

## 1. ABSTRACT

Attendance management is important to every single organization; it can decide whether or not an organization such as educational institutions, public or private sectors will be successful in the future. Organizations will have to keep a track of people within the organization such as employees and students to maximize their performance.

Managing student attendance during lecture periods has become a difficult challenge. For the stated reason, an efficient system for attendance management system is designed to track student's activity in the class. This application takes attendance electronically and the records of the attendance are storing in a database.

In Current System staff has to manually mark each individual student absent or present, this limitation is overcome by taking input as only absent or present numbers and automatically marking other students. The process of recording attendances for students was in the form of hardcopy papers and the system was manually done. Besides wasting time and taking efforts for preparing sheets and documents, other disadvantages may be visible to the traditional one due to loss or damage to the sheets-sheet could be stolen. This System deals with all these problems.

## **2. INTRODUCTION**

### **2.1 Existing System**

In the existing system of attendance management, the work is heavy from staff's point of view. Existing system is manually driven process where staff has to mark each individual student present or absent also most of the existing system are still paper based and not digital. Thus, Existing system produces lots of work load for staff.

### **2.2 Proposed System**

To overcome the drawbacks of existing system proposed system has been evolved. This system focuses to reduce human efforts and saving time to generate effective results for both student and faculty.

In this system: -

- \* Admin will add/delete/update student's data.
- \* Using staff login credentials staff will update/view/edit record.
- \* In this system staffs can take attendance on the basis of
  1. Only present members.
  2. Only absent members.and remaining members would be marked automatically.

## **3. SYSTEM ANALYSIS**

### **3.1 Requirements**

#### **3.1.1 Functional**

Functional requirements define the fundamental actions that system must perform. The functional requirements for the system are divided into four main categories View, Add, Modify and Delete.

- View:  
This module includes view option.  
Faculty can search student's account and view
- Add:  
This module includes add option.
- Modify:  
This module includes modify option.
- Delete:  
This module includes delete option.  
User can delete student data.

#### **3.1.2 Non-functional**

##### **3.1.2.1 Performance Requirement**

Easy and efficient tracking and updating of attendance record can be done.

##### **3.1.2.2 Availability**

Software will be available only to the authorized users with Log in credential. For staff to add, view, update data. For student to view their attendance and for admin to add and update student and course record.

##### **3.1.2.3 Security**

Software should be managed only by the admin, only the users with Log in IDs can be able to access system based on their priorities assigned.

### **3.1.2.4 Maintainability**

No direct Back up in System, however admin can create his/her own backups of record from database.

## **3.2 Problem Definition**

In the existing system, the staffs have to individually mark each student present or absent which is very time consuming and tedious job. In existing system there were possibilities of errors thus causing wrong calculation of attendance percentage.

## **3.2 SRS-System Requirement Specification**

### **1. Introduction**

Student attendance is import factor in modern education system; it shows active participation of student in academics. Hence student attendance should be managed and stored efficiently. In this system student attendance is stored and managed efficiently. This document shows purpose, advantages of proposed system.

#### **1.1 Purpose**

The purpose of this document is to present detailed description of Student attendance management system. It will explain features, purpose of the software, interface of the software, what will software do, constraint under which software work. This document is intended for both end users and Developers.

#### **1.2 Scope**

The document covers the requirement for Student Attendance management system. This system will provide platform for users to perform various task related to attendance that is related to storing, updating or viewing student attendance record.

#### **1.3 Overview**

The purpose of this document is to present a detail description of the student attendance management system. It will explain purpose and features of the system, the interface, what will it do and constraint under which it must operate.

## **2. Overall Description**

This describes the general factors that affect the product and its requirements. This section does not state specific requirements.

### **2.1 Product Perspective**

The Student attendance management System is an independent stand-alone system. It is totally self-contained.

#### **2.1.1 Hardware Interfaces:**

System will be placed on PC's throughout the department.

#### **2.1.2 Software Interfaces:**

Software is written in C\C++ language. Log in will be required in order to access features of system like adding attendance, viewing and editing. No log in for wrong Id and Passwords.

### **2.2 Product Functions:**

- Login System

This system includes the username and password.

No user can access the software without the username and password.

- Operations

According to the interface decided the operations are done.

This operation's include Take attendance, View attendance. As the user needs the operations are done.

- Final report

Overall report of the attendance is displayed on student's as well as faculty's account.

### **2.3 Assumptions and Dependencies**

1. We assume that all the computers used for the system is part of college LAN.
2. End Users should have basic knowledge of using Computers.
3. Database of the students can be accessed.

### **3. Specific Requirements**

#### **3.1 External interface requirements**

The Student's attendance management System will use the standard input/output devices for a personal computer.

This includes the following:

- 1) Keyboard
- 2) Mouse
- 3) Monitor
- 4) Printer

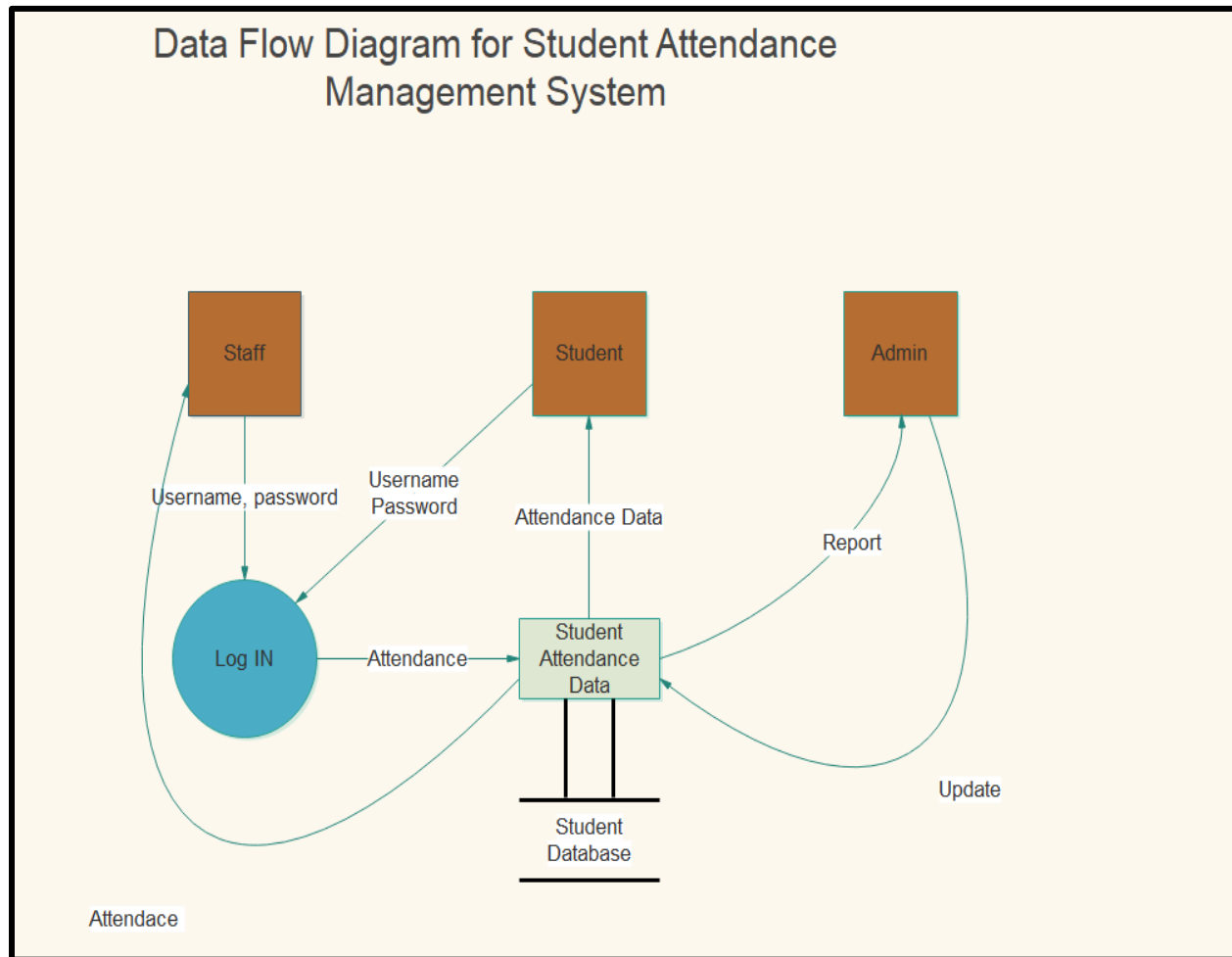
#### **3.2 Design Constraints**

The system provides security so Invalid Log is should be discarded. Only authorized user should be given rights to modify as per requirement. C/C++ is used as the programming language.

## 4. PROPOSED SYSTEM

### 4.1 Design

#### 4.1.1 System Design-DFD



#### 4.1.2 Detailed Design-User Interfaces

**Log In**  
  
**Username:** \_\_\_\_\_  
  
**Password:** \_\_\_\_\_

*Log IN Interface*

Roll No.	Name	Class	Semester

*Admin View of All Student*

Course	Date	Absent No.

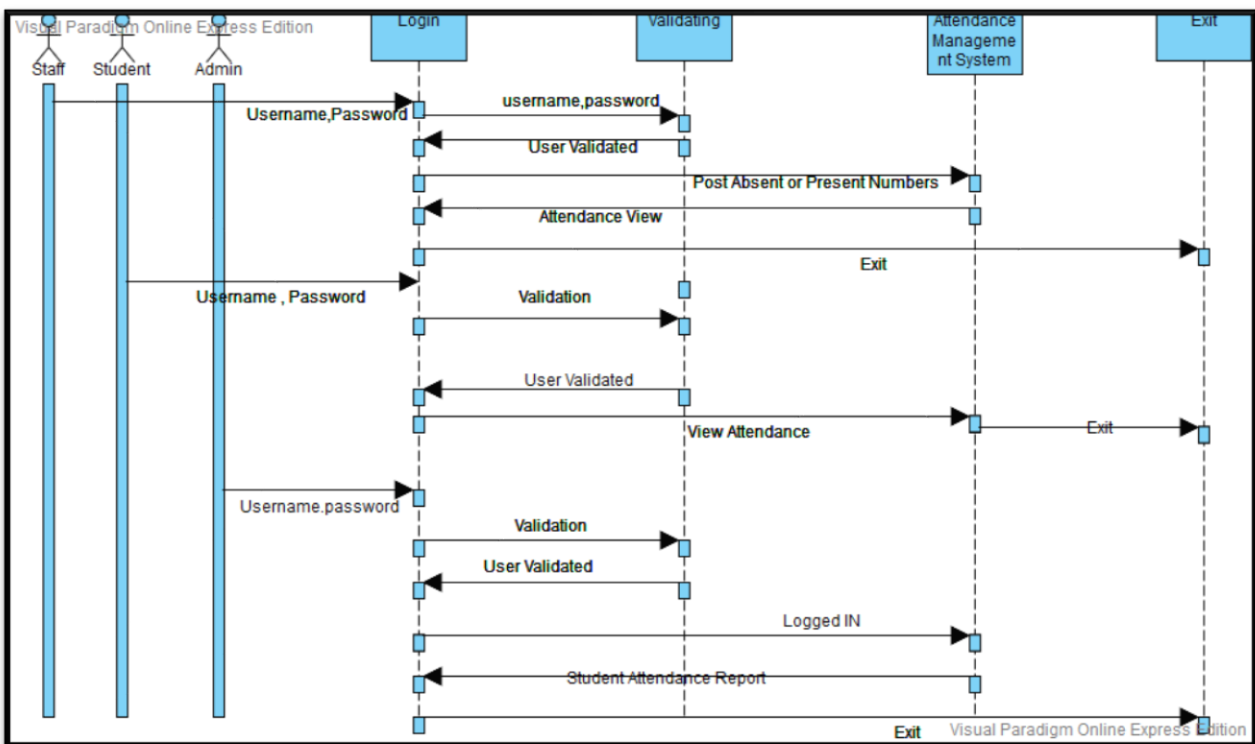
*Staff View of Attendance Data*



<b>Roll No:___</b>	
<b>Name:_____</b>	<b>Class:___ Sem:___</b>
<b>Course</b>	<b>Absent Date</b>

*Student View*

#### 4.1.2 Sequence Diagram



## 5. HARDWARE AND SOFTWARE REQUIREMENTS

### ➔ Recommended Operating Systems

Windows: 7 or newer

Software's: Visual Studio or Code blocks

### ➔ Hardware Requirements

We strongly recommend a computer fewer than 5 years old.

**Processor:** Minimum 1 GHz; Recommended 2GHz or more

**Hard Drive:** Minimum 32 GB; Recommended 64 GB or more

**Memory (RAM):** Minimum 1 GB; Recommended 4 GB or above

**Monitor/Display:** 14' LCD monitor, resolution 1600 x 900 or better

**Other peripherals:** Keyboard, Mouse.

## 6. IMPLEMENTATION

### 6.1 Coding

The System is completely implemented using C++ programming language. Files are used to store data. In this system 4 Files are used out of which 2 files are used to store log in credentials and 1 file for attendance Data and the remaining one for Student data. Data is stored in file through structure. 2 types of structure are used one for student data and other for attendance data. Student structure includes fields like Name of student, Roll Number, Class and Semester.

Attendance structure consist of Course name, Date and Boolean array for Absent or present.

#### →Log IN

For Log in Username and password are taken as input and stored in strings. Login.dat file is opened as read using ifstream, Each line from Login.dat is compared with username and password (using getline() method) and flags are set if comparison is true, after the end of file flags are compared and if all flags are 1 user is logged in else he is re-directed to log in screen.

#### →Take Attendance

First attendance data file is opened in append mode, Data like Department, class and course is selected from menu's provided through switch case. Date is taken as input in the form of strings in the format DD-MM-YY. Finally Students roll number is taken as input in form of string separated by commas. This string is converted into array of integers using stoi() method.

Structure to be written contains data fields like { course\_name, Date, Absent\_no\_bool }

Boolean value of Absent\_No is set to true if number is present in array of integer else it is false.

Whole structure is then written inside data file.

Finally the file is closed.

## 7. TESTING

In a software development project, errors can be introduced at any stage during development. To ensure quality of the final delivered software these defects will have to be removed. During testing, the software under test (SUT) is executed with a finite set of test cases, and the behaviour of the system of these test cases is evaluated to determine if the system is performing as expected. While discussing testing, we commonly use terms like error, fault, failure, etc. The term error is used in two different ways, it refers to the discrepancy between a computed, observed or measured value and the true specified or theoretically correct value. Fault causes a system to fail in performing its required function. The term error is also used to refer the defects. Failure is the inability of a system or component to perform a required function according to its specifications.

### 7.1 Verification

The verifying process includes checking documents design, code and program. It does not involve executing the code. Verification uses methods like review, walkthrough, inspection, desk checking, etc. It finds bugs early in the development cycle. Target is application and software architecture, specification complete design, etc. QA team does verification and make sure, that software is as per requirement in SRS document. It comes before validation.

In attendance management system all the requirements are verified check listed with the requirement document.

#### 7.1.1 Verification Techniques:

Verification is static type of s/w testing. It means code is not executed. The product is evaluated by going through the code. Types of verification are:

- i. Walkthrough
- ii. Inspection
- iii. Reviews

**Walkthroughs** are informal, initiated by the author of the s/w product to a colleague for

assistance in locating defects or suggestions for improvements. They are usually unplanned. Author explains the product; colleague comes out with observations and author notes down relevant points and takes corrective actions.

**Inspection** is a thorough word-by-word checking of a software product with the intention of:

- Locating defects
- Confirming traceability of relevant requirements
- Checking for conformance to relevant standards and conventions

**Review** is a subsequent examination of a product for the purpose of monitoring earlier changes. It is a process in which one or more persons check the changed documents or data to determine if the changes are correct. It is also an analysis undertaken at a fixed point in time to determine the degree to which stated objectives have been reached.

### **7.1.2 Checklist for SRS**

#### **General (SRS requirements) Checklist**

1. Is a functional overview of the system provided?

**Yes**, all the required functions as per requirements are implemented.

2. Have the software and hardware environments been specified?

**Yes**

3. If assumptions that affect implementation have been made, are they stated?

**No**

4. Are all the requirements, interfaces, constraints, definitions, etc. listed in the appropriate sections?

**Yes**

#### **Interface Checklist**

1. Are all inputs and outputs to the system specified?

**Yes**

2. Are all interface requirements between hardware, software, personnel, and procedures included?

**Yes**

**Behavioural Requirements Checklist**

1. Have all requirements described in the problem statement and in subsequent communications with the customer been specified?

**Yes, customer requirements are fulfilled.**

2. Are all inputs to a function sufficient to perform the required function?

**No.**

3. Are undesired events/inputs considered and their required responses specified?

**No.**

**Non-Behavioural Requirements Checklist**

1. Is the reliability specified, including the consequences of software failure, the vital information that needs to be protected from failure, and the strategy for error detection and recovery?

**No, since scope is limited.**

2. Are planned changes specified (i.e., maintainability)?

**Yes**

**Requirements Quality**

1. Does each requirement avoid conflicts with other requirements?

**Yes, concurrency is provided.**

2. Does each requirement have a priority? (e.g. essential program functionality should not be on a low priority)

**Yes, as per priority and level each requirement is provided.**

3. Is each requirement testable? Will it be possible for independent testing to determine whether each requirement has been satisfied?

**Yes, required test cases are to be provided.**

## **7.2 Validation**

Validation is a dynamic mechanism of validating and testing the actual product. It always involves executing the code. It is computer-based execution of program. Validation uses methods like black box (functional) testing, grey box testing, and white box (structural) testing etc. Validation is to check whether software meets the customer expectations and requirements. It generally follows after verification.

### **7.2.1 Levels of Testing**

There are four levels of testing as described below:

#### **7.2.1.1 Unit Testing**

During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as designed. In this phase, a unit can refer to a function, individual program or even a procedure, and a White-box Testing method is usually used to get the job done. One of the biggest benefits of this testing phase is that it can be run every time a piece of code is changed, allowing issues to be resolved as quickly as possible. It's quite common for software developers to perform unit tests before delivering software to testers for formal testing.

In our project, we have chosen date as a parameter for unit testing. Only specific date module was tested individually. User enters the date which is stored in a structure in the form DD/MM/YYYY. The date is validated by giving condition to the DD MM and YYYY. Wherein if DD is greater than 31, or MM is greater than 12, the date will be treated invalid. Also condition is checked for 0, wherein if any of DD or MM or YYYY is zero then date is treated invalid. Thus only valid date is taken into the file, for rest invalid message is displayed at the output screen.

### **7.2.1.2 Integration Testing**

Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to find interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they aren't properly integrated, it will affect the functionality of the software program. In order to run these types of tests, individuals can make use of various testing methods, but the specific method that will be used to get the job done will depend greatly on the way in which the units are defined.

In our System while showing Attendance Record to Staff or Student various units like date, course name and Roll Numbers combined results are considered. Thus date along with attendance view is integrated.

### **7.2.1.3 System Testing**

System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven't played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

### **7.2.1.4 Acceptance Testing**

The final level, Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business' needs. Once this process has been completed and the software has passed, the program will then be delivered to production.



## **7.2.2Types of Testing**

There are two types of testing that is black box testing and the white box testing.

### **7.2.2.1 Black Box Testing:**

In black box testing , the tester only knows the inputs that can be given to the system and what output the system should give. The basis for deciding test cases in black box testing is the requirement or specifications of system. So, it is also called as functional or behavioural testing.

Techniques that can be used to select test cases are:

#### **7.2.2.1.1 Equivalence class partitioning:**

An equivalence class is formed of the inputs for which the behavior of the system is specified or expected to be similar. The relational of forming equivalence classes like this is the assumption that if the specifications require the same behavior for each element in a class of values, then the program is likely to be constructed so that it either succeeds fails for each of the values in that class.

#### **7.2.2.1.2 Boundary Value Analysis (BVA):**

In BVA, we choose an input for a test case from an equivalence classes, such that the input lies at the edge of the equivalence classes. Boundary value test case is a set of the input data that lies on the edge or boundary of a class of input data or that generates output that lies at the boundary of a class of the output data.

In This System, login function is black box tested where only inputs such username and password is satisfying with all the conditions present in database is tested.

### **White Box Testing:**

White-box testing, on the other hand, is concerned with testing the implementation of the program. The intent of this testing is not to exercise all the different input or output conditions

(although that may be a by-product) but to exercise the different programming structures and data structures used in the program. White-box testing is also called structural testing, and we will use the two terms interchangeably. To test the structure of a program, structural testing aims to achieve test cases that will force the desired coverage of different structures. Various criteria have been proposed for this. Unlike the criteria for functional testing, which are frequently imprecise, the criteria for structural testing are generally quite precise as they are based on program structures, which are formal and precise.

Three different approaches to structural testing –

- Control flow based testing
- Data flow based system
- Mutation testing

## **8. PROJECT PLANNING**

The project planning process involves a set of interrelated activities followed in an orderly manner to implement user requirements in software and includes the description of a series of project planning activities and individual(s) responsible for performing these activities. In addition, the project planning process comprises the following.

- Objectives and scope of the project
- Techniques used to perform project planning
- Effort (in time) of individuals involved in project
- Project schedule and milestones
- Resources required for the project
- Risks associated with the project.

Project planning process comprises several activities, which are essential for carrying out a project systematically. These activities refer to the series of tasks performed over a period of time for developing the software. These activities include estimation of time, effort, and resources required and risks associated with the project.

1. Defining problem statement
2. Project planning Activities
3. Planning and scheduling

## Planning and scheduling:

Task to be planned	Week
Group Formation	1 <sup>st</sup> week of January
Searching for problem statement	2 <sup>nd</sup> Week of January
Presenting and Finalizing topic	3 <sup>rd</sup> and 4 <sup>th</sup> Week of January
Information Gathering and Survey	1 <sup>st</sup> and 2 <sup>nd</sup> Week of February
System design document writing	3 <sup>rd</sup> Week of February
Implementation and Coding	4 weeks of March
Testing	1 <sup>st</sup> week of April
Report	2 <sup>nd</sup> week of April

### 8.1 Project Model

There are 5 types of models in software development life cycles:

1. Waterfall model.
2. Iterative model.
3. Spiral model.
4. V-shaped model.
5. Agile model.

The simplest process model is the waterfall model, which states that the phases are organized in a linear order. The model was originally proposed by Royce, though variations of the model have evolved depending on the nature of activities and the flow of control between them. In this model, a project begins with feasibility analysis. Upon successfully demonstrating the feasibility of a project, the requirements analysis and project planning begins. The design starts after the requirements analysis is complete, and coding begins after the design is complete. Once the programming is completed, the code is integrated and testing is done. Upon successful completion of testing, the system is installed. After this, the regular operation and maintenance of the system take's place. The model is shown in Figure

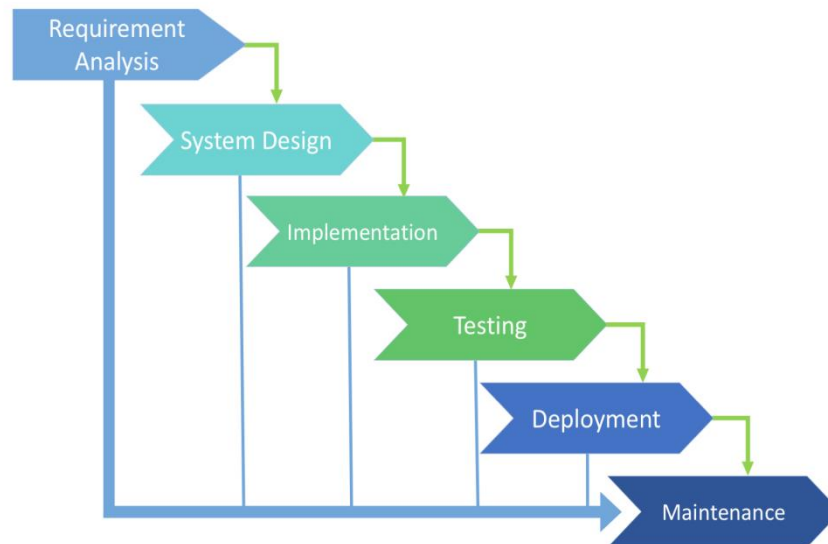


Figure 8.1.2 Project Model: Water Fall

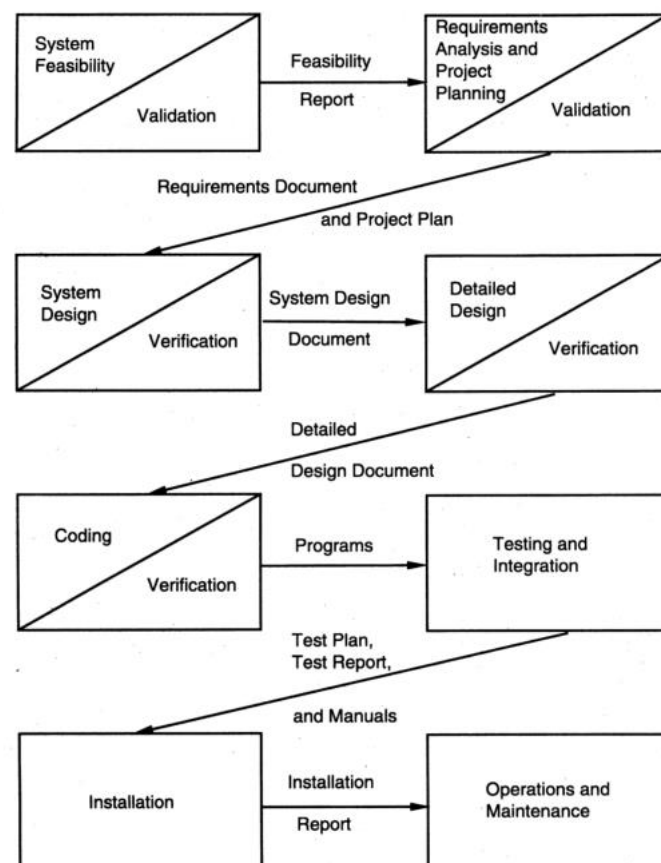


Figure 8.1.2 Water Fall

## **1. Requirement analysis**

Software requirement is a functional or non-functional need to be implemented in the system. Functional means providing particular service to the user.

For the requirement analysis of this project, survey of contineo and moodle was done.

Various requirement of the faculty as well as students were considered to design this software.

## **2. System design**

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

## **3. Implementation and Testing**

System testing is an essential step for the development of a reliable and error-free system. Once source code has been generated, software must be tested to uncover and correct as many errors as possible before delivery to your customer.

The requirements gathered were implemented and tested, whether the requirements are fulfilled or not.

The waterfall model, although widely used, has some strong limitations.

Some of the key limitations are:

1. It assumes that the requirements of a system can be frozen (i.e., baselined)

before the design begins. This is possible for systems designed to automate an existing manual system. But for new systems, determining the requirements is difficult as the user does not even know the requirements. Hence, having unchanging requirements is unrealistic for such projects.

2. Freezing the requirements usually requires choosing the hardware (because it forms a part of the requirements specification). A large project might take a few years to complete. If the hardware is selected early, then due to the speed at which hardware technology is changing, it is

likely that the final software will use a hardware technology on the verge of becoming obsolete. This is clearly not desirable for such expensive software systems.

3. It follows the “big bang” approach—the entire software is delivered in one shot at the end. This entails heavy risks, as the user does not know until the very end what they are getting. Furthermore, if the project runs out of money in the middle, then there will be no software. That is, it has the “all or nothing” value proposition.

4. It encourages “requirements bloating”. Since all requirements must be specified at the start and only what is specified will be delivered, it encourages

5. It is a document-driven process that requires formal documents at the end of each phase.

## 9. SCREENSHOTS

"F:\Storage\SEM 4\v4\V5\main.exe"

```
*****Attendance Management System*****  
  
Enter Login Credentials  
Username: CSE.staff1  
Password: *****
```

*Figure 9.1 Main Login Page*

"F:\Storage\SEM 4\v4\V5\main.exe"

```
MAIN MENU  
1. Take Attendance  
2. View Attendance  
3. Close Application  
  
Enter your choice:
```

*Figure 9.2 Staff Main Page*



"F:\Storage\SEM 4\V4\V5\main.exe"

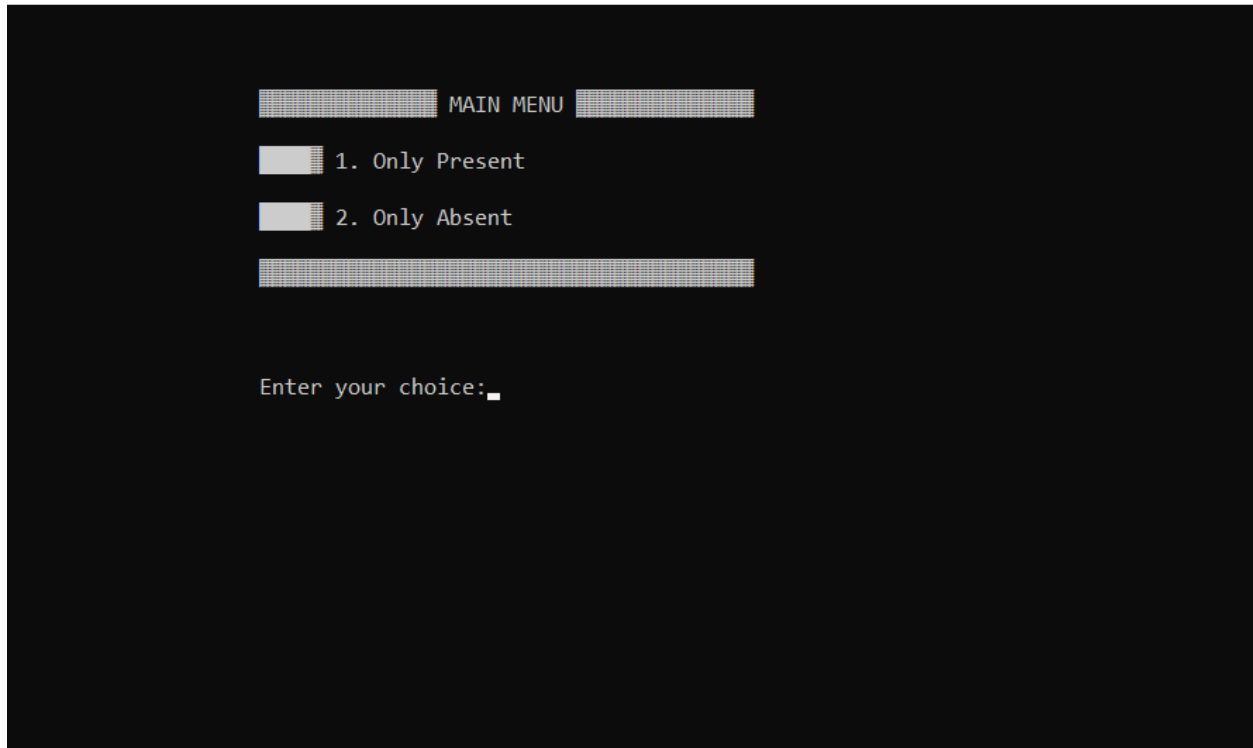


Figure 9.3 Staffs Sub Menu

"F:\Storage\SEM 4\V4\V5\main.exe"

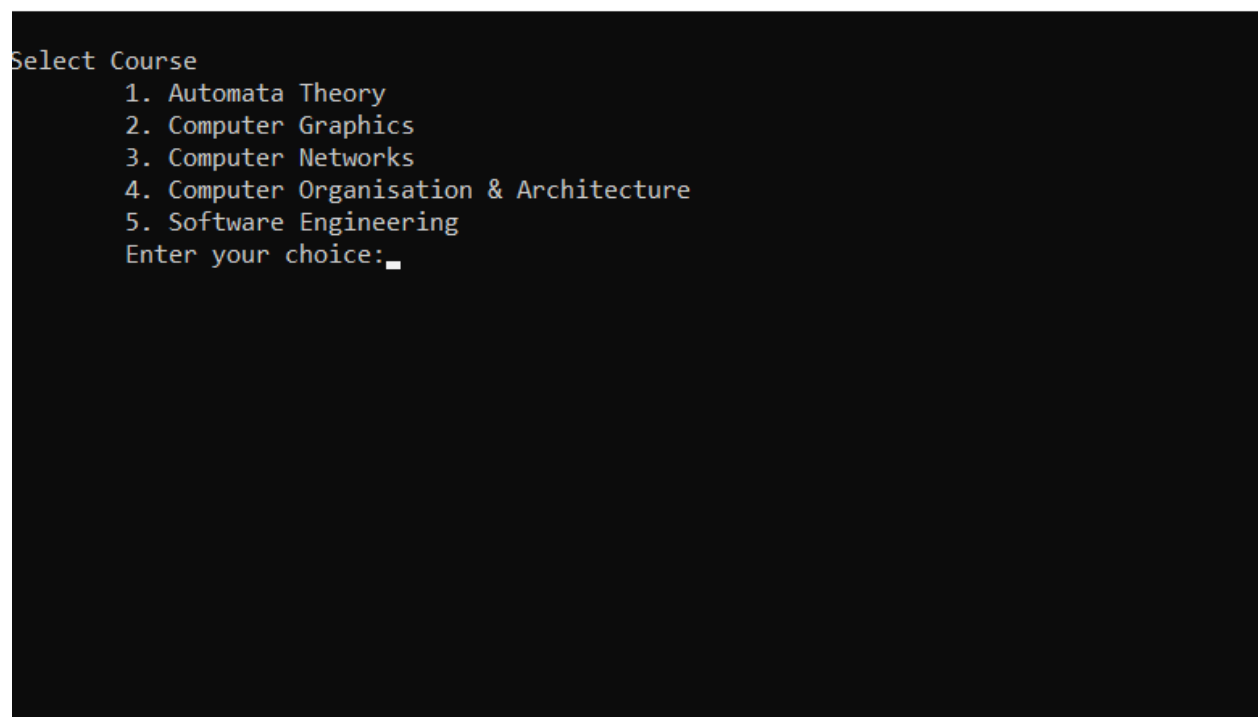


Figure 9.4 Course Selection Menu

"F:\Storage\SEM 4\v4\V5\main.exe"

```
Enter Date (DD\MM\YYYY): 25/04/19
Enter Absent number separated by comma
press enter to save: _
```


*Figure 9.5 Entry of Attendance*

"F:\Storage\SEM 4\v4\V5\main.exe"

```
MAIN MENU
1. View All Students
2. Add Student


Enter your choice: _
```

*Figure 9.6 Admin Main Menu*

 "F:\Storage\SEM 4\4\5\main.exe"

```
*****Absent Record*****
Roll NO      Name                Class  Semester
1           Sadaf_Attar           SY     2nd
2           Mehul_Banghe          SY     2nd
3           Sneha_Bhosale          SY     2nd
4           Sidhant_Bhukshete       SY     2nd
5           Sunil_Bilur            SY     2nd
6           Tanishka_Chougule       SY     2nd
7           Sharvari_Dalvi         SY     2nd
8           Richa_Desai            SY     2nd
9           Rugved_Etavadekar       SY     2nd
10          Karan_Gavali           SY     2nd
11          Saloni_Ghadi           SY     2nd
12          Shubham_Ghorpade       SY     2nd
13          Rushikesh_Gurav        SY     2nd
14          Vedika_Hardikar        SY     2nd
15          Karan_Hariyani         SY     2nd
16          Shivam_Hasurkar        SY     2nd
17          Samiksha_Ingole        SY     2nd
18          Aditya_Jadhav          SY     2nd
19          Akshay_Jadhav          SY     2nd
20          Parth_Jadhav           SY     2nd
21          Ganesh_Wakade          SY     2nd
22          Sameesh_Yadav          SY     2nd
23          Sneha_Yadav            SY     2nd
24          Vageshwar_Yadav        SY     2nd
25          Anmol_Waghmare         SY     2ndPress any key to exit_
```

Figure 9.7 Admin view: List of all Student

 Select "F:\Storage\SEM 4\4\5\main.exe"


```
*****Attendance Management System*****  
  
Enter Login Credentials  
Username: Student01  
Password: *****  
  
-----Student Log in-----
```

Roll NO:1

Name: Sadaf Attar

Class: SY

Semester: 2nd

*Figure 9.8 Student View 1* Select "F:\Storage\SEM 4\4\5\main.exe"

```
*****Absent Record*****  
  
Course          Date  
CG              25/04/1919  
  
Press any Key to exit
```

*Figure 9.9 Student View 2*

## 10. CONCLUSION AND LIMITATIONS

The Attendance Management System is developed in C++ fully meets the objectives of the system which it has been developed. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency and all the teachers and user associated with the system understands its advantage. The system solves the problem. It was intended to solve as requirement specification.

### → Limitations

- The machine on which the project is to be executed should be able to execute C++ program and should also support dos.h library function.
- The system was developed with minimum scope, there is no backup in case database file gets lost or corrupted.
- In case number of students increases manual changes are needed to be done in the code.

## 11. REFERENCES

[I]. <https://www.geeksforgeeks.org/readwrite-structure-file-c/>

[II]. <https://stackoverflow.com/questions/50847099/writing-structure-to-a-text-file-in-c>

[III]. **Software Engineering a precise approach-** by Pankaj Jalote, SRS document and requirement analysis; page no. :38-44

[IV]. **Software Engineering a precise approach** -By Pankaj Jalote Data Flow Diagram Level 0,1; page no. :59

[V]. **Software Engineering a precise approach** -By Pankaj Jalote Testing: Unit testing page no :230

[VI]. **Object Oriented Modelling and Design with UML** –By Michel R Blaha second edition Class diagram: page no :87