VISVESVARAYA TECHNOLOGICAL UNIVERSITY

UNIVERSITY MANAGEMENT SYSTEM

FILE STRUCTURES MINI PROJECT REPORT

Submitted by

ANJU T M	1RF20IS006
ANKITA KUMARI	1RF20IS007

Under the Guidance of
Dr. Vinoth Kumar M,
Associate Professor,
ISE Department, RV Institute of Technology and Management

in partial fulfillment for the award of degree
of
Bachelor of Engineering

IN

Information Science and Engineering

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT

BANGALORE-560076

2022-23

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BANGALORE - 560076
(Affiliated to VTU, Belagavi, Approved by AICTE New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the mini project work titled **'UNIVERSITY MANAGEMENT SYSTEM'** is carried out by **ANJU T M (1RF20IS006), ANKITA KUMARI (1RF20IS007),** who are bonafide students of RV Institute of Technology and Management, Bangalore, in partial fulfillment for the award of the degree of **Bachelor of Engineering** in **INFORMATION SCIENCE AND ENGINEERING** of the Visvesvaraya Technological University, Belgaum during the year **2022- 2023**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report deposited in the departmental library. The mini-project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution for the said degree.

Signature of Guide: **Signature of Head of the Department:**

External Viva

Name of Examiners	**Signature with date**
1	
2	

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BANGALORE - 560076
(Affiliated to VTU, Belagavi, Approved by AICTE New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DECLARATION

We, **ANJU T M (1RF20IS006), ANKITA KUMARI (1RF20IS007)**, the students of sixth semester B.E., **INFORMATION SCIENCE AND ENGINEERING,** hereby declare that the mini project titled **"UNIVERSITY MANAGEMENT SYSTEM"** has been carried out by us and submitted in partial fulfillment for the award of the degree of Bachelor of Engineering in **INFORMATION SCIENCE AND ENGINEERING.** We do declare that this work is not carried out by any other students for the award of a degree in any other branch.

Place: Bangalore **Signature**

Date: **1. ANJU T M**

2. ANKITA KUMARI

ACKNOWLEDGEMENT

The satisfaction and the euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. The constant guidance of these persons and the encouragement provided crowned our efforts with success and glory. Although it is not possible to thank all the members who helped with the completion of the mini-project individually, we take this opportunity to express our gratitude to one and all.

We wish to place it on record, our grateful thanks to **Dr. Vinoth Kumar M, Associate Professor**, Department of ISE, RV Institute of Technology and Management, our guide, for the constant inspiration, encouragement, timely guidance, and valid suggestions given to us.

We wish to thank **Dr. Latha C A, Head of Department**, Department of ISE, R V Institute of Technology and Management, for the support and encouragement.

We express our sincere gratitude to **Dr. Jayapal R**, Principal, R V Institute of Technology and Management for the support and encouragement.

We would like to thank the VTU, Belagavi, for having this mini project work as part of its curriculum, which gave us a wonderful opportunity to work on our research and presentation abilities. We are thankful to the Project Coordinator and all the staff members of the department for providing relevant information and helping in different capacities in carrying out the project.

ABSTRACT

The University Management System (UMS) is a comprehensive software platform designed to transform and simplify various user operations. It allows users to perform various operations such as adding records of colleges, deleting records, modifying records, searching for colleges based on different criteria, and sorting the records. The code utilizes file handling to store the college records in separate files.

Users can search for colleges based on rank, name, city, or eligibility rank. The search methods iterate through the record file and display the matching records. The sorting methods use bubble Sort to sort the records based on rank, name, or eligibility rank in ascending or descending order. Overall, the code provides basic functionality for managing college records, allowing users to add, delete, modify, search, and sort the records effectively.

The University Management System undertaken as a project is based on relevant technologies.This project has been developed to carry out the processes easily and quickly, which is not possible with the manual systems, which are overcome by this software. This project is developed using C++ language and executed in codeblocks and files as backend.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The University Management System presented in this code is a software solution designed to assist in the organization and management of college records. It provides a user-friendly interface to perform various operations such as adding, deleting, modifying, searching, and sorting college records. The system utilizes file handling to store and retrieve the records, ensuring data persistence.

Effective management of college records is crucial for educational institutions to maintain accurate and up-to-date information about colleges, their rankings, branches, cities, and eligibility ranks. This system aims to streamline the record management process and improve efficiency in accessing and manipulating the stored data.

Moreover, the system offers search functionality, enabling users to retrieve specific college records based on rank, name, city, or eligibility rank. Additionally, sorting methods are implemented to arrange the records in ascending or descending order according to rank, name, or eligibility rank.

By employing this university management system, educational institutions can efficiently manage their college records, ensuring accuracy, accessibility, and ease of manipulation. It provides a valuable tool for administrators, staff, and other authorized personnel involved in maintaining and utilizing college-related information.

1.1 Problem Statement

Program for managing college records. It allows users to add, modify, delete, and search for college records based on various criteria such as college name, rank, city, and eligibility rank. The program uses text files to store the records and provides functionalities for sorting the records based on different attributes. However, the code lacks proper documentation and structure, making it difficult to understand and maintain. It needs to be refactored and optimized to improve code readability, organization, and efficiency.

1.2 Scope of the project

The scope of the project is to develop a comprehensive university management system. The system should provide functionalities for adding, modifying, deleting, and searching college records based on different criteria. It should allow users to sort the records based on attributes such as college name, rank, city, and eligibility rank. The system should utilize text files for data storage and retrieval. Additionally, it should ensure proper code documentation, structure, and optimization for improved maintainability and efficiency.

1.3 Methodology

The methodology for developing the above code can follow the Agile software development approach, specifically using the Scrum framework. The project can be divided into several sprints, each consisting of a defined set of tasks to be completed within a specific time frame. The development process can be structured as follows:

Requirement Gathering: Identify and document the specific requirements of the college record system through discussions with stakeholders and users.Sprint Planning: Break down the requirements into smaller, manageable tasks and create a prioritized backlog. Select a set of tasks to be completed during each sprint.

Development: Implement the functionalities of the college record management system according to the defined tasks. Utilize modular and object-oriented programming principles to ensure code reusability and maintainability.

Testing: Conduct unit testing to ensure that each module functions as intended. Perform integration testing to verify the interactions between different components of the system. Use test cases to validate the system against various scenarios.

Review and Feedback: Regularly review the progress of the project with the team and stakeholders. Gather feedback and make necessary adjustments to meet the requirements and expectations.

Documentation: Maintain comprehensive documentation throughout the development process, including system architecture, code comments, and user manuals.

Deployment: Prepare the system for deployment by ensuring proper configuration and compatibility with the target environment. Conduct user acceptance testing and address any issues that arise.

Maintenance and Updates: Provide ongoing maintenance and support for the deployed system, addressing bug fixes and incorporating user feedback for future updates.Throughout the development process, collaboration, communication, and flexibility should be emphasized to adapt to changing requirements and ensure the successful delivery of the college record management system.

Chapter 2

ANALYSIS

2.1 System Requirement Specification

System Requirement Specification is a structured collection of information that embodies the requirements of the system. It describes all data, functional and behavioral requirements of the software under production or development.

2.1.1 Functional Requirements

The University Management System should fulfill the following functional requirements:

User Authentication: The system should provide a secure login mechanism to authenticate users, allowing them to access the system based on their roles (e.g., students, faculty, administrators).

Student Management: The system should allow administrators and faculty members to manage student records, including adding new students, updating student information (e.g., name, contact details, enrollment status), and maintaining academic records (e.g., grades, courses enrolled).

Course Management: The system should enable administrators and faculty members to manage courses offered by the college. This includes adding new courses, updating course information (e.g., title, description, prerequisites), and assigning instructors to courses.

Enrollment Management: The system should facilitate the enrollment process, allowing students to select and enroll in courses. It should handle prerequisites, seat availability, and provide confirmation of successful enrollment.

Grade Management: The system should allow instructors to enter and update grades for students based on their performance in assignments, exams, and other assessments. Students should be able to view their grades through the system.

Transcript Generation: The system should be capable of generating official transcripts for students, including their academic records, grades, and any other relevant information. Transcripts should be printable or available for download.

Reporting and Analytics: The system should provide reporting and analytics features to generate various reports, such as enrollment statistics, student performance analysis, and course evaluations. This can assist administrators and faculty in making data-driven decisions.

Communication and Notifications: The system should support communication between students, faculty, and administrators. It should include features like messaging, notifications, and announcements to facilitate important information sharing.

Data Security and Privacy: The system should ensure the security and privacy of sensitive student data. It should employ appropriate encryption, access controls, and data backup mechanisms to protect against unauthorized access or data loss.

System Administration: The system should provide administrative functionalities, allowing authorized personnel to manage system settings, user roles and permissions, and perform system backups and maintenance.These functional requirements provide a foundation for the development of the college record management system, but additional requirements may arise based on the specific needs and context of the institution.

2.1.2 Non-Functional Requirements

Non-functional requirements define the qualities and constraints of a system that are not directly related to its specific functionality. Here are some non-functional requirements that can be considered for the college record management system:

Usability: The system should have an intuitive and user-friendly interface, making it easy for users to navigate and perform tasks. It should provide clear instructions and helpful feedback to assist users in understanding and using the system effectively.

Performance: The system should be capable of handling a significant number of concurrent users and large volumes of data without significant performance degradation. Response times for user interactions, such as loading pages or executing queries, should be minimal to ensure a smooth user experience.

Reliability: The system should be highly reliable, minimizing downtime and ensuring data integrity. It should be able to recover from failures or system crashes without data loss.

Security: The system should have strong security measures in place to protect sensitive student data from unauthorized access, modification, or disclosure. It should comply with relevant data protection regulations and standards. User authentication and access controls should be implemented to ensure that users can only access the information they are authorized to view or modify.

Scalability: The system should be scalable to accommodate future growth in terms of user base, data volume, and system usage. It should be able to handle increasing demands without significant architectural changes or performance degradation.

Maintainability: The system should be designed and implemented in a modular and maintainable manner, making it easy to understand, modify, and enhance. Code should be well-documented, and appropriate version control and change management practices should be followed.

Interoperability: The system should support integration with other existing systems or external services, such as authentication systems, payment gateways, or learning management systems, as required by the institution. It should adhere to relevant standards and protocols to ensure smooth interoperability.

Accessibility: The system should be accessible to users with disabilities, complying with accessibility standards and guidelines. It should support features like keyboard navigation. Compatibility: The system should be compatible with a range of devices, browsers, and operating systems commonly used by the target users. It should be responsive and adaptable to different screen sizes and resolutions.

Performance Efficiency: The system should utilize system resources efficiently, minimizing unnecessary processing or resource consumption. It should optimize database queries, cache frequently accessed data, and employ techniques like load balancing to ensure optimal performance.

These non-functional requirements are essential for ensuring the overall quality, usability, and reliability of the college record management system. Additional non-functional requirements may arise based on specific organizational or regulatory considerations.

2.1.3 Constraints

The University Management System relies on file-based storage for account data, using binary files to store and retrieve account information.The system operates in a single-user mode, allowing one user to use the system at a time.

2.1.4 Future Enhancements

Mobile Application: Develop a mobile application version of the system to provide convenient access to students, faculty, and staff from their smartphones or tablets. The mobile app can offer features such as notifications, event calendars, course registrations.

2.2 Theory and fundamentals of area related to problem statement

1. Record Structure: Each record in a binary file should have a well-defined structure. The structure typically consists of fields that represent different attributes of the data. The fields

should be defined with appropriate data types (e.g., integer, floating-point, string) and have a fixed length to facilitate efficient storage and retrieval.

2. File Header: Binary files may include a file header at the beginning of the file. The file header contains essential information about the file, such as the number of records, file format, version, and any additional metadata. The file header helps in validating the file's integrity and provides necessary information for file operations.

3. Data Serialization: When writing records to a binary file, data serialization is often employed to convert the structured data into a binary format that can be easily written to the file. Serialization involves encoding the data into a stream of bytes, following a specific protocol (such as using a specific byte order, padding, or data alignment rules) to ensure consistency during serialization and deserialization processes.

4. Random Access: Binary files enable random access, allowing direct access to specific records based on their position or key. To support random access, an indexing mechanism can be employed. One common approach is to maintain an index file that stores the key values along with their corresponding record positions in the binary file. This index facilitates quick searching and retrieval of records based on keys.

5. Endianness: Endianness refers to the byte order used to store multibyte data types (e.g., integers) in binary files. There are two common endianness formats: big-endian and little-endian. It is crucial to ensure that the bank management system handles endianness correctly, especially when working with binary files across different platforms or when data needs to be exchanged with other systems.

6. File Access Control: Access control measures should be implemented to restrict unauthorized access to binary files containing sensitive data. This includes authentication mechanisms.

7. Security and Access Control: so security and access control measures are vital. File structures should incorporate authentication mechanisms, encryption techniques, and access control lists to protect data from unauthorized access.

Chapter 3

SYSTEM DESIGN

3.1 Design Overview

The university management system is designed to have a user-friendly interface and a robust architecture to efficiently handle student records and administrative tasks. The system incorporates a database to store and manage student information, including personal details, academic records, and attendance data. The user interface allows administrative staff to easily navigate and perform various functions such as adding, updating, and retrieving student records. The system employs authentication and access control mechanisms to ensure data security and privacy. It also includes automated processes for generating reports, managing course registrations, and tracking student progress. Overall, the design aims to provide a comprehensive solution that simplifies record management processes, improves data accuracy, and enhances administrative efficiency.

3.2 System Architecture:

The system architecture of the college record management system can be divided into three main components: the user interface, the application logic, and the database.

User Interface: This component provides a graphical interface for users to interact with the system. It includes forms, screens, and menus that allow administrative staff to perform tasks such as adding or updating student records, generating reports, and managing course registrations. The user interface should be intuitive, user-friendly, and responsive to ensure a smooth user experience.

Application Logic: This component handles the core functionality of the system. It consists of modules and functions responsible for processing user inputs, performing validations, and executing business rules. The application logic ensures that data integrity is maintained, and the system operates correctly. It also includes security measures such as authentication and access control to protect sensitive information.

Database: The database component stores and manages all the student records and related data. It provides a structured storage mechanism for efficient retrieval, storage, and manipulation of information. The database should be designed to accommodate various data entities such as student profiles, course details, attendance records, and academic performance. It also supports queries and data manipulation operations to support the application logic.

3.3 Design Diagrams:

3.3.1 Use Case

A use case diagram typically consists of actors, use cases, and relationships between them. An actor is a person, organization, or external system that interacts with the system. A use case is a sequence of related activities performed by the system in response to a request from an actor. The relationships between the actors and use cases indicate which actors initiate the use cases. The use case diagram can be used to define the scope of a system and its boundaries. It can also be used to identify the major components of a system, the tasks that need to be performed, and the interactions between the components. The use case diagram provides an overview of the system and helps identify potential areas of improvement.

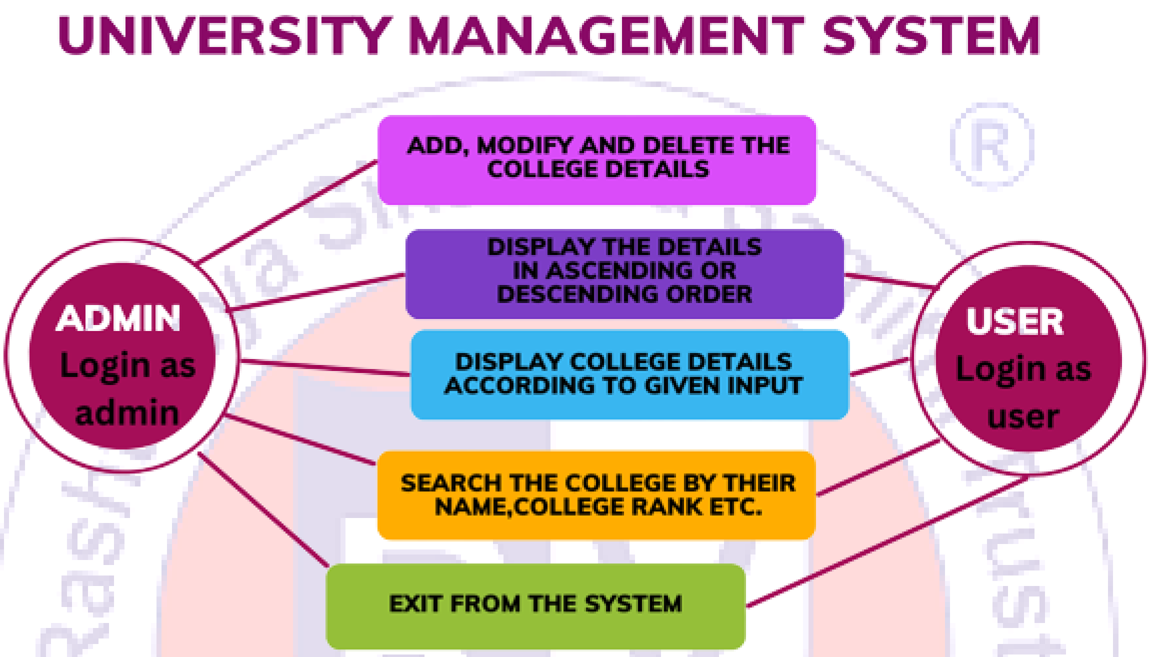

UNIVERSITY MANAGEMENT SYSTEM
ADMIN
Login as admin
ADD, MODIFY AND DELETE THE COLLEGE DETAILS
DISPLAY THE DETAILS IN ASCENDING OR DESCENDING ORDER
DISPLAY COLLEGE DETAILS ACCORDING TO GIVEN INPUT
SEARCH THE COLLEGE BY THEIR NAME,COLLEGE RANK ETC.
EXIT FROM THE SYSTEM
USER
Login as user

Fig 3.1. Use Case diagram

3.3.2 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) may be a quite interaction diagram that shows however processes operate with each other and in what order. it's a construct of a Message Sequence Chart. Sequence diagrams square measures typically known as event diagrams, event eventualities, and temporal order diagrams.

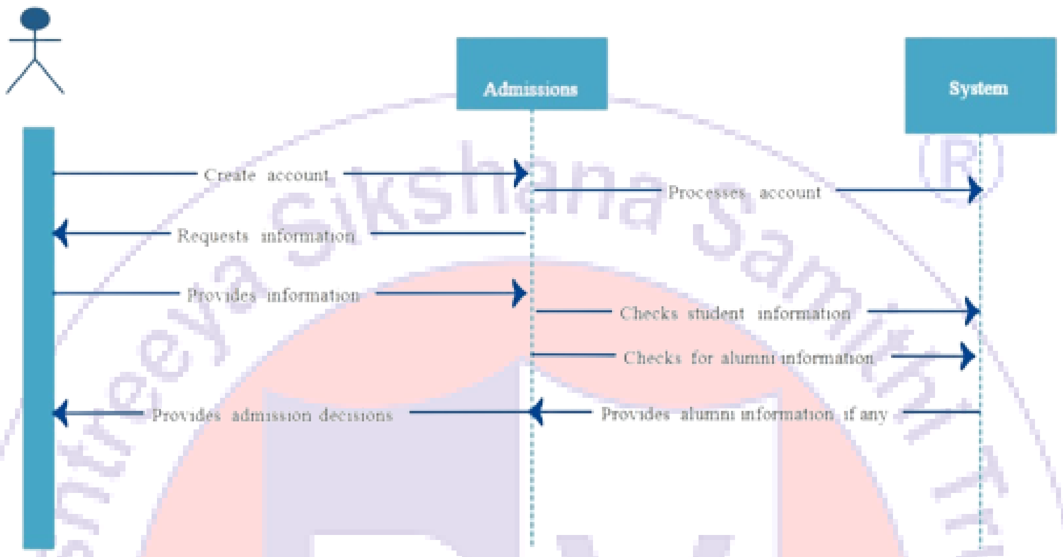

Admissions
System
Create account
Processes account
Requests information
Provides information
Checks student information
Checks for alumni information
Provides admission decisions
Provides alumni information if any

Fig 3.2. Sequence diagram

Chapter 4

IMPLEMENTATION

4.1 Steps for implementation:

1. Define the data structures:

Identify the entities and their attributes, such as students, courses, professors, departments, etc.

2. Set up the development environment:

Install the necessary software and tools, such as a programming language (e.g., Python).

3. Implement views:

Create views to handle different functionalities, such as displaying college information, adding a new college, the number of branches in that college, and eligibility rank etc.
Write the necessary logic in the views to interact with the models and perform operations on the data.

4. Implement forms:

Create forms to handle user input, such as adding a new college or updating their information.
Validate the form data and handle any errors or invalid input.

5. Implement authentication and authorization:

Set up user authentication to allow users to log in and access the system securely.

Implement role-based access control to restrict certain functionalities based on user roles (e.g., admin).

6. Test the system:

Test the implemented functionalities to ensure they work as expected.

Perform unit tests, integration tests, and user acceptance tests to identify and fix any issues or bugs.

7. Deploy the system:

Prepare the system for deployment by configuring the production environment.

9. Conduct thorough testing:

Test the program with various scenarios and inputs to ensure it handles all cases correctly. Check for any potential memory leaks or runtime errors. Validate the program against the expected behavior and requirements.

8. Provide ongoing maintenance and support:

Monitor the system for any errors or performance issues and address them promptly.

Regularly update and maintain the system to ensure it remains secure and up to date with any changes in requirements or technologies.

4.2 Code

```
      @staticmethod
      def search_college_rank():
      rank = input("Enter the College Rank : ")
       temp_rank = []
       with open("record.txt", 'r') as f:
          for i in f:

             line = i.split()

             temp_rank.append(line)

       for i in range(len(temp_rank)):

          if temp_rank[i][1] == rank:
             print("\nCollege found\n")
print(f"College Name : {temp_rank[i][2]}\t\tRank : {temp_rank[i][1]}\t\t
        Number Of Branches :
{temp_rank[i][3]}\t\tCity : {temp_rank[i][4]}\t\tEligible Rank : {temp_rank[i][5]}")
             returnprint(f"\nCollege with the rank {rank} does not exist")
@staticmethod

   def search_college_name():

       name = input("Enter the College Name : ")

       temp = []

       with open("record.txt", 'r') as f:

for i in f:

             line = i.split()

             temp.append(line)

       for i in range(len(temp)):

          if temp[i][2].lower() == name.lower():
            print("\nCollege found\n")

        print( f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches :
{temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")

  return

        print(f"\n{name} does not exist")

@staticmethod

    def search_college_city():

        city = input("Enter the City Name : ")

        temp = []

        count = True

        with open("record.txt", 'r') as f:

            for i in f:

                line = i.split()

                temp.append(line)

        for i in range(len(temp)):

            if temp[i][4].lower() == city.lower():
                count = False

                print("\nCollege found")

print(  f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")

if count:

            print(f"\nCollege does not exist in {city}")
    @staticmethod
    def search_college_eRank():
        eRank = input("Enter the Eligible Rank : ")
        temp = []
        count = True
        with open("record.txt", 'r') as f:
            for i in f:
                line = i.split()
                temp.append(line)
        for i in range(len(temp)):
            if temp[i][5] == eRank:
                count = False
                print("\nCollege found")
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
        if count:
            print(f"\nCollege with Eligible rank {eRank} does not exist")
    @staticmethod
    def sort_college_rank():
        order = input("Sort by Ascending or Descending\nEnter A or D : ").lower()
        temp = College.bubble_sort(1)
        if order == 'a':
            for i in range(len(temp)):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches :
{temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")


elif order == 'd':
            for i in range(len(temp) - 1, -1, -1):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches :
{temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
        else:
            for i in range(len(temp)):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches :
{temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")

  @staticmethod
    def sort_college_eRank():
        order = input("Sort by Ascending or Descending\nEnter A or D : ").lower()
        temp = College.bubble_sort(3)
        if order == 'a':
            for i in range(len(temp)):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches :
{temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
elif order == 'd':
            for i in range(len(temp) - 1, -1, -1):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
        else:
            for i in range(len(temp)):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
  @staticmethod
    def sort_college_name():
        order = input("Sort by Ascending or Descending\nEnter A or D : ").lower()
        temp = College.bubble_sort(2)

  if order == 'a':
            for i in range(len(temp)):
               print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
        elif order == 'd':
            for i in range(len(temp) - 1, -1, -1):
                print(f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")
        else:
            for i in range(len(temp)):
                print( f"College Name : {temp[i][2]}\t\tRank : {temp[i][1]}\t\tNumber Of Branches : {temp[i][3]}\t\tCity : {temp[i][4]}\t\tEligible Rank : {temp[i][5]}")

    @staticmethod
            col = 1
      with open("record.txt", 'r') as f:
                for i in f:
                    line = i.split()
            temp.append(line)
        if col == 2:
            for j in range(len(temp)):
                for i in range(0, len(temp) - j - 1):
                    if temp[i][col].lower() > temp[i + 1][col].lower():
                        temp[i], temp[i + 1] = temp[i + 1], temp[i]
    else:
            for j in range(len(temp)):
                for i in range(0, len(temp) - j - 1):
                    if int(temp[i][col]) > int(temp[i + 1][col]):
                        temp[i], temp[i + 1] = temp[i + 1], temp[i]
        return temp
def main():
    c = College()
    print(f'''
+----------------------------------------------+
|                                              |
|                                              |
|   Visvesvaraya Technological University      |
|                                              |
|                                              |
+----------------------------------------------+
    ''')
    while True:
        print(f'''\nChoose an option
        1. For Admin
        2. For User
        3. To Exit''')
        ans = int(input("Enter your choice : "))
        if ans == 1:
            password = input("Enter the password : ")
            if password == "admin":
                print("\nAccess Granted")
                print("Welcome Admin\n")

print(f'''Choose an option
                1. Add Data
                2. Modify Data
                3. Delete Data
                4. Display
                5. Exit''')
                ch = int(input("Enter your choice : "))
                if ch == 1:
                    c.add_record()
                    c.create_file()
                elif ch == 2:
                    c.modify()
                elif ch == 3:
                    c.delete_record()
                elif ch == 4:
                    c.display()
                elif ch == 5:
                    break
                else:
                print("\nInvalid Choice\n")
            else        print("\nInvalid Password\n")
    elif ans == 2:
            print(f"'Choose an option
            1. Display
            2. Search
            3. Exit"')
            ch = int(input("Enter your choice : "))
            if ch == 1:
              print(f"'Choose an option
              1. Display College's by Name
              2. Display College's by Rank

     3. Display College's by Eligible Rank
              4. Exit"')
    ch = int(input("Enter your choice : "))
              if ch == 1:
                c.sort_college_name()
              elif ch == 2:
                c.sort_college_rank()
              elif ch == 3:
                c.sort_college_eRank()
              elif ch == 4:
                break
            else:
              print("\nInvalid Choice\n")
        elif ch == 2:
            print(f'''Choose an option
            1. Search by College rank
            2. Search by College name
            3. Search by City
            4. Search by Eligible Rank
            5. Exit''')
    ch = int(input("Enter your choice : "))
            if ch == 1:
              c.search_college_rank()
            elif ch == 2:
              c.search_college_name()
            elif ch == 3:
              c.search_college_city()
            elif ch == 4:
              c.search_college_eRank()
            elif ch == 5:
               break
             else:
               print("\nInvalid Choice\n")
          elif ch == 3:
            break
          else:
            print("\nInvalid Choice\n")
        elif ans == 3:
          print("\nSuccessfully Logged Out\n")
          break
```

```
else:
  print("\nInvalid Choice\n")
 main()
```

CHAPTER 5

TESTING AND RESULTS

5.1 Component testing

In unit testing, the program modules that make up the application are tested individually. Unit testing focuses on locating errors in the working modules that are independent to each other. This enables it to detect errors in coding and the logic within the module alone. The various routines were checked and the corresponding output was tested. Table 5.1 gives details of validation. Test cases used in the project as follows:

TABLE 5.1: TEST CASES

TEST UNIT	TEST CASE	RESULT
Login as admin	Admin option is chosen And enter the password	System does the respected operations.
Add the data	Add college option is chosen	System does the respected operations.
Modify the data	Modification in data is added	System does the respected operations.
Delete the data	Deletion of college data	Delete the college from the file

Display the data	Colleges name, branch,city, etc is displayed.	Display college details from the file.
Display the student records in the file	Display option will be chosen.	Display the all students in the file
Login as a user	Display, search and exit Options are there	Choose one of the options
Display options is chosen	Three options will be there: Display College by Name, Display College by Rank, Display College's by Eligible Rank	Choose any \ one of the options and It will function according to the chosen Option. It will ask for ascending or descending order as we used the sorting algorithm.
Search option is chosen	Four options will be there: Search by College rank, Search by College name, Search by City,Search by Eligible Rank	Work according to the chosen option.
Exit option is chosen	Exit	It will exit from the file.

Fig 5.1:Test-cases

University Management System Test Cases

1. Test Case: Create Login as Admin

Description: Verify if a admin is entering the correct password or not.

Expected Result: The system should allow the admin to login and show the result as welcome admin.

2. Test Case: Add New College data

Description: Adding of new college data like college name,college city etc. are done by the admin only.

Expected Result: Addition of new colleges in the system.

3. Test Case:Modify and delete the college details

Description: The admin only can modify or delete the data from the system.

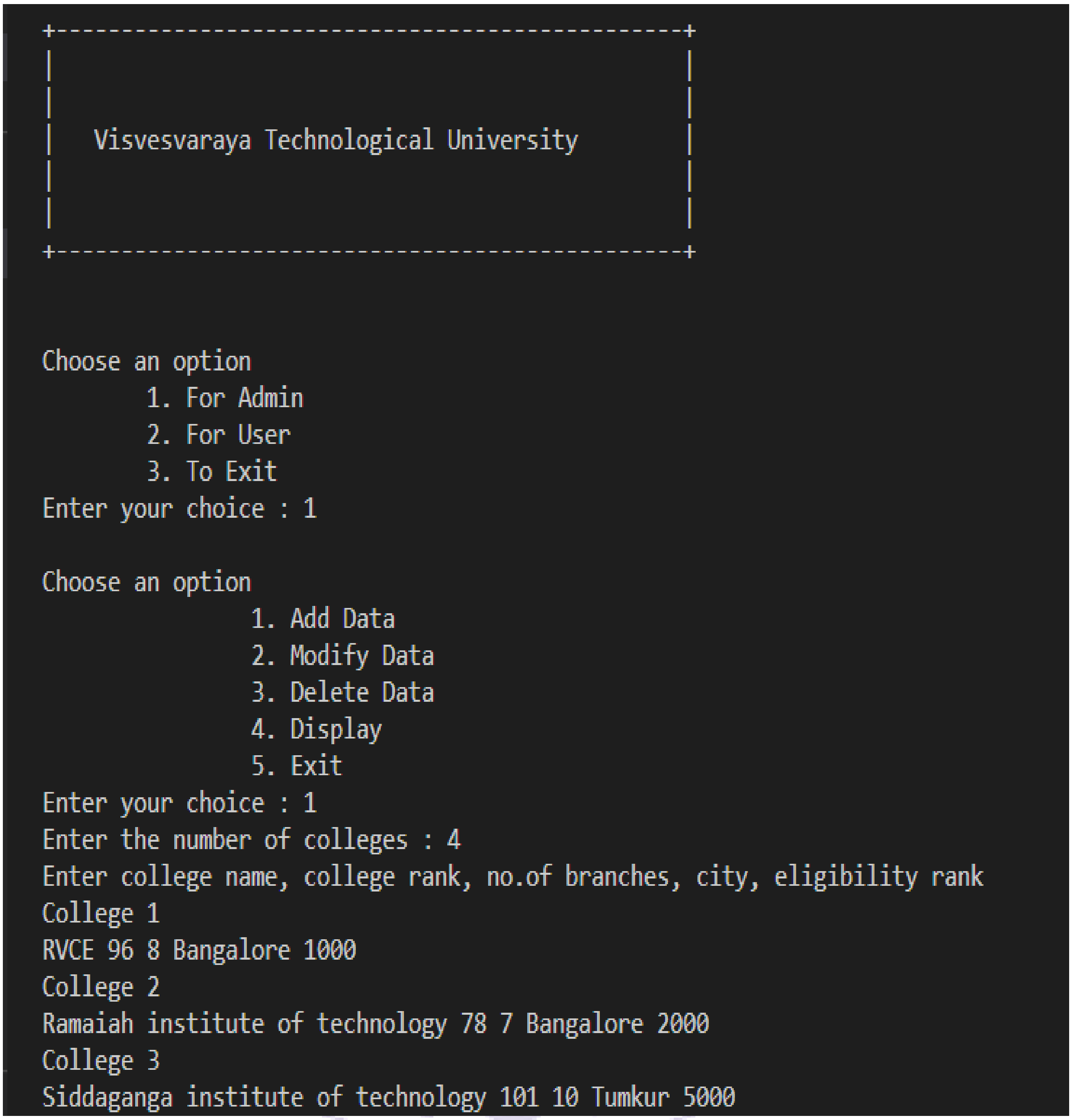
Expected Result: The system will work according to the chosen option.

4. Test Case: Login as a user

Description: Anyone can login as user to search the college details according to their requirements.

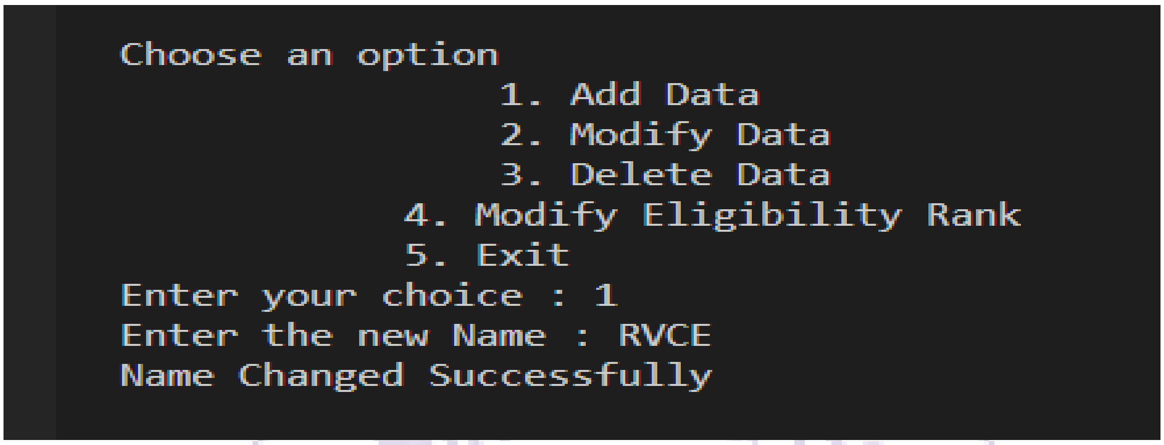
Expected Result:The system will show the results according to the requirements of user.

5.2 RESULTS:



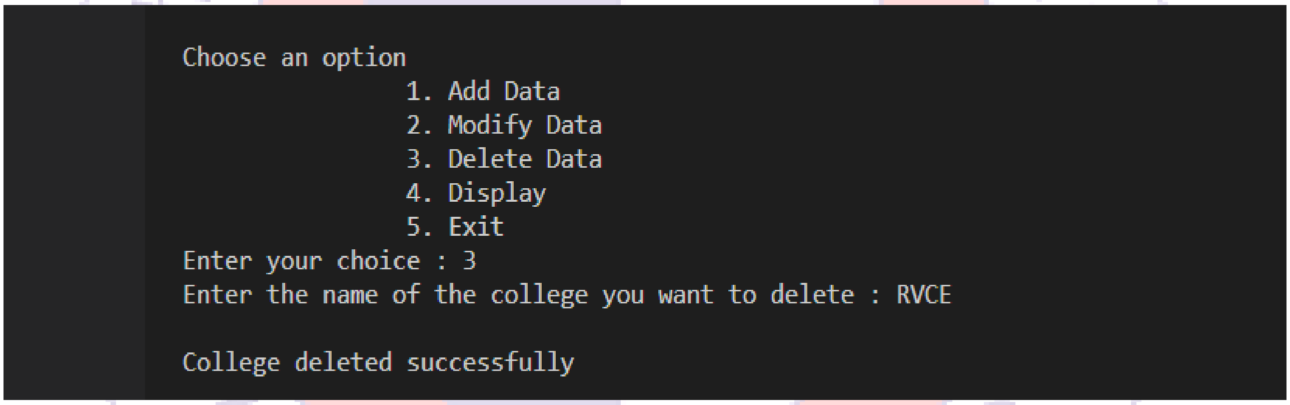
```
+-------------------------------------------------+
|                                                 |
|                                                 |
|   Visvesvaraya Technological University         |
|                                                 |
|                                                 |
+-------------------------------------------------+



Choose an option
        1. For Admin
        2. For User
        3. To Exit
Enter your choice : 1

Choose an option
                1. Add Data
                2. Modify Data
                3. Delete Data
                4. Display
                5. Exit
Enter your choice : 1
Enter the number of colleges : 4
Enter college name, college rank, no.of branches, city, eligibility rank
College 1
RVCE 96 8 Bangalore 1000
College 2
Ramaiah institute of technology 78 7 Bangalore 2000
College 3
Siddaganga institute of technology 101 10 Tumkur 5000
```

Fig.5.2. Login as Admin and add the college details



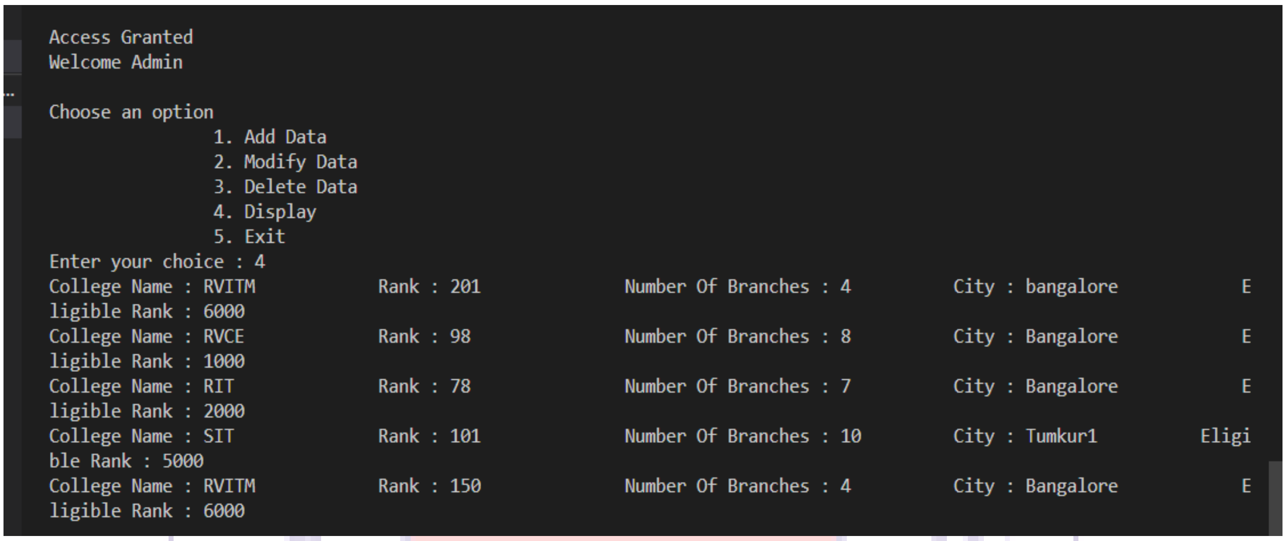
```
Choose an option
                    1. Add Data
                    2. Modify Data
                    3. Delete Data
               4. Modify Eligibility Rank
               5. Exit
Enter your choice : 1
Enter the new Name : RVCE
Name Changed Successfully
```

Fig.5.3. Modification of data



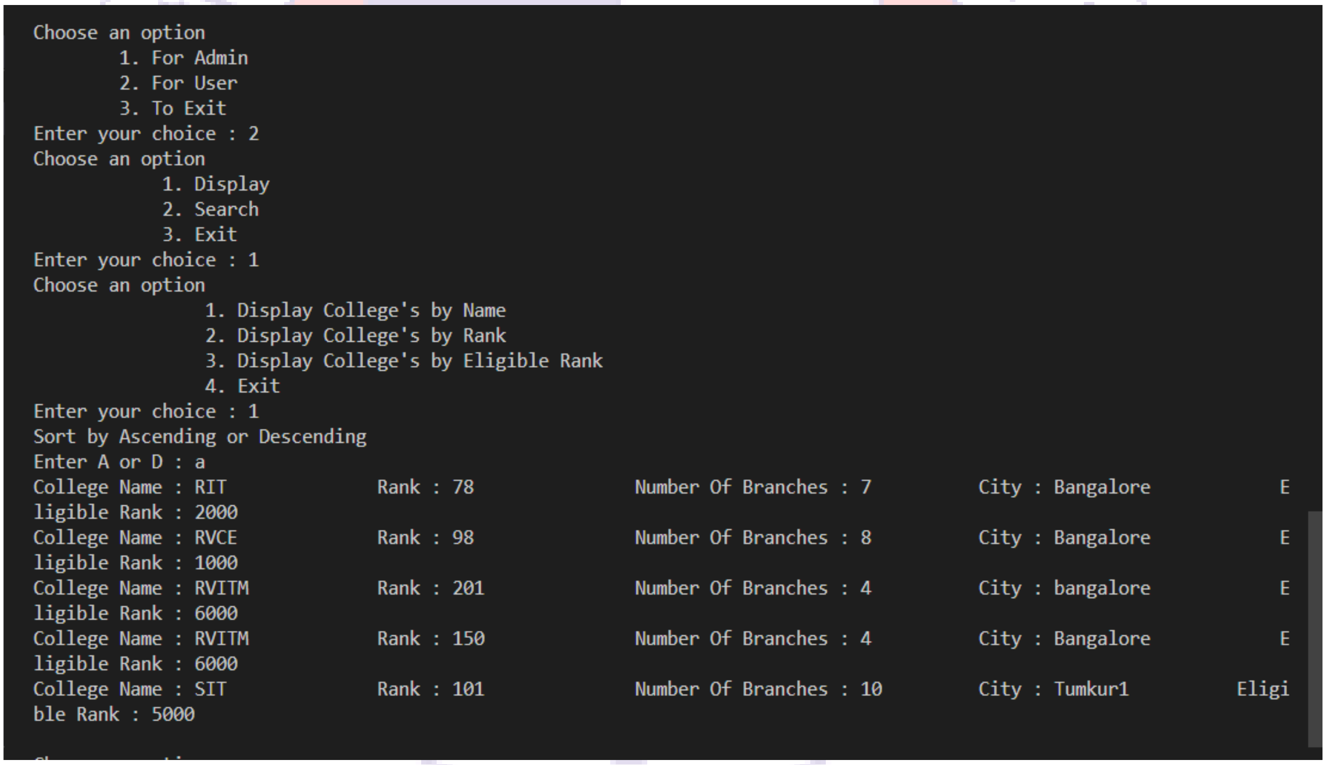
```
Choose an option
                1. Add Data
                2. Modify Data
                3. Delete Data
                4. Display
                5. Exit
Enter your choice : 3
Enter the name of the college you want to delete : RVCE

College deleted successfully
```

Fig.5.4. Deletion of data



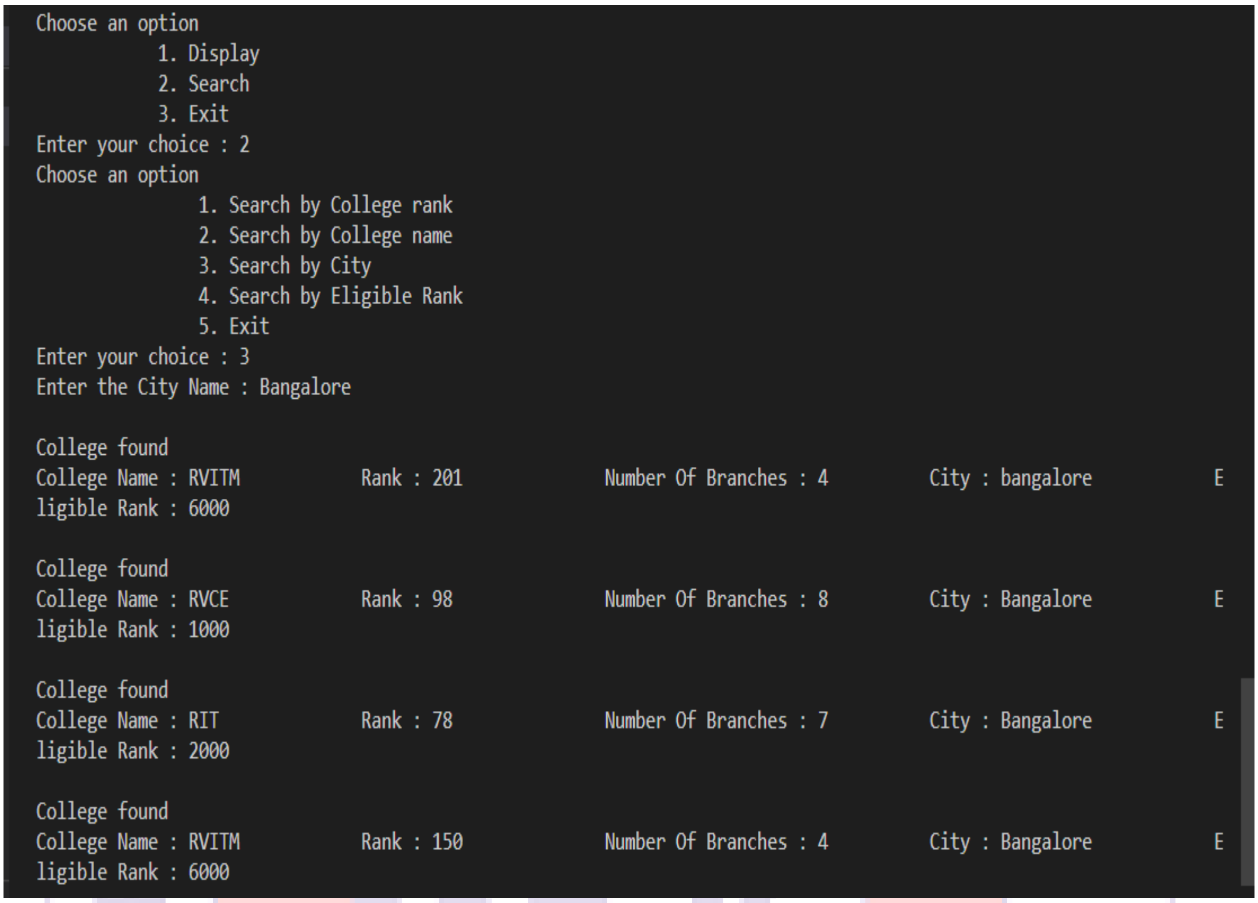
```
Access Granted
Welcome Admin

Choose an option
                1. Add Data
                2. Modify Data
                3. Delete Data
                4. Display
                5. Exit
Enter your choice : 4
College Name : RVITM        Rank : 201          Number Of Branches : 4      City : bangalore        E
ligible Rank : 6000
College Name : RVCE         Rank : 98           Number Of Branches : 8      City : Bangalore        E
ligible Rank : 1000
College Name : RIT          Rank : 78           Number Of Branches : 7      City : Bangalore        E
ligible Rank : 2000
College Name : SIT          Rank : 101          Number Of Branches : 10     City : Tumkur1      Eligi
ble Rank : 5000
College Name : RVITM        Rank : 150          Number Of Branches : 4      City : Bangalore        E
ligible Rank : 6000
```

Fig.5.5. Display of all the data



```
Choose an option
        1. For Admin
        2. For User
        3. To Exit
Enter your choice : 2
Choose an option
            1. Display
            2. Search
            3. Exit
Enter your choice : 1
Choose an option
                1. Display College's by Name
                2. Display College's by Rank
                3. Display College's by Eligible Rank
                4. Exit
Enter your choice : 1
Sort by Ascending or Descending
Enter A or D : a
College Name : RIT          Rank : 78           Number Of Branches : 7      City : Bangalore        E
ligible Rank : 2000
College Name : RVCE         Rank : 98           Number Of Branches : 8      City : Bangalore        E
ligible Rank : 1000
College Name : RVITM        Rank : 201          Number Of Branches : 4      City : bangalore        E
ligible Rank : 6000
College Name : RVITM        Rank : 150          Number Of Branches : 4      City : Bangalore        E
ligible Rank : 6000
College Name : SIT          Rank : 101          Number Of Branches : 10     City : Tumkur1      Eligi
ble Rank : 5000
```

Fig.5.6. Login as user and viewing all the data according to ascending or descending order



```
Choose an option
            1. Display
            2. Search
            3. Exit
Enter your choice : 2
Choose an option
                1. Search by College rank
                2. Search by College name
                3. Search by City
                4. Search by Eligible Rank
                5. Exit
Enter your choice : 3
Enter the City Name : Bangalore

College found
College Name : RVITM            Rank : 201              Number Of Branches : 4          City : bangalore            E
ligible Rank : 6000

College found
College Name : RVCE             Rank : 98               Number Of Branches : 8          City : Bangalore            E
ligible Rank : 1000

College found
College Name : RIT              Rank : 78               Number Of Branches : 7          City : Bangalore            E
ligible Rank : 2000

College found
College Name : RVITM            Rank : 150              Number Of Branches : 4          City : Bangalore            E
ligible Rank : 6000
```

Fig.5.7. Searching of the college details

Chapter 6

CONCLUSION

The university management system developed using a simple Python program provides a comprehensive solution for managing college details. The program allows for the addition of essential information such as college name, college city, number of branches, and eligibility rank. Additionally, it incorporates the functionality to display the college details in both ascending and descending order.

Sorting the college details allows users to analyze and compare the information based on specific criteria. Administrators, faculty, and other users can make informed decisions by examining colleges based on factors such as their names, cities, number of branches, or eligibility ranks. The ascending and descending order functionality offers flexibility and convenience in presenting the information in a meaningful way.

Overall, the university management system implemented through this Python program streamlines administrative tasks, facilitates efficient decision-making, and improves the overall management of colleges within the university. Its ability to add, organize, and sort college details provides a powerful tool for users to effectively manage and access information related to various colleges.

REFERENCES

[1]Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object-Oriented Approach withC++, 3rd Edition, Pearson Education, 1998.

[2]K.R.Venugopal, K.G.Srinivas,P.M. Krishnaraj: File Structures Using C++, Tata McGrawHill, 2008.

[3]https://itsourcecode.com/fyp/online-university-management-system-project-report-documentation/

[4]https://www.freeprojectz.com/python-project/university-management-system

[5] GeeksForGeeks-File Handling in Python(https://www.geeksforgeeks.org/file-handling-python/)

[6] Stack Overflow (https://stackoverflow.com/)

[7]JavaPoint-Python File Handling(https://www.javatpoint.com/python-files-io)