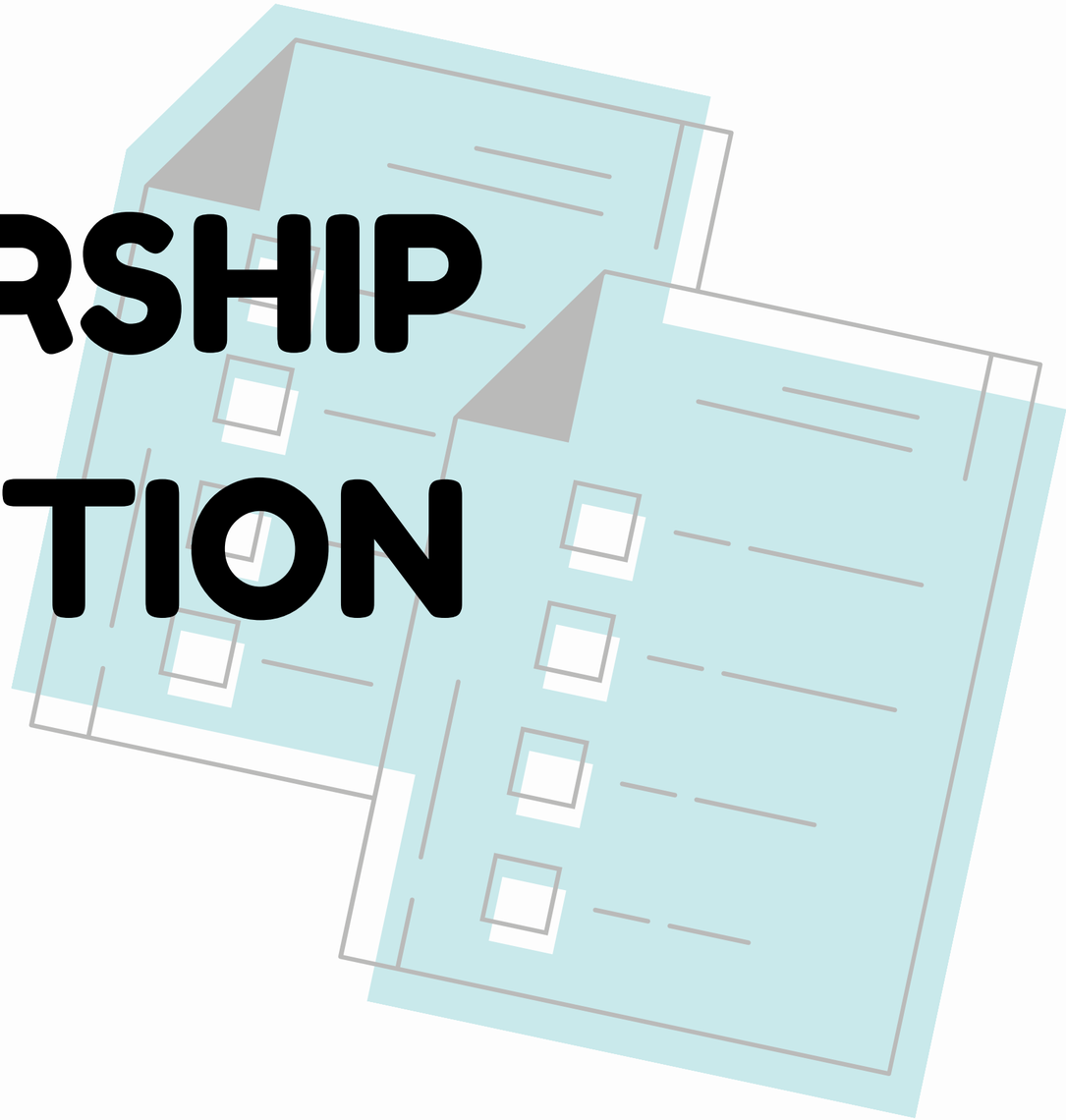
SCHOLARSHIP APPLICATION SYSTEM



Introduction

The scholarship application system typically involves students who need financial help to support their education. It is also used to award the students who get a good result. For merit based scholarship, the selection between the applicants will be chosen based on the academic performance while for the need-based scholarship academic performance and family income will be used to determine whether they qualify for this scholarship or not.

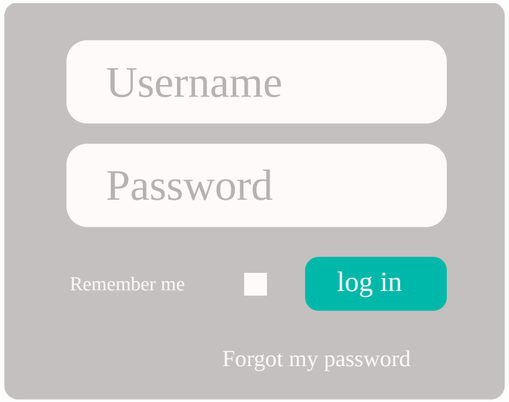
Objective

- To give awards to students who get a good result.
- Assign scholarships that fulfill the requirements based on student’s information
- To encourage students to pursue higher education by providing financial support.

Project Designs

User registration / login

New user need to sign in by using the basic information in the scholarship application system to create an account.



Home page

Include 2 options which are merit based scholarship and need Based scholarship.

Meritbased scholarship

This scholarship will given based on the CGPA.

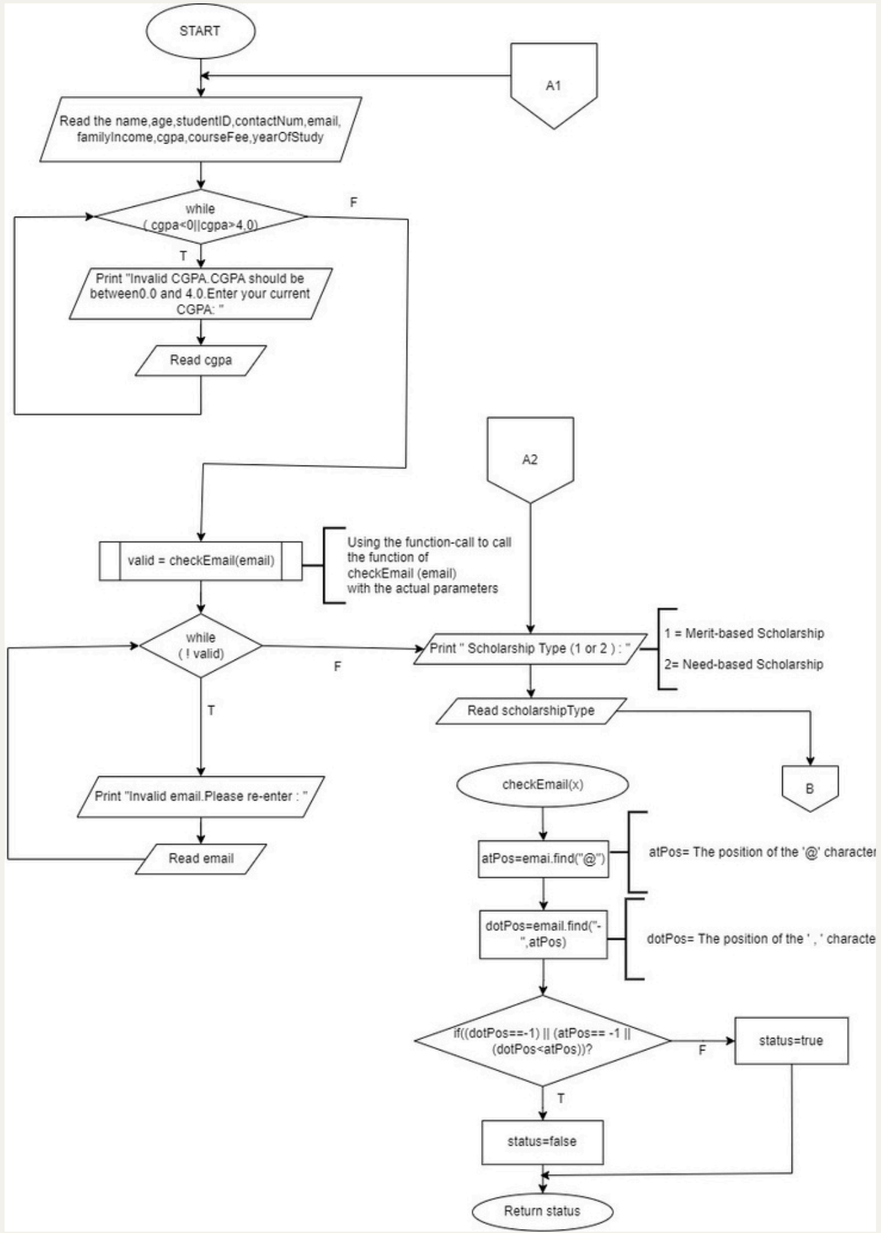
Needbased scholarship

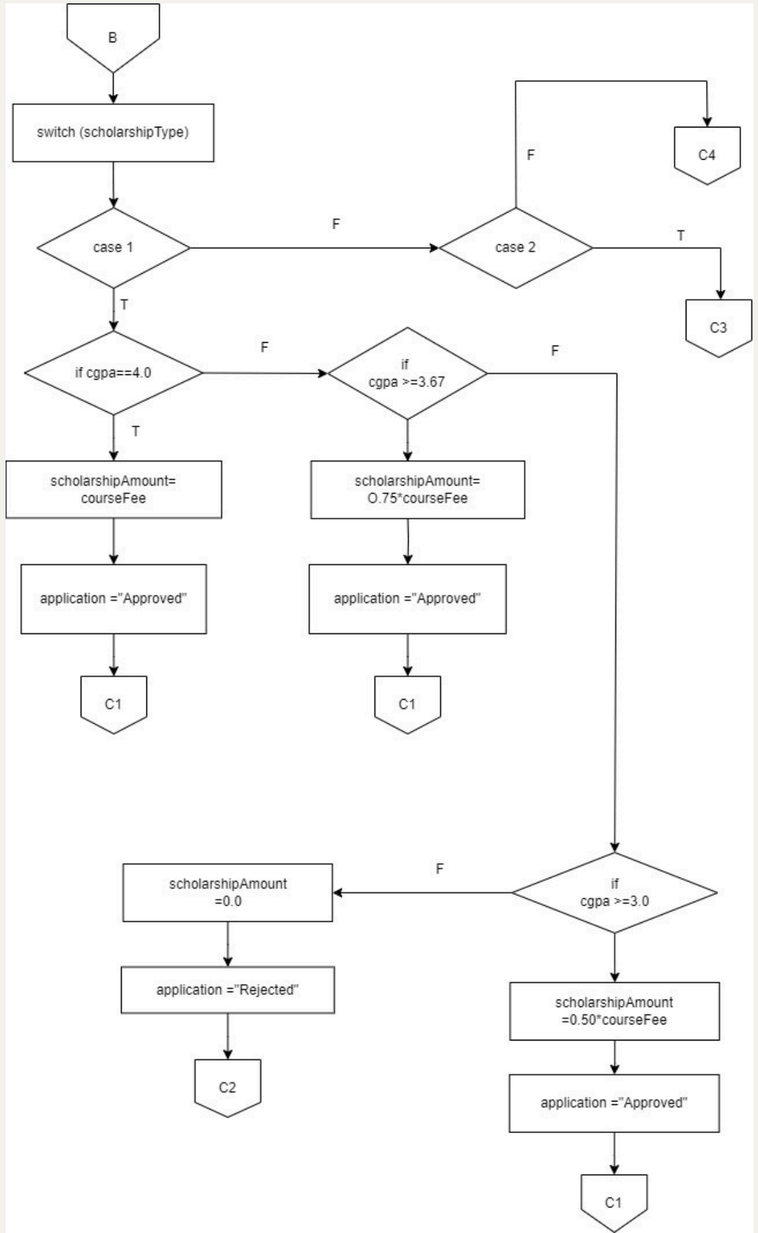
This scholarship will given based on the current CGPA and family income.

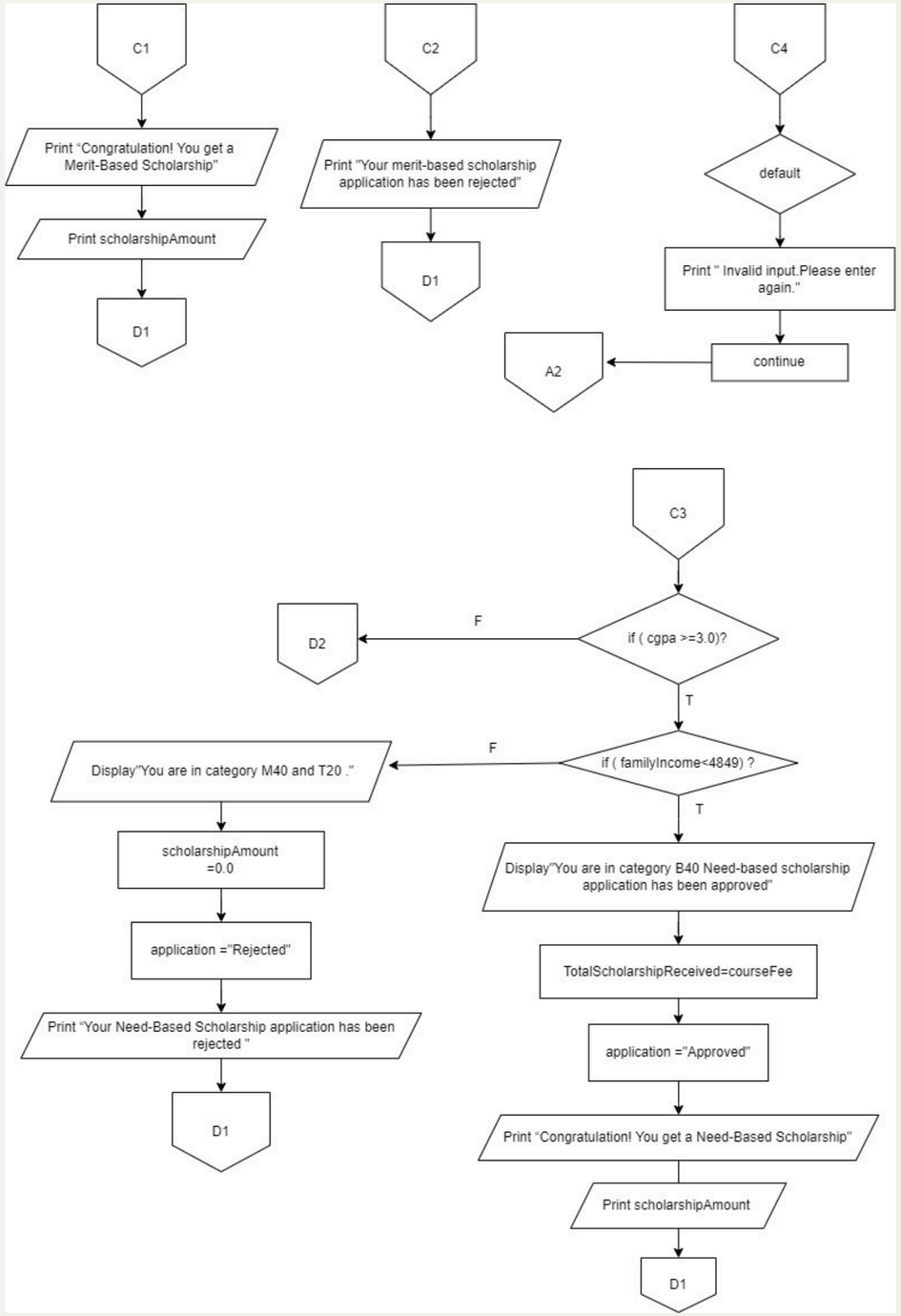
Application review

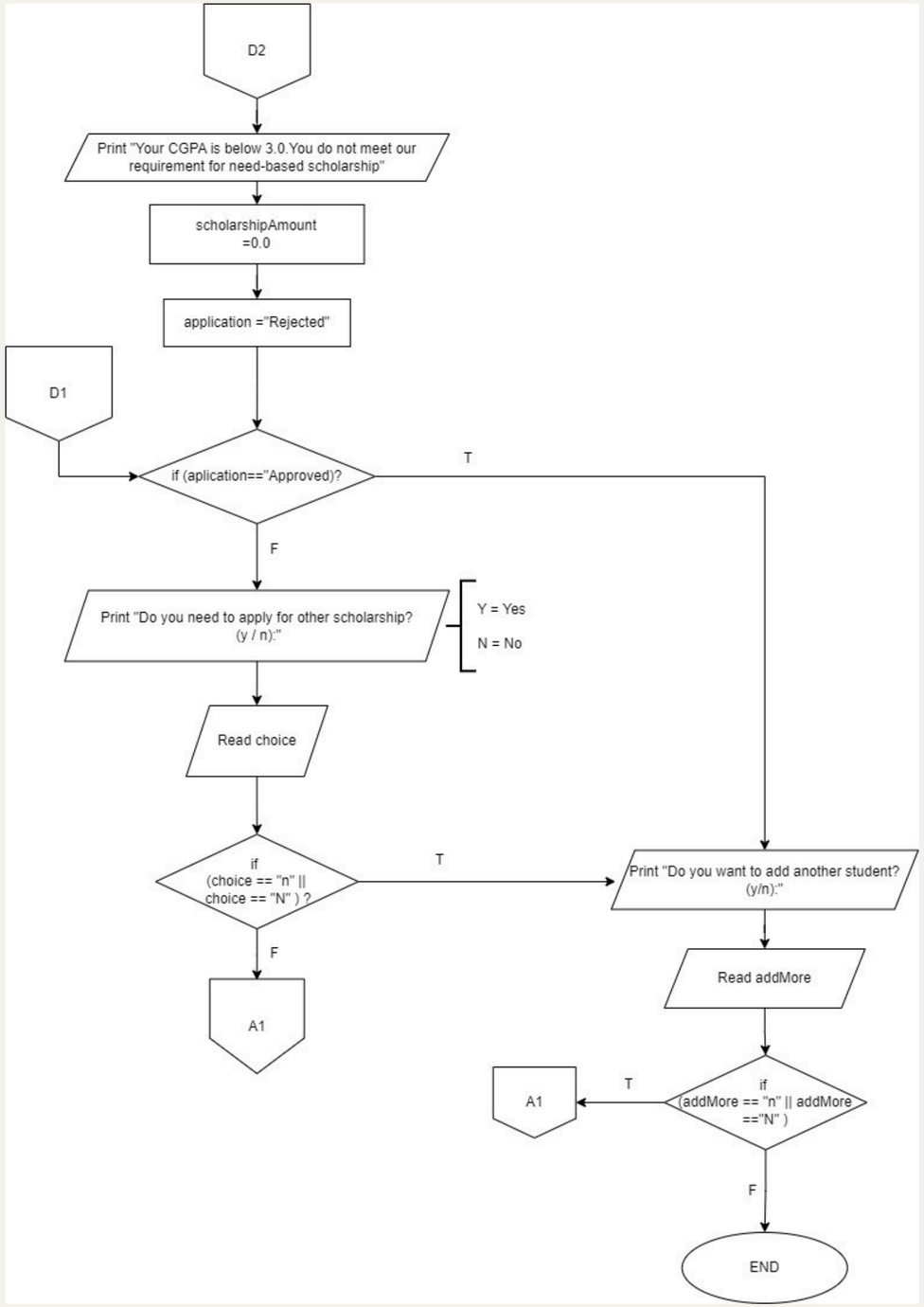
The system will check whether the information key in by the students are accurate.

Flow Chart

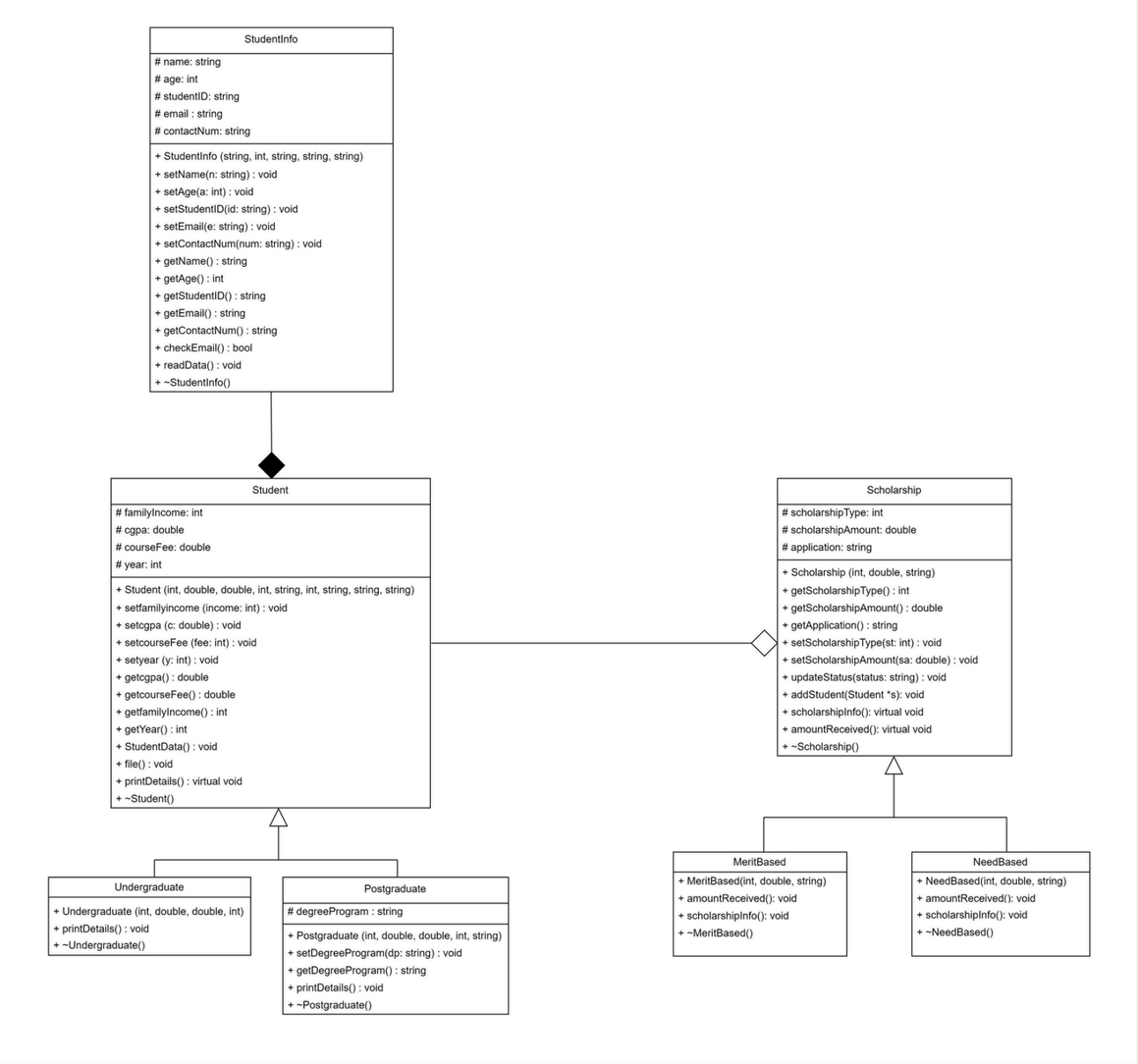

START
A1
Read the name,age,studentID,contactNum,email, familyIncome,cgpa,courseFee,yearOfStudy
while (cgpa<0||cgpa>4.0)
F
T
Print "Invalid CGPA.CGPA should be between0.0 and 4.0.Enter your current CGPA: "
Read cgpa
A2
valid = checkEmail(email)
Using the function-call to call the function of checkEmail (email) with the actual parameters
while (! valid)
F
T
Print " Scholarship Type (1 or 2) : "
1 = Merit-based Scholarship
2= Need-based Scholarship
Read scholarshipType
B
Print "Invalid email.Please re-enter : "
Read email
checkEmail(x)
atPos=emai.find("@")
atPos= The position of the '@' character
dotPos=email.find("-",atPos)
dotPos= The position of the ' , ' characte
if((dotPos==-1) || (atPos== -1 || (dotPos<atPos))?
F
status=true
T
status=false
Return status


B
switch (scholarshipType)
case 1
F
case 2
F
C4
T
C3
T
if cgpa==4.0
F
if cgpa >=3.67
F
T
scholarshipAmount= courseFee
scholarshipAmount= O.75*courseFee
application ="Approved"
application ="Approved"
C1
C1
if cgpa >=3.0
F
scholarshipAmount =0.0
application ="Rejected"
C2
scholarshipAmount =0.50*courseFee
application ="Approved"
C1


C1
Print “Congratulation! You get a Merit-Based Scholarship"
Print scholarshipAmount
D1
C2
Print "Your merit-based scholarship application has been rejected"
D1
C4
default
Print " Invalid input.Please enter again."
continue
A2
C3
if (cgpa >=3.0)?
F
D2
T
if (familyIncome<4849) ?
F
Display"You are in category M40 and T20 ."
scholarshipAmount =0.0
application ="Rejected"
Print “Your Need-Based Scholarship application has been rejected "
D1
T
Display"You are in category B40 Need-based scholarship application has been approved"
TotalScholarshipReceived=courseFee
application ="Approved"
Print “Congratulation! You get a Need-Based Scholarship”
Print scholarshipAmount
D1


D2
Print "Your CGPA is below 3.0.You do not meet our requirement for need-based scholarship"
scholarshipAmount =0.0
application ="Rejected"
D1
if (aplication=="Approved)?
T
F
Print "Do you need to apply for other scholarship? (y / n):"
Y = Yes
N = No
Read choice
if (choice == "n" || choice == "N") ?
T
F
A1
Print "Do you want to add another student? (y/n):"
Read addMore
if (addMore == "n" || addMore =="N")
T
A1
F
END

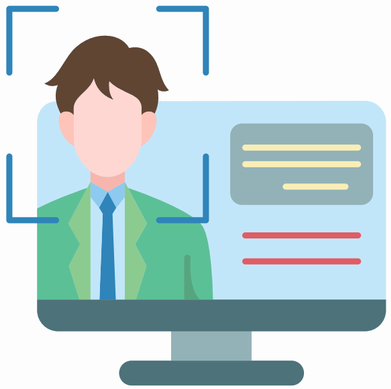
UML Diagram


StudentInfo
name: string
age: int
studentID: string
email : string
contactNum: string
+ StudentInfo (string, int, string, string, string)
+ setName(n: string) : void
+ setAge(a: int) : void
+ setStudentID(id: string) : void
+ setEmail(e: string) : void
+ setContactNum(num: string) : void
+ getName() : string
+ getAge() : int
+ getStudentID() : string
+ getEmail() : string
+ getContactNum() : string
+ checkEmail() : bool
+ readData() : void
+ ~StudentInfo()
Student
familyIncome: int
cgpa: double
courseFee: double
year: int
+ Student (int, double, double, int, string, int, string, string, string)
+ setfamilyincome (income: int) : void
+ setcgpa (c: double) : void
+ setcourseFee (fee: int) : void
+ setyear (y: int) : void
+ getcgpa() : double
+ getcourseFee() : double
+ getfamilyIncome() : int
+ getYear() : int
+ StudentData() : void
+ file() : void
+ printDetails() : virtual void
+ ~Student()
Scholarship
scholarshipType: int
scholarshipAmount: double
application: string
+ Scholarship (int, double, string)
+ getScholarshipType() : int
+ getScholarshipAmount() : double
+ getApplication() : string
+ setScholarshipType(st: int) : void
+ setScholarshipAmount(sa: double) : void
+ updateStatus(status: string) : void
+ addStudent(Student *s): void
+ scholarshipInfo(): virtual void
+ amountReceived(): virtual void
+ ~Scholarship()
Undergraduate
+ Undergraduate (int, double, double, int)
+ printDetails() : void
+ ~Undergraduate()
Postgraduate
degreeProgram : string
+ Postgraduate (int, double, double, int, string)
+ setDegreeProgram(dp: string) : void
+ getDegreeProgram() : string
+ printDetails() : void
+ ~Postgraduate()
MeritBased
+ MeritBased(int, double, string)
+ amountReceived(): void
+ scholarshipInfo(): void
+ ~MeritBased()
NeedBased
+ NeedBased(int, double, string)
+ amountReceived(): void
+ scholarshipInfo(): void
+ ~NeedBased()

Object-Oriented Programming

1. *Encapsulation*

Combining attributes and methods



StudentInfo

Attribute :
Name , Student Id , Email ,
Contact Number

Method:

- StudentInfo (string, int, string, string, string)
- setName (string): void
- setAge (int): void
- setStudentID (string): void
- setEmail (string): void
- setContactNum(string): void
- getName(): string
- getAge(): int
- getStudentID(): string
- getEmail(): string
- getContactNum(): string
- checkEmail(): bool
- readData(): void
- ~StudentInfo ()

2. Association

- Relationships between Student and Scholarship could be done by **aggregation**. This is because a Student can have a Scholarship but Scholarship does not necessarily need to have a Student.

- Relationships between Student and StudentInfo could be done by **composition**. This is because a Student consists of StudentInfo, when a Student object is created, the StudentInfo is also created.

3. *Inheritance*

- **Base Class : Student**
- **Derived Class: Undergraduate, Postgraduate**

Undergraduate students and postgraduate students are a special type of student. They have “is a” relationship with the student as an undergraduate student is a student and the postgraduate student is also a student. By inheriting from students, both undergraduate and postgraduate classes can **reuse common attributes** and **methods** from student class while they can also create additional methods to specify their own class.

- **Base Class : Scholarship**
- **Derived Class: MeritBased, NeedBased**

Need-based and merit-based are both specific types of scholarship. Three of the classes have the same criteria which are related to financial need. Thus, by inheriting, both MeritBased and NeedBased classes can use the same attribute and method of Scholarship class and they can add some method to specify their class. This ensures different types of scholarships are organized under a common hierarchy and ensure that they are easy to manage.

4. Polymorphism

- The Student and Scholarship classes have purs virtual functions. Student class consist of printDetails() method while Scholarship class consist of scholarshipInfo() and amountReceived().
- Declare a parent class pointer and pointing to their child class.
- Dynamic binding and compiler will determine the function of the appropriate class

5. *Array of Object*

- Student Array

Student *stud[MAXSTUDENT] declares an array of pointers to Student objects. Each element of the array can hold the address of a Student object or any of its derived classes.

- Scholarship Array

Scholarship *_scholarship[MAXSTUDENT] declares an array of pointer to Scholarship objects. Each element of the array can hold the address of a Scholarship object or any of its derived classes.

SYSTEM DEMONSTRATION

THANK YOU