23578

MINI PROJECT REPORT ON

# CREDIT CALCULATIONS

SUBMITTED TO THE DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

ALL INDIA SHRI SHIVAJI MEMORIAL SOCIETY'S INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

# SUBMITTED BY

**20 Kimaya Chavan**

**16 Rutuja Bidarkar**

**07 Pooja Balpande**

UNDER THR GUIDANCE OF

**Ms. Pradnya Bormane**

# Department of Artificial Intelligence and Data Science

# BY

ALL INDIA SHRI SHIVAJI MEMORIAL SOCIETY'S INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

ACADEMIC YEAR: 2024-25

# Department of Artificial Intelligence and Data Science

CERTIFICATE

# This is to certify that the mini project report titled

# “ CREDIT CALCULATIONS ”

Submitted by

A-07 23610008 Pooja Balpande

A-20 23610021 Kimaya Chavan

A-15 23610016 Rutuja Bidarkar

is a bonafide student of this institute and the work has been carried out by them under the supervision of **Ms. Pradnya Bormane** and is approved for the partial fulfillment of the Department of Artificial Intelligence and Data Science, AISSMS IOIT, Pune.

Ms. Pradnya Bormane Dr. R. A. Jamadar

Guide Head of the Department

(Department of AI & DS) (Department of AI & DS)

Place:

Date:

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. no.** | **Topic Name** | | **Page No** |
| 1. | Abstract | | 5 |
| 2. | Acknowledgement | | 6 |
| 3. | Introduction | | 7 |
| 4. | Problem Statement | | 8 |
| 5. | Proposed System | | 9 |
|  | 5.1 | System Workflow | 11 |
|  | 5.2 | Workflow Diagram | 12 |
| 6. | System Requirements | | 13 |
|  | 6.1 | Software Requirements | 13 |
|  | 6.2 | Hardware Requirements | 14 |
| 7. | UML Diagrams | | 15 |
| 8. | Implementation | | 16 |
| 9. | Results / Output | | 21 |
| 10. | Testing | | 22 |
| 11. | Advantages and Disadvantages | | 23 |
| 12. | Future Scope | | 24 |
| 13. | Project Plan | | 25 |
| 14. | Conclusion | | 26 |
| 14. | References | | 27 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Topic Name** | **Page No.** |
| 1. | PROPOSD SYSTEM DIGRAM | 10 |
| 2. | SYSTEM WORKFLOW | 12 |
| 3. | WORKFLOW DIAGRAM | 14 |
| 4. | UML DIAGRAMS | 15 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Topic Name** | **Page No.** |
| 1. | TEST PLAN | 22 |
| 2. | PROJECT PLAN | 25 |

**ABSTRACT**

This project aims to develop a system for managing and calculating academic credits for college students, ensuring accurate tracking of their progress toward graduation. The system will automate the process of calculating credits based on course completion, academic performance, and program requirements, replacing the traditional manual methods that are prone to errors and inefficiencies. The primary objectives of this project include creating a user-friendly interface for students and administrators, establishing a reliable database to store course information and credit records, and implementing a formula that calculates cumulative GPA and total credits earned. The system will also offer functionalities for managing elective courses, credit transfers, and prerequisites, while allowing students to track their progress in real-time.

This project will leverage programming languages like Python or Java for backend processing, with a potential frontend interface using HTML/CSS/JavaScript for ease of access. The final product aims to improve the accuracy and efficiency of credit calculations, enabling students and educational institutions to better manage academic records, make informed decisions about course enrollment, and ensure that students meet graduation requirements on time.

# ACKNOWLEDGEMENT

We had a great time working on this project and learned new skills as a result of it. However, it would not have been possible without the tremendous support and cooperation of many individuals. Apart from the e orts of the project members, the success of every project is heavily reliant on the support and guidance of many others. We would like to appreciate each and every one of them from the bottom of our hearts. Our guide, **Ms. Pradnya Bormane**, deserves our heartfelt gratitude. We can't thank her enough for all of her help and support. Her assistance and continual guidance were really beneficial in resolving our issues. We would also like to thank **HOD Dr. R. A. Jamadar** for his amazing assistance during the mini-project. We would like to take this opportunity to thank everyone who assisted and guided us during the project.

Name of students

Kimaya Chavan

Rutuja Bidarkar

Pooja Balpande

# INTRODUCTION

# 

# The college credit system plays a crucial role in tracking students' academic progress, determining their eligibility for graduation, and ensuring that they meet the requirements of their chosen programs. Each course completed by a student typically carries a specific number of credits, which represent the amount of time and effort required to complete the course. These credits accumulate over the duration of a student’s academic journey, and once the required number of credits is achieved, the student becomes eligible to graduate. However, manually calculating and managing credits can be time-consuming and prone to errors, particularly in larger institutions where students enroll in various programs, take elective courses, or transfer credits from other institutions. The complexity increases further when considering the impact of academic performance on cumulative GPA, which is essential for meeting program requirements or qualifying for honors.

# The "Credit Calculations" project aims to simplify and automate this process, providing a reliable system that accurately tracks credits and calculates students' progress in real-time. By implementing this system, educational institutions can ensure that both students and administrators have a clear, up-to-date understanding of where a student stands in relation to their academic goals, enabling smoother transitions and timely graduation. This project will not only save time and reduce errors but also allow for better academic planning and management. By building a credit calculation system, institutions can help students better manage their academic pathways, reducing the risk of errors and improving overall educational outcomes. It also enables students to make informed decisions about their course selections and academic trajectory, ensuring a smoother and more predictable academic journey.

# 

**PROBLEM STATEMENT**

**Design a C++ program that calculates credits.**

**The program should** Capture user input for multiple courses, including course name,

course code, credit hours,and grade received.Calculate the total credits earned and

the cumulative GPA based on the grades and credit hours of each course. Display the

detailed course information, credit hours, and GPA on the console.Save all the captured

course details and computed results (total credits and GPA) to a text file in a readable format.

# PROPOSED SYSTEM

The proposed college credit calculation system is designed to automate the process of

tracking and calculating student credits, academic progress, and GPA. By creating a user-friendly

interface for both students and administrators, this system aims to simplify the management of

academic records, reduce errors in credit calculations, and ensure students are meeting their

program requirements in a timely manner.

KEY COMPONENTS :

1. User Input Capture :

The system will capture essential information from the student, including:

* **Student ID and Name**: This ensures the system is personalized and allows tracking for each

student individually.

* **Course Details**: For each course a student is enrolled in, the system will capture:
  + Course Name
  + Course Code
  + Credit Hours
  + Grade Received

The interface will allow students to input details for multiple courses at a time.

1. Credit Calculation :

The system will automate the calculation of total credits earned by summing the credit hours

of each course that has been successfully completed.

This is critical for students to know how many credits they have accumulated toward graduation.

1. GPA Calculation :

The system will calculate the Grade Point Average (GPA) using the standard formula, which

considers both the grade received in each course and its corresponding credit hours.

1. Progress Tracking :

* The system will track the student's progress toward degree completion by calculating:
* Total credits earned against the required credits for graduation.
* Remaining credits needed, based on the student’s program or major requirements.
* Any outstanding prerequisites for higher-level courses, if applicable.

This functionality provides students with a clear picture of their academic standing.

1. Display Results :

After the calculations are complete, the system will display the following results on the

console:

* List of courses taken, along with credit hours and grades.
* Total number of credits earned.
* Cumulative GPA.
* Any remaining credits required for graduation.

1. Data Storage and Retrieval :

The system will store each student’s data in a local database or file system. This will allow

administrators to retrieve, update, or review student records at any time. It can also help in

generating academic reports for a larger number of students.

1. Error Handling and Validation :

The system will include error handling mechanisms to:

* Ensure valid data entry (e.g., restricting grades to A, B, C, etc.).
* Prevent duplicate course entries.
* Ensure credit hours fall within expected ranges for courses (e.g., typically 1 to 4 credits).
* Handle missing or incomplete data entries with appropriate error messages.

# SYSTEM WORKFLOW

# The workflow of the system involves a series of steps to capture student data, process it for credit and GPA calculations, and display the results. This workflow is designed to be intuitive for both students and administrators, ensuring that academic records are accurately tracked and managed.

# System Initialization: Load input screen, setup database.

# User Input: Enter student info, course details (credit hours, grades).

# Credit Calculation: Calculate total credits based on completed courses.

# GPA Calculation: Compute cumulative GPA using grades and credit hours.

# Display Results: Show total credits, GPA, and progress toward graduation.

# Save to File: Save all details and results to a text file.

# Error Handling: Manage invalid inputs and missing data with alerts.

# Exit or Restart: User can exit the program or restart for new entries.

# This workflow ensures that students and administrators can easily manage and calculate academic progress efficiently, while also maintaining accuracy in the records.

**WORKFLOW DIAGRAM**

Start System

Load Input Screen

**WORKFLOW DIAGRAM OF CREDIT CALCULATIONS**

Exit Program

Save Data to File

Display Results

Calculate cumulative GPA

Calculate total credits earned

Store Data & move on to next course entry

Validate Inputs

Enter Course Details

Enter Student Details

# SYSTEM REQUIREMENTS

**Software Requirements:**

Operating System:

Windows 7 or later, macOS 10.12 or later, Linux (Ubuntu 18.04 or later)

Compiler:

C++ Compiler: Any modern C++ compiler (e.g., GCC, MinGW, Microsoft Visual C++)

IDE (Optional): Code::Blocks, Visual Studio, or Eclipse for development.

Libraries:

Standard C++ library (no external libraries required for this basic system).

Text Editor (for file viewing):

Any text editor like Notepad, Sublime Text, VS Code, or built-in text editors in operating

systems (for viewing and editing output files).

**Hardware Requirements:**

Processor:

Minimum: Dual-core processor (Intel Core i3 or equivalent)

Recommended: Quad-core processor (Intel Core i5/i7 or equivalent)

RAM:

Minimum: 4 GB

Recommended: 8 GB or more

Storage:

Minimum: 100 MB of free storage space for application and data files.

Recommended: SSD with at least 1 GB of free storage for faster file I/O operations.

Display:

Minimum: 1024x768 resolution

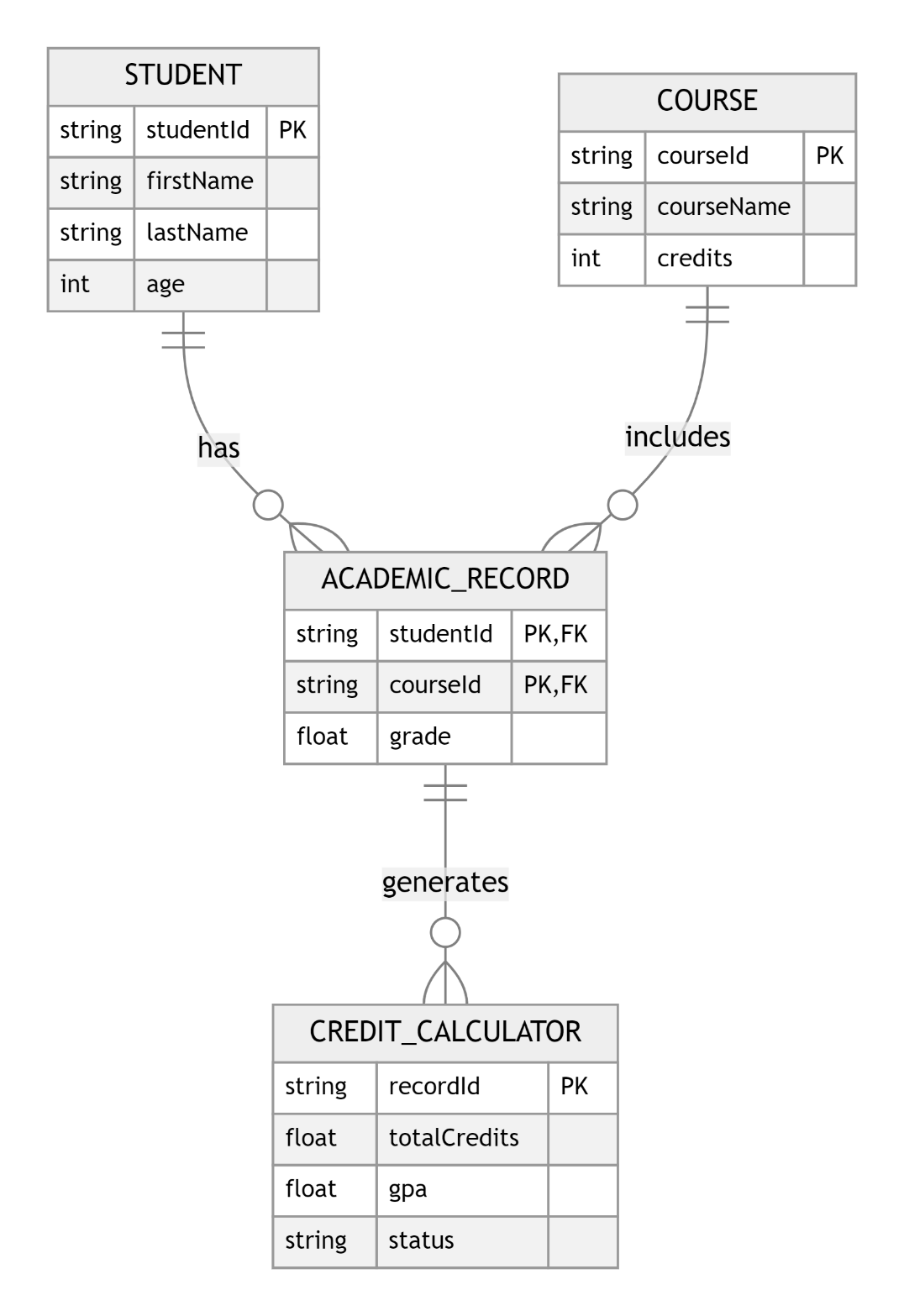
Recommended: 1920x1080 (Full HD) or higher

Keyboard and Mouse: Required for input and navigation.

Operating System:

Windows 7 or later, macOS 10.12 or later, Linux (Ubuntu or Fedora)

UML DIAGRAM



# Diagram 1: ER Diagram(UML)

**IMPLEMENTATION**

**SOURCE CODE**

#include <iostream>

#include <string>

#include <stdexcept>

using namespace std;

class Subject

{

private:

string name;

double marks;

double credits;

public:

Subject() : name(""), marks(0.0), credits(0.0) {}

Subject(const string &n, double m, double c) : name(n), marks(m), credits(c) {}

double getCredits() const { return credits; }

double getGradePoints() const

{

if (marks >= 90)

return 10;

else if (marks >= 80)

return 9;

else if (marks >= 70)

return 8;

else if (marks >= 60)

return 7;

else if (marks >= 50)

return 6;

else if (marks >= 40)

return 5;

else

return 0;

}

string getName() const { return name; }

double getMarks() const { return marks; }

void setValues(const string &n, double m, double c)

{

name = n;

marks = m;

credits = c;

}

};

class Student

{

private:

Subject \*subjects;

int totalSubjects;

public:

Student(int numSubjects) : totalSubjects(numSubjects)

{

if (numSubjects <= 0)

throw invalid\_argument("Number of subjects must be positive.");

subjects = new Subject[numSubjects];

}

~Student() { delete[] subjects; }

void addSubject(int index, const string &name, double marks, double credits)

{

if (index < 0 || index >= totalSubjects)

throw out\_of\_range("Index out of range.");

subjects[index].setValues(name, marks, credits);

}

double calculateGPA() const

{

double totalCredits = 0;

double totalGradePoints = 0;

for (int i = 0; i < totalSubjects; ++i)

{

totalCredits += subjects[i].getCredits();

totalGradePoints += subjects[i].getGradePoints() \* subjects[i].getCredits();

}

return totalCredits > 0 ? totalGradePoints / totalCredits : 0;

}

int getTotalCredits() const

{

int totalCredits = 0;

for (int i = 0; i < totalSubjects; ++i)

{

totalCredits += static\_cast<int>(subjects[i].getCredits());

}

return totalCredits;

}

void displaySubjects() const

{

cout << "\n| Subject Name | Marks | Credits | Grade Points |\n";

cout << "|-------------------|-------|---------|--------------|\n";

for (int i = 0; i < totalSubjects; ++i)

{

cout << "| " << subjects[i].getName()

<< string(18 - subjects[i].getName().length(), ' ')

<< "| " << subjects[i].getMarks()

<< string(6 - to\_string((int)subjects[i].getMarks()).length(), ' ')

<< "| " << subjects[i].getCredits()

<< string(7 - to\_string((int)subjects[i].getCredits()).length(), ' ')

<< "| " << subjects[i].getGradePoints()

<< string(12 - to\_string((int)subjects[i].getGradePoints()).length(), ' ')

<< "|\n";

}

cout << "|-------------------|-------|---------|--------------|\n";

}

};

double getInput(const string &prompt, double minValue, double maxValue)

{

double value;

while (true)

{

cout << prompt;

cin >> value;

if (!cin.fail() && value >= minValue && value <= maxValue)

{

cin.ignore(10000, '\n');

return value;

}

else

{

cin.clear();

cin.ignore(10000, '\n');

cout << "Invalid input. Please enter a value between " << minValue << " and " << maxValue

<< ".\n";

}

}

}

string getStringInput(const string &prompt)

{

string value;

cout << prompt;

cin.ignore();

getline(cin, value);

return value;

}

int main()

{

try

{

int totalSubjects;

cout << "Enter the total number of subjects: ";

cin >> totalSubjects;

if (cin.fail() || totalSubjects <= 0)

{

throw invalid\_argument("Invalid input. Please enter a positive integer for the number of subjects.");

}

Student student(totalSubjects);

cout << "\n=====================\n";

cout << " Subject Details\n";

cout << "=====================\n";

for (int i = 0; i < totalSubjects; ++i)

{

string subjectName = getStringInput("Enter name for subject " + to\_string(i + 1) + ": ");

double marks = getInput("Enter marks for subject " + to\_string(i + 1) + " (0-100): ", 0.0, 100.0);

double credits = getInput("Enter credits for subject " + to\_string(i + 1) + ": ", 1.0, 10.0);

student.addSubject(i, subjectName, marks, credits);

}

student.displaySubjects();

double GPA = student.calculateGPA();

int totalCredits = student.getTotalCredits();

cout << "\n=============================\n";

cout << " Summary of Results\n";

cout << "=============================\n";

cout << "Total Credits: " << totalCredits << endl;

cout << "GPA out of 10: " << GPA << endl;

cout << "=============================\n";

}

catch (const exception &e)

{

cerr << "Error: " << e.what() << endl;

}

return 0;

}

# RESULTS / OUTPUT

[?2004l

Enter the total number of subjects: 3

=====================

Subject Details

=====================

Enter name for subject 1: DSA

Enter marks for subject 1 (0-100): 89

Enter credits for subject 1: 3

Enter name for subject 2: OOP

Enter marks for subject 2 (0-100): 84

Enter credits for subject 2: 3

Enter name for subject 3: SE

Enter marks for subject 3 (0-100): 78

Enter credits for subject 3: 2

| Subject Name | Marks | Credits | Grade Points |

|-------------------|-------|---------|--------------|

| DSA | 89 | 3 | 9 |

| OP | 84 | 3 | 9 |

| SE | 78 | 2 | 8 |

|-------------------|-------|---------|--------------|

=============================

Summary of Results

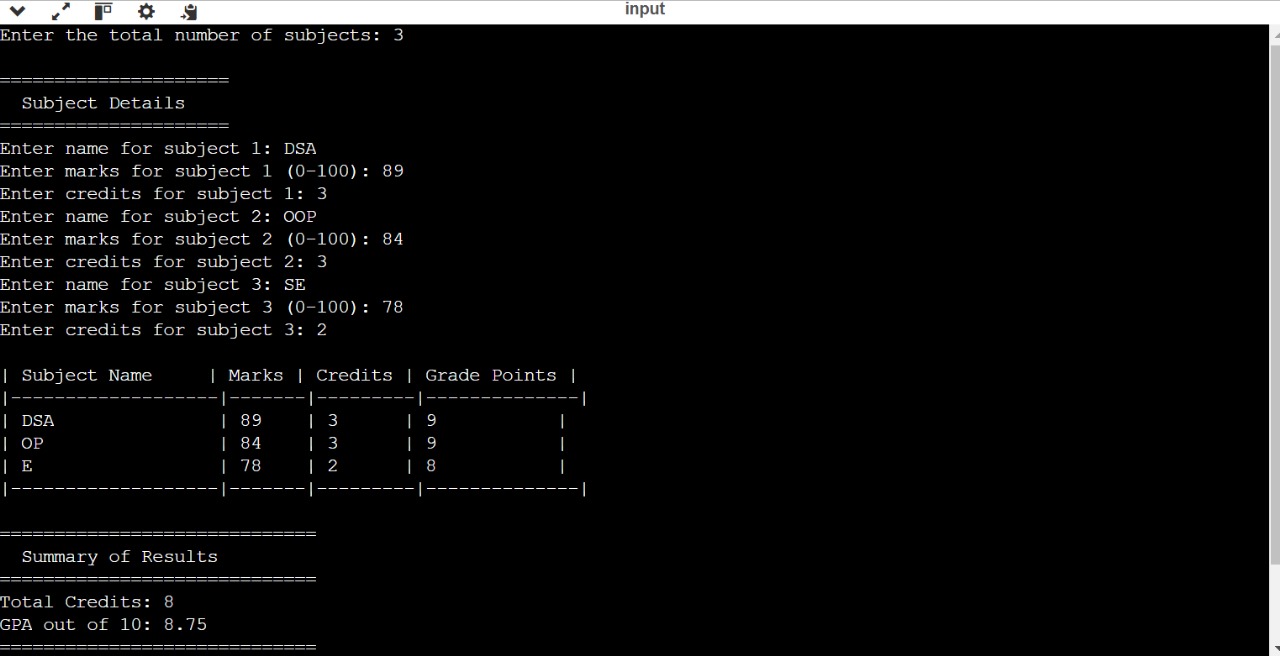
=============================

Total Credits: 8

GPA out of 10: 8.75

=============================

[?2004h



**TESTING**

***Table 1: Test Plan***

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Type of Test** | **Description** | **Hardware/Software Components** |
| 1 | Requirement Testing | Ensures that all specified  requirements of the system are correctly implemented. | Validate that the system meets the functional and non-functional requirements. |
| 2 | Unit testing | Focuses on individual components or functions of the system to ensure they work as expected in isolation. | Test specific functions, such as getInput(), displayDetails(), and saveToFile(), independently. |
|  | Integration | Tests the interactions between different modules of the system to ensure they work together properly. | Ensure that when functions like getInput(), displayDetails(), and saveToFile() work together, the system behaves correctly. |
| 4 | Performance | Measures the system’s efficiency in terms of speed, memory usage, and response time. | Ensure that the system handles multiple items and large amounts of data without significant performance degradation. |
| 5 | System Testing | Tests the complete system as a whole, ensuring that it functions according to the requirements. | Verify that all components of the system work together seamlessly. |
| 6 | GUI Testing | Tests the graphical user interface (if any) to ensure it is user-friendly and all elements function as expected. | Validate that input fields, buttons, and text display properly if a GUI is present. |

# ADVANTAGES AND DISADVANTAGES

**Advantages :**

* **Accuracy:** Precise GPA and credit calculations, reducing human error.
* **Time-Saving:** Quick processing of multiple courses and grades.
* **Progress Tracking:** Easy monitoring of academic progress.
* **Standardization:** Uniform calculations for consistency.
* **Record Management:** Efficient storage and retrieval of data.
* **Error Detection:** Identifies incorrect inputs for correction.
* **Transparency:** Clear and accessible results for students.
* **Flexibility:** Adaptable to different grading systems**.**

**Disadvantages :**

* **Tech Dependence:** System failures can disrupt operations.
* **Special Cases:** May struggle with unique academic scenarios.
* **Learning Curve:** Initial setup and user adaptation required.
* **Data Entry Errors:** Inaccurate inputs can affect results.
* **Maintenance:** Requires regular updates and support.
* **No Human Judgment:** Lacks ability to handle special circumstances.
* **Data Security:** Risk of privacy breaches and cyber threats.
* **Customization Complexity:** Can be challenging to tailor to specific policies.

**FUTURE SCOPE**

* + - **Integration with Learning Management Systems (LMS)**:

The system can be integrated with popular LMS platforms (e.g., Canvas, Moodle) to

automatically sync student data, grades, and course information, reducing manual

data entry and enhancing accuracy.

* + - **Web and Mobile Applications**:

Developing a web-based or mobile application version would allow students and

faculty to access credit and GPA calculations anytime, anywhere, providing greater

accessibility and ease of use.

* + - **Advanced Analytics and Reporting**:

The system could offer advanced analytics, such as tracking academic

performance trends, predicting GPA outcomes based on future course enrollments,

and providing detailed performance reports to students and faculty.

* + - **AI and Machine Learning Integration**:

Implementing AI algorithms could help analyze student performance patterns,

predict risks of academic failure, and suggest personalized course

recommendations or improvements.

* + - **Support for Diverse Grading Systems**:

The system can be expanded to support multiple grading scales

(e.g., weighted GPAs, international grading systems), allowing for greater flexibility

across different educational institutions.

* + - **Automated Alerts and Notifications**:

Future versions could include automated notifications for students, informing

them of critical academic milestones, such as reaching credit requirements for

graduation or academic warnings when GPA falls below a threshold.

* + - **Cloud Storage and Collaboration**:

Cloud-based storage can allow students and faculty to collaborate, share academic

data, and ensure that data is accessible and secure, with real-time updates.

* + - **Blockchain for Secure Transcripts**:

Integrating blockchain technology could help secure student records, ensuring

tamper-proof academic transcripts and reducing fraud in academic qualifications.

**PROJECT PLAN**

***Table 2: Projectt Plan***

|  |  |  |  |
| --- | --- | --- | --- |
| **SR. No.** | **Activity** | **Start Date** | **END Date** |
| 1 | Project Initiation | 2024-09-01 | 2024-09-07 |
| 2 | Requirement Analysis | 2024-09-08 | 2024-09-14 |
| 3 | System Design | 2024-09-15 | 2024-09-21 |
| 4 | Development | 2024-09-22 | 2024-09-28 |
| 5 | Testing | 2024-09-29 | 2024-10-05 |
| 6 | Deployment and Maintenance | 2024-10-06 | 2024-10-12 |

# CONCLUSION

# The college credit calculation system is a vital tool for simplifying academic management by automating the calculation of credits and GPA. It ensures accuracy, saves time, and provides transparency for both students and administrators, enabling better tracking of academic progress. With features like error detection and standardized calculations, the system enhances fairness and consistency across institutions. However, the reliance on technology, potential data entry errors, and the need for regular maintenance highlight areas that require careful management.

# Looking forward, the system can evolve with advanced features such as integration with LMS, mobile access, AI-driven analytics, and blockchain for secure transcripts, further enhancing its functionality. Overall, the system plays an essential role in supporting academic decision-making and ensuring that students have a clear understanding of their educational trajectory.

# REFERENCES

1. Goodrich, M. T., & Tamassia, R. (2014). *Data structures and algorithms in C++*

(2nd ed.) Wiley.

2. Thelin, J. R. (2011). *A history of American higher education*. Johns Hopkins

University Press.

3. DeAngelis, C. A., & McMillan, J. (2021). The impact of academic credit systems

on student learning. *Journal of Educational Psychology, 113*(4),

609-622. <https://doi.org/10.1037/edu0000465>

4. Sweeney, P. (2020). Automated systems for GPA and credit calculation in

higher education. *International Journal of Educational Technology,*

*16*(2), 45-60. <https://doi.org/10.1007/s41239-020-00218-9>