FINAL OOP PROJECT

MILESTONE 3(FINAL) AND MILESTONE 2

MUHAMMAD RAFAY ZAFAR

29014

Student managment system

By Muhammad Rafay Zafar(29014)

SETUP REQUIRMENTS:

PLEASE ENSURE YOU HAVE THE “saved data 2.0” FILE SAVED ON YOUR COMPUTER THEN COPY ITS FILE-PATH TO THE “save_students” AND “load_students” FUNCTIONS.

THEN YOU WILL REQUIRE THE FOLLOWING LOGINS:

- ADMIN LOGIN:
  NAME: Adminiba
  PASSWORD: admin@iba
- STUDENT LOGINS:

1. ERP: 29014 , PASSWORD: rafay@iba
2. ERP: 29015 , PASSWORD: ahmed@iba
3. ERP: 29017 , PASSWORD: anis@iba
4. ERP: 29004 , PASSWORD: hussain@iba

INTRODUCTION:

I SET OUT TO BUILD TO BUILD A STUDENT MANAGMENT SYSTEM THAT WILL PROVIDE A STUDENT AND ADMIN LOGIN SYSTEM. THE STUDENT & ADMIN WILL HAVE ACCESS TO THEIR OWN FUNCTIONALITIES AS THEY WOULD IF IT WERE A REAL-LIFE APPLICATION IN A UNIVERSITY. THE STUDENT HAS ACCESS TO ONLY THEIR RECORDS WHILE THE ADMIN CAN ACCESS AND MANIPULATE ALL SAVED STUDENT RECORDS. STUDENTS WILL BE ABLE TO ADD/REMOVE CLASSES FOR THE SEMESTER WHILE THE ADMIN WILL BE ABLE TO MANIPULATE THEIR DEEPER RECORDS LIKE UPDATING THEIR GPA,CHANGING THEIR NAME,ERP,ACCOUNT PASSWORD AND MORE. THE PROJECT AIMS TO REPLICATE AN ACTUAL STUDENT MANAGEMENT SYSTEM FROM A UNIVERSITY BUT MAKE IT VERY SIMPLE TO FOLLOW FOR WHO EVER THE USER IS WHETHER A STUDENT OR THE ADMIN, THEREBY IMPROVING USER SATISFACTION.

OBJECTIVES:

THE MOTIVATION WAS TO WORK SOLO ON A SIMPLE YET COMPREHENSIVE IDEA TO ACTUALLY LEARN PROGRAMMING WHILE HAVING THE FREEDOM TO DEVELOP MY OWN PROJECT IDEA AND TRY TO APPLY ALL CONCEPTS LEARNED IN THE CLASS IN MY OWN WAY. SO THAT I COULD BECOME A BETTER CODER AND LEARN THE THINKING PATTERNS REQUIRED FOR REAL LIFE APLLICATIONS SUCH AS THIS ON A LARGER SCALE.

C++ CONCEPTS AIMED TO BE COVERED IN THE PROJECT:

- INHERITANCE
- POLYMORPHISM
- EXCEPTION HANDLING
- FILE HANDLING
- CIN STATE FAILURE HANDLING
- RANGES
- STL
- FUNCTORS
- VECTORS
- LAMBDA FUNCTIONS
- OPERATOR OVERLOADING
- FRIEND FUNCTIONS
- VIRTUAL FUNCTIONS

SCOPE OF THE PROJECT:

- COURSE MANAGMENT: ADDING AND REMOVING COURSES FOR A STUDENT.
- FILE HANDLING: READING AND WRITING STUDENT OBJECTS TO AND FROM A FILE.
- EDITING RECORDS: EDIT EXISTING STUDENT RECORDS
- RECORD MANAGMENT: ADDING OR REMOVING STUDENT RECORDS.
- RECOMENDATION SYSTEM: RECOMMENDING COURSES ACCORDING TO STUDENTS ACADEMIC BACKGROUND.
- DISPLAY: DISPLAYING INDIVIDUAL RECORDS AND ALL RECORDS IN A SORTED AND PROPERLY FORMATTED MANNER.
- ERROR HANDLING: GRACEFULLY MANAGE WRONG INPUTS FROM USER TO INSURE PROGRAM FLOW.
- EXCEPTION HANDLING: TO ENSURE SPECIFIC FORMATS IN SOME DATA (ERP,PASSWORDS ETC).

SCOPE OF THE PROJECT (EXCLUDED):

- STUDENT RECORDS HAVE BEEN KEPT SIMPLE TO PROVIDE CLARITY TO THE ACTUAL CONCEPTS AT PLAY.
- GUI IS EXCLUDED, CODE RUNS AND TAKES INPUT IN THE TERMINAL.
- STUDENT IN THE ATTATCHED SAVED FILE HAVE BEEN KEPT TO A MINIMUM SO TERMINAL IS NOT CLUTTERED WITH OUTPUTS.

DELIVERABLES:

- C++ SOURCE CODE FOR COMPLETE PROJECT: Final full.cpp
- SAVED STUDENT RECORDS FILE: saved data 2.0.txt
- CLASS DIAGRAM
- FLOWCHART
- TESTCASES
- PROJECT DOCUMENTATION(THIS DOCUMENT)

TIMELINE:

- WEEK 1: PLAN

1. Mind map on how to structure code and classes
2. Submit project idea

- WEEK 2: START CODE

1. Choose and setup IDE
2. List out classes and their functions
3. Make validation functions.
3. Start learning how to do simple file handling

- WEEK 3-6: OOP CONCEPTS

1. Courselist base class implemented
2. Admin and Student classes derived
3. Polymorphism through a Virtual function added.
4. Exception handling for test cases added.
5. Lambda functions used individually and with STL.
6. Ranges added
7. Functors added
8. Operator overloaded through friend functions.

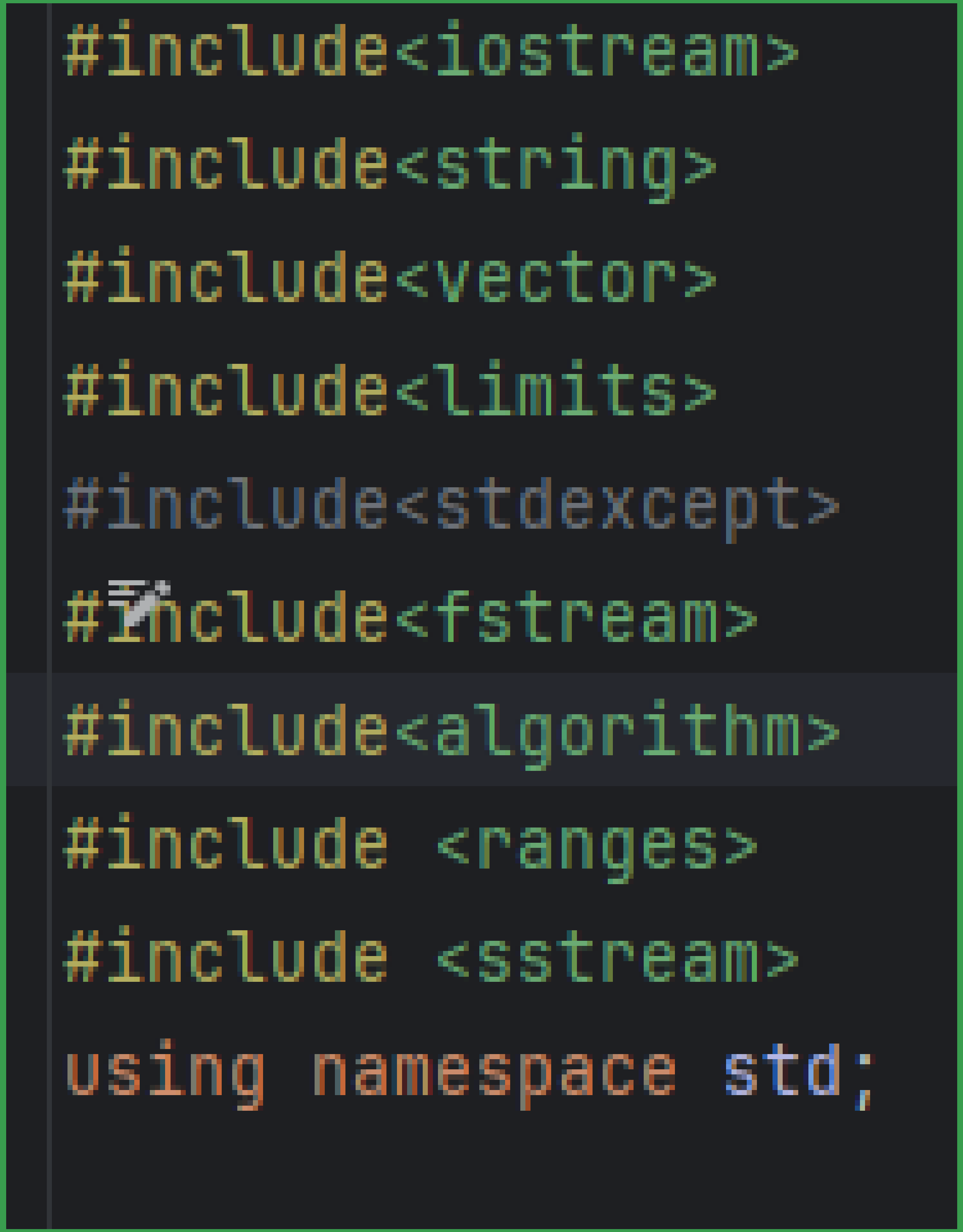
- WEEK 7: BRING IT TOGETHER

1. Implement proper file handling.
2. Address all cin state failures.
3. Remove bugs.
4. Test all features.

- LAST WEEK: FINALIZE

1. Documentation
2. Class diagrams
3. Flowchart
4. Final testing

TOOLS AND LIBRARIES



```cpp
#include<iostream>
#include<string>
#include<vector>
#include<limits>
#include<stdexcept>
#include<fstream>
#include<algorithm>
#include <ranges>
#include <sstream>
using namespace std;
```

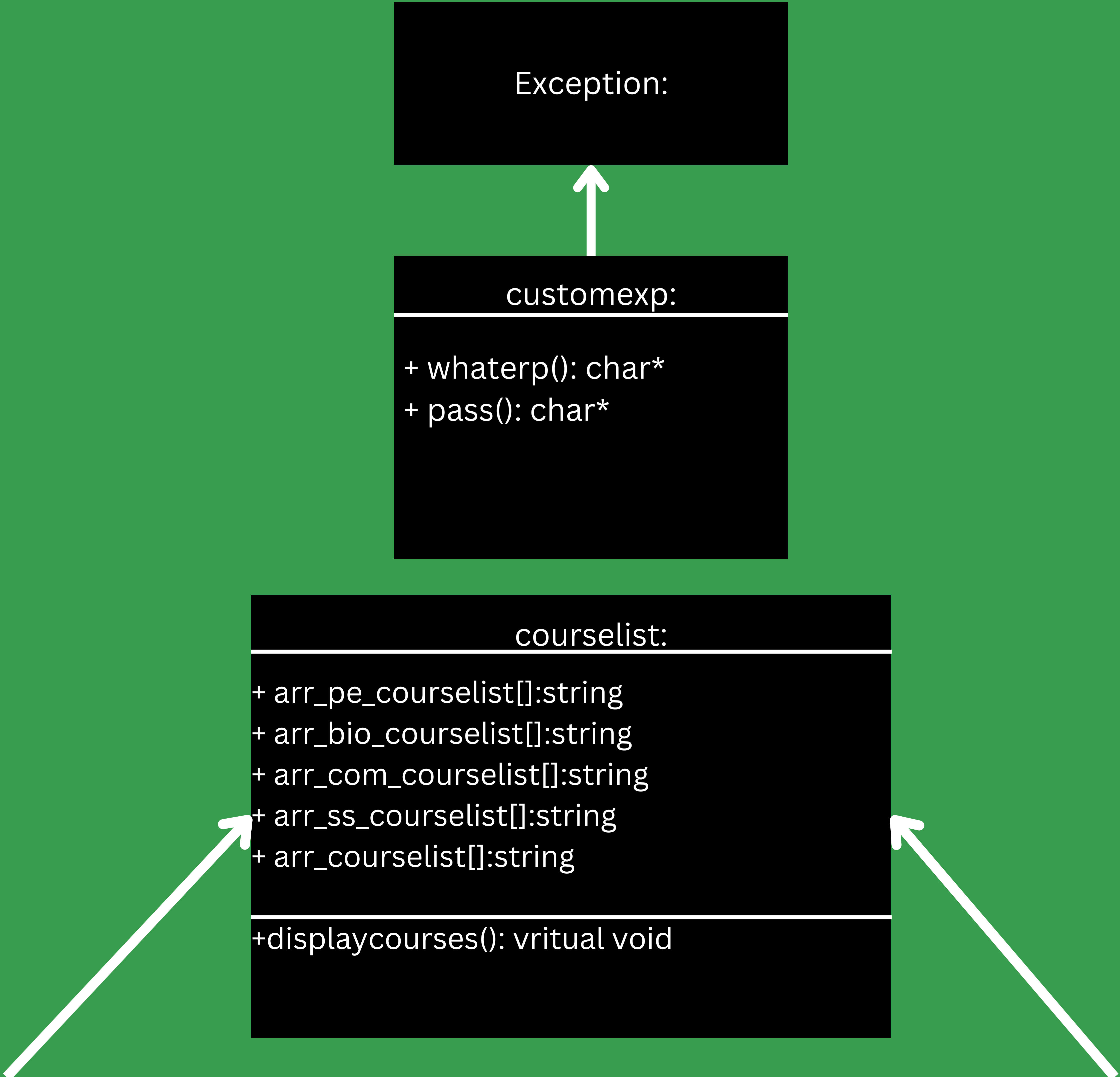
Libraries in use.

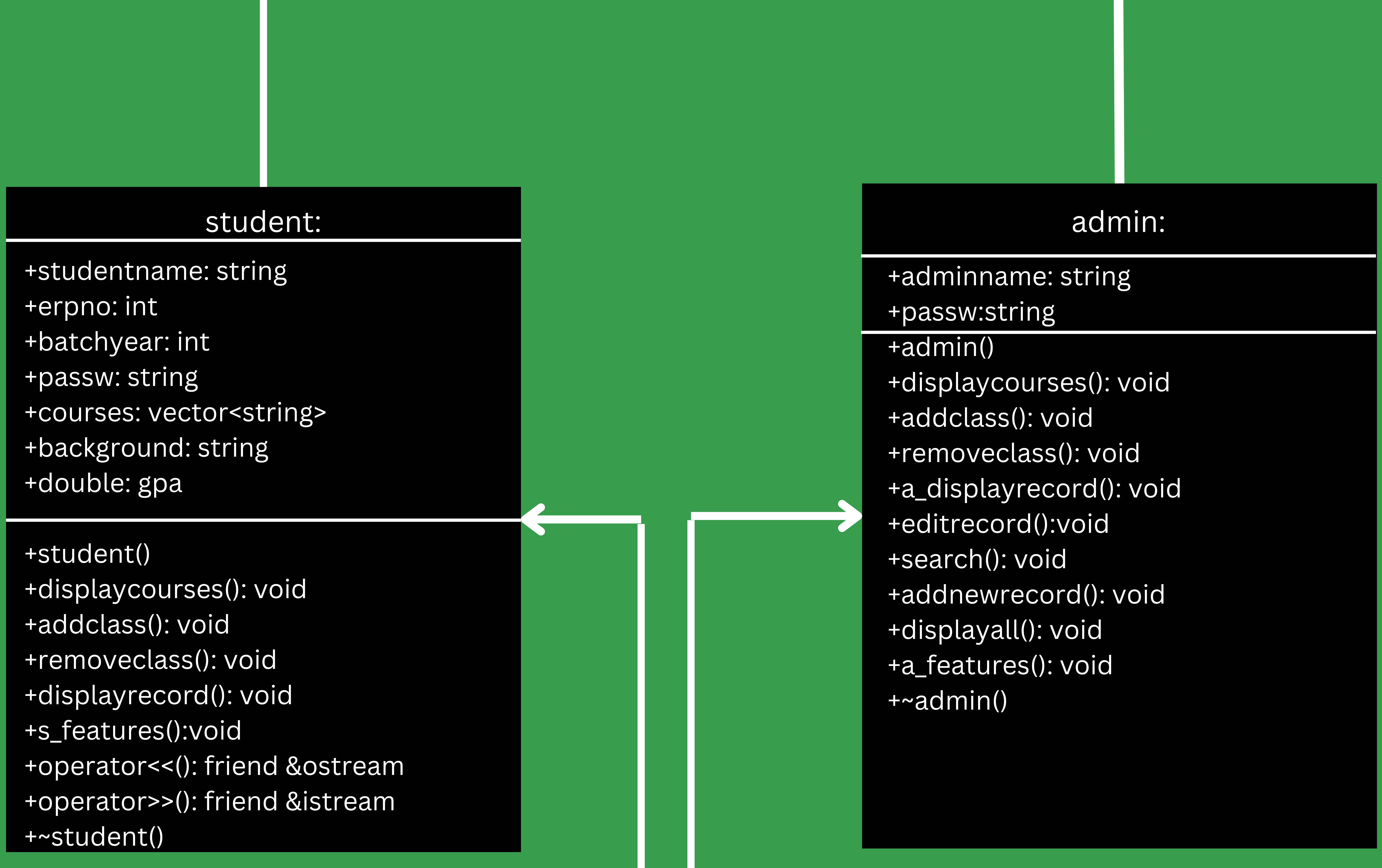
1. **<iostream> is our standard input/output library necesasry for all programs**
2. **<string> used to calculate string sizes mostly using sizeof/size functions.**
3. **<vector> used for the vector container for storage of objects and courses etc.**
4. **<limits> is being used in conjunction with std::numeric_limits<streamsize>::max() to handle invalid input from the standard input stream throughout the code.**
5. **<stdexcept> for custom excpetions.**
6. **<fstream> for object serialization and deserialization.**
7. **<algorithm> to used to call STL functions where required in the code.**
8. **<ranges> used for standard ranges and ::views function.**
9. **<sstream> used for “istringstream ss(line)” to read comma seprated values for student objects from the file.**

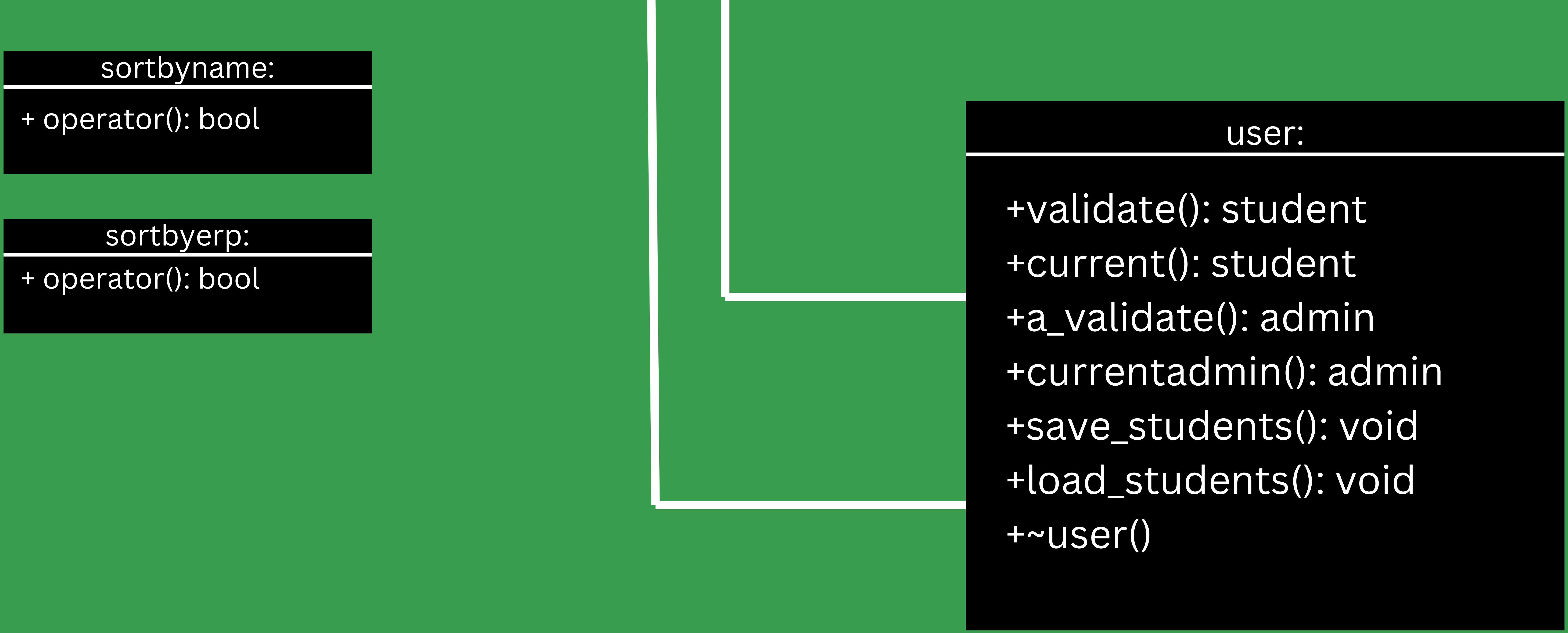
IDE’S USED:

1. **DEV C++**
2. **VS CODE**
3. **CLION**

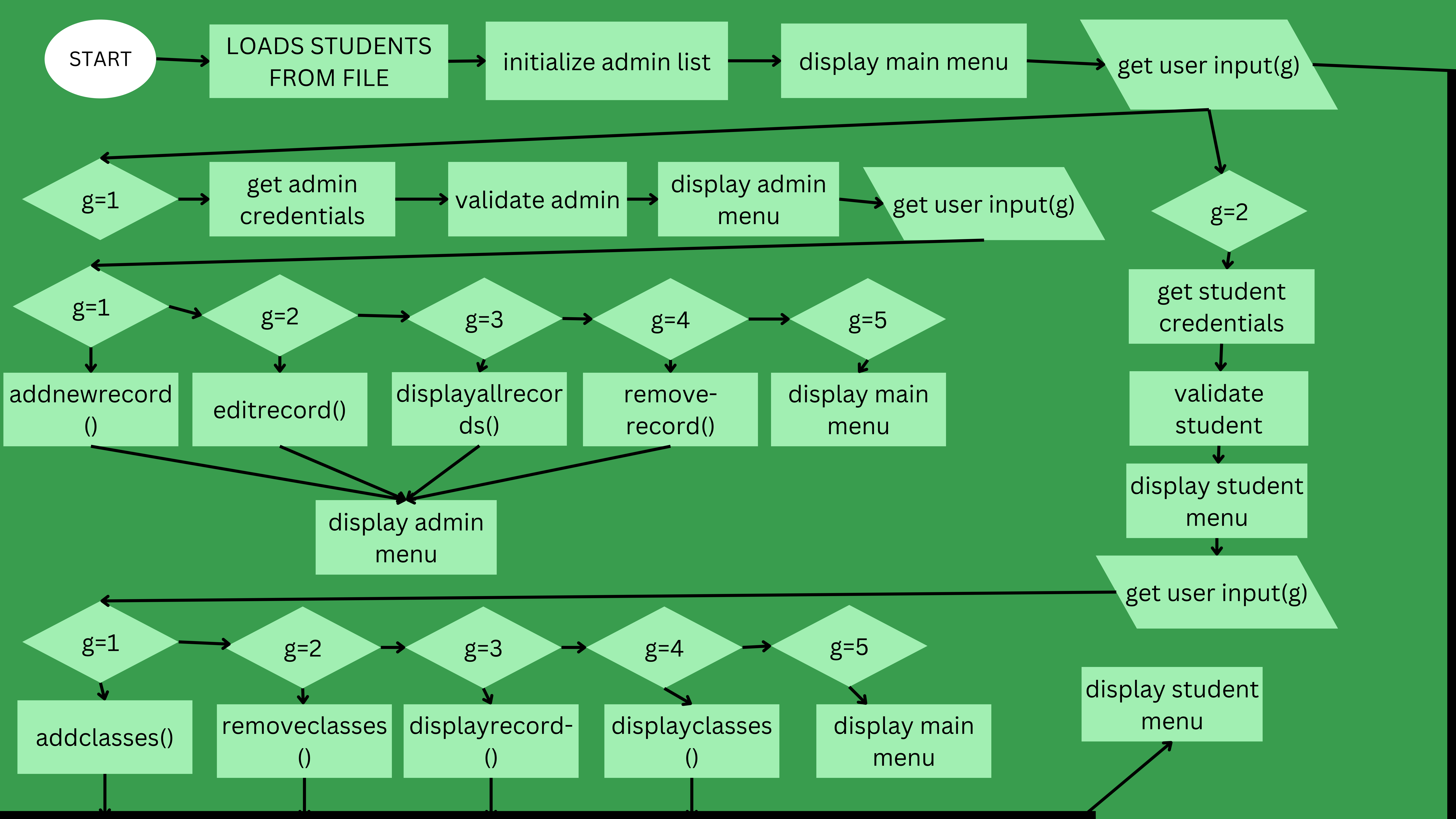
CLASS DIAGRAM

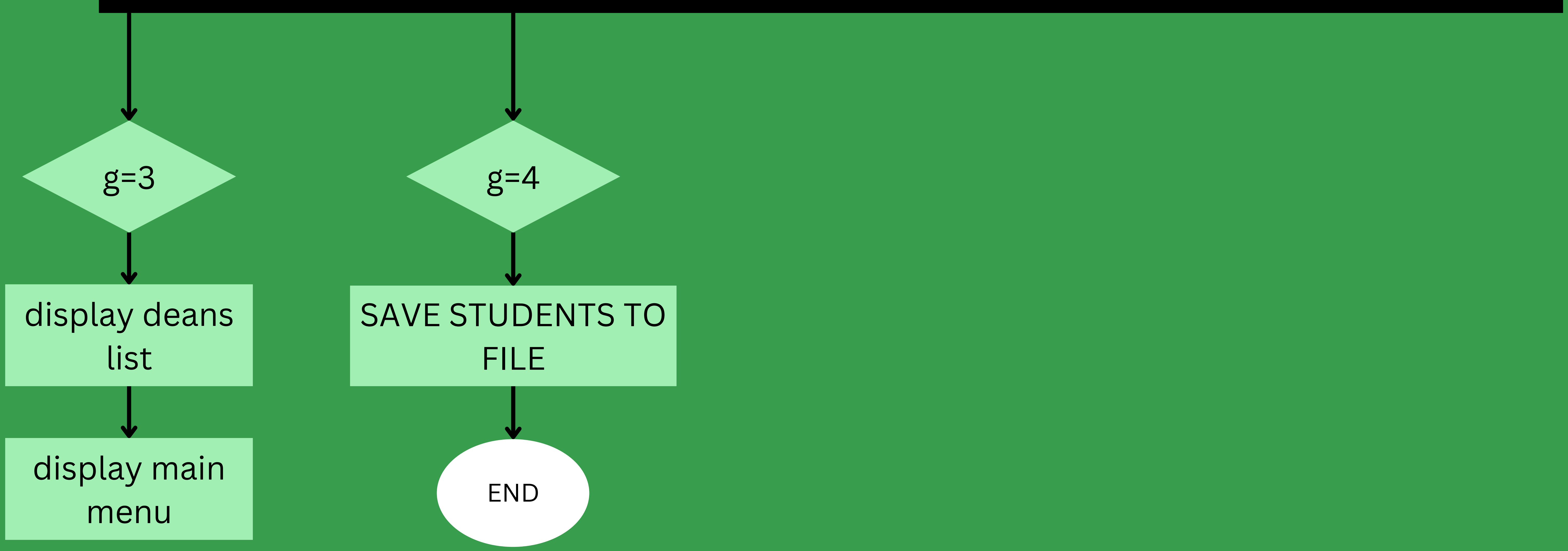

Exception:
customexp:
+ whaterp(): char*
+ pass(): char*
courselist:
+ arr_pe_courselist[]:string
+ arr_bio_courselist[]:string
+ arr_com_courselist[]:string
+ arr_ss_courselist[]:string
+ arr_courselist[]:string
+displaycourses(): vritual void


student:
+studentname: string
+erpno: int
+batchyear: int
+passw: string
+courses: vector<string>
+background: string
+double: gpa
+student()
+displaycourses(): void
+addclass(): void
+removeclass(): void
+displayrecord(): void
+s_features():void
+operator<<(): friend &ostream
+operator>>(): friend &istream
+~student()
admin:
+adminname: string
+passw:string
+admin()
+displaycourses(): void
+addclass(): void
+removeclass(): void
+a_displayrecord(): void
+editrecord():void
+search(): void
+addnewrecord(): void
+displayall(): void
+a_features(): void
+~admin()


sortbyname:
+ operator(): bool
sortbyerp:
+ operator(): bool
user:
+validate(): student
+current(): student
+a_validate(): admin
+currentadmin(): admin
+save_students(): void
+load_students(): void
+~user()

FLOWCHART


START
LOADS STUDENTS FROM FILE
initialize admin list
display main menu
get user input(g)
g=1
get admin credentials
validate admin
display admin menu
get user input(g)
g=2
g=1
g=2
g=3
g=4
g=5
get student credentials
addnewrecord ()
editrecord()
displayallrecords()
remove-record()
display main menu
validate student
display student menu
display admin menu
get user input(g)
g=1
g=2
g=3
g=4
g=5
display student menu
addclasses()
removeclasses ()
displayrecord-()
displayclasses ()
display main menu


g=3
display deans list
display main menu
g=4
SAVE STUDENTS TO FILE
END

TEST CASES

TEST CASES:

- ERP:

Has to be a 5 digit number
Has to be an integer

- BATCH YEAR:

Has to be a b/w 2024 and 2027(inclusive)
Has to be an integer

- PASSWORD:

Has to be more than 4 characters and contain no numbers in it.

- GPA:

Has to be in range 2.2 to 4.0(inclusive)
Has to be a double

- VALIDATION FUNCTIONS:

Correct login credentials have to be entered for students and admin, otherwise user is notified

- CIN STATE FAILURE:

Incorrect type should not result in cin state failure(int i->cin(w)->fail)

- MENU EXPLORATION:

Incorrect menu inputs should be handled[options(1-5)->cin(6)]

- ADD CLASS FUNCTION:

Course for a student will not exceed 6 at a time

- INCORRECT ERPS ENTERED:

In all functions that require searching through erp, it will notify when erp not found

- REMOVE CLASS FUNCTION:

If course vector is empty classes cant be removed, user will be notified

- LOAD AND SAVE FUNCTIONS:

Ensure correct filepath is entered and file is opened properly

- ADMIN FEATURES FUNCTION:

Ensure all menu options work properly
Handle invalid inputs
Ensure inner menus like in editrecord() in option 2 work properly
save edited student object to the main studentlist vector

- STUDENT FEATURES FUNCTION:

Ensure all menu options work properly
Handle invalid inputs
Save edited student object to the main studentlist vector

THANK YOU
MUHAMMAD RAFAY ZAFAR
29014