VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018, KARNATAKA

A Mini Project Report on

"CRICKET SCORE SHEET MAINTAINANCE"

Submitted in the partial fulfillment for the requirements for the FS Lab with Mini Project (18ISL68)

in

INFORMATION SCIENCE AND ENGINEERING

By

Mr. Lakshya Agarwal	**USN: 1BY20IS072**
Mr. Manish Kumar Yadav	**USN: 1BY20IS078**
Mr. Mridul Sadashiv	**USN: 1BY20IS090**

Under the guidance of

Dr. Shanti D L
Assistant Professor
Department of ISE, BMSIT&M.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU-560064

2022-2023

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI – 590 018, KARNATAKA

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU-560064

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"Cricket Score Sheet Maintainance using Indexing"** is a bonafide work carried out by **Mr. Lakshya Agarwal (1BY20IS072)**, **Mr. Manish Kr Yadav (1BY20IS078), Mr. Mridul Sadashiv (1BY20IS090)** in partial fulfillment of File structures Lab with Mini Project (15ISL68) for the award of **Bachelor of Engineering Degree in Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2017-18. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report. The project report has been approved as it satisfies the academic requirements in respect of Mini Project work for the B.E Degree.

Signature of the Guide	**Signature of the HOD**
Dr. Shanti D L	Dr. Pushpa S K
Assistant Professor	Professor and Head
Department of ISE	Department of ISE

EXTERNAL EXAMINERS

Name of the Examiners	Signature with Date
1.	
2.	

ACKNOWLEDGEMENT

We are happy to present this mini project after completing it successfully. This mini project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this mini project a success.

We heartily thank our **Principal, Dr. Mohan Babu G.N, B M S Institute of Technology & Management** for his constant encouragement and inspiration in taking up this mini project.

We heartily thank our **Head of Department Dr. Manjunath T. N, Dept. of Information Science and Engineering, B M S Institute of Technology& Management** for his constant encouragement and inspiration in taking up this mini project.

We gracefully thank our Project guide**, Dr. Shanti D L, Asst. Professor, Dept. of Information Science and Engineering,** for her encouragement and advice throughout the course of the mini project work.

Special thanks to all the staff members of Information Science Department for their help and kind co-operation.

We also thank our parents and friends for their unconditional love and encouragement and support given to us in order to finish this precious work.

Last but not the least we would like to thank God for giving us the strength and motivation through the course of this Project.

By,

Lakshya Agarwal

Manish Kumar Yadav

Mridul Sadashiv

ABSTRACT

The Cricket Score Sheet Maintainance is a project written in Java that utilizes indexing to efficiently manage and store cricket information. This system aims to provide a streamlined and reliable solution for managing Cricket data, allowing for easy retrieval and manipulation of information.

The project leverages, a specialized data structure, to organize and index the cricket data. These are well-suited for handling large datasets and provide efficient search, insert, and delete operations. By implementing , the Cricket Score Sheet Maintainance can achieve optimal performance in terms of speed and scalability.

The system offers various operations to interact with the cricket data, including adding new player records, searching for specific records, updating existing records, and deleting records. These operations are designed to be intuitive and user-friendly, allowing administrators or authorized personnel to efficiently manage cricket information.

The cricket sheet File Management System incorporates robust error handling mechanisms to ensure data integrity and prevent unauthorized access. It includes features such as authentication and access control to restrict actions based on user privileges. The system also incorporates data validation techniques to enforce data consistency and accuracy.

The project is implemented in Java, making it platform-independent and easily deployable on different operating systems. It follows modular and object-oriented design principles, promoting code reusability and maintainability. The user interface is designed to be intuitive and user-friendly, ensuring ease of use for administrators and staff.

Overall, the Cricket Score Sheet Maintainance provides an efficient and reliable solution for managing cricket information using indexing. It aims to enhance the efficiency of cricket sheet registration processes, streamline data management, and ensure the integrity and security of Player records.

TABLE OF CONTENTS

LIST OF FIGURES/TABLES

Introduction

1.1 Outline

The CRICKET DATA System provides a user-friendly, interactive Menu Driven Interface (MDI) based on local file systems. All data is stored in files on disk. The system uses file handles to access the files. This System is used by the person to view his/her CRICKET details.

The system is initially used to add person records containing the C_ID, name, dob, address, gender. The system can then be used to search, delete, modify or display existing records of all the citizens In the SCORE SHEET System, the ID field is a character array that can hold a numeric value of some maximum size. The size of the array is larger than the longest string it can hold. We preserve the identity of fields by separating them with. delimiters. We have chosen the vertical bar character, as the delimiter here.

Cricket Score Sheet project is a simple project built using the. It uses file handling to store various information regarding runs, wickets, overs, extras, and many more.

The program can display runs, wickets, names of batsmen and bowlers, overs, extras, economy of bowler, strike rate of batsmen, etc. It also displays the date and time of the is complete, error-free and easy to understand.

In this System, we have a fixed number of fields, each with a maximum length, that combine to make a data record. Fixing the number of fields in a record does not imply that the size of fields in the record is fixed. The records are used as containers to hold a mix of fixed and variable-length fields within a record. We have 5 contiguous fields.

This approach ensures that the score data file management system can handle a large volume of data while maintaining excellent performance, making it suitable for electoral authorities and organizations responsible for managing and verifying Player information.

The problem statement for using a Player ID file management system using B-trees indexing is to efficiently store and retrieve player information while ensuring fast search, insertion, and deletion operations

We use of indexes to keep byte offsets for each record in the original file. The byte offsets allow us to find the beginning of each successive record and compute the length of each record. We look up the position of a record in the

INDEX table, and then seek to the record in the data file. Our choice of a record delimiter for the data files is the end-of-line (new-line) character (\n).

A flow of simple indexes on the primary key is used to provide direct access to data records. Each node in the consists of a primary key and reference pair of fields. The primary key field is the C_ID field while the reference field is the starting byte offset of the matching record in the data file.

1.2 Motivation and Scope

The motivation behind using a Cricket Score Sheet Maintainance with indexing lies in the need for efficient and scalable storage and retrieval of player information in a secure manner. B-trees are balanced tree data structures that provide fast search, insertion, and deletion operations. By utilizing B-trees for indexing, the system can organize the Cricket data files based on unique identifiers, such as cricket numbers, enabling quick access to specific records. Additionally, allow for efficient range queries, enabling the system to retrieve records within a certain range, such as players within a specific age group or residing in a particular district. This approach ensures that the score data file management system can handle a large volume of data while maintaining excellent performance, making it suitable for electoral authorities and organizations responsible for managing and verifying Cricket information.

1.3 Problem Statement

The problem statement for using a Cricket Sheet file management system using indexing is to efficiently store and retrieve players information while ensuring fast search, insertion, and deletion operations.

1.4 Limitations

One limitation of using a Cricket Sheet file management system with indexing is the potential for increased storage requirements. It uses additional space for index nodes, which can result in larger file sizes compared to simpler data structures.

Another limitation is the complexity and overhead associated with maintaining and updating the B-tree index. As the number of records in the system grows, the index needs to be frequently adjusted, which can impact system performance and resource utilization.

Additionally, B-trees indexing may not be suitable for scenarios where real-time updates to player information are critical, as the process of rebalancing the tree after each update can introduce delays in accessing the most up-to-date data.

Chapter 2

Requirements Specification

2.1 Functional Requirements

Creating a File

This requires the user to create a new file handle, use this file handle to enter its contents into a buffer and save the buffer contents to memory when done editing it.

Opening an Existing File

Here the user must open a valid file existing on disk open this file by creating a file handle copy file contents to main memory edit or delete its contents and use the same file handle contents and write the contents to disk.

Saving a File

When a user opens a file, through the help of a file handle the same file handle must successfully save the file contents on disk to the location specified by the user.

Writing a File

Once a user opens a file and decides that he has edited enough the program must save each and every character stored in the file buffer onto disk without faults of any sort.

Modify File Contents

If the user chooses to edit an existing file it should be opened and its content should be presented to the user as it is in the file it should also provide the user the choice either to overwrite the same file or save contents to a new file.

Finding keywords within Files

If the user wants to quickly to find specific keywords within a file, the program must provide the Find functionality and highlight keywords entered by user.

Replace

The program allows the users to enter keywords that are highlighted in the content of the file for each keyword, the user is given the choice to replace the given keyword with a replacement keyword.

2.2 Non-Functional Requirements

Performance

Performance of Quark should always vary between a few hundred milliseconds. Time taken to create files, open existing files, flushing contents of buffer onto disk when user hits save, find keywords within the file, replacing keywords within the file should be minimal.

Reliability

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

To provide a 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 Quark 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

Should 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
Control functions: The software must be very user-friendly and display appropriate error messages.
Dependencies: Requires JDK 8, Java SE and JFeonix Library.
Parallel operations: It must support many file operations simultaneously.

Safety/security considerations: The application must be exited always normally.

Chapter 3

System/Requirements Analysis

3.1 Overall System Description

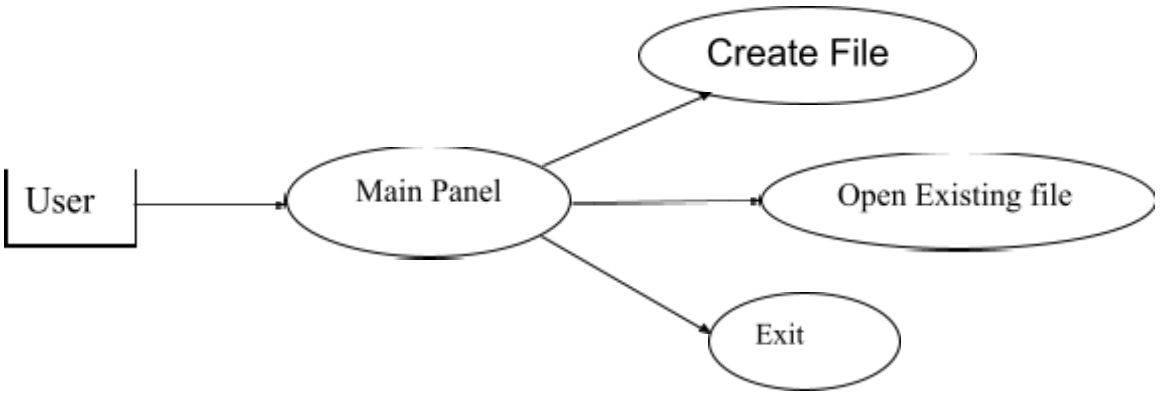


Fig. 3.1.1: Overall System Design

The overall description of the system is as follows:

The user is first presented with the Main Panel. Here user has a set of three options.

Create file: - In this Module User can create a file and write the content to the file. Next, he can perform basic file operations such as changing the content of the file, find and replace any selected word from the file and save the file.

Open Existing file: - In this Module the user opens the Existing file from the computer and can edit it, find and replace any selected word and save it.

Exit: - If the user wants to close the application, he can do so using the Exit Button.

3.2 Components/Subsystem Design

Create File Module

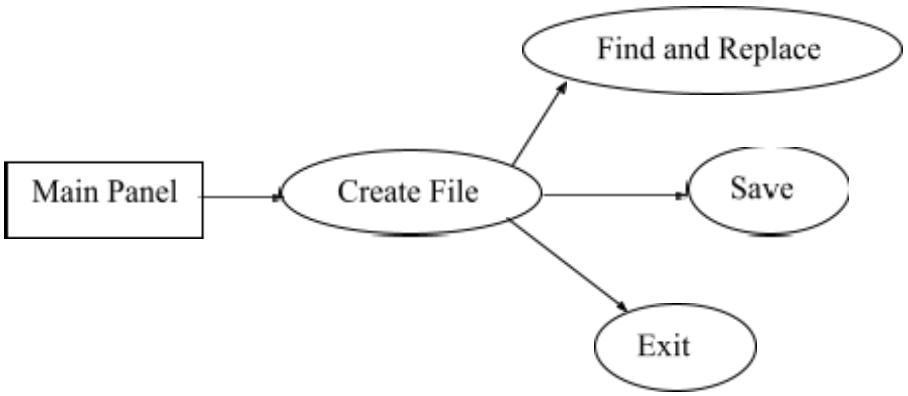


Fig. 3.2.1: Create File Flow Diagram

In the Create file module the user creates and writes the content into a new file. Next, he is provided with three options, Find and Replace, Save and Exit. If he needs to Find any word in the file, he selects the Find option furthermore he can replace it with any word he wants. After successful file operation, he can save the file by any name. Lastly, he can exit the window and discard contents by clicking the Exit button.

Open File Module

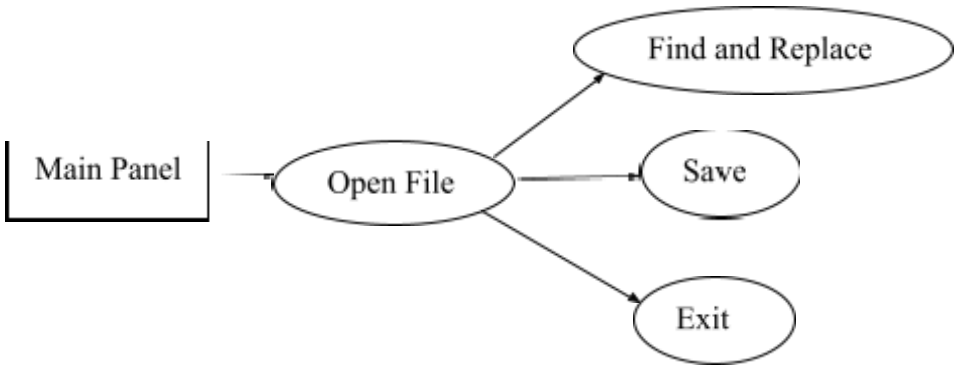


Fig. 3.2.2 Open File Flow Diagram

This module allows the user to open the Existing file from the computer. Next, he is provided with three options, Find and Replace, Save and Exit. If he needs to Find any word in the file, he selects

the Find option furthermore he can replace it with any word he wants. After successful file operation, he can save the file by any name. Lastly, he can exit the window and discard contents by clicking the Exit button.

Find and Replace Module

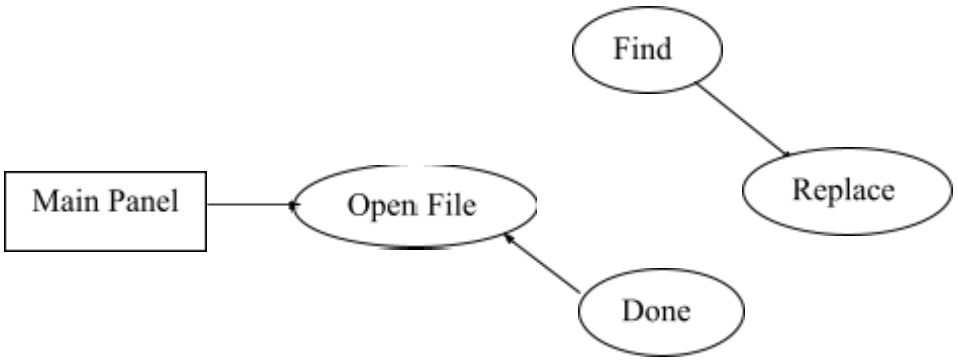


Fig. 3.2.3: Find and Replace Flow Diagram

In Find and Replace Module, initially the user opens the existing file or creates a new file for further file operations. If the user wants to find any word in the file, he can Search for it using Find Button and Replace it with any word using Replace Button. Once the Done button is pressed the system saves the buffer contents.

Cricket Score Sheet Maintainance is used in any leagues and associations to store Cricket details. The system is initially used to add the details with all its specifications entered correctly into a file corresponding to its Sheet data, which is different for each and every match. The system can be used to search, delete, modify or display existing records of all the players.

Structure used to store the Fields and Records

a) Storing Fields

Fixing the Length of Fields:

In the Cricket Score Sheet Maintainance, the cricket sheet field is a character array that can hold an integervalue of fixed size. Here other fields are character arrays, here the Cricket is the primary key. By using primary key the entire records can be easily obtained. We preserve the identity of fields by seperating them with delimiters. We have chosen the vertical bar character (), as the delimiter here.

b) Storing Records

Making Records a Predictable Number of Fields

In this system, we have a variable number of fields, each with a maximum length, that combine to make a data record. Varying the number of fields in a record does not imply that the size of fields in the record is fixed. The records are used as containers to hold a mix of fixed and variable-length fields within a record.

C) Using an Index to Keep Track of Addresses:

We use secondary indexes to keep byte offsets for each record in the original file. Secondary indexes can be built on any field of data file or on combination of fields. Secondary indexes will typically have multiple locations for a single key. Our choice of a record delimiter for the data files is the end-of-line (new-line) character (n).

OPERATIONS USED:

a) INSERTION: The system is initially used to add scores records containing the cricket attributes like sheet ID, name of the player, runs scored, average and strike rate. Records with duplicate ID fields are not allowed to be inserted. If we try insert a duplicate score data, it displays a message data is already present.

b) DISPLAY: The system can then be used to display existing records of all players. The records are displayed based on which we have entered the sheet details which is not in sorted order.

e) SEARCH: The system can then be used to search for existing records of all the players. If we select secondary indexing it asks for the player's name which is the secondary key in searching for the records in the secondary index files. The secondary key is searched to obtain the desired primary key of the player, which is then used to seek to the desired data record in any of the player files. The details of the requested record, if found, are displayed, with suitable headings on the user's screen. If absent, a "record not found message is displayed to the user

4) DELETE: The system can then be used to delete existing records from all player details. The reference to a deleted record is removed from index file and $ is placed in data file for a particular record which is deleted. The requested record, if found, is marked for deletion, a "record deleted message is displayed. If absent, a "record not found" message is displayed to the user

e) MODIFY: If selected for modify option, the System will ask to enter the particular sheet ID which is to be modified and thereby after entering the data, the corresponding details of the player are also displayed and the option to modify any of the field excluding the primary key which is sheet data here is asked to enter. The user can give all related information and then press enter. And the updated or modified values will reflect back into the file.

Indexing Used

Simple indexes on the primary key is used to provide direct access to data records. Each node in the consists of a primary key with the reference to record. The primary key field is the cricket field while the reference field is the starting byte offset of the matching record in the data file. Each node can have max of 2 child node.

The sheet red in the indexing, and hence written to an index file. On retrieval the matched is used, before it is used to seek to a data record, as in the case of requests for a search, delete, modify operation.

As records are added, nodes of the undergo splitting (on detection of overflow), merging (on detection of underflow) or redistribution (to improve storage utilization), in order to maintain a balanced tree structure.

The data files are entry-sequenced, that is, records occur in the order they are entered into the file. The contents of the index files are loaded into their respective. prior to use of the system, each time. Each is updated as requests for the available operations are performed. Finally, the are written back to theirindex files, after every insert, delete and modify operation.

Chapter 4

System Design

This is a self-balancing tree data structure that maintains sorted data and allows efficient operations such as insertion, deletion, and searching. In the context of a Player ID management system, a can be used to store and manage the player ID information efficiently. Here's a high-level system design for a player ID management system using indexing:

Data Structure:

Each node represents a range of player IDs.
Each node contains a list of keys that represent the minimum value of the range for that node.
Each key in a node corresponds to a child node.
Each leaf node contains the actual player ID records within its
range Parameters:

Order: Determine the maximum number of keys in each node. It affects the maximum number of child nodes as well.
Balance Factor: Determine the minimum number of keys in each node to maintain balance. Usually, it is set to half the order.
B-tree Operations:

Search: Start from the root node and recursively search for the desired player ID by comparing the keys in each node. Once a leaf node is reached, search within the leaf node for the specific player ID.
Insertion: Start from the root node and recursively find the appropriate leaf node for insertion based on the player ID range. If the leaf node is full, split it into two nodes and promote the median key to the parent node. Repeat this process until a suitable leaf node is found for insertion.
Deletion: Start from the root node and recursively find the appropriate leaf node for deletion. If the leaf node has more than the minimum required keys after deletion, simply remove the player ID record. Otherwise, if the neighboring sibling node has more than the minimum required keys, perform key redistribution. If both the leaf node and its sibling have the minimum required keys, merge them into a single node.
Storage:

Each node in the B-tree can be stored as a separate data record in a database or a file system. The player ID records within the leaf nodes can be stored in a separate database table or file.
Additional Features:

Indexing: Maintain an index of other attributes related to player ID, such as player's name, score,etc., to support efficient searching and retrieval based on those attributes.
Concurrency Control: Implement appropriate mechanisms like locks or multi-version concurrency control to handle concurrent read and write operations on the sheet data.

Chapter 5

Implementation

5.1 Description of Frameworks Used

C Developer Interface

C interface is a GUI (Graphical User Interface) framework provided by c language for creating desktop applications. It provides a set of classes and components that allow developers to build interactive and visually appealing user interfaces. Here's a description of the framework in the context of a cricket score sheet maintainance:

1. Containers and Components:
    - Containers: C provides various container classes such as Function, SGLIB, and JPanel that act as the main windows or containers for holding other components.
    - Components: Swing offers a wide range of components such as STL libraries, templates, etc., to build the user interface. These components are used to display and interact with data.
2. Layout Managers:
    - language provides different layout managers that control the positioning and resizing of components within a container. Examples include BorderLayout, FlowLayout, GridLayout, and GroupLayout. Layout managers ensure that the components are arranged in an organized manner and adapt to different window sizes.
3. Event Handling:
    - supports event-driven programming. It uses listeners and counters to capture user interactions such as button clicks, mouse movements, etc. Developers can register event listeners and write corresponding event handler methods to respond to these events.
4. Customization and Styling:
    - Components can be customized and styled to fit the application's visual requirements. Properties like font, color, size, and border can be modified to achieve

the desired look and feel. C design also supports the use of custom icons, images, andbackgrounds.

5. Internationalization and Localization:

   - Swing provides support for internationalization (i18n) and localization (l10n) by allowing the application's user interface to be translated into different languages. It provides mechanisms for managing resource bundles that contain localized strings and formatting patterns.

6. Multithreading:

   - Applications often require performing time-consuming tasks without blocking the user interface. Class that allows background tasks to be executed on a separate thread while keeping the UI responsive. It provides methods for handling task progress and updating UI components safely.

7. Data Presentation:

   - Tabular assignment for displaying tabular and list-based data. These components can be customized to present player information in a structured and organized manner. Developers can define models, renderers, and editors to control the data display and interaction.

8. Look and Feel:

   - Applications can adopt different look and feel styles, including the system default, cross-platform, or custom-designed look and feel. The look and feel can be set programmatically, allowing the application to match the desired visual style.

9. Accessibility:

   - It incorporates accessibility features to ensure that the application can be used by individuals with disabilities. It provides support for assistive technologies and follows accessibility guidelines, allowing developers to create applications that are accessible to a wider range of users.

Chapter 6

Testing

6.1 Component Test

Main Panel

TEST UNIT	TEST CASE	RESULT
Create File Panel	Create File button is pressed	The system invokes the respective function and displays the window to create a file.
Open File Panel	Open Existing File Button is pressed.	The system invokes the respective function and displays the window to open a file.
Exit Program	Exit button is pressed.	The system exits normally.

Fig 6.1.1: Main Panel Tests

TEST UNIT	TEST CASE	RESULT
Find Panel	Find button is pressed	The system invokes the respective function and displays the window to find a keyword.
Save Panel	Save Button is pressed.	The system invokes the respective function and displays the window to save a file.
Cancel Program	Cancel button is pressed.	The system returns to the Main Panel.

Create Panel

Fig 6.1.2: Create File Module Tests

Open File Panel

TEST UNIT	TEST CASE	RESULT
Find Panel	Find button is pressed	The system invokes the respective function and displays the window to find a keyword.
Save Panel	Save Button is pressed.	The system invokes the respective function and displays the window to save a file.
Cancel Program	Cancel button is pressed.	The system returns to the Main Panel.

Fig 6.1.3: Open File Module Tests

TEST UNIT	TEST CASE	RESULT
Find Function	Find button is clicked	The system invokes the respective function and highlights the keyword entered by user
Replace Function	Replace Button is clicked.	The system invokes the respective function and replaces the highlighted keyword.
Save Changes	Done button is clicked.	The system returns to the File Editing Panel.

Find and Replace Panel

Fig 6.1.4: Find and Replace Module Tests

6.2 System Testing

TEST UNIT	TEST CASE	RESULT
Creating a File	Click on Create File and Save	Creates and saves a new file onto disk with little or no content.
Opening a File	Click on Open File	Opens a previously created file from Disk.
Modifying File Content	Click and open file edit the content	Modifies Files Content
Finding Keywords	Click on Find Button is editor Window	Highlights the corresponding keyword
Replace	Click on Replace button in Editor Window	Replaces the keyword specified by user.
Close	Click on Exit, Cancel button	Closes the Program/Current Panel

Fig 6.2.1: Complete System Tests

Chapter 7

Interpretation of Results

Fig 7.1 Menu

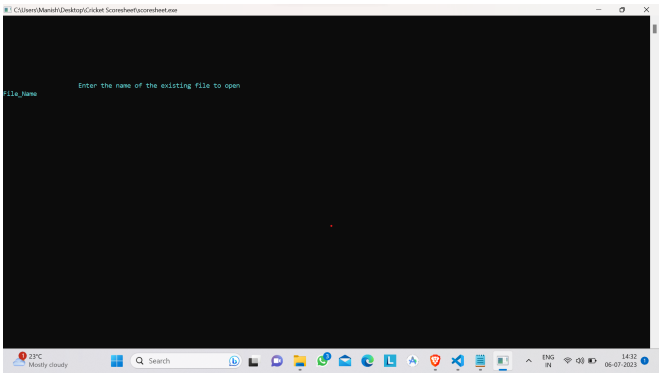
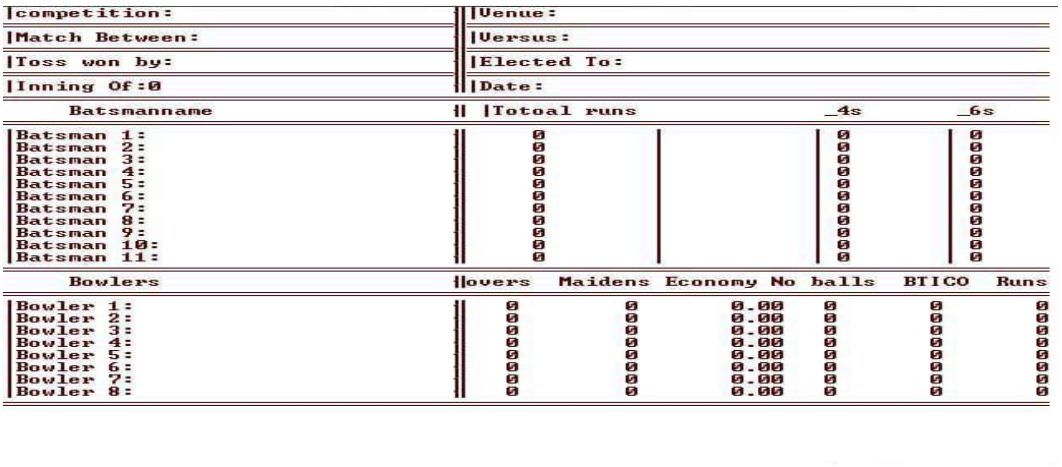


Fig 7.2 Search



competition:		Venue:				
Match Between:		Versus:				
Toss won by:		Elected To:				
Inning Of:0		Date:				
Batsmanname		Totoal runs			_4s	_6s
Batsman 1:		0			0	0
Batsman 2:		0			0	0
Batsman 3:		0			0	0
Batsman 4:		0			0	0
Batsman 5:		0			0	0
Batsman 6:		0			0	0
Batsman 7:		0			0	0
Batsman 8:		0			0	0
Batsman 9:		0			0	0
Batsman 10:		0			0	0
Batsman 11:		0			0	0

Bowlers	overs	Maidens	Economy No	balls	BTICO	Runs
Bowler 1:	0	0	0.00	0	0	0
Bowler 2:	0	0	0.00	0	0	0
Bowler 3:	0	0	0.00	0	0	0
Bowler 4:	0	0	0.00	0	0	0
Bowler 5:	0	0	0.00	0	0	0
Bowler 6:	0	0	0.00	0	0	0
Bowler 7:	0	0	0.00	0	0	0
Bowler 8:	0	0	0.00	0	0	0

Fig 7.3 View

Conclusion

In conclusion, the design and implementation of a Score Sheet Maintainance are crucial for ensuring the efficient management and organization of player information. The system should be able to handle tasks such as storing player IDs, searching for specific IDs, and managing updates anddeletions. Here are the key points to consider:

1. System Design: The use of a backend structure is recommended for efficient storage and retrieval of player and cricket IDs. It provides balanced searching and support operations likeinsertion and deletion while maintaining a sorted order of data.
2. User Interface: The C language framework can be utilized to create a user-friendly interface for the Cricket Score Sheet Maintainance. Swing offers a wide range of components and layouts to design an interactive and visually appealing interface.
3. Functionality: The system should support operations such as searching for sheet IDs, adding new IDs, updating players information, and removing obsolete or duplicate IDs. It should also provide error handling and validation mechanisms to ensure data integrity.
4. Security: The system should incorporate security measures to protect Player ID information. This includes implementing authentication and authorization mechanisms to restrict access, encrypting sensitive data, and following best practices to prevent unauthorized manipulation or disclosure of player information.
5. Performance: The system's performance is crucial, especially during peak times such as elections. The data structure offers efficient search and retrieval operations, ensuring quick access to Player IDs.
6. Scalability: The system should be designed to handle a growing number of Player IDs and support concurrent access by multiple users.
7. Compliance: The system should adhere to relevant laws and regulations regarding the management and privacy of player information. Compliance with data protection laws and regulations ensures the privacy and security of players data.

References

- Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object-OrientedApproach with C++, 3rd Edition, Pearson Education, 1998.
- Herbert Schildt: C the Complete Reference, 7th/9th Edition, Tata McGraw Hill, 2007.
- Jim Keogh: J2EE-TheCompleteReference, McGraw Hill, 2007.
- StackOverflow: www.stackoverflow.com
- Codeproject: www.codeproject.com
- Javacreek: www.javacreek.com