



CS101 Course Project

Autumn Semester 2011

Lab Group 542

Student Management System Software Requirements Specifications

Members :

- a. Pratyush Nalam
- b. Pravinkumar Raut
- c. Premchandra Rathaur
- d. Pratiik Malik
- e. Rishiraj Singh
- f. Rohit Bhange
- g. Prakhar Pankaj

Mentor : Avishek Dan



Revision History

Date	Version	Details	Author
19/10/2011	1.0	Added 1.a., 1.b., 1.c., 2.a., 2.b.	Pratyush Nalam
22/10/2011	1.1	Added 2.b.iii., 2.c.	Pratyush Nalam
23/10/2011	1.2	Updated 2.c.i. Added 2.c.ii.	Pravinkumar Raut
23/10/2011	2.0	Final Revision: Added 2.c.iii., 2.d., 3., 4.	Pratyush Nalam



Table of Contents

1. Introduction	- 4 -
a. Purpose	- 4 -
b. Technological compatibility	- 4 -
i. Operating System	- 4 -
ii. Programming Language	- 4 -
iii. Interface	- 4 -
c. References	- 4 -
d. Abbreviations	- 4 -
2. Project Description	- 5 -
a. Overview	- 5 -
b. User Interaction	- 5 -
i. Student	- 6 -
ii. Professor	- 7 -
iii. Administrator	- 8 -
c. Data Storage	- 9 -
i. Dynamic Arrays	- 9 -
ii. Files	- 9 -
iii. Integrating Dynamic Arrays and Files	- 9 -
d. Data Organisation	- 11 -
i. Student	- 11 -
ii. Professor	- 11 -
iii. Administrator	- 11 -
3. Clarifications	- 12 -
4. Future Scope	- 12 -



1. Introduction

a. Purpose

The purpose of this document is to specify the various requirements for the CS101 Course Project. It also describes the various functionalities included in the project. This document is intended for:

- Project Team
- End User(s)

b. Technological compatibility

i. Operating System

The software developed only runs on the Linux Operating System. Software portability wasn't considered as it is beyond our level.

ii. Programming Language

We have used the programming language, C++ (GNU C/C++ Compiler) to develop this software.

iii. Interface

The program runs completely on the Linux Terminal and hence, a Character User Interface (CUI) is used. The only Graphical User Interface (GUI) we learnt was EZWindows which doesn't possess the functionalities required by our project. Due to lack of time, learning and using another GUI wasn't considered. Since it runs completely on a Terminal, adequate importance was given so that the commands are absolutely clear and is as user-friendly as possible. In this way, it is ensured that the user doesn't miss a GUI.

c. References

The Internet was our main source of reference and we used it extensively to learn new features of C++ and how we can implement them in this project. Google, Wikipedia, C++ Resources Network among others were widely used.

d. Abbreviations

IIT	Indian Institute of Technology
CPI	Cumulative Performance Index
CUI	Character User Interface
GUI	Graphical User Interface

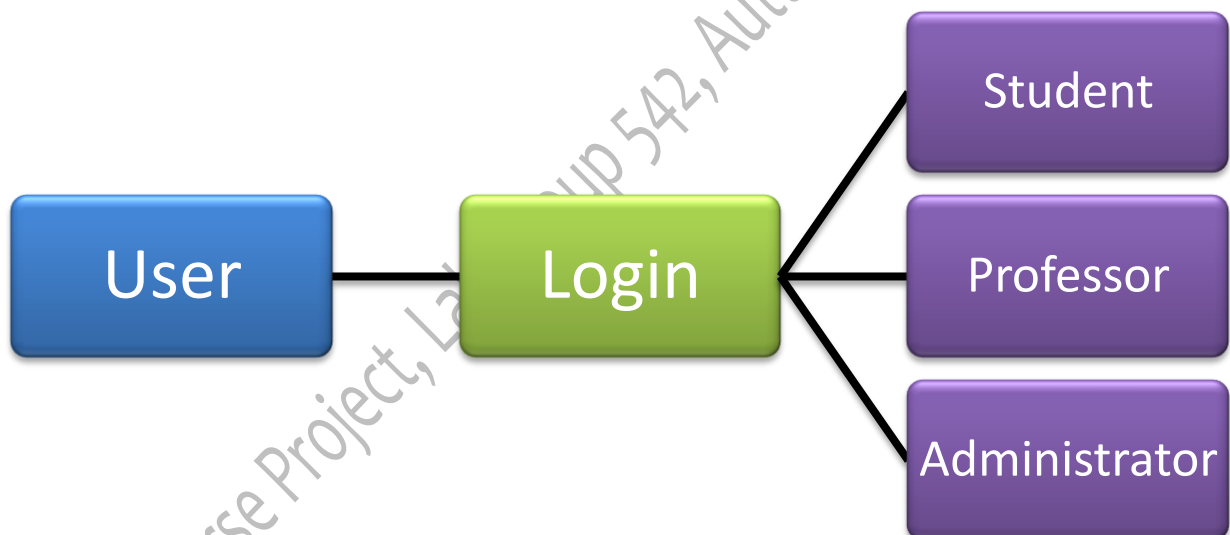
2. Project Description

a. Overview

The software developed is primarily a student management system. It contains a database of all students and professors at IIT Bombay. The main inputs given are the marks obtained by each student in various courses pursued by him/her and the percentage of his/her attendance in each course. Based on these inputs, the Cumulative Performance Index (CPI) of the student is calculated.

There are other supplementary details too. The database also contains the contact details of the students and professors. So, a student who would like to contact his/her professor just needs to search the database and (s)he is done. Similarly, this can also be used by a professor to search for any student etc.

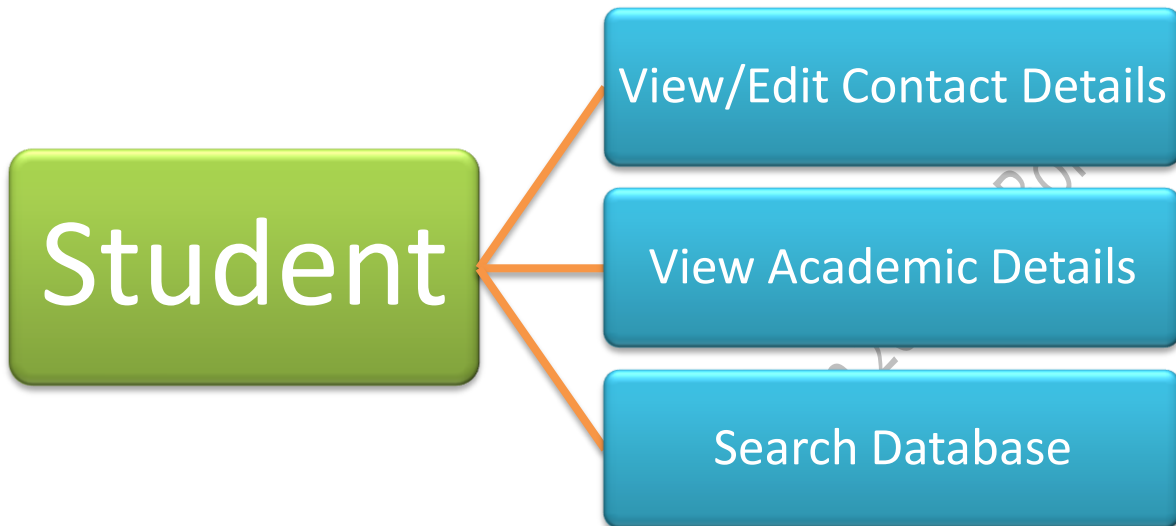
b. User Interaction



The first thing to be done by the user is to login into the system. This prevents unauthorised use and unwarranted manipulation of data. Depending on whether the user is a student or professor or administrator, the system will accordingly show various options available as detailed below.

i. Student

When a student logs into the system, (s)he is automatically recognised as a student and his/her name, roll number, department and which year and semester (s)he is studying will be shown. Further, three options will be displayed to the student. The details are described below.



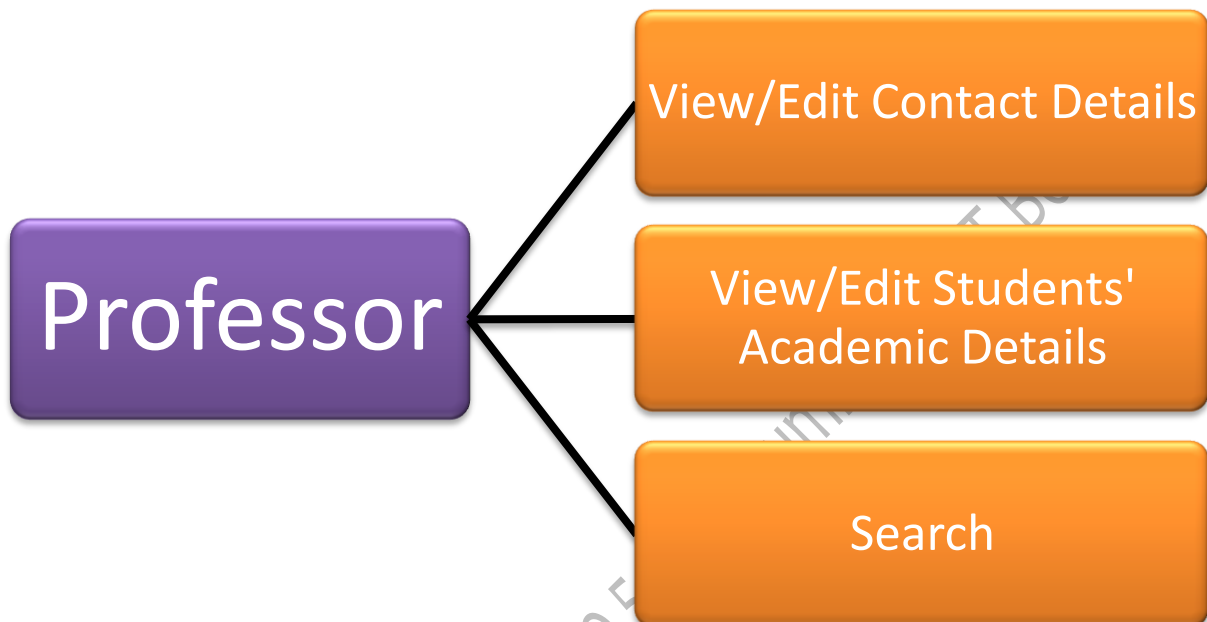
The first option is for the student to view or edit his/her contact details like name, phone number, department, hostel number etc. Except the roll number which doubles up as his/her username, department and year of study, the student has complete control over his/her contact details and can change them as he wishes. (S)he will also be able to change his/her password as and when required.

The second option for the student is to view his/her academic details, namely marks and attendance. For each course the student is enrolled into, (s)he will be able to view the marks (s)he has got out of 10 and the grade attained for the respective course. Also, displayed will be his/her percentage of attendance in each course. Moreover, at the end, the CPI or Cumulative Performance Index of the student will be displayed.

The final option available to the student will be "Search". The student will be able to search the entire database of students and professors by typing his/her name and will be only able to view the contact details of either the student or professor. This is so that the student finds it easy to contact his/her professor if (s)he has any doubt. It also makes it easy for him/her to contact his/her classmate(s) for any assistance.

ii. Professor

When a professor logs into the system, (s)he is automatically recognised as a professor and his/her name, department and the courses taught by him/her are displayed. Then, three options are presented to the professor so that (s)he can choose what to do.



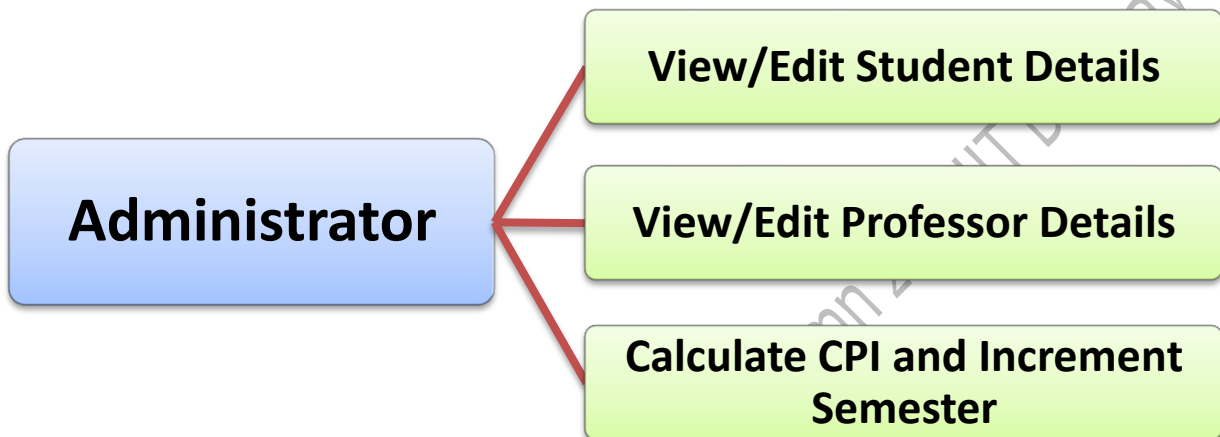
Just as in the case of the student, the professor will also be able to view and edit his/her contact details. These include name, email id, work phone number etc. The professor has complete control over them and can edit any of his details as per his wish.

The second option is the main and most important of them all. The professor will enter the marks and attendance of the students who are enrolled into his/her course(s) and will be able to edit them as per requirement. This is required in order to calculate the CPI of the students which forms the most important component of this project.

The last option is "Search". Similar to the "Search" in the student interface, here too, the professor will be able to search for any student/professor in the database and retrieve his contact details. As mentioned before, this is to facilitate easy communication between the various users. In this way, the professor can contact any student(s) to pass on any important information.

iii. Administrator

The administrator is a user with complete control over the student management system. A series of options are displayed to him/her and described below are what functions the administrator can perform.



The first option is the facility to view and edit student details. Since the administrator must have complete powers, it was imperative to include this option. This has a very important application. Suppose the student changes his/her hostel and wants to change it, the administrator by virtue of his/her all-encompassing powers can change his hostel and room number.

Similar to the student case, the administrator has the power to view and edit professor details as well.

The last functionality is the most important feature for the administrator which is the purpose of this project. Once he gets the news that for that particular semester, all the professors have entered the marks and attendance for all students, the administrator will invoke the last function, "Calculate CPI and Increment Semester". This will calculate the CPI of all the students, increment the semester of the student to the next semester.



c. Data Storage

This project involves a huge amount of data generation and two concerns immediately come to the mind. The first is permanent data storage and the second is fast and easy accesses to this data without a noticeable time lag. Both these issues would have been sufficiently addressed if a proper database management system such as SQL could be used. But, since C++ lacks the functionality to co-ordinate with databases, we had to devise our own method using the features provided by C++. We are using two types of data storage systems: Dynamic Array Allocation and Files. First, these two systems will be explained followed by how they are integrated into our project.

i. Dynamic Arrays

Fast and easy access to data is one of the most important concerns in a database management system. By using dynamic arrays, the program uses the memory or RAM of the system and this leads to very fast access of data. Some algorithms which could make data access even faster has not been included due to lack of time.

ii. Files

This project will have three data files containing the databases of student and professor and also the login details of the administrator. When the program is executed, it creates two dynamic arrays using the data inside these files. While the dynamic array was a temporary storage, files are the backbone of the project. They contain all the data too and are a method of permanent storage.

iii. Integrating Dynamic Arrays and Files

The important question here is how we integrate the two. While dynamic arrays are an excellent way to store data and provide fast access for easy manipulation, they are only temporary. Even though the program is expected to run continuously, there might be an event when the program might be shut down. In that case, all the data will be lost which is undesirable.

On the other hand, files are a good way to store data permanently. Once written, it will never get lost even if you shut down the system as it is stored on the hard disk. But, the main irritant with files is that data access and manipulation is slow. It is approximately 10 times than doing it dynamically with temporary memory and using dynamic arrays. Even this is undesirable.



Hence, a compromise has to be reached. Thus, it was decided that both dynamic arrays and files will be used. As soon as the program is started, the initial data will be read from the files. Any further manipulations and modifications will be done to the dynamic arrays. When the user decides to shut down the program, all such modifications will be carried out into the files. In this way, we have fast and efficient access to data and also it is safely stored on permanent memory.

IMPORTANT ASSUMPTION: Over here, we assume that the program is not suddenly terminated due to system crashing or any other reasons. In that case, the data last stored on the file will be intact. The data is written to the file when the user explicitly says "Log Out" in the program by invoking the required option.

CS101 Course Project, Lab Group 542, Autumn 2011, IIT Bombay

d. Data Organisation

i. Student

Variable	Data Type
username	char[50]
password	char[20]
firstName	char[20]
lastName	char[20]
rollNo	int
mobile	long int
email	char[40]
hostel	int
roomNo	int
dept	char[3]
dobDate	int
dobMonth	int
dobYear	int
courseProg	int
semOfStudy	int
yearOfJoin	int
noOfCourses	int
course	char[10][6]
marks	int[10]
grade	char[10][3]
maxCredit	int[10]
semCredit	int[10]
spi	float[10]
cpi	float
attendance	float[10]

ii. Professor

Variable	Data Type
username	char[20]
password	char[20]
firstName	char[20]
lastName	char[20]
emailId	char[40]
officeNo	int
myDept	char[3]
myCoursesNo	int
myCourses	char[10][6]

iii. Administrator

Variable	Data Type
username	char[20]
password	char[20]



3. Clarifications

Any clarifications or questions can be mailed to cs101.542@gmail.com.

4. Future Scope

This project has a large scope for amazing improvements. Limited time to complete the project forced us to ignore some useful functionalities.

The first improvement is making data access faster. Dynamic arrays are still comparatively slow and large databases can cause time lags. Binary sort trees can speed up data access by a huge amount and this will be a huge advantage.

Secondly, the administrator currently cannot add or delete professors and students from the program. (S)he needs to go the text file and do the required jobs. This is highly inefficient and can be included in later versions of the program.

Thirdly, the administrator cannot change his username and password. This also can be added in future editions.

Fourthly, the username for each student and professor is fixed by the administrator and the user cannot edit it.

An important improvement urgently required is the invisibility of the password when logging in. Making it invisible was beyond our ability.

Lastly, there is no way to safely exit the program and this can only be done by pressing [Ctrl] + C. This also can be included in later versions of the program.