# BlackJack Game - Project Report

## 1. Abstract

This project implements a graphical BlackJack card game using Java Swing for the interface and JDBC for persistent data storage.   
The game allows a player to compete against a dealer following standard Blackjack rules — “Hit” to draw a new card or “Stand” to stop drawing.   
It demonstrates Object-Oriented Programming (OOP) principles such as classes, encapsulation, and modularity, while also integrating a database backend to store player profiles and match history.

## 2. Objectives

• To simulate the real-world card game “BlackJack” using Java.  
• To apply OOP principles in structuring game logic and player-dealer interaction.  
• To design an interactive GUI using Swing and AWT.  
• To implement JDBC connectivity for storing player data and results.  
• To demonstrate the integration of logic, GUI, and database layers in a modular application.

## 3. Requirements

Software Requirements:

- Java Development Kit (JDK 17 or above)  
- IDE (IntelliJ IDEA / Eclipse / VS Code)  
- MySQL Database Server  
- JDBC Connector (MySQL Connector/J)

Knowledge Requirements:

- Object-Oriented Programming (Java)  
- Basic SQL and JDBC operations  
- Event handling and GUI design in Swing  
- Image rendering using Java AWT

## 4. System Design

Use Case Flow:  
1. Player starts the game.  
2. System shuffles and deals initial cards.  
3. Player chooses Hit or Stand.  
4. Dealer automatically plays according to Blackjack rules.  
5. System evaluates and displays the result.  
6. Game result is stored in the database.

## 5. Class Structure

Main Module – App.java  
- Contains the main() method.  
- Launches the game by instantiating the BlackJack class.  
  
Game Controller – BlackJack.java  
- Central logic of the application.  
- Handles deck creation, shuffling, and rule enforcement.  
- Manages GUI rendering and player input.  
- Integrates JDBC methods for data storage.  
  
Inner Class – Card  
- Represents each card with value and type.  
- Includes methods: getValue(), isAce(), getImagePath().  
  
Database Module  
- Establishes connection to MySQL using JDBC.  
- Manages players and games tables.  
- Provides connect(), getOrCreatePlayerId(), saveGameResult().  
  
GUI Module  
- Built using JFrame, JPanel, and JButton.  
- Custom paintComponent() for card rendering.  
- Buttons: Hit, Stand, New Game.

## 6. Module Breakdown

|  |  |
| --- | --- |
| Module | Description |
| Main Module | Entry point (App.java) |
| Game Logic | Handles game rules, scoring, and ace reduction |
| Deck & Card Handling | Builds, shuffles, and manages 52-card deck |
| Player/Dealer Management | Tracks hands, sums, and ace counts |
| GUI Module | User interface, card rendering, and event handling |
| Database Module | Player data management and score recording |

## 7. Database Integration (DBMS Application)

Database: blackjack\_db  
  
Tables:  
CREATE TABLE players (  
 player\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 player\_name VARCHAR(100) NOT NULL  
);  
  
CREATE TABLE games (  
 game\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 player\_id INT,  
 game\_outcome VARCHAR(10),  
 player\_score INT,  
 dealer\_score INT,  
 played\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (player\_id) REFERENCES players(player\_id)  
);  
  
Features:  
- Persistent player identity across sessions.  
- Automatic creation of new player records.  
- Game result logging for analytics or leaderboard implementation.

## 8. Gameplay Flow

1. Player enters name → retrieved/created in DB.  
2. Cards are shuffled and dealt.  
3. Player plays via “Hit” or “Stand”.  
4. Dealer plays automatically.  
5. Game result (Win/Loss/Tie) displayed on screen.  
6. Result saved to the database.  
7. “New Game” button allows replay.

## 9. OOP Concepts Demonstrated

- Encapsulation: Data (player hands, scores) contained within classes.  
- Abstraction: GUI hides complex game logic.  
- Composition: Deck built using Card objects.  
- Polymorphism: Event handling via action listeners.  
- Modularity: Game, GUI, and DB modules operate independently.

## 10. Results and Testing

- Game runs smoothly with GUI updates after each action.  
- Database correctly logs outcomes and player info.  
- Edge cases (Aces adjustment, player bust, dealer bust) handled correctly.  
- Visual assets dynamically loaded from the /cards directory.

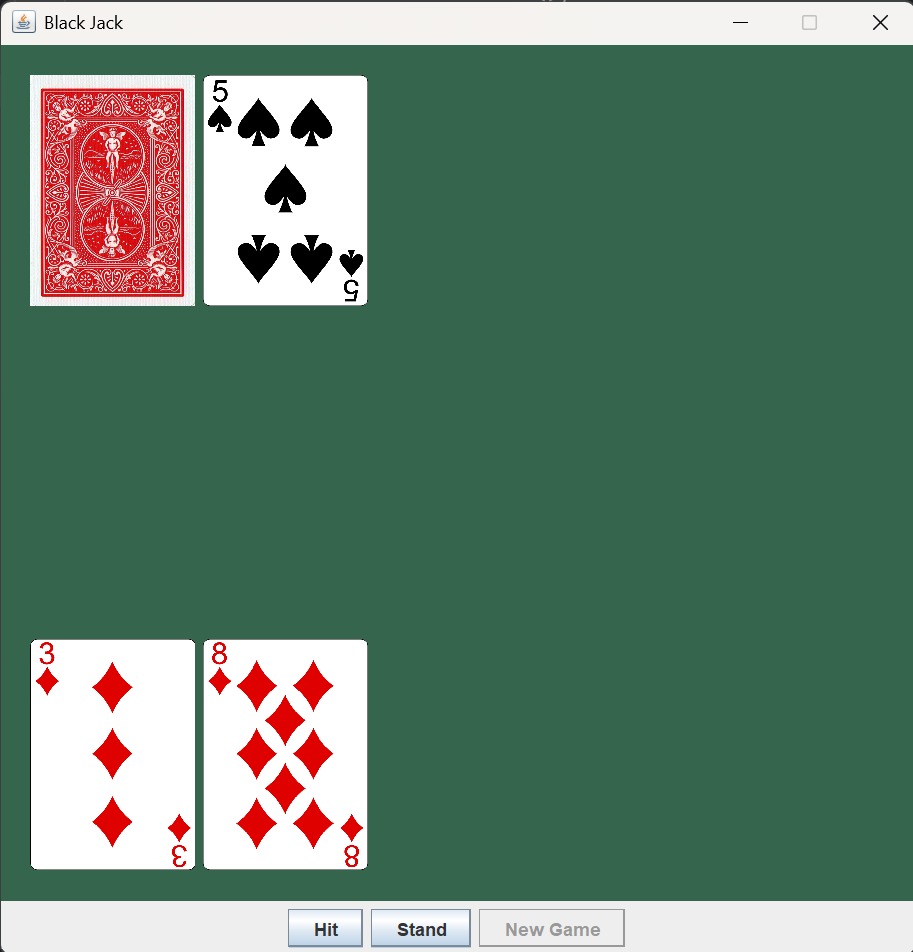
## 11. Future Enhancements

- Add Leaderboard view showing top players.  
- Implement Login system for multiple players.  
- Add Sound effects and Animations for enhanced UX.   
- Deploy as a web-based application using JavaFX or Spring Boot.

## 12. Sample Run



When the game runs, it asks for the name of the player to play the game and store the result



* The player is dealt 2 random cards at the start and is given the option to either **Hit** or **Stand**.
* Since the total card value is currently 11 (8 + 3 = 11), there’s a high chance that if the player hits, the total will be less than or equal to 21.
* Therefore, the player decides to **Hit** for another card.

## 

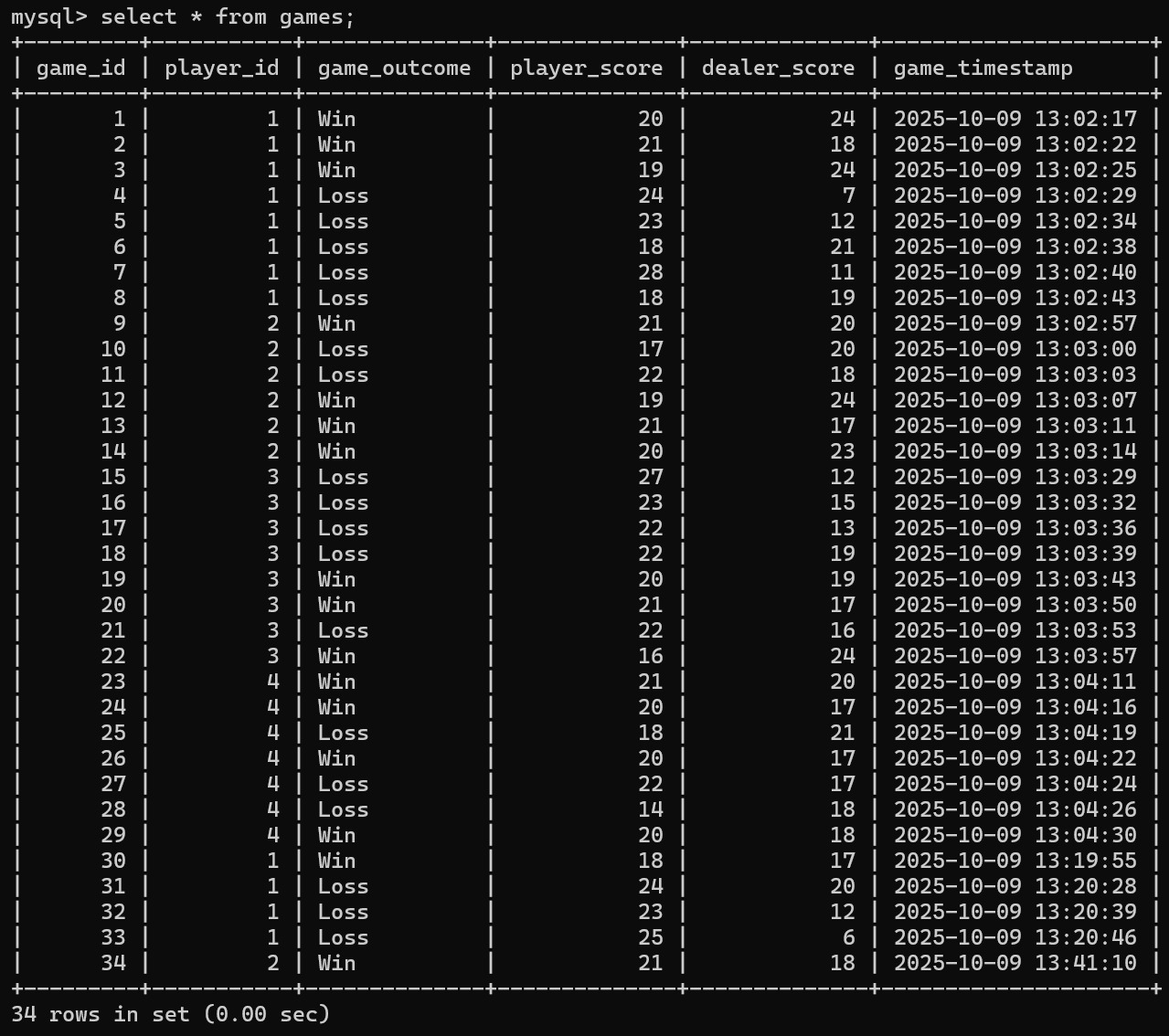
* Another card is dealt to the player. Since the value of all face cards is 11, the new total is 8 + 3 + 10 = 21.
* Since the player is not supposed to go above 21, they choose to stand to stop further dealing of cards.
* Now the dealer deals cards to themselves.

## 

* The dealer deals themselves trying to reach a number greater than the player’s without busting.
* Here, the dealer stopped at 18. Since he has a total less than the player, the player wins the game.
* The player then gets the option to start a **New Game**.

## 13. Databases

The game uses two databases to log the results of every game played, and to store different players by assigning unique ids to them.





## 14. Conclusion

The BlackJack Game Project successfully integrates Java’s object-oriented features with GUI and database components to deliver an interactive, real-world application.   
It effectively demonstrates modular programming, user interface design, and persistent storage — providing a strong foundation for further extension into multiplayer or online modes.

## Team Members

- Jatin Swami (RA2411030010186)  
- Nitya Mehrotra (RA2411030010169)  
- Sai Shravya (RA2411030010190)  
- Abritha John (RA2411030010168)