice == 1 || choice == 2)
        {
            std::cout << "\nEnter your obtained marks in matric: ";
            while (1)
            {
                fflush(stdin);
                std::cin >> getInput;
                checkNumber(&getInput);
```

```
                p.m.obtMarks = stoi(getInput);
                if (p.m.obtMarks > 1100 || p.m.obtMarks < 0)
                {
                    std::cout << "\nInvalid Marks !\nEnter it out of 1100 again :
";
                    continue;
                }
                else
                {
                    break;
                }
            }
            p.m.percentage = (p.m.obtMarks / 1100) * 100;
            p.m.grade = (p.m.percentage >= 80 ? "A+" : p.m.percentage >= 70 ? "A"
                                                : p.m.percentage >= 60   ? "B"
                                                : p.m.percentage >= 50   ? "C"
                                                                         :
"D");
        }
        if (choice == 2)
        {
            std::cout << "\nEnter your obtained marks in hsc: ";
            while (1)
            {
                fflush(stdin);
                std::cin >> getInput;
                checkNumber(&getInput);
                p.hsc.obtMarks = stoi(getInput);
                if (p.hsc.obtMarks > 1100 || p.hsc.obtMarks < 0)
                {
                    std::cout << "\nInvalid Marks!Enter it again out of 1100: ";
                    continue;
                }
                else
                {
                    break;
                }
            }
            p.hsc.percentage = (p.hsc.obtMarks / 1100) * 100;
            p.hsc.grade = (p.hsc.percentage >= 80 ? "A+" : p.hsc.percentage >= 70
? "A"
```

```
                                                        : p.hsc.percentage >= 60
? "B"
                                                        : p.hsc.percentage >= 50
? "C"

: "D");
        }
        std::string filePath = std::to_string(uD.uId) + "post.txt";
        std::ofstream Write(filePath, std::ios::app);
        Write << p.instituteName << "\n"
              << p.amount << "\n"
              << p.m.obtMarks << "\n"
              << p.m.percentage << "\n"
              << p.m.grade << "\n";
        if (choice == 2)
        {
            Write << p.hsc.obtMarks << "\n"
                  << p.hsc.percentage << "\n"
                  << p.hsc.grade << "\n";
        }
        std::cout << "\nYour post has been created!!";
        Write.close();

        Write.open("postsStudent.txt", std::ios::app);
        Write << "\n"
              << filePath;
        Write.close();
        filePath = std::to_string(uD.uId) + "answers.txt";
        Write.open(filePath);
        Write << 0;
        Write.close();
    }
```

Patient:

```
void createPost()
    {
        p.purpose = "";
        std::cout << "\nFor which purpose you want loan: \n1:Business \n2:Debt
Consolidation \n3:Personal Emergencies"
```

```
                << "\n4:Medical Expenses \n5:Home Improvements \n6:Any other
purpose";
        std::cout << "\nEnter your choice: ";
        int choice;
        while (1)
        {
            fflush(stdin);
            std::cin >> choice;
            if (choice < 1 || choice > 6)
            {
                std::cout << "\nInvalid Choice!\nEnter again: ";
            }
            else
            {
                break;
            }
        }
        if (choice == 6)
        {
            std::cout << "\nEnter your purpose: ";
            fflush(stdin);
            std::cin >> p.purpose;
            check_string(&p.purpose);
        }
        p.purpose = (choice == 1 ? "Business" : choice == 2 ? "Debt
Consolidation"
                                            : choice == 3   ? "Personal
Emergencies"
                                            : choice == 4   ? "Medical Expenses"
                                                            : "Home
Improvements");
        if (p.purpose != "")
        {
            std::string getInput;
            std::cout << "\nHow much amount you need: ";
            fflush(stdin);
            std::cin >> getInput;
            checkNumber(&getInput);
            p.amount = stoi(getInput);
            std::cout << "\nWhat is your employeement type: \n1: Employeer
\n2:Self-Employeed";
            std::cout << "\nEnter your choice: ";
```

```
            while (1)
            {
                fflush(stdin);
                std::cin >> choice;
                if (choice == 1 || choice == 2)
                {
                    p.employeementType = choice == 1 ? "Employeer" : "Self-
Employeed";
                    break;
                }
                else
                {
                    std::cout << "\nInvalid Choice!\nEnter again:";
                }
            }
            if (choice == 1)
            {
                std::cout << "\nEnter your job title: ";
            }
            else
            {
                std::cout << "\nEnter which type of sel-employeed you are: ";
            }
            fflush(stdin);
            std::cin >> p.jobTitle;
            check_string(&p.jobTitle);
            std::cout
                << "\nWhat is your monthly income: ";
            while (1)
            {
                fflush(stdin);
                std::cin >> getInput;
                checkNumber(&getInput);
                p.monthlyIncome = stoi(getInput);
                if (p.monthlyIncome < 0)
                {
                    std::cout << "\nInvalid Income!\nEnter again: ";
                }
                else
                {
                    break;
                }
```

```
            }
            std::cout << "\nEnter your annual tax:";
            while (1)
            {
                fflush(stdin);
                std::cin >> getInput;
                checkNumber(&getInput);
                p.t.annualTax = stoi(getInput);
                if (p.t.annualTax < 0)
                {
                    std::cout << "\nInvalid Number!\nEnter again: ";
                }
                else
                {
                    break;
                }
            }

            std::cout << "\nDo you have paid your tax( 1 for true, any other
keyword for false): ";
            fflush(stdin);
            std::cin >> p.t.paidStatus;
            if (p.t.paidStatus == true)
            {
                std::string filePath = std::to_string(uD.uId) + "post.txt";
                std::ofstream Write(filePath, std::ios::app);
                Write << p.amount << "\n"
                      << p.purpose << "\n"
                      << p.employeementType << "\n"
                      << p.jobTitle << "\n"
                      << p.monthlyIncome << "\n"
                      << p.t.annualTax << "\n"
                      << p.t.paidStatus << "\n";
                Write.close();
                Write.open("loaneePosts.txt", std::ios::app);
                Write << filePath << "\n";
                Write.close();
            }
            else
            {
                std::cout << "\nSorry you can't create post for loan: ";
                return;
```

```
            }
        }
    }
```

Loanee:

```
void createPost()
    {
        std::string getInput;
        std::cout << "\nHow much amount you need: ";
        fflush(stdin);
        std::cin >> getInput;
        checkNumber(&getInput);
        p.amount = stoi(getInput);
        std::cout << "\nEnter the details of the surgery/operation! ";
        std::cout << "\nEnter hospital name: ";
        fflush(stdin);
        std::cin >> p.hospitalName;
        check_string(&p.hospitalName);
        std::cout << "\nEnter disease name: ";
        fflush(stdin);
        std::cin >> p.disease;
        check_string(&p.disease);
        std::cout << "\nEnter surgery/opeartion name:";
        fflush(stdin);
        std::cin >> p.processName;
        check_string(&p.processName);
        std::cout << "\nEnter your monthly income:";
        fflush(stdin);
        std::cin >> getInput;
        checkNumber(&getInput);
        p.income = stoi(getInput);
        if (p.income > 300000)
        {
            std::cout << "\nSorry you are not eligible for the loan/supportation:
";
            return;
        }
        std::cout << "\nEnter your monthly expenses:";
        fflush(stdin);
        std::cin >> getInput;
```

```
        checkNumber(&getInput);
        p.expenses = stoi(getInput);
        if (p.expenses > p.income)
        {
            std::cout << "\nSorry you are not eligible for the loan/supportation:
";
            return;
        }
        std::string filePath = std::to_string(uD.uId) + "post.txt";
        std::ofstream Write(filePath, std::ios::app);
        Write << p.hospitalName << "\n"
              << p.amount << "\n"
              << p.disease << "\n"
              << p.processName << "\n"
              << p.income << "\n"
              << p.expenses << "\n";
        Write.close();
        Write.open("patientPosts.txt", std::ios::app);
        Write << filePath << "\n";
        Write.close();
    }
```

Test Taken:

```
void takeATest()
    {
        std::ifstream Questions("questions.txt");
        std::string answers[5];
        int counter = 0;
        while (!Questions.eof())
        {
            std::string line;
            for (int i = 0; i < 5; i++)
            {
                std::getline(Questions, line);
                if (i == 0)
                {
                    std::cout << "\nQuestion " << counter + 1;
                }
                std::cout << "\n"
                          << line;
            }
```

```
            std::cout << "\nEnter your answer: ";
            fflush(stdin);
            std::cin >> answers[counter];
            checkAnswerChoice(&answers[counter]);
            counter++;
        }
        Questions.close();
        std::ifstream Answers("answers.txt");
        int correctAnswers = 0;
        counter = 0;
        while (!Answers.eof())
        {
            std::string readInput;
            std::getline(Answers, readInput);
            if (readInput == answers[counter])
            {
                correctAnswers++;
            }
        }
        Answers.close();
        std::string filePath = std::to_string(uD.uId) + "answers.txt";
        std::ofstream Write(filePath);
        Write << correctAnswers;
        Write.close();
    }
```

Donate Action:

```
void donateAction()
    {
        std::cout << "\nWant to See home section: \n1: yes \n2: no";
        std::cout << "\nEnter your choice: ";
        int choice;
        while (1)
        {
            fflush(stdin);
            std::cin >> choice;
            if (choice == 1 || choice == 2)
            {
                break;
            }
```

```
            else
            {
                std::cout << "\nInvalid!\nEnter again: ";
                continue;
            }
        }
        if (choice == 1)
        {
            int postChoice;
            std::cout << "\nWhich needy person posts you want to see: ";
            std::cout << "\n1: Students\n2: Loanee\n3:Patients ";
            std::cout << "\nEnter your choice: ";
            std::string stringInput;
            while (1)
            {
                fflush(stdin);
                std::cin >> stringInput;
                checkNumber(&stringInput);
                postChoice = stoi(stringInput);
                if (postChoice >= 1 && postChoice <= 3)
                {
                    break;
                }
                else
                {
                    std::cout << "\nInvalid! Enter again: ";
                    continue;
                }
            }
            std::cout << "\nIn how much amount range you want to see posts: ";
            std::cout << "\nEnter starting range: ";
            int startingRange;
            fflush(stdin);
            std::cin >> stringInput;
            checkNumber(&stringInput);
            startingRange = stoi(stringInput);
            std::cout << "\nEnter end range: ";
            float endRange;
            fflush(stdin);
            std::cin >> stringInput;
            checkNumber(&stringInput);
            endRange = stoi(stringInput);
```

```
            homeFunc(postChoice, startingRange, endRange);
            std::cout << "\nWhich post number profile you want to donate: ";
            int postNo;
            fflush(stdin);
            std::cin >> stringInput;
            checkNumber(&stringInput);
            postNo = stoi(stringInput);
            float amount = 0;
            std::string userProfilePath;
            std::string postFilePath;
            seeAProfile(postNo, postChoice, &amount, &userProfilePath,
&postFilePath);
            if (amount == 0)
            {
                return;
            }
            char donateChoice;
            std::cout << "\nDonate Now(Y/y for yes, N/n for no) ";
            std::cout << "--> (Other keywords are not accepted!): ";
            while (1)
            {
                fflush(stdin);
                std::cin >> donateChoice;
                if (donateChoice == 'y' || donateChoice == 'Y' || donateChoice ==
'N' || donateChoice == 'n')
                {
                    break;
                }
                else
                {
                    std::cout << "\nInvalid keyword!\nEnter again: ";
                    continue;
                }
            }
            if (donateChoice == 'Y' || donateChoice == 'y')
            {
                donateNow(amount);
                std::ifstream Read(userProfilePath);
                userData u;
                verification v;
                Read >> u.uId >> u.fName >> u.lName >> u.email >> u.password >>
u.phoneNo >> u.role >>
```

```
                    u.dOB.day >> u.dOB.month >> u.dOB.year >> v.verify >>
                    v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo >>
u.amountWithdrawOrDonate;
                float newAmount;
                Read >> newAmount;
                Read.close();
                newAmount += amount;
                std::ofstream Write(userProfilePath);
                Write << u.uId << "\n"
                      << u.fName << "\n"
                      << u.lName << "\n"
                      << u.email << "\n"
                      << u.password << "\n"
                      << u.phoneNo << "\n"
                      << u.role << "\n"
                      << u.dOB.day << "\n"
                      << u.dOB.month << "\n"
                      << u.dOB.year << "\n"
                      << v.verify << "\n"
                      << v.cnicORcompanyNo << "\n"
                      << v.accountType << "\n"
                      << v.bankAccountNo << "\n"
                      << u.amountWithdrawOrDonate << "\n"
                      << newAmount << "\n";
                Write.close();
                Read.open(std::to_string(uD.uId) + ".txt");
                Read >> u.uId >> u.fName >> u.lName >> u.email >> u.password >>
u.phoneNo >> u.role >>
                    u.dOB.day >> u.dOB.month >> u.dOB.year >> v.verify >>
                    v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo >>
u.amountWithdrawOrDonate;
                Read.close();
                Write.open(std::to_string(uD.uId) + ".txt");
                Write << u.uId << "\n"
                      << u.fName << "\n"
                      << u.lName << "\n"
                      << u.email << "\n"
                      << u.password << "\n"
                      << u.phoneNo << "\n"
                      << u.role << "\n"
                      << u.dOB.day << "\n"
                      << u.dOB.month << "\n"
```

```
                << u.dOB.year << "\n"
                << v.verify << "\n"
                << v.cnicORcompanyNo << "\n"
                << v.accountType << "\n"
                << v.bankAccountNo << "\n"
                << u.amountWithdrawOrDonate + amount << "\n";
        Write.close();
        int result = std::remove(postFilePath.c_str());
        if (postChoice == 1)
        {
            std::ifstream Read("postsStudent.txt");
            Write.open("temp.txt");
            while (!Read.eof())
            {
                std::string line;
                std::getline(Read, line);
                if (line != postFilePath)
                {
                    Write << line << "\n";
                }
            }
            Write.close();
            Read.close();
            std::remove("postsStudent.txt");
            std::rename("temp.txt", "postsStudent.txt");
        }
        else if (postChoice == 2)
        {
            std::ifstream Read("postsLoanee.txt");
            Write.open("temp.txt");
            while (!Read.eof())
            {
                std::string line;
                std::getline(Read, line);
                if (line != postFilePath)
                {
                    Write << line << "\n";
                }
            }
            Write.close();
            Read.close();
            std::remove("postsLoanee.txt");
```

```
                std::rename("temp.txt", "postsLoanee.txt");
            }
            else if (postChoice == 3)
            {
                std::ifstream Read("patientPosts.txt");
                Write.open("temp.txt");
                while (!Read.eof())
                {
                    std::string line;
                    std::getline(Read, line);
                    if (line != postFilePath)
                    {
                        Write << line << "\n";
                    }
                }
                Write.close();
                Read.close();
                std::remove("patientPosts.txt");
                std::rename("temp.txt", "patientPosts.txt");
            }
        }
    }
}
```

```
void seeAProfile(int postNo, int postChoice, float *amount, std::string
*userProfilePath,
                 std::string *userPostFilePath)
{
    int lineNo = 1;
    std::string filePath, postFilePath;
    if (postChoice == 1)
    {
        std::ifstream Read("postsStudent.txt");
        if (Read.is_open())
        {
            while (!Read.eof())
            {
                std::string line;
                std::getline(Read, line);
                if (lineNo == postNo)
                {
                    postFilePath = line;
```

```
                        *userPostFilePath = postFilePath;
                        int counter = 0;
                        while (line[counter] != 'p')
                        {
                            counter++;
                        }
                        filePath = line.substr(0, counter);
                        getch();

                        filePath = filePath + ".txt";
                        *userProfilePath = filePath;
                        break;
                    }
                    lineNo++;
                }
            }
            else
            {
                std::cout << "\nInvalid post no!";
                std::cout << "\nPress any key to continue...";
                getch();
                return;
            }
            Read.close();
            userData u;
            verification v;
            Read.open(filePath);
            if (Read.is_open())
            {
                Read >> u.uId >> u.fName >> u.lName >> u.email >>
                    u.password >> u.phoneNo >> u.role >> u.dOB.day >>
                    u.dOB.month >> u.dOB.year >> v.verify >>
                    v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo;
                Read.close();
                std::cout << "\nUser Profile";
                std::cout << "\nName: " << u.fName << " " << u.lName;
                std::cout << "\nPhone No: " << u.phoneNo;
                std::cout << "\nDate Of Birth: " << u.dOB.day << "/" <<
u.dOB.month << "/" << u.dOB.year;
                std::cout << "\nCnic No: " << v.cnicORcompanyNo;
                std::cout << "\nProfile Posts: ";
                Read.open(postFilePath);
```

```
                StudentPost sP;
                Read >> sP.instituteName >> sP.amount >> sP.m.obtMarks >>
sP.m.percentage >> sP.m.grade >> sP.hsc.obtMarks >> sP.hsc.percentage >>
sP.hsc.grade;
                Read.close();
                *amount = sP.amount;
                showingStudentPost(sP);
            }
        }
        else if (postChoice == 2)
        {
            std::ifstream Read("loaneePosts.txt");
            if (Read.is_open())
            {
                while (!Read.eof())
                {
                    std::string line;
                    std::getline(Read, line);
                    if (lineNo == postNo)
                    {
                        postFilePath = line;
                        *userPostFilePath = postFilePath;
                        int counter = 0;
                        while (line[counter] != 'p')
                        {
                            counter++;
                        }
                        filePath = line.substr(0, counter);
                        getch();

                        filePath = filePath + ".txt";
                        *userProfilePath = filePath;
                        break;
                    }
                    lineNo++;
                }
            }
            else
            {
                std::cout << "\nInvalid post no!";
                return;
            }
```

```
            Read.close();
            userData u;
            verification v;
            Read.open(filePath);
            if (Read.is_open())
            {
                Read >> u.uId >> u.fName >> u.lName >> u.email >>
                    u.password >> u.phoneNo >> u.role >> u.dOB.day >>
                    u.dOB.month >> u.dOB.year >> v.verify >>
                    v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo;
                Read.close();
                std::cout << "\nUser Profile";
                std::cout << "\nName: " << u.fName << " " << u.lName;
                std::cout << "\nPhone No: " << u.phoneNo;
                std::cout << "\nDate Of Birth: " << u.dOB.day << "/" <<
u.dOB.month << "/" << u.dOB.year;
                std::cout << "\nCnic No: " << v.cnicORcompanyNo;
                std::cout << "\nProfile Posts: ";
                Read.open(postFilePath);
                loaneePost lP;
                Read >> lP.amount >> lP.purpose >> lP.employeementType >>
                    lP.jobTitle >> lP.monthlyIncome >> lP.t.annualTax >>
lP.t.paidStatus;
                Read.close();
                *amount = lP.amount;
                showingLoaneePost(lP);
            }
        }
        else if (postChoice == 3)
        {
            std::ifstream Read("patientPosts.txt");
            if (Read.is_open())
            {
                while (!Read.eof())
                {
                    std::string line;
                    std::getline(Read, line);
                    if (lineNo == postNo)
                    {
                        postFilePath = line;
                        *userPostFilePath = postFilePath;
                        int counter = 0;
```

```
                    while (line[counter] != 'p')
                    {
                        counter++;
                    }
                    filePath = line.substr(0, counter);
                    getch();

                    filePath = filePath + ".txt";
                    *userProfilePath = filePath;
                    break;
                }
                lineNo++;
            }
        }
        else
        {
            std::cout << "\nInvalid post no!";
            return;
        }
        Read.close();
        userData u;
        verification v;
        Read.open(filePath);
        if (Read.is_open())
        {
            Read >> u.uId >> u.fName >> u.lName >> u.email >>
                u.password >> u.phoneNo >> u.role >> u.dOB.day >>
                u.dOB.month >> u.dOB.year >> v.verify >>
                v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo;
            Read.close();
            std::cout << "\nUser Profile";
            std::cout << "\nName: " << u.fName << " " << u.lName;
            std::cout << "\nPhone No: " << u.phoneNo;
            std::cout << "\nDate Of Birth: " << u.dOB.day << "/" <<
u.dOB.month << "/" << u.dOB.year;
            std::cout << "\nCnic No: " << v.cnicORcompanyNo;
            std::cout << "\nProfile Posts: ";
            Read.open(postFilePath);
            patientPost p;
            Read >> p.hospitalName >> p.amount >> p.disease >> p.processName
>>
                p.income >> p.expenses;
```

```
                Read.close();
                *amount = p.amount;
                showingPatientPost(p);
            }
        }
    }
```

Donate Now:

```
void donateNow(float amount)
    {
        "\nEnter your credit card Details: ";
        cardDetails.cardNo = creditCardValid();
        std::cout << "\nEnter name on card: ";
        fflush(stdin);
        std::cin >> cardDetails.nameOnCard;
        check_string(&cardDetails.nameOnCard);
        std::string stringInput;
        std::cout << "\nEnter card cvv: ";
        fflush(stdin);
        std::cin >> stringInput;
        checkNumber(&stringInput);
        cardDetails.cvv = stoi(stringInput);
        std::string dateInput;
        std::cout << "\nExpire Data: ";
        std::cout << "\nmonth: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        cardDetails.expireDate.expiry_month = stoi(dateInput);
        std::cout << "\nyear: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        cardDetails.expireDate.expiry_year = stoi(dateInput);
        creditCardDateValid(cardDetails.expireDate);
        std::cout << "\nPerfect! " << amount << " Rs has been deducted from your
card!";
        std::cout << "\nPress any key to continue......";
        getch();
        return;
```

```
    }
```

Withdraw:

```
void withDrawAction(userData &u, verification &v)
    {
        int withDrawChoice;
        std::cout << "\nDo you want to withdraw amount: \n1: yes \n2: no \nEnter
your choice: ";
        while (1)
        {
            fflush(stdin);
            std::cin >> withDrawChoice;
            if (withDrawChoice == 1)
            {
                amountwithDraw(u, v);
                break;
            }
            else if (withDrawChoice == 2)
            {
                break;
            }
            else
            {
                std::cout << "\nInvalid! Enter again: ";
            }
        }
    }
```

```
void amountwithDraw(userData &u, verification &v)
    {
        std::string filePath = std::to_string(u.uId) + ".txt";
        std::ifstream Read(filePath);
        std::cout << "\nHow much amount you want to withdraw (Maximum limit is
100000): ";
        std::string getInput;
        int amount;
        while (1)
        {
            fflush(stdin);
            std::cin >> getInput;
```

```
            checkNumber(&getInput);
            amount = stoi(getInput);
            if (amount > 1 && 100000)
            {
                break;
            }
            else
            {
                std::cout << "\nInvalid !Try again: ";
            }
        }
        Read >> u.uId >> u.fName >> u.lName >> u.email >> u.password >> u.phoneNo
>> u.role >> u.dOB.day >> u.dOB.month >> u.dOB.year >> v.verify >>
            v.cnicORcompanyNo >> v.accountType >> v.bankAccountNo >>
u.amountWithdrawOrDonate >> u.amountInAccount;
        Read.close();
        if (amount > u.amountInAccount)
        {
            std::cout << "\nInsufficient Balance!";
            return;
        }
        std::ofstream Write(filePath);
        Write << u.uId << "\n"
              << u.fName << "\n"
              << u.lName << "\n"
              << u.email << "\n"
              << u.password << "\n"
              << u.phoneNo << "\n"
              << u.role << "\n"
              << u.dOB.day << "\n"
              << u.dOB.month << "\n"
              << u.dOB.year << "\n"
              << v.verify << "\n"
              << v.cnicORcompanyNo << "\n"
              << v.accountType << "\n"
              << v.bankAccountNo << "\n"
              << u.amountWithdrawOrDonate + amount << "\n"
              << u.amountInAccount - amount << "\n";
        Write.close();
        std::cout << "Amount has been withdrawed!";
        std::cout << "\nPress any key to continue ";
        getch();
```

```
    }
```

Hackathon Creation:

```
void createHackathon()
    {
        std::string inputString;
        std::cout << "\nHackathon name: ";
        fflush(stdin);
        std::cin >> hack.name;
        check_string(&hack.name);
        std::cout << "\nNo Of Participants: ";
        fflush(stdin);
        std::cin >> inputString;
        checkNumber(&inputString);
        hack.noOfParticipants = stoi(inputString);
        std::cout << "\nTheme: ";
        fflush(stdin);
        std::cin >> hack.theme;
        check_string(&hack.theme);
        std::cout << "\nVenue: ";
        fflush(stdin);
        std::cin >> hack.venue;
        check_string(&hack.venue);
        std::cout << "\nDate: ";
        std::string dateInput;
        std::cout << "day: ";
        fflush(stdin);
        std::cin >> dateInput;
        checkNumber(&dateInput);
        hack.date.day = stoi(dateInput);
        while (1)
        {
            if (hack.date.day >=1 && hack.date.day < 32)
            {
                break;
            }
            else
            {
                std::cout <<"\nInvalid day! Enter again: " ;
                std::cin >> dateInput;
```

```
            checkNumber(&dateInput);
            hack.date.day = stoi(dateInput);
            continue ;
        }
    }
    std::cout << "\nmonth: ";
    fflush(stdin);
    std::cin >> dateInput;
    checkNumber(&dateInput);
    hack.date.month = stoi(dateInput);
    while (1)
    {
        if (hack.date.month >= 5  && hack.date.month < 13)
        {
            break;
        }
        else
        {
            std::cout <<"\nInvalid Month! Enter again: " ;
            std::cin >> dateInput;
            checkNumber(&dateInput);
            hack.date.month = stoi(dateInput);
            continue ;
        }
    }
    std::cout << "\nyear: ";
    fflush(stdin);
    std::cin >> dateInput;
    checkNumber(&dateInput);
    hack.date.year = stoi(dateInput);
    while (1)
    {
        if (hack.date.year == 2023)
        {
            break;
        }
        else
        {
            std::cout <<"\nInvalid Year! Enter again: " ;
            std::cin >> dateInput;
            checkNumber(&dateInput);
            hack.date.year = stoi(dateInput);
```

```
            continue ;
        }
    }
    std::cout << "\nPrize Money: ";
    fflush(stdin);
    std::cin >> hack.prize;
    checkNumber(&inputString);
    hack.prize = stoi(inputString);
    std::cout << "\nEnter judges: ";
    for (int i = 0; i < 3; i++)
    {
        std::cout << "\njudge " << i + 1 << " name: ";
        fflush(stdin);
        std::cin >> hack.judges[i];
        check_string(&hack.judges[i]);
    }
    std::cout << "\nEnter any 5 rules and regulations: ";
    for (int i = 0; i < 5; i++)
    {
        std::cout << "\nRule " << i + 1 << ":";
        fflush(stdin);
        std::cin >> hack.rules[i];
        check_string(&hack.rules[i]);
    }
    std::string filePath = std::to_string(this->uD.uId) + "hack.txt";
    std::ofstream Write(filePath);
    Write << hack.name << "\n"
          << hack.noOfParticipants << "\n"
          << hack.theme << "\n"
          << hack.prize << "\n"
          << hack.venue << "\n"
          << hack.date.day << "\n"
          << hack.date.month << "\n"
          << hack.date.year << "\n";
    for (int i = 0; i < 3; i++)
    {
        Write << hack.judges[i] << "\n";
    }
    for (int i = 0; i < 5; i++)
    {
        Write << hack.rules[i] << "\n";
    }
```

```
        Write.close();
        Write.open("hackathonspost.txt", std::ios::app);
        Write << filePath << "\n";
        Write.close();
        std::cout << "\nYour hackathon post has been created!";
        std::cout << "\nPress any key to coninue...";
        getch();
    }
```

Hackathon Apply:

```
void hackathonList()
    {
        std::ifstream Write("hackathonspost.txt");
        if (Write.is_open())
        {
            int noOfPosts = 0;
            while (!Write.eof())
            {

                std::string filePath;
                std::getline(Write, filePath);
                std::ifstream Post(filePath);
                hackathon hack;
                if (Post.is_open())
                {
                    Post >> hack.name >> hack.noOfParticipants >>
                        hack.theme >> hack.prize >>
                        hack.venue >> hack.date.day >> hack.date.month >>
hack.date.year;
                    for (int i = 0; i < 3; i++)
                    {
                        Post >> hack.judges[i];
                    }
                    for (int i = 0; i < 5; i++)
                    {
                        Post >> hack.rules[i];
                    }
                    Post.close();
                    std::cout << "\nHackathon Post " << noOfPosts + 1;
                    std::cout
```

```
                        << "\nName: " << hack.name << "\nNo Of Participants
Allowed: " << hack.noOfParticipants
                        << "\nTheme: " << hack.theme << "\nPrize: " << hack.prize
<< "\nVenue: " << hack.venue;
                    std::cout << "\nDate: " << hack.date.day << "/" <<
hack.date.month << "/" << hack.date.year;
                    std::cout << "\nJudges: ";
                    for (int i = 0; i < 3; i++)
                    {
                        std::cout << "\nJudge " << i + 1 << ":" <<
hack.judges[i];
                    }
                    std::cout << "\nRules: ";
                    for (int i = 0; i < 5; i++)
                    {
                        std::cout << "\nRule " << i + 1 << ":" << hack.rules[i];
                    }
                }
                noOfPosts++;
            }
        }
    }
```

```
void hackathonApply(int postNo)
    {
        int lineNo = 1;
        std::string filePath, originalFilePath, hackathonFilePath;
        std::ifstream Read("hackathonspost.txt");
        if (Read.is_open())
        {
            while (!Read.eof())
            {
                std::string line;
                std::getline(Read, line);
                if (lineNo == postNo)
                {
                    int counter = 0;
                    while (line[counter] != 'h')
                    {
                        counter++;
                    }
                    filePath = line.substr(0, counter);
```

```
                originalFilePath = filePath + ".txt";
                hackathonFilePath = line;
                filePath = filePath + "participants.txt";
                break;
            }
            lineNo++;
        }
    }
    else
    {
        std::cout << "\nWrong post no: ";
    }
    Read.close();
    std::string participantFilePath;
    participantFilePath = std::to_string(uD.uId) + ".txt";
    int noOfParticipants = 0;
    Read.open(filePath);
    if (Read.is_open())
    {
        while (!Read.eof())
        {
            noOfParticipants++;
        }
    }
    Read.close();
    Read.open(hackathonFilePath);
    std::string wasteString;
    int fileParticipants;
    Read >> wasteString >> fileParticipants;
    Read.close();
    if (fileParticipants == 1 + noOfParticipants)
    {
        Read.open("hackathonspost.txt");
        std::ofstream Write("temp.txt");
        while (!Read.eof())
        {
            std::string line;
            std::getline(Read, line);
            if (line != hackathonFilePath)
            {
                Write << line;
            }
```

```
            }
            Write.close();
            Read.close();
            std::remove("hackathonspost.txt");
            std::rename("temp.txt", "hackathonspost.txt");
        }
        std::ofstream Write(filePath, std::ios::app);
        Write << participantFilePath << "\n";
        Write.close();
        std::cout << "\nCongratulations! You have applied for hackathon!!";
        std::cout << "\nPress any key to continue....";
        getch();
    }
```

CONCLUSION

In conclusion, NAIKI is a platform that can make a significant difference in the lives of people who are in need of financial assistance. We believe that everyone deserves a chance to live a happy and healthy life, and NAIKI aims to help make this a reality.