STUDENT RECORD MANAGEMENT SYSTEM

High Level Design & Low-Level Design

The purpose of this document is to provide a template for documenting both HLD & LLD.

**Document Control:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  | |  |  | |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | **Approver Signature** | | |
| 16-01-2023 | HLD\_LLD\_Group 7 version 0.1 | Saisree E |  | | | Prashanth | | |
| 16-01-2023 | HLD\_LLD\_Group 7 version 0.1 | Vidyalakshmi D B |  | | | Prashanth | | |
| 16-01-2023 | HLD\_LLD\_Group 7 version 0.1 | Umajyothi Yeleti |  | | | Prashanth | | |
| 16-01-2023 | HLD\_LLD\_Group 7 version 0.1 | Mutyala Tharaka Saieswari |  | | | Prashanth | | |

[**1. Introduction**](#_heading=h.3znysh7) **5**

[1.1. Intended Audience](#_heading=h.2et92p0) 5

[1.2. Acronyms/Abbreviations](#_heading=h.tyjcwt) 5

[1.3. Project Purpose](#_heading=h.3dy6vkm) 5

[1.4. Key Project Objectives](#_heading=h.1t3h5sf) 5

[1.5. Project Scope and Limitation](#_heading=h.4d34og8) 6

[1.6. Functional Overview](#_heading=h.3rdcrjn) 6

[1.7. Assumptions, Dependencies & Constraints](#_heading=h.26in1rg) 7

[1.8. Risks](#_heading=h.lnxbz9) 7

[**2. Design Overview**](#_heading=h.35nkun2) **7**

2[.1 Design Objectives](#_heading=h.1ksv4uv) 7

[*2.1.1. Recommended Architecture*](#_heading=h.44sinio) 8

[2.2. Architectural Strategies](#_heading=h.2jxsxqh) 8

[*2.2.1. Design Alternative*](#_heading=h.z337ya) 8

[*2.2.2. Reuse of Existing Common Services/Utilities*](#_heading=h.3j2qqm3) 8

[*2.2.3. Creation of New Common Services/Utilities*](#_heading=h.1y810tw) 9

[*2.2.4. User Interface Paradigms*](#_heading=h.4i7ojhp) 9

[*2.2.5. System Interface Paradigms*](#_heading=h.2xcytpi) 9

[*2.2.6. Error Detection / Exceptional Handling*](#_heading=h.1ci93xb) 9

[*2.2.7. Memory Management*](#_heading=h.3whwml4) 9

[*2.2.8. Performance*](#_heading=h.2bn6wsx) 9

[*2.2.9. Security*](#_heading=h.qsh70q) 9

[*2.2.10. Concurrency and Synchronization*](#_heading=h.3as4poj) 9

[*2.2.11. Housekeeping and Maintenance*](#_heading=h.1pxezwc) 9

[**3. System Architecture**](#_heading=h.49x2ik5) **10**

[*3.1. ER DIAGRAM*](#_heading=h.2p2csry) *10*

[*3.2. FLOWCHART*](#_heading=h.147n2zr) *11*

[*3.3. USECASE DIAGRAM*](#_heading=h.3o7alnk) *12*

*3.4. SEQUENCE DIAGRAM* 13

[3.5. System Interfaces](#_heading=h.23ckvvd) 14

[*3.5.1. Internal Interfaces*](#_heading=h.ihv636) 14

[*3.5.2. External Interfaces*](#_heading=h.32hioqz) 14

[**4. Detailed System Design**](#_heading=h.1hmsyys) **14**

[4.1. Key Entities](#_heading=h.41mghml) 14

[4.2. Detailed-Level Database Design](#_heading=h.2grqrue) 14

[*4.2.1. Data Mapping Information 1*](#_heading=h.vx1227)4

[*4.2.2. Data Conversion 1*](#_heading=h.3fwokq0)4

[4.3. Archival and retention requirements](#_heading=h.1v1yuxt) 14

[4.4. Disaster and Failure Recovery](#_heading=h.4f1mdlm) 15

[4.5. Business Process workflow](#_heading=h.2u6wntf) 15

[4.6. Business Process Modeling and Management (as applicable)](#_heading=h.19c6y18) 15

[4.7. Business Logic](#_heading=h.3tbugp1) 15

[4.8. Variables](#_heading=h.28h4qwu) 16

[4.9.PSEUDOCODE](#_heading=h.nmf14n) 16

[4.10. Data Migration](#_heading=h.37m2jsg) 19

[*4.10.1. Architectural Representation*](#_heading=h.1mrcu09) 19

[*4.10.2. Architectural Goals and Constraints*](#_heading=h.46r0co2) 19

[*4.10.3. Logical View*](#_heading=h.2lwamvv) 20

[*4.10.4. Architecturally Significant Design Packages*](#_heading=h.111kx3o) 20

[*4.10.5. Data model*](#_heading=h.3l18frh) 21

[*4.10.6. Deployment View*](#_heading=h.1egqt2p) 21

[**5. Environment Description**](#_heading=h.3ygebqi) **22**

[5.1. Time Zone Support](#_heading=h.2dlolyb) 22

[5.2. Language Support](#_heading=h.sqyw64) 22

[5.3. User Desktop Requirements](#_heading=h.3cqmetx) 22

[5.4. Server-Side Requirements](#_heading=h.1rvwp1q) 22

[*5.4.1. Deployment Considerations*](#_heading=h.4bvk7pj) 22

[*5.4.2. Application Server Disk Space*](#_heading=h.2r0uhxc) 23

[*5.4.3. Database Server Disk Space*](#_heading=h.1664s55) 23

[*5.4.4. Integration Requirements*](#_heading=h.3q5sasy) 23

[*5.4.5. Jobs*](#_heading=h.25b2l0r) 23

[*5.4.6. Network*](#_heading=h.kgcv8k) 23

[*5.4.7. Others*](#_heading=h.34g0dwd) 23

[5.5. Configuration](#_heading=h.1jlao46) 23

[*5.5.1. Operating System*](#_heading=h.43ky6rz) 23

[*5.5.2. Database*](#_heading=h.2iq8gzs) 23

[*5.5.3. Network*](#_heading=h.xvir7l) 23

[*5.5.4. Desktop*](#_heading=h.3hv69ve) 23

[**6. References**](#_heading=h.1x0gk37) **24**

[**7. Appendix**](#_heading=h.4h042r0) **24**

# 

# Introduction

A student record management software is a tool that tracks and records the details of students in institutes including their exam performance and their personal information which can be easily searched and retrieved later on. The software stores years of data online without hampering their functioning. The software helps to generate reports easily in required formats. The system can help users to find and retrieve student’s data over a few clicks. The is easy to use and paperless system.

## 1.1 Intended Audience

|  |  |
| --- | --- |
| BU Authority |  |
|  |  |

## 1.2 Acronyms/Abbreviations

|  |  |
| --- | --- |
| HLD | High Level Design |
| LLD | Low-Level Design |
| SRMS | Student Record Management System |

## 1.3 Project Purpose

This project is aimed to developing a student record system. It is a system meant to maintain all the details regarding student details. We have developed it using C++. It will also reduce the paperwork and reduce the time.

## 1.4 Key Project Objectives

The main objective is to create a the tool that automates data tracking and record-keeping related to various academic aspects and their personal information.

**1.5 Project Scope and Limitation**

The scope of the student record management system is an automation tool that organizes, safeguards and maintained educational databases. But there are also many other thing and its precise definition is depend on who use it.

To academic institutions it’s an instrument for meeting the needs of students.

To administrators, it’s a means of bypassing the manual work susceptible to many

human errors.

## 1.6 Functional Overview

## These functionalities have been implemented in the program:

* ADMIN: admin can login to the application with their login credentials and then he can perform the following functionalities
* ADD STUDENT RECORD: Admin can add the student details like name, roll no, standard, email id, contact number and address
* MODIFY STUDENT RECORD: Admin can modify their details like name, roll no, standard, email id, contact number and address
* DISPLAY STUDENT RECORD: Admin can view their personal details
* DELETE STUDENT RECORD: Admin can delete the entire student record based on the roll no they entered
* SEARCH STUDENT RECORD: Admin can search for the student record based on the roll no they entered.

## 1.7 Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

Operating environment for implementing UDP are:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu/C

## 1.8 Risks

No Risk(As it is for educational purpose).

# Design Overview

1. START

This is the start block which indicates the start of the program.

1. ADMIN

Admin has to login by providing the required credentials.

Admin manages all the operations such as add, delete, modify, search and display the students record.

1. LOGIN

Admin has to login by providing the required credentials.

1. ADD RECORD

Admin can add the students details such as student’s name, roll no, course, email id, contact no and address.

1. MODIFY RECORD

Admin can modify students record.

1. DELETE RECORD

Admin can delete students record.

1. SEARCH RECORD

Admin can search students record by providing students roll no.

1. DISPLAY RECORD

Admin can view all students record.

## 2.1 Design Objectives

Admin can login to the system by providing login credentials.

Admin can add student details, delete student details, modify student details, search

student details and also, he can display all student details.

Admin can manage all student records.

**2.1.1 Recommended Architecture**

UML Architecture

## 2.2 Architectural Strategies

* Header files
* Structures
* Macros

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<string.h>

### 

### 2.2.3 Creation of New Common Services/Utilities

NA

**2.2.4 User Interface Paradigms**

Command Line Interface: Terminal

**2.2.5 Housekeeping and Maintenance**

NA

**2.2.6 System Interface Paradigms**

Command Line Interface: Terminal

**2.2.7 Error Detection / Exceptional Handling**

Admin needs to enter the required login credentials else an error will occur and admin has to re-login.

Phone number should contain the 10digits.

Students email should contain one ‘@’ and ‘.’ Symbol.

Students roll number should be with a length 6.

**2.2.8** **Memory Management**

NA

* + 1. **Performance**

NA

**2.2.10** **Security**

For security purposes the system asks for login credentials from admin.

* + 1. **Concurrency and Synchronization**

NA

# System Architecture

**3.1 ER Diagram**

***Diagram

Description automatically generated***

## 

## 3.2 Flowchart

Diagram

Description automatically generated

## 3.3 System Use-Cases

Diagram

Description automatically generated

**3.4 Sequence Diagram**

**Diagram

Description automatically generated**

**3.5 Subsystem Architecture**

NA

**3.6 System Interfaces**

NA

**3.6.1 Internal Interfaces**

NA

**3.6.2 External Interfaces**

NA

# Detailed System Design

# The code starts by declaring admin login function as admin can login to the system by providing required login credentials like username and password. And admin can perform the operations such as add, delete, modify, search and display the student’s detail.

## 4.1 Key Entities

* Admin login
* Student details
* add record, delete record, modify record, search record and display record

## 4.2 Detailed-Level Database Design

NA

### 4.2.1 Data Mapping Information

NA

### 4.2.2 Data Conversion

NA

## 4.3 Archival and retention requirements

NA

## Disaster and Failure Recovery

We don’t have any control over the system. In case of failure, source code is safe.

Use of Git.

## 4.5 Business Process workflow

NA

## 4.6 Business Process Modeling and Management (as applicable)

NA

## 4.7 Business Logic

NA

## 4.8 Variables

NA

## 4.9 Activity / Class Diagrams (as applicable)

NA

**Pseudocode:**

**1. Admin Login and Menu Function:**

Admin has to login by providing the required credentials. Admin manages all the operations such as add, delete, modify, search and display the students record.

Declare admin\_login

Declare username[20], password[20]

Declare choice

Enter admin username

Enter admin password

Open file in read mode

//Compare Entered username with username in file and also Entered password with password(which is stored)in file

if(username(entered) == username(in file))

{

if(password(entered) == password(in file))

{

Print “Logged in Successfully”

//Admin has to view the menu

Select the choice

1. Add Students details
2. Delete Students details
3. Modify Students details
4. Search Students details
5. Display Students details

}

}

else

print "password is wrong or username is wrong"

admin\_login

**2. Add Record Function:**

Admin can add the students details such as student’s name, roll no, course, email id, contact no and address.

Declare add\_record function

Declare the ofstream outfile

Open outfile in append and binary mode

if(outfile.fail())

print "File is not present"

Enter the student details as Name, Roll No, Email Id, Phone No, Age, Gender, Address, Stream, Subject Marks

total marks = (physics + chemistry + maths + biology + computers + electronics + english)

Output Total Marks

percentage = (Total Marks \* 100)/500

Output Percentage

Write the data to outfile.

outfile.write()

Close the file

outfile.close()

Print "The File is successfully saved"

**3. Delete Record Function:**

Admin can delete students record.

Declare delete\_record function

Declare the ifstream infile

Open infile in append and binary mode

if(!infile)

print "File is not present"

Enter the Roll No to delete the data

Open outfile in append and binary mode

Write the data to the outfile

while(!file.eof())

if(Roll\_No != Roll\_No)

write data to outfile

else

increment found

print "Successfully Delete Data"

write data to outfile

outfile.write()

if(found == 0)

print "Student Roll\_No not found"

Close the infile

infile.close()

Close the outfile

outfile.close()

remove the outfile text file and rename it

**4. Modify Record Function:**

Admin can modify students record.

Declare modify\_\_student\_record function

Declare the fstream file and file1

Open file in read mode

if(!file)

print "no data is present"

close the file.

else

Enter the Roll No to modify the data

Open file1 in write mode

Write the data to the file1

while(!file.eof())

if(RollNo != Roll\_No)

write data to file1

else

modify the details

increment found

if(found == 0)

print "Student Roll\_No not found to modify"

Close the file1

file1.close()

Close the file

file.close()

**5. Search Record Function:**

Admin can search students record by providing students roll no.

Declare search\_\_student\_record function

Declare the fstream file

Open the file in read mode

if(!file)

print "no data is present"

else

Enter the Roll No to search the data

while(!file.eof())

if(RollNo == Roll\_No)

print the student data from file

increment found

if(found == 0)

print "Student Roll\_No not found"

Close the file

file.close()

**6. Display Record Function:**

Admin can view all the students record.

Declare display\_students\_details

Open file in read mode

if(file == empty)

print"No Data is present."

else

open file

print data from file

Close file

**7. Data Validation**

**7.1 Phone Number Validation Function:**

The phone number length should be 10. If phone number length is 10, it should return True else false.

Declare int phone\_number\_validation(int phone\_number)

if(phone\_number.length() == 10)

return true

else

return false

**7.2 Roll No Validation Function:**

The Roll No length should be 6. If Roll No length is 6, it should return True else false.

Declare int roll\_no\_validation(int roll\_no)

if(roll\_no.length() == 6)

return true

else

return false

**7.3 Email Id Validation Function:**

The Email Id length should be 30 and it should contain @ and . symbol. If it is present, it is true else false.

Declare char email\_id\_validation(char email)

Declare i

Declare and initialise int length

length = email.length()

Loop for i=0 and i<length and i++

if(length <= 30)

if(email[i] == '@' && email[i] == '.')

return true

else

return false

**7.4 Course Validation Function:**

If the Course is PCMB or PCMC or PCME, it should return true else false.

Declare char course\_validation(char course)

if(course == "PCMB" || course == "PCMC" || course == "PCME")

return true

else

return false

## 4.10 Data Migration

NA

### 4.10.1 Architectural Representation

NA

### 4.10.2 Architectural Goals and Constraints

This project is a tool that is meant to maintain the student records.

### 4.10.3 Logical View

NA

### 4.10.4 Architecturally Significant Design Packages

NA

### 4.10.5 Data model

NA

**Legacy system data model**

**Proposed system data model**

**Interface data model**

**4.10.6 Deployment View**

NA

# Environment Description

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for the various programming languages. It is mainly used to compile the C and C++ programs.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GITHUB: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

NA

## 5.2 Language Support

NA

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 

## 5.4 Server-Side Requirements

Linux, Ubuntu

### 5.4.1 Deployment Considerations

NA

### 

### 5.4.2 Application Server Disk Space

NA

### 5.4.3 Database Server Disk Space

NA

### 5.4.4 Integration Requirements

NA

### 5.4.5 Jobs

NA

### 5.4.6 Network

NA

### 5.4.7 Others

NA

## 5.5 Configuration

NA

### 5.5.1 Operating System

Linux desktop editions with 8 GB RAM. A GUI-based LINUX system must be

used

### 5.5.2 Database

NA

### 5.5.3 Network

NA

### 5.5.4 Desktop

* CPU: Intel i3/i5/i7 generation 3 and later
* RAM: 4GB or greater - For optimal performance, 6GB or 8GB are recommended if you will be running multiple browser tabs and/or multiple applications at the same time
* Internal memory:476 GB SSD/HDD.

# References

* <https://www.scaler.com/topics/student-record-management-system-cpp/>
* <https://coderspacket.com/student-record-management-system>
* <https://www.geeksforgeeks.org/student-data-management-c/>
* <https://www.javatpoint.com/student-data-management-in-cpp>