COLLEGE MANAGEMENT SYSTEM

OBJECT ORIENTED METHODOLOGY PROJECT

Presented By

Prerak Mathur

LIT2020009

Shreya Tarwey

LIT2020059

Tejas Taneja

LIT2020025

Purpose of our project

The main purpose of our project is to build a college database management system. The database consists of basic student information like their name, year of study in the present organization, date of birth, date of joining, enrollment number, grades and some optional details like their phone number, blood group and address.

Home

As soon as we compile our program, we are presented with a set of task that the user would like to operate

- To sort and filter the databse.
- To enter the Admin Portal.
- To clear the screen.
- To exit from the program.

Accessing the Admin Portal

The admin portal is protected since it involves tampering with sensitive information of students.

Admin's Accessibility

Logout and return to home menu

Select another student

Add grade to current student

Add new student

Change details of current student

Remove a student

Functionality of the available options

Selecting a student

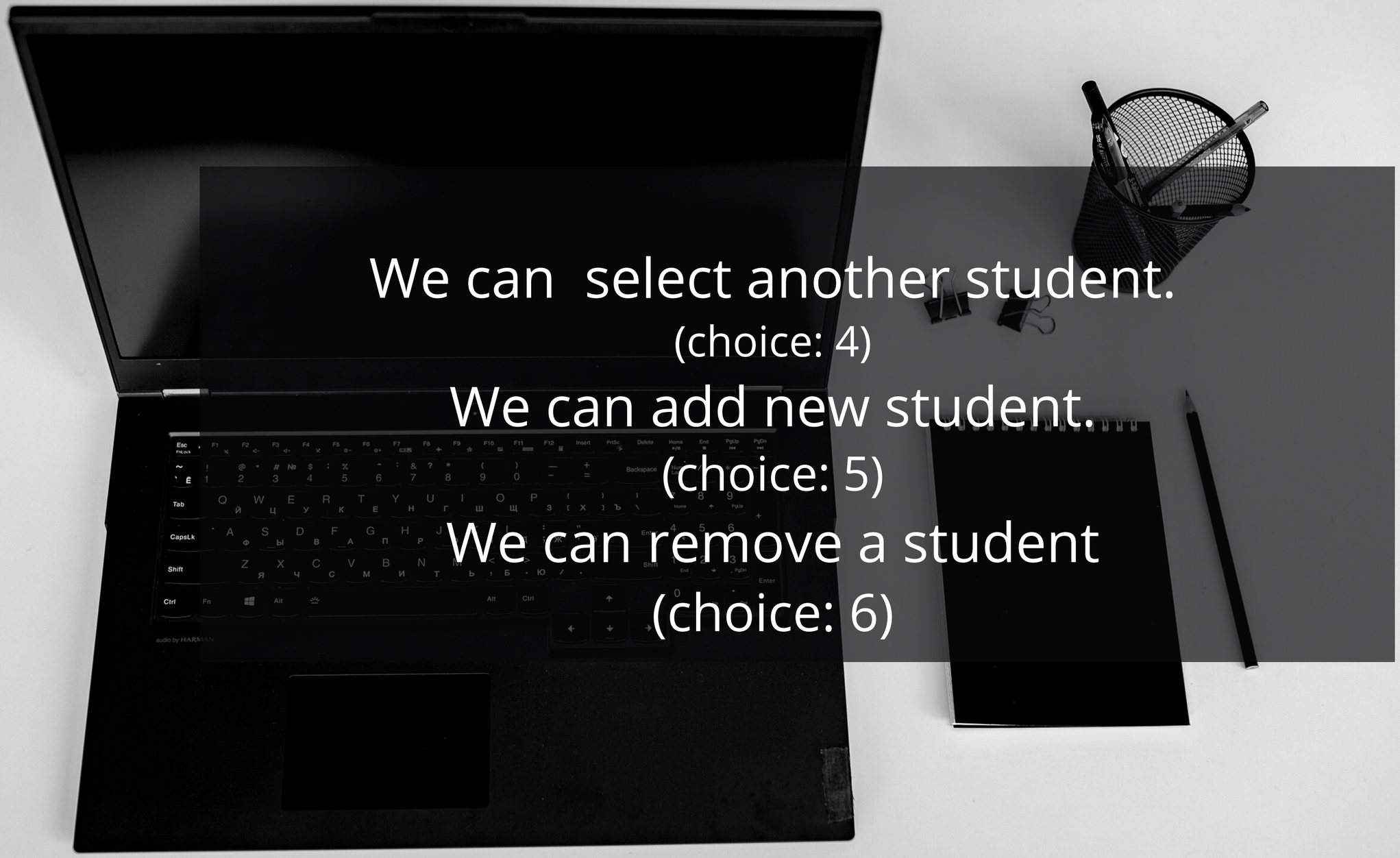
(Choosing Option 4)

We are required to enter the "First" and "Last" name of our student in order to perform various operations on them!

Now we have a current student to work with , we can perform all other options on our current student

- Add grade to current student
- Change details of current student
- Display information of the student

PERFORMING THE VARIOUS OPERATIONS



We can select another student.

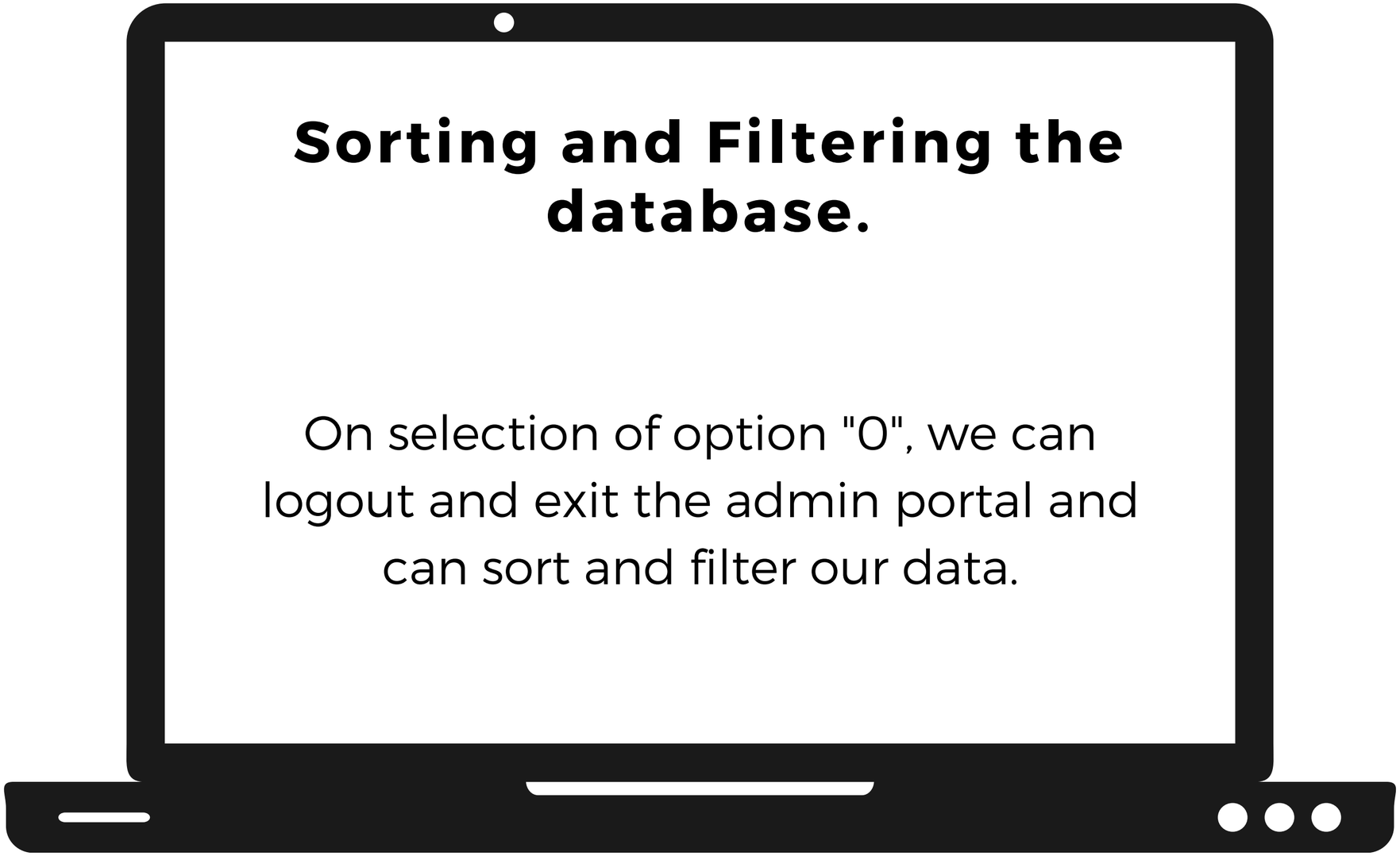
(choice: 4)

We can add new student.

(choice: 5)

We can remove a student

(choice: 6)


Sorting and Filtering the database.
On selection of option "0", we can logout and exit the admin portal and can sort and filter our data.

SORTING AND FILTERING OPTIONS

DOB_ASC/ DOB_DSC	TO SORT THE LIST OF STUDENTS ACCORING TO THEIR DATE OF BIRTH IN ASCENDING OR DESCENDING ORDER.
NAME_ASC/ NAME_DSC	TO SORT THE LIST OOF STUDENTS ACCORDING TO THEIR NAME IN ASCENDING OR DESCING ORDER
DOJ_ASC/ DOJ_DSC	TO SORT THE LIST OOF STUDENTS ACCORDING TO THEIR DATE OF JOINING IN ASCENDING OR DESCING ORDER
ENROLL_ASC/ ENROLL_DSC	TO SORT THE LIST OOF STUDENTS ACCORDING TO THEIR ENROLLMENT NUMBER IN ASCENDING OR DESCING ORDER

Key Features

Of our project

Object oriented Methodology:

The project works on multiple classes and structs interacting with each other to produce the results modelling real life

Security of Data:

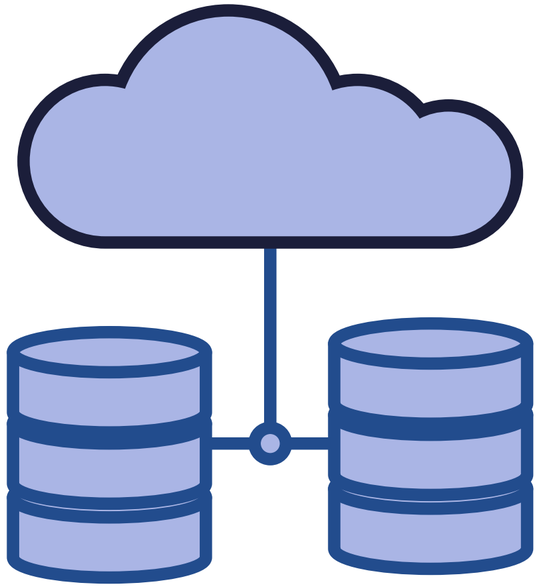
Student data can be read without admin privileges but student details can not be altered outside the admin portal so the data can't be tampered with

Fast access to information:

All of the student details are loaded into the memory at the time the program is called, so the information can be accessed quickly compared to reading from a file everytime

TARGET RESULT

TO DISPLAY SORTED DATA OF STUDENTS ACCORDING TO USER PREFERENCE



MAINTAIN A STUDENT DATABASE WITH THE THE HELP OF ADMIN TOOLS

TO PERFORM VARIOUS OPERATIONS ON DATA VIA THE ADMIN PORTAL

Design Process

PHASE 1

Design the student class along with all the required setter and getter functions

PHASE 2

Make the two structs (date and stud_details) for use in the student class

PHASE 3

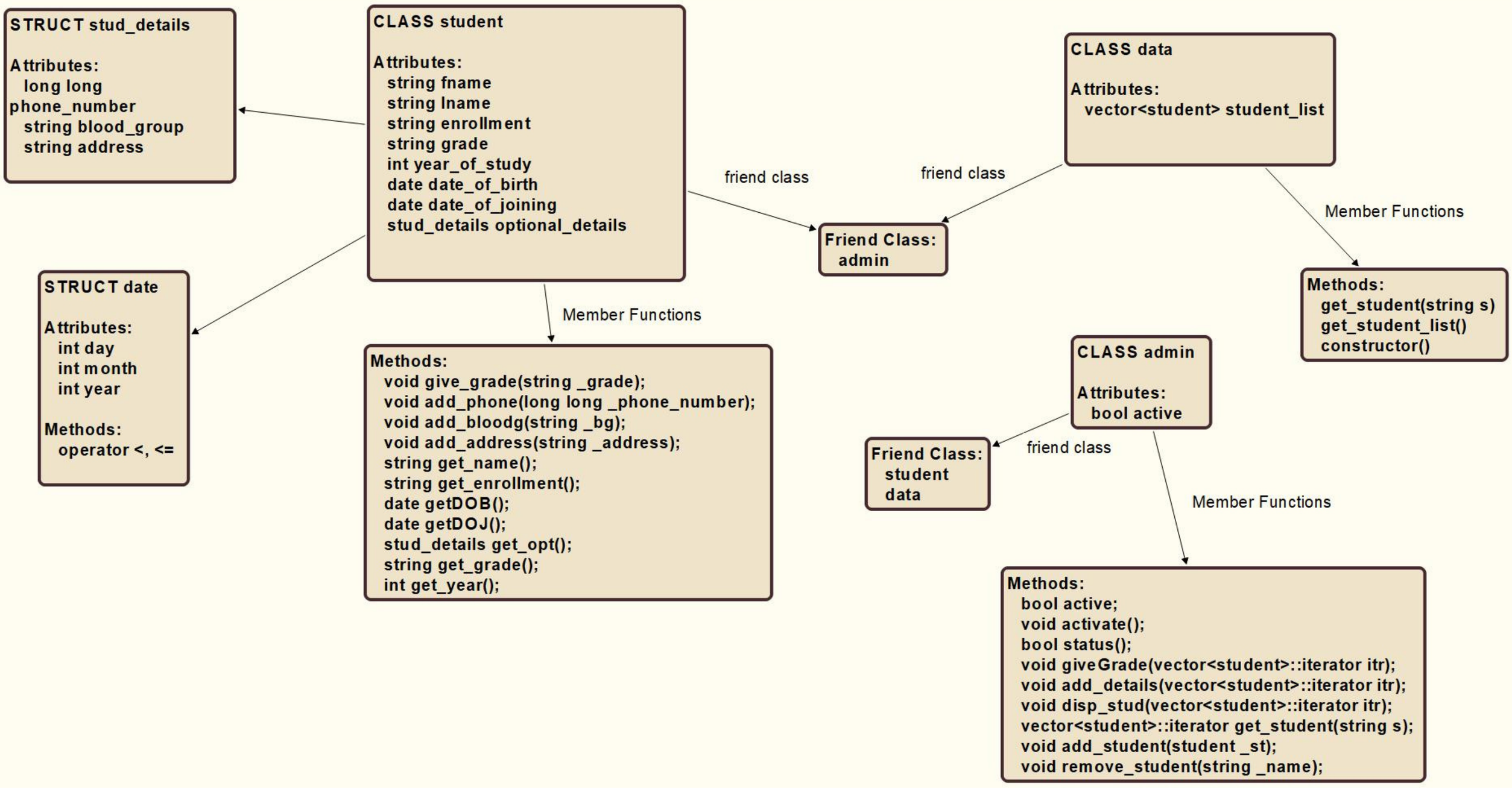
Make a class to fetch and store all of the students information - the data class

PHASE 4

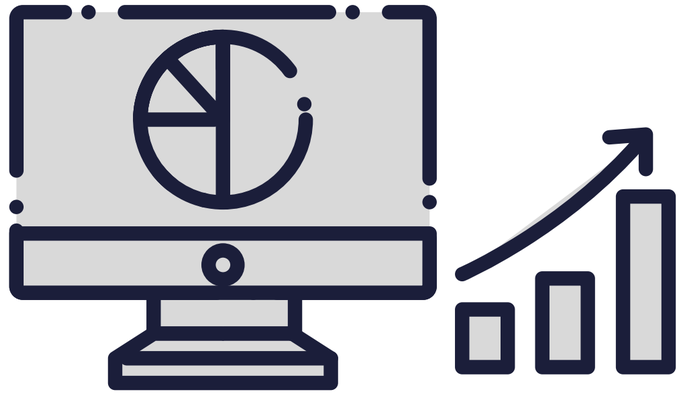
Decide all the functionality of the admin class to maintain the student records

PHASE 5

Finalise the desgin of the admin class and which functions from data and student class it accesses


STRUCT stud_details
Attributes:
long long phone_number
string blood_group
string address
CLASS student
Attributes:
string fname
string lname
string enrollment
string grade
int year_of_study
date date_of_birth
date date_of_joining
stud_details optional_details
CLASS data
Attributes:
vector<student> student_list
friend class
friend class
Member Functions
Friend Class:
admin
Methods:
get_student(string s)
get_student_list()
constructor()
STRUCT date
Attributes:
int day
int month
int year
Methods:
operator <, <=
Member Functions
Methods:
void give_grade(string _grade);
void add_phone(long long _phone_number);
void add_bloodg(string _bg);
void add_address(string _address);
string get_name();
string get_enrollment();
date getDOB();
date getDOJ();
stud_details get_opt();
string get_grade();
int get_year();
CLASS admin
Attributes:
bool active
Friend Class:
student
data
friend class
Member Functions
Methods:
bool active;
void activate();
bool status();
void giveGrade(vector<student>::iterator itr);
void add_details(vector<student>::iterator itr);
void disp_stud(vector<student>::iterator itr);
vector<student>::iterator get_student(string s);
void add_student(student _st);
void remove_student(string _name);

Our Execution Strategy



PHASE 1

Design the various classes that help make the application work

PHASE 2

Figure out a solution to make the data quickly accessbile

PHASE 3

Turn the designed system into code and make sure all the functions work correctly

PHASE 4

Make a user friendly command line interface so that the application can be used by all users

PHASE 5

Put the finishing touches on the CLI and fix minor bugs in the various classes and functions


Thank
you