***THE PROJECT REPORT OF***

**NAME: - SUBODH KUMAR**

**Registration Number: - 1475194**

**ON**

STUDENT MANAGEMENT SYSTEM



In partial fulfillment of the requirement FOR the award of DIPLOMA of

A LEVEL PROJECT

TO

NIELIT, INDIA

.

**DECLARATION**

I hereby declare that this project work titled **“STUDENT MANAGEMENT SYSTEM”** is my original work and no part of it has been submitted for any other degree purpose or published in any other from till date.

**NAME: - SUBODH KUMAR**

**Registration Number: - 1475194**

**ACKNOWLEDGEMENT**

I hereby take this opportunity to express our sincere gratitude to all the people who have contributed to this project with their knowledge and experience.

I consider myself lucky to work under the guidance of **Mr. Vikas Sharma**. His rare combination of experience in different fields, humbleness, untiring patience and meticulous care has helped this work to acquire the shape of what it is today. This project would have been difficult to complete without his advice, ideas and zeal. He gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. He taught me how to question thoughts and express ideas. I owe an immense intellectual debt to him. It was always a new and unique experience while working with him.

Finally I would like to express my deep appreciation to my family and friends who have been a constant source of inspiration and spiritual support in all aspects of my life. I am eternally grateful to them for always encouraging and being with me wherever and whenever I needed them during the time period of the project.

**NAME: - SUBODH KUMAR**

**Registration Number: - 1475194**

Table of Contents

[1. Introduction: 6](#_Toc165900039)

[2. Objective of Student Management System: 7](#_Toc165900040)

[3. Drawback of Manual System: 8](#_Toc165900041)

[4. Need of New System: 9](#_Toc165900042)

[5. Proposed System: 11](#_Toc165900043)

[6. Front-End: 13](#_Toc165900044)

[7. Back-End: 13](#_Toc165900045)

[Tools / Platform, Hardware and Software Requirement Specification 14](#_Toc165900046)

[HARDWARE 14](#_Toc165900047)

[SOFTWARE 14](#_Toc165900048)

[Phase 1: System Study 15](#_Toc165900049)

[Preliminary Investigation: - 16](#_Toc165900050)

[SYSTEM DEVELOPMENT LIFE CYCLE 17](#_Toc165900051)

[DEFINING A SYSTEM: - 17](#_Toc165900052)

[SYSTEM LIFE CYCLE: - 17](#_Toc165900053)

[Phase 2: System Analysis 20](#_Toc165900054)

[Importance of Computerized: - 21](#_Toc165900055)

[Transaction System: 21](#_Toc165900056)

[Decision Support System: 21](#_Toc165900057)

[Principles of System Analysis 22](#_Toc165900058)

[Principles: 22](#_Toc165900059)

[Complete Structure: 22](#_Toc165900060)

[Topic Understanding: 22](#_Toc165900061)

[Modular Break –Up of the System: 22](#_Toc165900062)

[System Design: - 23](#_Toc165900063)

[Data Modeling: - 26](#_Toc165900064)

[Data Dictionary (Definition): 26](#_Toc165900065)

[ER Diagram: 27](#_Toc165900066)

[The need for an ER diagram in my project includes: 27](#_Toc165900067)

[DFD: - 29](#_Toc165900068)

[The need for a DFD in my project includes 29](#_Toc165900069)

[Context level DFD (0 level) 30](#_Toc165900070)

[Level 1 DFD: 31](#_Toc165900071)

[Level 2 DFD: 32](#_Toc165900072)

[Class diagram: - 34](#_Toc165900073)

[The need for a Class Diagram in my project includes: - 34](#_Toc165900074)

[CLASS DIAGRAM: - 35](#_Toc165900075)

[Modules: 36](#_Toc165900076)

[**Login Module**: - 36](#_Toc165900077)

[**Course Module**: - 36](#_Toc165900078)

[**Student Module**: - 36](#_Toc165900079)

[**Result Module**: - 37](#_Toc165900080)

[**Report Module**: - 37](#_Toc165900081)

[Phase 3: System Planning 38](#_Toc165900082)

[System Planning: - 39](#_Toc165900083)

[Estimates of Effort 39](#_Toc165900084)

[Phase 4: Screen Shot 41](#_Toc165900085)

[Login window: 42](#_Toc165900086)

[Dashboard window: 43](#_Toc165900087)

[Course window: 44](#_Toc165900088)

[Student window: 45](#_Toc165900089)

[Result window: 46](#_Toc165900090)

[Report window: 47](#_Toc165900091)

[Registration window: 48](#_Toc165900092)

[Forget Password window: 49](#_Toc165900093)

[Phase 5: Source Code 50](#_Toc165900094)

[Loginpage 51](#_Toc165900095)

[Dashboard page: 56](#_Toc165900096)

[Course Page: 63](#_Toc165900097)

[Student page: 72](#_Toc165900098)

[Result Page: 80](#_Toc165900099)

[Report Page: 85](#_Toc165900100)

[Registration Page: 90](#_Toc165900101)

[Forget Password Page: 96](#_Toc165900102)

[Database Page: 99](#_Toc165900103)

[Code Efficiency: - 100](#_Toc165900104)

[Phase 6: System Testing 101](#_Toc165900105)

[Testing Phase: - 101](#_Toc165900106)

[Levels of testing 102](#_Toc165900107)

[Test Cases 106](#_Toc165900108)

[Why System Testing? 114](#_Toc165900109)

[PREPARE TEST: 114](#_Toc165900110)

[Verification and Validation (V&V) 116](#_Toc165900111)

[Phase: System Implementation 117](#_Toc165900112)

[System implementation maintenance and review: - 118](#_Toc165900113)

[Hardware Evaluation Factors: - 118](#_Toc165900114)

[Software Evaluation Factors: - 119](#_Toc165900115)

[Gantt Chart: - 122](#_Toc165900116)

[Scope of future application 124](#_Toc165900117)

[Bibliography 126](#_Toc165900118)

[Glossary 127](#_Toc165900119)

# 1. Introduction:

Welcome to the **Student Management System (SMS)**, a comprehensive and user-friendly platform designed to streamline the process of managing and accessing student academic records. In today's fast-paced educational environment, efficient management of student results is essential for educators, administrators, and students alike.

The **SMS** provides a centralized hub where educational institutions can effortlessly manage, track, and analyze student performance data. From recording grades and generating transcripts to monitoring student progress and identifying areas for improvement, our system empowers institutions to make informed decisions that enhance the academic experience for all stakeholders.

With intuitive interfaces and robust features, the **SMS** offers a seamless experience for administrators, teachers, and students. Administrators can efficiently manage student information, course details, and examination schedules, while teachers can easily input grades, provide feedback, and generate comprehensive reports. Meanwhile, students gain access to their academic records, allowing them to track their progress, identify strengths and weaknesses, and take ownership of their learning journey.

At **SMS**, we understand the importance of data security and privacy. Our system is equipped with advanced security measures to safeguard sensitive information and ensure compliance with regulatory standards. Institutions can trust that their data is protected and accessible only to authorized users.

Whether you're a small school, a large university, or anything in between, the **Student Management System** is customizable to meet your institution's unique needs. Our dedicated support team is committed to providing ongoing assistance and guidance to ensure a smooth implementation process and continued success.

Experience the power of efficient student result management with **SMS** and unlock new possibilities for academic excellence. Welcome to a smarter way of managing student results. Welcome to the Student Management System.

# 2. Objective of Student Management System:

The primary objective of the Student Management System (SMS) is to provide a centralized and efficient platform for managing, processing, and analyzing student academic records. The system aims to streamline the result management process and enhance the overall academic experience for students, educators, and administrators.

Key objectives of the SMS include:

1. **Efficiency**: Automate and streamline result management tasks such as recording grades, calculating GPA, and generating transcripts to save time and reduce manual errors.

2. **Accessibility**: Provide easy access to student academic records anytime, anywhere, allowing students, teachers, and administrators to view, update, and analyze data conveniently.

3. **Transparency**: Promote transparency in the result management process by providing clear and accurate information about student performance, grading criteria, and assessment outcomes.

4. **Accuracy**: Ensure the accuracy and integrity of student academic records through standardized data entry processes, validation checks, and secure storage mechanisms.

5. **Analytics**: Enable educators and administrators to analyze student performance data, identify trends, and make data-driven decisions to improve teaching and learning outcomes.

6. **Student Empowerment**: Empower students to take ownership of their learning journey by providing access to their academic records, allowing them to track their progress, set goals, and seek assistance when needed.

7. **Compliance**: Ensure compliance with regulatory requirements and institutional policies related to result management, data privacy, and security.

8. **Customization**: Offer flexibility and customization options to adapt the system to the unique needs and preferences of educational institutions, including support for different grading systems, course structures, and reporting formats.

By achieving these objectives, the Student Management System aims to optimize the result management process, enhance collaboration and communication among stakeholders, and ultimately contribute to the overall success and satisfaction of students, educators, and administrators.

# 3. Drawback of Manual System:

A significant drawback of a manual system in student management is the increased likelihood of errors and inconsistencies in data entry and processing.

In a manual system:

1. **Data Entry Errors**: Transcribing grades, personal information, and other details manually into records leaves room for typographical errors, omissions, or incorrect entries. Even with careful oversight, human error is inevitable and can lead to inaccuracies in student records.

2. **Calculation Mistakes**: Manual calculation of averages, totals, or other statistical measures increases the risk of mathematical errors. These errors can result in incorrect grade calculations, affecting students' academic standing and potentially leading to misunderstandings or disputes.

3. **Data Duplication and Inconsistency**: Without automated checks and validations, there's a higher chance of duplicate records or inconsistencies in data across different documents or systems. This can create confusion and make it challenging to maintain data integrity and ensure accuracy.

4. **Time-Consuming Processes**: Manual result management processes, such as entering grades, updating records, and generating reports, are time-consuming and resource-intensive. This can lead to delays in accessing updated information and increase administrative burden on staff.

5. **Limited Accessibility and Collaboration**: Physical records or documents in a manual system may not be easily accessible to all stakeholders, limiting collaboration and communication between teachers, students, and administrators. This can hinder timely feedback, intervention, and decision-making.

6. **Vulnerability to Loss or Damage**: Physical records are susceptible to loss, damage, or theft, especially in environments with inadequate storage or handling procedures. Losing important documents can disrupt result management processes and compromise data security.

Overall, the reliance on manual processes in student management systems increases the risk of errors, inefficiencies, and security vulnerabilities, negatively impacting the quality of education and administrative effectiveness. Transitioning to automated systems can mitigate these drawbacks and improve the accuracy, efficiency, and accessibility of result management processes.

# 4. Need of New System:

The need for a new system in student management arises from several challenges and shortcomings associated with existing manual or outdated systems. Here are some key reasons why a new system is necessary:

1. **Efficiency**: Manual management processes are time-consuming and labor-intensive, involving tasks such as data entry, calculation, and record-keeping. A new system can automate these processes, saving time and resources for educational institutions and enabling staff to focus on more strategic activities.

2. **Accuracy**: Manual systems are prone to errors and inconsistencies in data entry, calculation, and reporting. A new system with built-in validations, automated calculations, and error-checking mechanisms can ensure the accuracy and integrity of student records, minimizing the risk of mistakes and discrepancies.

3. **Accessibility**: In manual systems, accessing and updating student records may be cumbersome and restricted to certain individuals or physical locations. A new system can provide secure online access to student information for authorized users, facilitating collaboration, communication, and transparency among stakeholders.

4. **Data Analysis**: Manual systems often lack robust tools for analyzing student performance data and identifying trends or patterns. A new system can offer advanced analytics capabilities, allowing educators and administrators to gain insights into student progress, performance, and learning outcomes.

5. **Scalability**: As educational institutions grow or evolve, manual systems may struggle to accommodate increasing data volumes or changing requirements. A new system designed with scalability in mind can adapt to the needs of growing institutions, supporting larger student populations, additional courses, and diverse grading systems.

6. **Security**: Physical records in manual systems are vulnerable to loss, theft, or damage, compromising the confidentiality and privacy of student information. A new system with robust security features, such as encryption, access controls, and data backups, can safeguard sensitive data and ensure compliance with privacy regulations.

7. **User Experience**: Manual systems may lack user-friendly interfaces and intuitive workflows, leading to frustration and inefficiency for users. A new system can prioritize usability and user experience, offering intuitive interfaces, personalized dashboards, and streamlined workflows to enhance user satisfaction and productivity.

Overall, the need for a new system in student management is driven by the desire to overcome the limitations of existing systems, improve operational efficiency, enhance data accuracy and security, and provide a better experience for all stakeholders involved in the educational process.

# 5. Proposed System:

The proposed system in student management aims to revolutionize the way academic records are managed, ensuring efficiency, accuracy, and accessibility for all stakeholders. Here's an outline of the proposed system:

1. **Automated Result Entry**: Implement a system for automated result entry, where grades, assessments, and other academic data are directly inputted into the system through electronic means. This reduces the reliance on manual data entry, minimizing errors and saving time.

2. **Integrated Grading System**: Develop an integrated grading system that supports various grading scales, calculation methods, and assessment criteria. This system will allow educators to easily input grades, calculate averages, and generate comprehensive reports based on standardized grading practices.

3. **Online Student Portals**: Create online portals for students to access their academic records, including grades, transcripts, and course schedules. These portals will provide real-time updates on student progress, allowing students to track their performance and take proactive measures to improve their academic standing.

4. **Analytics and Reporting Tools**: Incorporate analytics and reporting tools into the system to enable educators and administrators to analyze student performance data, identify trends, and generate actionable insights. These tools will support data-driven decision-making and help educators tailor their teaching strategies to meet student needs.

5. **Secure Data Management**: Implement robust data management protocols to ensure the security and privacy of student information. This includes encryption mechanisms, access controls, and regular data backups to protect sensitive data from unauthorized access, loss, or theft.

6. **User Training and Support**: Provide comprehensive training and support to users, including educators, administrators, and students, to ensure they are proficient in using the new system. This may involve training workshops, online tutorials, and dedicated support channels to address user queries and issues.

7. **Scalability and Flexibility**: Design the system to be scalable and flexible, capable of accommodating growing student populations, additional courses, and evolving educational requirements. This will future-proof the system and ensure it can adapt to the changing needs of educational institutions over time.

8. **Continuous Improvement**: Establish mechanisms for continuous improvement and feedback gathering to solicit input from stakeholders and make iterative improvements to the system. This may involve regular user surveys, feedback sessions, and software updates to address emerging needs and challenges.

Overall, the proposed system in student management represents a comprehensive solution for transforming the way academic records are managed, leveraging technology to enhance efficiency, accuracy, and accessibility while supporting the needs of educators, administrators, and students alike.

# 6. Front-End:

Python programming tools are complete programming environments. It allows programmers to build a GUI program using the various on-screen controls such as buttons, text, menus, boxes etc. These controls are placed on a form and then the processing details related with each control are filled in.

In the business world, competitive strategies have become the order of the day to improve quality, cut costs and provide a high response customer service base. Most organizations today need to be market driven and do a lot of value addition to their products and services. This naturally calls for rational decision making, which requires information. Information Technology or IT provides that effective channel to support and implement this strategy. Client/Server is the technology that empowers the desktop, thus setting a trend for the way successful organizations will use technology in the next decade.

# 7. Back-End:

Databases offer numerous functionalities by which one can manage large amounts of information easily over the web and high-volume data input and output over a typical file such as a text file. SQL is a query language and is very popular in databases. SQLite is a “light” version that works over syntax very much similar to SQL. SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine on the world wide web. Python has a library to access SQLite databases, called sqlite3, intended for working with this database which has been included with Python package since version 2.5. SQLite has the following features.

1. Serverless
2. Self-Contained
3. Zero-Configuration
4. Transactional
5. Single-Database

To start, you must have a computer that runs an appropriate operating system like Microsoft Windows >= 2007 Home Edition: In this case, you must install Python SQLite3.

# Tools / Platform, Hardware and Software Requirement Specification

## HARDWARE

|  |  |
| --- | --- |
| Processor | : CORE i3 and above |
| Memory | : 2GB or more |
| Cache Memory | : - |
| Hard Disk | : 520 GB or more |
| Pen Drive | : Optional |
| Printer | : Not required as of now |

## 

## SOFTWARE

|  |  |  |
| --- | --- | --- |
| Operating System |  | : Windows 10 |
| Font-End Tool |  | : Tkinter |
| Back-End |  | : SQLite3 |
| Ide |  | : Python Idle, VSCode, Jupyter notebook |

# Phase 1: System Study

## Preliminary Investigation: -

System development, a process consisting of two major steps of system analysis and design, start when management or sometimes system development personnel feel that a new system or an improvement in the existing system is required. The system development life cycle is classically thought of as the set of activities that analysts, designers and users carry out to develop and implement an information system. The system development life cycle consists of the following activities:

* Preliminary investigation
* Determination of system requirements
* Design of system
* Development of software
* System testing
* Implementation, evaluation, and maintenance

A request to take assistance from information system can be made for many reasons, but in each case someone in the organization initiates the request is made, the first system activity the preliminary investigation begins. This activity has three parts:

1. Request clarification
2. Feasibility study
3. Request approval

Request clarification: Many requests from employees and users in the organizations are not clearly defined, therefore it becomes necessary that project request must be examined and clarified properly before considering systems investigation.

## SYSTEM DEVELOPMENT LIFE CYCLE

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD), mainly deals with the software development activities.

## DEFINING A SYSTEM: -

A collection of components that work together to realize some objective forms a system. Basically, there are three major components in every system, namely input, processing and output.

In a system the different components are connected with each other and they are interdependent. For example, human body represents a complete natural system. We are also bound by many national systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs.

## SYSTEM LIFE CYCLE: -

System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and subprocesses required for developing a system.

System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the System Analysis and Design terminology, the system development life cycle means software development life cycle.

Following are the different phases of software development cycle:

* System study
* Feasibility study
* System analysis
* System design
* Coding
* Testing
* Implementation
* Maintenance

The Different Phases Of Software Development Life Cycle Are Shown Below.

PRELIMINARY

INVESTIGATION

DETERMINATION OF

REQUIREMENTS

REVIEW RUNNING

SYSTEM AND SYSTEM

MAINTENANCE

SYSTEM

IMPLEMENTATION

SYSTEM TESTING

DEVELOPMENT OF

SOFTWARE AND CODING

DESIGN OF SYSTEM

DEVELOPMENT OF

PROTOTYPE SYSTEM

SYSTEM

ANALYSIS &

DESIGN 50%

DESIGN

OF

SYSTEM

30

%

CODING

20%

FIG: SHOWING GENERAL LIFE CYCLE PROCESS AND PERCENTAGE OF TIME

A system analysis is a separation of a substance into parts for study and their implementation and detailed examination.

Before designing any system it is important that the nature of the business and the way it currently operates are clearly understood. The detailed examination provides the specific data required during designing in order to ensure that all the client's requirements are fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study. Rather it would not be wrong to say that the analysis and feasibility phases overlap. High-level analysis begins during the feasibility study. Though analysis is represented as one phase of the system development life cycle (SDLC), this is not true. Analysis begins with system initialization and continues until its maintenance. Even after successful implementation of the system, analysis may play its role for periodic maintenance and up gradation of the system.

One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding of the requirements is the poor planning of system analysis.

Analysis requires us to recall the objectives of the project and consider following three questions:

* What type of information is required?
* What are the constraints on the investigation?
* What are the potential problems that may make the task more difficult?

# Phase 2: System Analysis

## Importance of Computerized: -

There are several attributes in which the computer-based information works. Broadly the working of computer system is divided into two main groups:

* Transaction System
* Decision Support System

### Transaction System:

A transaction is a record of some well-defined single and usually small occurrence in a system. Transactions are input into the computer to update the database files. It checks the entering data for its accuracy. This means that numeric data appears in numeric field and character data in character field. Once all the checks are made, transaction is used to update the database. Transaction can be inputted in on-line mode or batch mode. In on-line mode, transactions are entered and updated into the database almost instantaneously. In batch mode, transactions are collected into batches, which may be held for a while and inputted later.

### Decision Support System:

It assists the user to make analytical decision. It shows the various data in organized way called analysis. This analysis can be made to syrdy preferences and help in making decisions.

Computer system works out best with record maintenance. It will tell you which customer would get how much pending/reports statements. It will also help to search the information about a particular person by simply entering his telephone number. User can store information as per requirement, which can be used for comparison with other reports

## Principles of System Analysis

### Principles:

1. Understand the problem before you begin to create the analysis model.
2. Develop prototypes that enable a user to understand how human machine interaction will occur.
3. Record the origin of and the reason for every requirement.
4. Use multiple views of requirements like building data, function and behavioral models.
5. Work to eliminate ambiguity.

## Complete Structure:

The limited time and resources have restricted us to incorporate, in this project, only the main activities that are performed in news sites, but utmost care has been taken to make the system efficient and user friendly.

For the optimum use of practical time it is necessary that every session is planned. Planning of this project will include the following things:

* Topic Understanding.
* Modular Break – Up of the System.
* Processor Logic for Each Module.
* Database Requirements.

## Topic Understanding:

It is vital that the field of application as introduced in the project may be totally a new field. So as soon as the project was allocated to me, I carefully went through the project to identify the requirements of the project.

### Modular Break –Up of the System:

* Identify The Various Modules In The System.
* List Them In The Right Hierarchy.
* Identify Their Priority Of Development
* Description Of The Modules:

## System Design: -

The design document that we will develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. Since solution to complex problems isn’t usually found in the first try, iterations are most likely required. This is true for software design as well. For this reason, any design strategy, design method, or design language must be flexible and must easily accommodate changes due to iterations in the design. Any technique or design needs to support and guide the partitioning process in such a way that the resulting sub-problems are as independent as possible from each other and can be combined easily for the solution to the overall problem. Sub-problem independence and easy combination of their solutions reduces the complexity of the problem. This is the objective of the partitioning process. Partitioning or decomposition during design involves three types of decisions: -

Define the boundaries along which to break;

Determine into how money pieces to break; and

Identify the proper level of detail when design should stop and implementation should start.

Basic design principles that enable the software engineer to navigate the design process suggest a set of principles for software design, which have been adapted and extended in the following list:

Free from the suffer from "tunnel vision." A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.

The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

The design should not repeat the same thing. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist.

The design should "minimize the intellectual distance" between the software and the problem as it exists in the real world. That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.

The design should exhibit uniformity and integration. A design is uniform if it appears that one person developed the entire thing. Rules of style and format should be defined for a design team before design work begins. A design is integrated if care is taken in defining interfaces between design components.

The design activity begins when the requirements document for the software to be developed is available. This may be the SRS for the complete system, as is the case if the waterfall model is being followed or the requirements for the next "iteration" if the iterative enhancement is being followed or the requirements for the prototype if the prototyping is being followed. While the requirements specification activity is entirely in the problem domain, design is the first step in moving from the problem domain toward the solution domain. Design is essentially the bridge between requirements specification and the final solution for satisfying the requirements.

The design of a system is essentially a blueprint or a plan for a solution for the system. We consider a system to be a set of components with clearly defined behavior that interacts with each other in a fixed defined manner to produce some behavior or services for its environment. A component of a system can be considered a system, with its own components. In a software system, a component is a software module.

The design process for software systems, often, has two levels. At the first level, the focus is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called detailed design or logic design. Detailed design essentially expands the system design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

Because the detailed design is an extension of system design, the system design controls the major structural characteristics of the system. The system design has a major impact on the testability and modifiability of a system, and it impacts its efficiency. Much of the design effort for designing software is spent creating the system design.

The input to the design phase is the specifications for the system to be designed. Hence, reasonable entry criteria can be that the specifications are stable and have been approved, hoping that the approval mechanism will ensure that the specifications are complete, consistent, unambiguous, etc. The output of the top-level design phase is the architectural design or the system design for the software system to be built. This can be produced with or without using a design methodology. Reasonable exit criteria for the phase could be that the design has been verified against the input specifications and has been evaluated and approved for quality.

A design can be object-oriented or function-oriented. In function-oriented design, the design consists of module definitions, with each module supporting a functional abstraction. In object-oriented design, the modules in the design represent data abstraction (these abstractions are discussed in more detail later). In the function-oriented methods for design and describe one particular methodology the structured design methodology in some detail. In a function- oriented design approach, a system is viewed as a transformation function, transforming the inputs to the desired outputs. The purpose of the design phase is to specify the components for this transformation function, so that each component is also a transformation function. Hence, the basic output of the system design phase, when a function-oriented design approach is being followed, is the definition of all the major data structures in the system, all the major modules of the system, and how the modules interact with each other. Once the designer is satisfied with the design he has produced, the design is to be precisely specified in the form of a document. To specify the design, specification languages are used. Producing the design specification is the ultimate objective of the design phase. The purpose of this design document is quite different from that of the design notation. Whereas a design represented using the design notation is largely to be used by the designer, a design specification has to be so precise and complete that it can be used as a basis of further development by other programmers. Generally, design specification uses textual structures, with design notation helping in understanding

# Data Modeling: -

Introduction to data dictionary:

Data dictionaries are an integral component of structured analysis, since data flow diagrams by themselves do not fully describe the subject of the investigation. The data flow diagrams provide the additional details about the project/system.

## Data Dictionary (Definition):

A data dictionary is a catalog- a repository- of the elements in a system. These elements center on the data and the way they are structured to meet user requirements and organization needs. A data dictionary consists of a list of all the elements composing the data flowing through a system. The major elements are data flows, data stores, and processes. The data dictionary stores details and descriptions of these elements.

# ER Diagram:

An ER diagram, or Entity-Relationship diagram, is a visual representation of the relationships between entities in a database. It uses various symbols and connectors to illustrate how different entities relate to each other.

In the context of your hotel management system project, an ER diagram would show entities like customers, rooms, bookings, payments, etc., and their relationships such as a customer booking multiple rooms or a room being booked for multiple days by different customers. It helps in understanding the structure of the database and how different data entities are connected.

# The need for an ER diagram in my project includes:

1. **Database Design:** It helps in designing the database structure by defining entities, attributes, and relationships, ensuring a well-organized database.
2. **Data Integrity:** Ensures data accuracy and integrity by defining constraints and relationships between entities, preventing inconsistencies in the database.
3. **Understanding Relationships:** Provides a clear visual representation of how different entities are related, making it easier for developers and stakeholders to understand the data model.
4. **Efficient Querying:** Optimizes database querying and performance by structuring data in a way that reduces redundancy and improves data retrieval efficiency.

SMS ER Diagram: -

M

M

1

1

1

M

M

M

Has

Has

Has

Has

STUDENT

RESULT

COURSE

ADMIN

# DFD: -

DFD stands for Data Flow Diagram. It's a graphical representation that illustrates the flow of data through a system, showcasing how data is input, processed, stored, and outputted.

In your hotel management system project, a DFD would depict the flow of data from various sources like customers, room bookings, payments, etc., through different processes such as booking management, billing, and reporting, and finally to outputs like invoices or reports.

The need for a DFD in my project includes: -

1. **System Understanding:** It provides a clear and visual understanding of how data moves through different parts of the system, helping in system analysis and design.
2. **Identifying Processes:** Helps in identifying the processes or functions involved in managing hotel operations, such as booking rooms, handling payments, managing inventory, etc.
3. **Data Flow:** Illustrates how data flows between different components of the system, including inputs, outputs, and data storage, aiding in data management and integration.
4. **System Optimization:** Enables you to analyse and optimize the flow of data within the system, improving efficiency, reducing bottlenecks, and enhancing overall system performance.

### Context level DFD (0 level)

**ADMIN**

**Request for Login**

**Response**

DATABASE

0 Level DFD for student management system project

### Level 1 DFD:

Request for login

Check for login

Manage Student & Admin user

View Report

Add Student

Add Course

Display Data

Response

Response

Response

Response

Send Data

Send Data

Insert Data

Insert Data

Reply

Reply

Reply

Reply

Reply

StudentMst, AdminMst

ResultMst

StudentMst

CourseMst

AdminMst

ADMIN

### Level 2 DFD:

Request for login

Check for login

Send Data

Update Fees

Add Fees

Add Student

Add Course

Response

Response

Response

Response

Response

Update Data

Insert Data

Insert Data

Reply

Reply

Reply

Reply

Reply

StudentMst, AdminMst

CourseMst

StudentMst

CourseMst

AdminMst

ADMIN

Level 2 DFD:

Request for login

Check for login

Insert/Delete Data

Add User Detail

Send Student ID

Send Course ID

Response

Response

Response

Response

Delete Data

Delete Data

Reply

Reply

Reply

Reply

LoginMst

StudentMst

CourseMst

AdminMst

ADMIN

# Class diagram: -

A Class Diagram is a type of UML (Unified Modelling Language) diagram that represents the structure and relationships of classes in a system or software application.

In the context of your hotel management system project, a Class Diagram would illustrate the various classes, their attributes, methods, and associations within the system. For example, classes like Customer, Room, Booking, Payment, etc., would be represented along with their relationships and attributes.

# The need for a Class Diagram in my project includes: -

1. **Structural Representation:** It provides a structural representation of classes and their relationships, helping in understanding the architecture and organization of the system.
2. **Class Relationships:** Illustrates how different classes are related to each other, such as inheritance, composition, aggregation, and association, which is crucial for designing and developing the system.
3. **Attributes and Methods:** Shows the attributes (data) and methods (functions) of each class, aiding in designing class interfaces and defining functionalities.
4. **Design Visualization:** Enables visualizing the design of the system, making it easier to communicate and collaborate with stakeholders, developers, and designers.

### CLASS DIAGRAM: -

Student

Roll No.

Name

Email

Gender

State

Address

D.O.B

Contact

Admission

Course

City

Pin

Save ()

Update ()

Delete ()

Clear ()

Course

Course Name

Duration

Charges

Description

Save ()

Update ()

Delete ()

Clear ()

Result

Select Student

Name

Course

Marks Obtained

Full Marks

Submit ()

Clear ()

Student Management system

Email

Password

Login ()

# Modules:

## **Login Module**: -

This module page allows for different types of login details for students and administrators. If someone does not have a login ID and password, they can register. Additionally, if someone has forgotten their password, they can create a new password.

## **Course Module**: -

This module allows users to add courses and provides information about the fee and duration of each course, along with a detailed description. Users can easily input these details for each course they wish to offer or manage within the system.

Upon adding a new course, users can specify its duration in terms of weeks, months, or any other appropriate unit of time. Additionally, users can input the fee or charges associated with the course, ensuring transparency and clarity for prospective students.

Furthermore, users have the opportunity to provide a comprehensive description of each course, highlighting its objectives, curriculum, learning outcomes, and any other relevant details. This detailed description helps students make informed decisions about which courses to enroll in based on their interests, goals, and educational needs.

Overall, this module streamlines the process of adding new courses to the system, providing essential information about course duration, fee structure, and detailed descriptions to facilitate effective course management and student enrollment.

## **Student Module**: -

This module maintains the information of students and allows for the addition of new student information to the database. Additionally, it enables users to update the information of existing students and delete student records as needed. Furthermore, users can search for specific students by roll number, facilitating efficient retrieval of student data.

Upon adding new student information, users can input details such as the student's name, roll number, contact information, and any other relevant information. In cases where existing student information requires correction or update, users can easily modify the relevant fields to ensure accuracy.

Moreover, the module provides functionality to delete student records if necessary, thereby allowing for the removal of outdated or redundant information from the database.

Overall, this module serves as a comprehensive tool for managing student information, offering capabilities for addition, modification, deletion, and retrieval of student records to meet the needs of educational institutions and administrators.

## **Result Module**: -

This module assists in adding marks for students. It first searches for the student's name and course using the roll number. Then, it allows the addition of the marks obtained out of the total marks.

## **Report Module**: -

This module assists in viewing the report card of a student. It displays the complete report card of the student based on their roll number. Additionally, it provides the functionality to delete a student's result if necessary.

# Phase 3: System Planning

## System Planning: -

Scheduling of a software project does not differ greatly from scheduling of any multi- task engineering effort. Therefore, generalized project scheduling tools and techniques can be applied with little modification to software projects.

Program evaluation and review technique (PERT) and critical path method (CPM) are two project scheduling methods that can be applied to software development. Both techniques are driven by information already developed in earlier project planning activities.

# Estimates of Effort

* A decomposition of the product function.
* The selection of the appropriate process model and task set.
* Decomposition of tasks.

Interdependencies among tasks may be defined using a task network. Tasks, sometimes called the project Work Breakdown Structure (WBS) are defined for the product as a whole or for individual functions.

Both PERT and CPM provide quantitative tools that allow the software planner to (1) determine the critical path-the chain of tasks that determines the duration of the project; (2) establish "most likely" time estimates for individual tasks by applying statistical models; and (3) calculate "boundary times" that define a time window" for a particular task.

Boundary time calculations can be very useful in software project scheduling. Slippage in the design of one function, for example, can retard further development of other functions. It describes important boundary times that may be discerned from a PERT or CPM network: (I) the earliest time that a task can begin when preceding tasks are completed in the shortest possible time, (2) the latest time for task initiation before the minimum project completion time is delayed, (3) the earliest finish-the sum of the earliest start and the task duration, (4) the latest finish- the latest start time added to task duration, and (5) the total float-the amount of surplus time or leeway allowed in scheduling tasks so that the network critical path maintained on schedule. Boundary time calculations lead to a determination of critical path and provide the manager with a quantitative method for evaluating progress as tasks are completed.

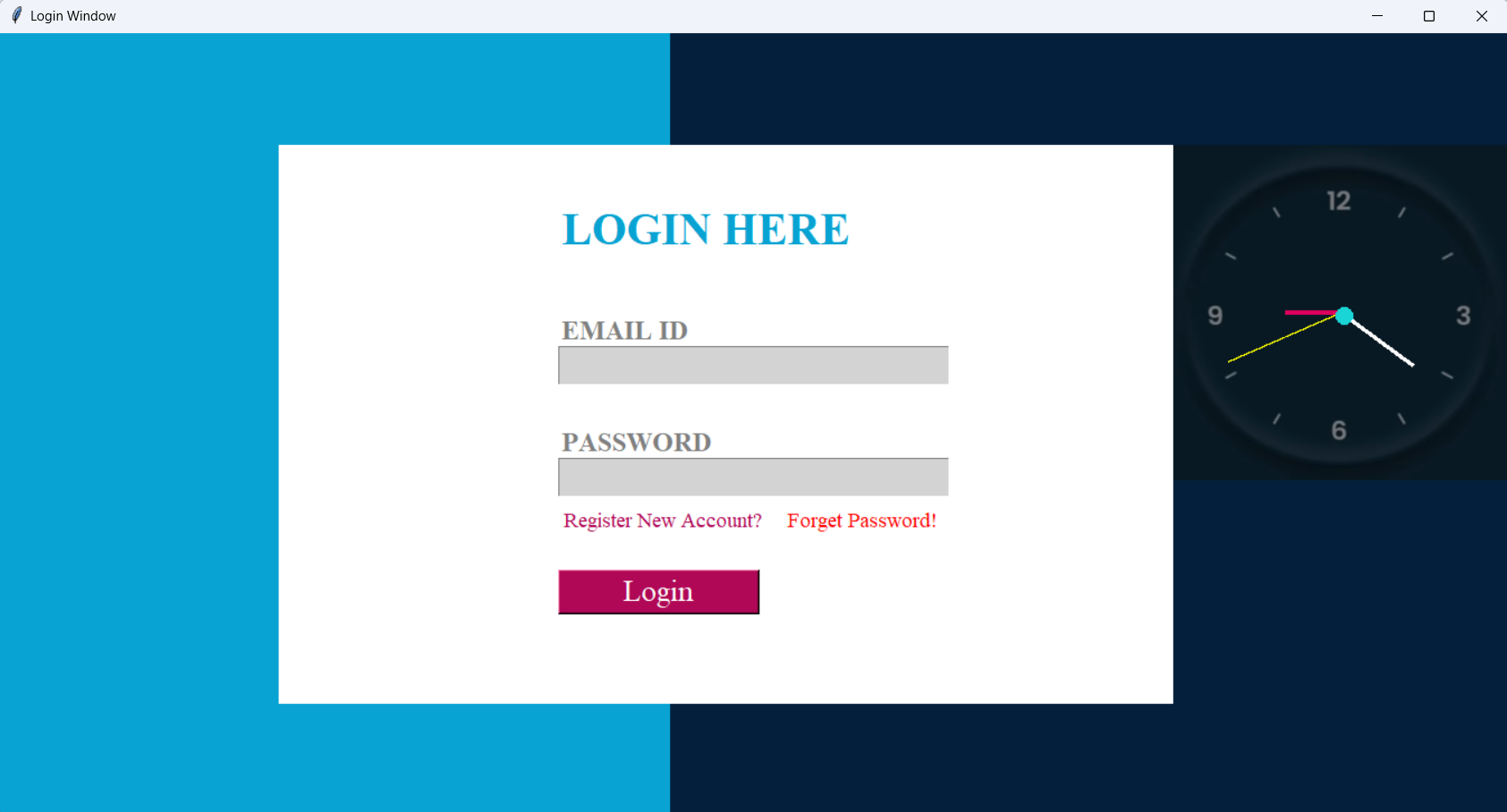
Both PERT and CPM have been implemented in a wide variety of automated tools that are available for the personal computer. Such tools are easy to use and take the scheduling methods described previously available to every software project manager.

# 

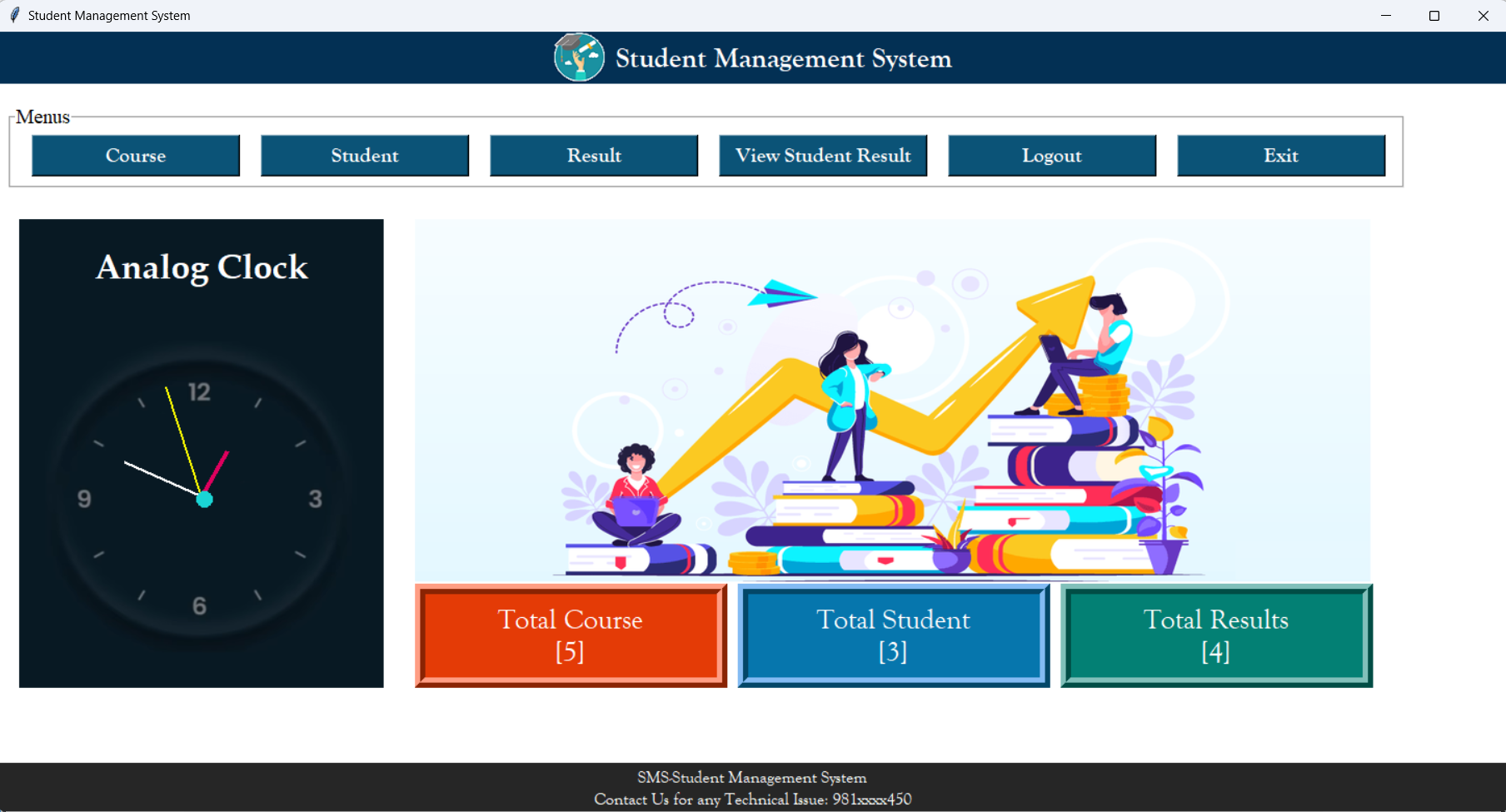
# Phase 4: Screen Shot



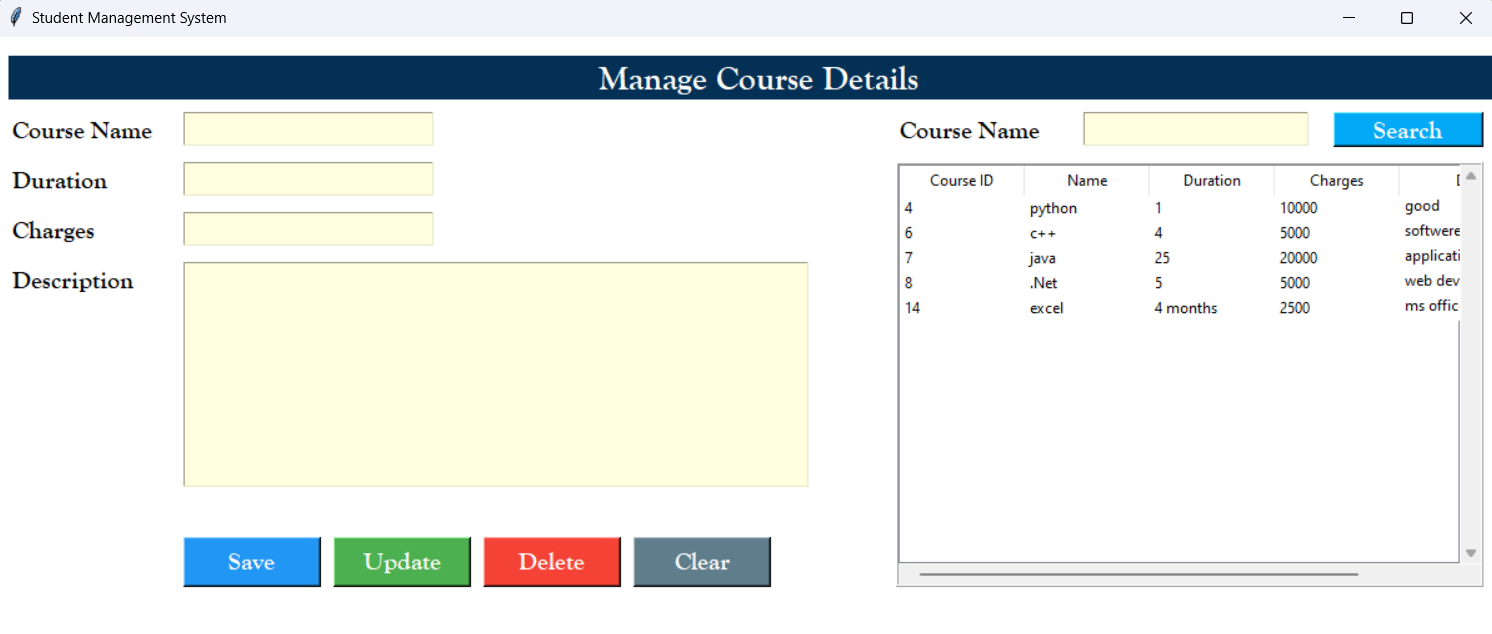
## Login window:



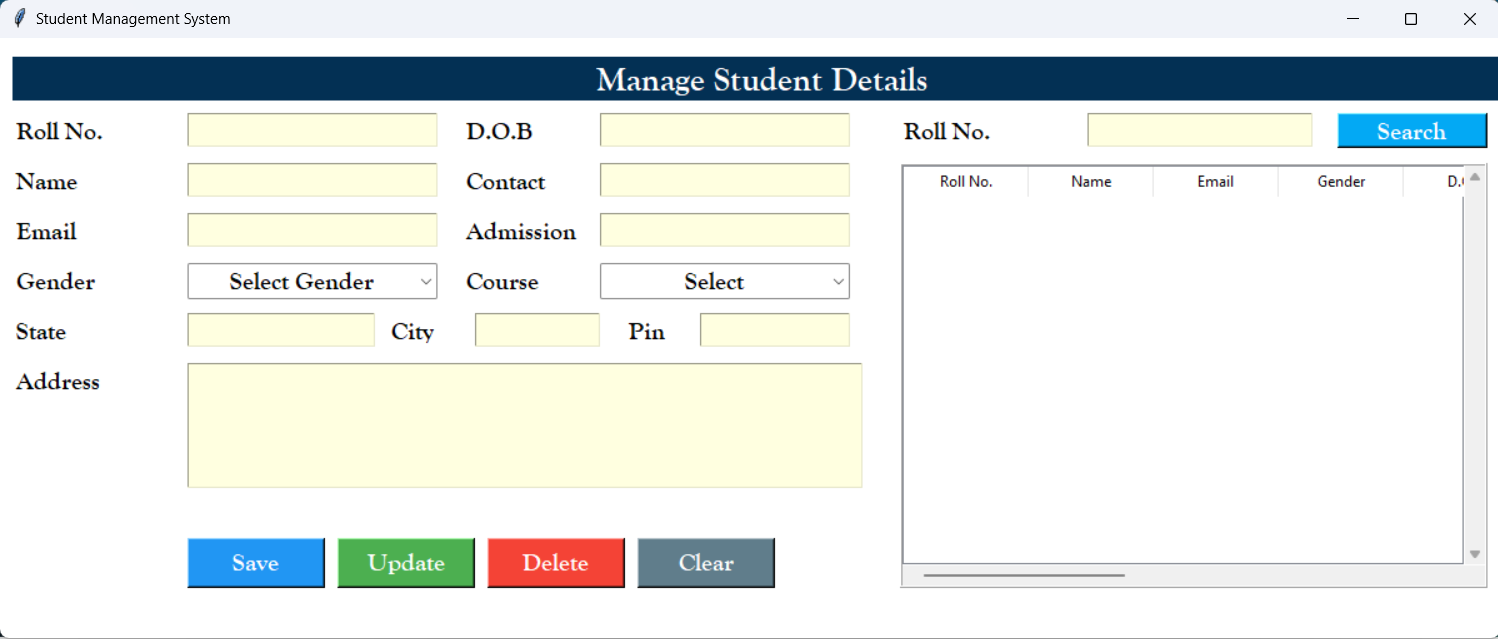
## Dashboard window:



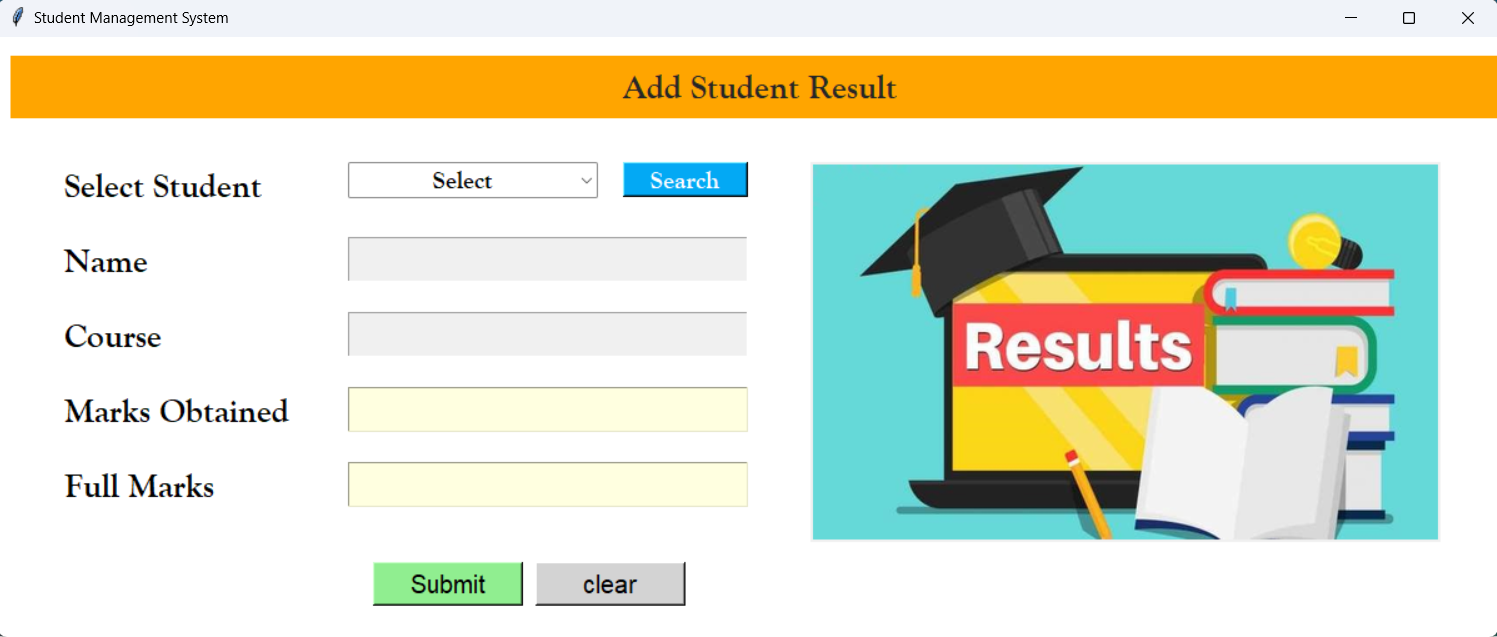
## Course window:



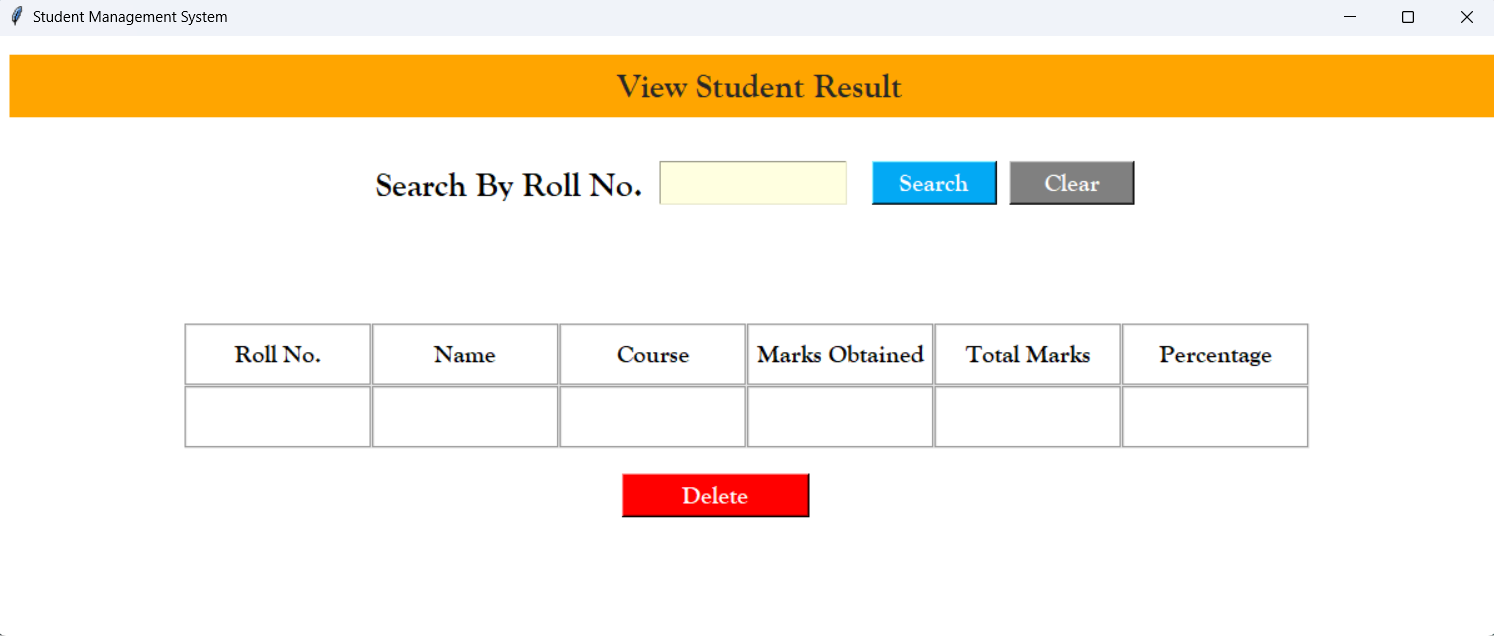
## Student window:



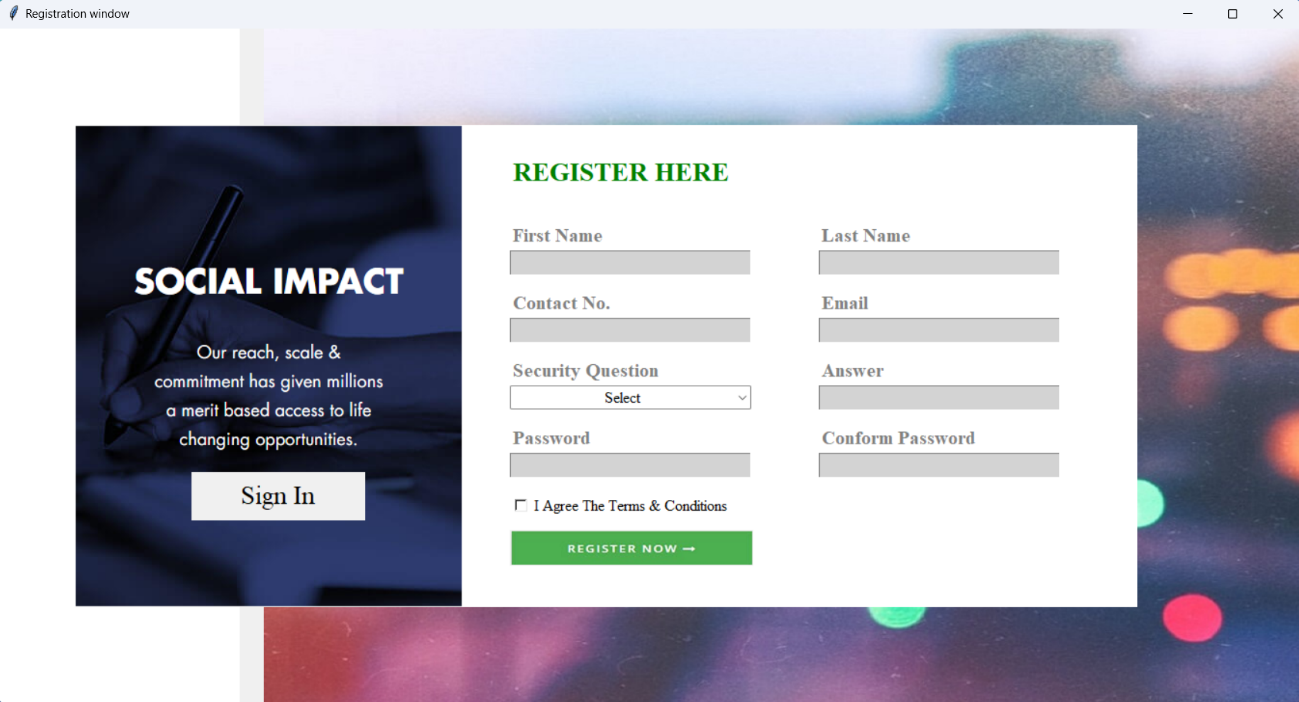
## Result window:



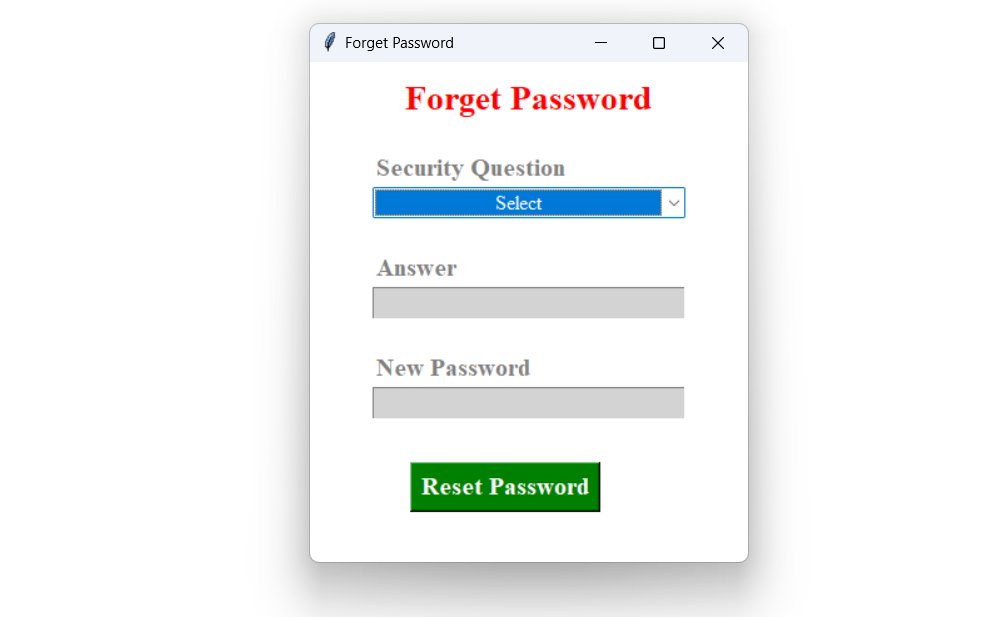
## Report window:



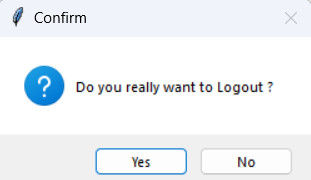
## Registration window:



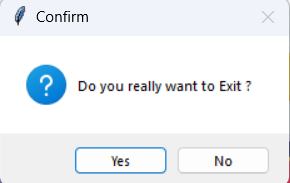
## Forget Password window:



Logout window:



Exit window:



# 

# Phase 5: Source Code



## Loginpage

from tkinter import \*

from PIL import Image,ImageTk,ImageDraw

from datetime import \*

import time

from math import \*

from tkinter import messagebox

import sqlite3

import os

from forget import Forget\_pass

class Login:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Login Window")

self.root.geometry("1350x700+0+0")

self.root.config(bg="#021e2f")

#=======backgroung colours=========>

left\_lbl=Label(self.root,bg="#08A3D2",bd=0).place(x=0,y=0,relheight=1,width=600)

right\_lbl=Label(self.root,bg="#031F3C",bd=0).place(x=600,y=0,relheight=1,relwidth=1)

#==========Frames=============>

login\_frame=Frame(self.root,bg="white")

login\_frame.place(x=250,y=100,height=500,width=800)

title=Label(login\_frame,text="LOGIN HERE",font=("times new roman",30,"bold"),bg="white",fg="#08A3D2").place(x=250,y=50)

email=Label(login\_frame,text="EMAIL ID",font=("times new roman",18,"bold"),bg="white",fg="gray").place(x=250,y=150)

self.txt\_email=Entry(login\_frame,font=("times new roman",15),bg="lightgray")

self.txt\_email.place(x=250,y=180,width=350,height=35)

password=Label(login\_frame,text="PASSWORD",font=("times new roman",18,"bold"),bg="white",fg="gray").place(x=250,y=250)

self.txt\_password=Entry(login\_frame,font=("times new roman",15),bg="lightgray",show="\*")

self.txt\_password.place(x=250,y=280,width=350,height=35)

but\_reg=Button(login\_frame,text="Register New Account?",font=("times new roman",14),bg="white",bd=0,fg="#B00857",cursor="hand2",command=self.register).place(x=250,y=320)

but\_forget=Button(login\_frame,text="Forget Password!",font=("times new roman",14),bg="white",bd=0,fg="red",cursor="hand2",command=self.forget).place(x=450,y=320)

but\_login=Button(login\_frame,text="Login",font=("times new roman",20),fg="white",bg="#B00857",cursor="hand2",command=self.login).place(x=250,y=380,width=180,height=40)

#=======Clock label=========>

self.lbl=Label(self.root,font=("Book Antiqua",25,"bold"),bg="#081923",fg="white",compound=BOTTOM,bd=0)

self.lbl.place(x=1050,y=100,height=300,width=300)

self.working()

#========Function===========>

def login(self):

if self.txt\_email.get()=="" or self.txt\_password.get()=="":

messagebox.showerror("Error","All Fields Are Required",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=? and password=?",(self.txt\_email.get(),self.txt\_password.get()))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Invalid USERNAME & PASSWORD",parent=self.root)

else:

messagebox.showinfo("Success",f"Welcome: {self.txt\_email.get()}",parent=self.root)

self.root.destroy()

os.system("python deshboard.py")

con.close()

except Exception as es:

messagebox.showerror("Error",f"Error Due to: {str(es)}",parent=self.root)

def clock\_image(self,hr,mint,secn):

cloks=Image.new("RGB",(400,400),(8,25,35))

draw=ImageDraw.Draw(cloks)

#=======for clock Image======>

bg=Image.open("images/c.png")

bg=bg.resize((300,300),Image.Resampling.LANCZOS)

cloks.paste(bg,(50,50))

origin=200,200

#formula t rotate the clock

#angle\_in\_radians= angle\_in\_degrees \* math.pi/180

#line\_length=100

#center\_x= 250

#center\_y= 250

#end\_x= center\_x + line\_length \* math.sin(angle\_in\_radians)

#end\_y= center\_y - line\_length \* math.cos(angle\_in\_radians)

#=========hour line image===========>

draw.line((origin,200+50\*sin(radians(hr)),200-50\*cos(radians(hr))),fill="#DF005E",width=4)

#=========Min line image===========>

draw.line((origin,200+80\*sin(radians(mint)),200-80\*cos(radians(mint))),fill="white",width=3)

#=========Sec line image===========>

draw.line((origin,200+110\*sin(radians(secn)),200-110\*cos(radians(secn))),fill="yellow",width=2)

draw.ellipse((195,195,210,210),fill="#1AD5D5")

cloks.save("images/new.png")

def working(self):

h=datetime.now().time().hour

m=datetime.now().time().minute

s=datetime.now().time().second

hr=(h/12)\*360

mint=(m/60)\*360

secn=(s/60)\*360

self.clock\_image(hr,mint,secn)

#self.img=Image.open("images/new.png")

self.img=ImageTk.PhotoImage(file="images/new.png")

self.lbl.config(image=self.img)

self.lbl.after(200,self.working)

def register(self):

self.root.destroy()

os.system("python register.py")

def forget(self):

if self.txt\_email.get()=="":

messagebox.showerror("Error","Please Enter the Email Address to Reset Your Password",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=?",(self.txt\_email.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please Enter the Valid Email to Reset Your Password",parent=self.root)

else:

self.senEmail=self.txt\_email.get()

self.new\_win=Toplevel(self.root)

self.new\_obj=Forget\_pass(self.new\_win,self.senEmail)

self.clear()

con.close()

except Exception as es:

messagebox.showerror("Error",f"Error Due to: {str(es)}",parent=self.root)

def clear(self):

self.txt\_email.delete(0,END)

self.txt\_password.delete(0,END)

#=======Main==============>

root=Tk()

obj=Login(root)

root.mainloop()

## Dashboard page:

from tkinter import \*

from PIL import Image,ImageTk,ImageDraw

from course import CourseClass

from student import studentClass

from result import resultClass

from report import reportClass

from tkinter import messagebox

import os

import sqlite3

from datetime import \*

import time

from math import \*

class RMS:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Management System")

self.root.geometry("1450x750+0+0")

self.root.config(bg="white")

#====icons=====>

self.logo\_dash=ImageTk.PhotoImage(file="images/logo\_p.png")

#====Title====>

title=Label(self.root,text="Student Management System",padx=10,compound=LEFT,image=self.logo\_dash,font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=0,y=0,relwidth=1,height=50)

#====Menu====>

M\_Frame=LabelFrame(self.root,text="Menus",font=("times new roman",15),bg="white")

M\_Frame.place(x=10,y=70,width=1340,height=80)

btn\_course=Button(M\_Frame,text="Course",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_course).place(x=20,y=5,width=200,height=40)

btn\_student=Button(M\_Frame,text="Student",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_student).place(x=240,y=5,width=200,height=40)

btn\_result=Button(M\_Frame,text="Result",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_result).place(x=460,y=5,width=200,height=40)

btn\_view=Button(M\_Frame,text="View Student Result",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_report).place(x=680,y=5,width=200,height=40)

btn\_logout=Button(M\_Frame,text="Logout",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.logout).place(x=900,y=5,width=200,height=40)

btn\_exit=Button(M\_Frame,text="Exit",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.exits).place(x=1120,y=5,width=200,height=40)

#=====content window====>

self.bg\_img=Image.open("images/bg.png")

self.bg\_img=self.bg\_img.resize((920,350),Image.Resampling.LANCZOS)

self.bg\_img=ImageTk.PhotoImage(self.bg\_img)

self.lbl\_bg=Label(self.root,image=self.bg\_img).place(x=400,y=180,width=920,height=350)

#====updata\_details=====>

self.lbl\_course=Label(self.root,text="Total courses\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#e43b06",fg="white")

self.lbl\_course.place(x=400,y=530,height=100,width=300)

self.lbl\_student=Label(self.root,text="Total Student\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#0676ad",fg="white")

self.lbl\_student.place(x=710,y=530,height=100,width=300)

self.lbl\_result=Label(self.root,text="Total Results\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#038074",fg="white")

self.lbl\_result.place(x=1020,y=530,height=100,width=300)

#=======Clock label=========>

self.lbl=Label(self.root,text="\nAnalog Clock",font=("Book Antiqua",25,"bold"),bg="#081923",fg="white",compound=BOTTOM,bd=0)

self.lbl.place(x=20,y=180,height=450,width=350)

self.working()

#====Footer====>

footer=Label(self.root,text="SRMS-Student Result Managment System\nContact Us for any Technical Issue: 981xxxx450",font=("goudy old style",12),bg="#262626",fg="white").pack(side=BOTTOM,fill=X)

self.update()

#========Functions==============>

def add\_course(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=CourseClass(self.new\_win)

def add\_student(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=studentClass(self.new\_win)

def add\_result(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=resultClass(self.new\_win)

def add\_report(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=reportClass(self.new\_win)

def logout(self):

op=messagebox.askyesno("Confirm","Do you really want to Logout ?",parent=self.root)

if op==True:

self.root.destroy()

os.system("python login.py")

def exits(self):

op=messagebox.askyesno("Confirm","Do you really want to Exit ?",parent=self.root)

if op==True:

self.root.destroy()

def clock\_image(self,hr,mint,secn):

cloks=Image.new("RGB",(400,400),(8,25,35))

draw=ImageDraw.Draw(cloks)

#=======for clock Image======>

bg=Image.open("images/c.png")

bg=bg.resize((300,300),Image.Resampling.LANCZOS)

cloks.paste(bg,(50,50))

origin=200,200

#formula t rotate the clock

#angle\_in\_radians= angle\_in\_degrees \* math.pi/180

#line\_length=100

#center\_x= 250

#center\_y= 250

#end\_x= center\_x + line\_length \* math.sin(angle\_in\_radians)

#end\_y= center\_y - line\_length \* math.cos(angle\_in\_radians)

#=========hour line image===========>

draw.line((origin,200+50\*sin(radians(hr)),200-50\*cos(radians(hr))),fill="#DF005E",width=4)

#=========Min line image===========>

draw.line((origin,200+80\*sin(radians(mint)),200-80\*cos(radians(mint))),fill="white",width=3)

#=========Sec line image===========>

draw.line((origin,200+110\*sin(radians(secn)),200-110\*cos(radians(secn))),fill="yellow",width=2)

draw.ellipse((195,195,210,210),fill="#1AD5D5")

cloks.save("images/new.png")

def working(self):

h=datetime.now().time().hour

m=datetime.now().time().minute

s=datetime.now().time().second

hr=(h/12)\*360

mint=(m/60)\*360

secn=(s/60)\*360

self.clock\_image(hr,mint,secn)

#self.img=Image.open("images/new.png")

self.img=ImageTk.PhotoImage(file="images/new.png")

self.lbl.config(image=self.img)

self.lbl.after(200,self.working)

def update(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from course")

cr=cur.fetchall()

self.lbl\_course.config(text=f"Total Course\n[{str(len(cr))}]")

cur.execute("select \* from student")

st=cur.fetchall()

self.lbl\_student.config(text=f"Total Student\n[{str(len(st))}]")

cur.execute("select \* from results")

rs=cur.fetchall()

self.lbl\_result.config(text=f"Total Results\n[{str(len(rs))}]")

self.lbl\_course.after(200,self.update)

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=RMS(root)

root.mainloop()

## Course Page:

from tkinter import \*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class CourseClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====Title====>

title=Label(self.root,text="Manage Course Details",font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,relwidth=1,height=35)

#=======Variables=======>

self.var\_course=StringVar()

self.var\_duration=StringVar()

self.var\_charges=StringVar()

#=======widgets=========>

lbl\_courseName=Label(self.root,text="Course Name",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=60)

lbl\_duration=Label(self.root,text="Duration",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=100)

lbl\_charges=Label(self.root,text="Charges",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=140)

lbl\_description=Label(self.root,text="Description",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=180)

#========Entry Fields========>

self.txt\_courseName=Entry(self.root,textvariable=self.var\_course,font=("goudy old style",15,'bold'),bg="lightyellow")

self.txt\_courseName.place(x=150,y=60,width=200)

txt\_duration=Entry(self.root,textvariable=self.var\_duration,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=150,y=100,width=200)

txt\_charges=Entry(self.root,textvariable=self.var\_charges,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=150,y=140,width=200)

self.txt\_description=Text(self.root,font=("goudy old style",15,'bold'),bg="lightyellow")

self.txt\_description.place(x=150,y=180,width=500,height=180)

#=====Buttons===========>

self.btn\_add=Button(self.root,text="Save",font=("goudy old style",15,"bold"),bg="#2196f3",fg="white",cursor="hand2",command=self.add)

self.btn\_add.place(x=150,y=400,width=110,height=40)

self.btn\_update=Button(self.root,text="Update",font=("goudy old style",15,"bold"),bg="#4caf50",fg="white",cursor="hand2",command=self.update)

self.btn\_update.place(x=270,y=400,width=110,height=40)

self.btn\_delete=Button(self.root,text="Delete",font=("goudy old style",15,"bold"),bg="#f44336",fg="white",cursor="hand2",command=self.delete)

self.btn\_delete.place(x=390,y=400,width=110,height=40)

self.btn\_clear=Button(self.root,text="Clear",font=("goudy old style",15,"bold"),bg="#607d8b",fg="white",cursor="hand2",command=self.clear)

self.btn\_clear.place(x=510,y=400,width=110,height=40)

#=======Search panel=======>

self.var\_search=StringVar()

lbl\_search\_courseName=Label(self.root,text="Course Name",font=("goudy old style",15,'bold'),bg="white").place(x=720,y=60)

txt\_search\_courseName=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=870,y=60,width=180)

btn\_search=Button(self.root,text="Search",font=("goudy old style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).place(x=1070,y=60,width=120,height=28)

#======Content========>

self.C\_Frame=Frame(self.root,bd=2,relief=RIDGE)

self.C\_Frame.place(x=720,y=100,width=470,height=340)

scrolly=Scrollbar(self.C\_Frame,orient=VERTICAL)

scrollx=Scrollbar(self.C\_Frame,orient=HORIZONTAL)

self.courseTable=ttk.Treeview(self.C\_Frame,columns=("cid","name","duration","charges","description"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)

scrollx.pack(side=BOTTOM,fill=X)

scrolly.pack(side=RIGHT,fill=Y)

scrollx.config(command=self.courseTable.xview)

scrolly.config(command=self.courseTable.yview)

self.courseTable.heading("cid",text="Course ID")

self.courseTable.heading("name",text="Name")

self.courseTable.heading("duration",text="Duration")

self.courseTable.heading("charges",text="Charges")

self.courseTable.heading("description",text="Description")

self.courseTable["show"]='headings'

self.courseTable.column("cid",width=100)

self.courseTable.column("name",width=100)

self.courseTable.column("duration",width=100)

self.courseTable.column("charges",width=100)

self.courseTable.column("description",width=150)

self.courseTable.pack(fill=BOTH,expand=1)

self.courseTable.bind("<ButtonRelease-1>",self.get\_data)

self.show()

#=========function=========>

def clear(self):

self.show()

self.var\_course.set("")

self.var\_duration.set("")

self.var\_charges.set("")

self.var\_search.set("")

self.txt\_description.delete('1.0',END)

self.txt\_courseName.config(state=NORMAL)

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name should be required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please select course from the list first",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete ?",parent=self.root)

if op==True:

cur.execute("delete from course where name=?",(self.var\_course.get(),))

con.commit()

con.close()

messagebox.showinfo("Delete","Course deleted Successfully",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def get\_data(self,ev):

self.txt\_courseName.config(state="readonly")

r=self.courseTable.focus()

content=self.courseTable.item(r)

row=content["values"]

self.var\_course.set(row[1])

self.var\_duration.set(row[2])

self.var\_charges.set(row[3])

self.txt\_description.delete('1.0',END)

self.txt\_description.insert(END,row[4])

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name should be required",parent=self.root)

elif self.var\_duration.get()=="":

messagebox.showerror("Error","Duration should be required",parent=self.root)

elif self.var\_charges.get()=="":

messagebox.showerror("Error","Charges should be required",parent=self.root)

elif self.txt\_description.get('1.0',END).strip()=="":

messagebox.showerror("Error","Description should be required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

#print(row)

if row!=None:

messagebox.showerror("Error","Course Name Already Present",parent=self.root)

else:

cur.execute("insert into course(name,duration,charges,description) values(?,?,?,?)",(

self.var\_course.get(),self.var\_duration.get(),self.var\_charges.get(),self.txt\_description.get("1.0",END)

))

con.commit()

con.close()

messagebox.showinfo("Success","Course Added Successfully",parent=self.root)

self.clear()

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def update(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name should be required",parent=self.root)

elif self.var\_duration.get()=="":

messagebox.showerror("Error","Duration should be required",parent=self.root)

elif self.var\_charges.get()=="":

messagebox.showerror("Error","Charges should be required",parent=self.root)

elif self.txt\_description.get('1.0',END).strip()=="":

messagebox.showerror("Error","Description should be required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Select Course From List",parent=self.root)

else:

cur.execute("update course set duration=?,charges=?,description=? where name=?",(

self.var\_duration.get(),self.var\_charges.get(),self.txt\_description.get("1.0",END),self.var\_course.get()

))

con.commit()

messagebox.showinfo("Success","Course Update Successfully",parent=self.root)

self.show()

con.close()

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def show(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from course")

rows=cur.fetchall()

self.courseTable.delete(\*self.courseTable.get\_children())

for row in rows:

self.courseTable.insert('',END,values=row)

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute(f"select \* from course where name LIKE '%{self.var\_search.get()}%'")

rows=cur.fetchall()

self.courseTable.delete(\*self.courseTable.get\_children())

for row in rows:

self.courseTable.insert('',END,values=row)

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=CourseClass(root)

root.mainloop()

## Student page:

from tkinter import \*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class studentClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====Title====>

title=Label(self.root,text="Manage Student Details",font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,relwidth=1,height=35)

#=======Variables=======>

self.var\_roll=StringVar()

self.var\_name=StringVar()

self.var\_email=StringVar()

self.var\_gender=StringVar()

self.var\_dob=StringVar()

self.var\_contact=StringVar()

self.var\_course=StringVar()

self.var\_a\_date=StringVar()

self.var\_state=StringVar()

self.var\_city=StringVar()

self.var\_pin=StringVar()

#=======widgets=========>

#======column 1===========>

lbl\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=60)

lbl\_Name=Label(self.root,text="Name",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=100)

lbl\_Email=Label(self.root,text="Email",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=140)

lbl\_gender=Label(self.root,text="Gender",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=180)

lbl\_state=Label(self.root,text="State",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=220)

lbl\_address=Label(self.root,text="Address",font=("goudy old style",15,'bold'),bg="white").place(x=10,y=260)

lbl\_pin=Label(self.root,text="Pin",font=("goudy old style",15,'bold'),bg="white").place(x=500,y=220)

txt\_pin=Entry(self.root,textvariable=self.var\_pin,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=560,y=220,width=120)

#========Entry Fields column 1========>

self.txt\_roll=Entry(self.root,textvariable=self.var\_roll,font=("goudy old style",15,'bold'),bg="lightyellow")

self.txt\_roll.place(x=150,y=60,width=200)

txt\_name=Entry(self.root,textvariable=self.var\_name,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=150,y=100,width=200)

txt\_email=Entry(self.root,textvariable=self.var\_email,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=150,y=140,width=200)

self.txt\_gender=ttk.Combobox(self.root,textvariable=self.var\_gender,values=("Select Gender","Male","Female","Other"),font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_gender.place(x=150,y=180,width=200)

self.txt\_gender.current(0)

txt\_state=Entry(self.root,textvariable=self.var\_state,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=150,y=220,width=150)

#=======Text Address========>

self.txt\_address=Text(self.root,font=("goudy old style",15,'bold'),bg="lightyellow")

self.txt\_address.place(x=150,y=260,width=540,height=100)

#=======widgets=========>

#======column 2===========>

lbl\_dob=Label(self.root,text="D.O.B",font=("goudy old style",15,'bold'),bg="white").place(x=370,y=60)

lbl\_contact=Label(self.root,text="Contact",font=("goudy old style",15,'bold'),bg="white").place(x=370,y=100)

lbl\_addmission=Label(self.root,text="Admission",font=("goudy old style",15,'bold'),bg="white").place(x=370,y=140)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",15,'bold'),bg="white").place(x=370,y=180)

lbl\_city=Label(self.root,text="City",font=("goudy old style",15,'bold'),bg="white").place(x=310,y=220)

#========Entry Fields column 2========>

self.course\_list=[]

#====Function\_call to update the list====>

self.fetch\_course()

txt\_dob=Entry(self.root,textvariable=self.var\_dob,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=480,y=60,width=200)

txt\_contact=Entry(self.root,textvariable=self.var\_contact,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=480,y=100,width=200)

txt\_addmission=Entry(self.root,textvariable=self.var\_a\_date,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=480,y=140,width=200)

self.txt\_course=ttk.Combobox(self.root,textvariable=self.var\_course,values=self.course\_list,font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_course.place(x=480,y=180,width=200)

self.txt\_course.set("Select")

txt\_city=Entry(self.root,textvariable=self.var\_city,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=380,y=220,width=100)

#=====Buttons===========>

self.btn\_add=Button(self.root,text="Save",font=("goudy old style",15,"bold"),bg="#2196f3",fg="white",cursor="hand2",command=self.add)

self.btn\_add.place(x=150,y=400,width=110,height=40)

self.btn\_update=Button(self.root,text="Update",font=("goudy old style",15,"bold"),bg="#4caf50",fg="white",cursor="hand2",command=self.update)

self.btn\_update.place(x=270,y=400,width=110,height=40)

self.btn\_delete=Button(self.root,text="Delete",font=("goudy old style",15,"bold"),bg="#f44336",fg="white",cursor="hand2",command=self.delete)

self.btn\_delete.place(x=390,y=400,width=110,height=40)

self.btn\_clear=Button(self.root,text="Clear",font=("goudy old style",15,"bold"),bg="#607d8b",fg="white",cursor="hand2",command=self.clear)

self.btn\_clear.place(x=510,y=400,width=110,height=40)

#=======Search panel=======>

self.var\_search=StringVar()

lbl\_search\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,'bold'),bg="white").place(x=720,y=60)

txt\_search\_roll=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",15,'bold'),bg="lightyellow").place(x=870,y=60,width=180)

btn\_search=Button(self.root,text="Search",font=("goudy old style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).place(x=1070,y=60,width=120,height=28)

#======Content========>

self.C\_Frame=Frame(self.root,bd=2,relief=RIDGE)

self.C\_Frame.place(x=720,y=100,width=470,height=340)

scrolly=Scrollbar(self.C\_Frame,orient=VERTICAL)

scrollx=Scrollbar(self.C\_Frame,orient=HORIZONTAL)

self.courseTable=ttk.Treeview(self.C\_Frame,columns=("roll","name","email","gender","dob","contact","admission","course","state","city","pin","address"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)

scrollx.pack(side=BOTTOM,fill=X)

scrolly.pack(side=RIGHT,fill=Y)

scrollx.config(command=self.courseTable.xview)

scrolly.config(command=self.courseTable.yview)

self.courseTable.heading("roll",text="Roll No.")

self.courseTable.heading("name",text="Name")

self.courseTable.heading("email",text="Email")

self.courseTable.heading("gender",text="Gender")

self.courseTable.heading("dob",text="D.O.B")

self.courseTable.heading("contact",text="Contact")

self.courseTable.heading("admission",text="Admission")

self.courseTable.heading("course",text="Course")

self.courseTable.heading("state",text="State")

self.courseTable.heading("city",text="City")

self.courseTable.heading("pin",text="PIN")

self.courseTable.heading("address",text="Address")

self.courseTable["show"]='headings'

self.courseTable.column("roll",width=100)

self.courseTable.column("name",width=100)

self.courseTable.column("email",width=100)

self.courseTable.column("gender",width=100)

self.courseTable.column("dob",width=100)

self.courseTable.column("contact",width=100)

self.courseTable.column("admission",width=100)

self.courseTable.column("course",width=100)

self.courseTable.column("state",width=100)

self.courseTable.column("city",width=100)

self.courseTable.column("pin",width=100)

self.courseTable.column("address",width=100)

self.courseTable.pack(fill=BOTH,expand=1)

self.courseTable.bind("<ButtonRelease-1>",self.get\_data)

#self.show()

#=========function=========>

def clear(self):

self.show()

self.var\_roll.set(""),

self.var\_name.set(""),

self.var\_email.set(""),

self.var\_gender.set("Select Gender"),

self.var\_dob.set(""),

self.var\_contact.set(""),

self.var\_a\_date.set(""),

self.var\_course.set("Select"),

self.var\_state.set(""),

self.var\_city.set(""),

self.var\_pin.set(""),

self.txt\_address.delete("1.0",END)

self.txt\_roll.config(state=NORMAL)

self.var\_search.set("")

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll No. should be required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please select Student from the list first",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete ?",parent=self.root)

if op==True:

cur.execute("delete from student where roll=?",(self.var\_roll.get(),))

con.commit()

con.close()

messagebox.showinfo("Delete","student deleted Successfully",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def get\_data(self,ev):

self.txt\_roll.config(state="readonly")

r=self.courseTable.focus()

content=self.courseTable.item(r)

row=content["values"]

self.var\_roll.set(row[0]),

self.var\_name.set(row[1]),

self.var\_email.set(row[2]),

self.var\_gender.set(row[3]),

self.var\_dob.set(row[4]),

self.var\_contact.set(row[5]),

self.var\_a\_date.set(row[6]),

self.var\_course.set(row[7]),

self.var\_state.set(row[8]),

self.var\_city.set(row[9]),

self.var\_pin.set(row[10]),

self.txt\_address.delete("1.0",END)

self.txt\_address.insert(END,row[11])

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll Number should be required",parent=self.root)

elif self.var\_name.get()=="":

messagebox.showerror("Error","Name should be required",parent=self.root)

elif self.var\_email.get()=="":

messagebox.showerror("Error","Email should be required",parent=self.root)

elif self.var\_gender.get()=="Select Gender":

messagebox.showerror("Error","Gender should be required",parent=self.root)

elif self.var\_dob.get()=="":

messagebox.showerror("Error","D.O.B should be required",parent=self.root)

elif self.var\_contact.get()=="":

messagebox.showerror("Error","Contact should be required",parent=self.root)

elif self.var\_a\_date.get()=="":

messagebox.showerror("Error","Admission date should be required",parent=self.root)

elif self.var\_course.get()=="Select":

messagebox.showerror("Error","Course should be required",parent=self.root)

elif self.var\_state.get()=="":

messagebox.showerror("Error","State should be required",parent=self.root)

elif self.var\_city.get()=="":

messagebox.showerror("Error","City should be required",parent=self.root)

elif self.var\_pin.get()=="":

messagebox.showerror("Error","Pin should be required",parent=self.root)

elif self.txt\_address.get('1.0',END).strip()=="":

messagebox.showerror("Error","Description should be required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","Roll Number Already Present",parent=self.root)

else:

cur.execute("insert into student(roll,name,email,gender,dob,contact,admission,course,state,city,pin,address) values(?,?,?,?,?,?,?,?,?,?,?,?)",(

self.var\_roll.get(),self.var\_name.get(),self.var\_email.get(),self.var\_gender.get(),self.var\_dob.get(),

self.var\_contact.get(),self.var\_a\_date.get(),self.var\_course.get(),self.var\_state.get(),self.var\_city.get(),self.var\_pin.get(),

self.txt\_address.get("1.0",END)

))

con.commit()

messagebox.showinfo("Success","Student Added Successfully",parent=self.root)

self.show()

con.close()

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def update(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll No. should be required",parent=self.root)

elif self.var\_name.get()=="":

messagebox.showerror("Error","Name should be required",parent=self.root)

elif self.var\_email.get()=="":

messagebox.showerror("Error","Email should be required",parent=self.root)

elif self.var\_gender.get()=="Select Gender":

messagebox.showerror("Error","Gender should be required",parent=self.root)

elif self.var\_dob.get()=="":

messagebox.showerror("Error","D.O.B should be required",parent=self.root)

elif self.var\_contact.get()=="":

messagebox.showerror("Error","Contact should be required",parent=self.root)

elif self.var\_a\_date.get()=="":

messagebox.showerror("Error","Admission date should be required",parent=self.root)

elif self.var\_course.get()=="Select":

messagebox.showerror("Error","Course should be required",parent=self.root)

elif self.var\_state.get()=="":

messagebox.showerror("Error","State should be required",parent=self.root)

elif self.var\_city.get()=="":

messagebox.showerror("Error","City should be required",parent=self.root)

elif self.var\_pin.get()=="":

messagebox.showerror("Error","Pin should be required",parent=self.root)

elif self.txt\_address.get('1.0',END).strip()=="":

messagebox.showerror("Error","Description should be required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Select Student From List",parent=self.root)

else:

cur.execute("update student set name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,city=?,pin=?,address=? where roll=?",(

self.var\_name.get(),self.var\_email.get(),self.var\_gender.get(),self.var\_dob.get(),

self.var\_contact.get(),self.var\_a\_date.get(),self.var\_course.get(),self.var\_state.get(),self.var\_city.get(),self.var\_pin.get(),

self.txt\_address.get("1.0",END),self.var\_roll.get()

))

con.commit()

con.close()

messagebox.showinfo("Success","student Update Successfully",parent=self.root)

self.clear()

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def show(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from student")

rows=cur.fetchall()

self.courseTable.delete(\*self.courseTable.get\_children())

for row in rows:

self.courseTable.insert('',END,values=row)

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def fetch\_course(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select name from course")

rows=cur.fetchall()

if len(rows)>0:

for row in rows:

self.course\_list.append(row[0])

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute(f"select \* from student where roll=?",(self.var\_search.get(),))

row=cur.fetchone()

if row!=None:

self.courseTable.delete(\*self.courseTable.get\_children())

self.courseTable.insert('',END,values=row)

con.close()

else:

self.show()

#messagebox.showerror("Error","No record found",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=studentClass(root)

root.mainloop()

## Result Page:

from tkinter import \*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class resultClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====Title====>

title=Label(self.root,text="Add Student Result",font=("goudy old style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,relwidth=1,height=50)

#=======Variables=======>

self.var\_roll=StringVar()

self.var\_name=StringVar()

self.var\_course=StringVar()

self.var\_marks=StringVar()

self.var\_full\_marks=StringVar()

self.roll\_list=[]

self.fetch\_roll()

#=======widgets============>

lbl\_select=Label(self.root,text="Select Student",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=100)

self.txt\_student=ttk.Combobox(self.root,textvariable=self.var\_roll,values=self.roll\_list,font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_student.place(x=280,y=100,width=200)

self.txt\_student.set("Select")

btn\_search=Button(self.root,text="Search",font=("goudy old style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).place(x=500,y=100,width=100,height=28)

lbl\_name=Label(self.root,text="Name",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=160)

txt\_name=Entry(self.root,textvariable=self.var\_name,font=("goudy old style",20,'bold'),bg="lightyellow",state="readonly").place(x=280,y=160,width=320)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=220)

txt\_course=Entry(self.root,textvariable=self.var\_course,font=("goudy old style",20,'bold'),state="readonly",bg="lightyellow").place(x=280,y=220,width=320)

lbl\_marks\_ob=Label(self.root,text="Marks Obtained",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=280)

txt\_marks=Entry(self.root,textvariable=self.var\_marks,font=("goudy old style",20,'bold'),bg="lightyellow").place(x=280,y=280,width=320)

lbl\_full\_marks=Label(self.root,text="Full Marks",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=340)

txt\_full\_marks=Entry(self.root,textvariable=self.var\_full\_marks,font=("goudy old style",20,'bold'),bg="lightyellow").place(x=280,y=340,width=320)

#=====Buttons===========>

btn\_submit=Button(self.root,text="Submit",font=("time new roman",15),bg="lightgreen",activebackground="lightgreen",cursor="hand2",command=self.add).place(x=300,y=420,width=120,height=35)

btn\_clear=Button(self.root,text="clear",font=("time new roman",15),bg="lightgray",activebackground="lightgray",cursor="hand2",command=self.clear).place(x=430,y=420,width=120,height=35)

#============image===========>

self.bg\_img=Image.open("images/result.jpg")

self.bg\_img=self.bg\_img.resize((500,300),Image.Resampling.LANCZOS)

self.bg\_img=ImageTk.PhotoImage(self.bg\_img)

self.lbl\_bg=Label(self.root,image=self.bg\_img).place(x=650,y=100)

#=========function==========>

def fetch\_roll(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select roll from student")

rows=cur.fetchall()

if len(rows)>0:

for row in rows:

self.roll\_list.append(row[0])

con.close()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select name,course from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row!=None:

self.var\_name.set(row[0])

self.var\_course.set(row[1])

con.close()

else:

messagebox.showerror("Error","No record found",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_name.get()=="":

messagebox.showerror("Error","Please first search student record",parent=self.root)

else:

cur.execute("select \* from results where roll=? and course=?",(self.var\_roll.get(),self.var\_course.get()))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","Result Already Present",parent=self.root)

else:

if self.var\_marks.get()=="" or self.var\_full\_marks.get()=="":

messagebox.showerror("Error","Please Fill The Marks",parent=self.root)

else:

per=(int(self.var\_marks.get())\*100)/int(self.var\_full\_marks.get())

cur.execute("insert into results(roll,name,course,marks\_ob,full\_marks,per) values(?,?,?,?,?,?)",(

self.var\_roll.get(),self.var\_name.get(),self.var\_course.get(),self.var\_marks.get(),self.var\_full\_marks.get(),str(per)

))

con.commit()

con.close()

messagebox.showinfo("Success","Result Added Successfully",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def clear(self):

self.var\_roll.set("Select"),

self.var\_name.set(""),

self.var\_course.set(""),

self.var\_marks.set(""),

self.var\_full\_marks.set("")

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=resultClass(root)

root.mainloop()

## Report Page:

from tkinter import \*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class reportClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====Title====>

title=Label(self.root,text="View Student Result",font=("goudy old style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,relwidth=1,height=50)

#===========Search===========>

self.var\_search=StringVar()

self.var\_id=""

lbl\_search=Label(self.root,text="Search By Roll No.",font=("goudy old style",20,"bold"),bg="white").place(x=300,y=100)

txt\_search=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",20),bg="lightyellow").place(x=530,y=100,width=150)

btn\_search=Button(self.root,text="Search",font=("goudy old style",15,"bold"),bg="#03a9f4",fg="white",cursor="hand2",command=self.search).place(x=700,y=100,width=100,height=35)

btn\_clear=Button(self.root,text="Clear",font=("goudy old style",15,"bold"),bg="gray",fg="white",cursor="hand2",command=self.clear).place(x=810,y=100,width=100,height=35)

#=========result\_labels=========>

lbl\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=150,y=230,width=150,height=50)

lbl\_name=Label(self.root,text="Name",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=300,y=230,width=150,height=50)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=450,y=230,width=150,height=50)

lbl\_marks=Label(self.root,text="Marks Obtained",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=600,y=230,width=150,height=50)

lbl\_full=Label(self.root,text="Total Marks",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=750,y=230,width=150,height=50)

lbl\_per=Label(self.root,text="Percentage",font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE).place(x=900,y=230,width=150,height=50)

self.roll=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.roll.place(x=150,y=280,width=150,height=50)

self.name=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.name.place(x=300,y=280,width=150,height=50)

self.course=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.course.place(x=450,y=280,width=150,height=50)

self.marks=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.marks.place(x=600,y=280,width=150,height=50)

self.full=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.full.place(x=750,y=280,width=150,height=50)

self.per=Label(self.root,font=("goudy old style",15,'bold'),bg="white",bd=2,relief=GROOVE)

self.per.place(x=900,y=280,width=150,height=50)

#==========Button Delete=========>

btn\_delete=Button(self.root,text="Delete",font=("goudy old style",15,"bold"),bg="red",fg="white",cursor="hand2",command=self.delete).place(x=500,y=350,width=150,height=35)

#=========Function===========>

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_search.get()=="":

messagebox.showerror("Error","Roll No. should be required",parent=self.root)

else:

cur.execute("select \* from results where roll=?",(self.var\_search.get(),))

row=cur.fetchone()

if row!=None:

self.var\_id=row[0]

self.roll.config(text=row[1])

self.name.config(text=row[2])

self.course.config(text=row[3])

self.marks.config(text=row[4])

self.full.config(text=row[5])

self.per.config(text=row[6])

con.close()

else:

messagebox.showerror("Error","No record found",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def clear(self):

self.var\_id=""

self.var\_search.set("")

self.roll.config(text="")

self.name.config(text="")

self.course.config(text="")

self.marks.config(text="")

self.full.config(text="")

self.per.config(text="")

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_id=="":

messagebox.showerror("Error","Search Student Result First",parent=self.root)

else:

cur.execute("select \* from results where rid=?",(self.var\_id,))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Invalid Studnet Result",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete ?",parent=self.root)

if op==True:

cur.execute("delete from results where rid=?",(self.var\_id,))

con.commit()

con.close()

messagebox.showinfo("Delete","Result deleted Successfully",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=reportClass(root)

root.mainloop()

## Registration Page:

from tkinter import \*

from tkinter import ttk,messagebox

from PIL import Image,ImageTk

import sqlite3

import os

class Register:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Registration window")

self.root.geometry("1350x700+0+0")

self.root.config(bg="white")

#======bg\_Img============>

self.bg=ImageTk.PhotoImage(file="images/b2.jpg")

bg\_img=Label(self.root,image=self.bg).place(x=250,y=0,relheight=1,relwidth=1)

#======left\_Img============>

self.left=ImageTk.PhotoImage(file="images/side.png")

img\_left=Label(self.root,image=self.left).place(x=80,y=100,height=500,width=400)

#=========register Frame======>

frame1=Frame(self.root,bg="white")

frame1.place(x=480,y=100,width=700,height=500)

title=Label(frame1,text="REGISTER HERE",font=("times new roman",20,"bold"),bg="white",fg="green").place(x=50,y=30)

#=======Variables=======>

self.var\_fname=StringVar()

self.var\_lname=StringVar()

self.var\_contact=StringVar()

self.var\_email=StringVar()

self.var\_ques=StringVar()

self.var\_ans=StringVar()

self.var\_pass=StringVar()

self.var\_conpass=StringVar()

#-------------------Row 1------------------------------------

f\_name=Label(frame1,text="First Name",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)

txt\_fname=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_fname).place(x=50,y=130,width=250)

l\_name=Label(frame1,text="Last Name",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=100)

txt\_lname=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_lname).place(x=370,y=130,width=250)

#-------------------Row 2------------------------------------

contact=Label(frame1,text="Contact No.",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=170)

txt\_contact=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_contact).place(x=50,y=200,width=250)

email=Label(frame1,text="Email",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=170)

txt\_email=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_email).place(x=370,y=200,width=250)

#-------------------Row 3------------------------------------

question=Label(frame1,text="Security Question",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=240)

cmb\_question=ttk.Combobox(frame1,font=("times new roman",12),state="readonly",justify=CENTER,textvariable=self.var\_ques)

cmb\_question['values']=("Select","Your First Pet Name","Your Birth Place","Your Best Friend Name")

cmb\_question.place(x=50,y=270,width=250)

cmb\_question.current(0)

answer=Label(frame1,text="Answer",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=240)

txt\_answer=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_ans).place(x=370,y=270,width=250)

#-------------------Row 4------------------------------------

password=Label(frame1,text="Password",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=310)

txt\_password=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_pass).place(x=50,y=340,width=250)

cpassword=Label(frame1,text="Conform Password",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=310)

txt\_cpassword=Entry(frame1,font=("times new roman",15),bg="lightgray",textvariable=self.var\_conpass).place(x=370,y=340,width=250)

#---------------terms------------

self.var\_chk=IntVar()

chk=Checkbutton(frame1,text="I Agree The Terms & Conditions",variable=self.var\_chk,onvalue=1,offvalue=0,bg="white",font=("times new roman",12)).place(x=50,y=380)

self.btn\_img=ImageTk.PhotoImage(file="images/register.png")

btn\_register=Button(frame1,image=self.btn\_img,bd=0,cursor="hand2",command=self.register\_data).place(x=50,y=420)

btn\_login=Button(root,text="Sign In",font=("times new roman",20),bd=0,cursor="hand2",command=self.signin).place(x=200,y=460,width=180)

#=========function============>

def clear(self):

self.var\_fname.set("")

self.var\_lname.set("")

self.var\_contact.set("")

self.var\_email.set("")

self.var\_ques.set("Select")

self.var\_ans.set("")

self.var\_pass.set("")

self.var\_conpass.set("")

def register\_data(self):

if self.var\_fname.get()=="":

messagebox.showerror("Error","First Name is Required",parent=self.root)

elif self.var\_contact.get()=="":

messagebox.showerror("Error","Contact is Required",parent=self.root)

elif self.var\_email.get()=="":

messagebox.showerror("Error","Email is Required",parent=self.root)

elif self.var\_ques.get()=="Select":

messagebox.showerror("Error","Select the Question",parent=self.root)

elif self.var\_ans.get()=="":

messagebox.showerror("Error","Answer is Required",parent=self.root)

elif self.var\_pass.get()=="":

messagebox.showerror("Error","Password is Required",parent=self.root)

elif self.var\_conpass.get()=="":

messagebox.showerror("Error","Conform Password is Required",parent=self.root)

elif self.var\_pass.get()!=self.var\_conpass.get():

messagebox.showerror("Error","Password & Conform Password should be Same",parent=self.root)

elif self.var\_chk.get()==0:

messagebox.showerror("Error","Please Agree Our Terms & Condition",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=?",(self.var\_email.get(),))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","User Already Exist, Please Try With Another Email",parent=self.root)

else:

cur.execute("insert into employee(f\_name,l\_name,contact,email,question,answer,password) values(?,?,?,?,?,?,?)",

(self.var\_fname.get(),self.var\_lname.get(),self.var\_contact.get(),self.var\_email.get(),self.var\_ques.get(),

self.var\_ans.get(),self.var\_pass.get()

))

con.commit()

con.close()

messagebox.showinfo("Success","Register Successfull",parent=self.root)

self.clear()

self.signin()

except Exception as es:

messagebox.showerror("Error",f"Error due to:{str(es)}",parent=self.root)

def signin(self):

self.root.destroy()

os.system("python login.py")

#=======Main==============>

root=Tk()

obj=Register(root)

root.mainloop()

## Forget Password Page:

from tkinter import \*

from tkinter import ttk,messagebox

import sqlite3

class Forget\_pass:

def \_\_init\_\_(self,root,email):

self.root=root

self.email=email

self.root.title("Forget Password")

self.root.geometry("350x400+500+150")

self.root.config(bg="white")

self.root.focus\_force()

self.root.grab\_set()

t=Label(self.root,text="Forget Password",font=("times new roman",20,"bold"),bg="white",fg="red").place(x=0,y=10,relwidth=1)

#=============Forget password===============>

self.var\_ques=StringVar()

self.var\_ans=StringVar()

self.var\_Npass=StringVar()

question=Label(root,text="Security Question",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=70)

cmb\_question=ttk.Combobox(root,font=("times new roman",12),state="readonly",justify=CENTER,textvariable=self.var\_ques)

cmb\_question['values']=("Select","Your First Pet Name","Your Birth Place","Your Best Friend Name")

cmb\_question.place(x=50,y=100,width=250)

cmb\_question.current(0)

answer=Label(root,text="Answer",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=150)

txt\_answer=Entry(root,font=("times new roman",15),bg="lightgray",textvariable=self.var\_ans).place(x=50,y=180,width=250)

new\_pass=Label(root,text="New Password",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=230)

txt\_new\_pass=Entry(root,font=("times new roman",15),bg="lightgray",textvariable=self.var\_Npass).place(x=50,y=260,width=250)

but\_change\_password=Button(root,text="Reset Password",font=("times new roman",15,"bold"),bg="green",fg="white",cursor="hand2",command=self.forgetpassword).place(x=80,y=320)

#============Function=========>

def clear(self):

self.var\_ques.set("Select")

self.var\_ques.set("")

self.var\_Npass.set("")

def forgetpassword(self):

if self.var\_ans.get()=="Select" or self.var\_ans.get()=="" or self.var\_Npass.get()=="":

messagebox.showerror("Error","All Fields Are Required",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=? and question=? and answer=?",(self.email,self.var\_ques.get(),self.var\_ans.get()))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please Select the Correct Security Question / Enter Answer",parent=self.root)

else:

cur.execute("update employee set password=? where email=?",(self.var\_Npass.get(),self.email))

con.commit()

con.close()

messagebox.showinfo("Success","Your Password has been reset, Please Login with New password",parent=self.root)

self.clear()

self.root.destroy()

except Exception as es:

messagebox.showerror("Error",f"Error Due to: {str(es)}",parent=self.root)

#=======Main==============>

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

e=""

obj=Forget\_pass(root,e)

root.mainloop()

## Database Page:

import sqlite3

def create\_db():

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS course(cid INTEGER PRIMARY KEY AUTOINCREMENT,name text,duration text,charges text,description text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS student(roll INTEGER PRIMARY KEY AUTOINCREMENT,name text,email text,gender text,dob text,contact text,admission text,course text,state text,city text,pin text,address text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS results(rid INTEGER PRIMARY KEY AUTOINCREMENT,roll text,name text,course text,marks\_ob text,full\_marks text,per text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS employee(eid INTEGER PRIMARY KEY AUTOINCREMENT,f\_name text,l\_name text,contact text,email text,question text,answer text,password text)")

con.commit()

con.close()

create\_db()

## Code Efficiency: -

Reviewing of Code efficiency for a module is carried out after the module is successfully compiled and all the syntax errors eliminated. Code efficiency review is extremely cost-effective strategies for reduction in coding errors in order to produce high quality code. Normally, two types of efficiency are carried out on the code of a module -code optimization and code inspection. The procedure and final objective of these two efficiency techniques are very different as discussed below.

Optimization of Code:

Code optimization is an informal code analysis technique. In this technique, after a module has been coded, it is successfully compiled and all syntax errors are eliminated. Some members of the development team are given the code a few days before the optimization meeting to read and understand the code. Each member selects some test cases and simulates execution of the code by hand (i.e. trace execution through each statement and function execution). The main objectives of the optimization are to discover the algorithmic and logical errors in the code. The members note down their findings to discuss these in a optimization meeting where the coder of the module is also present.

Even though a code optimization is an informal analysis technique, several guidelines have evolved over the years for making this naïve technique more effective and useful. Of course, these guidelines are based on personal experience, common sense, and several subjective factors. Therefore are based on personal experience, common sense, and several subjective factors. Therefore, guidelines should be considered as examples rather than as rules to be applied dogmatically. Some of these guidelines are the following:

The team performing the code optimization should not be either too big or too small. Ideally, it should consist of three to seven members.

# Phase 6: System Testing

## Testing Phase: -

One of the purposes of the testing is to validate and verify the system. Verification means checking the system to ensure that it is doing what the function is supposed to do and Validation means checking to ensure that system is doing what the user wants it to do.

No program or system design is perfect; communication between the user and the designer is not always complete or clear, and time is usually short. The result is errors and more errors. Theoretically, a newly designed system should have all the pieces in working order, but in reality, each piece works independently. Now is the time to put all the pieces into one system and test it to determine whether it meets the user's requirements. This is the best chance to detect and correct errors before the system is implemented. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limits. If we implement the system without proper testing then it might cause the problems.

1. Communication between the user and the designer.
2. The programmer's ability to generate a code that reflects exactly the system specification.
3. The time frame for the design.

Theoretically, a new designed system should have all the pieces in working order, but in reality, each piece works independently. Now is the time to put all the pieces into one system and test it to determine whether it meets the requirements of the user.The process of system testing and the steps taken to validate and prepare a system for final implementation are:

## Levels of testing

The different types of testing are as follows:

UNIT TESTING

VALIDATION

INTEGRATION

SYSTEM

1. UNIT TESTING:

This is the smallest testable unit of a computer system and is normally tested using the white box testing. The author of the programs usually carries out unit tests.

1. INTEGRATION TESTING:

In integration testing, the different units of the system are integrated together to form the complete system and this type of testing checks the system as whole to ensure that it is doing what is supposed to do. The testing of an integrated system can be carried out top down, bottom-up, or big-bang. In this type of testing, some parts will be tested with white box testing and some with black box testing techniques. This type of testing plays very important role in increasing the systems productivity. We have checked our system by using the integration testing techniques.

1. SYSTEM TESTING:

A part from testing the system to validate the functionality of software against the requirements, it is also necessary to test the non-functional aspect of the system. Some examples of non-functional tools include tests to check performance, data security, usability/user friendliness, volume, load/stress that we have used in our project to test the various modules.

System testing consists of the following steps:

* + 1. Program(s) testing.
    2. String testing.
    3. System testing.
    4. System documentation.
    5. User acceptance testing.

1. FIELD TESTING:

This is a special type of testing that may be very important in some projects. Here the system is tested in the actual operational surroundings. The interfaces with other systems and the real world are checked. This type of testing is very rarely used. So far our project is concerned; we haven't tested our project using the field testing.

1. ACCEPTANCE TESTING:

After the developer has completed all rounds of testing and he is satisfied with the system, then the user takes over and re-tests the system from his point of view to judge whether it is acceptable according to some previously identified criteria. This is almost always a tricky situation in the project because of the inherent conflict between the developer and the user. In this project, it is the job of the bookstores to check the system that whether the made system fulfills the goals or not.

Why System Testing?

Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. Inadequate testing results in two types of problems:

* 1. The time lag between the cause and the appearance of the problem.
  2. The effect of system errors on the files and records within the system.

Activity Network for System Testing

The test plan entails the following activities:

* 1. Prepare test plan.
  2. Specify conditions for user acceptance testing.
  3. Prepare test data for program testing.
  4. Prepare test data for transaction path testing.
  5. Plan user training.
  6. Compile/assemble programs.
  7. Prepare job performance aids.
  8. Prepare operational documents.

PREPARE TEST:

A workable test plan must be prepared in accordance with established design specifications. It includes the following items:

* + - Outputs expected from the system.
    - Criteria for evaluating outputs.
    - A volume of test data.
    - Procedure for using test data.
    - Personnel and training requirements.

SPECIFY CONDITIONS FOR USER ACCEPTANCE TESTING

Planning for user acceptance testing calls for the analyst and the user to agree on conditions for the test.

PREPARE TEST DATA FOR PROGRAM TESTING

As each program is coded, test data are prepared and documented to ensure that all aspects of the program are properly tested.

PREPARE TEST DATA FOR TRANSACTION PATH TESTING

This activity develops the data required for testing every condition and transactions to be introduced into the system. The path of each transaction from origin to destination is carefully tested reliable results.

PLAN USER TRAINING

User training is designed to prepare the user for testing and converting the system. User involvement and training take place parallel with programming for three reasons:

* The system group has time available to spend on training while the programs are being written.
* Initiating a user-training program gives the systems group a clearer image of the user's interest in the new system.
* A trained user participates more effectively in system testing.

The training plan is followed by preparation of the user training manual and other text materials.

COMPILE / ASSEMBLE PROGRAMS

All programs have to be compiled / assembled for testing.

PREPARE JOB PERFORMANCE AIDS

In this activity the materials to be used by personnel to run the system are specified and scheduled. This includes a display of materials.

PREPARE OPERATIONAL DOCUMENTS

During the test plan stage, all operational documents are finalized including copies of the operational formats required by the candidate system.

SYSTEMS TESTING

The computer department to ensure that the system functions as specified does this testing. This testing is important to ensure that a working system is handed over to the user for acceptance testing.

ACCEPTANCE TESTING

The user to ensure that the system functions, as the user actually wanted performs this testing. With prototyping techniques, this stage becomes very much a formality to check the accuracy and completeness of processing. The screen layouts and output should already have been tested during the prototyping phase.

An error in the program code can remain undetected indefinitely. To prevent this from happening the code was tested at various levels. To successfully test a system, each condition, and combinations of conditions had to be tested. Each program was tested and linked to other programs. This unit of program is tested and linked to other units and so on until the complete system has been tested.

The purpose of testing is to ensure that each program is fully tested. To do so a test plan had to be created. The test plan consists of a number of test runs such as the valid paths through the code, and the exception and error handling paths. For each test run there is a list of conditions tested, the test data used and the result expected. The test plan was then reviewed to check that each path through the code is tested correctly. It is the responsibility of the programmer to collect the data that will produce the required test condition.

## Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TEST DATA  Student Management System | | | | | |
|  | Test Date | 25 Mar 2024 | Programmer name: | Subodh Kumar | | |
|  | Tested By: | Subodh Kumar | Project Name | STUDENT MANAGEMENT SYSTEM | | |
|  |  |  |  |  | | |
|  | Test Cases for Login Page | | | | |  |
|  | User ID the field is required to match set value | | | | | |
|  | Password the field is required to match set value | | | | | |
|  | |  | | --- | | LOGIN |     Will direct to home page after validation | | | | | |
|  | Positive Test cases for Login Page | | | | | |
| T.C ID | PRE-CONDITION | T.C  DESCRIPTION | T.C DATA | EXPECTED | ACTUAL | RESULT |
| 1 | User should be on Login page | Check the  functionality of  User ID | king@gmail.com | Will accept only  Email ID which is set in System | Ok | Pass |
| 2 | User should be on Login page | Check the  functionality of  Password | 1234 | Will accept only  Value which is set in System | Ok | Pass |
|  | Negative Test cases for User Details form | | | | | |
| T.C ID | PRE-CONDITION | T.C  DESCRIPTION | T.C DATA | EXPECTED | ACTUAL | RESULT |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | User should be on Login page | Check the  functionality of  User ID | ram@gmail.com | Will accept only  Email ID which is set in System | INVALID USER NAME AND PASSWORD | Pass |
| 2 | User should be on Login page | Check the  functionality of  Password | 11111 | Will accept only  Value which is set in System | INVALID USER NAME AND PASSWORD | Pass |

**Test Case of Registration Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Registration** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Registration Panel | Check The Functionality Of First Name Field | Maddy | Will Accept Any Length of String | Ok | Pass |
| **2** | User in Registration Panel | Check The Functionality Of Last Name Field | Saxena | Will Accept Any Length of String | Ok | Pass |
| **3** | User in Registration Panel | Check The Functionality Of Contact No Field | 9873454565 | Will Accept Any Length of String | Ok | Pass |
| **4** | User in Registration Panel | Check The Functionality Of Email Field | maddy@email.com | Will Accept Any Length of String | Ok | Pass |
| **5** | User in Registration Panel | Check The Functionality Of Security Question Field | Your Best Friend Name | Will Accept Question in Combobox | Ok | Pass |
| **6** | User in Registration Panel | Check The Functionality Of Answer Field | Akash | Will Accept Any Length of String | Ok | Pass |
| **7** | User in Registration Panel | Check The Functionality Of Password Field | 1234 | Will Accept Any Integer Value | Ok | Pass |
| **8** | User in Registration Panel | Check The Functionality Of Conform Password Field | 1234 | Will Accept Value of password Field | Ok | Pass |
| **Negative Test cases of Registration** | | | | | | |
| **1** | User in Registration Panel | Check The Functionality Of First Name Field | **-** | First Name is Required | Rejected | Pass |
| **2** | User in Registration Panel | Check The Functionality Of Last Name Field | **-** | Optional | Rejected | Pass |
| **3** | User in Registration Panel | Check The Functionality Of Contact No Field | **-** | Contact is Required | Rejected | Pass |
| **4** | User in Registration Panel | Check The Functionality Of Email Field | **-** | Email is Required | Rejected | Pass |
| **5** | User in Registration Panel | Check The Functionality Of Security Question Field | Select | Select the Question | Rejected | Pass |
| **6** | User in Registration Panel | Check The Functionality Of Answer Field | **-** | Answer is Required | Rejected | Pass |
| **7** | User in Registration Panel | Check The Functionality Of Password Field | **-** | Password is Required | Rejected | Pass |
| **8** | User in Registration Panel | Check The Functionality Of Conform Password Field | **-** | Conform Password is Required | Rejected | Pass |

**Test Case of Forget Password Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Forget password** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Forget password Panel | Check The Functionality Of Security Question Field | Your Best Friend Name | Will Accept Question in Combobox | Ok | Pass |
| **2** | User in Forget password Panel | Check The Functionality Of Answer Field | Akash | Will Accept Any Length of String | Ok | Pass |
| **3** | User in Forget password Panel | Check The Functionality Of Password Field | 123456 | Will Accept Any Length of String | Ok | Pass |
| **Negative Test cases of Forget password** | | | | | | |
| **1** | User in Forget password Panel | Check The Functionality Of Security Question Field | Select | All Fields are Required | Rejected | Pass |
| **2** | User in Forget password Panel | Check The Functionality Of Answer Field | **-** | All Fields are Required | Rejected | Pass |
| **3** | User in Forget password Panel | Check The Functionality Of Password Field | **-** | All Fields are Required | Rejected | Pass |

**Test Case of Course Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Course** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Course Page | Check The Functionality Of Course Name Field | Python | Will Accept Any Length of String | Ok | Pass |
| **2** | User in Course Page | Check The Functionality Of Duration Field | 6 Months | Will Accept Any Length of String | Ok | Pass |
| **3** | User in Course Page | Check The Functionality Of Charges Field | 10000 | Will Accept Any Integer Value | Ok | Pass |
| **4** | User in Course Page | Check The Functionality Of Description Field | Python is an **interpreted, object-oriented, high-level programming language** with dynamic semantics. | Will Accept Any Length of String | Ok | Pass |
| **Negative Test cases of Course** | | | | | | |
| **1** | User in Course Page | Check The Functionality Of Course Name Field | **-** | Course Name Should be Required | Rejected to Save /Update | Pass |
| **2** | User in Course Page | Check The Functionality Of Duration Field | **-** | Duration Should be Required | Rejected to Save /Update | Pass |
| **3** | User in Course Page | Check The Functionality Of Charges Field | **-** | Charges Should be Required | Rejected to Save /Update | Pass |
| **4** | User in Course Page | Check The Functionality Of Description Field | **-** | Description Should be Required | Rejected to Save /Update | Pass |

**Test Case of Student Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Add Student** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Student Panel | Check The Functionality Of Roll No. Field | 23 | Will Accept Any Integer Value | Ok | Pass |
| **2** | User in Student Panel | Check The Functionality Of Name Field | Manish | Will Accept Any Length of String | Ok | Pass |
| **3** | User in Student Panel | Check The Functionality Of Contact Field | 9873454565 | Will Accept Any Integer Value | Ok | Pass |
| **4** | User in Student Panel | Check The Functionality Of Email Field | manish@email.com | Will Accept Any Length of String | Ok | Pass |
| **5** | User in Student Panel | Check The Functionality Of D.O.B Field | 09-04-1999 | Will Accept Date | Ok | Pass |
| **6** | User in Student Panel | Check The Functionality Of Admission Field | 10-04-2022 | Will Accept Date | Ok | Pass |
| **7** | User in Student Panel | Check The Functionality Of Gender Field | Male | Will Accept Option in Combobox | Ok | Pass |
| **8** | User in Student Panel | Check The Functionality Of Course Field | Java | Will Accept Option in Combobox | Ok | Pass |
| **9** | User in Student Panel | Check The Functionality Of State Field | Delhi | Will Accept Any Length of String | Ok | Pass |
| **10** | User in Student Panel | Check The Functionality Of City Field | New Delhi | Will Accept Any Length of String | Ok | Pass |
| **11** | User in Student Panel | Check The Functionality Of Pin Field | 110045 | Will Accept Any Integer Value | Ok | Pass |
| **12** | User in Student Panel | Check The Functionality Of Address Field | Mahavir Enclave | Will Accept Any Length of String | Ok | Pass |
| **Negative Test cases of Add Student** | | | | | | |
| **1** | User in Student Panel | Check The Functionality Of Roll No. Field | **-** | Roll No. Should be Required | Rejected to Save /Update | Pass |
| **2** | User in Student Panel | Check The Functionality Of Name Field | **-** | Name Should be Required | Rejected to Save /Update | Pass |
| **3** | User in Student Panel | Check The Functionality Of Contact Field | **-** | Contact Should be Required | Rejected to Save /Update | Pass |
| **4** | User in Student Panel | Check The Functionality Of Email Field | **-** | Email Should be Required | Rejected to Save /Update | Pass |
| **5** | User in Student Panel | Check The Functionality Of D.O.B Field | **-** | D.O.B Should be Required | Rejected to Save /Update | Pass |
| **6** | User in Student Panel | Check The Functionality Of Admission Field | **-** | Admission Should be Required | Rejected to Save /Update | Pass |
| **7** | User in Student Panel | Check The Functionality Of Gender Field | Select Gender | Gender Should be Required | Rejected to Save /Update | Pass |
| **8** | User in Student Panel | Check The Functionality Of Course Field | Select | Course Should be Required | Rejected to Save /Update | Pass |
| **9** | User in Student Panel | Check The Functionality Of State Field | **-** | State Should be Required | Rejected to Save /Update | Pass |
| **10** | User in Student Panel | Check The Functionality Of City Field | **-** | City Should be Required | Rejected to Save /Update | Pass |
| **11** | User in Student Panel | Check The Functionality Of Pin Field | **-** | Pin Should be Required | Rejected to Save /Update | Pass |
| **12** | User in Student Panel | Check The Functionality Of Address Field | **-** | Address Should be Required | Rejected to Save /Update | Pass |

**Test Case of Report Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Report** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Report Page | Check The Functionality Of Search by Roll No. Field | 12 | Will Accept Any Integer Value | Ok | Pass |
| **Negative Test cases of Report** | | | | | | |
| **1** | User in Report Page | Check The Functionality Of Search by Roll No. Field | **-** | Roll No. Should be Required | Rejected to Search | Pass |

**Test Case of Result Module**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Positive Test Case of Result** | | | | | | |
| **T.C ID** | **PRE-CONDITION** | **T.C DESCRIPTION** | **T.C DATA** | **EXPECTED** | **ACTUAL** | **RESULT** |
| **1** | User in Result Page | Check The Functionality Of Select Student Field | 12 | Will Accept Roll No. in Combobox | Ok | Pass |
| **2** | User in Result Page | Check The Functionality Of Name Field |  | Automatically Fill by Select Student Field | Ok | Pass |
| **3** | User in Result Page | Check The Functionality Of Course Field |  | Automatically Fill by Select Student Field | Ok | Pass |
| **4** | User in Result Page | Check The Functionality Of Marks Obtained Field | 500 | Will Accept Any Integer Value | Ok | Pass |
| **5** | User in Result Page | Check The Functionality Of Full Marks Field | 600 | Will Accept Any Integer Value | Ok | Pass |
| **Negative Test cases of Result** | | | | | | |
| **1** | User in Result Page | Check The Functionality Of Select Student Field | Select | Please first Search Student Record | Rejected to Submit | Pass |
| **2** | User in Result Page | Check The Functionality Of Name Field | **-** | Read Only | Rejected to Submit | Pass |
| **3** | User in Result Page | Check The Functionality Of Course Field | **-** | Read Only | Rejected to Submit | Pass |
| **4** | User in Result Page | Check The Functionality Of Marks Obtained Field | **-** | Please Fill the Marks | Rejected to Submit | Pass |
| **5** | User in Result Page | Check The Functionality Of Full Marks Field | **-** | Please Fill the Marks | Rejected to Submit | Pass |

## Why System Testing?

Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. Inadequate testing results in two types of problems:

1. The time lag between the cause and the appearance of the problem.
2. The effect of system errors on the files and records within the system.

Activity Network for System Testing

The test plan entails the following activities:

* 1. Prepare test plan.
  2. Specify conditions for user acceptance testing.
  3. Prepare test data for program testing.
  4. Prepare test data for transaction path testing.
  5. Plan user training.
  6. Compile/assemble programs.
  7. Prepare job performance aids.
  8. Prepare operational documents.

## PREPARE TEST:

A workable test plan must be prepared in accordance with established design specifications. It includes the following items:

* + - Outputs expected from the system.
    - Criteria for evaluating outputs.
    - A volume of test data.
    - Procedure for using test data.
    - Personnel and training requirements.

SPECIFY CONDITIONS FOR USER ACCEPTANCE TESTING

Planning for user acceptance testing calls for the analyst and the user to agree on conditions for the test.

PREPARE TEST DATA FOR PROGRAM TESTING

As each program is coded, test data are prepared and documented to ensure that all aspects of the program are properly tested.

PREPARE TEST DATA FOR TRANSACTION PATH TESTING

This activity develops the data required for testing every condition and transactions to be introduced into the system. The path of each transaction from origin to destination is carefully tested reliable results.

PLAN USER TRAINING

User training is designed to prepare the user for testing and converting the system. User involvement and training take place parallel with programming for three reasons:

* The system group has time available to spend on training while the programs are being written.
* Initiating a user-training program gives the systems group a clearer image of the user's interest in the new system.
* A trained user participates more effectively in system testing.

The training plan is followed by preparation of the user training manual and other text materials.

COMPILE / ASSEMBLE PROGRAMS

All programs have to be compiled / assembled for testing.

PREPARE JOB PERFORMANCE AIDS

In this activity the materials to be used by personnel to run the system are specified and scheduled. This includes a display of materials.

PREPARE OPERATIONAL DOCUMENTS

During the test plan stage, all operational documents are finalized including copies of the operational formats required by the candidate system.

## Verification and Validation (V&V)

The objectives of verification, validity activities are to assess and improve the quality of the work products generated during development and modification of the software. Quality depends upon the various attributes like correctness, completeness, consistency, reliability, usefulness, usability, efficiency and conformance to standards.

The terms verification and validation are used synonymously. These are defined as under: -

Verification: “Are we building the product right?”

Validation: “Are we building the right product?”

Verification activities include proving, testing, and reviews. Validation is the process of evaluating software at the end of the software development to ensure compliance with the software requirements. Testing is a common method of validation. Clearly, for high reliability we need to perform both activities. Together, they are often called V&V activities.

The major V&V activities for software development are inspection, reviews, and testing (both static and dynamic). The V&V plan identifies the different V&V tasks for the different phases and specifies how these tasks contribute to the project V&V goals. The methods to be used for performing these V&V activities, the responsibilities and milestones for each of these activities, inputs and outputs for each V&V task, and criteria for evaluating the outputs are also specified.

The two major V&V approaches are testing and inspections. Testing is an activity that can be generally performed only on code. It is an important activity and is discussed in detail in a later chapter. Inspection is a more general activity that can be applied to any work product, including code. Many of the V&V tasks are such that for them, an inspection type of activity is the only possible way to perform the tasks (e.g. trace ability and document evaluation). Due to this, inspections play a significant role in verification.

# Phase: System Implementation

## System implementation maintenance and review: -

As we know, creating software is one thing and the implementation of the created software is another. The process of implementing software is much difficult as compared to the task of creating the project. First we have to implement the software on a small scale for removing the bugs and other errors in the project and after removing them we can implement the software on a large scale. Before we think in terms of implementing the Software on a large basis, we must consider the Hardware requirements.

Whenever we develop software or project a certain hardware and software is being used by the programmer for developing the project. The hardware and software to be used by the programmer for developing the project should be such that it would result in the development of a project, which would satisfy all the basic needs for which the project has been created by the programmer. The Hardware should be such that cost constraints of the Client should also be taken into account without affecting the performance.

## Hardware Evaluation Factors: -

When we evaluate computer hardware, we should first investigate specific physical and performance characteristics for each hardware component to be acquired. These specific questions must be answered concerning many important factors. These hardware evaluation factors questions are summarized in the below figure.

Notice that there is much more to evaluating hardware than determining the fastest and cheapest computing device. For e.g. the question of possible obsolescence must be addressed by making a technology evaluation. The factor of ergonomics is also very important. Ergonomics is the science and technology that tries to ensure that computers and other technologies are "user-friendly", that is safe, comfortable and easy to use. Connectivity is another important evaluation factor, since so many computer systems are now interconnected within wide area or local area telecommunications networks.

1. Performance
2. Cost
3. Reliability
4. Availability
5. Compatibility
6. Modularity
7. Technology
8. Ergonomics
9. Connectivity
10. Environmental requirements
11. Software
12. Support

## Software Evaluation Factors: -

Software can be evaluated according to many factors similar to the hardware evaluation. Thus, the factors of performance, cost, reliability, compatibility, modularity, technology, ergonomics, and support should be used to evaluate proposed software acquisitions. In addition, however, the software evaluation factors are summarized in below figure. For e.g. some software packages require too much memory capacity and are notoriously slow, hard to use, or poorly documented. They are not a good selection for most end users, even if offered at attractive prices.

SOFTWARE EVALUATION FACTORS:

1. EFFICIENCY: Efficiency is the software a well-written system of computer instructions that does not use much memory capacity or CPU time?
2. FLEXIBILITY: can it handle its processing assignments easily without major modifications?
3. SECURITY: Security does it provide control procedures for errors, malfunctions and improper use?
4. LANGUAGE: Language do our computer programmers and users write it in a programming language that is used?
5. DOCUMENTATION: Documentation is the s/w well documented? Does it include helpful user instructions?
6. HARDWARE: Hardware does existing hardware have the features required to best use this software?
7. Other characteristics of hardware such as its performance, what about the cost, how much is reliable and etc.

Conversion and Training:

An important aspect of is to make sure that the new design is implemented to establish standards. The term implementation has different meanings, ranging form the conversion of a basic application to a complete replacement of a computer system. Implementation is used here to mean the process of converting a new or revise system into an operational one. Conversion is one aspect of implementation. Conversion means changing form one system to another. The objective is to put the tested system into operation while holding costs, risks, and personnel irritation to a minimum. It involves creating computer-compatible files, training the operation staff, and installing terminal and hardware. A critical aspect of conversion is not disrupting the functioning of the organization.

When a new system is used over and old, existing and running one, there are always compatibility errors. These errors are caused because of the lack of equipment or personnel to work the new system. Running any specified system at an organization does require some or other hardware or, in this case, software requirement as well.

Conversion is one aspect of implementation review & software maintenance.

There are three types of implementations:

1. Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, creating accurate files and verifying printouts for integrity.
2. Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems. Some large computer systems have taken as long as year to convert.
3. Implementation of a modified application to replace an existing one, using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files.

Training Needs:

Training needs refer to the gaining of knowledge required for running the system.

First of all the system is a computer based system therefore the person should have good knowledge about computer and its working.

He should know how to use software's on the computer.

For a better usage and working of the software the organization should appoint a person who has good knowledge of all the required software. The organization gets a person trained through different institutes present in the market. The training should be as per the above requirements.

Cost Estimation of the Project:

Cost in a project is due to the requirements for software, hardware, and human resources. Hardware resources are computer time, terminal time and memory required for the project. Software resources include the tools and compilers needed during development. The bulk of cost of software development is due to human resources needed. Cost estimates are determined in terms of person-months (PM).

Total No. Of Persons Involved In This Project:

1. Project Manager
2. Senior Programmer

Since this Project will complete in 6 weeks

Wage/hour=100

Working hours a day= 8

Total number of days=16

Total Wage = 100 x 8 x 16 x 2 = Rs. 25,600/- Installation Charges= Rs. 1000 Cost Estimate:

= Salary of Person Involved + Deployment Fees + Rs. 30,000(Desktop) = 25,600+1000+30,000\*

 Rs. 25,600/- to 56,600/-

## Gantt Chart: -

Gantt charts mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is special type of bar chart where each bar represents an activity. The bars are drawn along a timeline. The length of each bar is proportional to the duration of the time planned for the corresponding activity.

Gantt chart is a project scheduling technique. Progress can be represented easily in a Gantt chart, by coloring each milestone when completed. The project started in the month of December 2023 and end in the February 2024.

Security and Validation Checks:

In this project we have used following validation checks.

* While entering the data into the form it will check for the name of the client is properly filled & it should not be null.
* Whenever we enter the data for the new customer, company, or user will automatically check the details from the database tables and also generate the connection number automatically.
* Similarly in the complaint table complaint number will generate automatically.
* Entered text / number should not exceed the limit (width).

Almost for all fields we have used the validation for example if name of the fields requires the text type of data then it will check for the string and if the data is numeric then it will check if the number entered is proper numeric or not.

# Scope of future application

This Software will be developed for the Online interested in information about customer and inventory. Utmost care and back-up procedures must be established to ensure 100% successful implementation of the Customer and inventory Management System. In case of system failure, the organization should be in a position to process the transaction with another organization or if the worst comes to the worst, it should be in a position to complete it manually.

The decision to automate generally depends on the needs to have accurate, consistent and timely data in a variety of reporting formats. But the most important factor that should be considered is –

* Be sure to take future growth into consideration and evaluate whether the software package could be modified if the organization expands in the future or you need to revise the system.

* The next generation of Windows-based computerized systems would be changing the way owners and management think about running their businesses. It would a powerful tool that gives them more ways to get their financial Information so that they can better manage and grow their business.
* The next generation of systems would take advantage of the latest technologies, including Microsoft Windows>=2007 and Orcale11i to offer access and integrate with all aspects of a business. Keeping this in view we could develop systems that would work efficiently and integrate seamlessly in virtually any industry setting, even up to mid-sized corporations and divisions of larger organizations.

* Consider another setting, where the users are not comfortable on computers, for such users we can develop systems where the users can have their own personalized menu setup in the accounting software so that they no longer have to search around to find what they need.

* Now consider the Internet. This wide-open information exchange network comes with many benefits for businesses, including breaking down physical barriers as to where he/she can consult to the companies.

* Adding other capabilities can be added time to time. These are  Application may be installed client server.
  + Application can upgrade according to User Requirements with little changes made.
  + Application may be transferred to latest RDMBS like Oracle with little changes in current code. Use for Security purpose

NOTE: In a nutshell we have a lot of scope to further enhance the futures and functionalities of the proposed Solution

# Bibliography

* Foundation, P. S. (n.d.). The Python Tutorial. Retrieved from Python Documentation: https://docs.python.org/3/tutorial/
* Foundation., P. S. (n.d.). Tkinter-Python interface to Tk. Retrieved from Python Documentation:

* https://docs.python.org/3/library/tkinter.html

* freepick.com. (n.d.). Retrieved from freepik.com

* Jain, S. Core Python.

* Python Tutorial. (n.d.). Retrieved from TutorialPoint: https://www.tutorialspoint.com/python/index.htm

* Rao, N. Core Python.

# Glossary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No. | Module | Methods | | | Other |
| 1 | Tkinter ttk  messagebox | 1. Tk() 2. Geometry() 3. Resizable() 4. Mainloop() 5. .get() 6. .set() 7. .config() 8. .askyesno() 9. .bind() 10. Toplevel() | | | Widgets:   1. Button 2. Label 3. Entry 4. Radiobutton 5. Canvas:    * create\_image()    * create\_window()    * create\_text()    * canvas.delete() 6. Treeview    * delete()    * get\_children()    * insert()    * heading()    * column()    * .indentify\_row(event.y)  .item(.focus())     Container:   1. frame 2. labelframe 3. canvas Layouts: 4. place() 5. pack() 6. grid() |
| 2 | SQLite 3 | | 1. | sqlite3.connect(database="database name") | |
|  |  | | 2. | con.cursor() | |
|  |  | | 3. | con.commit() | |
|  |  | | 4. | con.rollback() | |
|  |  | | 5. | con.is\_connected() | |
|  |  | | 6. | cur.execute() | |
|  |  | | 7. | cur.fetchone() | |
|  |  | | 8. | cur.fetchall() | |
| 3 | PIL(Image,  ImageTk, ImageDraw) | | 1. 2. | image.open()  resize() | |
|  |  | | 3. | image.antialias | |
|  |  | | 4.  5. | imagetk.photoimage()  ImageDraw() | |
| 4 | Os | | 1. | destroy() | |
|  |  | | 2. | system() | |
| 5 | File Handling | | 1. | f=open(“file.txt”) | |
|  |  | | 2. | f.readline() | |
|  |  | | 3. | f.write() | |
|  |  | | 4. | f. close() | |
| 6 | String | |  | str.split(‘delimiter’) | |
| 7 | Datetime(date) | |  | .today()  .now() | |