**Final Report (CA-2)**

Title: Leaderboard Manager – Java Swing Based Competitive Scoring System

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by:**

**Name:** Yukesh Kumar Dillibabu

**Registration number:** 12313569



**Submitted To: Dr. Preetjot Kaur**

**Lovely Professional University**

**Jalandhar, Punjab, India.**

**Technologies Used**

* **Java (JDK 17+)**
* **Java Swing** – GUI development
* **Collections Framework** – HashMap, ArrayList, LinkedList
* **File I/O** – For exporting leaderboard to CSV
* **Sorting Algorithms** – Custom comparator logic
* **OOP Principles** – Encapsulation, Modularization, MVC hints

**Project Overview**

The **Leaderboard Manager** is a robust, desktop-based Java Swing application designed to manage, update, and display scores for participants in competitive environments like coding contests, quizzes, e-sports tournaments, or classroom assessments. This project combines GUI development, core Java programming, and essential data structures to simulate real-time leaderboard systems seen on platforms like LeetCode, Codeforces, or gaming platforms.

Users can **add, update, search, sort, export**, and **auto-refresh** participant rankings using a clean, responsive interface, simulating real-world competition scoreboards.

**Objectives:**

* **To implement a real-time, GUI-based leaderboard that supports score updates and dynamic sorting.**
* **To integrate core Data Structures and Algorithms (DSA) such as HashMaps, sorting algorithms, and priority lists.**
* **To design a user-friendly interface using Java Swing with responsive and interactive elements.**
* **To demonstrate file handling (CSV export), event-driven programming, and GUI layout management.**

**Core Functionalities**

1. **Participant Score Entry:**
   * Add a new participant with name and score.
   * If participant exists, score gets updated (with new timestamp).
2. **Leaderboard Display:**
   * Displays participants in descending order of scores.
   * If scores are tied, earlier submission gets higher rank.
3. **Top-K Filtering:**
   * Displays top K participants based on current leaderboard.
4. **Search Function:**
   * Retrieve full details of a participant using their name.
5. **Export to CSV:**
   * Download the entire leaderboard in .csv format with name, score, and timestamp.
6. **Reset Leaderboard:**
   * Wipes the data for fresh competition tracking.
7. **Auto Refresh Feature:**
   * Leaderboard updates automatically every 10 seconds.

**Design & Architecture**

The application is modular, with two core classes:

* **Participant:** Holds details like name, score, and submission time.
* **Leaderboard:** Manages logic for sorting, storing, and retrieving participants.

The UI follows a minimal yet effective structure using:

* JTextField, JButton, JTextArea, JScrollPane
* Grid & Flow Layouts for positioning
* Timer for periodic auto-refresh

**Algorithms & Logic Used**

| **Concept** | **Application** |
| --- | --- |
| HashMap | For storing participant records with O(1) access |
| Custom Sort Comparator | Sorts by score, then timestamp |
| Collections.sort() | Used to reorder participants dynamically |
| FileWriter | Writing structured leaderboard data to CSV |
| Auto-refresh timer | Scheduled GUI updates using Swing's javax.swing.Timer |

**🔹 1. Frequency Update Algorithm**

* **Used in: addParticipant(String name, int score)**
* **What it does: Updates participant score and submission time.**
* **Type: Hashing + Conditional Update**
* **Time Complexity: O(1)**
* **Why it matters: It's similar to frequency counting — a very common problem in competitive coding and interviews.**

**🔹 2. Custom Sorting Algorithm**

* **Used in: getSortedLeaderboard()**
* **What it does: Sorts participants based on:**
  + **Score (descending)**
  + **Submission time (ascending, if scores are equal)**
* **Implementation: Uses Collections.sort() with a custom comparator**
* **Type: Comparison-based Sorting (uses TimSort internally in Java)**
* **Time Complexity: O(n log n)**
* **Why it matters: This logic is algorithmic — sorting based on multiple conditions.**

**🔹 3. Top-K Selection Algorithm (Partial Sort)**

* **Used in: getTopK(int k)**
* **What it does: After sorting the leaderboard, returns only the top k participants.**
* **Type: Sublist extraction after sort**
* **Time Complexity: O(k)**
* **Alternative: You could also implement a Min-Heap for better performance with very large data.**

**🔹 4. File Export Algorithm**

* **Used in: exportToCSV(String filename)**
* **What it does: Iterates the sorted list and writes formatted strings to a file.**
* **Type: Simple iteration + string formatting**
* **Time Complexity: O(n)**

**Real-World Applications:**

* **Coding Platforms: Similar to how Codeforces or LeetCode shows real-time contest leaderboards.**
* **Hackathons & Quizzes: For showing participant standings in classroom or live competitions.**
* **Gaming Tournaments: For real-time rank tracking in events like e-sports matches.**
* **Corporate Training Portals: To motivate learners using score-based leaderboards.**
* **Education Portals: In schools/colleges to display rankings of students across assessments.**

**Sample Use Case:**

* A classroom or coding bootcamp wants to keep real-time track of students solving problems.
* The trainer inputs names and scores after each round.
* At any time, the leaderboard can be viewed, filtered, searched, or exported for records.

**Scenario:** A campus coding event with 50+ participants where each user’s solution is evaluated and scored in real-time.

* Judges input scores using the GUI.
* The system automatically sorts and displays ranks.
* Top 10 participants are shown instantly.
* Leaderboard is exported for certification purposes.

**Learning Outcomes:**

* **Java Collections Mastery** – Efficient use of HashMaps, Lists, and Iterators.
* **GUI Event Handling** – Hands-on practice with ActionListeners and real-time UI updates.
* **Algorithmic Thinking** – Built-in logic to handle ties, sorting, and edge cases.
* **Clean Code Design** – Modular functions and encapsulated OOP principles.
* **Practical Exposure** – How competitive programming sites manage scores internally.

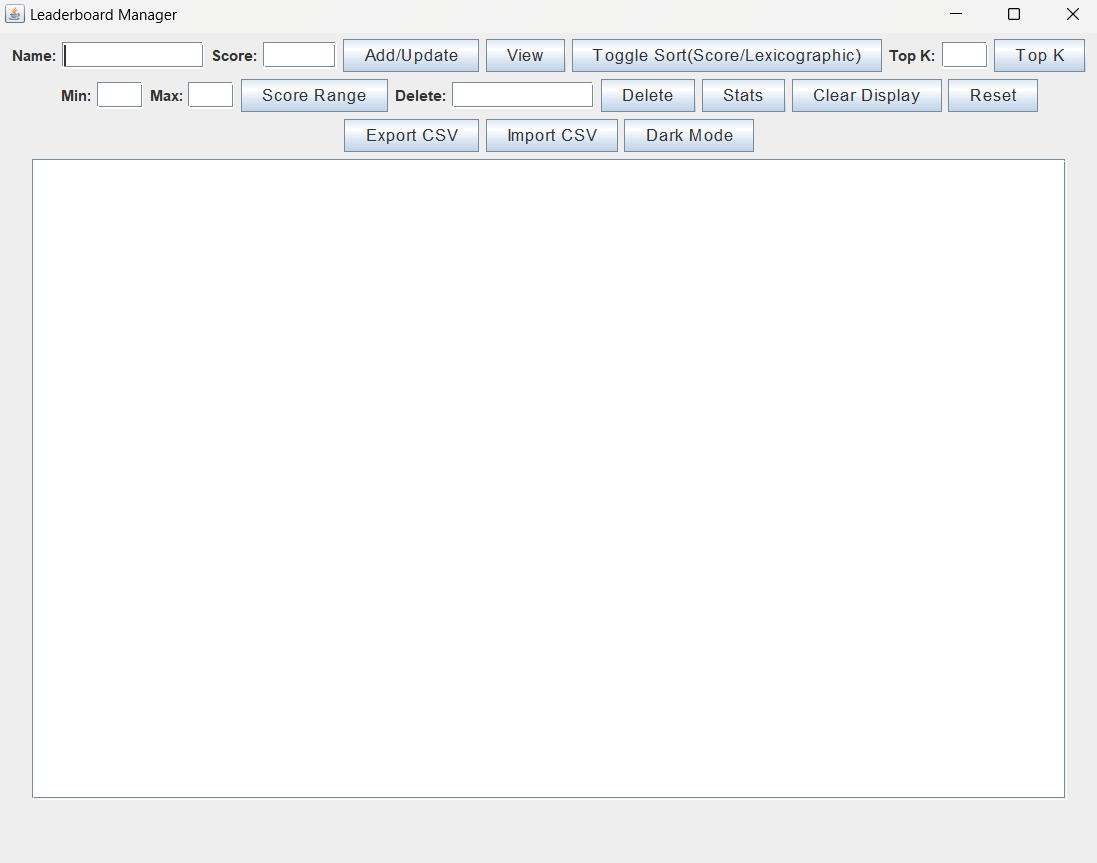
**Challenges Faced:**

* Handling tied scores with different timestamps accurately.
* Ensuring data consistency during real-time updates.
* Designing a user-friendly UI with limited screen space.
* Exporting to CSV without third-party libraries.

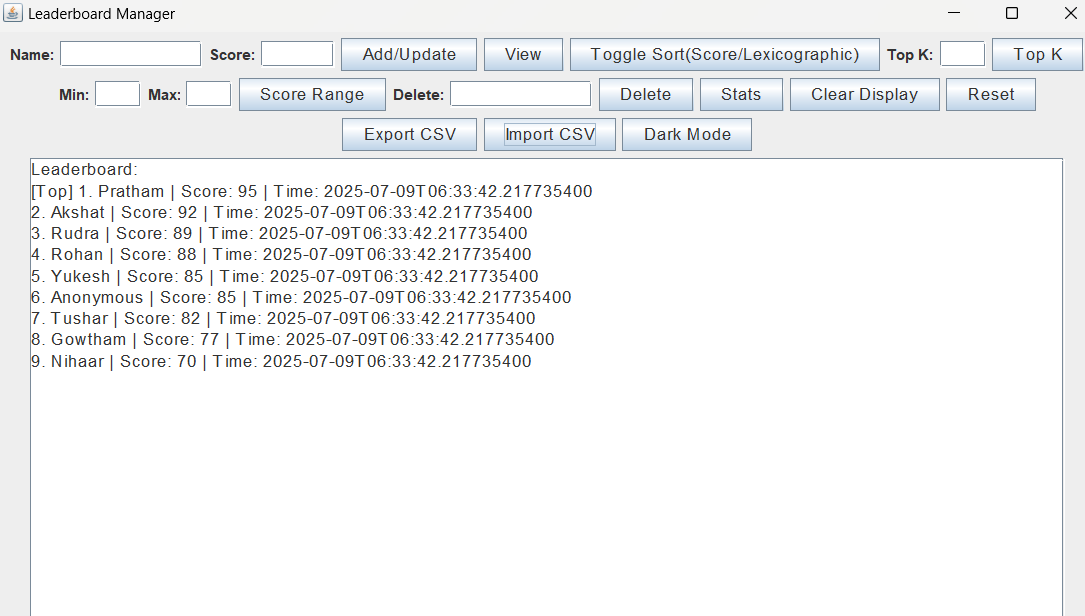
**Future Enhancements:**

* **Add persistent storage (like SQLite or JSON-based saving).**
* **User authentication and login roles (admin/viewer).**
* **Real-time syncing across multiple devices using socket programming.**
* **Export to Excel and PDF formats.**
* **Dark mode or UI customization features.**

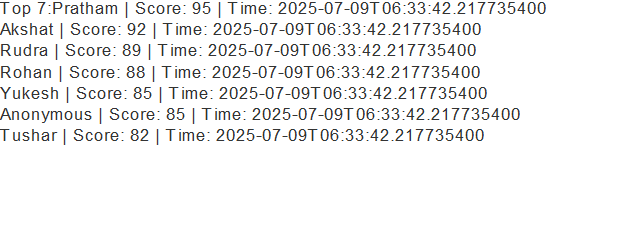
**Photo Snippets of My Project:**

****

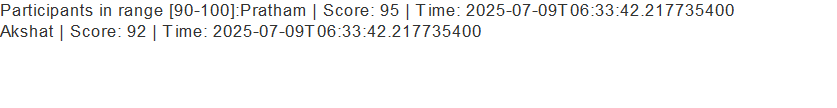
**Fig 1. Leaderboard Manager Dashboard**

****

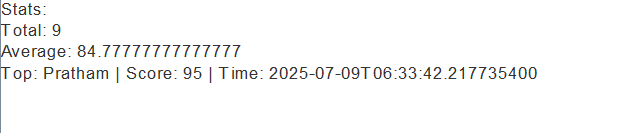
**Fig 2. After Importing .csv file already present in the same folder as .java main file,refreshes automatically every 10 secs**

****

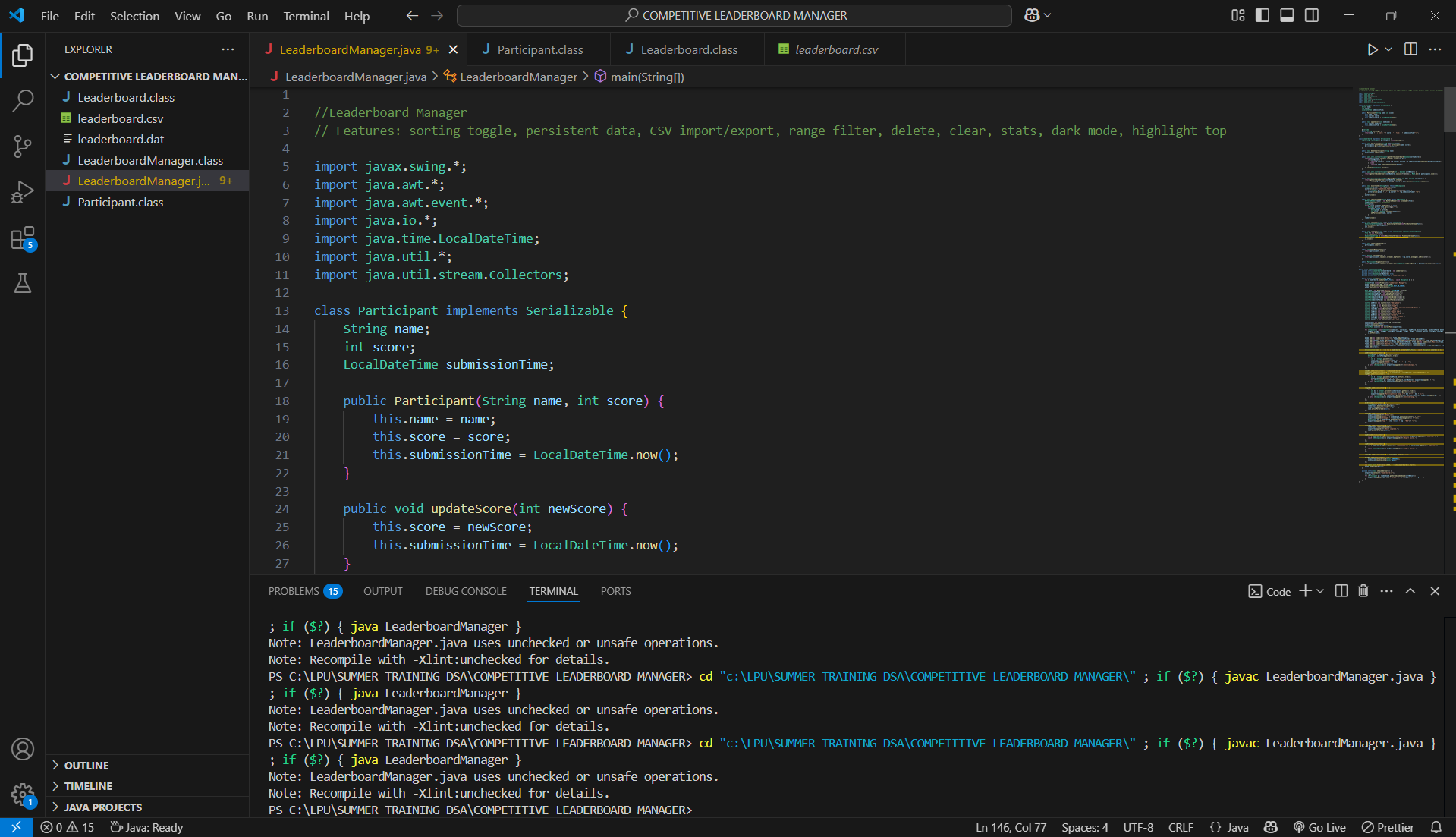
**Fig 3. Listing Top K List of Competitors Name,Score and Time of Submission(can sort Scorewise/Lexicographic order)**

****

**Fig 4. Listing Competitors between min and max Scores as requested by admin/organizer**

****

**Fig 5. Listing Total Number of Students Participated, Average and Top performer**

****

**Fig 6. Display of code files used, .dat file for serialization, and .csv file for Exporting Competitor details**

**Conclusion**

The Leaderboard Manager project is a robust implementation that bridges fundamental DSA logic and GUI programming. It highlights how core Java skills can solve real problems efficiently, while providing a clean, intuitive interface for the end user.

This project is a strong demonstration of:

* Algorithmic thinking
* Real-world system modeling
* UI/UX design using Java Swing
* File handling and data persistence
* Modular, maintainable Java code

**Programming & API References:**

1. **Java SE Documentation – Oracle**  
   <https://docs.oracle.com/en/java/javase/17/>  
   *(Official documentation for Java language syntax, collections, file I/O, and Swing framework.)*
2. **Java Swing Tutorial – Oracle & GeeksforGeeks**  
   https://www.geeksforgeeks.org/java-swing/  
   *(Helpful for learning GUI layout management, event handling, and component integration.)*
3. **HashMap Documentation – Java Collections Framework**  
   <https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>
4. **LocalDateTime API – Java Time Package**  
   <https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html>
5. **Java FileWriter for CSV Export**  
   https://www.baeldung.com/java-csv  
   *(For implementing file export and saving data as comma-separated values.)*

**Data Structures & Algorithm References:**

1. **Data Structures and Algorithms in Java – Narasimha Karumanchi**  
   *(Used for DSA concepts applied in sorting, HashMap usage, and time complexity.)*
2. **Introduction to Algorithms – Cormen et al. (CLRS)**  
   *(Used as a general guide for sorting and data structure logic.)*

**Real-World Application Inspirations:**

1. **LeetCode & Codeforces Contest Leaderboards**  
   https://leetcode.com/contest/  
   https://codeforces.com/contests  
   *(Used for understanding how real coding platforms rank participants in real-time.)*
2. **TopCoder Scoreboard System**  
   <https://www.topcoder.com/>  
   *(Provided insights into contest evaluation and participant rank tracking.)*

**Educational & Conceptual Sources:**

1. **GeeksforGeeks – Java Collections and Comparators**  
   https://www.geeksforgeeks.org/sorting-a-hashmap-according-to-values/  
   *(Helped in implementing the sorted leaderboard feature.)*
2. **Stack Overflow Discussions**  
   <https://stackoverflow.com/>  
   *(Community support for solving specific bugs and design doubts.)*
3. **JavaTpoint – Java Timer and GUI Examples**  
   <https://www.javatpoint.com/java-timer>

**Github Link🔗**

<https://github.com/YukeshKumar2005/Competitive-Leaderboard-Manager>