CONTEST MANAGEMENT SYSTEM

INDEX

INTRODUCTION

The "Contest Management System" project is designed to streamline and optimize the processes involved in managing contests. Leveraging Object-Oriented Programming (OOP) concepts in C++, the project addresses various aspects, including security passwords, calculating total scores, maintaining a comprehensive database for each member of the organizers' team, the administrator team, and contestants for each contest. It also solves the problem of creating a leaderboard, declaring winners and allocating payrolls to each employee based on the total budget.

The project aims to enhance efficiency, accuracy, and transparency in contest management operations, offering a user-friendly interface for organizers, administrators, and contestants. By incorporating OOP principles, it provides a modular and maintainable solution to address the complex requirements of contest management, making it an ideal tool for various competition scenarios.

KEY FEATURES

Key features of the project include:

Score Calculation:
Utilizes OOP principles to efficiently calculate and manage total scores for contestants based on predefined criteria.

Database Management:
Implements a robust database structure for organizers, administrators, and contestants, ensuring organized and accessible information.

Leaderboard Display:
Utilizes OOP to generate a dynamic leaderboard, providing real-time updates on contestant rankings throughout the contest.

Winner Declaration:
Implements algorithms to determine winners based on scores, ensuring accuracy and fairness in the outcome.

Payroll Allocation:
Utilizes OOP for efficient payroll management, allocating compensation to each volunteer based on their role and contribution.

Security:
Uses password protection to offer security features over the employee and contestants database.

TECHNOLOGIES USED

The following OOPS concepts were used in C++ -

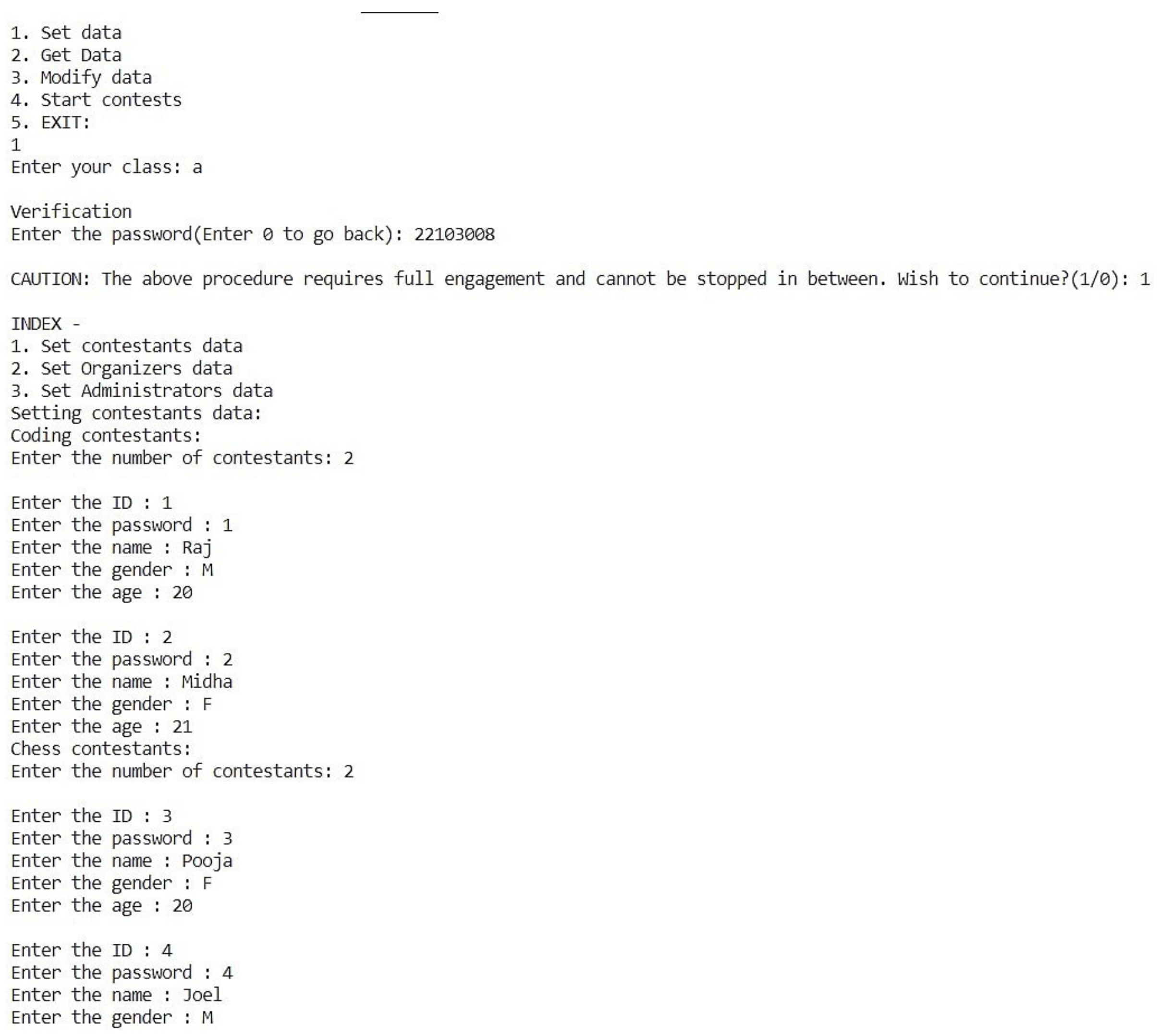
1. Classes and objects
2. Data abstraction and encapsulation using Constructors and Destructors. Includes data hiding using public, private and protected.
3. Static members and friend function.
4. Runtime polymorphism using virtual functions, Pure virtual functions and abstract classes.
5. Dynamic memory allocation using new operator
6. Pointer to an object, array of objects
7. Inheritance
8. Code genericity using Templates.
9. Functions of file handling.

Libraries imported -

1. Iostream
2. Fstream
3. Algorithm
4. Vector
5. Cstring

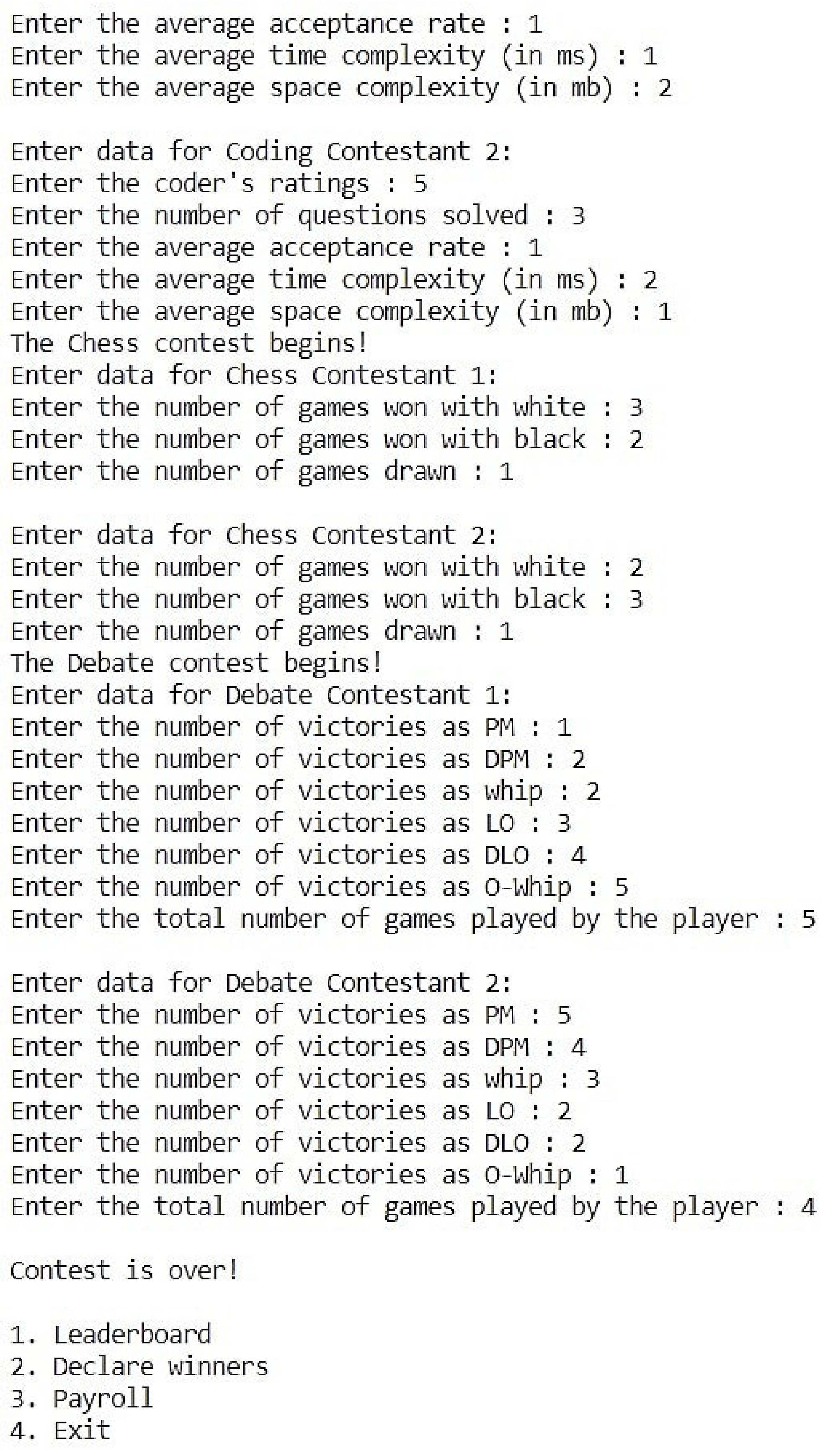
Apart from these, data structures like vectors, arrays and sorting algorithms like bubble sort were also used.

SAMPLE CODE RUN OUTPUT

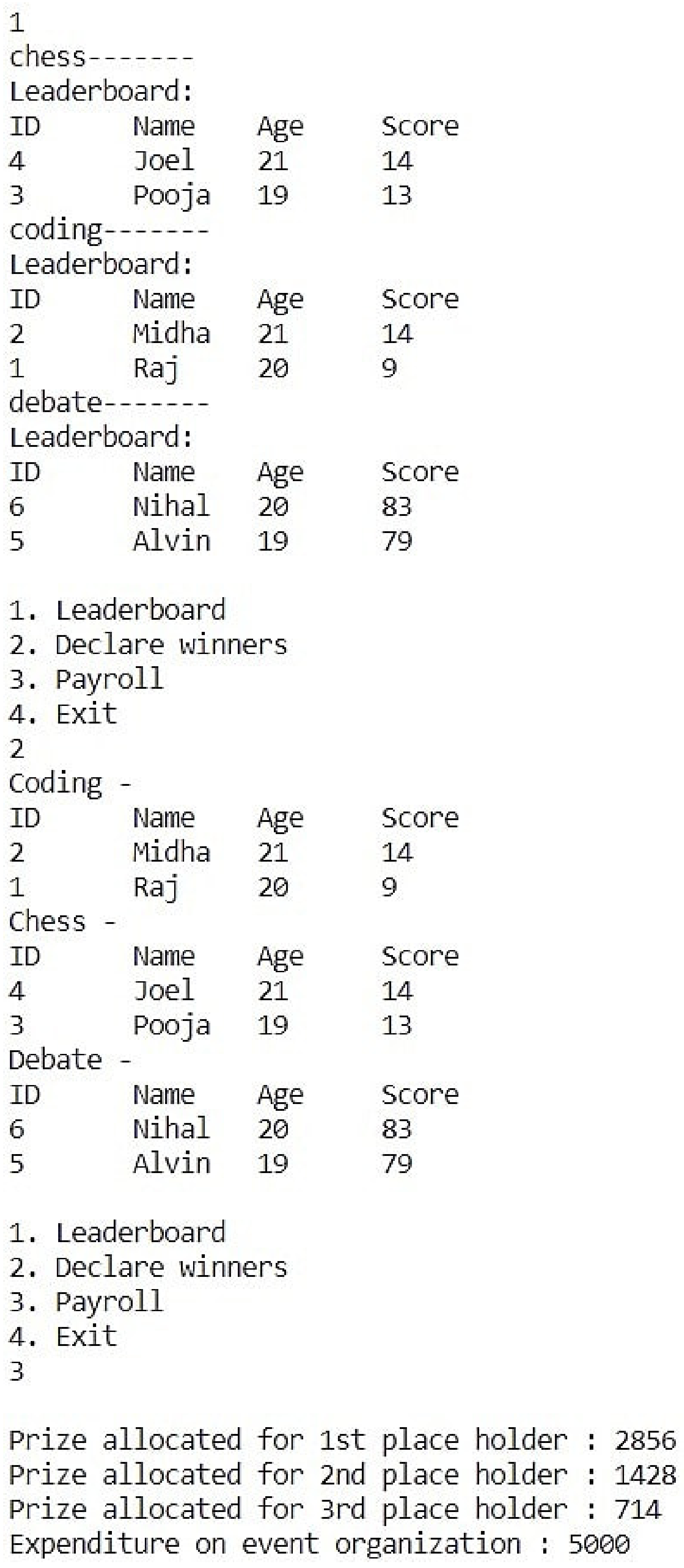


```
1. Set data
2. Get Data
3. Modify data
4. Start contests
5. EXIT:
1
Enter your class: a

Verification
Enter the password(Enter 0 to go back): 22103008

CAUTION: The above procedure requires full engagement and cannot be stopped in between. Wish to continue?(1/0): 1

INDEX -
1. Set contestants data
2. Set Organizers data
3. Set Administrators data
Setting contestants data:
Coding contestants:
Enter the number of contestants: 2

Enter the ID : 1
Enter the password : 1
Enter the name : Raj
Enter the gender : M
Enter the age : 20

Enter the ID : 2
Enter the password : 2
Enter the name : Midha
Enter the gender : F
Enter the age : 21
Chess contestants:
Enter the number of contestants: 2

Enter the ID : 3
Enter the password : 3
Enter the name : Pooja
Enter the gender : F
Enter the age : 20

Enter the ID : 4
Enter the password : 4
Enter the name : Joel
Enter the gender : M
Enter the age : 21
Debate contestants:
Enter the number of contestants: 2

Enter the ID : 5
Enter the password : 5
Enter the name : Alvin
Enter the gender : M
Enter the age : 19

Enter the ID : 6
Enter the password : 6
Enter the name : Nihal
Enter the gender : M
Enter the age : 20
Setting organizers data:
Finance team:
Enter the number of finance team members: 2

Enter the ID : 7
Enter the password : 7
Enter the name : Divin
Enter the gender : M

Enter the ID : 8
Enter the password : 8
Enter the name : Sara
Enter the gender : F

Management team:
Enter the number of management team members: 2

Enter the ID : 9
Enter the password : 9
Enter the name : Nihad
Enter the gender : M

Enter the ID : 10
Enter the password : 10
Enter the name : Rishan
Enter the gender : M

Decoration team:
Enter the number of decoration team members: 2
Enter the ID : 11
Enter the password : 11
Enter the name : Wasil
Enter the gender : M

Enter the ID : 12
Enter the password : 12
Enter the name : Helen
Enter the gender : F

Setting administrators data:
B class administrators data:
Enter the number of B-class administrators: 2

Enter the ID : 13
Enter the password : 13
Enter the name : Milen
Enter the gender : F

Enter the ID : 14
Enter the password : 14
Enter the name : Rahul
Enter the gender : M

Data added successfully!

1. Set data
2. Get Data
3. Modify data
4. Start contests
5. EXIT:
2

Coding Contestant 1 Data:
1       Raj     20
Coding Contestant 2 Data:
2       Midha   21
Chess Contestant 1 Data:
3       Pooja   20
Chess Contestant 2 Data:
4       Joel    21
Debate Contestant 1 Data:
5       Alvin   19
Debate Contestant 2 Data:
```

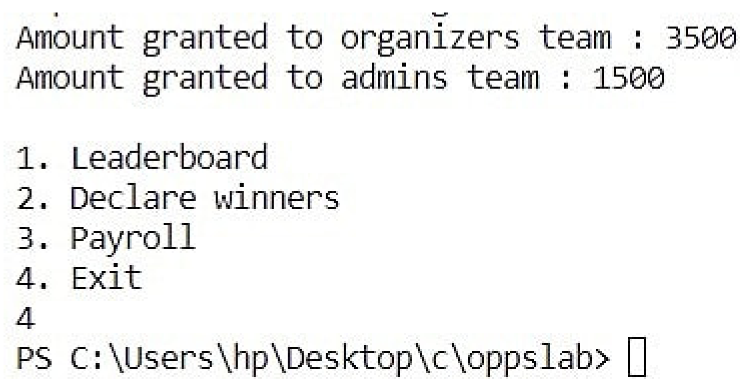
```
Debate Contestant 2 Data:
6       Nihal   20
Finance Team Data:
ID: 7, Name: Divin, Gender: M
ID: 8, Name: Sara, Gender: F

Management Team Data:
ID: 9, Name: Nihad, Gender: M
ID: 10, Name: Rishan, Gender: M

decoration Team Data:
ID: 11, Name: Wasil, Gender: M
ID: 12, Name: Helen, Gender: F

admin Team Data:
ID: 13, Name: Milen, Gender: F
ID: 14, Name: Rahul, Gender: M

1. Set data
2. Get Data
3. Modify data
4. Start contests
5. EXIT:
3

Search in -
1.Debating contestants database
2.Coding contestants database
3.Chess contestants database : 3
chess ------->>>>>>>
Enter the ID of the contestant you want to modify: 3

Enter new age for contestant 3: 19
Contestant data modified and saved to file.

1. Set data
2. Get Data
3. Modify data
4. Start contests
5. EXIT:
4
The Coding contest begins!
Enter data for Coding Contestant 1:
Enter the coder's ratings : 5
Enter the number of questions solved : 2
```



```
Enter the average acceptance rate : 1
Enter the average time complexity (in ms) : 1
Enter the average space complexity (in mb) : 2

Enter data for Coding Contestant 2:
Enter the coder's ratings : 5
Enter the number of questions solved : 3
Enter the average acceptance rate : 1
Enter the average time complexity (in ms) : 2
Enter the average space complexity (in mb) : 1
The Chess contest begins!
Enter data for Chess Contestant 1:
Enter the number of games won with white : 3
Enter the number of games won with black : 2
Enter the number of games drawn : 1

Enter data for Chess Contestant 2:
Enter the number of games won with white : 2
Enter the number of games won with black : 3
Enter the number of games drawn : 1
The Debate contest begins!
Enter data for Debate Contestant 1:
Enter the number of victories as PM : 1
Enter the number of victories as DPM : 2
Enter the number of victories as whip : 2
Enter the number of victories as LO : 3
Enter the number of victories as DLO : 4
Enter the number of victories as O-Whip : 5
Enter the total number of games played by the player : 5

Enter data for Debate Contestant 2:
Enter the number of victories as PM : 5
Enter the number of victories as DPM : 4
Enter the number of victories as whip : 3
Enter the number of victories as LO : 2
Enter the number of victories as DLO : 2
Enter the number of victories as O-Whip : 1
Enter the total number of games played by the player : 4

Contest is over!

1. Leaderboard
2. Declare winners
3. Payroll
4. Exit
```

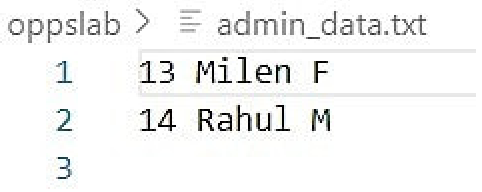


```
1
chess-------
Leaderboard:
ID      Name    Age     Score
4       Joel    21      14
3       Pooja   19      13
coding-------
Leaderboard:
ID      Name    Age     Score
2       Midha   21      14
1       Raj     20      9
debate-------
Leaderboard:
ID      Name    Age     Score
6       Nihal   20      83
5       Alvin   19      79

1. Leaderboard
2. Declare winners
3. Payroll
4. Exit
2
Coding -
ID      Name    Age     Score
2       Midha   21      14
1       Raj     20      9
Chess -
ID      Name    Age     Score
4       Joel    21      14
3       Pooja   19      13
Debate -
ID      Name    Age     Score
6       Nihal   20      83
5       Alvin   19      79

1. Leaderboard
2. Declare winners
3. Payroll
4. Exit
3

Prize allocated for 1st place holder : 2856
Prize allocated for 2nd place holder : 1428
Prize allocated for 3rd place holder : 714
Expenditure on event organization : 5000
```



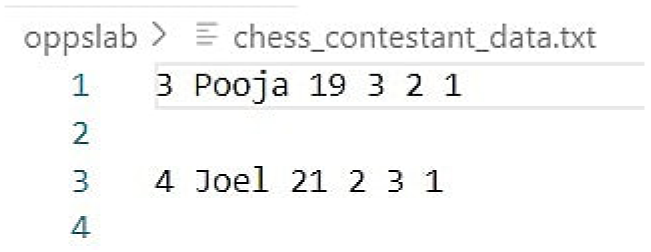
```
Amount granted to organizers team : 3500
Amount granted to admins team : 1500

1. Leaderboard
2. Declare winners
3. Payroll
4. Exit
4
PS C:\Users\hp\Desktop\c\oppslab>
```

admin_data.txt



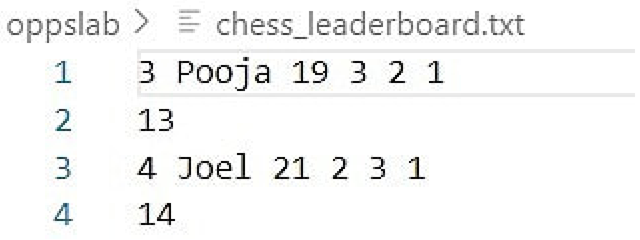
```
13 Milen F
14 Rahul M

```

chess_contestant_data.txt



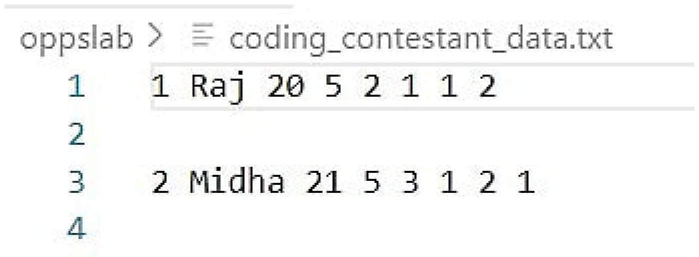
```
3 Pooja 19 3 2 1

4 Joel 21 2 3 1

```

chess_leaderboard.txt



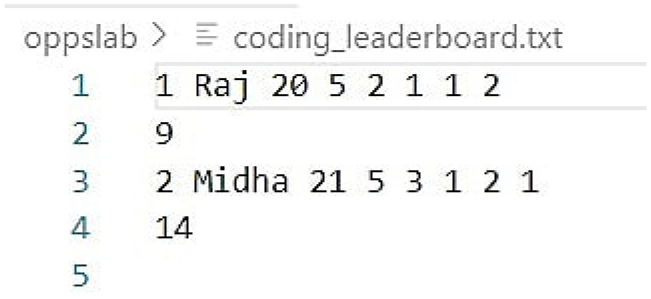
```
3 Pooja 19 3 2 1
13
4 Joel 21 2 3 1
14
```

coding_contestant_data.txt


oppslab > coding_contestant_data.txt

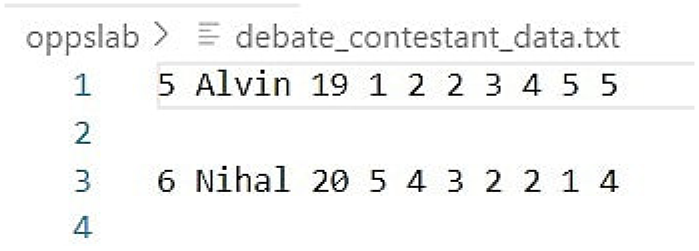
```
1 Raj 20 5 2 1 1 2

2 Midha 21 5 3 1 2 1

```

coding_leaderboard.txt


oppslab > coding_leaderboard.txt

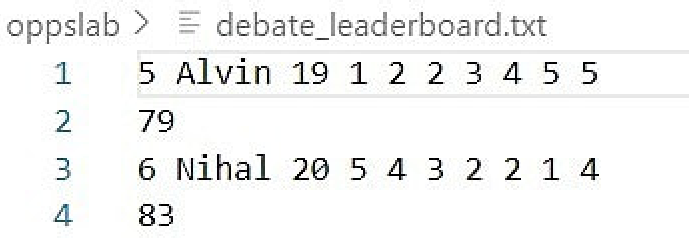
```
1 Raj 20 5 2 1 1 2
9
2 Midha 21 5 3 1 2 1
14

```

debate_contestant_data.txt


oppslab > debate_contestant_data.txt

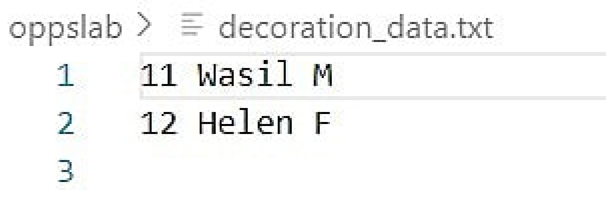
```
5 Alvin 19 1 2 2 3 4 5 5

6 Nihal 20 5 4 3 2 2 1 4

```

debate_leaderboard.txt


oppslab > debate_leaderboard.txt

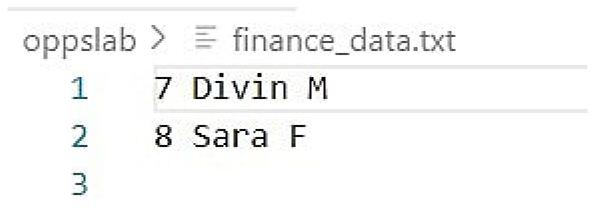
```
5 Alvin 19 1 2 2 3 4 5 5
79
6 Nihal 20 5 4 3 2 2 1 4
83
```

decoration_data.txt


oppslab > decoration_data.txt

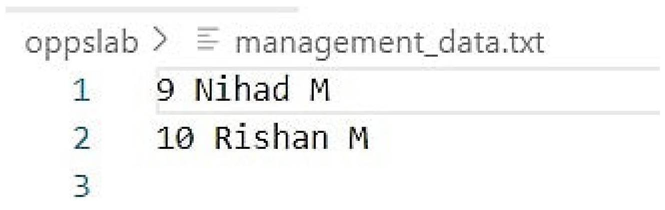
```
11 Wasil M
12 Helen F

```

finance_data.txt


oppslab > finance_data.txt

```
7 Divin M
8 Sara F

```

management_data.txt


oppslab > management_data.txt

```
9 Nihad M
10 Rishan M

```

LIMITATIONS

While using OOP concepts in C++ provides a solid foundation for the Contest Management System, there are certain limitations that might be encountered:

1. Limited Scalability:

Pure OOP leads to tight coupling between classes, making the system less scalable. As the project grows, managing dependencies and making changes could become challenging.

2. Performance Overhead:

In certain scenarios, the overhead of OOP concepts like polymorphism and dynamic dispatch may result in slightly slower performance compared to more low-level programming approaches, especially in critical sections of the code.

3. Memory Management Challenges:

C++ requires manual memory management, and without proper precautions, there might be risks of memory leaks or segmentation faults.

4. Dependency on Compiler:

The efficiency and success of the project is influenced by the compiler's ability to optimize the code. Different compilers may handle certain aspects of OOP features differently.

5. Difficulties in Testing:

Testing OOP-based code, especially with a deep hierarchy of classes, require more effort in terms of unit testing and mocking.

6. Inefficiencies in Resource Utilization:

OOP can lead to suboptimal utilization of system resources in certain cases. For performance-critical applications, more specialized programming paradigms might be preferred.

POSSIBLE IMPROVEMENTS

While the use of C++ and OOP principles provides a solid foundation for the Contest Management System, incorporating additional technologies can further enhance the project's capabilities. Here are some improvements that could be made by integrating other technologies:

Graphical User Interface (GUI):

Implement a graphical user interface using a GUI framework such as Qt or wxWidgets. This will enhance the user experience and make the system more visually appealing and user-friendly.

Database Management System (DBMS):

Integrate a dedicated database management system (e.g., MySQL, PostgreSQL) for more efficient data storage, retrieval, and management. This ensures scalability and facilitates data integrity.

Web Application:

Transform the project into a web-based application using technologies like HTML, CSS, and JavaScript, along with a backend framework like Django or Flask (in Python). This would make the system accessible from any browser, promoting ease of use and accessibility.

Cloud Integration:

Explore cloud services (e.g., AWS, Google Cloud, Microsoft Azure) for hosting the application and database. This enhances scalability, provides data redundancy, and ensures high availability.

Mobile Application:

Develop a mobile application using frameworks like React Native or Flutter. This allows organizers, administrators, and contestants to access the system conveniently from their mobile devices.

Data Analytics and Visualization:

Integrate data analytics tools (e.g., Python's Pandas and Matplotlib) to generate insightful reports and visualizations. This can aid organizers in gaining a deeper understanding of contest trends and performance metrics.

Security Measures:

Implement security measures such as encryption and secure authentication protocols to protect sensitive data. This is especially crucial when dealing with participant information and contest results.

Automated Notifications:

Incorporate a notification system using email or messaging services to keep participants and organizers informed about important updates, such as contest schedules, leaderboard changes, or winner announcements.

Machine Learning for Scoring Models:

Explore machine learning algorithms to create more sophisticated scoring models. This could involve analyzing historical data to refine the scoring criteria or predict potential issues.

Integration with Collaboration Tools:

Integrate with collaboration tools like Slack or Microsoft Teams for better communication and coordination among the organizing team members.

By incorporating these technologies, the Contest Management System can be elevated to a more sophisticated, user-friendly, and feature-rich platform, catering to the evolving needs of contest management in a modern technological landscape.