

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590018**



PROJECT REPORT ON
“PLACEMENT MANAGEMENT SYSTEM”
BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE & ENGINEERING

FILE STRUCTURES LABORATORY WITH MINI PROJECT [18ISL67]

Submitted by

Mayoori P – 4JK20IS028

Nischith Kulal – 4JK20IS032

Under the guidance of

Mrs Divya P

Assistant Professor

**Department of Information Science &
Engineering**

Mrs Sivapuram Jayasri

Assistant Professor

**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 ENGINEERING & 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 “**PLACEMENT MANAGEMENT SYSTEM**” is carried out by Ms. **MAYOORI P**, USN: **4JK20IS028**, and Mr. **NISCHITH**, USN: **4JK20IS032**, Students of sixth semester B.E. Information Science & Engineering, and submitted as a part of the course **FILE STRUCTURES LABORATORY WITH MINI PROJECTS [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 Project prescribed for the said Degree.

Mrs. Divya P
Project Guide

Mrs. Sivapuram Jayasri
Project Guide

Dr Suresha D
Head of the Department

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 **Mrs. Divya P, Assistant Professor**, Department of Information Science and Engineering, and **Mrs. Jayashri S, 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.

of Department of Information Science and Engineering for their valuable help and support.

Mayoori P 4JK20IS028

Nischith Kulal 4JK20IS032

ABSTRACT

The Placement Management System is a robust software application developed using the Java Swing Library, designed to efficiently manage college student placement information. It serves as a fast and flexible file-based computer system that aids in organizing and streamlining the data associated with campus placements. This feature allows users to input new student information and make necessary updates as required. Additionally, the system supports the deletion of existing records, ensuring that outdated or irrelevant data can be effectively removed. A comprehensive search feature is incorporated into the system, enabling users to search for student records based on different criteria such as unique student numbers (USN), branch of study, or specific companies of interest. The system also offers a display feature, allowing users to view and access all the records stored within the system. Furthermore, this functionality provides a transparent and traceable history of changes, ensuring accountability and facilitating audit processes. Moreover, the system incorporates an eligibility feature that allows users to view the placement eligibility of all students. This functionality categorizes students based on their eligibility criteria and displays the companies for which they are eligible. This information aids in the efficient allocation of resources and streamlining the placement process.

TABLE OF CONTENTS

Chapter No	Content	Page No
	Acknowledgement	i
	Abstract	ii
	Table of Contents	iii
	List of Figures	iv
1	INTRODUCTION	1-4
	1.2 Motivation and Scope	1
	1.3 Problem Statement	2
	1.4 Limitations	2
	1.5 Objectives of the Project	2
	1.6 Goals	3
2	REQUIREMENTS SPECIFICATION	5-7
	2.1 Software Requirements	6
	2.2 Functional Requirement	6
	2.3 Non -Functional Requirements	6
3	DESIGN OF THE APPLICATION	8-10
4	IMPLEMENTATION	10-17
	4.1 Description of Frame work used	11
	4.2 Description of Integrated Development Environment	14
	4.3 Description of Files Handling	15
5	RESULTS	18-24
6	CONCLUSION	25
	6.1 Future Scope	
	REFERENCES	

LIST OF FIGURES

Figure No.	Figure Name	Page No.
3.1	Use case Diagram	8
3.2	Class Diagram	9
4.1	Architectural Diagram of the Application	10
4.4.1	Inserting to the file	14
4.4.2	Search by Branch	15
4.4.3	Search by company	15
4.4.4	Search by Usn	16
4.4.5	Display	17
4.4.6	Delete	17
4.4.7	Home	18
5.1	Main Panel of Placement Management System	19
5.2	Insert Page	19
5.3	Delete Page	20
5.4	Display Page	20
5.5	Search Page	21
5.6	Search By USN Page (Match Found)	21
5.7	Search By Company Page	22
5.8	Search By Branch Page	22
5.9	Deleted data Page	23
5.10	Placement Eligibility Page	23
5.11	Student Records File	24
5.12	Deleted Changes File	24

CHAPTER 1

INTRODUCTION

Placement Management System stores record system that can be used as an application for the Placement Officer of the college to manage the student information with regards to placement. So all the information will store the details of the students including their background information, educational qualification, personal details, and all the information related to their resume. This system helps Company to access the student information with regards to placement.

Typical Features include:

- **Insert/Modify Student Records** – This system helps the placement team to insert each student record once and update it timely, and store it for further usages during placements, segregating students based on their CGPA, their interests and aspirations based on the eligibility criteria.
- **Delete Student Records** – This system enables options of deleting existing student records, or wrongly entered student record.
- **Eligibility View** – As a part of final consolidated view this system provides a view of student where each student's eligibility criteria is checked and corresponding eligible companies are displayed.
- **View Changes** – We have an option to view each and every changes done to each record from the time of insertion followed by any number of modifications done to the record.
- **Searching Student record** – As with placement system will always want to categorise students and view particular set of records at a time, this system enables such features by providing variety of search categories namely search by USN, search by branch, search by company.

Placement Management System provides a clear picture of all students where they stand in terms of placement eligibility criteria and which companies they are eligibility to, it provides an efficient search feature to the user which helps the placement handling team to search records of student by their USN, or set of records by each branch, or by the company name.

1.1 Motivation and Scope

Nowadays we all are using the internet to do multiple things like booking, academic search and blogging, apply for any job, etc. This system can be used as an application to manage student

information related to placement. The system handles student as well as company data and efficiently displays all this data to respective sides. This System does all work regarding placement like collecting student records and finding the eligibility of the students and is provided to Placement Officer, Company, College staff and students. This system can be used as an application for the Placement Officer of the college to manage the student information with regards to placements, which presently is a cumbersome task via excel sheet database.

1.2 Problem Statement

Design and develop a system for campus placement cell that manages student information in the college with regard to campus placements, which improves existing system. It has the facility of maintaining the details of the student. Placement Management System can be accessed throughout the college. Thus this stand-alone application will be used as an application for the Placement Officer of the college to manage the student information with regards to placements.

1.3 Limitations

Placement Management System uses components from the Java Swings which is pretty outdated in comparison to JFX and other GUI building technologies. There is scope of improvement in better GUI and more user interactivity options, People prefer HTML + JS over Java. We can embed AI system to categorise students based on their skills and interests and their eligibility. More over since our developed system is in the initial stages of development many functions are yet to be implemented such as,

- Data Compression
- Undo and Redo

1.4 Objectives of the project

Efficient Data Storage: The objective is to design and implement an optimized file structure that allows for efficient storage of student records. The file structure should minimize storage space requirements while ensuring quick and accurate retrieval of data.

Fast Retrieval and Searching: The objective is to develop indexing techniques within the file structure to enable fast retrieval and searching of student records. By implementing appropriate indexing mechanisms, the system can efficiently locate and access specific records based on search criteria, such as student ID, branch, or company.

Data Compression: The objective is to incorporate data compression techniques within the file structure to reduce the storage space needed for student records. Implementing compression algorithms can help optimize the utilization of storage resources while maintaining data integrity.

Data Integrity and Consistency: The objective is to ensure the integrity and consistency of student records stored in the file structure. Implementing mechanisms such as checksums, data validation, and transactional operations helps prevent data corruption and maintain the accuracy and reliability of the stored information.

Backup and Recovery: The objective is to establish a robust backup and recovery mechanism for the file structure. Regular backups should be performed to safeguard student records and facilitate the restoration of data in case of system failures or data loss.

Scalability and Flexibility: The objective is to design the file structure in a scalable and flexible manner to accommodate future growth and changes. The system should be able to handle a growing number of student records and adapt to evolving requirements without significant performance degradation.

Security and Access Control: The objective is to implement appropriate security measures and access control mechanisms within the file structure. This ensures that only authorized personnel can access and modify student records, protecting sensitive information from unauthorized access or tampering.

Integration with Other Modules: The objective is to integrate the file structure seamlessly with other modules or components of the Placement Management System. This allows for data exchange and interoperability between different system functionalities, ensuring a cohesive and interconnected system architecture.

By addressing these objectives within the file structure of the Placement Management System, the overall efficiency, performance, and usability of the system can be enhanced, resulting in an improved placement management experience for all stakeholders involved.

1.5 Goals

The Placement Management System aims to revolutionize the way student placement is handled in educational institutions. By leveraging the power of the Java Swing Library, the system provides a user-friendly interface that simplifies the process of managing student information. One of the primary objectives of the system is to centralize student records. By storing all student details in a structured and organized manner, the system eliminates the need for manual paperwork and scattered data. This centralization ensures that important information, such as

academic records, contact details, and eligibility criteria, is readily available and easily accessible.

The system also focuses on improving the efficiency of the placement process. Through the functionality of adding and modifying student records, placement officers can update student information as necessary, ensuring that the data is up to date and accurate. This enables seamless communication with prospective employers and simplifies the process of matching students with suitable job opportunities. Furthermore, the system's search functionality enhances productivity by allowing users to retrieve specific student records based on various search criteria. This saves time and effort, especially during high-intensity placement drives when prompt access to relevant information is crucial. The display feature of the system provides a comprehensive view of all student records, empowering placement officers to make informed decisions and analyze data effectively. This holistic perspective helps identify patterns, trends, and areas of improvement, enabling better planning and strategizing for future placement drives. The ledger feature plays a vital role in ensuring transparency and accountability. By tracking all modifications made to student records, the system creates an audit trail that can be referred to if needed. This feature fosters a sense of trust and integrity in the placement process, as all changes can be traced back to their source.

Overall, the Placement Management System is designed to optimize the placement process by centralizing student information, providing efficient search capabilities, facilitating data analysis, and ensuring transparency. By leveraging the Java Swing Library, the system offers a powerful and user-friendly platform that empowers placement officers, companies, and students to effectively manage and navigate the placement ecosystem.

CHAPTER 2

REQUIREMENT SPECIFICATION

This chapter specifies the functional and non-functional requirements of the Placement Management System which includes requirements like performance, reliability, usability, integrity.

2.1 Software requirements

The software used for the development of the project is:

- Operation system: Windows, Visual Studio Code
- Programming Language: Java, XML
- RAM: 8 GB
- Tools used: Java JDK tools,
- Database:

2.2 Hardware requirements

The hardware used for the development of the project is:

- Processor: Intel i3 or above
- RAM: 4GB minimum/8GB recommended

2.3 Functional requirements

• Inserting a record

This requires the user to create a new file handle, in this unit, User can insert a student record and write the content to the file. Next, if the record already exists then the student record is added to the Ledger File with modifications if any.

• Deleting an Existing Record

Here the user can delete the contents of a specific student record by entering the respective student's USN. If the record exists, then the contents of the record are deleted. Otherwise, an appropriate message is displayed.

• Displaying a Record

When a user enters this option, the user can view all the updated student records in the file.

• Searching a Record

Here the user can view the contents of a specific student record by entering the respective student's USN, search by branch or also search by company details. If the record exists, then

the contents of the record are displayed. Otherwise, an appropriate message is displayed.

View Changes

In this unit, the user can view the ledger file of each student and thereby can be updated with all the changes of all student records.

View Eligibility

Here the user can view the placement eligibility criteria of all the students in the record along with the various categories of placements based on college eligibility criteria.

2.4 Non-Functional Requirements

Performance

Performance of Placement Management System should always vary between a few hundred milliseconds. Time taken to creating records, Displaying records, Searching records, Deleting records, Viewing eligibility from within the file should be minimal.

Reliability

Placement Management System shall always provide fast and flexible file editing functionality no matter what the environment. System shall be robust enough to have high degree of fault tolerance. For example, if a user tries to open an invalid file it should display a proper error message. It shall be able to recover from hardware failures, power failures and other natural catastrophes and rollback the files to their most recent valid state.

Usability

Placement Management System shall provide an easy-to-use graphical interface similar to other existing text file editors so that the users do not have to learn a new style of interaction. Any notification or error messages generated by this system shall be clear, succinct, polite and free of jargon.

Integrity

The system must be programmed properly to prevent exploitation through buffer overflows etc. The system should be secure and could use encryption to protect the files. Users need to be authenticated before having access to any personal data.

Interoperability

The System shall minimize the effort required to couple it to another system, such as an Integrated Development Environment.

2.3 Domain Constraints

Hardware limitations: There must be a 64 MB on board memory. The software should be designed to utilize available hardware resources efficiently, considering memory constraints.

Inadequate memory may lead to performance issues, such as slow response times and system crashes. The system should have mechanisms in place to handle memory limitations gracefully, such as implementing data compression or optimizing memory usage.

Control functions: The software should provide a user-friendly interface with intuitive controls and clear instructions to facilitate ease of use. Error messages should be informative, concise, and presented in a user-friendly manner to assist users in troubleshooting issues.

The system should include appropriate validation checks and error handling mechanisms to ensure data integrity and prevent erroneous input. User input should be validated to prevent security vulnerabilities, such as SQL injection or cross-site scripting attacks. Control functions should follow industry-standard best practices for usability, including consistent navigation, logical flow, and responsive design.

Dependencies: The system relies on JDK 8 (Java Development Kit) for its core functionality and must be installed on the target system. Java SE (Standard Edition) is a prerequisite for the system, as it provides a platform for running Java applications. The system heavily utilizes the Java Swing Library for building the graphical user interface (GUI) components. Ensuring the compatibility and availability of the required dependencies is crucial for the successful installation and functioning of the system. Upgrading or modifying the dependencies should be carefully managed to maintain compatibility and avoid any disruption in system functionality.

Safety/security considerations: The system should prioritize data security by implementing robust authentication and authorization mechanisms to prevent unauthorized access.

Sensitive user information, such as passwords or personal data, should be securely encrypted and stored to protect against data breaches. The application should adhere to secure coding practices, including input validation, output encoding, and proper error handling, to mitigate security risks. Regular software updates and patches should be applied to address any identified security vulnerabilities and ensure a secure environment. The system should have a proper log management system in place to track and monitor user activities for auditing and security analysis purposes.

CHAPTER 3

SYSTEM/REQUIREMENTS ANALYSIS

This chapter depicts about all System/Requirement Analysis and the functions that user can perform in Placement Management System.

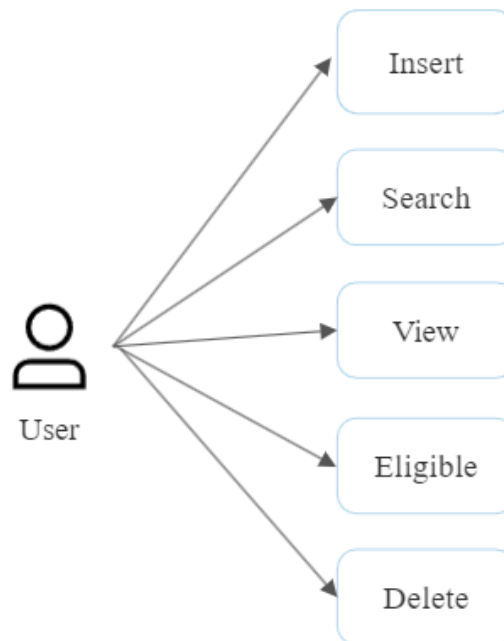


Figure 3.1 Use case diagram

The overall description of the system in the above figure 3.1 is as follows:

The user is first presented with the Main Menu. Here user has a set of six options:

Insert: In this unit, User can insert a student record and write the content to the file. Next, if the record already exists then the student record is added to the Ledger File.

Display: In this unit, the user can view all the updated student records in the file.

Delete: In this unit, the user can delete the contents of a specific student record by entering the respective student's USN.

Search: Here the user can retrieve the required student's record by entering his/her USN, branch, company.

View Changes: In this unit, the user can view the ledger file of each student and thereby can be updated with all the changes of all student records.

View Eligibility: In this unit, the user can view the entire list of records and students eligibility to various companies.

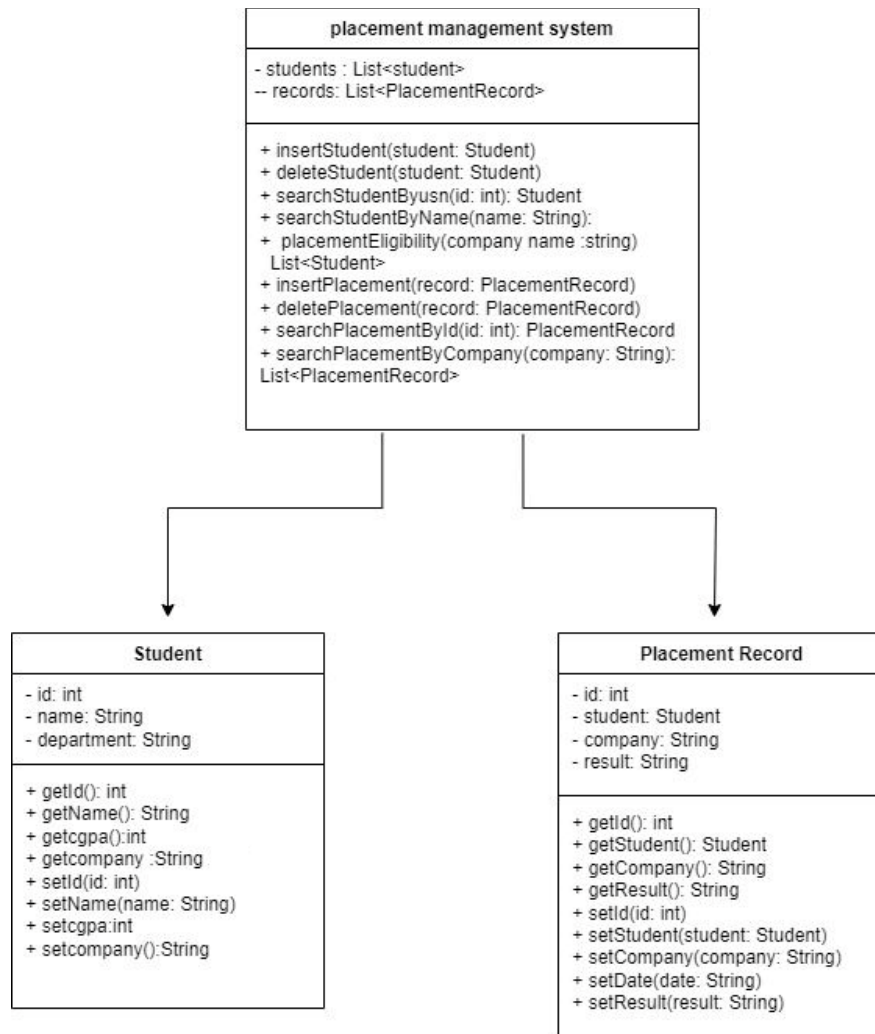


Figure 3.2 Class diagram

The UML diagram figure 3.2 depicts the Placement Management System, presenting the relationships between the main system class, Student class, and PlacementRecord class. It provides a visual representation of the system's structure and functionality. The main system class coordinates the management of students and placement records. The Student class encapsulates attributes like ID, name, and department, while the PlacementRecord class stores information about placements, including ID, student details, company name, date, and result. This comprehensive diagram offers a clear understanding of the system's components and their relationships, facilitating efficient and effective management of campus placements

CHAPTER 4

IMPLEMENTATION

Implementation is defined as specific set of activities designed to put into practice an activity or program of known dimensions. Implementation processes are purposeful and are described in sufficient details such that independent can detect the presence and strength of the "specific set of activities related to implementation.

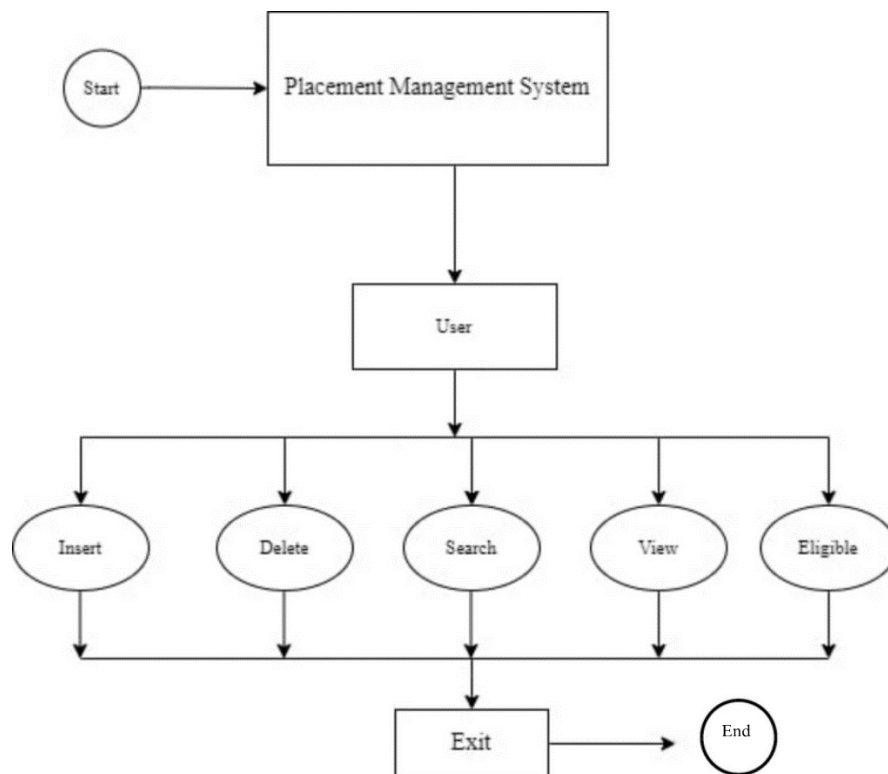


Figure 3.2 Architectural Diagram

The architectural diagram of the Campus Placement Management System illustrates the high-level structure and components of the system, showcasing how different modules and layers interact to support its functionality. It depicts the overall system architecture, including the client-side interfaces, server-side components, and the interaction between them. The diagram highlights the key architectural patterns and design principles employed in the system, such as the use of a layered architecture, a client-server model, and various integration points with external systems. It provides a visual representation of the system's architectural decisions and serves as a blueprint for developers and stakeholders to understand and communicate the system's overall design and structure.

4.1 Description of Frameworks Used

GUI framework that provides a comprehensive set of classes and components for creating visually appealing and user-friendly graphical interfaces in Java applications. It offers a wide range of features, including various GUI controls, layout managers, and customizable look and feel options.

Java Swing

Java Swing is a GUI Framework that contains a set of classes to provide more powerful and flexible GUI components than AWT. Swing provides the look and feel of modern Java GUI. Swing library is an official Java GUI tool kit released by Sun Microsystems. It is used to create graphical user interface with Java. It defines infrastructure common to most desktop applications, making Swing applications easier to create.

Swing classes are defined in javax.swing package and its sub-packages. Features of JFC:

- Swing GUI components.
- Look and Feel support.
- Java 2D.

Swing Classes

JPanel: JPanel is Swing's version of AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent.

JFrame: JFrame is Swing's version of Frame and is descended directly from Frame class. The component which is added to the Frame, is referred as its Content.

JWindow: This is Swing's version of Window and has descended directly from Window class. Like Window it uses BorderLayout by default.

JLabel: JLabel has descended from JComponent, and is used to create text labels.

JButton: JButton class provides the functioning of push button. JButton allows an icon, string or both associated with a button.

JTextField: JTextField allow editing of a single line of text.

4.2 Description of Integrated Development Environment

Eclipse

Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc.

The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plug-ins and is designed to be extensible using additional plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available.

The Java Development Tools (JDT) project provides a plug-in that allows Eclipse to be used as a Java IDE, PyDev is a plugin that allows Eclipse to be used as a Python IDE, C/C++ Development Tools (CDT) is a plug-in that allows Eclipse to be used for developing application using C/C++, the Eclipse Scala plug-in allows Eclipse to be used an IDE to develop Scala applications and PHPEclipse is a plug-in to eclipse that provides complete development tool for PHP.

Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated development tools

Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License, although it is incompatible with the GNU General Public License.[11] It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

4.3 Description of Files Handling

Chosen File Organization Technique: Sequential File Structure

The sequential file structure has been selected for the Placement Management System due to its simplicity and suitability for sequential record access. This file organization technique maintains records in a sequential manner, enabling efficient insertion and deletion operations. Moreover, it preserves the order of records, which is important for maintaining historical data integrity.

- Designing the File Structure

A well-designed file structure is crucial for optimal performance and data management within

the Placement Management System. This section focuses on the key components of the file structure design.

- **Record Layout and Structure**

The file structure design incorporates a well-defined record layout and structure to store student information. Each record consists of fields representing different attributes such as student ID, name, CGPA, branch, and contact details. The layout is carefully designed to ensure efficient access and retrieval of specific fields.

- **Metadata and Data Dictionary**

To facilitate data management and interpretation, the file structure implementation includes metadata and a data dictionary. Metadata provides essential information about the file, such as the number of records and record length. The data dictionary contains descriptions of the fields and their attributes, aiding in data integrity and interpretation.

- **Insertion Operation**

Inserting student records into the file structure is a fundamental operation. This section outlines the process flow and steps involved in inserting records.

- **Deletion Operation**

The deletion operation enables the removal of student records from the file structure when necessary. This section presents an overview of the deletion process.

The deletion process starts with a request to delete a specific student record. The system locates the record in the file structure based on the search key (e.g., student ID) and removes it from the file. The deletion algorithm follows these steps:

Search for the record to be deleted using the specified search key.

Remove the record from the file structure.

Update the metadata and rearrange the file if necessary to maintain order.

- **Search and Retrieval Operations**

Efficient search and retrieval capabilities are vital for the Placement Management System. This section highlights the implementation of search functionality using the sequential file structure. The sequential search algorithm scans each record sequentially until the desired record is found. This approach is suitable for small to medium-sized datasets. To optimize search operations, enhancements such as indexing can be implemented. Indexes, such as primary or secondary indexes, allow for faster record retrieval based on specific search criteria, reducing the search time complexity.

4.4 Source code

The provided source code offers insights into the implementation and functionality of the software program. The code has been meticulously designed and structured to ensure clarity, maintainability, and scalability. It adheres to best practices and coding standards, promoting seamless collaboration among developers and enhancing the overall quality of the software.

```
String formattedData = name + "|" + usn1 + "|" + sem + "|" + branch + "|" + cgpa + "|" + nob + "|" + company
+ "|" + ctc + "|" + comments + "|";
try {
    File file = new File(pathname:"student.txt");
    if (!file.exists()) {
        file.createNewFile();
    }

    // Read existing data from file
    List<String> existingData = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(file));
    String line;
    boolean isUSNExisting = false; // Flag to check if USN already exists
    while ((line = reader.readLine()) != null) {
        existingData.add(line);
        String[] Data = line.split(regex:"\\|");
        if (Data.length > 1 && Data[1].equals(usn1)) {
            isUSNExisting = true; // USN already exists
        }
    }
    reader.close();

    if (isUSNExisting) {
        showMessageDialog(parentComponent:null, message:"USN already exists!");
        return; // Stop further execution
    }

    // Add the new data and sort by USN
    existingData.add(formattedData);
    Collections.sort(existingData, new Comparator<String>() {
        public int compare(String s1, String s2) {
            String[] usn1 = s1.split(regex:"\\|");
            String[] usn2 = s2.split(regex:"\\|");
            return usn1[1].compareTo(usn2[1]);
        }
    });
}
```

Figure 4.4.1 Inserting to the file

The 4.4.1 provided code snippet focuses on the insertion of new student records into a file called "student.txt" while ensuring the prevention of duplicate records with the same University Serial Number (USN). The code first formats the data to be inserted, combining various student attributes using the "|" delimiter. It then checks if the file "student.txt" exists and creates it if necessary. The existing data in the file is read line by line and stored in a list called "existingData". During this process, a flag called "isUSNExisting" is set to determine if the USN already exists in the file. If a duplicate USN is found, a message dialog is displayed, and the execution is halted. If the USN is unique, the new record is added to the "existingData" list. Finally, the "existingData" list is sorted based on the USN in ascending order using a custom comparator. This code ensures the integrity of the data by avoiding the inclusion of duplicate records with the same USN in the "student.txt" file.

```

if(ae.getSource()==search)
{
    String branch1 = (String) branchT.getSelectedItem();
    try
    {
        String name = "", usn = "", sem = "", branch = "", cgpa = "", nob = "", company = "", ctc = "", comments = "", r;

        File temp = new File(pathname+"temp.txt");
        Boolean createNewFile1 = temp.createNewFile();
        BufferedWriter pw = new BufferedWriter(new FileWriter(fileName+"temp.txt"));
        String b = "NAME\\t|USN\\t|SEM\\t|BRANCH\\t|CGPA\\t|NOB\\t|COMPANY\\t|CTC\\t|COMMENTS";
        pw.write(b);
        pw.write(str+"\n");

        BufferedReader br = new BufferedReader(new FileReader(fileName+"student.txt"));
        while((r= br.readLine()) !=null)
        {
            String[] result = r.split(regex:"\\|");
            name=result[0];
            usn=result[1];
            sem= result[2];
            branch=result[3];
            cgpa=result[4];
            nob=result[5];
            company=result[6];
            ctc=result[7];
            comments=result[8];
            if(name.equals(anObject:"999"))
                break;
            if(branch.equals(branch1)) {
                String bb = name + "\\t|" + usn + "\\t|" + sem + "\\t|" + branch + "\\t|" + cgpa + "\\t|" + nob + "\\t|" + company + "\\t|" + ctc + "\\t|" + comments;
                pw.write(bb);
                pw.write(str+"\n");
            }
        }
    }
}

```

Figure 4.4.2 Search by Branch

The 4.4.2 code snippet provided is responsible for performing a search operation based on the selected branch from a dropdown menu. It reads the data from the "student.txt" file and filters the records based on the branch selected. The filtered records are then written to a temporary file called "temp.txt" for display purposes. The content of the "temp.txt" file is read and displayed in the output text area. Finally, the temporary file is deleted. This code enables the user to search and display student records based on a specific branch, providing a convenient way to retrieve relevant information.

```

if(ae.getSource()==search)
{
    String company1 = (String) companyT.getSelectedItem();
    //String company1 = companyT.getText();
    try
    {
        String name = "", usn = "", sem = "", branch = "", cgpa = "", nob = "", company = "", ctc = "", comments = "", r;

        File temp = new File(pathname+"temp.txt");
        Boolean createNewFile1 = temp.createNewFile();
        BufferedWriter pw = new BufferedWriter(new FileWriter(fileName+"temp.txt"));
        String b = "NAME\\t|USN\\t|SEM\\t|BRANCH\\t|CGPA\\t|NOB\\t|COMPANY\\t|CTC\\t|COMMENTS";
        pw.write(b);
        pw.write(str+"\n");

        BufferedReader br = new BufferedReader(new FileReader(fileName+"student.txt"));
        while((r= br.readLine()) !=null)
        {
            String[] result = r.split(regex:"\\|");
            name=result[0];
            usn=result[1];
            sem= result[2];
            branch=result[3];
            cgpa=result[4];
            nob=result[5];
            company=result[6];
            ctc=result[7];
            comments=result[8];
            if(name.equals(anObject:"999"))
                break;
            if(company.equals(company1)) {
                String bb = name + "\\t|" + usn + "\\t|" + sem + "\\t|" + branch + "\\t|" + cgpa + "\\t|" + nob + "\\t|" + company + "\\t|" + ctc + "\\t|" + comments;
                pw.write(bb);
                pw.write(str+"\n");
            }
        }
    }
}

```

Figure 4.4.3 Search by Company

The 4.4.3 code snippet provided is responsible for performing a search operation based on the selected company from a dropdown menu. It reads the data from the "student.txt" file and filters

the records based on the branch selected. The filtered records are then written to a temporary file called "temp.txt" for display purposes. The content of the "temp.txt" file is read and displayed in the output text area. Finally, the temporary file is deleted. This code enables the user to search and display student records based on a specific branch, providing a convenient way to retrieve relevant information.

```

if(ae.getSource()==search)
{
    String usn = usnT.getText();
    try
    {
        String name = "", usn1 = "", sem = "", branch = "", cgpa = "", nob = "", company = "", ctc = "", comments = "", r;
        BufferedReader br = new BufferedReader(new FileReader(fileName+"student.txt"));
        while((r= br.readLine()) !=null)
        {
            String[] result = r.split(regex:"\\|");
            name=result[0];
            usn1=result[1];
            sem= result[2];
            branch=result[3];
            cgpa=result[4];
            nob=result[5];
            company=result[6];
            ctc=result[7];
            comments=result[8];
            if(usn1.equals(usn))
            {
                File temp = new File(pathname+"temp.txt");
                Boolean createNewFile = temp.createNewFile();
                BufferedWriter pw = new BufferedWriter(new FileWriter(temp));
                String b = "NAME\\t|USN\\t|SEM\\t|BRANCH\\t|CGPA\\t|NOB\\t|COMPANY\\t|CTC\\t|COMMENTS";
                String bb = name + "\\t|" + usn1 + "\\t|" + sem + "\\t|" + branch + "\\t|" + cgpa + "\\t|" + nob + "\\t|" + company + "\\t|" + ctc
                pw.write(b);
                pw.write(str: "\\n");
                pw.write(bb);
                pw.write(str: "\\n");
                pw.close();
                br.close();
                File file = new File(pathname+"temp.txt");
            }
        }
    }
}

```

Figure 4.4.4 Search by Usn

The 4.4.4 code snippet provided is responsible for performing a search operation based on the selected Usn from a dropdown menu. It reads the data from the "student.txt" file and filters the records based on the branch selected. The filtered records are then written to a temporary file called "temp.txt" for display purposes. The content of the "temp.txt" file is read and displayed in the output text area. Finally, the temporary file is deleted. This code enables the user to search and display student records based on a specific branch, providing a convenient way to retrieve relevant information.

The 4.4.5 provided code snippet is responsible for reading data from a file, either "student.txt" or "deleted_records.txt" based on the selected option. It then processes each line of the file, splitting the data into individual fields. The field values are assigned to corresponding variables such as name, USN, semester, branch, CGPA, number of offers, company, CTC, and comments. If the name value is "999", the loop is terminated. The data is then formatted and written to a temporary file called "temp.txt" with appropriate column headers. This code segment allows for reading and writing data from files, facilitating the display or storage of records based on the selected option.

```
try {
    String name = "", usn = "", sem = "", branch = "", cgpa = "", nob = "", company = "", ctc = "", comments = "", r;

    File temp = new File(pathname:"temp.txt");
    Boolean createNewFile1 = temp.createNewFile();
    BufferedWriter pw = new BufferedWriter(new FileWriter(fileName:"temp.txt"));
    String b = "NAME\t|USN\t|SEM\t|BRANCH\t|CGPA\t|NOB\t|COMPANY\t|CTC\t|COMMENTS";
    pw.write(b);
    pw.write(Str:"\n");

    String fileToRead = "";
    if (selectedOption.equals(anObject:"Display records")) {
        fileToRead = "student.txt";
    } else if (selectedOption.equals(anObject:"Deleted records")) {
        fileToRead = "deleted_records.txt";
    }

    BufferedReader br = new BufferedReader(new FileReader(fileToRead));
    while ((r = br.readLine()) != null) {
        String[] result = r.split(regex:"\\|");
        name = result[0];
        usn = result[1];
        sem = result[2];
        branch = result[3];
        cgpa = result[4];
        nob = result[5];
        company = result[6];
        ctc = result[7];
        comments = result[8];
        if (name.equals(anObject:"999"))
            break;
        String bb = name + "\t|" + usn + "\t|" + sem + "\t|" + branch + "\t|" + cgpa + "\t|" + nob + "\t|"
            + company + "\t|" + ctc + "\t|" + comments;
        pw.write(bb);
        pw.write(Str:"\n");
    }
}
```

Figure 4.4.5 Display

```
    }
    bwStudent.write(line);
    bwStudent.newLine();
}

} catch (FileNotFoundException e) {
    JOptionPane.showMessageDialog(parentComponent:null, "File not found: " + e.getMessage());
    return;
} catch (IOException e) {
    JOptionPane.showMessageDialog(parentComponent:null, "Error reading/writing file: " + e.getMessage());
    return;
}

if (!foundInStudent) {
    JOptionPane.showMessageDialog(parentComponent:null, message:"No matching records found.");
} else {
    JOptionPane.showMessageDialog(parentComponent:null, message:"Record(s) Deleted!");
}

File studentFile = new File(pathname:"student.txt");
File tempStudentFile = new File(tempStudentPath);

if (studentFile.delete()) {
    if (tempStudentFile.renameTo(studentFile)) {
        this.dispose();
        Home h = new Home();
        h.setSize(width:2300, height:790);
        h.setVisible(b:true);
    } else {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Error renaming the file.");
    }
} else {
    JOptionPane.showMessageDialog(parentComponent:null, "Error deleting the file: " + studentFile.getAbsolutePath());
}
}
```

Figure 4.4.6 Delete

The provided code snippet handles the process of deleting records from a file. It checks for specific conditions based on the selected field (USN or Name) and the corresponding value. If a match is found, the record is considered as "foundInStudent" and is written to a separate file called "bwDeleted". If no match is found, the record is written to the original "bwStudent" file. After the deletion process, appropriate messages are displayed to indicate the success or failure of the deletion operation. Next, the code deletes the original "student.txt" file and renames the temporary file to replace it. If the deletion and renaming operations are successful, the current window is closed, and a new Home window is created and displayed.

Additionally, the code includes an action listener for the "back" button, which closes the current window and opens a new Home window when clicked. The main method initializes and displays an instance of the Delete class, setting its size and visibility accordingly.

```
public void actionPerformed(ActionEvent ae)
{
    if(ae.getSource()==insert)
    {
        this.dispose();
        Insert in=new Insert();
        in.setSize(width:2300,height:790);
        in.setVisible(b:true);
    }

    if(ae.getSource()==delete)
    {
        this.dispose();
        Delete del=new Delete();
        del.setSize(width:2300,height:790);
        del.setVisible(b:true);
    }

    if(ae.getSource()==display)
    {
        this.dispose();
        Display dis=new Display();
        dis.setSize(width:2300,height:790);
        dis.setVisible(b:true);
    }

    if(ae.getSource()==search1)
    {
        this.dispose();
        Search1 ser=new Search1();
        ser.setSize(width:2300,height:790);
        ser.setVisible(b:true);
    }
}
```

Figure 4.4.7 Home

The provided code snippet represents the actionPerformed method, which handles different actions based on the event source. If the "insert" button is clicked, the current window is closed, and an instance of the Insert class is created and displayed. If the "delete" button is clicked, the current window is closed, and an instance of the Delete class is created and displayed. If the "display" button is clicked, the current window is closed, and an instance of the Display class is created and displayed. If the "search1" button is clicked, the current window is closed, and an instance of the Search1 class is created and displayed. If the "eligible" button is clicked, the current window is closed, and an instance of the Eligible class is created and displayed. The code also includes a nested class named RoundBtn, which implements the Border interface to create rounded buttons. The main method creates an instance of the Home class, sets its size, and makes it visible.

CHAPTER 5

RESULT

Home page of the application which helps navigating to different functionalities.

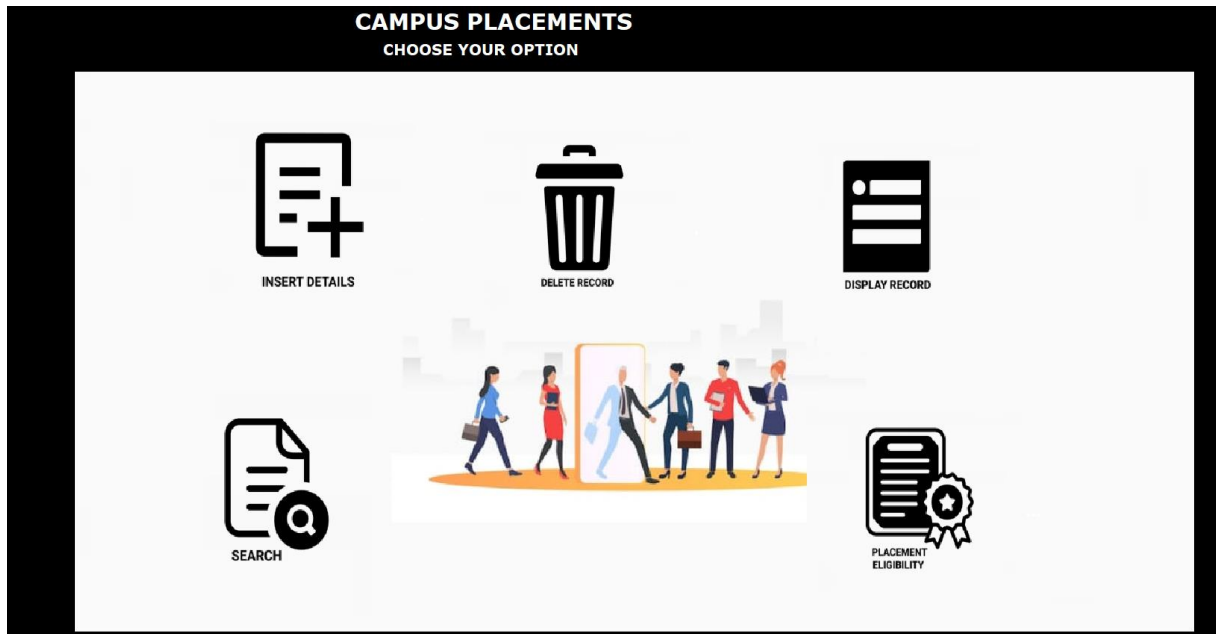


Fig 5.1: Main Panel of Placement Management System

Insert/Modify page takes all user inputs of student records and checks for duplicates.

Fig 5.2: Insert Page

Delete record page of the application which takes USN as user input and on the press of delete button the corresponding record is deleted.

Fig 5.3: Delete Page

Display record page of the application, when the Display button is pressed all the records in the file are displayed, Go back button navigates to home page.

The records are: Display records

NAME	USN	SEM	BRANCH	CGPA	NOB	COMPANY	CTC	COMMENTS
mayu	001	01	CSE	10	0	Amazon	24,00,000	nil
nischith	01	01	CSE	8.9	0	SAP_LABS	18,00,000	no
Ashwitha	013	06	ISE	9.8	0	Amazon	24,00,000	no
Mayoori	014	06	ISE	9	0	Amazon	24,00,000	no

Go Back

Fig 5.4: Display Page

Search record page of the application, provides us with different categories of search and navigates us to corresponding search page.

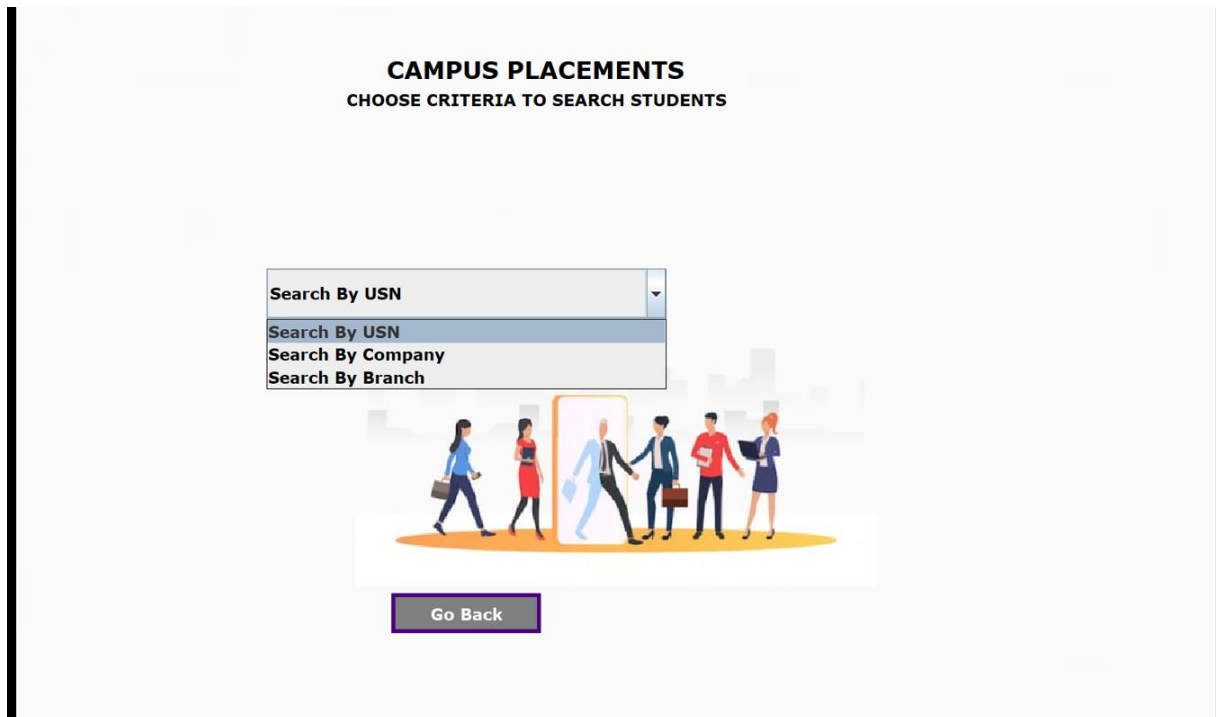


Fig 5.5: Search Page

Search by USN page of the application, when the USN is entered & button is pressed the record matching with USN and if found in file then it is displayed.

Enter usn of record to be searched

NAME	USN	SEM	BRANCH	CGPA	NOB	COMPANY	CTC	COMMENTS
Mayoori	014	06	ISE	9	0	Amazon	24,00,000	no

Fig 5.6: Search By USN Page (Match Found)

Search by Branch page of the application, when the branch is selected from dropdown is entered & button is pressed all the records matching with branch selected, and if found in file then it is displayed.

Enter Branch of record to be searched

CSE

NAME	USN	SEM	BRANCH	CGPA	NOB	COMPANY	CTC	COMMENTS
mayu	001	01	CSE	10	0	Amazon	24,00,000	nil
nischith	01	01	CSE	8.9	0	SAP_LABS	18,00,000	no

Search Go Back

Fig 5.7: Search By Branch

Search by Company page of the application, when the branch is selected from dropdown is entered button is pressed all the records matching with company selected, and if found in file then it is displayed.

Enter Company of record to be searched

Amazon

NAME	USN	SEM	BRANCH	CGPA	NOB	COMPANY	CTC	COMMENTS
mayu	001	01	CSE	10	0	Amazon	24,00,000	nil
Ashwitha	013	06	ISE	9.8	0	Amazon	24,00,000	no
Mayoori	014	06	ISE	9	0	Amazon	24,00,000	no

Search Go Back

Fig 5.8: Search By Company

View Changes page acts as a deleted file and displays all the changes done to a particular record and keeps a log of all activities performed.

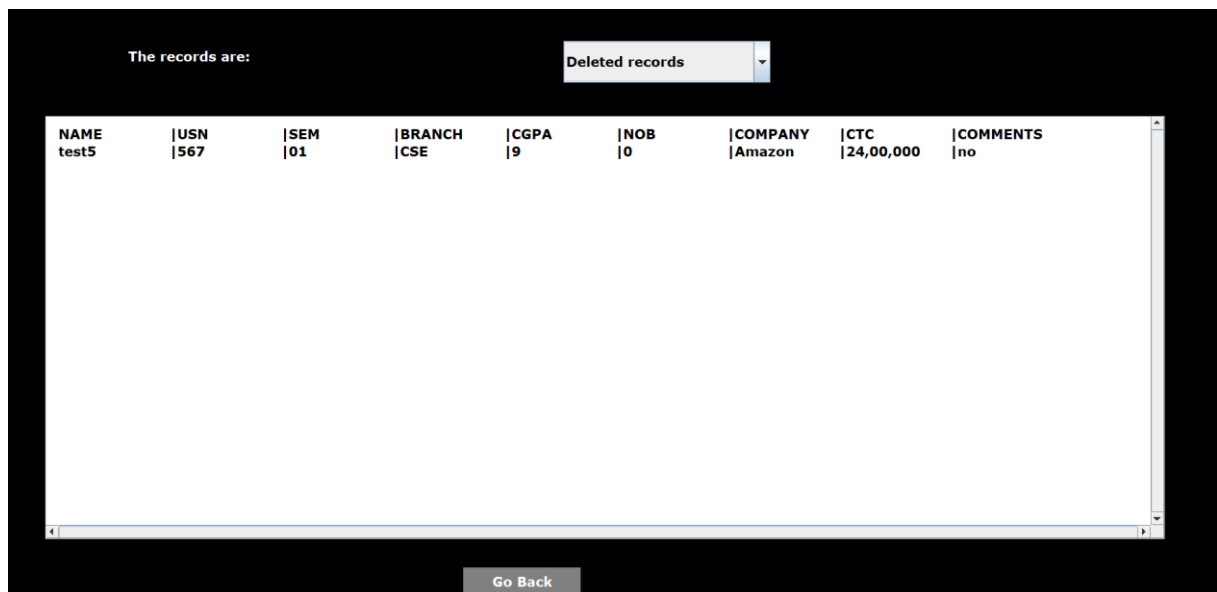


Fig 5.9: Deleted data Page

Placement Eligibility page displays records of all students and their eligibility to companies based on the credentials entered while inserting record.

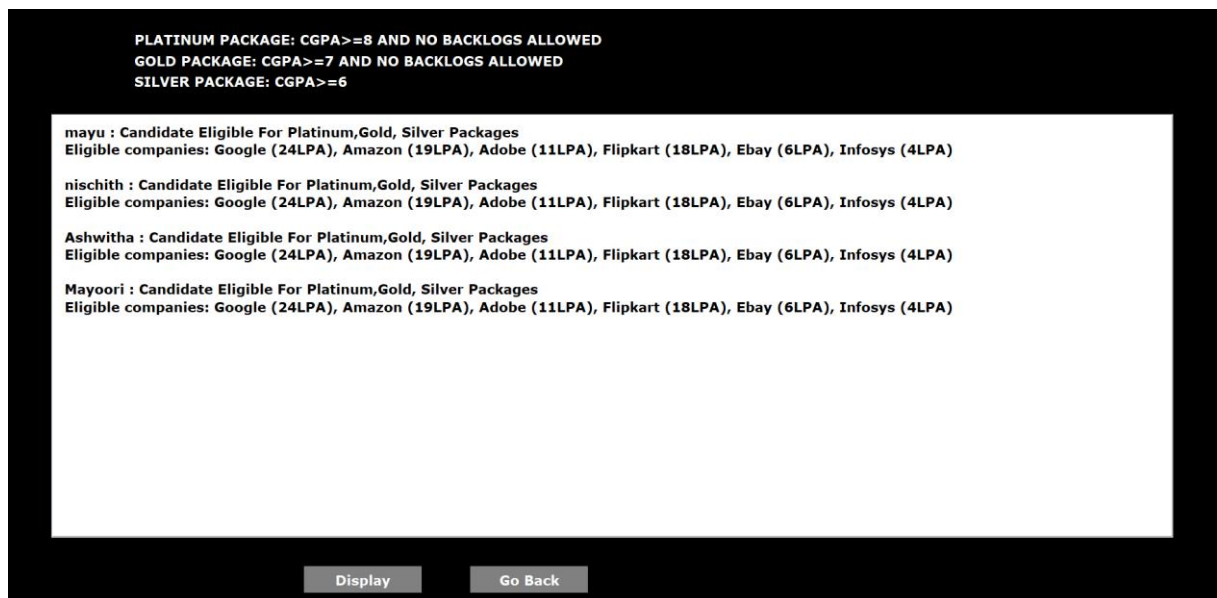
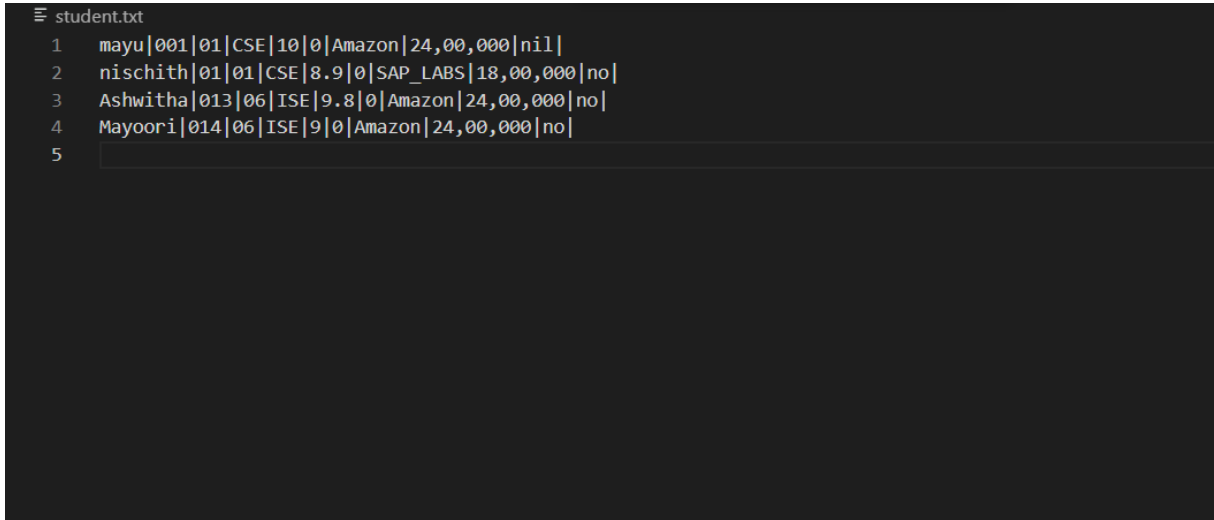


Fig 5.10: Placement Eligibility Page

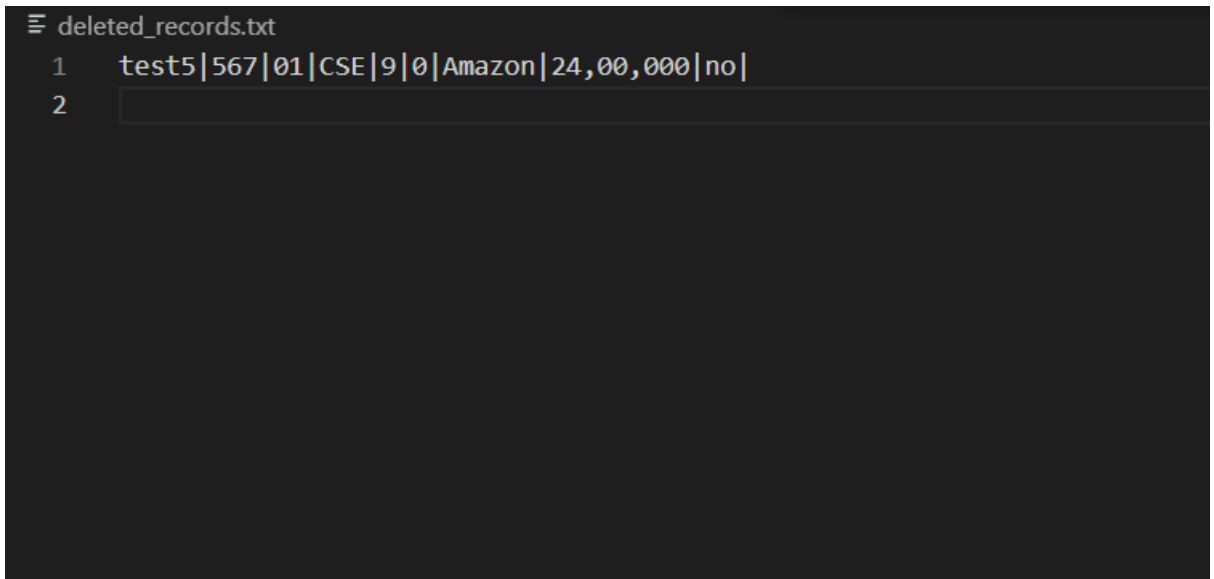
Student.txt file stored on disk containing all student record and terminated by default 999 record.



```
student.txt
1 mayu|001|01|CSE|10|0|Amazon|24,00,000|nil|
2 nischith|01|01|CSE|8.9|0|SAP_LABS|18,00,000|no|
3 Ashwitha|013|06|ISE|9.8|0|Amazon|24,00,000|no|
4 Mayoore|014|06|ISE|9|0|Amazon|24,00,000|no|
5
```

Fig 5.11: Student Records File

Deleted.txt file stored on disk containing all student record and the modifications done on them.



```
deleted_records.txt
1 test5|567|01|CSE|9|0|Amazon|24,00,000|no|
2
```

Fig 5.12: Deleted Changes File

CHAPTER 6

CONCLUSION

Once this project is completed it allows users to easily create new student records and also allows them to modify it whenever there is a change in CGPA, number of backs, etc. It allows the users to delete records if needed. All records with the latest details can be displayed whenever the user needs it. Users can search for a particular record as well which displays the requested record with the latest details and offers categories of search like search by USN, search by branch and search by company. Using General Ledger Problem, users are provided with a feature where they can see the history of all changes made to all records. Finally, it provides information about the eligibility criteria for various packages and displays the list of all the company and packages each student is eligible for. Overall, it provides the necessary features for managing a placement database in the form of files. This application is a Cross Platform and Architecture independent and can run efficiently on Windows, Mac or Linux as long as JAVA installation files are present on the system

6.1 Future scope

We are planning to keep managing the project and improving it based on user feedback. Here is our to do list for future :

- Add some more categories.
- Make it more user friendly than it is now.
- Add application submitting features to this.

REFERENCES

- [1] Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.
- [2] Herbert Schildt: JAVA the Complete Reference, 7th/9th Edition, Tata McGraw Hill, 2007.
- [3] Jim Keogh: J2EE-TheCompleteReference, McGraw Hill, 2007.
- [4] StackOverflow: www.stackoverflow.com
- [5] Codeproject: www.codeproject.com
- [6] Javapoint: www.javapoint.com