COLLEGE MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

GOKULSHANKAR P	**- 71052002013**
NAVEEN S	**- 71052002033**
SHASHANK CHAKRAWARTHI	**- 71052002052**
VINOSHAN RAJ D	**- 71052002056**
JAYA SURYA R	**- 71052002303**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING
COIMBATORE INISTITUTE OF ENGINEERING AND TECHNOLOGY,

COIMBATORE

ANNA UNIVERSITY:COIMBATORE 641109

PRACTICAL EXAMINATION-2023

AUTONOMOUS: COIMBATORE 641 109

BONAFIDE CERTIFICATE

Certified that this project report **“COLLEGE MANAGEMENT SYSTEM”** is the bonafide work of **“GOKULSHANKAR P , NAVEEN S , SHASHANK CHAKRAWARTHI ,VINOSHAN RAJ D , JAYA SURYA R ”** who carried out the project work under my supervision.

Dr.K. PUSHPALATHA M.E., Ph.D.,	**Dr.K. PUSHPALATHA M.E., Ph.D.,**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Coimbatore Institute of Engineering and	Coimbatore Institute of Engineering and
Technology,	Technology,
Coimbatore – 641109.	Coimbatore – 641109

Submitted for the university examination held on _________ at Coimbatore Instituteof Engineering and Technology, Coimbatore – 641109

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We, have great and immense pleasure in expressing the acknowledgment to the numerous contributors for the success of this project work. They all deserve credit and thanks to carry out this project successfully.

We would like to record our sincere indebtedness and gratitude to our beloved Director, **Dr.K.A.Chinnaraju M.Sc.,M.B.A.,Ph.D.,** for his noteworthy efforts to enhance our professional dexterity and co-curricular excellence.

We are grateful to our respected Principal**, Dr.N.Nagarajan, M.E., Ph.D**.,for providing us with necessary facilities to carry out our project work.

. We express our sincere thanks to our Head of the Department, project coordinator **Dr.K.Pushpalatha M.E.,Ph.D.,** for her timely help and cooperation.

We are very much pleased to acknowledge our sincere thanks to our guide for **Dr.K.Pushpalatha M.E.,Ph.D.,** her guidance and valuable suggestions.

Finally,We extend our thanks to the management, faculty members, parents and our student friends for their support and encouragement and to all others, who extended their helping hands to us in the completion of our final year project.

PROJECT STUDENTS

Gokulshankar P	- 71052002013
Naveen S	- 71052002033
Shashank Chakrawarthi	- 71052002052
Vinoshan Raj D	- 71052002056
Jaya Surya R	- 71052002303

ABSTRACT

- The College Management System is a comprehensive software solution designed to streamline and automate various administrative and academic processes within a college or educational institution.
- The system aims to improve efficiency, accuracy, and communication among administrators, faculty members, and students.
- By leveraging technology, the College Management System facilitates tasks such as admission management, course enrollment, examination and grading, attendance tracking, financial management, and communication and collaboration.
- It provides user-friendly interfaces tailored to different user roles, ensuring ease of use and accessibility.
- The system incorporates robust security measures to protect sensitive data and complies with data protection regulations.
- With real-time data management and reporting capabilities, the College Management System empowers educational institutions to effectively manage their operations, enhance student experience, and foster a collaborative learning environment.
- In conclusion, the College Management System is a comprehensive solution designed to meet the unique requirements of educational institutions.
- It enables colleges to streamline operations, improve academic outcomes, and create a collaborative and efficient learning environment.

TABLE OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Overview

We have to design the Desktop application for “COLLEGE MANAGEMENT SYSTEM ” that will full fill following :

- There will be Two modes of Sign up
    - Admin
    - Visitor
- Creation of Course Record,Student List,Placement Record,new registration maintained by Admin.
- This will happen from anywhere (location independent).
- Keep & maintenance large number of records easily .
- Finding and Listing Record quickly.
- In a Visitor mode they Overview the college history,Fee details,Courses,Staffs Experiences,Placement details.

1.2 System Objective

This is a Desktop application allows us to access the whole information about the college, staffs, students, fees etc. This application provides a virtual tour of Campus.Here we will get the latest information about the students and staffs. This generic application designed for assisting the students of an institute regarding information on the courses, subjects, classes,infrastructure,fees and staffs. Here admi manage the student,staffs,Notices and Placement Company lists details.

1.3 Document Scope

- College information: Through this service one can access the complete information about the college campus such as courses vailable,admission procedure, placements, college events, achievements etc.

- Student tracking: Any company or any organization that want to check the summary about the student of the college, so that they will be able to choose the particular students for their campus placement And for that purpose they will be given a particular link through which they can access the information required.
- Notice Board: This facility notifies new events or upcoming events about the college.
- Events: It will give information about different events that will be conducted by college time to time. Information about these events will be updated by administrator.
- Information about staff: It will help in maintaining complete information about college faculty members such as their department, cadre, date of joining, salary, etc. Administrator will register new faculties and remove their account when they leave the college.
- Company List : This contains the number of students placed and then show the list of companies visited the campus.
- Change Password : Admin manages the student and staff login activites and they have acces to provide new passwords.

1.4 Description of Project

We identify several problems including unauthorized privilege escalation, incorrect use of cryptography, vulnerabilities to network threats, and poor software development processes. We show that only college administrator can start the system.administrator can search the particular student by his/her enrollment number or student idAnd we are adding notification module where administrator should add the notification with start and end date.

CHAPTER 2

SYSTEM ARCHITECTURE

2.1 System Components

The system components refer to the individual building blocks or modules that collectively contribute to the overall functionality of the system.

- Database
- Front-End Interfaces
- Back-End Modules

2.1.1 Database

The database plays a crucial role in storing and managing various types of data related to students, faculty, courses, attendance, examinations, finances, and more. The database serves as a centralized repository of information that can be accessed and manipulated by the system's components. Here are some key aspects of the database in a College Management System:

Relational Database Management System (RDBMS):

The database system used in a College Management System is typically a Relational Database Management System (RDBMS) such as MySQL, PostgreSQL, or Oracle. An RDBMS organizes data into tables with predefined relationships and allows for efficient querying, indexing, and manipulation of the data.

Database Design:

The database design involves defining the structure and relationships of the database tables. It includes creating tables, specifying columns, data types, primary keys, foreign keys, and establishing relationships between tables using referential integrity constraints.

Entity-Relationship (ER) Model:

The ER model is commonly used to represent the database schema and relationships between entities (such as students, courses, faculty) in the College Management System. It helps to visualize and plan the database structure before implementation.

Database Tables:

The database consists of multiple tables, each representing a specific entity or concept in the College Management System. For example, there may be tables for students, courses, faculty, fee records, and more. Each table has columns to store attributes of the entity, such as student ID, name, course code, etc.

Data Integrity:

Data integrity ensures the accuracy and consistency of data stored in the database. It is maintained through constraints such as primary keys, unique keys, foreign keys, and check constraints. These constraints enforce rules and relationships, preventing the insertion of invalid or inconsistent data.

Database Operations:

The College Management System performs various database operations, including inserting, updating, deleting, and retrieving data. These operations are carried out through MYSQL statements executed by the system components, such as PHP scripts.

Data Relationships and Joins:

The database establishes relationships between tables using primary key and foreign key constraints. This allows for joining related tables to retrieve data that spans multiple entities. For example, joining the student and course tables can provide information on which courses a student is enrolled in.

Database Indexing:

Indexing is used to improve the performance of database queries. It involves creating indexes on specific columns to speed up the retrieval of data. Indexing is commonly used on columns frequently used in search and filter operations, such as student ID, course code, or date.

Data Security and Access Control:

The database should implement proper security measures to protect sensitive data. This includes defining user roles and permissions, encrypting sensitive data, and implementing secure access controls to prevent unauthorized access or modification of data.

Database Backup and Recovery:

Regular database backups and a robust recovery mechanism are essential to protect against data loss. Periodic backups ensure that data can be restored in the event of hardware failures, system errors, or other unforeseen circumstances.

Proper database design, data integrity, and efficient data management are crucial for the effective functioning of a College Management System. The database serves as a reliable and structured repository of information, enabling the system to store, retrieve, and manipulate data accurately and efficiently.

2.1.2 Front-End Interfaces

Front-end interfaces in a College Management System are the user-facing components that allow users, such as administrators, faculty members, students, and parents/guardians, to interact with the system. These interfaces are designed to provide a user-friendly and intuitive experience. Here are some front-end interfaces found in a College Management System:

Login and Registration:

This interface allows users to authenticate themselves by entering their credentials, such as username and password, to access the system. It may also include a registration feature for new users to create an account.

Dashboard:

The dashboard serves as the main landing page after logging in. It provides an overview of important information and quick access to key functionalities, such as notifications, upcoming events, and important announcements.

User Profiles:

Each user, including administrators, faculty members, and students, typically has a dedicated profile page. This interface allows users to view and admins allows to update their personal information, profile picture, contact details, and other relevant information.

Course Management:

This interface enables faculty members to manage their courses. It may include features such as adding course materials, uploading assignments, posting announcements, and managing course schedules.

Student Management:

Administrators and faculty members can use this interface to view and manage student-related information. It may include features such as searching for students, viewing student profiles, managing enrollment, and generating reports.

Financial Management:

This interface is used by administrators to manage financial aspects such as fee collection, payment tracking, and generating financial reports. It may include features like fee payment portals and financial analytics.

These front-end interfaces should be designed with a user-centric approach, ensuring a clean and intuitive layout, responsive design for different devices, and accessibility considerations. The interfaces should be visually appealing, easy to navigate, and provide a seamless user experience to enhance productivity and user satisfaction.

2.1.3 Back-End Modules

In a College Management System, the back-end modules are responsible for handling the core functionalities and data management of the system. These modules typically consist of server-side components and database operations. Here are some common back-end modules found in a College Management System:

Admin Management:

This module handles user authentication, registration, and authorization. It includes functionalities such as user login, password encryption, user role management, and access control.

Database Management:

The database management module is responsible for interacting with the database system. It includes tasks such as establishing database connections, executing SQL queries, managing transactions, and handling database operations.

Student Management:

The student management module handles tasks related to student information. It includes functionalities such as student registration, admission management, enrollment, student profiles, and academic records management.

Faculty Management:

This module manages faculty-related operations. It includes functionalities such as faculty registration, course assignment, scheduling, performance evaluation, and faculty profiles management.

These back-end modules work together to ensure the smooth functioning of the College Management System. They handle the logic, data management, and business rules required to provide the desired functionalities to users. The modules interact with the database, process user requests, and ensure the integrity and security of the data.

2.2 Architecture Diagram

2.2.1 System Architecture Chart

There are two functional component of this systemi.e the admin, Visitor. In this

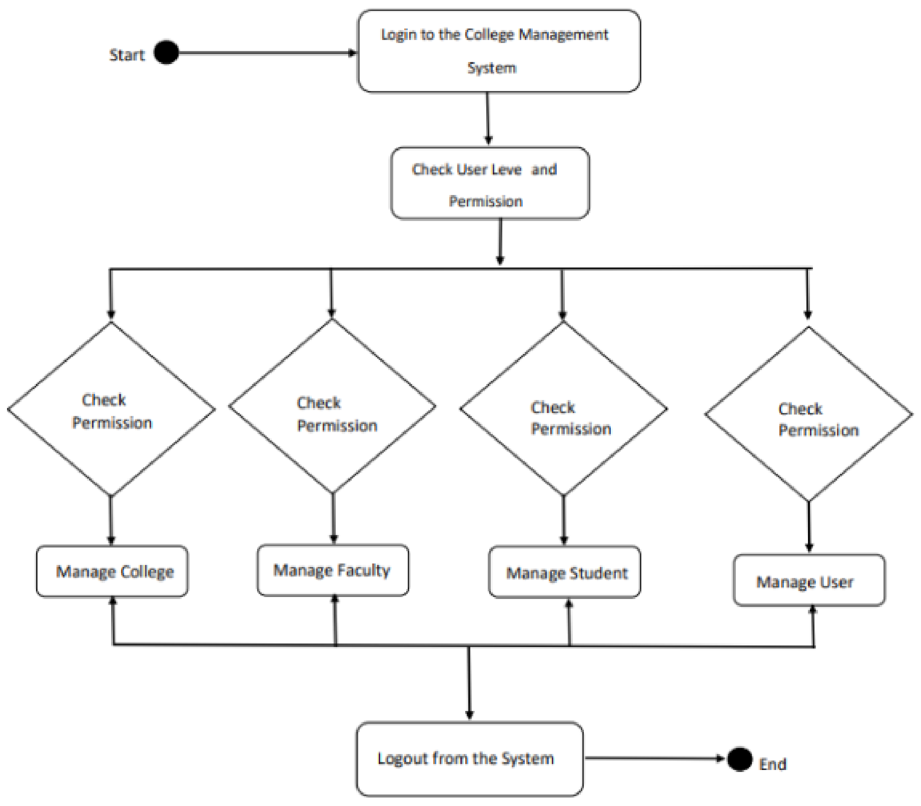


Fig 2.2.1 System Architecture Chart

system admin has highest priority.Main functionality of admin is to add the employee and manage the database dashboard. Admin gives the login credential to the student,faculty and employee.

2.2.2 Modules

The System is implemented in java and consist of two modules:

- Admin Module
- Visitor Module

Admin Module :

This is the main module in our college management system. It play very important role in the college management system. This module contains the main registration part of the employee in the college system. This is main task in our system because it is necessary to maintain the security. Admin create the accounts of employee using there Username and password. This account created are stored according to there department and designation. Admin have the highest priority in our college management system. Admin have the all access of the database. They have permission like view the information, edit the information, delete the information etc

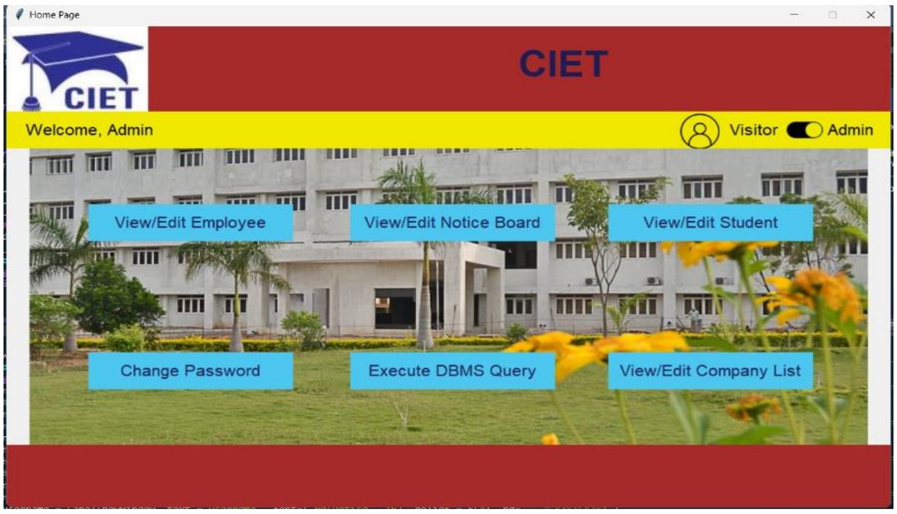


Fig 2.2.2 Admin Modulel

Visitor Module :

This module only contain the basic information about the collge , faculties etc... The system has facility to forgot there password. This module are further divided into submodule as :

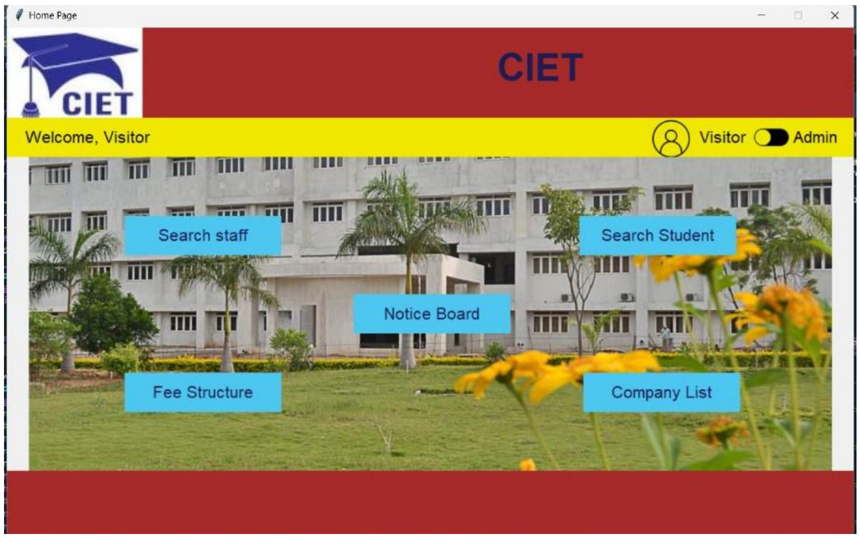


Fig 2.2.2 Visitor Module

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Functional Requirements :

Functional requirements specify the specific functionalities and features that the College Management System should provide. These requirements define what the system is expected to do and how it should behave in various scenarios.

3.1.1 Admin Management :

Registration: The system should allows admin to register with their personal information and create an account.

Login: Users should be able to securely log in to the system using their credentials provided by admin.

Password Management: The system allows only the admin to change the passwords.

Role-Based Access Control: The system should enforce different levels of access based on user roles, such as administrator, faculty, or student.

3.1.2 Student Management :

The student management module handles tasks related to student information. It includes functionalities such as student registration, enrollment, and academic records management.The system should store and display student information, including personal details, contact information, and academic records.

3.1.3 Faculty Management :

Faculty Profiles: The system should store and display faculty information,experiences including personal details, contact information, and academic records.

3.1.4 Placement :

It contains list of placement companies can be challenging as it varies depending on factors such as location, industry, and specific college or university affiliations.It also contains the record of Number of Students placed and working company .

3.1.5 Notices :

It refers to the functionality that handles the creation, distribution, and management of notices or announcements within the college or educational institution. It involves a centralized system for administrators or authorized users to publish notices and ensure that they are effectively communicated to the intended recipients.

3.2 Non-Functional Requirements :

It define the qualities and characteristics of a system that are not directly related to its specific functionalities but rather focus on the system's performance, usability, security, and other aspects. Here are some examples of non-functional requirements for a College Management System :

3.2.1Performance :

It directly impacts user satisfaction, system usability, and overall efficiency.The system can deliver a responsive, efficient, and reliable experience for users. Optimizing system performance ensures that administrative tasks, academic processes, and student interactions can be carried out seamlessly, contributing to an effective and productive college management environment.

3.2.2 Security:

It safeguards sensitive information, protects user privacy, and ensures the integrity of the system.The system can protect sensitive information, maintain data integrity, and ensure user privacy. Robust security measures instill confidence in users, establish trust in the system, and mitigate potential risks and vulnerabilities.

3.2.3 Scalability :

It enables the system to handle growth in terms of users, data, and workload without compromising performance. A scalable system ensures that the college management processes can effectively support a growing number of students, faculty, and administrative staff

3.2.4 Usability :

It focuses on enhancing the user experience, ease of use, and overall user satisfaction. A system that is user-friendly and intuitive promotes efficiency and productivity for students, faculty, and administrative staff.

CHAPTER 4

INSTALLATION AND CONFIGURATION

4.1 System Installation

System Installation for a College Management System typically involves the setup and configuration of both software and hardware components.

4.1.1 Hardware Requirements

The hardware requirements for a College Management System can vary based on factors such as the size of the institution, the number of users, and the specific functionalities and modules included in the system. Here are some general hardware requirements to consider:

Server:

A dedicated server or a server cluster may be required to host the College Management System. The server should have sufficient processing power, memory, and storage capacity to handle the system's expected workload and user concurrency. The server hardware should meet the recommended specifications provided by the system's software vendor.

Storage:

Sufficient storage capacity is needed to store the system's data, including student records, faculty information, course details, and administrative documents. Consider using a reliable storage solution such as hard disk drives (HDDs) or solid-state drives (SSDs) with adequate capacity and performance to handle the anticipated data growth.

Networking Equipment:

Networking equipment such as routers, switches, and network cables are required to establish connectivity between the server hosting the College Management System and client devices. The network infrastructure should provide sufficient bandwidth to support concurrent user access, data transfer, and communication within the system.

Client Devices:

The hardware requirements for client devices, such as computers or mobile devices used by students, faculty, and administrative staff, should be considered. These devices should meet the minimum specifications recommended by the system's software vendor. Common requirements include a modern web browser, sufficient memory and processing power, and an internet connection.

Backup Systems:

Implement a robust backup system to ensure data protection and disaster recovery. This may involve backup servers or storage devices to regularly back up the system's data. Consider using redundant storage configurations, such as RAID (Redundant Array of Independent Disks), to provide data redundancy and minimize the risk of data loss.

Security Infrastructure:

Hardware components related to system security should be considered, such as firewalls, intrusion detection and prevention systems, and security appliances. These components help protect the system from unauthorized access, malware, and other security threats.

Power and Cooling:

Ensure that the hardware infrastructure has adequate power supply and cooling mechanisms. This includes uninterruptible power supply (UPS) systems to provide backup power in case of outages, as well as cooling systems to maintain optimal temperature levels and prevent overheating of server rooms or data centers.

Requirements :

Disc Space	: 40 GB
PC Used	: IBM Compatible
Processor	: Pentium 3
Memory	: 512 MB RAM
File System	: 32 Bit

4.1.2 Software Requirements

The software requirements for a College Management System can vary depending on the specific system being implemented. However, here are some common software components and requirements to consider:

Operating System:

Determine the compatible operating system(s) for the College Management System. This may include Windows, Linux, or macOS. Ensure that the chosen operating system is supported by the system's software and meets the recommended version and configuration.

Web Server:

A web server is typically required to host the College Management System. Commonly used web servers include Apache, Nginx, or Microsoft Internet Information Services (IIS). Verify the compatibility of the chosen web server with the system's software and ensure it meets the recommended version.

Database Management System (DBMS):

The College Management System may require a database management system to store and manage data. Commonly used DBMS options include MySQL, PostgreSQL, or Oracle. Verify the compatibility of the chosen DBMS with the system's software and ensure it meets the recommended version.

Programming Languages and Frameworks:

The software requirements may specify certain programming languages and frameworks that the College Management System is built upon. For example, if the system is developed using PHP, ensure that the required version of PHP and any related frameworks (such as Laravel or CodeIgniter) are installed.

Client-Side Technologies:

The College Management System's user interface may rely on client-side technologies such as HTML, CSS, and JavaScript. Ensure that the required versions of these technologies are supported by the users' web browsers.

Additional Software Dependencies:

Check for any additional software dependencies that are required by the College Management System. These dependencies could include libraries, modules, or runtime environments necessary for the system's proper functioning. Ensure that all required dependencies are installed and configured correctly.

Integrated Development Environment (IDE):

If the system requires further development or customization, consider using an appropriate integrated development environment (IDE) for coding and debugging purposes. Examples include Visual Studio Code, PhpStorm, or Eclipse, depending on the programming languages and frameworks used.

Security Software:

Implement security software to protect the College Management System from threats such as malware, viruses, and unauthorized access. This may include antivirus software, firewalls, and intrusion detection systems. Regularly update and maintain the security software to ensure optimal protection.

Requirements :

• Operating System (Server Side)	: Windows XP.
• Operating System (Client Side)	: Windows XP.
• Client End Language	: Python-PIL
• Local Validation	: PHP
• Server Side Language	: PHP
• Library	: Pyhton-Pickler
• Database	: My Sql
• Server	: XAMPP server
• Platform	: Anaconda-Spyder
• Application	: Desktop Application

4.1.3 Installation Steps

The installation steps for a College Management System desktop application can depending on installation package. However, here is a general outline of the installation process:

Step 1 : Obtain the Installation Package.

Step 2 : Read the Documentation.

Step 3 : Run the Installer.

Step 4 : Choose the Installation Location.

Step 5 : Accept the License Agreement.

Step 6 : Select Components and Features.

Step 7 : Configure Application Settings.

Step 8 : Start the Installation Process.

Step 9 : Complete the Installation.

Step 10: Verify the Installation.

4.2 Database Configuration

It is important to consult the specific documentation or guidelines provided with your College Management System application for any additional configuration steps or database-specific instructions.

4.2.1 Database Setup

Admin maintains all the records and databases using Sql Language.Administrator store the records of Students and Faculties .They also maintain the Password Management and Login Credentials.

Sample Code :

```
--
-- Database: `nsec_db`
--
CREATE DATABASE IF NOT EXISTS `nsec_db` DEFAULT CHARACTER SET latin1
COLLATE latin1_swedish_ci;
USE `nsec_db`;
--
-- Table structure for table `company_list`
--
DROP TABLE IF EXISTS `company_list`;
CREATE TABLE `company_list` (
  `name` varchar(30) NOT NULL,
  `dept` text NOT NULL,
  `year` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Table structure for table `employee_table`

--

DROP TABLE IF EXISTS `employee_table`;

CREATE TABLE `employee_table` (

  `id` char(5) DEFAULT NULL,

  `name` char(30) NOT NULL,

  `desg` char(15) NOT NULL,

  `dept` char(10) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Table structure for table `global_values`

--

DROP TABLE IF EXISTS `global_values`;

CREATE TABLE `global_values` (

  `x` char(10) NOT NULL,

  `y` char(10) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;


--

-- Dumping data for table `global_values`

--
```

```
INSERT INTO `global_values` (`x`, `y`) VALUES
('admin', 'admin');
--
-- Table structure for table `notice_board`
--
DROP TABLE IF EXISTS `notice_board`;
CREATE TABLE `notice_board` (
  `id` char(5) NOT NULL,
  `topic` char(20) NOT NULL,
  `description` longtext NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Table structure for table `student_table`
--
DROP TABLE IF EXISTS `student_table`;
CREATE TABLE `student_table` (
  `id` char(5) DEFAULT NULL,
  `name` char(30) NOT NULL,
  `sem` int(11) NOT NULL,
  `stream` char(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Indexes for dumped tables
-- Indexes for table `company_list`
```

```
--
ALTER TABLE `company_list`
  ADD UNIQUE KEY `name` (`name`);
--
-- Indexes for table `employee_table`
--
ALTER TABLE `employee_table`
  ADD UNIQUE KEY `id` (`id`);
--
-- Indexes for table `global_values`
--
ALTER TABLE `global_values`
  ADD UNIQUE KEY `x` (`x`);
--
-- Indexes for table `notice_board`
--
ALTER TABLE `notice_board`
  ADD UNIQUE KEY `id` (`id`);
--
-- Indexes for table `student_table`
--
ALTER TABLE `student_table`
  ADD UNIQUE KEY `id` (`id`);
COMMIT;
```

4.2.2 Connection Configuration

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
/*!40101SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
```

4.3 System Configuration

System configuration for a College Management System desktop application involves setting up the necessary software and environment to ensure the smooth installation and operation of the application. Here are some key aspects to consider for system configuration:

Operating System:

Ensure that the operating system on the computer meets the minimum requirements specified by the College Management System desktop application. This may include the specific version of Windows, macOS, or Linux.

Hardware Requirements:

Verify that the computer hardware meets the recommended specifications for the College Management System desktop application. This may include the processor, memory (RAM), and storage capacity. Ensure that there is sufficient disk space available to install and run the application.

Software Dependencies:

Identify any software dependencies required by the College Management System desktop application. This may include runtime environments, libraries, or frameworks. Install and configure these dependencies according to the application's documentation or instructions.

Database Setup:

If the College Management System desktop application requires a database, set up the appropriate database management system (DBMS) and configure it to meet the application's requirements. Create the necessary database schema or import any provided database scripts.

Network Connectivity:

Ensure that the computer has a stable internet connection, especially if the College Management System desktop application requires online features, data synchronization, or updates. Verify that network configurations (such as proxy settings or firewall rules) do not hinder the application's connectivity.

Application Settings:

Configure any application-specific settings as required. This may include options related to language preferences, user interface themes, database connection details, or default behavior. Refer to the application's documentation or configuration files for guidance.

User Accounts and Roles:

Set up user accounts and roles within the College Management System desktop application as per the institution's requirements. Assign appropriate roles and permissions to users based on their responsibilities and access needs.

Testing and Validation:

Before deploying the College Management System desktop application in a production environment, thoroughly test its functionality and validate that all features are working as intended. Conduct user acceptance testing to ensure that the system meets the institution's requirements.

4.3.1 User Roles and Permissions

User roles and permissions in a College Management System desktop application help define the access levels and privileges granted to different users based on their roles or responsibilities within the system. Here are some common user roles and their corresponding permissions:

Administrator:

The administrator role typically has the highest level of access and control over the College Management System. Administrators have permissions to perform administrative tasks such as system configuration, user management, database management, and overall system maintenance.

Faculty:

Faculty members or teachers have specific permissions related to their roles in the college. They can access and manage course-related information, such as creating and editing course schedules and communicating with students.

Student:

Students have permissions to access their personal academic information and perform tasks related to their studies. This includes viewing course schedules, registering for courses and communicating with faculty members.

Staff/Non-Teaching Personnel:

Staff members or non-teaching personnel may have permissions to perform administrative tasks related to college operations. This can include tasks such as managing student records and providing support to students and faculty.

4.3.2 System Settings

It refer to the configurable options and preferences that allow users to customize the behavior and appearance of the application according to their needs. Here are some common system settings that can be included in a College Management System desktop application:

Language Preference:

Allow users to select their preferred language for the application's user interface. Provide a list of supported languages and enable users to switch between them.

Theme and Appearance:

Provide different themes or color schemes for the application's interface. Allow users to choose their preferred theme to customize the visual appearance of the application.

Date and Time Format:

Allow users to customize the display format for dates and times within the application. Provide options such as different date formats (e.g., DD/MM/YYYY, MM/DD/YYYY) and time formats (12-hour or 24-hour clock).

Notification Preferences:

Enable users to specify their notification preferences, such as whether they want to receive email notifications or in-app notifications for specific events, such as course updates, deadlines, or announcements.

Accessibility Options:

Include accessibility settings to accommodate users with specific needs. This may include options for adjusting font size, enabling high contrast mode, or providing screen reader compatibility.

Privacy and Security Settings:

Provide options for users to manage their privacy and security preferences, such as password change, two-factor authentication, and data sharing settings.

It is important to consider the specific requirements and preferences of the users and institution when determining the system settings for the College Management System desktop application. Conduct user surveys or feedback sessions to gather insights on the desired customization options and prioritize the most essential settings to include in the application.

CHAPTER 5

USER GUIDE

5.1 User Roles and Permissions

User roles and permissions in a College Management System desktop application help define the access levels and privileges granted to different users based on their roles or responsibilities within the system. Here are some common user roles and their corresponding permissions:

5.1.1 Administrators

The administrator role typically has the highest level of access and control over the College Management System. Administrators have permissions to perform administrative tasks such as system configuration, user management, database management, and overall system maintenance.

5.1.2 Faculty Members

Faculty members or teachers have specific permissions related to their roles in the college. They can access and manage course-related information, such as creating and editing course schedules and communicating with students.

5.1.3 Students

Students have permissions to access their personal academic information and perform tasks related to their studies. This includes viewing course schedules, registering for courses and communicating with faculty members.

5.2 User Interface Overview

The user interface (UI) of a College Management System desktop application plays a crucial role in providing an intuitive and user-friendly experience for the application users. Here is an overview of the key components and features that can be included in the UI of a College Management System desktop application:

- Navigation
- Dashboard

- Student Management
- Faculty Management
- Placements
- User Profile

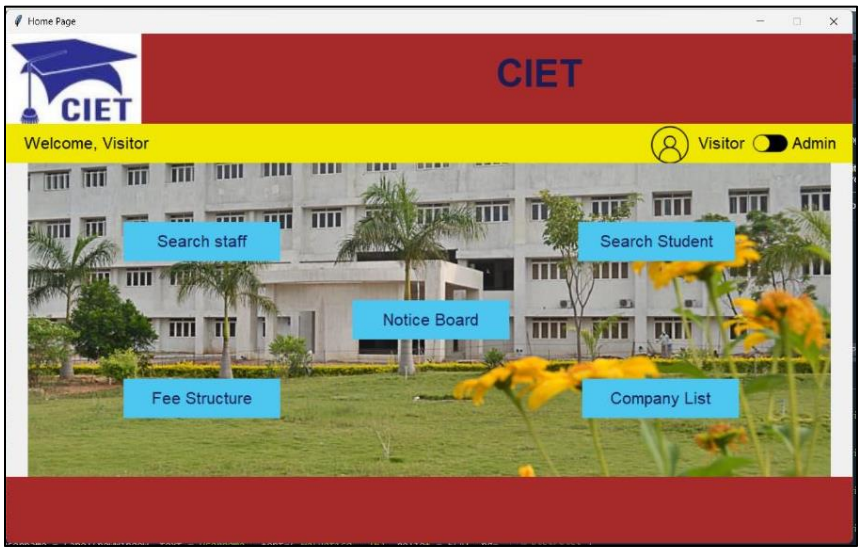


Fig 5.2 User Interface Overview

5.2.1 Navigation

The navigation in a College Management System desktop application is an essential element that allows users to access different modules and features of the system easily. Here is an example of a navigation structure for a College Management System desktop application:

Dashboard:

- Overview of key information, statistics, and shortcuts to important modules or features.

Students:

- View a list of students
- Add new student

- Search for a specific student
- Manage student details (personal information, contact details, enrollment status, etc.)
- View student grades and academic history

Faculty:

- View a list of faculty members
- Add new faculty member
- Search for a specific faculty member
- Manage faculty details (personal information, contact details, expertise, etc.)

Placements:

- View placement companies
- Post job opportunities
- Manage student applications
- Schedule interviews and track placement process
- Generate placement reports

Settings:

- User profile management (edit personal information, change password)
- Application settings (language preference, theme selection)
- Notification preferences (email notifications, in-app notifications)
- Backup and restore data
- Help and support resources

Logout:

- Allow users to securely log out of the application

5.2.2 Common Interface Elements

The Initial Page of the Apllication contains the Admin and Visitor options.Then it contains some common Interfaces :

- Search Staff
- Search Student
- Notice Board
- Fee Structure
- Company List

5.3 User Workflows

Initially it Contains the Admin mode and Visitor mode through the Admin mode the administrator maintains all the records of students and faculty members.Administrator is only responsible for making notices about the college events and maintains the records of regarding placement

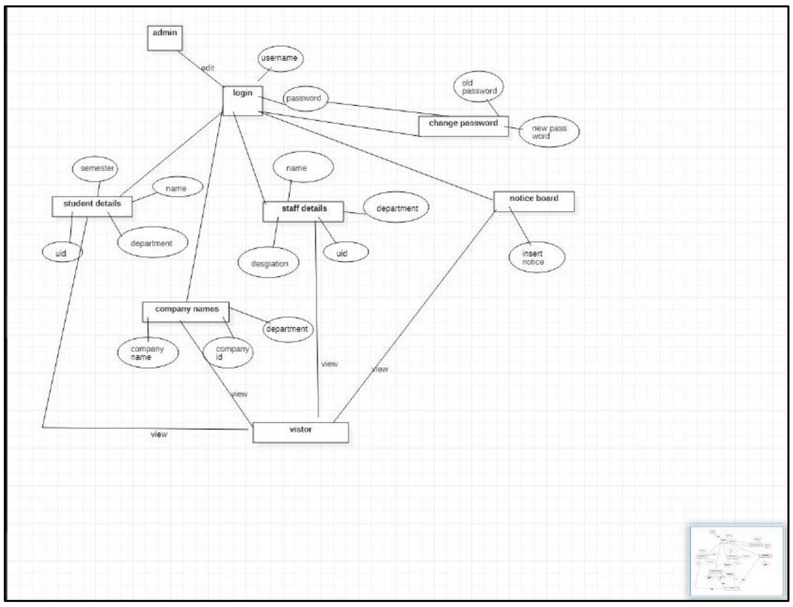


Fig 5.3 User Workflows

CHAPTER 6

SYSTEM ADMINISTRATION

6.1 Administrator Guide

An administrator guide for a College Management System desktop application provides instructions and guidelines for administrators to effectively manage and configure the application.

6.1.1 User Management

User management in a College Management System desktop application involves creating and managing user accounts, defining user roles and permissions, and ensuring secure access to the application. User management is a critical aspect of a College Management System desktop application to ensure secure and controlled access to the system's functionalities and data. It is important to implement robust security measures, regularly review and update user permissions, and provide proper training and support to users for efficient and effective use of the application.

6.1.2 System Configuration

System configuration for a College Management System desktop application involves setting up the necessary parameters and options to ensure the application operates optimally. It is important to document the system configuration settings and any changes made for future reference and troubleshooting purposes. Regularly review and update the system configuration as needed to accommodate changing requirements or technological advancements.

6.2 Data Management

Data management is a crucial aspect of a College Management System desktop application, as it involves handling and organizing the application's data effectively. Here are the key considerations for data management in a college management system:

6.2.1 Data Backup and Restore Procedures

Implement regular and automated data backup procedures to protect against data loss or corruption.Store backup copies in secure locations and perform periodic restoration tests to ensure the integrity of the backups.Establish a disaster recovery plan to restore data and resume operations in case of system failures or catastrophic events.

6.2.2 Data Maintanence and Archiving

Regularly maintain and update the data to reflect any changes in student enrollments, course offerings, faculty assignments, and other relevant information.Implement archiving mechanisms to preserve historical data for compliance, reporting, or auditing purposes.Define data retention policies to ensure compliance with data protection regulations and institutional policies.

6.3 Security and Access Control

Security and access control are critical aspects of a College Management System desktop application to protect sensitive data, prevent unauthorized access, and ensure data privacy. Here are key considerations for implementing security and access control measures:

6.3.1 User Authentication

Implement a robust authentication mechanism to verify the identity of users accessing the application.Use secure password storage techniques, such as hashing and salting, to protect user passwords.Consider implementing additional authentication methods, such as two-factor authentication, for enhanced security.

6.3.2 Data Encryption

Encrypt sensitive data, such as passwords, personal information, or financial details, when stored in the database or transmitted over the network.Use strong encryption algorithms and ensure proper key management practices.Implement secure communication protocols, such as HTTPS, to protect data during transmission.

6.3.3 Access Control

Restrict access to sensitive functionalities and data based on user roles and permissions.Implement access control mechanisms, such as user-based or role-based access control lists (ACLs), to control access at a granular level.Enforce access control policies consistently throughout the application to prevent unauthorized actions.

CHAPTER 7

SOURCE CODE

7.1 Main Function

The provided code appears to be a snippet of a College Management System desktop application implemented in Python using the tkinter library for the graphical user interface (GUI). It includes various functions and event handlers for different functionalities within the application.

The code includes the following key components:

- The tkinter library is imported to create the GUI elements.
- The PIL (Python Imaging Library) is imported to work with images.
- The mysql.connector library is imported to connect and interact with a MySQL database.
- The root window is created with specified dimensions and properties.
- The on_closing function handles the event when the application window is closed.
- The draw_login_page function creates a new window for the admin login page.
- The login function verifies the admin credentials and grants or denies access based on the input.
- The button_mode function toggles between admin and visitor modes.
- The draw_visitor and draw_admin functions create the respective dashboards based on the user mode.
- Various button functions, such as draw_search_employee, draw_search_student, etc., are defined to handle different actions in the application.
- The GUI elements are organized using frames and placed at specific coordinates within the root window.

Code: main.py

```
import tkinter
from tkinter import *
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
import os
import mysql.connector
root =Tk()
root.title("Home Page")
root.geometry("1080x650+100+20")
root.resizable(False, False)
global root_window
is_admin = False
root_window = root
def on_closing():
    global is_on
    if is_admin:
        on_.config(image=off)
        is_on = False
    else:
        on_.config(image=on)
        is_on = True
    window.destroy()
def draw_login_page():
    global window
    newWindow = Toplevel(root_window)
    # newWindow.attributes('-alpha',0.9)
    newWindow.title("Admin Login")
    newWindow.geometry("400x400+500+200")
    newWindow.configure(bg="#f0e800")
    header=Frame(newWindow, bg="#397ae2", bd=5)
    header.place(x=0,y=0,width=400,height=60)
    #heading label
    nsec=Label(header, text="CIET",font=("Helvetica",16,"bold"), bg =
"#397ae2",fg="#1d1d54")
    nsec.place(x=0, y=10, width=400)
    username = Label(newWindow, text ="Username", font=("Helvetica", 16), relief = FLAT,
bg="#f0e800")
    username.place(x=20, y=100)
    username_input = Text(newWindow, height = 1,
                font = ("Helvetica", 16),
                width = 20,
                bg = "light yellow")
    username_input.place(x=120, y=100)
    password = Label(newWindow, text ="Password", font=("Helvetica", 16), relief = FLAT,
bg="#f0e800")
    password.place(x=20, y=175)
    password_input = tkinter.Entry(newWindow,
                        show = '*',
                        font = ("Helvetica", 16),
                        width = 20,
                        bg = "light yellow"
                        )
    password_input.place(x=120, y=175)
    submit= Button(newWindow, text="LOGIN", command = login, font=("Helvetica", 16),
bd =2, bg = "#397ae2",fg="#1d1d54", relief=RAISED)
    submit.place(x=150, y=250, width = 100, height = 50)
    alert=Label(newWindow, text="Default Username: admin,
password:admin",font=("Helvetica",8,), bg = "#397ae2",fg="#1d1d54")
    alert.place(x=0, y=360, width=400)
    newWindow.protocol("WM_DELETE_WINDOW", on_closing)
    window = newWindow
```

```
def login():
    username = window.winfo_children()[2].get(1.0, "end-1c")
    password = window.winfo_children()[4].get()
    db = mysql.connector.connect(host="localhost", user="root", password="",
database="nsec_db")
    mycursor = db.cursor()
    mycursor.execute("SELECT y FROM global_values WHERE x = 'admin'")
    rows = mycursor.fetchone()
    if rows == None:
        messagebox.showinfo("Failure",  "Oops! Something went wrong")
        return
    server_pwd = rows[0]
    db.commit()
    db.close()
    global is_on
    if username == "admin" and password == server_pwd:
        tkinter.messagebox.showinfo("Success",  "Access Granted")
        is_admin = True
        on_.config(image=off)
        is_on = False
        draw_admin()
    else:
        tkinter.messagebox.showinfo("Failure",  "Access Denied")
        is_admin = False
        on_.config(image=on)
        is_on = True
        draw_visitor()
    window.destroy()
def button_mode():
   global is_on
   if is_on:
        on_.config(image=off)
        is_on = False
        draw_login_page()
    else:
        on_.config(image = on)
        is_on = True
        is_admin = False
        draw_visitor()
def draw_search_employee():
    # import search_employee
    os.system('python search_employee.py')
def draw_search_student():
    # import search_student
    os.system('python search_student.py')
def draw_company_list():
    os.system('python view_company_list.py')
def draw_visitor_fees():
    os.system('python visitor_fees_structure.py')
def draw_notice_board():
    os.system('python view_notice_board.py')
def draw_edit_student():
    os.system('python edit_student.py')
def draw_edit_employee():
    os.system('python edit_employee.py')
def draw_edit_notice_board():
    os.system('python edit_notice_board.py')
def draw_edit_company_list():
    os.system('python edit_company_list.py')
def draw_change_password():
    os.system('python change_password.py')
```

```
def draw_execute_dbms():
    os.system('python execute_dbms.py')
def draw_visitor():
    for widget in dashboard.winfo_children():
        widget.destroy()
    welcome_text["text"] = "Welcome, Visitor"
    image1 = Image.open("media/nsec.jpg")
    test = ImageTk.PhotoImage(image1)
    label1 = Label(dashboard,image=test)
    label1.photo = test
    label1.place(x=0, y=0, height = 400, width = 1080)
    option= Button(dashboard, text ="Search staff", command = draw_search_employee, bd
=0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=150, y=75, width = 200, height = 50)
    option= Button(dashboard, text ="Search Student", command =  draw_search_student, bd
=0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=730, y=75, width = 200, height = 50)
    option= Button(dashboard, text ="Fee Structure", command = draw_visitor_fees, bd =0,
font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=150, y=275, width = 200, height = 50)
    option= Button(dashboard, text ="Notice Board", command = draw_notice_board,bd =0,
font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=442, y=175, width = 200, height = 50)
    option= Button(dashboard, text ="Company List", command = draw_company_list,bd =0,
font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=735, y=275, width = 200, height = 50)
def draw_admin():
    for widget in dashboard.winfo_children():
        widget.destroy()
    welcome_text["text"] = "Welcome, Admin"
    image1 = Image.open("media/nsec.jpg")
    test = ImageTk.PhotoImage(image1)
    label1 = Label(dashboard,image=test)
    label1.photo = test
    label1.place(x=0, y=0, height = 400, width = 1080)
    option= Button(dashboard, text ="View/Edit Employee", command =
draw_edit_employee, bd =0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=100, y=75, width = 250, height = 50)
    option= Button(dashboard, text ="View/Edit Student", command =  draw_edit_student, bd
=0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=735, y=75, width = 250, height = 50)
    option= Button(dashboard, text ="Change Password", command =
draw_change_password, bd =0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=100, y=275, width = 250, height = 50)
    option= Button(dashboard, text ="View/Edit Notice Board", command =
draw_edit_notice_board, bd =0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=420, y=75, width = 250, height = 50)
    option= Button(dashboard, text ="Execute DBMS Query", command =
draw_execute_dbms, bd =0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=420, y=275, width = 250, height = 50)
    option= Button(dashboard, text ="View/Edit Company List", command =
draw_edit_company_list,bd =0, font=("Helvetica",16), bg = "#4dc7f0",fg="#1d1d54")
    option.place(x=735, y=275, width = 250, height = 50)
#Header
header=Frame(root, bg="brown", bd=0)
header.place(x=0,y=0,width=1080,height=720)
#logo
image1 = Image.open("media/logo.jpg")
test = ImageTk.PhotoImage(image1)
label1 = tkinter.Label(header,image=test)
label1.image = test
label1.place(x=0, y=0, height =120)
#heading label
nsec=Label(header, text="CIET",font=("Helvetica",36,"bold"), bg = "brown",fg="#1d1d54")
```

```
nsec.place(x=205, y=20, width=950)
#Profile frame
frame=Frame(root, bg="#f0e800")
frame.place(x=0,y=115,width=1080,height=50)
welcome_text = Label(frame, text = "Welcome, Visitor", font=("Minion Pro Regular", 16),
bg="#f0e800")
welcome_text.place(x=20, y=10)
is_on = True
on = PhotoImage(file ="media/on.png")
off = PhotoImage(file ="media/off.png")
# Create A Button
on_= Button(frame, image =on,bd =0, bg = "#f0e800", command = button_mode)
on_.place(x=950, y=0, width = 50, height = 50)
#visitor_text
visitor_text = Label(frame, text = "Visitor", font=("Minion Pro Regular", 16), bg="#f0e800")
visitor_text.place(x=880, y=10)
admin_text = Label(frame, text = "Admin", font=("Minion Pro Regular", 16), bg="#f0e800")
admin_text.place(x=1000, y=10)
#profile picture
image1 = Image.open("media/profile.png")
test = ImageTk.PhotoImage(image1)
label1 = tkinter.Label(frame,image=test, bg = "#f0e800")
label1.place(x=820, y=0)
dashboard=Frame(root, bg="#bbb", bd=0)
dashboard.place(x=0,y=165,width=1080,height=400)
draw_visitor()
#Footer
footer=Frame(root, bg="brown", bd=0)
footer.place(x=0,y=565,width=1080,height=85)
root.mainloop()
```

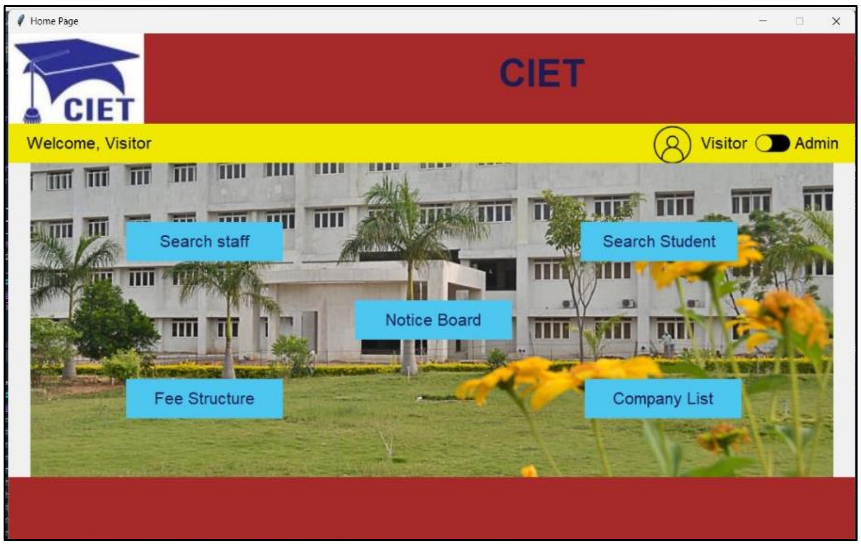


Fig 7.1 Main Function Output

7.2 Edit Function

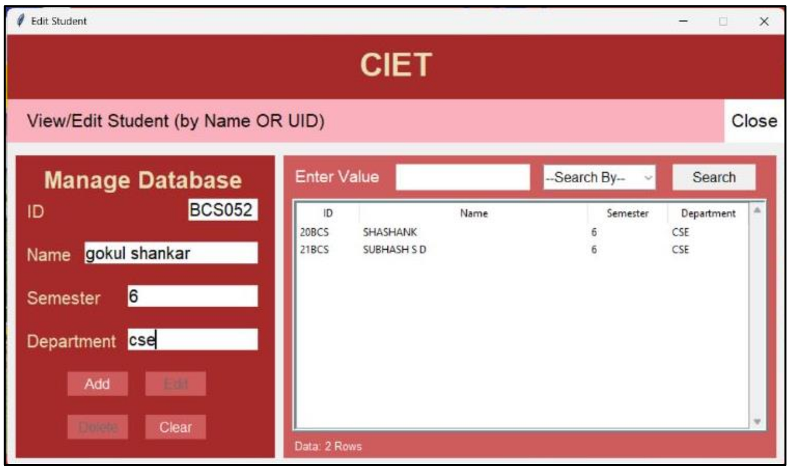
The Edit function will update the selected record in the database with the values entered in the entry fields.

7.2.1 Edit_Student Function

The code is provided to Tkinter application for editing student data stored in a MySQL database.

Code : Student.py

```
from tkinter import*
import mysql.connector
def search():
    option = dropdown.get()
    search_input = searchInput.get()
    db = mysql.connector.connect(host="localhost", user="root", password="", database="nsec_db")
    mycursor = db.cursor()
    if search_input == "":
        mycursor.execute("SELECT * FROM student_table")
    elif option == "Name":
        mycursor.execute("SELECT * FROM student_table WHERE name = '" + str(search_input).upper() + "'")
    elif option == "UID":
        mycursor.execute("SELECT * FROM student_table WHERE id = '" + str(search_input) + "'")
    elif option == "Semester":
        else:
        return
    rows = mycursor.fetchall()
    data_table.delete(*data_table.get_children())
    if rows == None:
        note_text['text'] = "Data: 0 Rows"
        return
    note_text['text'] = "Data: "+str(len(rows))+" Rows"
    for row in rows:
        data_table.insert('', END, values=row)
    db.commit()
    db.close()
    # pass
```



ID	BCS052
Name	gokul shankar
Semester	6
Department	cse

ID	Name	Semester	Department
20BCS	SHASHANK	6	CSE
21BCS	SUBHASH S D	6	CSE

Data: 2 Rows

Fig 7.2.1 Edit Student Detail

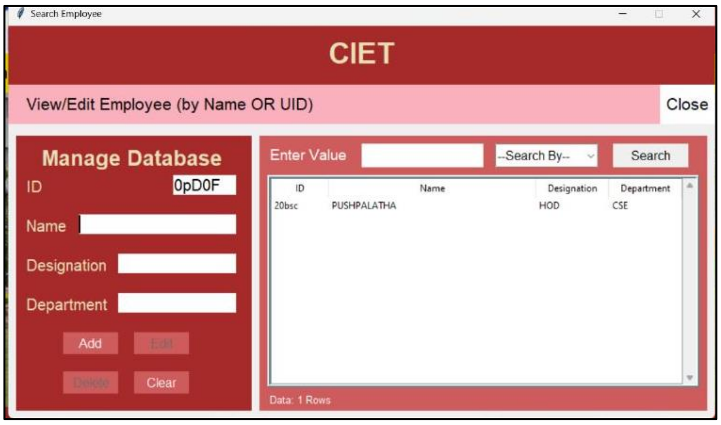
7.2.2 Edit_Employee Function

The code is provided to Tkinter application for editing student data stored in a MySQL database.

Code : Employee.py

```
from tkinter import*
import mysql.connector
root.title("Search Employee")
root.geometry("900x500+200+100")
root.resizable(False, False)
global root_window
root_window = root
def search():
   option = dropdown.get()
   search_input = searchInput.get()
   db = mysql.connector.connect(host="localhost", user="root", password="",
database="nsec_db")
   mycursor = db.cursor()
   if search_input == "":
      mycursor.execute("SELECT * FROM employee_table")
   elif option == "Name":
      mycursor.execute("SELECT * FROM employee_table WHERE name = '" +
str(search_input).upper() + "'")
   elif option == "UID":
      mycursor.execute("SELECT * FROM employee_table WHERE id = '" + str(search_input)
+ "'")
   elif option == "Designation":
   else:
      return
   rows = mycursor.fetchall()
   data_table.delete(*data_table.get_children())
```

```
    for row in rows:
        data_table.insert('', END, values=row)
    db.commit()
    db.close()
    # pass
```



ID: 0pD0F
Name:
Designation:
Department:

ID	Name	Designation	Department
20bsc	PUSHPALATHA	HOD	CSE

Data: 1 Rows

Fig 7.2.2 Edit Employee Detail

7.3 View Function

The view() function creates the main window, sets its title, geometry, and disables resizing

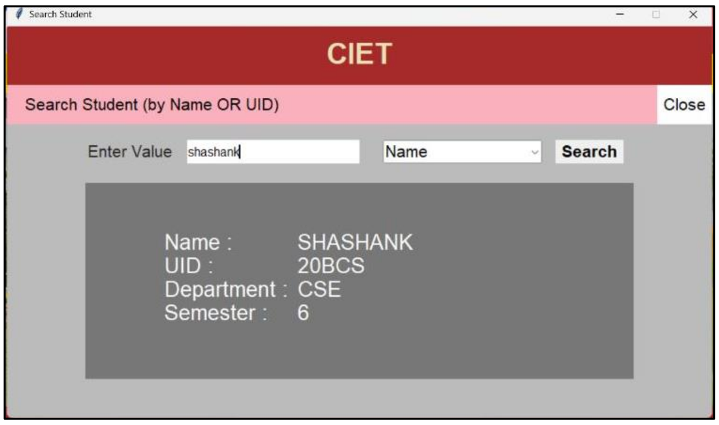
7.3.1 View_Student Function

The Tkinter application that allows searching for student records in a MySQL database. It includes various functions for handling the search, drawing the search result, and handling the GUI components.

Code : View_Stu.py

```
import tkinter
from tkinter import *
from tkinter import ttk, messagebox
def on_closing():
    root_window.destroy()
```

```
def search():
    option = dropdown.get()
    search_input = searchInput.get()
    db = mysql.connector.connect(host="localhost", user="root", password="",
database="nsec_db")
    mycursor = db.cursor(buffered = True)
    if option == "Name":
        mycursor.execute("SELECT * FROM student_table WHERE name = '" +
str(search_input) + "'")
    elif option == "UID":
        mycursor.execute("SELECT * FROM student_table WHERE id = '" + str(search_input)
+ "'")
    else:
        return
    rows = mycursor.fetchone()
    if rows == None:
        draw_no_result()
    global result, result_uid, result_name, result_dept, result_sem
    result_uid["text"] = rows[0]
    db.commit()
    db.close()
    # pass
```



Enter Value shashank Name

Name : SHASHANK
UID : 20BCS
Department : CSE
Semester : 6

Fig 7.3.1 View Student Detail

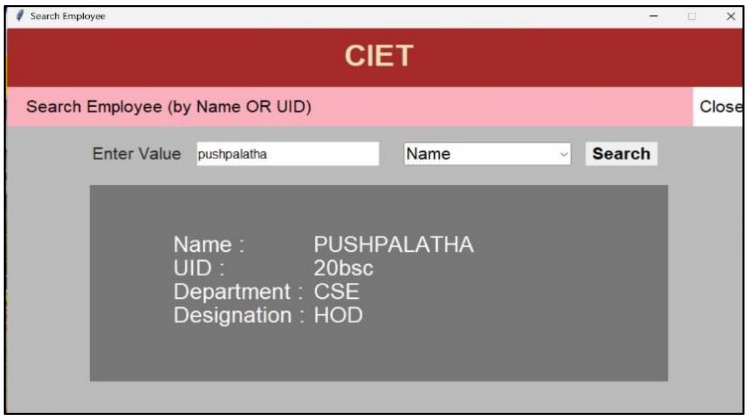
7.3.2 View_Employee Function

The Tkinter application that allows searching for student records in a MySQL database. It includes various functions for handling the search, drawing the search result, and handling the GUI components.

Code : View_Emp.py

```
import tkinter
from tkinter import *
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
import mysql.connector
root =Tk()
root.title("Search Employee")
root.geometry("900x500+200+100")
root.resizable(False, False)
global root_window
root_window = root
def on_closing():
   root_window.destroy()
def on_back():
   root_window.destroy()
   modulename = 'main'
   if modulename not in sys.modules:
      import main
   else:
      importlib.reload(main)
def search():
   option = dropdown.get()
   search_input = searchInput.get()
```

```
db = mysql.connector.connect(host="localhost", user="root", password="", database="nsec_db")

mycursor = db.cursor(buffered = True)

if option == "Name":

    mycursor.execute("SELECT * FROM employee_table WHERE name = '" + str(search_input) + "'")

elif option == "UID":

    mycursor.execute("SELECT * FROM employee_table WHERE id = '" + str(search_input) + "'")

else:

    return

rows = mycursor.fetchone()

if rows == None:

    draw_no_result()

global result, result_uid, result_name, result_dept, result_desg

result_uid["text"] = rows[0]

result_name["text"] = rows[1]

result_dept["text"] =  rows[3]

result_desg["text"] = rows[2]

db.commit()

db.close()

# pass
```



pushpalatha
Name

Name : PUSHPALATHA
UID : 20bsc
Department : CSE
Designation : HOD

Fig 7.3.2 View Employee Detail

8. GLOSSARY

Tkinter:

A standard Python library used for creating GUI applications. It provides a set of tools and widgets to build the graphical user interface.

Image and ImageTk:

Classes from the PIL (Python Imaging Library) module used for working with images in tkinter. They allow loading, manipulating, and displaying images within the GUI.

Python-PIL :

Python PIL (Python Imaging Library) is a popular library for image processing and manipulation in Python.It upports various image file formats, including JPEG, PNG, GIF, BMP, and TIFF.

Notice Board:

A function or method that is triggered in response to a specific event, such as a button click or window closure.

Admin:

Short for "Administrator," refers to a user role in the college management system with elevated privileges and permissions. The admin role typically involves managing and overseeing various aspects of the system, such as user management, database administration, and configuration.

Visitor:

Refers to a user role in the college management system with limited access and permissions. Visitors may have restricted functionality and are usually allowed to view certain information or perform specific tasks within the application.

9. LITERATURE SURVEY

1) **Srikant Patnaik, Khushboo kumari Singh, Rashmi Ranjan, Niki Kumari** College management system assists in modifying the existing system to site based system.This is a paperless work. It can be monitored and controlled remotely.
2) **Kartiki Datarkar, Neha Hajare, Nidhi Fulzele, Sonali Kawle, Vaibhav Suryavanshi, Dipeeka Radke** Online College Management System assists in automating the existing manual system. This is a paperless work. It can be monitored and controlled remotely. It reduces the man power required. It provides accurate information always. Malpractice can be reduced. All years together gathered information can be saved and can be accessed at any time.
3) **Ms. A. V. Sinhasane, Ms. A.N. Kashid, Ms. P. J. Kumbhar, Ms. P.R. Shirpale, Prof. S. L. Mortale** College Department Management System. The system provides guidance to the admin to keep track of each student. The admin have the access to the database of system .In an educational institute management is crucial thing. So in order to reduce the efforts of staff we are introducing our system.

10.REFERENCES

1) Srikant Patnaik, Khushboo kumari Singh, Rashmi Ranjan and Niki Kumari (2016) "College Management System, International Research Journal of Engineering and Technology (IRJET) Volume: 03 Issue: 05, May2016.

2) Kartiki Datarkar, Neha Hajare, Nidhi Fulzele, Sonali Kawle, Vaibhav Suryavanshi and Dipeeka Radke, "Online College Management System, International Journal of Computer Science and Mobile Computing Vol.5 Issue.4, April- 2016, pg. 118-122.

3) Java and Software Design Concepts by Apress

4) http://www.tutorialspoint.com

5) http://javapoint.com

6) http://www.abhiandroid.com

11. CONCLUSION

The project as College Management System is the system that deals with the issue related manual college management system. This project is successfully implemented with all the features required for college.

The application provides appropriate information to user according to the chosen activity. The project is designed keeping in view the day to day problem faced by a manual college management system. Deployment of our College Management System help the college to reduce unnecessary wastage of time in doing work using manual college management system.

The system is user-friendly, highly interactive and flexible for further enhancement. The system generates the reports as when required. The coding is done in a simplified and understandable manner.