CONTEST MANAGEMENT SYSTEM

ACKNOWLEDGEMENT

We extend our heartfelt appreciation to Dr Rajneesh Rani for her exceptional guidance and support during the development of the Contest Management System project based on OOP in C++. Her expertise and insightful feedback significantly enriched the project, and we are grateful for the invaluable learning experience under her mentorship.

Sincerely,

Achyut Gupta - 22103006
Alakhsimar Singh - 22103008
Allen Varughese John - 22103009
CSE A-G1

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.

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.