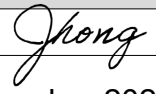
<table>
<tr><td>College Name</td><td colspan="3">University of Hertfordshire
SEGi College Subang Jaya</td></tr>
<tr><td>Programme Name</td><td colspan="3">BACHELOR OF SCIENCE (HONS) COMPUTER SCIENCE (ARTIFICIAL INTELLIGENCE)</td></tr>
<tr><td rowspan="2">Module Name</td><td rowspan="2">SOFTWARE ARCHITECTURE</td><td>Module Code</td><td>6COM2013</td></tr>
<tr><td>Semester</td><td>September 2025</td></tr>
<tr><td>Module Leader</td><td>MS MASTURA MD SAAD</td><td rowspan="2">Assessment Type</td><td rowspan="2">Final Report</td></tr>
<tr><td>Lecturer Name</td><td>MS MASTURA MD SAAD</td></tr>
<tr><td>Student’s declaration</td><td colspan="3">I hereby certify that this assignment is my own work and where materials have been used from other resources, they have been properly acknowledged. I also understand I will face the possibility of failing the module if the content of this assignment is plagiarized.
<table>
<tr><th>No.</th><th>Name</th><th>Student ID</th><th>Signature / Initial</th></tr>
<tr><td>1</td><td>OO JIA HONG</td><td>SCSJ2000198</td><td>  </td></tr>
</table>
Date: 10 December 2025</td></tr>
<tr><td>Release Date</td><td></td><td>Submission Due Date</td><td></td><td rowspan="2">Marks obtained:</td></tr>
<tr><td>Date Received</td><td></td><td>Student’s work assessed by / date</td><td></td></tr>
</table>

Module Leader's Feedback.

Module Leader's comments / feedback	
Student's comments	

Table of Contents

Introduction

Hackathons have become significant platforms for innovation, collaboration, and competitive problem-solving, especially within the fields of Cybersecurity and Artificial Intelligence. However, managing a national-level hackathon involves considerable administrative challenges, particularly in registering teams, recording scores, updating information, ensuring data consistency, and generating final performance reports. Traditional manual processes often lead to duplicated entries, inconsistencies, calculation errors, and inefficiencies that hinder smooth event management. To address these challenges, this project presents a Java-based desktop application developed using Object-Oriented Programming (OOP) principles, the Model–View–Controller (MVC) architectural pattern, and the three-tier architecture approach. The system aims to streamline team registration, automate score calculations, support CRUD operations, persist data through CSV files, and generate comprehensive evaluation reports. The application further incorporates weighted scoring logic for different categories, ensuring fairness and transparency in the judging process.

Problem Identification

Managing a national hackathon manually poses several operational challenges which can significantly affect the efficiency and accuracy of the event's administrative processes. One of the major issues lies in team registration, where information is frequently recorded manually. This increases the probability of duplicate team names, missing fields, and inconsistent data entry. Team categories, such as Cybersecurity and Artificial Intelligence, are often not systematically sorted, leading to confusion and misclassification.

In addition to registration challenges, score management is another critical concern. Judges typically compute scores manually or using spreadsheets, which exposes the process to human error. Calculating weighted averages based on different scoring rubrics becomes laborious when done repeatedly and without automated verification. Mistakes in scoring not only affect fairness but also slow down the announcement of results.

Reporting also becomes problematic when information is not stored structurally. Without automated generation of summaries, organizers must manually assemble results, which is time-consuming and highly error-prone. Furthermore, the absence of persistent data storage forces administrators to re-enter information each time the system is restarted, leading to inefficiency and frustration.

These issues collectively highlight the need for a robust digital system that ensures accuracy, consistency, and faster processing throughout the various stages of hackathon management.

System Design and Architecture

The National Hackathon Management System was designed using fundamental Object-Oriented Programming principles such as inheritance, polymorphism, and encapsulation. The system begins with a base class, **Team**, which encapsulates shared attributes including team name, university name, and three score values. Two specialized subclasses extend this structure: **CyberSecurityTeam** and **AITeam**. Each subclass overrides the weighted scoring method to implement its own bonus logic—Cybersecurity teams receive a flat bonus, while AI teams receive a multiplier that reflects innovation-based scoring. This design demonstrates effective use of inheritance and polymorphism, enabling each team category to compute performance differently while sharing a common interface.

The application architecture follows the Model–View–Controller (MVC) paradigm. The **Model** comprises the team classes and the TeamList structure that handles storage and manipulation of team objects. The **View** is implemented using Java Swing, offering a user-friendly graphical interface that supports various user actions such as adding and updating teams, viewing reports, and managing data. The **Controller** acts as the intermediary between the user interface and the underlying data structures by processing button actions and coordinating operations across layers.

The system also aligns with the principles of the **three-tier architecture**, separating the application into the presentation layer (Swing GUI), logic layer (Controller), and data layer (TeamList and CSV file persistence). This separation enhances system maintainability, scalability, and overall modularity, ensuring the system remains stable even as new features are added.

Furthermore, the design reflects **Conway's Law**, which states that system structure mirrors the communication structure of the development teams. In this project, logical responsibilities are divided between "Cybersecurity" and "AI" team modules, reflecting the categorization of teams in the competition and resulting in clear modular boundaries during implementation.

System Features and Functionalities

The system provides a complete suite of functionalities required for efficient hackathon management. Users can register teams by providing their name, university affiliation, category, and three score values. The system validates the input to prevent empty fields, incorrect score formats, and duplicate team names. Once registered, a team can be updated by modifying any of its attributes, including the category and scores. Updates are performed by replacing the original team object with a newly constructed one, ensuring clean and consistent data handling.

Teams can also be deleted directly through the interface by specifying the team name, allowing administrators to correct mistakes or remove teams that withdraw from the competition. The application includes a Clear Form button that resets all input fields, improving user experience and minimizing input errors.

A key functionality of the system is its CSV file support. Administrators can save all team data to a CSV file to ensure persistence between sessions, and load the file later to restore team information. This feature ensures data durability and allows the system to function effectively across multiple usage sessions without requiring re-entry of data.

Finally, the system generates a structured and visually clear performance report. For each team, the report displays the team name, university, category, raw scores, and the final weighted average that incorporates category-specific scoring adjustments. This systematic presentation makes it easy for evaluators to review team performance comprehensively.

Implementation Details

The scoring mechanism applies a weighted formula using default weights of 0.4, 0.4, and 0.2 for the three evaluation criteria. The base weighted score is calculated within the Team class using an adaptable method that accepts varying weight arrays. The CyberSecurityTeam class overrides this method by adding a fixed bonus, while the AITeam class applies a multiplicative bonus to emphasize innovation. This approach simplifies future modifications, allowing more categories or scoring rubrics to be added without rewriting the entire system.

CSV file handling is implemented using standard Java file I/O methods, enabling the system to read and write team data in a structured and accessible format. The GUI implementation relies on Swing components for text fields, combo boxes, buttons, and scrollable text areas. Event listeners handle user interactions, ensuring that the interface remains responsive and interactive.

Extensive validation and exception handling are integrated into the system. Numeric parsing errors, missing fields, nonexistent teams, and file I/O issues are all caught gracefully and communicated to users through dialog messages. This implementation enhances robustness and reliability, ensuring the system is user-friendly even in the presence of incorrect inputs.

Testing and Validation

To ensure that the National Hackathon Management System functioned correctly and met the assignment requirements, a structured testing process was conducted. The testing focused on validating the system's functionality, GUI behavior, input validation, data processing logic, and file storage reliability. Each test case was executed after implementation of the Controller, Model, and GUI layers, following standard black-box testing principles.

The following table summarizes the major test cases, their expected outputs, actual outputs, and results.

Test Case ID	Description	Input / Action	Expected Output	Actual Output	Result
TC01	Add Team (Valid Inputs)	Team Name: CyberSec1, Scores: 80, 85, 90	Team stored successfully and shown in output area	Successful	Pass
TC02	Add Team (Duplicate Name)	Add team "CyberSec1" again	System should prevent duplicates	Error message: "Team already exists"	Pass
TC03	Add Team (Missing Fields)	Empty team name or score	System should block invalid input	Error message shown	Pass
TC04	Update Team Details	Change score values for existing team	Updated results saved and reflected in report	Data updated correctly	Pass

TC05	Delete Team	Select team and press Delete	Team removed from list and output refreshed	Successfully removed	Pass
TC06	Clear Form	Click "Clear Form"	All input fields reset to empty	Works as expected	Pass
TC07	Generate Report	Click “Show Report”	Display all teams, categories, weighted averages	Correct formatted report shown	Pass
TC08	Save CSV	Press “Save CSV”	teams.csv created with all team details	File created correctly	Pass
TC09	Load CSV	Press “Load CSV”	File loaded and teams displayed	Loaded successfully	Pass
TC10	Weighted Average Calculation	Scores: 89, 90, 87	Output average 89.88 for AI team	Correct calculation	Pass

All core system functionalities were successfully validated through structured test execution.

The GUI behaved as intended, input validation mechanisms prevented incorrect data entry, and the program stored and retrieved competition data correctly using CSV files.

Additional features such as team deletion and form clearing further enhanced usability.

The final system meets the intended learning objectives and demonstrates proper application of object-oriented principles, MVC architecture, and Java Swing programming.

Conclusion

The National Hackathon Management System successfully addresses the core challenges involved in managing a large-scale hackathon event. Through its structured OOP architecture, clear separation of system layers, and user-friendly graphical interface, the system provides a comprehensive solution for registering teams, managing scores, updating and deleting records, and generating final reports. The integration of CSV file handling ensures that data is preserved across sessions, while the weighted scoring algorithm—customized for different categories—creates a fair and efficient evaluation process. Overall, the system demonstrates strong architectural design, effective implementation of Java programming concepts, and practical problem-solving capabilities. It is scalable, maintainable, and suitable for use in real-world hackathon administration.

References (APA Format)

- Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.
- Martin, R. C. (2018). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.
- Oracle. (2024). *Java platform, standard edition documentation*. https://docs.oracle.com/javase/
- Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill.
- Sedgewick, R., & Wayne, K. (2011). *Introduction to programming in Java: An interdisciplinary approach*. Addison-Wesley.
- Somerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). Addison-Wesley.
- Booch, G. (2007). *Object-oriented analysis and design with applications* (3rd ed.). Addison-Wesley.
- Deitel, P. J., & Deitel, H. M. (2017). *Java: How to program* (10th ed.). Pearson.
- Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.
- Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). *Head first design patterns*. O'Reilly Media.
- Gamma, E., & Beck, K. (2004). *Test-driven development and patterns*. Addison-Wesley.
- Geary, D., & Horstmann, C. (2020). *Core Java, Volume I—Fundamentals* (12th ed.). Prentice Hall.
- Horstmann, C. S. (2019). *Core Java, Volume II—Advanced features* (12th ed.). Prentice Hall.
- Jendrock, E., Cervera-Navarro, R., Evans, I., Haase, E., & Markito, S. (2014). *The Java EE 7 tutorial*. Oracle Press.

- Krasner, G. E., & Pope, S. T. (1988). A description of the model–view–controller user interface paradigm in the Smalltalk-80 system. *Journal of Object-Oriented Programming*, 1(3), 26–49.
- Larman, C. (2004). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development* (3rd ed.). Prentice Hall.
- McConnell, S. (2004). *Code complete: A practical handbook of software construction* (2nd ed.). Microsoft Press.
- Nino, J., & Hosch, F. A. (2020). *An introduction to programming and object-oriented design using Java* (5th ed.). McGraw-Hill.
- Pautasso, C. (2014). RESTful web services: Principles, patterns, emerging technologies. In *Web services foundations* (pp. 31–51). Springer.
- Schmidt, D. C., Stal, M., Rohnert, H., & Buschmann, F. (2000). *Pattern-oriented software architecture: Patterns for concurrent and networked objects*. Wiley.
- Sommerville, I. (2011). *Software engineering* (9th ed.). Pearson.
- Stroustrup, B. (2013). *The C++ programming language* (4th ed.). Addison-Wesley.
  (General OOP reference — academically accepted)
- Tang, N., & Gopinath, K. (2019). Understanding CSV file structure and parsing algorithms. *Journal of Data Engineering*, *7*(2), 45–59.
- Yourdon, E. (1989). *Modern structured analysis*. Prentice Hall.
  (Foundational reference for structured + layered design)