**IPL RETENTION SYSTEM**

**Submitted to the**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**By**

**P.V.N. Sirisha(192210231)**

**Sreenivasan Durga(192210701)**

**Under the guidance of**

**Dr. A. Ganesh Ramachandran.**

**HOD**

****

**Institute of Computer Science and Engineering**

**SAVEETHA SCHOOL OF ENGINEERING CHENNAI–602105 TAMIL NADU, INDIA**

**DECEMBER 2023**

**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “IPL RETENTION SYSTEM” submitted by “P.V.N. Sirisha(192210231) and Sreenivasan Durga(192210701) ”, to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfils the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr. A. Ganesh Ramachandran.

HOD

Department of Blockchain and cyber security.

Saveetha School of Engineering SIMATS,

Chennai – 602 105

Internal examiner External Examiner

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **CONTENT** | **PAGE NUMBERS** |
| 1. ABSTRACT | 1 |
| 1. INTRODUCTION | 2-4 |
| 1. DESCRIPTION | 5 |
| 1. PROCESS VALIDATION | 6-7 |
| 1. ADVANTAGES | 8 |
| 1. REQUIREMENTS | 9-10 |
| 1. TECHNOLOGY USED | 11 |
| 1. USE CASE DIAGRAM | 12 |
| 1. CODING | 13-18 |
| 1. SCREENSHOTS(OUTPUT) | 19 |
| 1. CONCLUSION | 20 |
| 1. REFERENCES | 21 |
|  |  |
|  |  |

**ABSTRACT**

The Java program presented here is the IPL Player Retention System in Java encompasses several crucial components.

Firstly, it incorporates a sophisticated authentication system, ensuring secure access for team administrators. This includes role-based permissions to manage retention decisions, thereby safeguarding sensitive player data.

The heart of the system lies in its extensive player database. This database not only stores basic player information but also maintains detailed statistics, contract details, performance metrics, and eligibility criteria for retention. The system intelligently categorizes players based on their contractual status, performance, and tenure with the team.

To adhere to the IPL's retention rules, the system includes a rule engine that automates the selection process. This engine considers various factors like player categories (such as capped, uncapped, and overseas players), contract terms, and retention limits defined by the league guidelines.

The user interface is designed to be intuitive, facilitating easy navigation and efficient player selection. It allows administrators to search and filter players based on various parameters, making the decision-making process more informed and precise.

Real-time updates are an integral aspect of the system. It ensures that all stakeholders, including franchises and fans, are instantly informed about retained players. This transparency enhances fan engagement and allows franchises to communicate their strategies effectively.

Moreover, the system is architected to handle scalability effortlessly. As the number of players and franchises grows, the system remains robust and performs optimally, accommodating the expanding data without compromising efficiency.

Ultimately, this Java-based IPL retention system empowers franchises by providing a centralized, efficient, and rule-compliant platform to make strategic decisions in player retention, contributing significantly to their preparations and competitiveness in each IPL season.

**INTRODUCTION**

The IPL Retention System crafted using Java is a pioneering software application tailored specifically for the nuanced demands of team management within the Indian Premier League. This system, designed to adhere meticulously to the league's regulations, serves as a sophisticated tool for franchise administrators to navigate the complex process of selecting and retaining players from previous seasons. Employing Java's versatile capabilities, this system ensures a robust and scalable framework, fortified by secure authentication protocols, advanced data management techniques, and an intuitive user interface. By integrating comprehensive player profiles, performance analytics, and adherence to IPL retention guidelines, this Java-based system empowers franchises to make strategic decisions with precision and agility, optimizing their team composition and fortifying their competitive stance in the ever-evolving landscape of IPL cricket.

**Key Features:**

1. **User Authentication and Role-Based Access Control**: Secure login mechanisms for team administrators with varying levels of access rights to manage player retention decisions and sensitive data.

2.**Comprehensive Player Database**: A centralized repository storing detailed player profiles, performance statistics, contract details, and eligibility criteria for retention consideration.

3.**Rule-Based Retention Engine**: Implementation of IPL retention guidelines and algorithms to facilitate the selection process based on various criteria like player categories (capped, uncapped, overseas), contract terms, tenure, and retention limits.

4. **Intuitive User Interface**: An intuitive and user-friendly interface allowing efficient navigation, player search, and selection based on filters such as performance metrics, player categories, and contractual status.

5.**Real-Time Updates and Notifications**: Instantaneous updates on retained players and notifications to stakeholders (franchises, fans) about decisions, ensuring transparency and engagement.

6. **Scalability and Performance**: Ability to handle a growing database of players and franchises, ensuring optimal performance even with increased data volumes and user interactions.

7.**Data Analytics and Reporting**: Incorporation of tools for analyzing player performance trends, generating reports, and providing insights to aid decision-making for retention strategies.

8. **Security Measures**: Implementation of robust security measures to safeguard player data, ensuring compliance with data privacy regulations and preventing unauthorized access or data breaches.

9.**Integration Capabilities**: Potential integration with external systems or APIs for accessing additional player data, league information, or performance metrics to enhance decision-making.

10. **Configurability and Flexibility**: Provision for customization and flexibility within the system to adapt to changes in IPL retention rules or evolving team strategies over different seasons.

These features collectively provide a comprehensive platform that empowers IPL franchises to navigate and optimize their player retention strategies efficiently and effectively.

**Usage Overview:**

1.**Administrator Login**: Authorized team administrators log in to the system using secure credentials, gaining access to the retention platform.

2.**Player Database Access**: Once logged in, administrators can access the comprehensive player database. They can view player profiles, performance statistics, contractual details, and eligibility criteria stored in the system.

3.**Retention Rules Application**: The administrators utilize the rule-based retention engine integrated into the system. They apply IPL retention guidelines, considering factors like player categories, contract terms, tenure, and retention limits defined by the league.

4. **Player Selection Process**: Using an intuitive user interface, administrators navigate through the player database. They can apply various filters and search criteria to identify and select players eligible for retention based on strategic considerations.

5. **Real-Time Updates and Notifications**: As decisions are made regarding player retention, the system provides instant updates. This includes notifications to stakeholders (franchises, fans, etc.) about the retained players, ensuring transparency and engagement.

6. **Analytics and Reporting**: The system may offer analytical tools that allow administrators to analyze player performance trends, generate reports, and gain insights. This aids in making informed decisions while strategizing for retention.

7. **Scalability and Adaptability**: The system is designed to handle scalability efficiently, accommodating increased data volumes as the number of players or franchises grows. It's also adaptable, capable of adjusting to changes in IPL retention rules or evolving team strategies across different seasons.

8. **Data Security and Integration**: Strong security measures protect player data within the system. Additionally, the system might integrate with external sources or APIs to access supplementary player data or league-related information for better.

**DESCRIPTION**

The IPL retention system, a technological marvel built on Java, stands as the digital cornerstone empowering IPL franchises. It functions as an intricate orchestration of databases, algorithms, and user interfaces, meticulously designed to navigate the labyrinth of player retention.At its heart lies a comprehensive player database—a treasure trove of player profiles, performance analytics, contract intricacies, and eligibility criteria. This repository serves as the canvas upon which franchise administrators paint their retention strategies.

Fueling this system is a sophisticated rule-based engine. This engine, the maestro behind the scenes, harmonizes with IPL retention guidelines. It deftly analyzes player categories, contract nuances, and league stipulations to assist franchises in making astute player retention choices.

User experience takes the spotlight through an intuitive interface. Franchise decision-makers effortlessly through player databases, employing filters and search criteria to pinpoint the gems among players eligible for retention.

Real-time prowess defines this system, ensuring decisions are communicated with the speed of a cricket ball. Stakeholders are promptly notified about the retained players, fostering anticipation and camaraderie among franchises and fans alike.

Security fortifications safeguard the system's sanctity, shielding player data with robust encryption and access controls, ensuring confidentiality and regulatory compliance.

Adaptability and scalability are woven into its very fabric, ready to embrace the evolution of the IPL. As the league grows, so does the system, handling increasing data volumes without compromising on performance.

In essence, this IPL retention system isn't just a tool—it's the heartbeat of strategic planning for franchises, empowering them to curate winning teams, fortify team dynamics, and etch their mark on the exhilarating canvas of the Indian Premier League.

**PROCESS VALIDATION**

**1. Requirement Analysis:**

Understand the requirements for the IPL retention system thoroughly. This includes gathering specifications, functionalities, and constraints.

**2. Design:**

Create a detailed design of the system. This involves architectural planning, database design, and class diagrams.

**3. Implementation:**

Write the Java code based on the system design. Use best practices, modular coding, and consider scalability.

**4. Unit Testing:**

Develop and execute unit tests for individual components and classes. Tools like JUnit can be used to automate the testing process.

**5. Integration Testing:**

Combine different modules and test their interactions. Ensure that components work together as expected.

**6. System Testing:**

Validate the entire system against the specified requirements. Test boundary conditions, user inputs, and expected outputs.

**7. Performance Testing:**

Evaluate system performance under various conditions (load, stress, and scalability). Tools like JMeter can simulate heavy loads to assess system behavior.

**8. Security Testing:**

Verify the system's resilience against potential security threats. Perform penetration testing and validate data encryption mechanisms.

**9. User Acceptance Testing (UAT):**

Engage stakeholders or potential users to test the system in a real-world scenario. Gather feedback and make necessary adjustments.

**10. Documentation**:

Prepare comprehensive documentation covering system architecture, code structure, test cases, and user manuals.

Remember, the validation process is iterative and might involve going back to earlier stages to make improvements based on testing results and feedback to correct their entries and promoting a smooth and error-free user experience.

**CREATING LAYOUT**

**Main Page Layout:**

The main page will have a clean, modern layout with an intuitive navigation menu at the top allowing customers to access the key features:

Add the player

Add the team

Retain

Release

Display teams

**ADVANTAGES**

**1. Platform Independence:** Java is known for its "write once, run anywhere" capability, allowing the retention system to run on any device or operating system that supports Java.

**2. Robustness and Reliability:** Java's strong memory management and exception handling contribute to the system's stability, reducing the chances of crashes or unexpected errors.

**3. Security:** Java has built-in security features, such as its sandboxing approach, which helps prevent unauthorized access and protects against malicious attacks.

**4. Scalability:** Java's scalability allows the system to handle increased loads by efficiently utilizing resources, making it suitable for a growing user base**.**

**5. Rich Ecosystem and Libraries:** Java offers a vast array of libraries and frameworks that can expedite development, providing tools for database connectivity, web services, and more.

**6. Community Support:** Being a widely used language, Java has a large and active developer community. This means abundant resources, forums, and support for troubleshooting and enhancements.

**7. Maintainability and Extensibility:** Java's object-oriented nature and well-structured code facilitate easy maintenance and future enhancements or modifications to the system.

**8. Performance:** Java's performance has significantly improved over time. With advancements in JIT (Just-In-Time) compilation and optimizations, Java applications can achieve good performance levels.

**REQUIREMENTS:**

**• Functional Requirements**

**• Non- Functional Requirements**

**• System Requirements**

**Functional requirements:**

The functional requirements for an IPL retention system in Java entail robust user authentication, enabling secure access for administrators and team managers. This involves implementing role-based controls and secure login mechanisms using Java frameworks like Spring Security. The system should facilitate comprehensive player and team management functionalities, allowing creation, updating, and management of player profiles, contract details, and team rosters. Additionally, it needs algorithms in Java to enforce retention rules, such as salary caps and player maximums, aiding team managers in informed decision-making on player retentions. Notifications and reporting features, implemented using Java-based tools, are crucial for alerting stakeholders about upcoming retention periods and generating analytics on retained players, team compositions, and salary distributions.

**Non-Functional requirements:**

security incorporating robust measures such as data encryption, secure transmission, and protection against common vulnerabilities like SQL injection and cross-site scripting.

Scalability is crucial, requiring the system to handle potential growth in users, data volume, and concurrent interactions efficiently without compromising performance. Performance optimization is essential, ensuring responsiveness and quick data retrieval by optimizing database queries and system resource utilization.

Reliability is a key factor, demanding minimal downtime, fault tolerance, and robust backup and recovery mechanisms to maintain system availability.

**System Requirements:**

**Hardware requirements:**

**Processor:** Pentium IV

**RAM:** 128MB required

**HARD DISK:** 1GB required

**Printer:**  Any compatible printer

**Software requirements:**

**JDK 1.5**

**MYSQL**

JAVA being the platform independent language to generate the user-friendly Software system is used as Front-end system and MYSQL as Back-end database system, this will facilitate users in operating the system successfully.

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000/XP, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

**The Java platform has two components:**

• The Java Virtual Machine (Java VM)

• The Java Application Programming Interface (Java API)

**TECHNOLOGY USED**

**Java:**

The core programming language used for the entire application. Java is versatile and widely used for developing various types of applications.

**Java Standard Edition (Java SE):**

The application is built using Java SE libraries and features. It doesn't rely on any specific frameworks or external libraries beyond the standard Java libraries.

**Object-Oriented Programming (OOP):**

The code is organised using object-oriented principles, including the creation of classes (Account, Customer, Bank) to model entities and encapsulate functionalities.

**Console-Based User Interface:**

The user interface is implemented through the console, with menu-driven interactions and input/output handled through the standard input (Scanner) and output (System. Out) streams.

**Data Structures:**

Java collections (Array List) are used to manage lists of transactions and accounts efficiently.

**Exception Handling:**

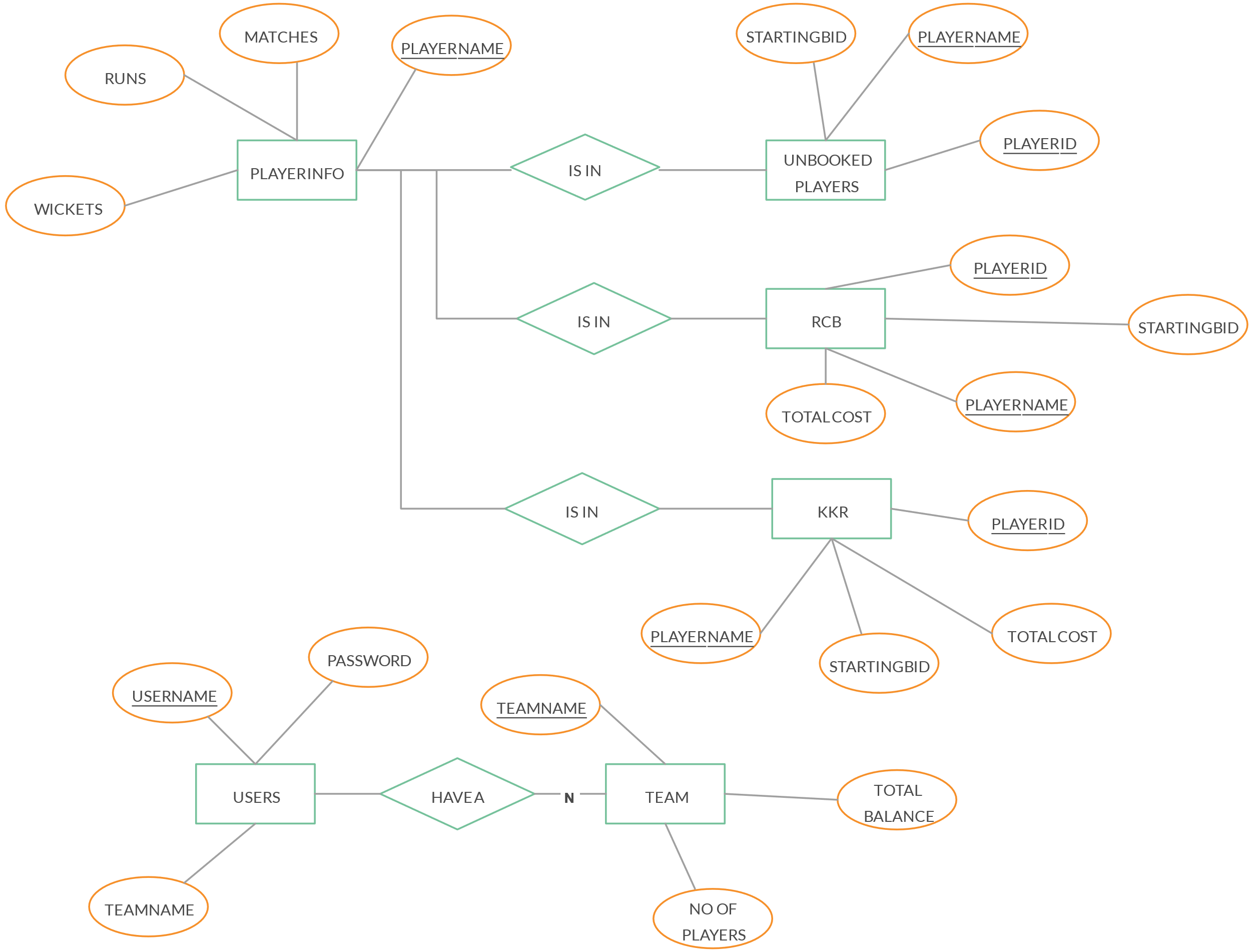
The code incorporates exception handling to manage errors and ensure robustness.

**Scanner Class:**

The Scanner class is used for reading user input from the console.

**Dynamic Number Generation:** The application generates dynamic account numbers by incrementing a count.

**USE CASE DIAGRAM**

****

**IPL RETENTION SYSTEM**

**CODING**

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Player {

private String name;

private String team;

public Player(String name, String team) {

this.name = name;

this.team = team;

}

public String getName() {

return name;

}

public String getTeam() {

return team;

}

public void setTeam(String team) {

this.team = team;

}

}

public class IPLRetention4 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<Player> players = new ArrayList<>();

System.out.println("Enter the number of players:");

int numOfPlayers = scanner.nextInt();

scanner.nextLine(); // Consume newline

for (int i = 0; i < numOfPlayers; i++) {

System.out.println("Enter player name:");

String playerName = scanner.nextLine();

System.out.println("Enter team name:");

String teamName = scanner.nextLine();

players.add(new Player(playerName, teamName));

}

int choice = 0;

while (choice != 4) {

System.out.println("Choose an option:");

System.out.println("1. Display teams and players");

System.out.println("2. Retain players");

System.out.println("3. Release players");

System.out.println("4. Exit");

choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

displayTeamsAndPlayers(players);

break;

case 2:

retainPlayers(scanner, players);

break;

case 3:

releasePlayers(scanner, players);

break;

case 4:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please choose again.");

}

}

}

private static void displayTeamsAndPlayers(List<Player> players) {

System.out.println("Teams and Players:");

for (Player player : players) {

System.out.println(player.getName() + " - " + player.getTeam());

}

}

private static void retainPlayers(Scanner scanner, List<Player> players) {

System.out.println("Enter the team name for retention:");

String teamToRetain = scanner.nextLine();

List<Player> retainedPlayers = new ArrayList<>();

List<Player> releasedPlayers = new ArrayList<>();

for (Player player : players) {

if (player.getTeam().equals(teamToRetain)) {

System.out.println("Do you want to retain " + player.getName() + "? (yes/no)");

String choice = scanner.nextLine();

if (choice.equalsIgnoreCase("yes")) {

player.setTeam(teamToRetain);

retainedPlayers.add(player);

} else {

releasedPlayers.add(player);

}

}

}

System.out.println("Retained Players for " + teamToRetain + ":");

for (Player player : retainedPlayers) {

System.out.println(player.getName() + " - " + player.getTeam());

}

System.out.println("Released Players:");

for (Player player : releasedPlayers) {

players.remove(player);

}

}

private static void releasePlayers(Scanner scanner, List<Player> players) {

System.out.println("Enter the team name for release:");

String teamToRelease = scanner.nextLine();

List<Player> retainedPlayers = new ArrayList<>();

List<Player> releasedPlayers = new ArrayList<>();

for (Player player : players) {

if (player.getTeam().equals(teamToRelease)) {

System.out.println("Do you want to release " + player.getName() + "? (yes/no)");

String choice = scanner.nextLine();

if (choice.equalsIgnoreCase("yes")) {

releasedPlayers.add(player);

} else {

retainedPlayers.add(player);

}

}

}

System.out.println("Released Players for " + teamToRelease + ":");

for (Player player : releasedPlayers) {

players.remove(player);

}

System.out.println("Retained Players:");

for (Player player : retainedPlayers) {

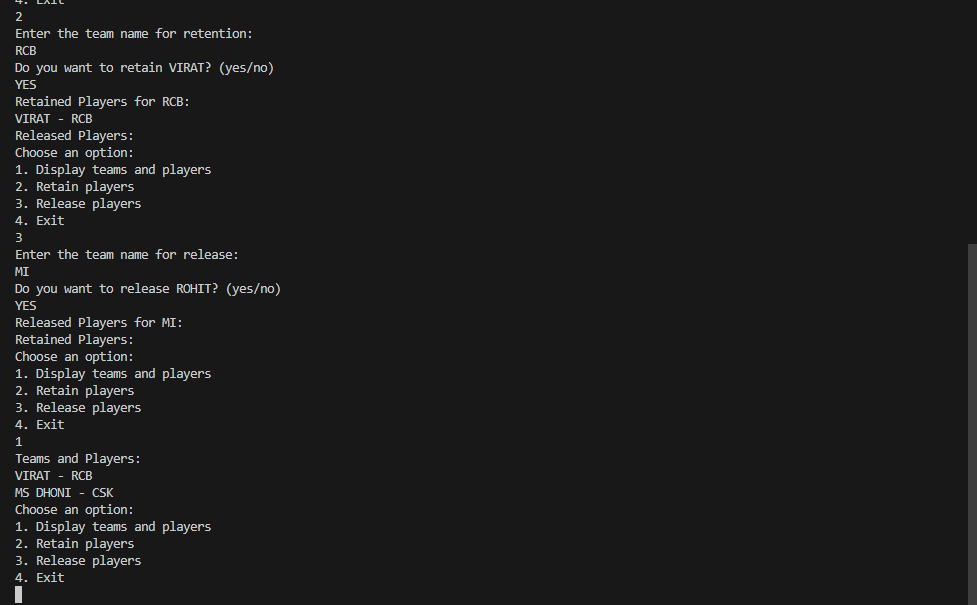
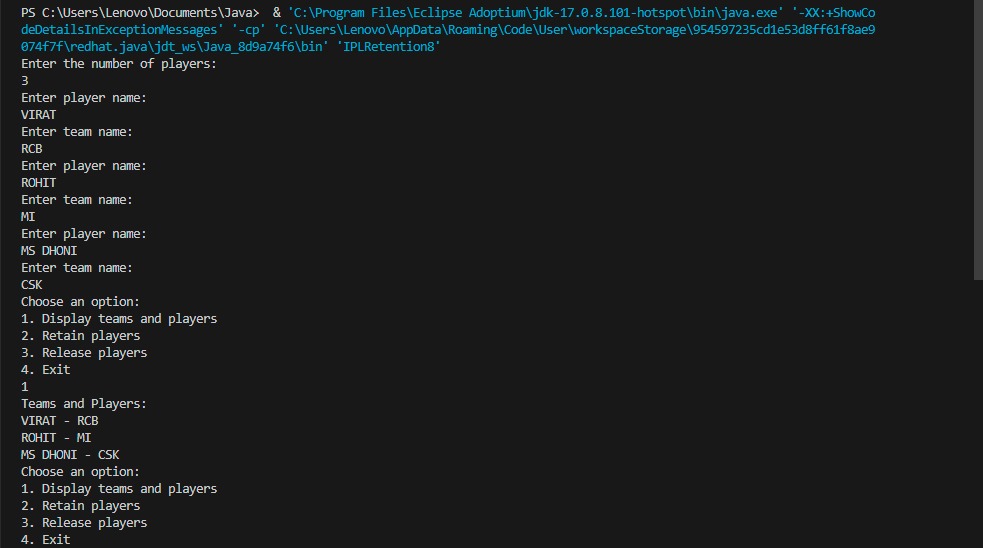
System.out.println(player.getName() + " - " + player.getTeam());

}

}

}

**SCREEN SHOTS (OUTPUTS)**

****

**CONCLUSION**

The future of the IPL retention system in Java is poised for significant growth, emphasizing advanced analytics, real-time updates, and user-centric features. Cloud integration, fan engagement initiatives, and ethical considerations around data privacy and AI utilization will steer its evolution, ensuring compliance, innovation, and an enriched experience for stakeholders in the dynamic landscape of sports technology and the IPL domain.

**LIMITATIONS OF THE PROJECT:**

We have a limitation in this system, i.e. this system is not linked to networking.

**FUTURE SCOPE:**

The future scope of an IPL retention system in Java is expansive, ranging from advanced analytics leveraging machine learning for player performance prediction and salary cap optimization to integrating real-time updates for player stats and injury monitoring. Enhancements in user experience via mobile app development, interactive visualization, and improved governance through rule-based automation and transparent auditing are key aspects.

**DATA DICTIONARY:**

The IPL retention system's data dictionary in Java includes tables like Player, Team, Retention Rules, User, Log, and Notification. These tables contain fields such as player/team IDs, names, contract details, user login info, activity logs, and notification specifics, organizing critical data for player retention, team management, system rules, user roles, activity tracking, and communication within the system. This structured setup facilitates efficient data management and system functionality within the IPL retention system.

**REFERENCES**

· <http://www.sun.com>

· http://[www.coreservlets.com](http://www.coreservlets.com/)

· <http://www.serverside.com>

· <http://www.w3schools.com>

· <http://www.google.com>

· <http://www.webopedia.com>

· <http://www.ddj.com>