VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI – 590018

PROJECT REPORT ON

"FIND COLLEGE"

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE & ENGINEERING

FILE STRUCTURES LABORATORY WITH MINI PROJECT [18ISL67]

Submitted by

Kiran Kumar V– 4JK20IS023

Harsharaj B – 4JK20IS019

Under the guidance of

Prof Divya P	Prof Sivapuram Jayasri
Assistant Professor	Assistant Professor
Department of Information Science & Engineering	Department of Information Science & Engineering

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

A.J. INSTITUTE OF ENGINEERING & TECHNOLOGY

NH-66, KOTTARA CHOWKI, MANGALURU – 575006

2022 – 2023

A. J. INSTITUTE OF ENGINNERING & TECHNOLOGY

NH – 66, Kottara Chowki, Mangaluru - 575006

A Unit of Laxmi Memorial Education Trust (R)

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certify that the Project entitled **"FIND COLLEGE"** is carried out by Mr. **KIRAN KUMAR V**, USN: **4JK20IS023**, and Mr. **HARSHARAJ B**, USN: **4JK20IS019**, Student of sixth semester B.E. Information Science & Engineering, and submitted as a part of the course **FILE STRUCTURE LABORATORY WITH MINI PROJECT [18ISL67]** during the academic year 2022-2023.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of File Structure with Mini Project prescribed for the said Degree.

Prof Divya P	**Prof Jayashri Sivapuram**	**Dr Suresha D**
Project Guide	**Project Guide**	**Head of the Department**

Dr Shantharama Rai C
Principal

Examiners **Signature with Date**

1.

2.

ACKNOWLEDGEMENT

First and foremost, we thank our parents for what we are and where we are today, without whose hard work and sacrifice we would not be here today.

We deem it a privilege to place on record the deep sense of gratitude to our Project Guide **Prof. Divya P, Assistant Professor,** Department of Information Science and Engineering, and **Prof. Sivapuram Jayasri, Assistant Professor,** Department of Information Science and Engineering, who always stood behind us and supported in each step of the project work.

We are grateful to **Dr. Suresha D.,** Head of the Department, Information Science and Engineering for his support and encouragement.

We are indebted to our respected Principal **Dr. Shantharama Rai. C,** beloved Vice President **Mr. Prashanth Shetty** and the management of **A. J. Institute of Engineering and Technology, Mangaluru** for providing all the facilities that helped us in timely completion of this project report.

Finally, we would like to thank all the teaching and non-teaching staff of Department of Information Science and Engineering for their valuable help and support.

Kiran Kumar V	4JK20IS023
Harsharaj B	4JK20IS019

ABSTRACT

The "Find College" project is a file handling application developed using the Java programming language, specifically Java Swing. The aim of this project is to provide users with a convenient tool for searching and retrieving information about colleges. Using the Find College application, users can browse through a database of colleges, access their details, and find specific information such as location, courses offered, admission requirements, and contact information. The application employs file handling techniques to store and retrieve data, ensuring efficient and organized access to college information. The Java Swing framework enables the development of a user-friendly graphical user interface (GUI), allowing users to navigate through the application seamlessly. The intuitive design and functionality of the Find College project make it a useful resource for students, parents, and education professionals seeking comprehensive college information. By leveraging Java's file handling capabilities and the power of Java Swing, Find College offers a reliable and user-friendly solution for finding relevant college information with ease.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Find College project is a file handling application developed using the Java programming language, specifically Java Swing. It aims to provide users with a convenient tool for searching and retrieving information about colleges. With thousands of colleges worldwide, finding accurate and comprehensive information about each institution can be a daunting task. Traditional methods such as browsing through websites or contacting individual colleges can be time-consuming and inefficient. The Find College application addresses these challenges by offering a streamlined and efficient approach to accessing college details.

1.1 Problem definition

The traditional methods of searching for college information, such as browsing through numerous websites or relying on outdated printed directories, present several challenges. Firstly, the information available online is often scattered across multiple sources, making it difficult for users to access comprehensive and accurate data in one place. Additionally, manually gathering information from different websites can be time-consuming and tedious. Furthermore, the lack of a centralized platform that organizes college data hampers the efficiency of the search process. These challenges make it essential to develop an application that simplifies the search for college information and provides users with a streamlined experience.

1.2 Objectives of the project

The Find College project has the following key objectives:

1.2.1 Simplify College Search

The primary objective of the project is to simplify the process of finding and gathering information about colleges. By providing a user-friendly interface and organized data, the application aims to streamline the college search experience.

1.2.2 Comprehensive Information

The project aims to compile a comprehensive database of colleges, including their location, courses offered, admission requirements, and contact information. This ensures that users have access to all relevant details in one centralized platform.

1.2.3 Efficient Data Storage and Retrieval

The project utilizes file handling techniques in Java to ensure efficient storage and retrieval of college data. By employing proper file handling mechanisms, such as reading and writing data to files, the application optimizes the speed and accuracy of accessing college information. This ensures that users can quickly retrieve the details they require without any significant delays or errors.

1.2.4 User-Friendly Interface

The Find College project utilizes Java Swing, a powerful framework for creating graphical user interfaces, to develop an intuitive and user-friendly interface. The GUI will enable users to navigate through the application seamlessly, browse colleges, and access detailed information effortlessly. The interface will incorporate intuitive search functionalities and filtering options to enhance the user experience and make the process of finding college information more convenient.

1.2.5 Reliable and Accurate Information

The project aims to provide reliable and accurate information about colleges by implementing data validation and regular updates. Users can trust the information provided by the application to make informed decisions about their college choices.

1.2.6 Updated and Accurate Information

The project aims to provide users with the most updated and accurate information about colleges. It will incorporate mechanisms to ensure that the data stored in the application's database is regularly updated, eliminating the risk of outdated information. By providing users with reliable and current details, Find College aims to enhance the decision-making process when it comes to selecting a college.

1.2.7 Centralized College Database

The project seeks to create a centralized database that stores relevant information about various colleges. This database will include details such as college names, locations, courses offered, admission requirements, contact information, and any other pertinent data. By organizing the information in a structured manner, users can easily access and retrieve specific details about colleges of interest.

CHAPTER 2

REQUIREMENT SPECIFICATION

The requirement specification for the Find College project outlines the necessary software and hardware components for the successful implementation and operation of the Java-based application. Meeting these requirements ensures compatibility, performance, and a user-friendly experience for accessing comprehensive college information.

2.1 Software requirements

The Find College project has the following software requirements:

2.1.1 Java Development Kit (JDK)

The app is developed using Android Studio, an integrated development environment (IDE) for Android application development. The latest stable version of Android Studio should be installed to ensure compatibility and access to the latest features and updates.

2.1.2 Integrated Development Environment (IDE)

An IDE like Eclipse, IntelliJ IDEA, or NetBeans is recommended for development purposes. These IDEs offer features such as code editing, debugging, and project management, which streamline the development process.

2.1.3 Java Swing Library

The project utilizes the Java Swing library, which is included in the JDK, to create a graphical user interface (GUI). The Java Swing library provides a rich set of components and layouts for building interactive GUI applications.

2.2 Hardware requirements

The Find College application has the following hardware requirements:

1. Computer: A desktop or laptop computer capable of running Android Studio Operating System: Windows 7/8/10, macOS, or Linux.
2. Processor: Intel Core i5 or higher.
3. RAM: Minimum 4GB.
4. Storage: Sufficient free disk space for Android Studio installation and project files.
5. Display: Minimum 1024x768 pixels is recommended to display the GUI effectively. The display quality impacts the clarity and usability of the application's interface.

2.2.1 Development Machine

A capable development machine with sufficient processing power, memory, and storage is required for developers working on the Find College project. It should also have the necessary software components and a stable internet connection for seamless development and collaboration.

2.3 Functional requirements

The functional requirements describe the desired functionalities and features of the Find College Project:

- ***Display College Information:***
  The dialog should be able to display the information of colleges, including their college IDs, names, percentages, fees, locations, contact details, email addresses, and websites.
- ***Retrieve College Data:***
  The dialog should retrieve the college data from a data source, such as a database, to populate the information to be displayed. It should handle exceptions gracefully and provide appropriate error messages if there is a problem accessing the data.
- ***Refresh College Data:***
  The dialog should provide a mechanism to refresh or update the displayed college data. This could be through a button or an automatic refresh feature, ensuring that the latest information is retrieved and displayed.
- ***Handle Empty College List:***
  If there are no colleges available in the data source, the dialog should display a message indicating that no colleges are found. It should also provide guidance on adding colleges to the system.
- ***Close Dialog:***
  The dialog should allow the user to close it when they are done viewing the colleges. This can be done through a button or by closing the dialog window.
- ***User Interface:***
  The dialog should have a visually pleasing and user-friendly interface, with appropriate spacing, layout, and formatting of the displayed college information. It should be resizable and responsive to different screen sizes.
- ***Integration:***
  The dialog should seamlessly integrate with the existing application or system, ensuring that it works correctly within the overall context and flow of the user interface.

CHAPTER 3

SYSTEM DESIGN

The Find College project offers an intuitive file handling system developed using Java Swing, empowering users to effortlessly search and access college information. In lieu of traditional database storage, the project employs a novel approach, leveraging multiple .txt files to effectively organize and manage data. With its user-friendly interface, the system allows administrators to securely add, view, and delete college details, while college users can update their information and students can input their marks to find suitable colleges. The project merges the power of Java programming and file handling techniques to streamline the college search process.

3.1 Algorithm

This algorithm describes the steps involved in the user interface and functionality of a program for updating user information in a dialog.

Step 1: START

Step 2: Create an instance of the UserUpdateDialog class.

Step 3: Set the content pane of the dialog.

Step 4: Set the dialog as modal.

Step 5: Set the default button of the dialog.

Step 6: Set the size of the dialog to 500x400 pixels.

Step 7: Set the title of the dialog as "User Update Dialog".

Step 8: Set the location of the dialog to the center of the screen.

Step 9: Add an action listener to the "OK" button of the dialog.

Step 10: Inside the action listener, perform the following steps:

- Call the onOK() method.
- Initialize a variable "valid" as true.
- Create an instance of the FS_Execute class.
- If the "Contact" text field is not empty:
- If the entered value matches the pattern of a 10-digit number:
- Call the StudentUpdate() method of FS_Execute to update the contact information.
- Else, display a message dialog indicating an invalid contact number and set "valid" to false.

- If the "Email" text field is not empty:
- If the entered value matches the pattern of a valid email address:
- Call the StudentUpdate() method of FS_Execute to update the email information.
- Else, display a message dialog indicating an invalid email address and set "valid" to false.
- If the "Location" text field is not empty:
- Call the StudentUpdate() method of FS_Execute to update the location information.
- If "valid" is true:
- If all the text fields are empty, dispose of the dialog.
- Else, display a message dialog indicating a successful update and dispose of the dialog.

Step 11: Add an action listener to the "Cancel" button of the dialog.

Step 12: Inside the action listener, perform the following steps:

- Call the onCancel() method.
- Dispose of the dialog.

Step 13: Set the default close operation of the dialog to DO_NOTHING_ON_CLOSE.

Step 14: Add a window listener to the dialog to handle the window closing event.

Step 15: Inside the window listener, perform the following steps:

- Call the onCancel() method.
- Dispose of the dialog.

Step 16: Register the ESCAPE key press event on the content pane of the dialog to call the onCancel() method.

Step 17: END

3.2 Flow Chart

The flowchart diagram illustrates the sequential steps and decision points in the "Find College" project, showcasing the functionalities for Admin, College, and Student users.

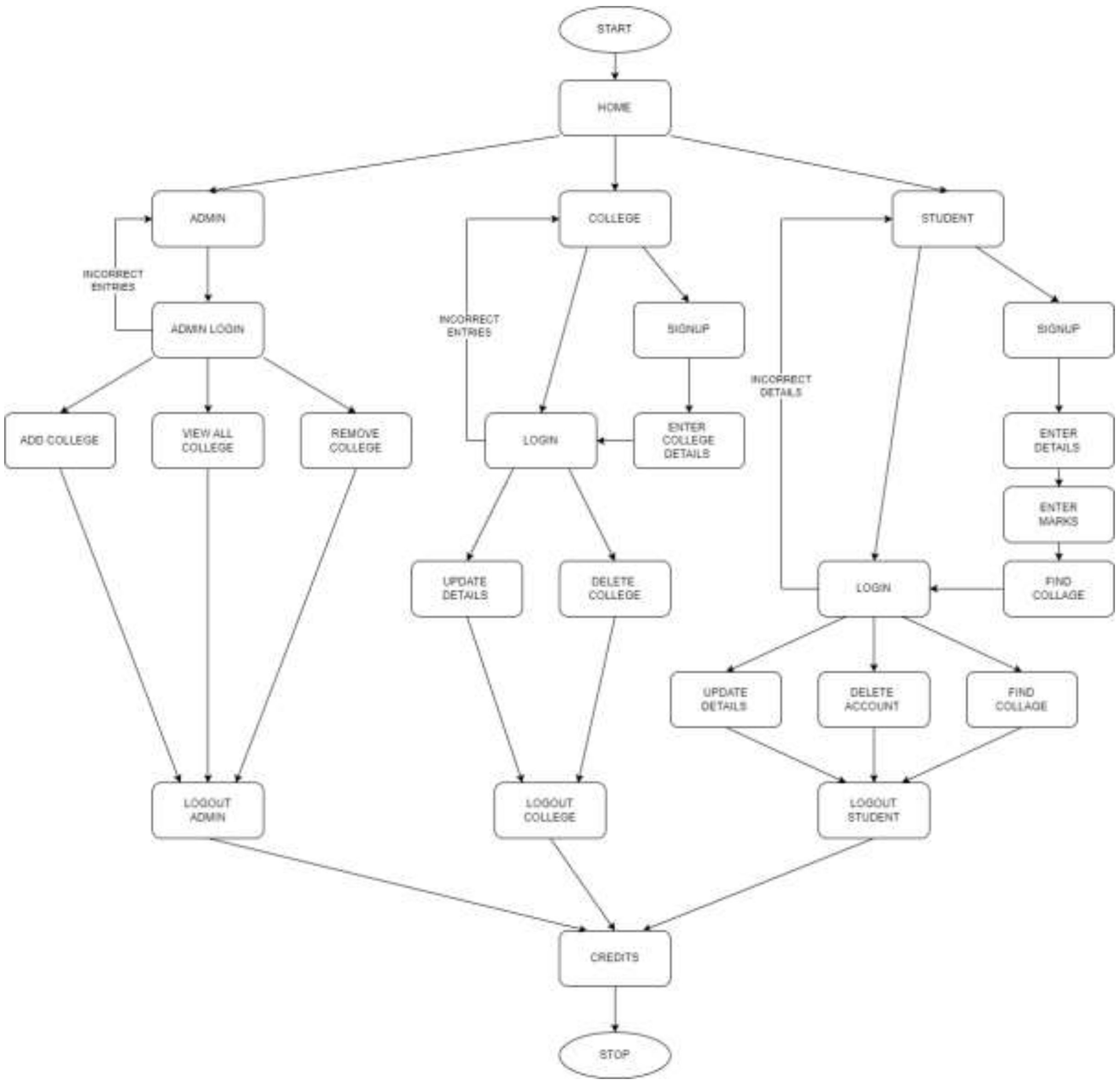


Figure 3.2.1: Flow Chart Diagram

The Find College flow chart diagram illustrates a comprehensive system where administrators can log in, manage colleges (add, view, and remove), while colleges can log in or sign up, manage their details (add, update, and remove), and students can log in, enter their details, search for colleges, update their information, delete their accounts, and view available colleges.

3.3 Use Case Diagram

The Use Case Diagram showcases the interactions between actors and use cases, offering a concise visual representation of the system's functionality and user interactions.

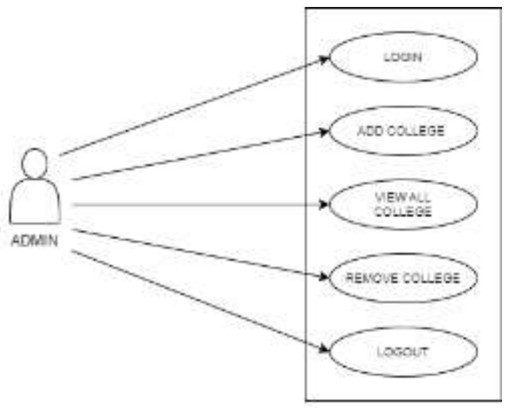


Figure 3.3.1: Admin Use Case Diagram

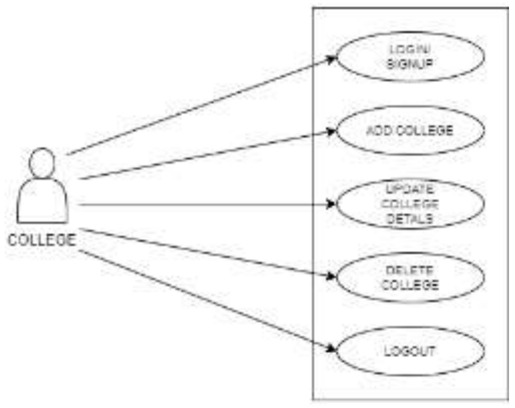


Figure 3.3.2: College Use Case Diagram

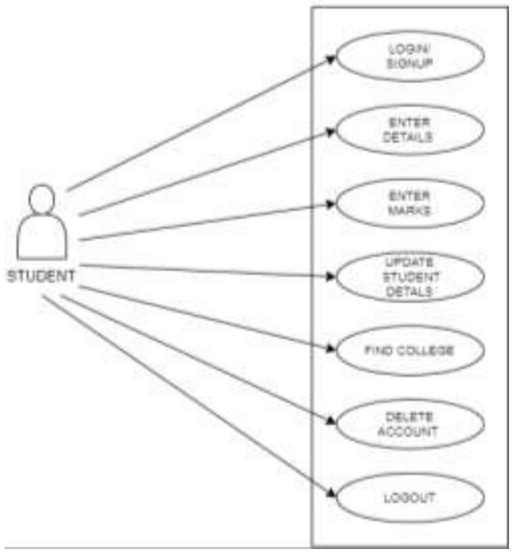


Figure 3.3.3: Student Use Case Diagram

In Find College use case: Admin manages operations, college provides services, students utilize resources.

CHAPTER 4

IMPLEMENTATION

The implementation of the "Find College" project involved utilizing the Java programming language with the Java Swing framework to develop a user-friendly interface for college searching and management. File handling techniques were employed instead of a traditional database, using multiple .txt files to store and retrieve the relevant data efficiently.

4.1 Programming languages used

The Find College project is developed using Java and the Java Swing framework, providing a user-friendly graphical interface for college search and management operations.

Java is a widely-used, high-level programming language known for its platform independence, strong community support, and extensive library ecosystem. Java Swing is a GUI (Graphical User Interface) toolkit provided by Java, offering a rich set of components and tools for creating interactive and visually appealing desktop applications. It enables developers to build cross-platform applications with ease, allowing for the development of intuitive and responsive user interfaces.

4.2 File Handling

In the "Find College" project, implemented in Java with the Java Swing framework, file handling techniques are utilized for data storage and retrieval. Instead of using a database, various .txt files are employed to store and manage specific information related to colleges, administrators, and students. File handling enables efficient data organization, retrieval, and modification within the project, providing a convenient and portable solution for data management without the need for a dedicated database system.

4.3 Data Structures

The provided code is part of a project that involves managing college information. It includes a user interface for updating user details and viewing a list of colleges. The project utilizes built-in Java data structures such as ArrayList and JTextArea for storing and displaying college data. While the specific data structures used in the project are limited, they facilitate the organization and presentation of college information within the application.

4.4 User Defined Functions

Admin Class:

addCollege(): Adds a new college with the provided details.

viewAllColleges(): Retrieves a list of all colleges.

deleteCollege(): Deletes a specific college based on the provided college ID.

logout(): Logs out the admin from the system.

College Class:

saveToFile(): Saves the college details to a file.

loadAllColleges(): Loads all colleges from the file and returns a list.

deleteCollege(): Deletes a specific college from the file.

getNextCollegeId(): Retrieves the next available college ID.

Student Class:

signUp(): Registers a new student with the provided details.

enterMarks(): Records the marks of a student for various subjects.

findCollege(): Searches for colleges based on student's marks or preferences.

4.5 Code snippets

InsertValue function:

```
int InsertValues(College clg) throws Exception {
    String fileName = "Colleges.txt";
    // This is the File Name of the table i.e.  Colleges.txt
    File file = new File(FS_FolderName + fileName);
    ViewAllColleges();
    for (College college : College_Main.clgs) {
       if (college.getName().equals(clg.getName()) ||
       college.getEmail().equals(clg.getEmail()) ||
       college.getWebsite().equals(clg.getWebsite()))
          return 1;
    }
    ArrayList<String> fieldsToWrite = new ArrayList<>();
    fieldsToWrite.add(clg.getName());
    fieldsToWrite.add(String.valueOf(clg.getPercentage()));
```

```
        fieldsToWrite.add(String.valueOf(clg.getFees()));
        fieldsToWrite.add(clg.getLocation());
        fieldsToWrite.add(clg.getContact());
        fieldsToWrite.add(clg.getEmail());
        fieldsToWrite.add(clg.getWebsite());
        // NOTE: "College_IDs.txt" contains all the used IDs so after reading the last line
        // we will generate the next number as the new ID.
        int newClgID = getNewClgID_O_1(); // Get new ID.
        writeTheseFields(file, fieldsToWrite, newClgID);
        String lineToWrite;
        fileName = "Login_Colleges.txt";
        file = new File(FS_FolderName + fileName);
        fieldsToWrite.add(String.valueOf(newClgID));
        fieldsToWrite.add(clg.getEmail());
        fieldsToWrite.add(clg.getPassword());
        lineToWrite = String.join("|", fieldsToWrite);
        writeTheseFields(file, fieldsToWrite, lineToWrite);
        fileName = "College_Name_ID.txt";
        file = new File(FS_FolderName + fileName);
        fieldsToWrite.add(clg.getName());
        fieldsToWrite.add(String.valueOf(newClgID));
        lineToWrite = String.join("|", fieldsToWrite);
        writeTheseFields(file, fieldsToWrite, lineToWrite);
        PointersRecord pointersRecord = new PointersRecord();
        pointersRecord.createPointerRecords();
        return 0;
    }
```

DeleteCollege function:

```
    void DeleteCollege(String nameOrEmail) throws IOException {
     if (!nameOrEmail.contains("@")) {
            File file = new File(FS_FolderName + "CollegesPointerName.txt");
            String found = search(file, nameOrEmail);
            file = new File(FS_FolderName + "Colleges.txt");
```

```
                long pos = Long.parseLong(found.split("\\|")[1]);
                String deletedString = delete(file, pos);
                String key = deletedString.split("\\|")[0];
                file = new File(FS_FolderName + "CollegesPointers.txt");
                found = search(file, key);
                file = new File(FS_FolderName + "Login_Colleges.txt");
                pos = Long.parseLong(found.split("\\|")[2]);
                delete(file, pos);
        } else {
                File file = new File(FS_FolderName + "CollegesPointerEmail.txt");
                String found = search(file, nameOrEmail);
                file = new File(FS_FolderName + "Colleges.txt");
                long pos = Long.parseLong(found.split("\\|")[1]);
                delete(file, pos);
                file = new File(FS_FolderName + "Login_Colleges.txt");
                pos = Long.parseLong(found.split("\\|")[2]);
                delete(file, pos);
        }
      PointersRecord pointersRecord = new PointersRecord();
      pointersRecord.createPointerRecords();
      College_Main.clgs.clear();
   }
```

ViewAllColleges function:

```
        void ViewAllColleges() throws IOException {
        String fileName = "Colleges.txt";
        // This is the File Name of the table i.e. Colleges.txt
         File file = new File(FS_FolderName + fileName);
        Scanner scanner = new Scanner(file);
        String readLine; // This to read the line from the TXT file.
        String[] currentLine;
        // This line which was read before was of a STRING datatype we need to split each //
        // field and store it as ARRAY.
        College_Main.clgs.clear();
        while (scanner.hasNextLine()) {
                readLine = scanner.nextLine();
                currentLine = readLine.split("\\|");
                College college = new College();
                college.setCollege_id(Integer.parseInt(currentLine[0]));
                college.setName(currentLine[1]);
                college.setPercentage(Float.parseFloat(currentLine[2]));
                college.setFees(Integer.parseInt(currentLine[3]));
                college.setLocation(currentLine[4]);
                college.setContact(currentLine[5]);
                college.setEmail(currentLine[6]);
                college.setWebsite(currentLine[7]);
                College_Main.addClg(college);
      }
      scanner.close();
   }
```

Search function:

```
        static String search(File file, String key) throws IOException {
        ArrayList<String> stringArrayList = new ArrayList<>();
        RandomAccessFile randomAccessFile = new RandomAccessFile(file, "r");
        ArrayList<Long> arrayListPos = new ArrayList<>();
        while (randomAccessFile.getFilePointer() < randomAccessFile.length()) {
                arrayListPos.add(randomAccessFile.getFilePointer());
                stringArrayList.add(randomAccessFile.readLine().strip().split("\\|")[0]);
      }
      int l = 0, r = stringArrayList.size() - 1;
      while (l <= r) {
        int m = l + (r - l) / 2;
        int res = key.compareTo(stringArrayList.get(m));
        // Check if x is present at mid
        if (res == 0) {
           randomAccessFile.seek(arrayListPos.get(m));
           String found = randomAccessFile.readLine();
```

```
            randomAccessFile.close();
            return found;
        }
        // If x greater, ignore left half
        if (res > 0)
            l = m + 1;
            // If x is smaller, ignore right half
        else
            r = m - 1;
    }
    randomAccessFile.close();
    return "false";
}
```

CollegeUpdate function:

```
        void CollegeUpdate(int fees, String email) throws IOException {
        File file1 = new File(FS_FolderName + "CollegesPointerEmail.txt");
        File file2 = new File(FS_FolderName + "Colleges.txt");
        int[] keyFieldsPointers = new int[]{1};
        String updatedValue = String.valueOf(fees);
        String oldValue = email;
        int[] fieldIndexesToChange = new int[]{3};
        UpdateWithNewValue(new File[]{file1, file2}, keyFieldsPointers, updatedValue,
                                    oldValue, fieldIndexesToChange);
        PointersRecord pointersRecord = new PointersRecord();
        pointersRecord.createPointerRecords();
}
```

sort function:

```
        static void sort(File file, int key) throws IOException {
        List<String> arrayList = new ArrayList<>(Files.readAllLines(file.toPath(),
                                                        StandardCharsets.UTF_8));
        String temp;
        for (int i = 0; i < arrayList.size(); i++) {
          for (int j = i + 1; j < arrayList.size(); j++) {
            if (arrayList.get(i).split("\\|")[key].compareTo(arrayList.get(j).split("\\|")[key]) > 0)
        {
                temp = arrayList.get(i);
                arrayList.set(i, arrayList.get(j));
                arrayList.set(j, temp);
            }
        }
    }
    try (FileWriter f = new FileWriter(file, false); BufferedWriter b = new
          BufferedWriter(f); PrintWriter p = new PrintWriter(b)) {
        for (String s : arrayList) {
            p.println(s); // Writes s to the text file.
        }
    } catch (IOException i) {
        i.printStackTrace();
    }
}
```

CHAPTER 5

RESULT

The Result section of the College Project will showcase snapshots of the Java Swing GUI interface, providing a visual representation of the project's file handling functionality and user-friendly design.

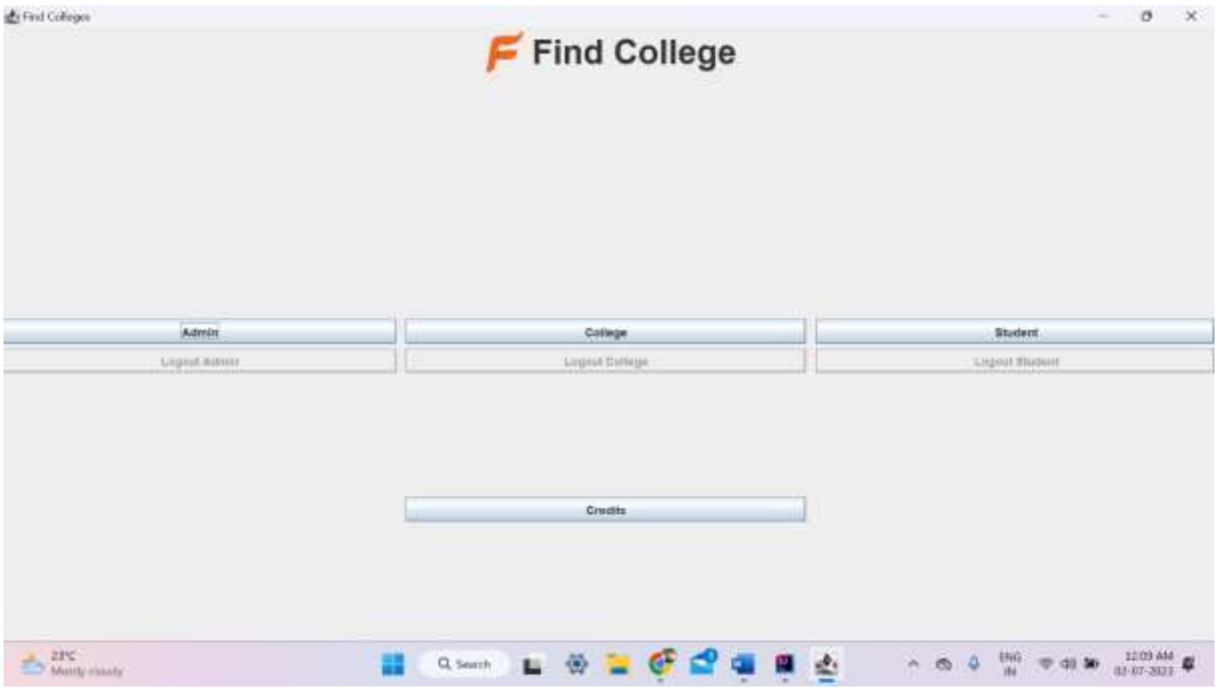


Figure 5.1: Home Page

The snapshot showcases the home screen of the College Project, featuring a neatly organized interface divided into three sections. The first section is dedicated to the admin, allowing them to access various administrative functionalities. The second section is specifically designed for college-related tasks, providing easy access to college-specific features. The third section is dedicated to students, enabling them to access student-related functionalities conveniently. Each section includes a "Logout" button, ensuring secure access control. At the bottom center of the interface, there is a credit button that reveals details about the creators of the project, giving due recognition to the individuals behind its development.

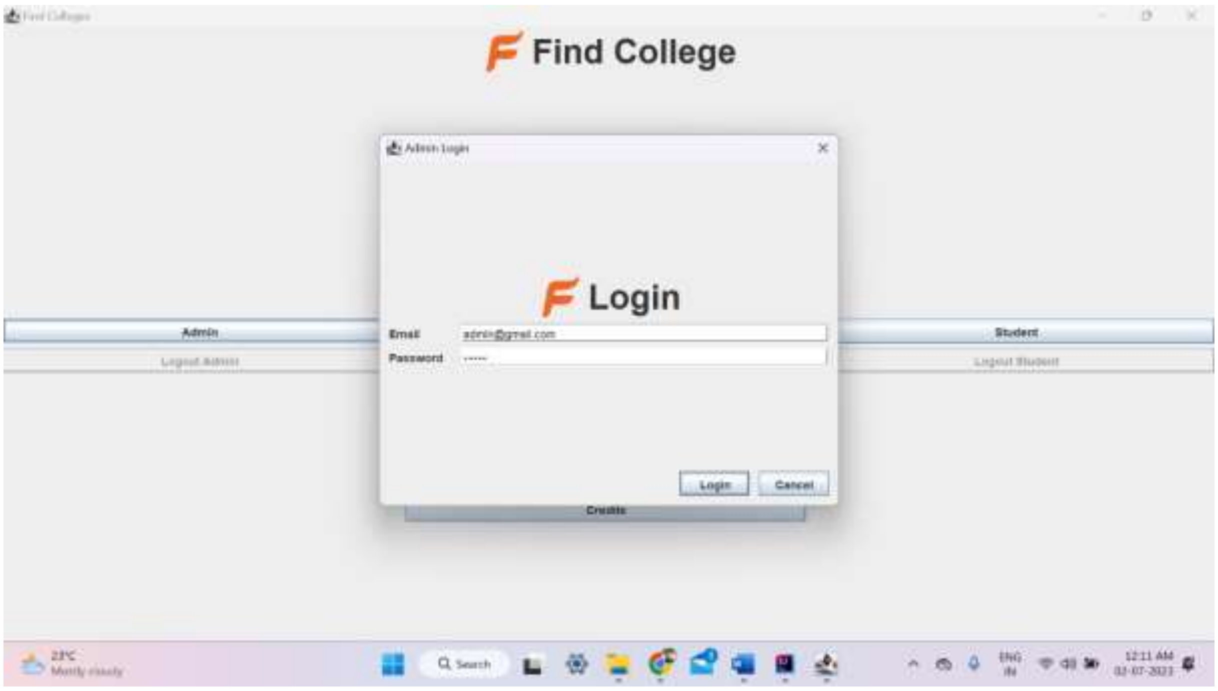


Figure 5.2: Admin Login Page

Secure access for administrators, providing exclusive control over the Find College functionalities.

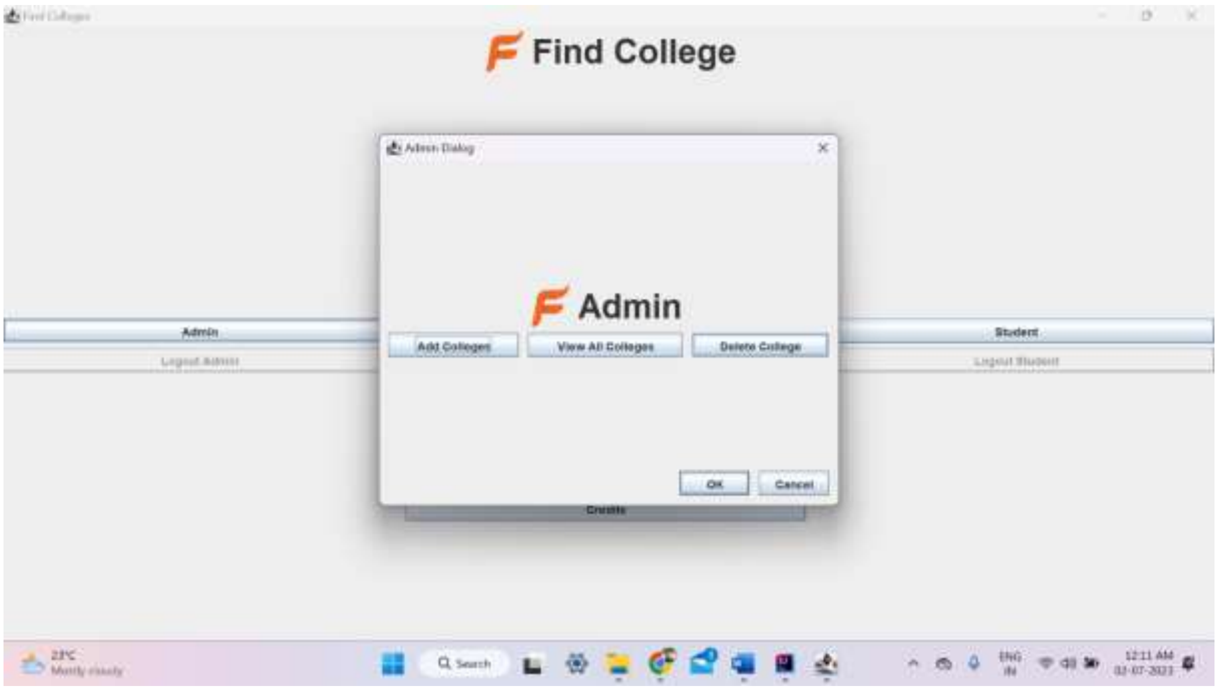
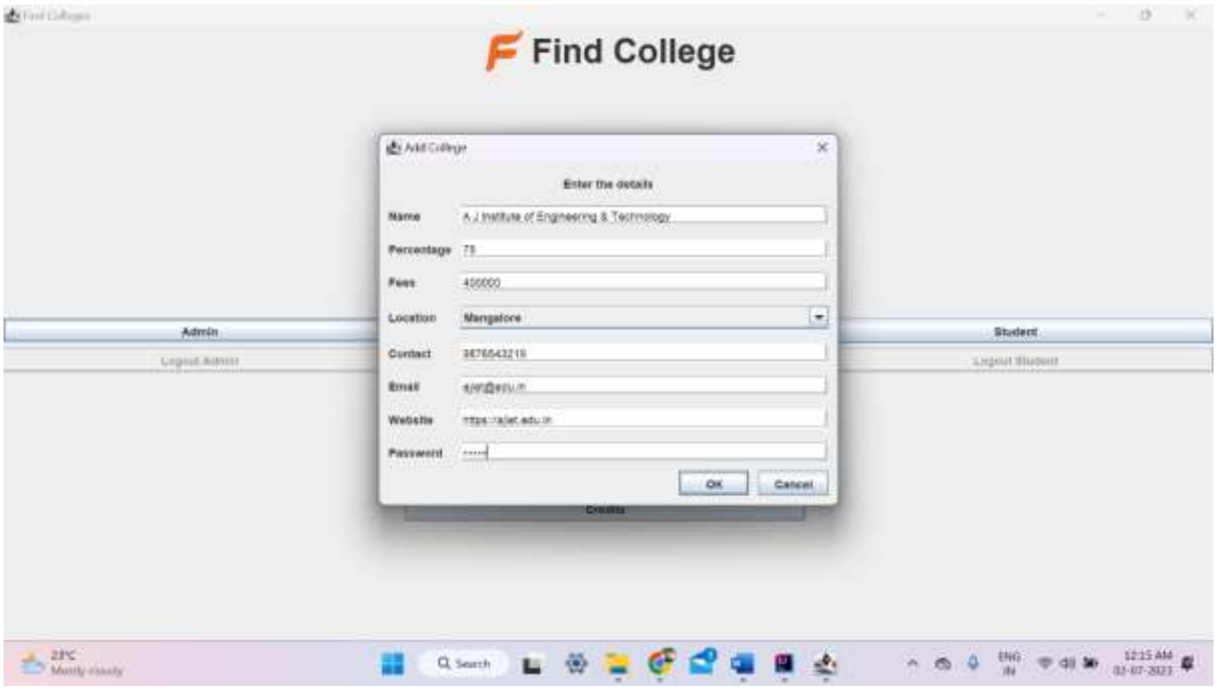


Figure 5.3: Admin Main Page

Manage Colleges effortlessly with Add, View, and Delete functionalities, streamlining administrative tasks.

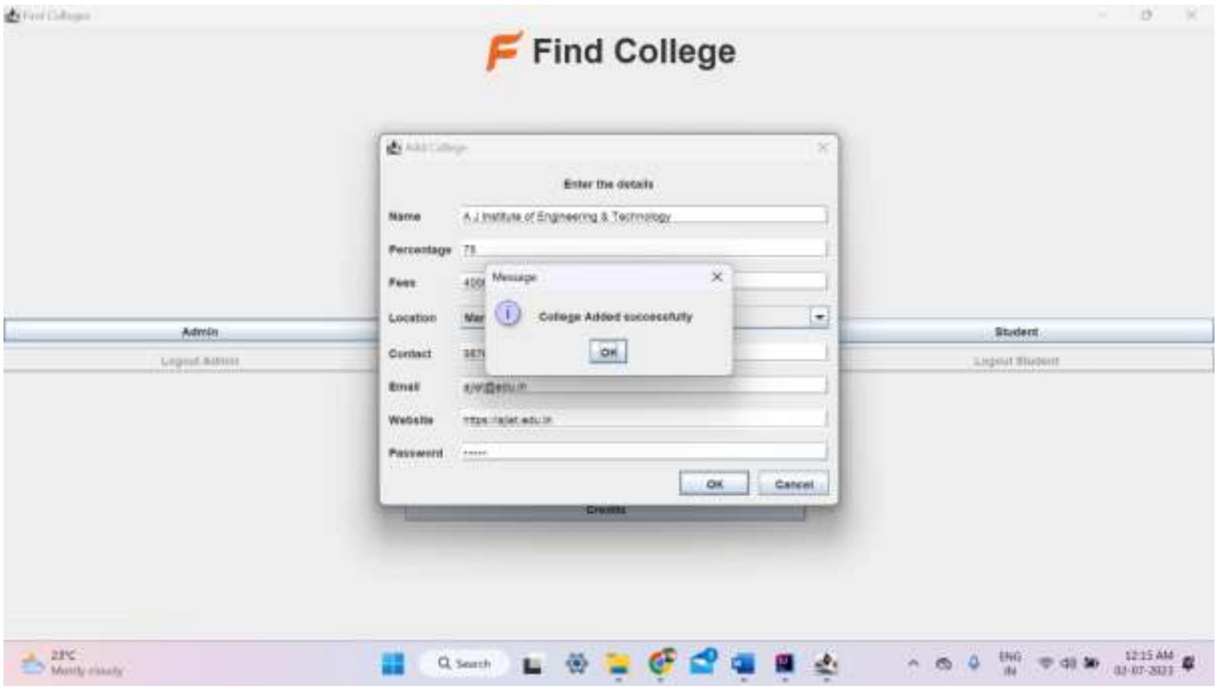


Enter the details

Name	A J Institute of Engineering & Technology
Percentage	75
Fees	400000
Location	Mangalore
Contact	9876543210
Email	ajiet@edu.in
Website	https://ajiet.edu.in
Password	••••••

Figure 5.4: Admin – Add Colleges

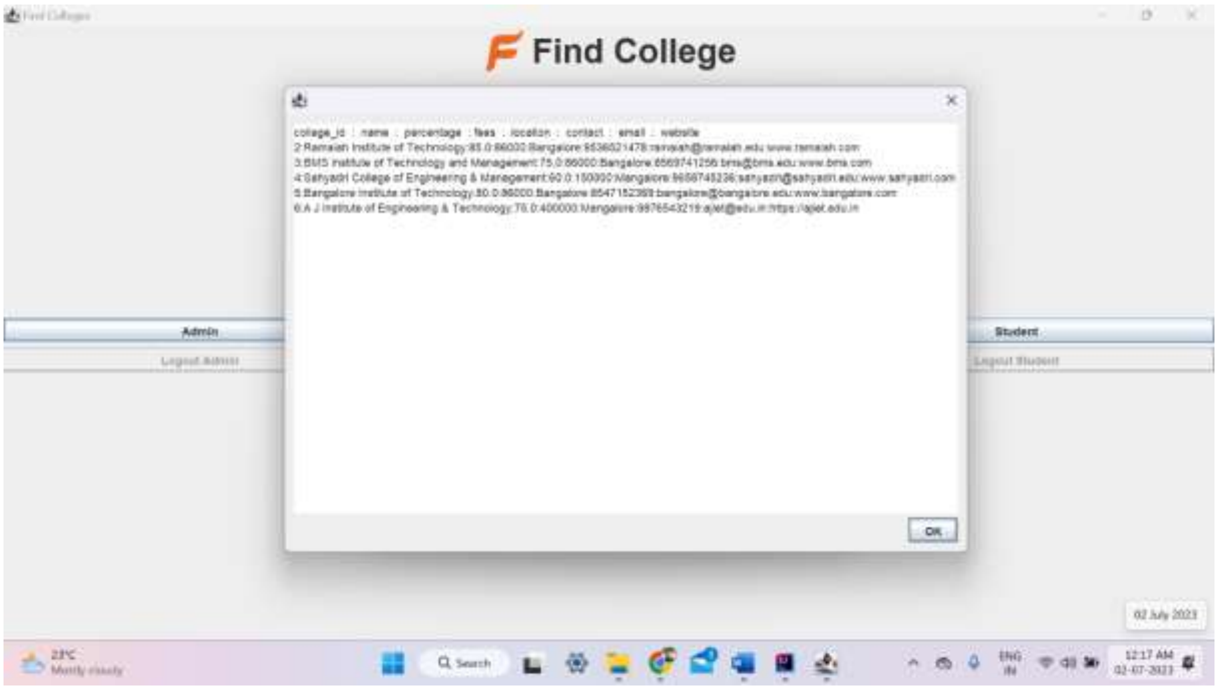
The snapshot showcases the "Enter college details" where users can input college details along with an email and password, facilitating seamless creation and management of college accounts.



College Added successfully

Figure 5.5: Admin - College Added Successfully Message

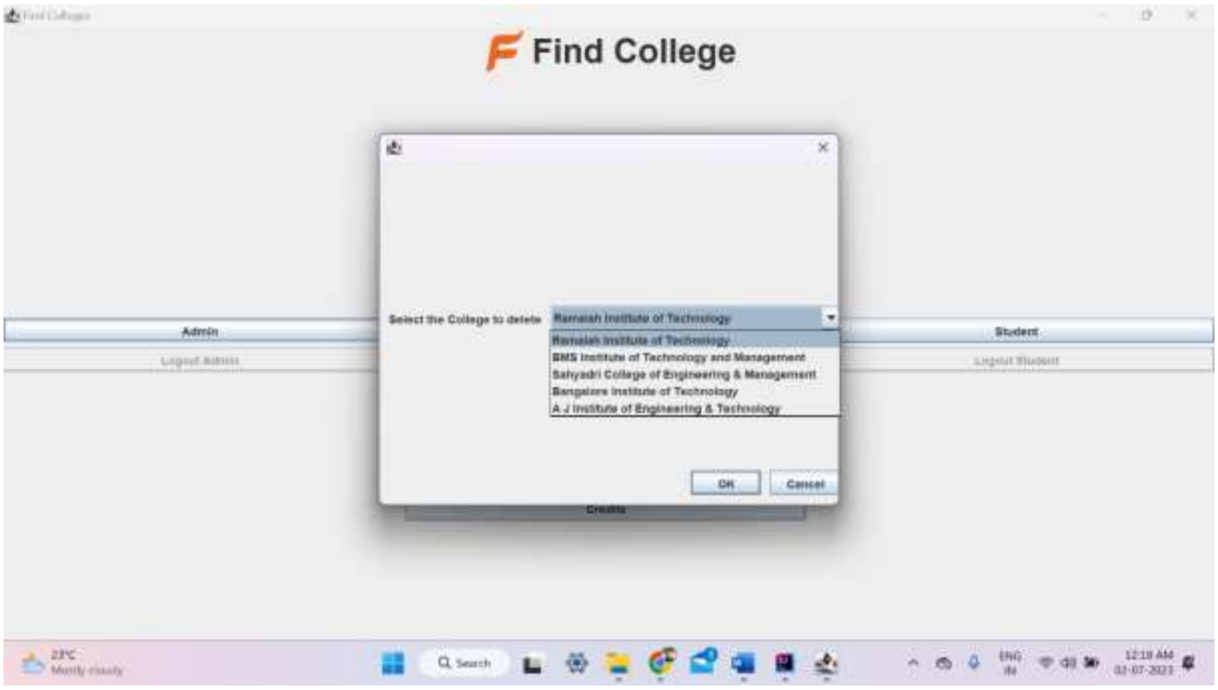
The snapshot captures the message "College Added Successfully," confirming the successful addition of a college in the College Project and providing a sense of accomplishment.



```
college_id : name : percentage : fees : location : contact : email : website
2:Ramaiah Institute of Technology:85.0:86000:Bangalore:9536521478:ramaiah@ramaiah.edu:www.ramaiah.com
3:BMS Institute of Technology and Management:75.0:86000:Bangalore:6569741256:bms@bms.edu:www.bms.com
4:Sahyadri College of Engineering & Management:90.0:150000:Mangalore:9656745236:sahyadri@sahyadri.edu:www.sahyadri.com
5:Bangalore Institute of Technology:80.0:86000:Bangalore:8547152369:bangalore@bangalore.edu:www.bangalore.com
6:A J Institute of Engineering & Technology:75.0:400000:Mangalore:9976543219:ajiet@edu.in:https://ajiet.edu.in
```

Figure 5.6: Admin - View All Colleges

The snapshot presents the "View All Colleges" list, offering a comprehensive view of all the colleges registered in the system, facilitating easy navigation and access to college information in the College Project.

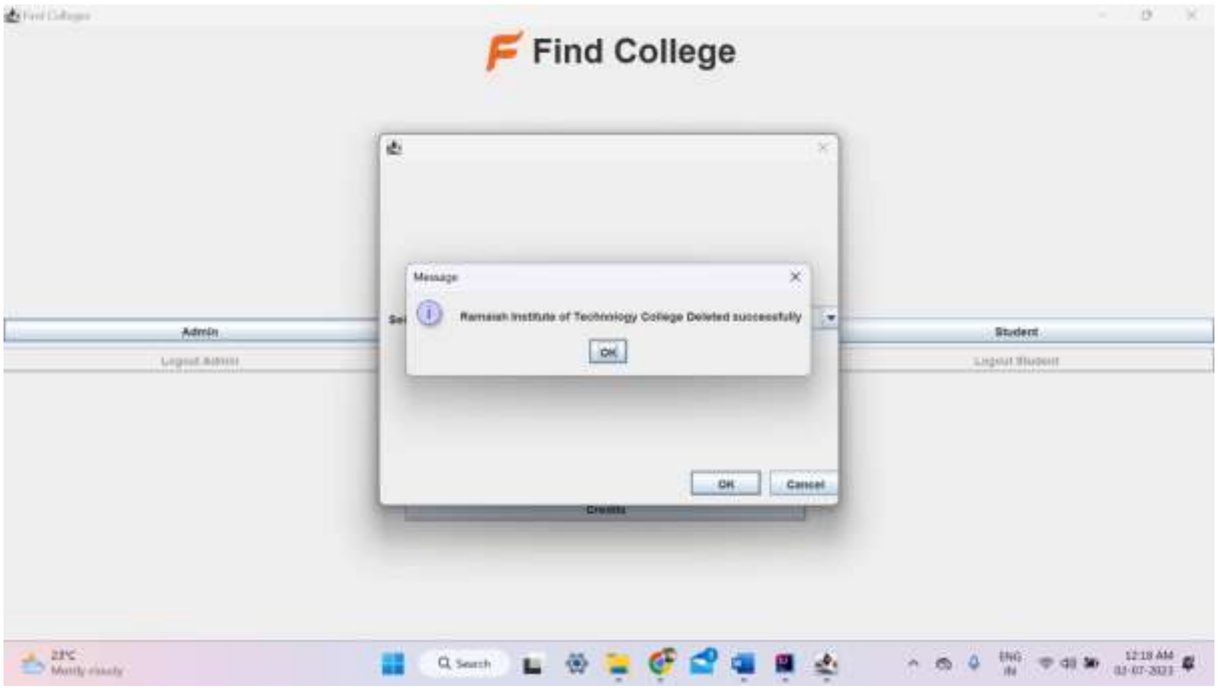


Select the College to delete: Ramaiah Institute of Technology

- Ramaiah Institute of Technology
- BMS Institute of Technology and Management
- Sahyadri College of Engineering & Management
- Bangalore Institute of Technology
- A J Institute of Engineering & Technology

Figure 5.7: Admin – Delete Colleges

The snapshot showcases the "Admin - Delete Colleges" functionality, empowering administrators to remove colleges from the system efficiently, ensuring effective management and maintenance of the college database in the Find College Project.



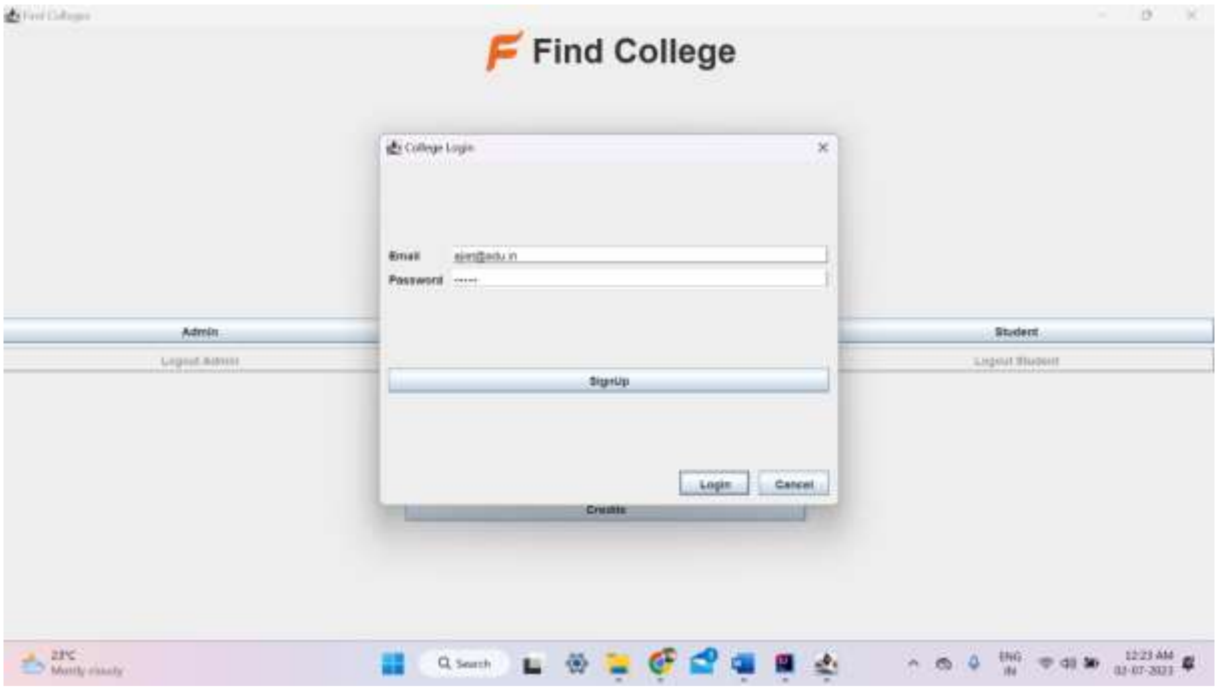
Message

Ramaiah Institute of Technology College Deleted successfully

OK

Figure 5.8: Admin – College Deleted Successfully Message

The snapshot captures the message "College Deleted Successfully," indicating the successful removal of a college from the system by the admin, ensuring efficient maintenance and reflecting the dynamic nature of the Find College Project.



College Login

Email ajiet@edu.in

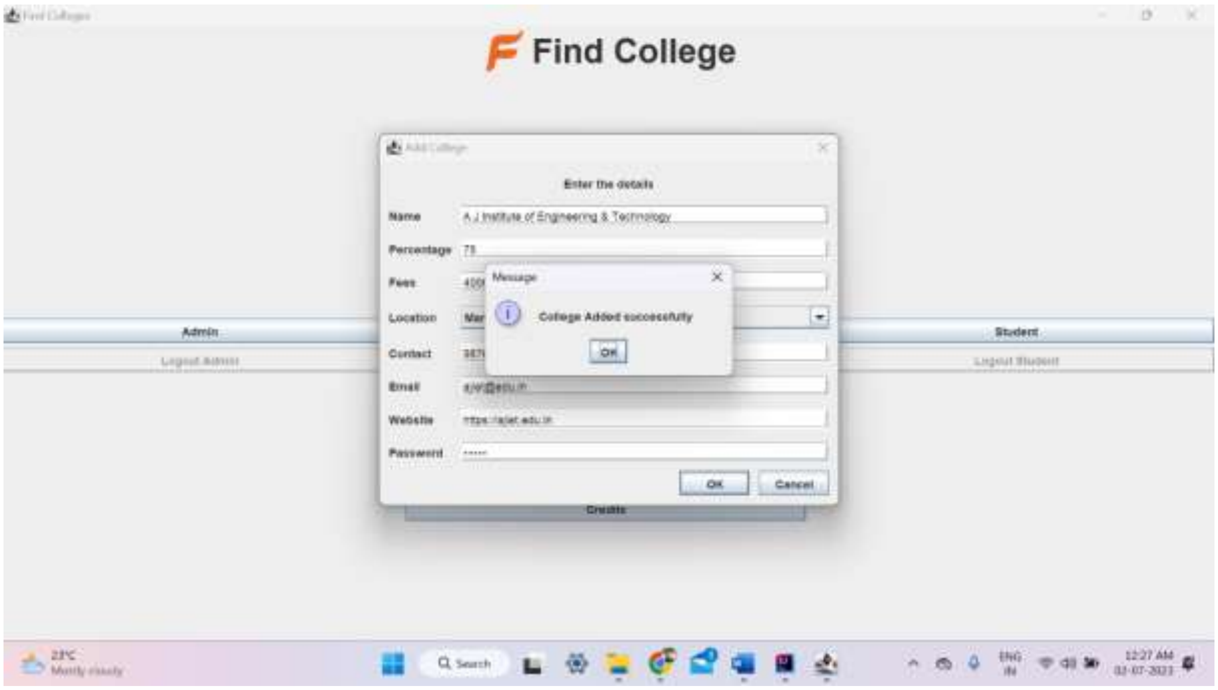
Password ------

SignUp

Login Cancel

Figure 5.9: College SignUp & Login Page

The snapshot showcases the "College SignUp & Login Page," facilitating colleges to register and log in efficiently, ensuring secure access to the Find College Project's functionalities.



Enter the details

Name	A J Institute of Engineering & Technology
Percentage	75
Fees	400
Location	Man
Contact	987
Email	ajiet@edu.in
Website	https://ajiet.edu.in
Password	******

Message

College Added successfully

Figure 5.10: SignUp and College Added Successful Message

The snapshot displays the "SignUp and College Added Successful" message, providing confirmation of the successful registration process for colleges in the system.

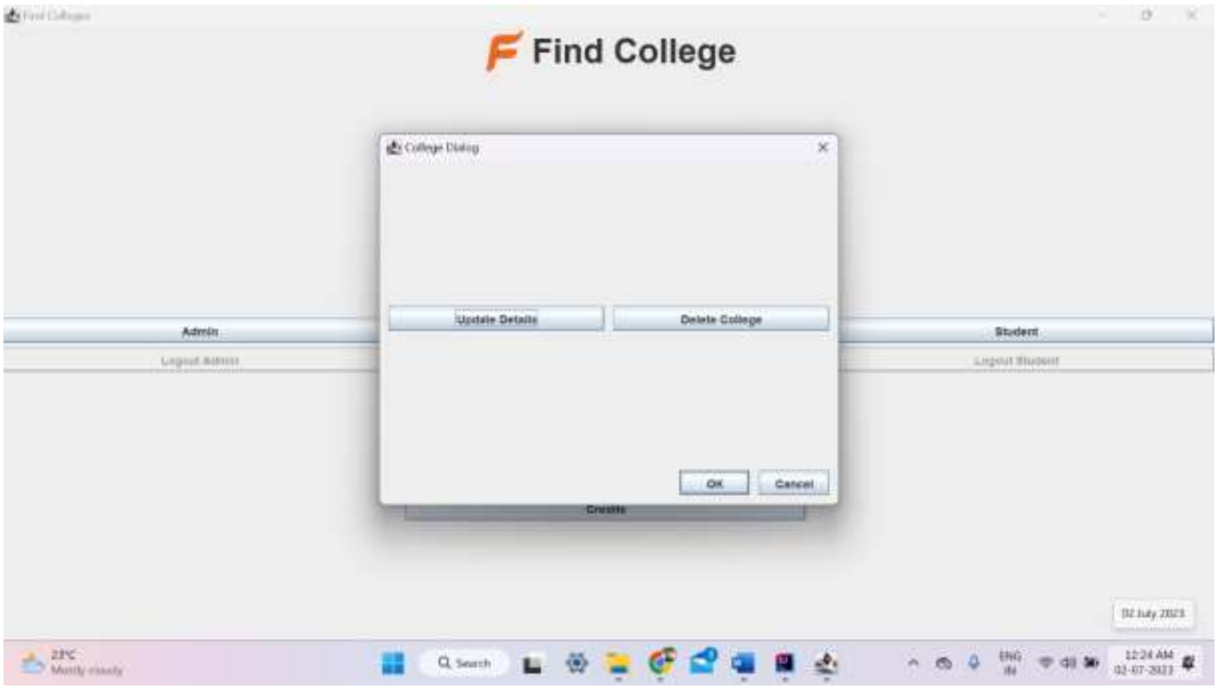
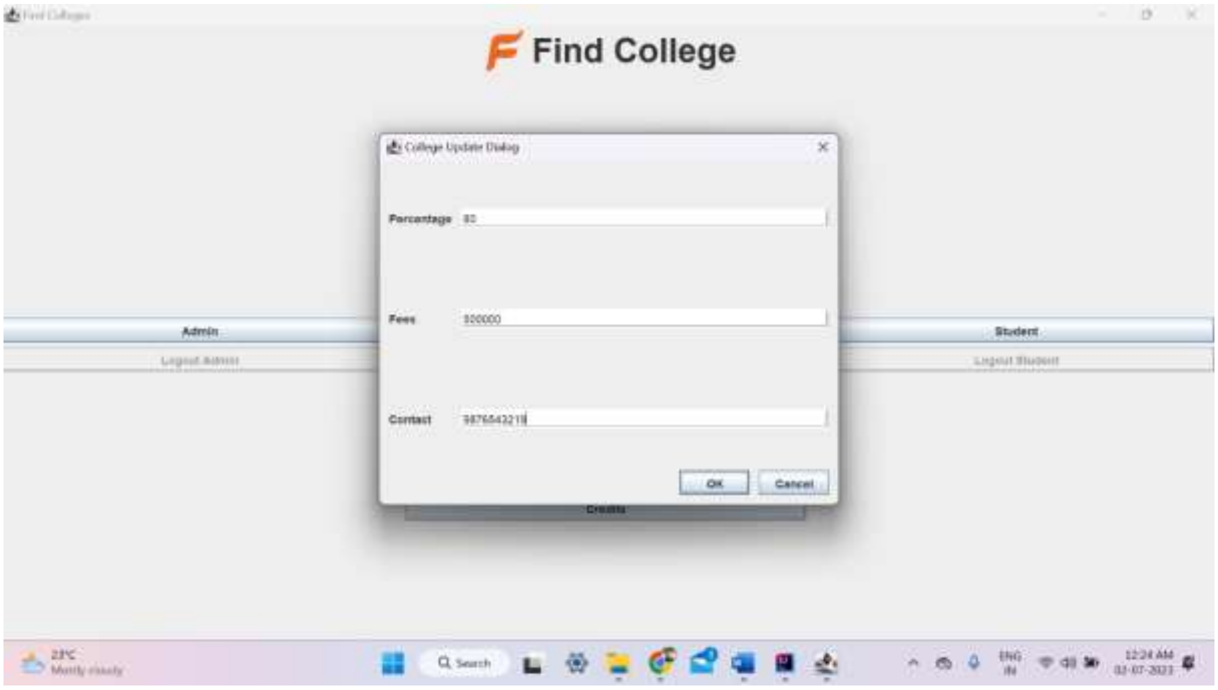


Figure 5.11: College – Update Details & Delete College Button

The snapshot showcases the "Update Details & Delete College" button, offering convenient options for modifying college information and removing colleges, ensuring flexibility and efficient management within the system.



College Update Dialog

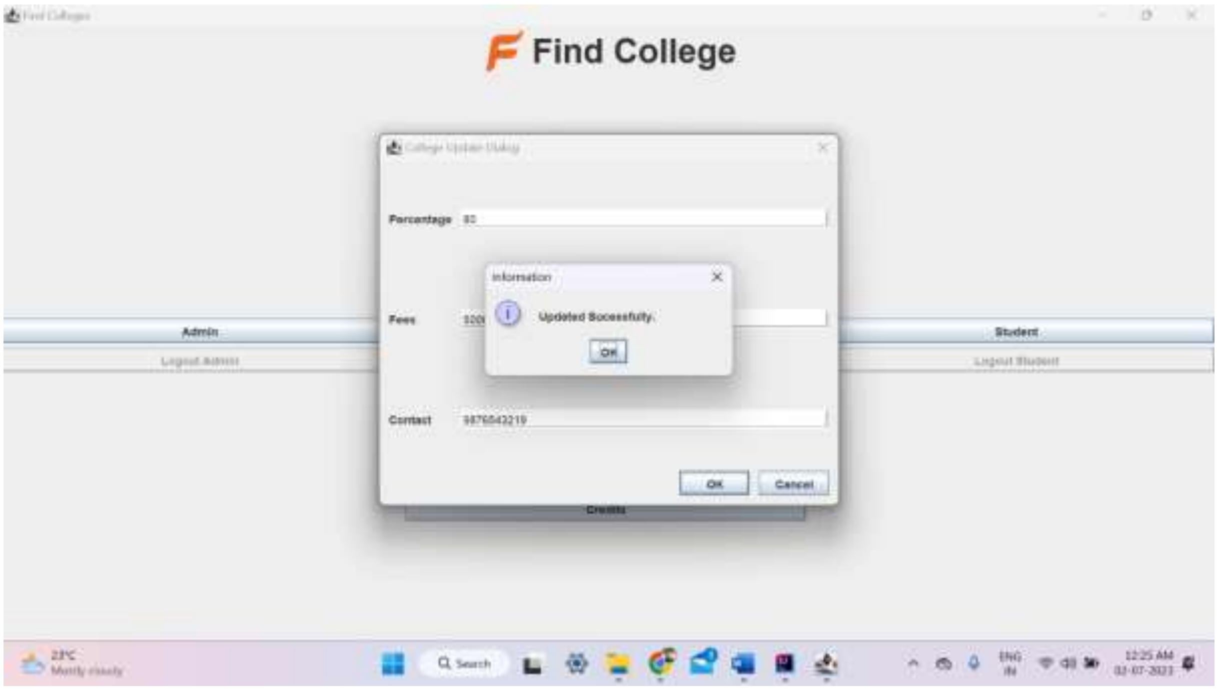
Percentage 80

Fees 300000

Contact 9876543219

Figure 5.12: College Update Dialog

The snapshot presents the "College Information Updated Successfully" message, indicating that the changes made to the college information have been saved successfully, providing reassurance and reflecting the smooth update process in the system.



College Update Dialog

Percentage 80

Fees 300

Contact 9876543219

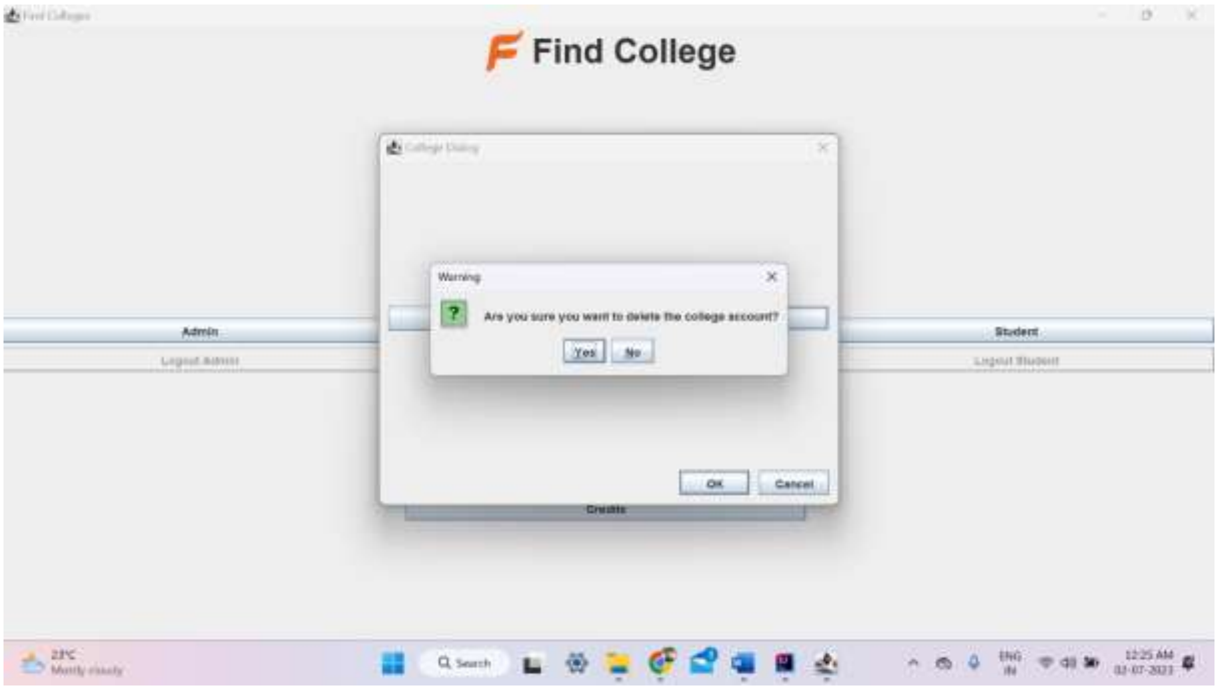
Information

Updated Successfully.

OK

Figure 5.13: College Information Updated Successfully Message

The snapshot showcases the "Information Updated Successfully" message, confirming the successful update of college information and ensuring accurate data management in the system.

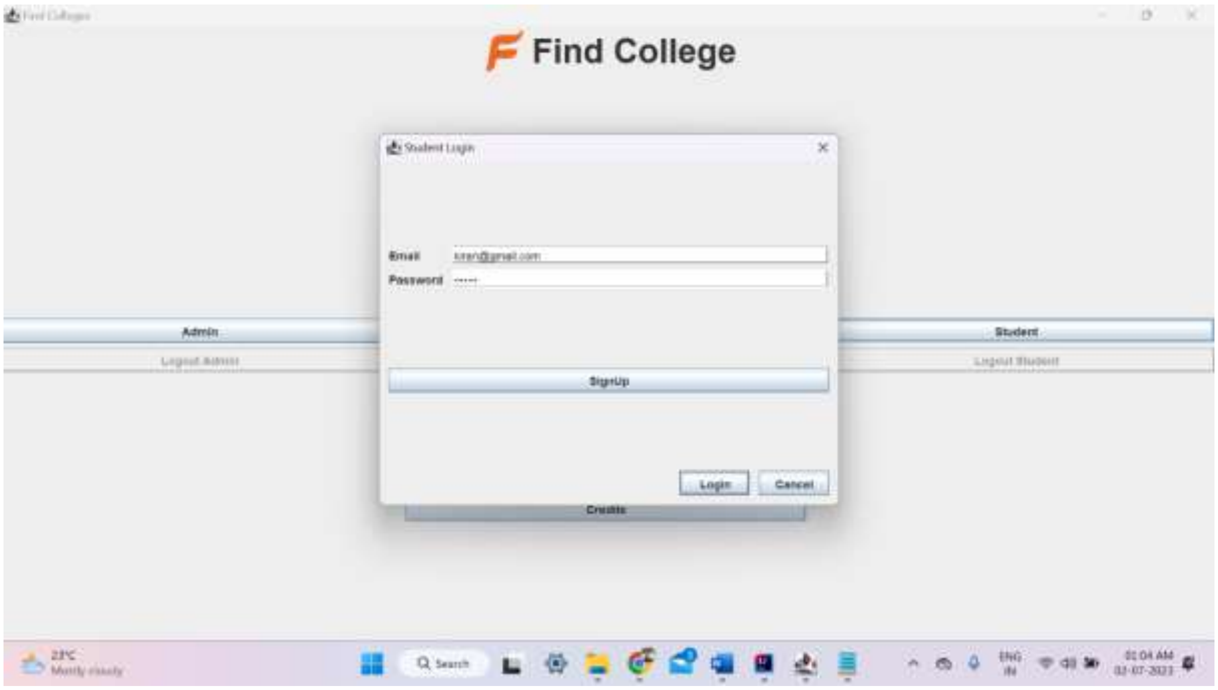


Warning

Are you sure you want to delete the college account?

Figure 5.14: Warning Message To Delete College Account

The snapshot displays a warning message, urging users to exercise caution before deleting their account, emphasizing the gravity of the action.



Email kiran@gmail.com
Password ······

Figure 5.15: Student Login Page

The snapshot showcases the "Student Login Page," offering a convenient and secure interface for students to log in, facilitating access to their personalized features and resources.

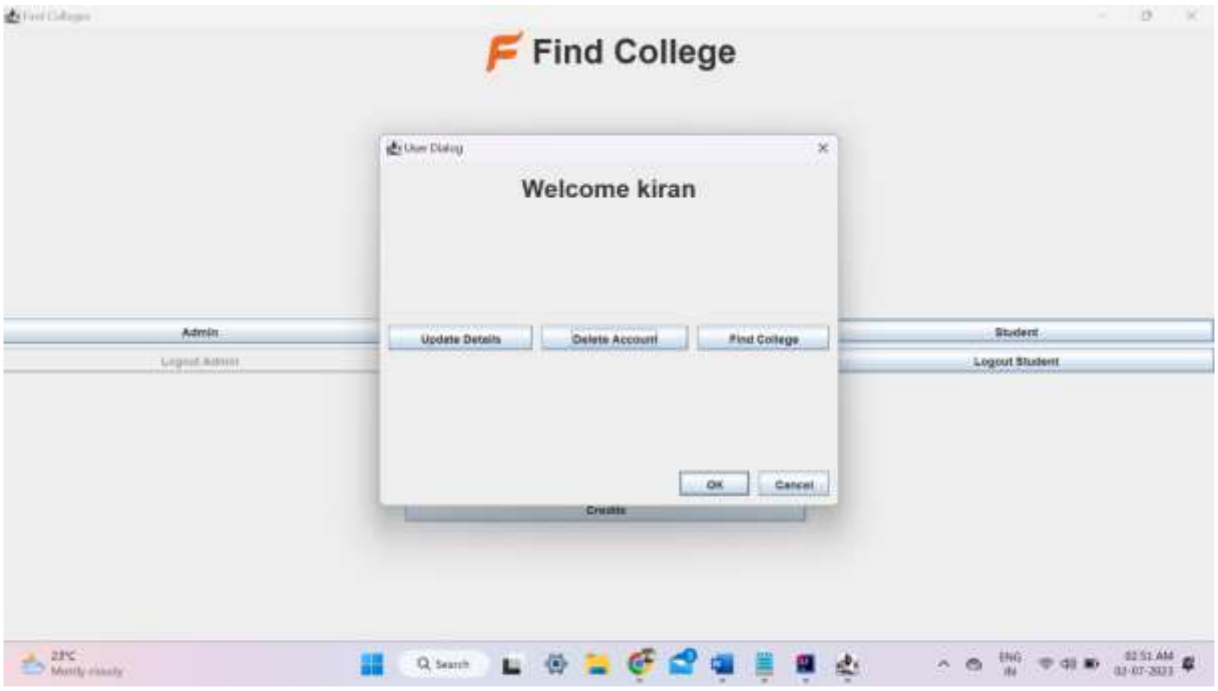
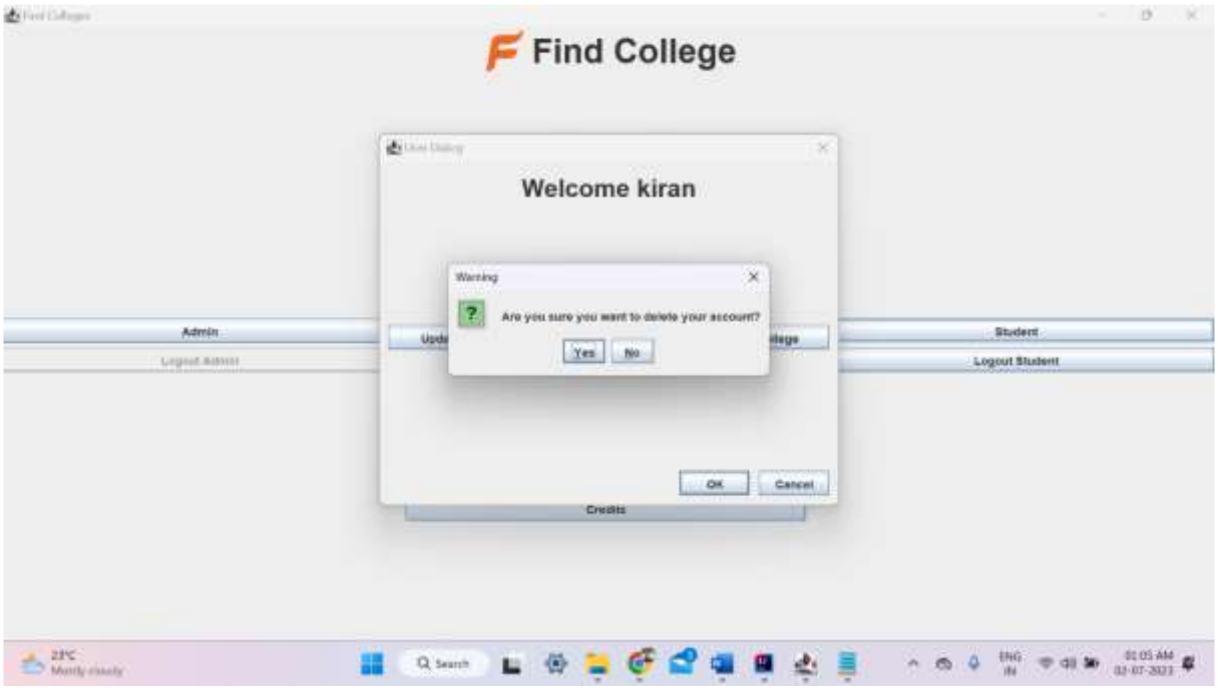


Figure 5.16: Student - Update Details, Delete Account & Find College Button

The snapshot highlights the "Update Details, Delete Account & Find College" buttons, providing students with options to modify their information, delete their account, and search for colleges, enhancing their control and functionality within the system.



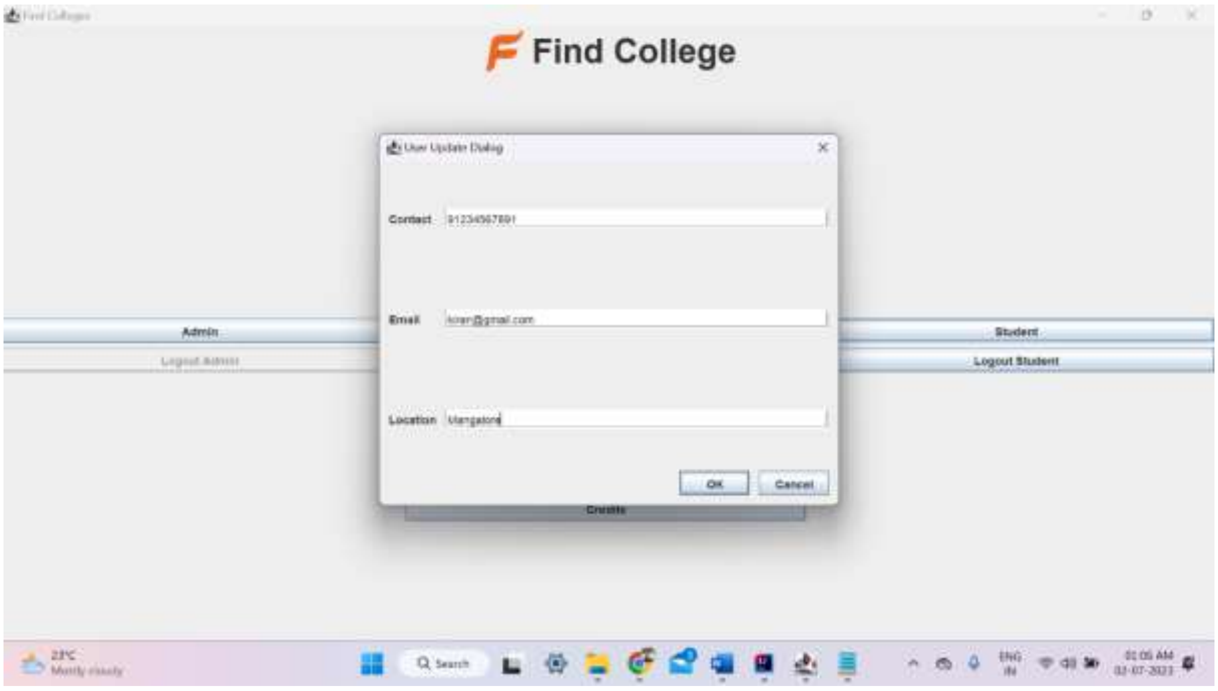
Warning

Are you sure you want to delete your account?

Yes No

Figure 5.17: Student – Delete Account With Warning Message

The snapshot features the "Delete Account" button with a warning message, emphasizing the irreversible nature of the action.



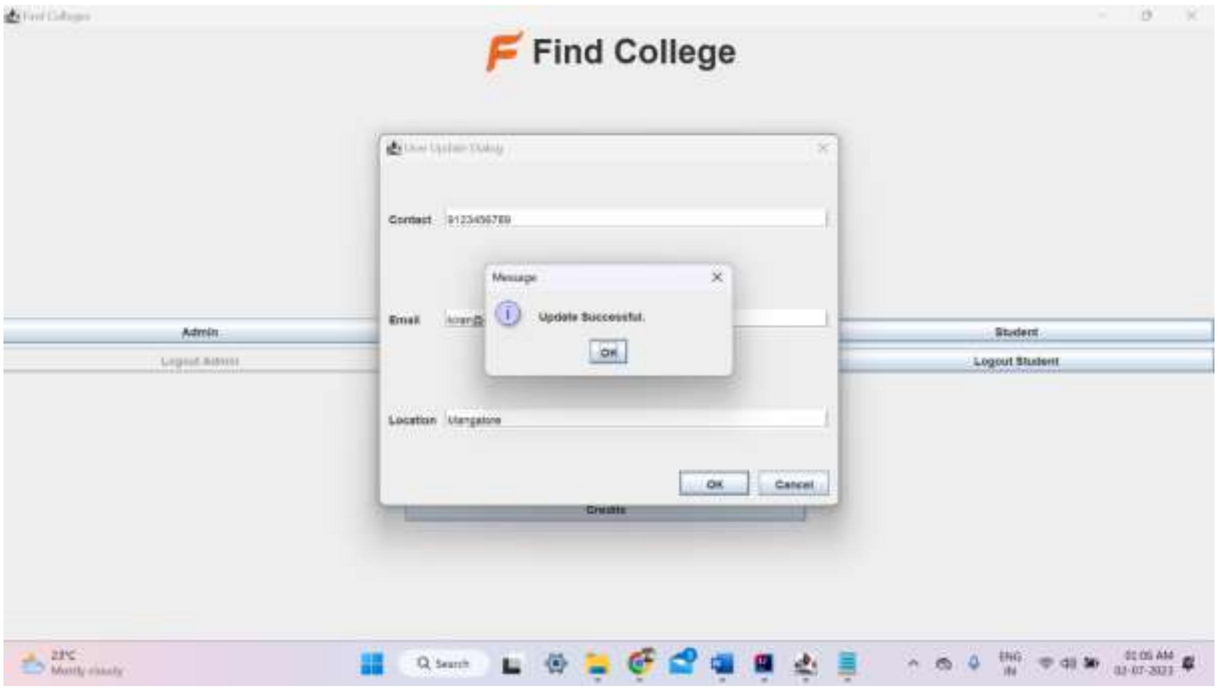
Contact 91234567891

Email kiran@gmail.com

Location Mangalore

Figure 5.18: Student Update Details

The snapshot showcases the "Student Update Details" feature, providing students with the ability to modify and update their personal information in the system.



Contact 9123456789

Email kiran@

Location Mangalore

Update Successful.

Figure 5.19: Student – Update Successful Message

The snapshot captures the "Update Successful" message, indicating that the requested changes to the student's details have been successfully saved, ensuring accurate and up-to-date information in the system.

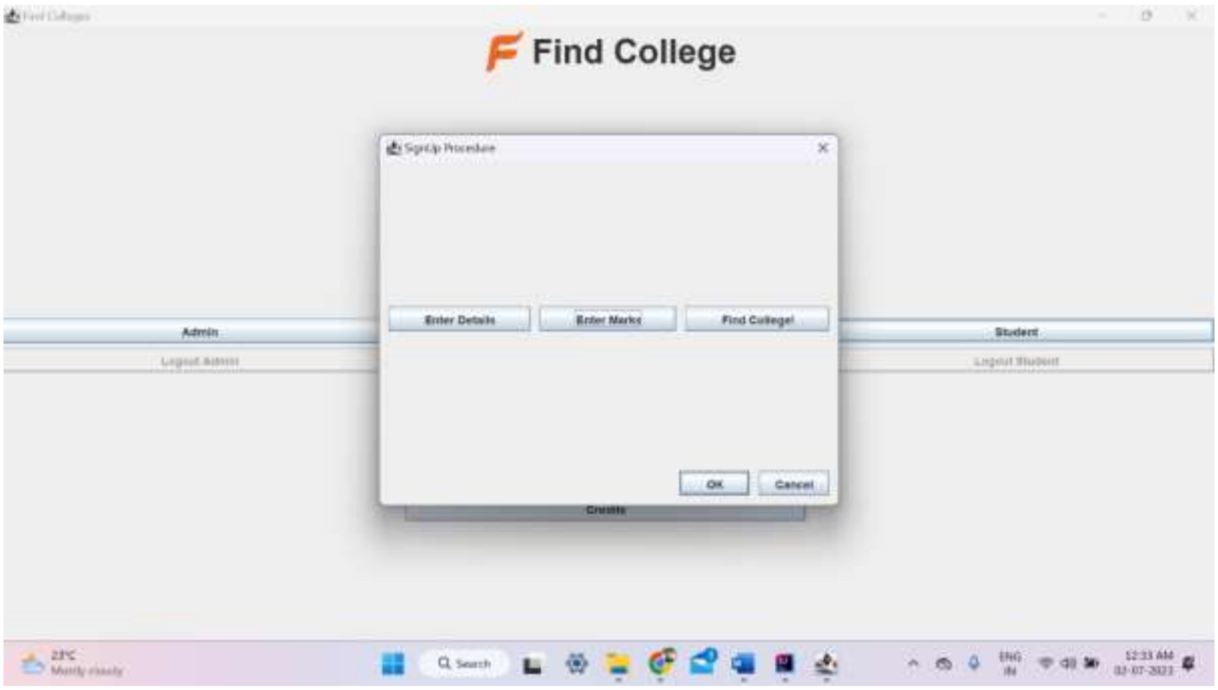
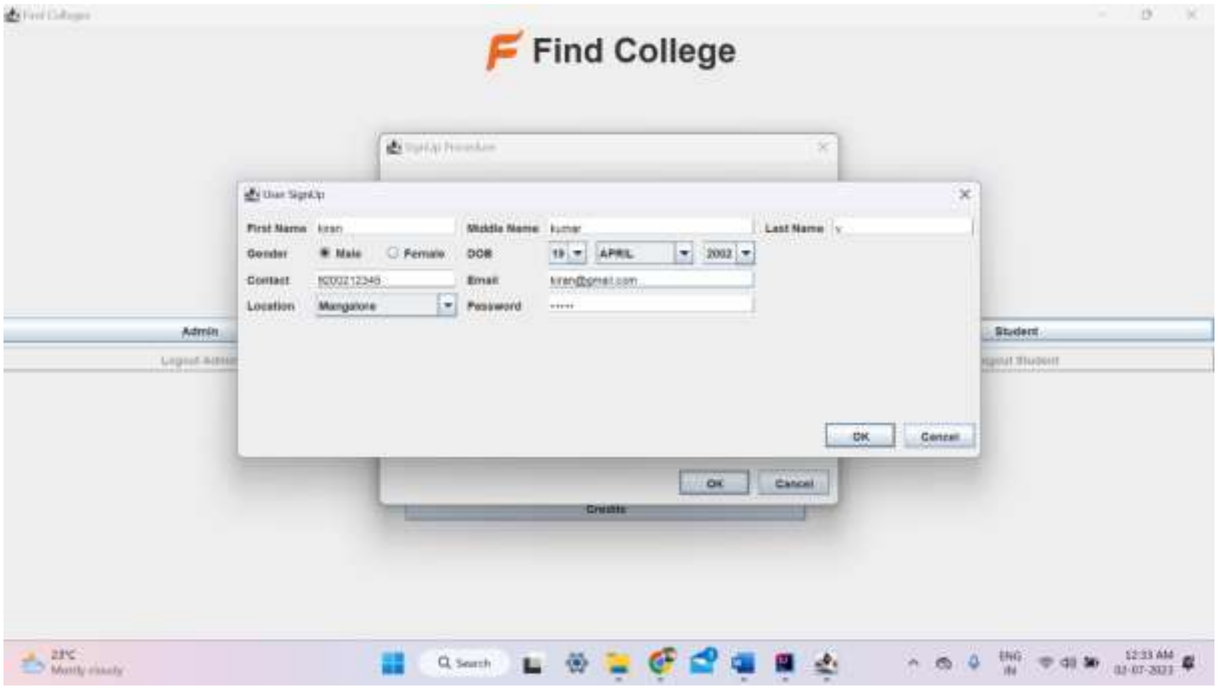


Figure 5.20: SignUp Procedure

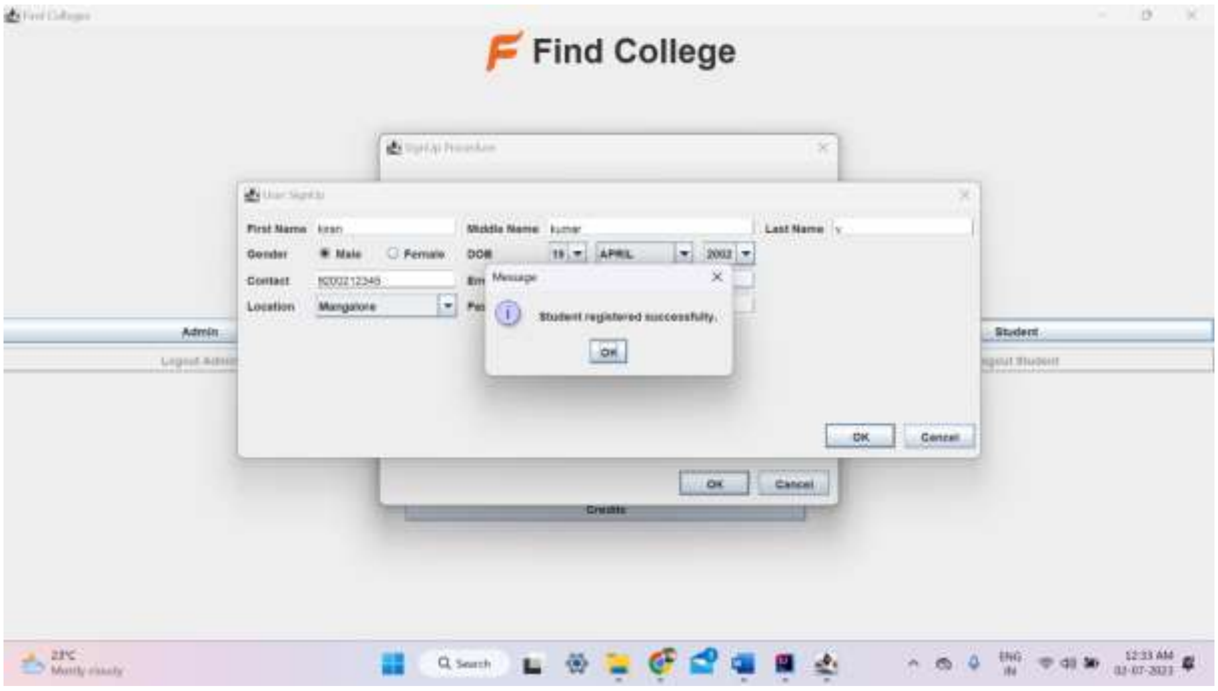
The snapshot showcases the "SignUp Procedure," providing a visual representation of the steps and input fields required for students to create their accounts and register successfully in the system.



First Name	kiran	Middle Name	kumar	Last Name	v
Gender	[✓ Male] Female	DOB	19 APRIL 2002		
Contact	9200212345	Email	kiran@gmail.com		
Location	Mangalore	Password	*****		

Figure 5.21: User SignUp Details

The snapshot features the "User SignUp Details" section, displaying the input fields for users to provide necessary information during the signup process, enabling account creation.



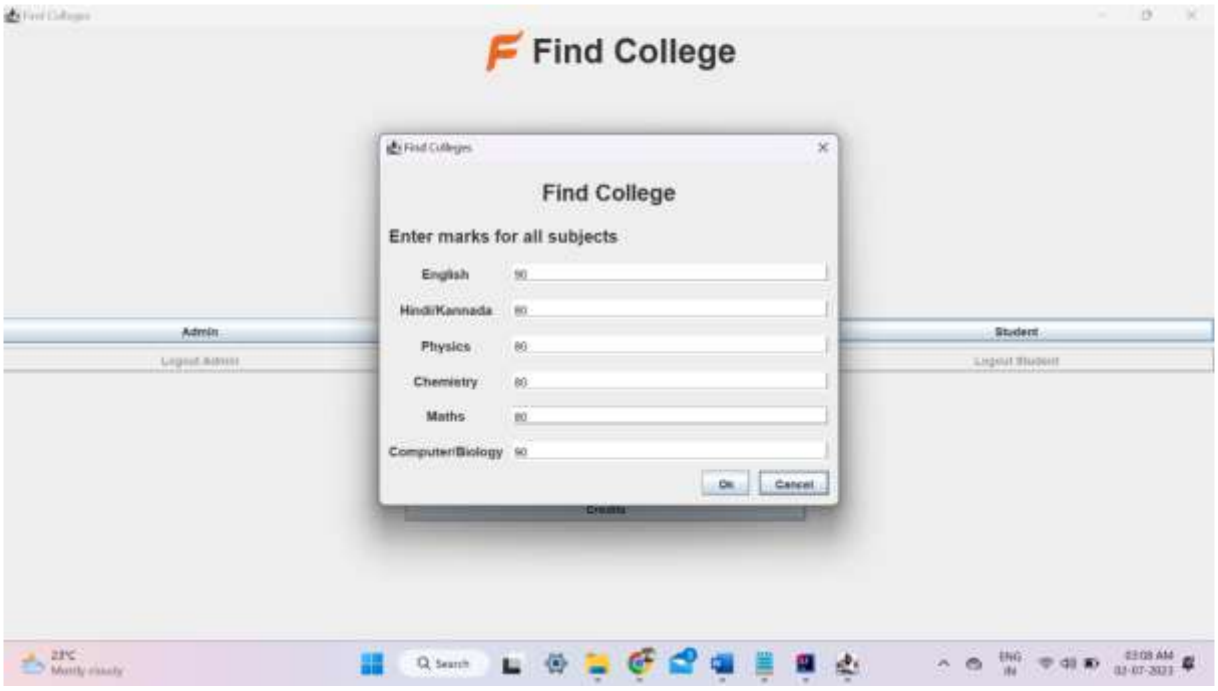
First Name	kiran	Middle Name	kumar	Last Name	v
Gender	[✓ Male] Female	DOB	19 APRIL 2002		
Contact	9200212345				
Location	Mangalore				

Message

Student registered successfully.

Figure 5.22: Student Registered Successfully Message

The snapshot captures the "Student Registered Successfully" message, indicating that the student has successfully completed the registration process, ensuring their successful entry into the system.



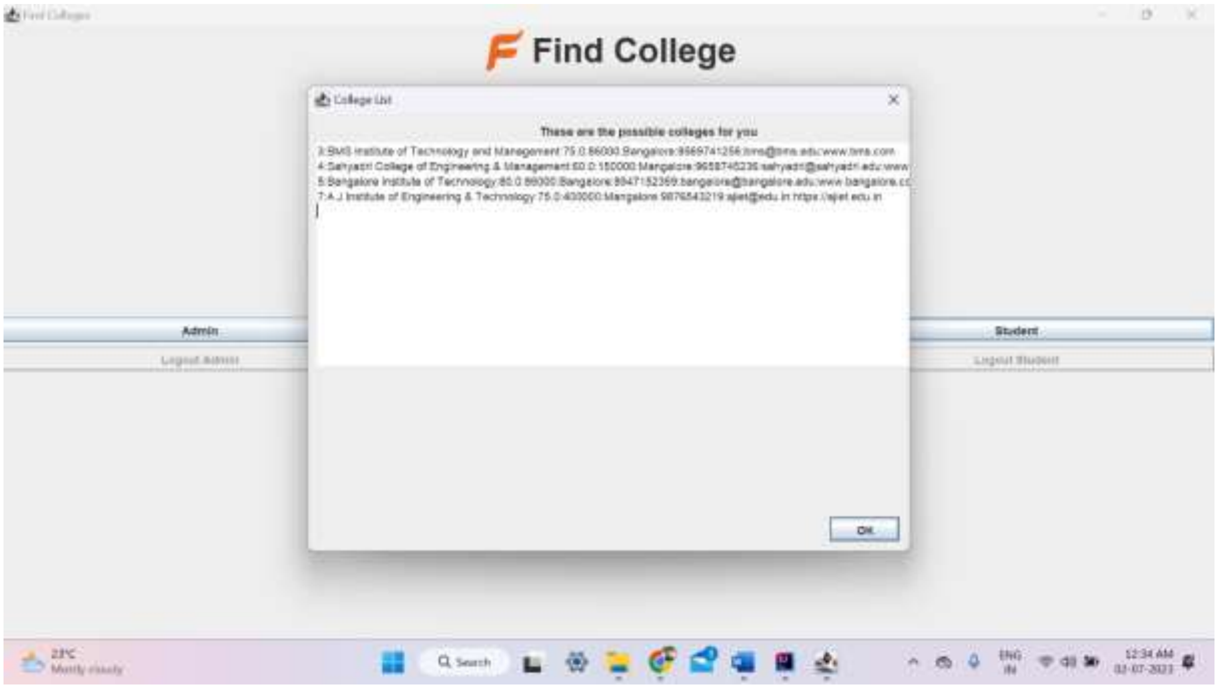
Find College

Enter marks for all subjects

English	90
Hindi/Kannada	80
Physics	80
Chemistry	80
Maths	80
Computer/Biology	90

Figure 5.23: Marks Details

The snapshot presents the "Enter Marks for All Subjects" section, providing input fields or an interface for students to input their marks or grades for each subject, enabling the system to check their eligibility for college admission based on the set criteria.

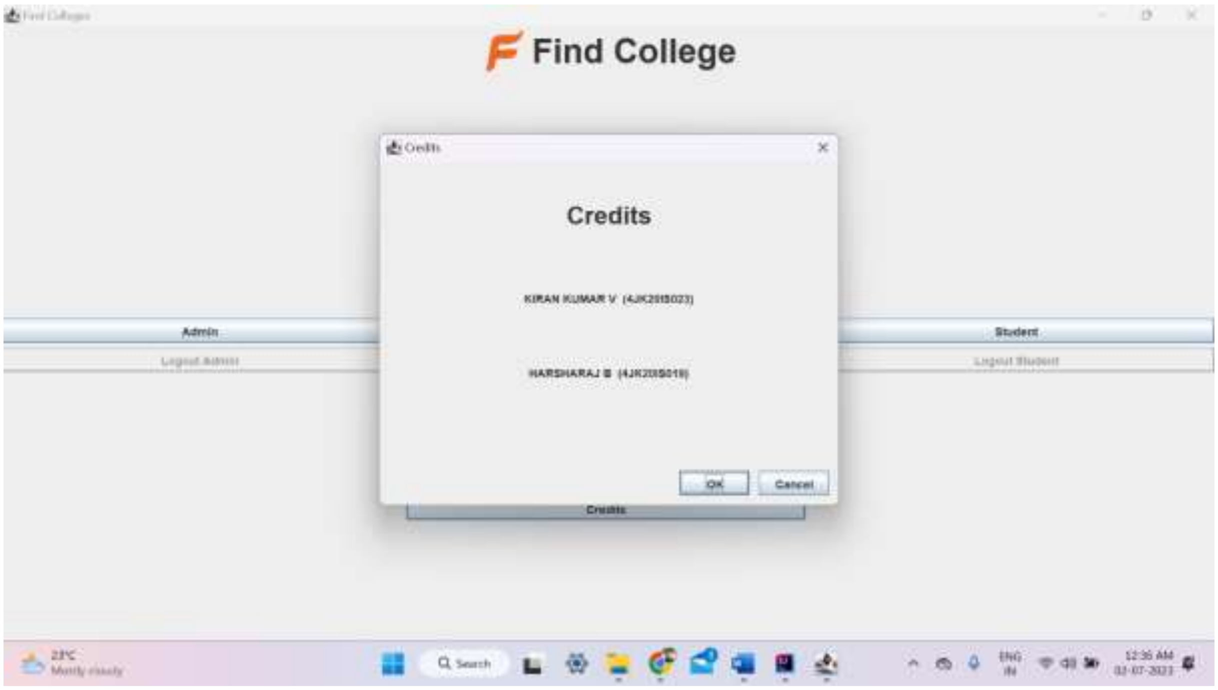


These are the possible colleges for you

```
3:BMS Institute of Technology and Management:75.0:86000:Bangalore:9569741256:bms@bms.edu:www.bms.com
4:Sahyadri College of Engineering & Management:60.0:150000:Mangalore:9658745236:sahyadri@sahyadri.edu:www
5:Bangalore Institute of Technology:80.0:86000:Bangalore:8947152369:bangalore@bangalore.edu:www.bangalore.cc
7:A.J Institute of Engineering & Technology:75.0:400000:Mangalore:9876543219:ajiet@edu.in:https://ajiet.edu.in
```

Figure 5.24: College List

The snapshot displays the "College List" section, showcasing a comprehensive list of colleges available within the system, providing students with easy access to explore and select their preferred choices for further education.



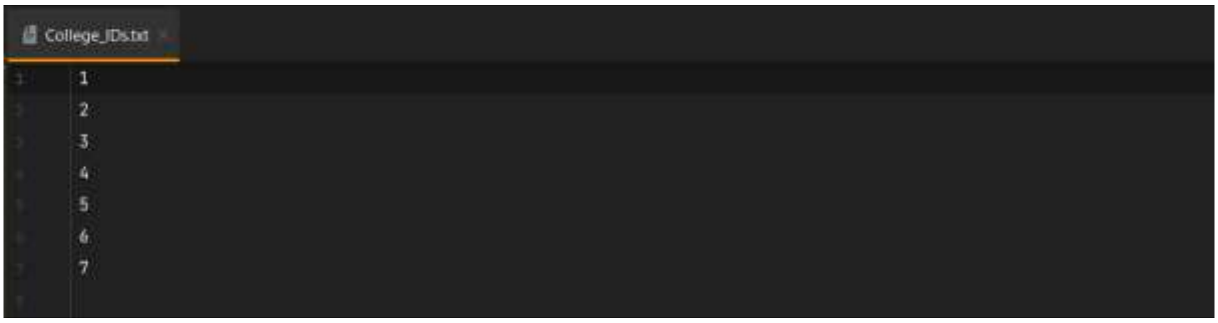
Credits

KIRAN KUMAR V (4JK20IS023)

HARSHARAJ B (4JK20IS019)

Figure 5.25: Credits

The snapshot showcases the "Credits" section, prominently displaying the name of the project creator(s), acknowledging their contribution and expertise in the development of the project.

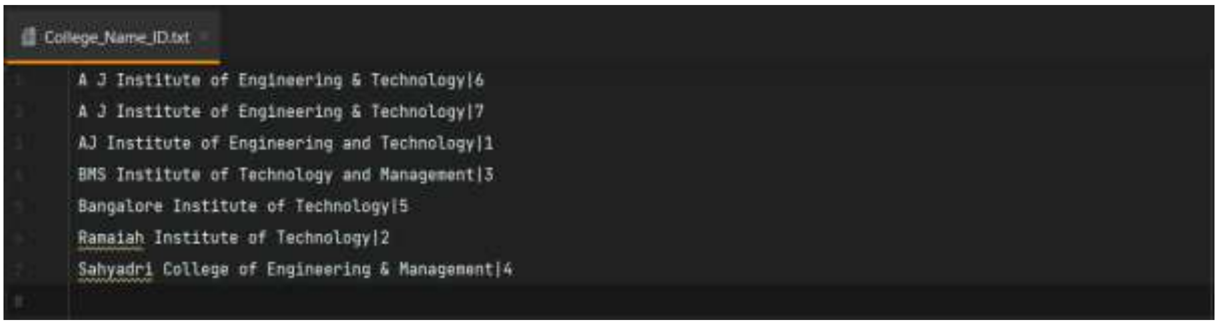


College_IDs.txt

```
1
2
3
4
5
6
7
```

Figure 5.26: College_ID.txt

College_ID.txt a text file that stores the unique identification of a college.

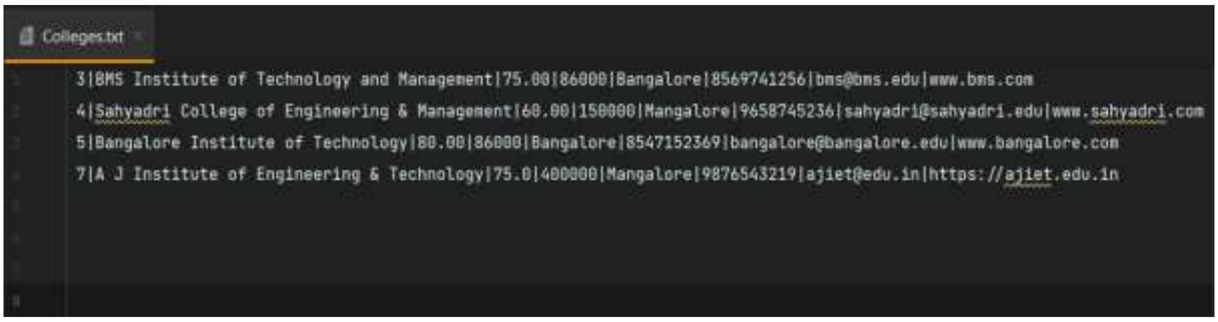


College_Name_ID.txt

```
A J Institute of Engineering & Technology|6
A J Institute of Engineering & Technology|7
AJ Institute of Engineering and Technology|1
BMS Institute of Technology and Management|3
Bangalore Institute of Technology|5
Ramaiah Institute of Technology|2
Sahyadri College of Engineering & Management|4
```

Figure 5.27: College_Name_ID.txt

College_Name_ID.txt a text file stores the college name and associated ID for easy reference.

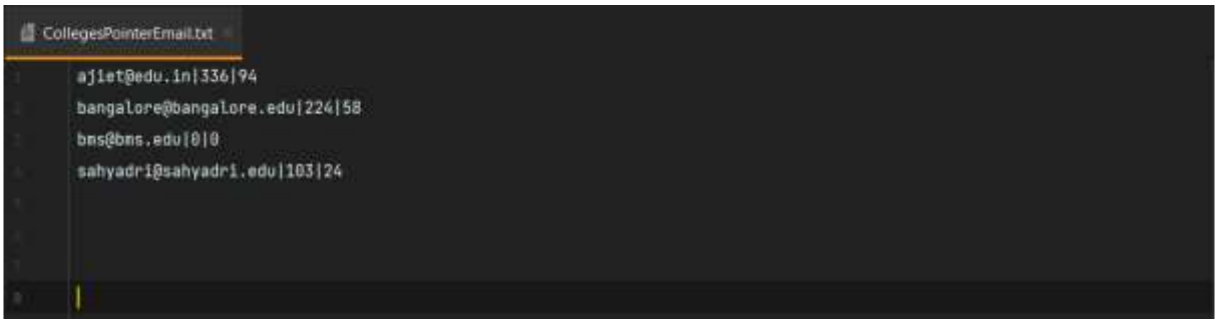


Colleges.txt

```
3|BMS Institute of Technology and Management|75.00|86000|Bangalore|8569741256|bms@bms.edu|www.bms.com
4|Sahyadri College of Engineering & Management|60.00|150000|Mangalore|9658745236|sahyadri@sahyadri.edu|www.sahyadri.com
5|Bangalore Institute of Technology|80.00|86000|Bangalore|8547152369|bangalore@bangalore.edu|www.bangalore.com
7|A J Institute of Engineering & Technology|75.0|400000|Mangalore|9876543219|ajiet@edu.in|https://ajiet.edu.in
```

Figure 5.28: Colleges.txt

Colleges.txt a comprehensive file containing college details including ID, name, email, website, password, contact number, location, fees, and percentage.

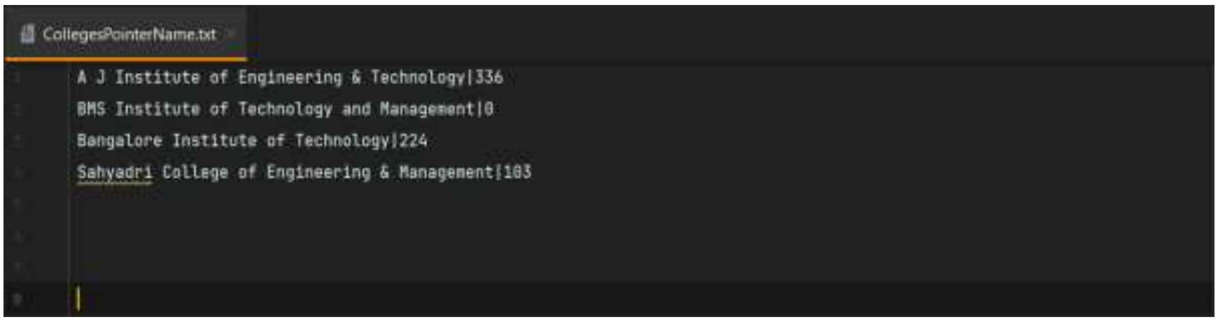


CollegesPointerEmail.txt

```
ajiet@edu.in|336|94
bangalore@bangalore.edu|224|58
bms@bms.edu|0|0
sahyadri@sahyadri.edu|103|24
```

Figure 5.29: CollegePointerEmail.txt

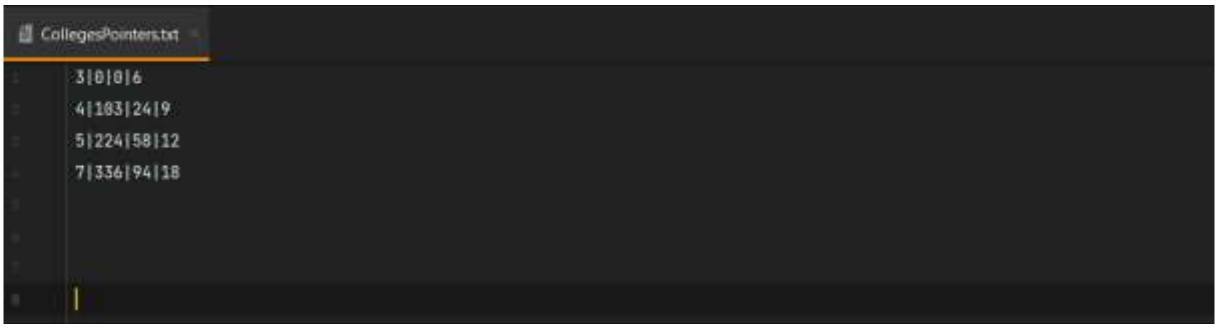
CollegePointerEmail.txt a file storing college email addresses and associated academic pointers.



```
A J Institute of Engineering & Technology|336
BMS Institute of Technology and Management|0
Bangalore Institute of Technology|224
Sahyadri College of Engineering & Management|103
```

Figure 5.30: CollegePointerName.txt

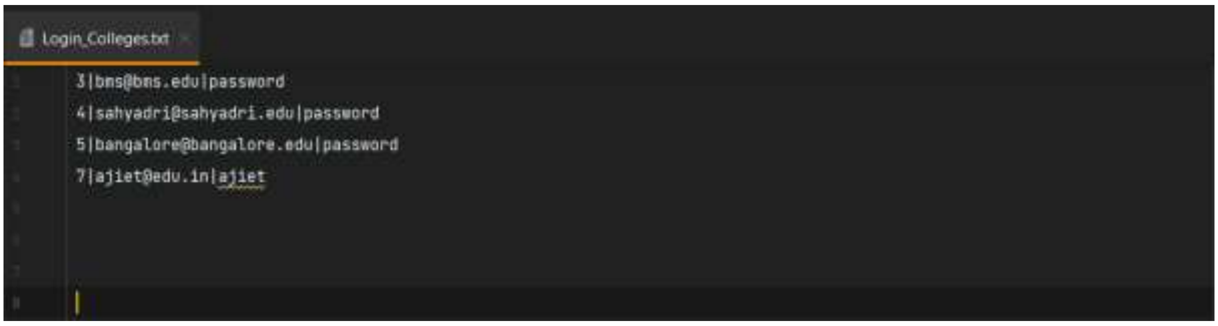
CollegePointerName.txt a file that stores college names and corresponding pointer data, enabling efficient referencing and retrieval of information within the college system.



```
3|0|0|6
4|103|24|9
5|224|58|12
7|336|94|18
```

Figure 5.31: CollegesPointers.txt

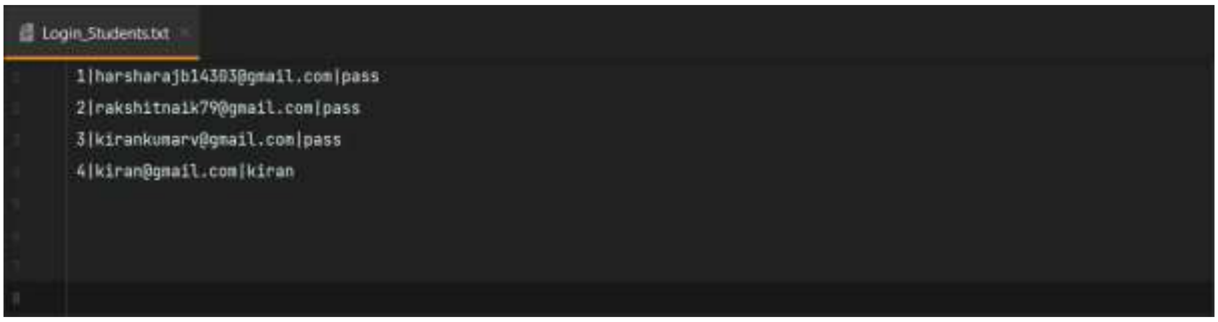
CollegesPointers.txt a file that stores college pointers, enabling efficient access and retrieval of information within the college system.



```
3|bms@bms.edu|password
4|sahyadri@sahyadri.edu|password
5|bangalore@bangalore.edu|password
7|ajiet@edu.in|ajiet
```

Figure 5.32: Login_Colleges.txt

"Login_Colleges.txt: A text file containing college IDs, email addresses, and passwords for accessing the college system's login credentials."

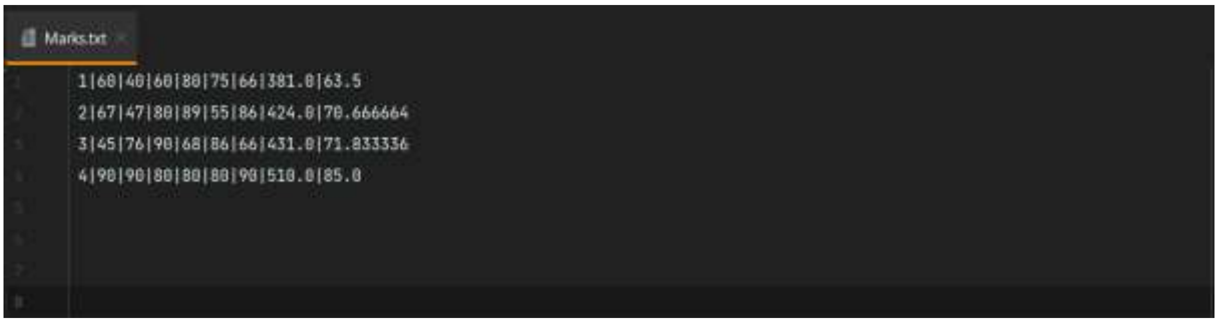


Login_Students.txt

```
1|harsharajb14303@gmail.com|pass
2|rakshitnaik79@gmail.com|pass
3|kirankumarv@gmail.com|pass
4|kiran@gmail.com|kiran
```

Figure 5.33: Login_Students.txt

Login_Students.txt a file containing student ID, email, and password data for authentication and access control within the college system.

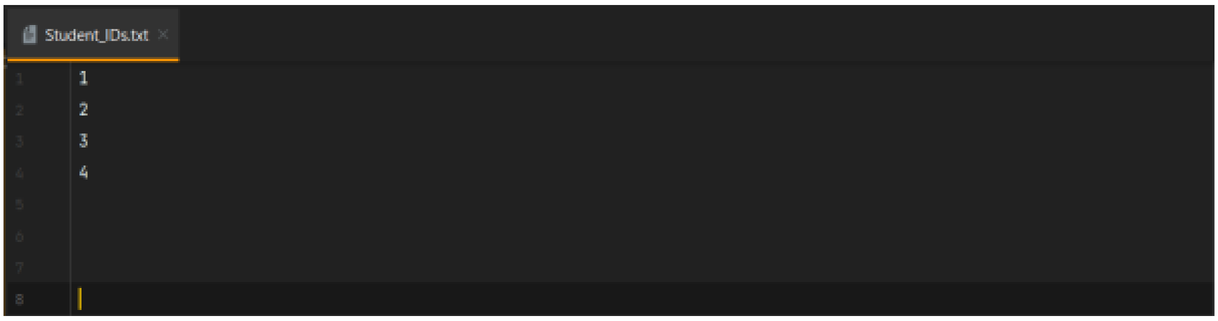


Marks.txt

```
1|60|40|60|80|75|66|381.0|63.5
2|67|47|80|89|55|86|424.0|70.666664
3|45|76|90|68|86|66|431.0|71.833336
4|90|90|80|80|80|90|510.0|85.0
```

Figure 5.34: Marks.txt

Result: Marks.txt - a file storing student IDs, subject marks, and percentages attained in the college examination, providing comprehensive academic performance data for reference and analysis.

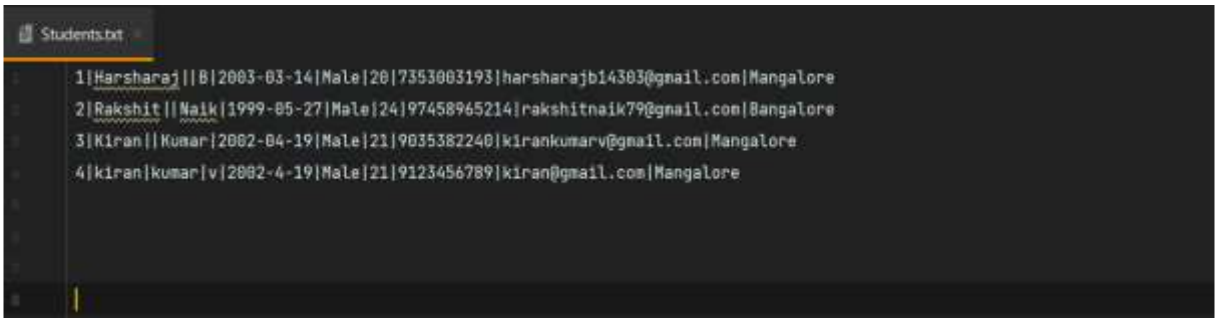


Student_IDs.txt

```
1
2
3
4
```

Figure 5.35: Student_IDs.txt

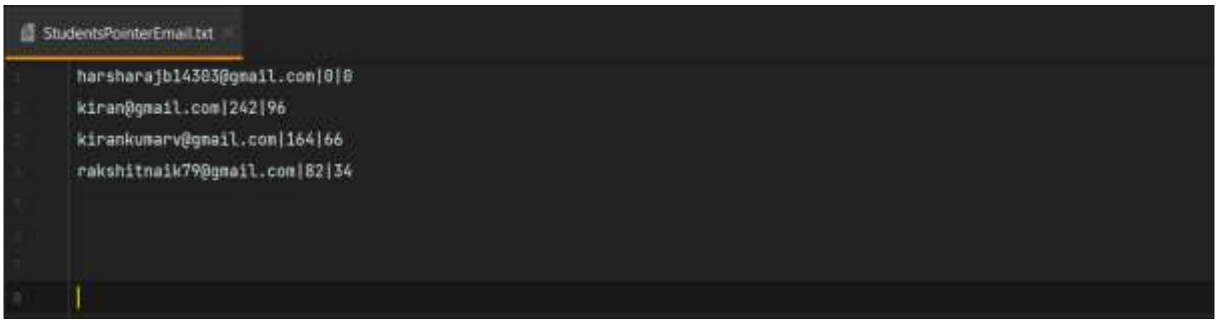
Student_IDs.txt is a file within the college system that holds all the student IDs, serving as a reference and facilitating easy access for administrative purposes. It plays a crucial role in maintaining student records and ensuring efficient management of academic information.



```
1|Harsharaj||B|2003-03-14|Male|20|7353003193|harsharajb14303@gmail.com|Mangalore
2|Rakshit||Naik|1999-05-27|Male|24|97458965214|rakshitnaik79@gmail.com|Bangalore
3|Kiran||Kumar|2002-04-19|Male|21|9035382240|kirankumarv@gmail.com|Mangalore
4|kiran|kumar|v|2002-4-19|Male|21|9123456789|kiran@gmail.com|Mangalore
```

Figure 5.36: Students.txt

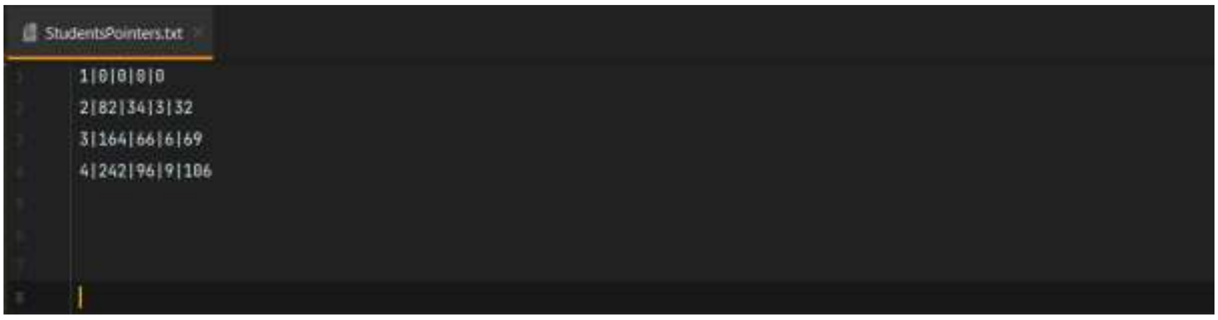
Result Students.txt: A comprehensive file containing student information such as ID, name, gender, date of birth, age, contact number, email ID, and location. This file serves as a repository for managing and accessing crucial student data within the system.



```
harsharajb14303@gmail.com|0|0
kiran@gmail.com|242|96
kirankumarv@gmail.com|164|66
rakshitnaik79@gmail.com|82|34
```

Figure 5.37: StudentsPointerEmail.txt

StudentsPointerEmail.txt a file with student email addresses and pointers, enabling efficient data management within the college system for improved communication and organization.



```
1|0|0|0|0
2|82|34|3|32
3|164|66|6|69
4|242|96|9|106
```

Figure 5.38: StudentsPointers.txt

StudentsPointers.txt: A file containing student pointers for efficient access and retrieval within the college system, facilitating seamless management and utilization of student information. The file ensures optimized performance and streamlined processes for easy navigation and referencing of student records. With this centralized storage, administrators can quickly retrieve specific student details based on unique identifiers, enhancing the overall efficiency of the college system.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In conclusion, the "Find College" project developed in Java Swing with file handling has successfully provided a comprehensive platform for college search and exploration. The project encompassed three user roles: Admin, College, and Student, each with specific functionalities and access privileges. Through the implementation of various features, such as user authentication, college registration, student profile creation, and college search, the project has achieved its objective of facilitating the college selection process.

The project utilized text files as a storage mechanism, avoiding the need for a traditional database system. This approach allowed for easy data management and retrieval, while also providing flexibility in terms of file organization and maintenance. The user-friendly interface, combined with efficient file handling techniques, ensured a smooth user experience.

The Admin module enabled administrators to manage colleges by adding new entries, viewing existing colleges, and deleting entries as needed. The College module allowed registered colleges to update their details and delete their profiles. The Student module allowed students to create profiles, enter their academic marks, and search for colleges based on their preferences.

6.2 Future scope

While the current implementation of the "Find College" project has provided valuable functionalities, there are several areas that can be further improved and expanded in the future:

6.2.1 Enhanced User Experience

Implementing more intuitive and visually appealing interfaces, as well as incorporating user feedback mechanisms, would enhance the overall user experience. Improving the responsiveness and usability of the application on different devices, such as tablets and mobile phones, would also be beneficial.

6.2.2 Advanced Search and Recommendation System

Developing an advanced search algorithm that takes into account additional factors like student preferences, academic performance, and extracurricular activities would provide more accurate and personalized college recommendations to students. Incorporating machine learning techniques could help in creating a recommendation engine based on historical data and user behavior.

6.2.3 Data Encryption and Security Measures

Implementing robust encryption mechanisms to protect sensitive user data, such as passwords and personal information, would enhance the security of the application. Incorporating secure authentication protocols, such as two-factor authentication, would provide an additional layer of security.

6.2.4 User Reviews and Ratings

Implement a feature that allows students and parents to leave reviews and ratings for colleges they have attended or researched. This feedback can help future students make informed decisions and provide valuable insights for college improvements.

6.2.5 Advanced Sorting and Filtering

Enhance the search functionality by adding more advanced sorting and filtering options. This could include sorting colleges by rankings, program availability, campus facilities, or other relevant criteria to help users find the most suitable options.

6.2.6 College Comparison Tool

Develop a tool that enables users to compare multiple colleges side by side. This tool could display key information such as fees, courses offered, campus amenities, and student satisfaction ratings, allowing users to make detailed comparisons before making a decision.

6.2.7 Alumni Network Integration

Integrate an alumni network feature that connects current students with alumni from different colleges. This would facilitate mentorship opportunities, career guidance, and networking possibilities, creating a stronger community within the platform.

REFERENCE

[1] Oracle Corporation. "Java™ Platform, Standard Edition 7 API Specification." Oracle, 2013. [Online]. Available: https://docs.oracle.com/javase/7/docs/api.

[2]"Java Swing Tutorial." Oracle, [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/.

[3] File Handling in Java: https://www.geeksforgeeks.org/file-handling-java/
This article on Geeks for Geeks explains file handling concepts in Java, which can be helpful for understanding how you manage data using .txt files in your project.

[4] Deitel, Paul and Harvey Deitel. "Java How to Program." Pearson, 2017.

[5] Liang, Y. Daniel. "Introduction to Java Programming, Comprehensive Version." Pearson, 2017.

[6] Stack Overflow. Retrieved from https://stackoverflow.com/

[7] Liang, Y. Daniel. "Introduction to Java Programming, Comprehensive Version." Pearson, 2017.