## HORSE RACE BETTING MANAGEMENT

**A MINI PROJECT REPORT**

|  |  |  |
| --- | --- | --- |
| **Submitted** | **by** |  |
| **SOWMIYA NARAYANNAN**  **GK** |  | **220701284** |
| **TAMILINI DK** |  | **220701299** |
|  |  |  |
|  |  |  |

**In partial fulfillment for the award of the degree of**

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE**

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM**

**CHENNAI-602105**



2023 - 24

**BONAFIDE CERTIFICATE**

Certified that this project report “**HORSE RACE BETTING SYSTEM**” is the bonafide work of **“SOWMIYA NARAYANNAN GK (220701284),**

**TAMILINI DK(220701299) ”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE SIGNATURE**

**Dr.R.SABITHA Ms.V.JANANEE**

**Professor and II Year Academic Head Assistant Professor (SG),**

**Computer Science and Engineering, Computer Science and Engineering, Rajalakshmi Engineering College Rajalakshmi Engineering College, (Autonomous), (Autonomous),**

**Thandalam, Chennai - 602 105 Thandalam, Chennai - 602 105**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

Horse race betting management system is a simple console application developed in java with use of java swings. The user can perform creating a account ,basic betting operations, bettings on a horse in a horse race can be done, adding money in the wallet, adding a horse in the race

All the betts will be stored in the database and after the game gets over the data will be accessed from the database through server and winners will be decided.Its a application with client - server architecture model.

Key words: java swings , database management system

**TABLE OF CONTENTS**

**1. INTRODUCTION**

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

1.4 SOFTWARE DESCRIPTION

1.5 LANGUAGES

**2. SURVEY OF TECHNOLOGIES**

2.1 SQL

2.2 JAVA

**3. REQUIREMENTS AND ANALYSIS**

3.1 REQUIREMENT SPECIFICATION

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM

3.4 ER DIAGRAM

3.5 NORMALIZATION

**4.PROGRAM CODE**

**5. RESULTS AND DISCUSSION**

**6.CONCLUSION**

**7.REFERENCES**

1. **INTRODUCTION**

**1.1 introduction**

TITLE: Data-driven online legal HORSE RACE betting , building a platform for wagering enthusiasts. This application “HORSE RACE wagers” an innovative online betting application designed specifically for horse race events. This is a online web-application for entertainment purposes only. This application can be addictive and there could be loss of money. The company won’t take any responsibility for their actions or decisions of individuals. Through this user interface,individuals can conveniently place bets on a variety of horse races and track their outcomes in real time. The user needs to make an account once on the server giving some useful personal data and with the next step succeeding to create an account. Betting a horse to win implies that you believe the horse will finish first and are willing to back your opinion with hard-earned cash. If your horse wins, the odds on the horse will determine how much you collect. For eg in a horse race,betting a horse to place means you think it will finish first or second. If either of those outcomes materialize, you will collect .Backing (the same as betting) a horse to show means that you believe it will finish somewhere in the top three. If it does, you will collect.It should come as no surprise that betting to place returns less than a successful win bet: You have, after all, effectively doubled your chances of collecting. Likewise, a show bet returns less than a bet to win or place.All the betts will be stored in the database and after the game gets over the data will be accessed from the database through server and winners will be decided.Its a application with client - server architecture model.

**1.2 objectives**

.

**Developing a horse race betting application involves multiple objectives to ensure the application is user-friendly, secure, and engaging. Here are some key objectives to consider:**

**1. User Experience (UX) and Interface Design**

Intuitive Design: Create a user-friendly interface that is easy to navigate for both novice and experienced users.

Responsive Design:Ensure the application works seamlessly across various devices (mobile, tablet, desktop).

Engaging Visuals: Use high-quality graphics and animations to enhance user engagement.

Personalization: Offer personalized recommendations and features based on user preferences and betting history.

**2. Functionality**

Real-time Data: Provide live updates on horse races, including odds, results, and other relevant statistics.

Betting Options:Offer a variety of betting types (e.g., win, place, show, exacta, trifecta) to cater to different user preferences.

Account Management: Allow users to create and manage their accounts, including viewing betting history and managing funds.

Notifications: Implement push notifications for race updates, bet outcomes, promotions, and other relevant information.

**3. Security**

Data Protection:Ensure robust data encryption and secure user data storage to protect against breaches.

Secure Transactions: Implement secure payment gateways for deposits and withdrawals.

Authentication:Use multi-factor authentication to enhance account security.

Regulatory Compliance: Adhere to legal and regulatory requirements for online betting in different regions.

**4. Performance and Reliability**

Scalability:Design the application to handle high traffic volumes, especially during major racing events.

Low Latency: Ensure minimal delay in real-time updates and transactions.

Reliability: Maintain high uptime and quickly address any technical issues or outages.

**5. Customer Support**

Help Center: Provide a comprehensive help center with FAQs, tutorials, and guides.

Live Support:Offer live chat, email, and phone support to assist users with any issues.

Community Engagement: Foster a community through forums or social media integration where users can share tips and experiences.

**6. Marketing and Growth**

Promotions and Bonuses: Offer promotions, bonuses, and loyalty programs to attract and retain users.

User Acquisition: Implement effective marketing strategies to acquire new users, including social media campaigns, partnerships, and influencer marketing.

Feedback and Improvement: Collect user feedback regularly and use it to improve the application continuously.

**7. Analytics and Reporting**

User Analytics: Track user behavior to understand preferences and improve the user experience.

Betting Analytics: Provide users with access to detailed analytics and insights to help them make informed betting decisions.

Performance Reports: Generate reports on the application's performance, user engagement, and financial metrics.

By focusing on these objectives, you can develop a comprehensive and competitive horse race betting application that meets user needs and stands out in the market.

**1.3 MODULES**

To achieve the objectives for a horse race betting application, the following modules should be developed. Each module addresses specific functionalities and collectively they form the core of the application:

**1. User Management Module**

User Registration and Login: Allows users to create accounts and log in securely.

Profile Management: Enables users to update their personal information, preferences, and settings.

Authentication and Security: Implements multi-factor authentication and password management.

**2. Betting Module**

Betting Options: Provides various betting types (win, place, show, exacta, trifecta, etc.).

Bet Placement:Facilitates the process of placing bets on selected races and horses.

Bet Tracking: Allows users to view and manage their active and past bets.

**3. Race Information Module**

Race Schedules: Displays upcoming races, including dates, times, and locations.

Horse and Jockey Profiles: Provides detailed information about the horses, jockeys, trainers, and their past performances.

Live Updates: Offers real-time race updates, including live streaming or

minute-by-minute text updates.

**4. Odds and Statistics Module**

Real-time Odds: Shows current betting odds for different horses and races.

Historical Data:Provides access to historical performance data and statistics.

Predictive Analytics: Uses algorithms to predict outcomes and suggest bets.

**5. Payment and Wallet Module**

Deposits and Withdrawals: Supports secure transactions for depositing funds into and withdrawing from user accounts.

Payment Methods: Integrates various payment options like credit/debit cards, e-wallets, and bank transfers.

Transaction History: Allows users to view their complete transaction history.

**6. Notification and Alerts Module**

Push Notifications: Sends updates on race results, betting outcomes, and promotional offers.

Email and SMS Alerts: Provides additional communication channels for important updates and reminders.

Customizable Alerts: Allows users to set their preferences for notifications and alerts.

**7. Customer Support Module**

Help Center: Contains FAQs, tutorials, and troubleshooting guides.

Live Chat Support: Offers real-time assistance through chat.

Support Ticketing System: Manages user queries and issues through a ticket-based system.

**8. Marketing and Promotion Module**

Promotional Campaigns: Manages marketing campaigns, including bonuses, free bets, and loyalty rewards.

Referral Program: Allows users to refer friends and earn rewards.

User Acquisition Analytics:Tracks the effectiveness of marketing campaigns and user acquisition strategies.

**9. Analytics and Reporting Module**

User Analytics:Tracks user behavior, preferences, and engagement metrics.

Betting Analytics: Provides insights into betting trends and user success rates.

Financial Reports: Generates reports on revenue, transactions, and other financial metrics.

Performance Monitoring: Continuously monitors the performance of the application.

**10. Admin and Management Module**

User Management: Allows admins to manage user accounts and permissions.

Content Management: Enables updating of race schedules, horse profiles, and other content.

System Monitoring: Provides tools for monitoring system health and performance.

Compliance Management: Ensures the application adheres to legal and regulatory requirements.

**11. Integration Module**

API Integration: Connects with third-party services for live data feeds, payment gateways, and marketing tools.

Social Media Integration: Allows users to share their betting activities and achievements on social media platforms.

Third-party Analytics: Integrates with analytics platforms for advanced data analysis.

By developing these modules, you can ensure that the horse race betting application is comprehensive, scalable, and able to meet the diverse needs of its users.

**2. SURVEY OF TECHNOLOGIES**

To create a horse race betting application using Java Swings and MySQL, you need to consider a few key technology components and design patterns. Here is a detailed survey of the technology stack and architecture:

**1. Java Swings for GUI**

Java Swings is a part of Java Foundation Classes (JFC) used to create window-based applications. It's suitable for building the user interface of your horse race betting application.

**Advantages of Java Swings:**

-Rich Set of Components:\*\* Swings offer a wide range of GUI components like buttons, labels, text fields, tables, trees, etc.

Customizable: Components can be easily customized to create a modern and attractive user interface.

Cross-Platform: Applications built with Swings are platform-independent and can run on any system with JVM.

Event Handling: Robust event-handling capabilities to manage user interactions.

Key Components:

JFrame: Main window of the application.

JPanel: Container to hold other components.

JButton, JLabel, JTextField:Basic interactive components.

JTable: To display tabular data like race schedules, betting options, and results.

JComboBox: To provide dropdown menus for selecting races, horses, etc.

JScrollPane:To handle scrolling within components.

**2. MySQL for Database**

MySQL is a popular relational database management system that is efficient for handling structured data and supports complex queries.

**Advantages of MySQL:**

-Reliability: Stable and reliable for managing large datasets.

Scalability:Can handle increased load by scaling vertically or horizontally.

Security: Provides robust security features to protect data.

Integration:Easily integrates with Java applications through JDBC.

**Key Database Components:**

User Table: Stores user information including account details and preferences.

Race Table: Contains information about upcoming and past races.

Horse Table: Stores data about horses, their profiles, and performance history.

Bet Table: Manages betting records, including user bets, odds, and outcomes.

Transaction Table: Keeps track of financial transactions like deposits and withdrawals.

**3. JDBC for Database Connectivity**

Java Database Connectivity (JDBC) is an API for connecting and executing queries on a database.

**Key Concepts:**

DriverManager: Manages a list of database drivers and establishes a connection to the database.

Connection: Interface that provides methods for interacting with the database.

Statement and PreparedStatement: Used to execute SQL queries and retrieve results.

ResultSet: Represents the result set of a query and allows traversal of the data.

**4. MVC Architecture**

Model-View-Controller (MVC) architecture is ideal for separating the application's logic, UI, and data management.

**Components:**

Model: Represents the data and business logic. For instance, classes representing User, Bet, Horse, etc.

View: The GUI components created using Java Swings that display the data to the user.

Controller: Handles the interaction between Model and View, processing user inputs, updating the model, and refreshing the view.

**5. Additional Tools and Libraries**

Apache Commons: Useful for various utility functions like String manipulation, IO operations, etc.

Log4j or SLF4J: For logging application events, useful for debugging and monitoring.

JUnit:For unit testing the application to ensure code quality and reliability.

SwingWorker: For handling background tasks and long-running operations without freezing the UI.

**Example Application Flow**

1. User Interface:

- Launch the application and display the main window (JFrame).

- Provide login and registration forms (JPanel with JTextField, JPasswordField, JButton).

- Main dashboard showing race schedules (JTable) and betting options (JComboBox, JButton).

**2. Database Interaction:**

- Establish a connection to the MySQL database using JDBC.

- Execute queries to fetch and display race information, odds, and user bets.

- Insert new bets and update user data based on interactions.

**3. Event Handling:**

- Handle user actions like placing a bet, checking results, and viewing transaction history.

- Use ActionListeners to manage button clicks and other user inputs.

**4. Data Processing:**

- Validate user inputs before processing bets.

- Calculate potential winnings based on the odds and update the user’s account balance.

**Requirements and Analysis for a Horse Race Betting Application**

**1. Requirements SPECIFICATIONS**

Functional Requirements

1. User Registration and Authentication:

- Users can create accounts and log in.

- Provide secure password storage and retrieval mechanisms.

- Implement multi-factor authentication for enhanced security.

**2. User Profile Management:**

- Users can update their personal information and preferences.

- Display user betting history and transaction details.

**3. Race Information:**

- Show a schedule of upcoming races with details such as date, time, location, and participating horses.

- Provide profiles of horses, jockeys, and trainers, including past performance data.

**4. Betting System:**

- Offer various types of bets (win, place, show, exacta, trifecta).

- Allow users to place, modify, and cancel bets.

- Display real-time odds and update them based on market conditions.

**5. Payment Processing:**

- Integrate secure payment gateways for deposits and withdrawals.

- Maintain a wallet system to manage user funds.

**6. Real-Time Updates:**

- Provide live updates on race progress and results.

- Notify users about bet outcomes and account changes.

**7. Notifications and Alerts:**

- Send push notifications, emails, or SMS for important events like race starts, results, and promotions.

- Allow users to customize their notification preferences.

**8. Customer Support:**

- Offer a help center with FAQs, guides, and troubleshooting.

- Provide live chat, email, and phone support.

**9. Admin Features:**

- Admins can manage user accounts, content, and system settings.

- Generate reports on user activity, betting trends, and financial performance.

Non-Functional Requirements

**1. Scalability:**

- The system should handle an increasing number of users and high traffic during major events.

**2. Performance:**

- Ensure fast loading times and minimal latency for real-time updates.

**3. Security:**

- Implement robust security measures to protect user data and transactions.

- Regularly update the system to address vulnerabilities.

**4. Usability:**

- Design a user-friendly interface that is intuitive and easy to navigate.

**5. Reliability:**

- Ensure high availability and minimal downtime.

- Provide mechanisms for data backup and recovery.

**6. Compliance:**

- Adhere to legal and regulatory requirements for online betting in various jurisdictions.

**2. Analysis**

Use Case Analysis

1. User Registration and Login:

- Actors: New User, Registered User

- Steps: User enters details → System validates → Account created/logged in.

2. Place Bet:

- Actors: Registered User

- Steps: User selects race and bet type → User enters bet amount → System validates funds → Bet placed and confirmed.

3. View Race Information:

- Actors: Registered User, Guest User

- Steps: User selects race → System displays race details and horse profiles.

4. Deposit Funds:

- Actors: Registered User

- Steps: User selects deposit option → Enters payment details → System processes payment → Funds added to wallet.

5. Withdraw Funds:

- Actors: Registered User

- Steps: User requests withdrawal → System validates → Funds transferred to user’s bank account.

Entity-Relationship (ER) Diagram

**1. Entities:**

-User: userID, username, password, email, balance, registrationDate.

- Race: raceID, date, time, location, status.

- Horse: horseID, name, age, jockey, trainer, performanceHistory.

- Bet: betID, userID, raceID, betType, amount, odds, result.

- Transaction: transactionID, userID, type (deposit/withdrawal), amount, date.

**2. Relationships:**

- User can place multiple Bets

- Race involves multiple Horses.

- User performs multiple Transactions

Sequence Diagrams

1. User Registration:

- User → System: Enter registration details.

- System → Database: Store user information.

- System → User: Confirm registration.

2. Place Bet:

- User → System: Select race and bet type.

- System → Database: Check user balance.

- Database → System: Return balance.

- System → User: Confirm bet placement.

3. Deposit Funds:

- User → System: Enter deposit amount and payment details.

- System → Payment Gateway: Process payment.

- Payment Gateway → System: Confirm transaction.

- System → Database: Update user balance.

- System → User: Confirm deposit.

Class Diagram

1. User Class:

- Attributes: userID, username, password, email, balance.

- Methods: register(), login(), updateProfile(), placeBet(), depositFunds(), withdrawFunds().

2. Race Class:

- Attributes: raceID, date, time, location, status.

- Methods: getRaceDetails(), updateRaceStatus().

3. Horse Class:

- Attributes: horseID, name, age, jockey, trainer, performanceHistory.

- Methods: getHorseDetails(), updatePerformanceHistory().

4. Bet Class:

- Attributes: betID, userID, raceID, betType, amount, odds, result.

- Methods: placeBet(), updateBetResult().

5. Transaction Class:

- Attributes: transactionID, userID, type, amount, date.

- Methods: processTransaction().

Data Flow Diagram (DFD)

1. Level 0:

- User → System: Registration, Login, Place Bet, View Race Information, Deposit/Withdraw Funds.

- System → Database: Store and retrieve data for the above actions.

2. Level 1:

- Registration: User inputs details → System validates and stores in Database.

- Betting: User selects bet → System checks balance → System stores bet details in Database.

- Transactions: User requests deposit/withdrawal → System processes payment → System updates Database.

By conducting a detailed requirements and analysis phase, you can ensure that the horse race betting application is well-structured, functional, and meets user needs efficiently. This foundation will guide the development process and help in building a robust application.

**architecture diagram**

****

## ER Diagram for Horse Race Betting App

**Here's an Entity-Relationship (ER) Diagram for your horse race betting application:**

**Entities:**

* **User: Represents a registered user of the application.**
  + **Attributes:**
    - **User ID (primary key)**
    - **Username**
    - **Password (hashed)**
    - **Name**
    - **Email**
    - **Balance**
* **Horse: Represents a horse participating in a race.**
  + **Attributes:**
    - **Horse ID (primary key)**
    - **Name**
    - **Age**
    - **Breed**
* **Race: Represents a scheduled horse race.**
  + **Attributes:**
    - **Race ID (primary key)**
    - **Race Name**
    - **Date**
    - **Time**
    - **Track (location)**
* **Bet: Represents a wager placed by a user on a specific horse in a race.**
  + **Attributes:**
    - **Bet ID (primary key)**
    - **User ID (foreign key referencing User)**
    - **Race ID (foreign key referencing Race)**
    - **Horse ID (foreign key referencing Horse)**
    - **Amount (amount wagered)**

**Relationships:**

* **User (1) : Many (N) Bets: A single user can place many bets.**
* **Race (1) : Many (N) Bets: A single race can have many bets placed on it.**
* **Horse (1) : Many (N) Bets: A single horse can be included in many bets.**

****

**Explanation:**

* **Rectangles represent entities.**
* **Diamonds represent relationships between entities.**
* **Lines connect entities and relationships.**
* **Cardinalities (1: one-to-one, N: many-to-many) indicate the number of instances of one entity that can relate to another entity instance.**
* **Primary keys are underlined.**
* **Foreign keys are shown in parentheses next to the attribute referencing another entity's primary key.**

**This ER diagram represents the core entities and their relationships for your horse race betting application. You can extend this model to include additional entities like race results, jockey information, or transaction history depending on your application's specific functionalities.**

**NORMALIZATION**

In your horse race betting application's database design, normalization is crucial to reduce data redundancy, improve data integrity, and ensure efficient data manipulation. Here's how normalization can be applied to the tables in your ER diagram:

**1. Unnormalized Table:**

Initially, your tables might look like a single table containing all the information:

**Race\_ID | Race\_Name | Date | Time | Track | User\_ID | Username | Horse\_ID | Horse\_Name | Amount (Bet)**

**------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- | --------**

**1 | Race X | 2024-05-25 | 10:00 | Track A | 1 | user123 | 2 | Horse A | 100**

**2 | Race Y | 2024-05-27 | 12:00 | Track B | 2 | user456 | 3 | Horse B | 50**

**This structure has redundancy:**

* User information (Username) is repeated for every bet placed by the user.
* Horse information (Horse\_Name) is repeated for every bet on the same horse.

**2. 1st Normal Form (1NF):**

The first step in normalization is to ensure each table cell contains a single atomic value (indivisible unit of data) and eliminate repeating groups. This can be achieved by separating the data into three tables:

* Race table: | Race\_ID (PK) | Race\_Name | Date | Time | Track | |---|---|---|---|---| | 1 | Race X | 2024-05-25 | 10:00 | Track A | | 2 | Race Y | 2024-05-27 | 12:00 | Track B |
* User table: | User\_ID (PK) | Username | Password | Name | Email | Balance | |---|---|---|---|---|---| | 1 | user123 | (hashed password) | John Doe | [email address removed] | 1000 | | 2 | user456 | (hashed password) | Jane Doe | [email address removed] | 500 |
* Bet table: | Bet\_ID (PK) | User\_ID (FK) | Race\_ID (FK) | Horse\_ID (FK) | Amount | |---|---|---|---|---| | 1 | 1 | 1 | 2 | 100 | | 2 | 2 | 2 | 3 | 50 |

**(PK: Primary Key, FK: Foreign Key)**

Foreign keys establish relationships between tables. In the Bet table, User\_ID and Race\_ID reference the primary keys in the User and Race tables respectively, indicating which user placed the bet and on which race. Similarly, Horse\_ID is a foreign key referencing the Horse table (which can be added later) to show which horse the bet is on.

**Benefits of 1NF:**

* Eliminates repeating groups.
* Reduces data redundancy.

**Limitations of 1NF:**

* Partial dependencies still exist (e.g., updating Horse\_Name in the Race table would require updating it in all Bets on that horse).

**3. Higher Normal Forms (2NF, 3NF):**

Further normalization can be achieved to minimize data redundancy and improve data integrity. However, for a simple betting application, 1NF might be sufficient. If your application grows in complexity, you can consider 2NF or 3NF to address partial dependencies and transitive dependencies that might arise. These involve decomposing tables further based on specific rules to ensure a more robust database structure.

**PROGRAM CODE**

**ADD BALANCE.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class AddBalance {

private JFrame selfJframe;

private JTextField cardOwner;

private JTextField cardNumber;

private JTextField cardExpiry;

private JTextField cardCvv;

private JTextField cardBalance;

private JButton updButton;

private int userId;

//Database Connection

Connection con = null;

PreparedStatement pst;

ResultSet rs;

/\*\*

\* Launch the application.

\*/

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

AddBalance uLogin = new AddBalance();

uLogin.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

public JFrame getFrame() {

return selfJframe;

}

/\*\*

\* Create the application.

\*/

public AddBalance() {

initialize();

Connect();

loadDefaultUserInfo();

}

public AddBalance(int userId) {

this.userId = userId;

initialize();

Connect();

loadDefaultUserInfo();

}

public int getUserId() {

return this.userId;

}

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void clear() {

cardNumber.setEditable(false);

cardExpiry.setEditable(false);

cardCvv.setEditable(false);

cardOwner.setEditable(false);

cardBalance.setEditable(false);

cardBalance.setEditable(false);

updButton.setVisible(false);

}

public void loadDefaultUserInfo() {

int dbUserID = 1;

try {

String sql = "SELECT \* FROM cards WHERE user\_id=?";

pst = con.prepareStatement(sql);

pst.setInt(1, dbUserID);

ResultSet rs = pst.executeQuery();

if (rs.next()) {

cardOwner.setText(rs.getString("card\_owner"));

cardNumber.setText(rs.getString("card\_number"));

cardExpiry.setText(rs.getString("card\_expiry"));

cardCvv.setText(rs.getString("card\_cvv"));

cardBalance.setText("20.10");

cardOwner.setEditable(false);

cardNumber.setEditable(false);

cardExpiry.setEditable(false);

cardCvv.setEditable(false);

} else {

JOptionPane.showMessageDialog(null, "Card information is not available");

}

} catch (SQLException sqlException) {

JOptionPane.showMessageDialog(null, "Unable to pull card information");

sqlException.printStackTrace();

}

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Add Balance");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Card Owner");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1.setBounds(30, 50, 150, 24);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("Card Number");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_1.setBounds(30, 90, 150, 24);

panel.add(lblNewLabel\_1\_1);

JLabel lblNewLabel\_1\_2 = new JLabel("Card Expiry");

lblNewLabel\_1\_2.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_2.setBounds(30, 130, 150, 24);

panel.add(lblNewLabel\_1\_2);

JLabel lblNewLabel\_1\_3 = new JLabel("Card CVV");

lblNewLabel\_1\_3.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_3.setBounds(30, 180, 150, 24);

panel.add(lblNewLabel\_1\_3);

JLabel lblNewLabel\_1\_4 = new JLabel("New Balance");

lblNewLabel\_1\_4.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_4.setBounds(30, 230, 150, 24);

panel.add(lblNewLabel\_1\_4);

cardOwner = new JTextField();

cardOwner.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardOwner.setBounds(150, 50, 287, 24);

panel.add(cardOwner);

cardOwner.setColumns(10);

cardNumber = new JTextField();

cardNumber.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardNumber.setColumns(10);

cardNumber.setBounds(150, 90, 287, 24);

panel.add(cardNumber);

cardExpiry = new JTextField();

cardExpiry.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardExpiry.setColumns(10);

cardExpiry.setBounds(150, 130, 287, 24);

panel.add(cardExpiry);

cardCvv = new JTextField();

cardCvv.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardCvv.setColumns(10);

cardCvv.setBounds(150, 170, 287, 24);

panel.add(cardCvv);

cardBalance = new JTextField();

cardBalance.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardBalance.setColumns(10);

cardBalance.setBounds(150, 210, 287, 24);

panel.add(cardBalance);

updButton = new JButton("Update");

updButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

double dbBalance = Double.parseDouble(cardBalance.getText());

if (dbBalance == 0.0 ) {

JOptionPane.showMessageDialog(null, "Please Enter Balance");

cardBalance.requestFocus();

return;

}

try {

String sql = "SELECT balance FROM balance WHERE user\_id=?";

pst = con.prepareStatement(sql);

pst.setInt(1, getUserId());

ResultSet rs = pst.executeQuery();

if (rs.next()) {

String sql\_1 = "UPDATE balance SET balance=? WHERE user\_id=?";

pst = con.prepareStatement(sql\_1);

pst.setDouble(1, rs.getDouble("balance") + dbBalance);

pst.setInt(2, getUserId());

pst.executeUpdate();

JOptionPane.showMessageDialog(null, "Balance updated successfully");

clear();

} else {

String sql\_2 = "INSERT INTO balance (user\_id, balance) VALUES (?,?)";

pst = con.prepareStatement(sql\_2);

pst.setInt(1, getUserId());

pst.setDouble(2, dbBalance);

pst.executeUpdate();

JOptionPane.showMessageDialog(null, "Balance added successfully");

clear();

}

} catch (SQLIntegrityConstraintViolationException e1) {

JOptionPane.showMessageDialog(null, "Duplicate Payment. Failed to update balance");

}catch (SQLException e1) {

JOptionPane.showMessageDialog(null, "Failed to update balance");

}

}

});

updButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

updButton.setBounds(150, 300, 89, 23);

panel.add(updButton);

JButton backButton = new JButton("Back");

backButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

PlayBet pBet = new PlayBet(getUserId());

pBet.getFrame().setVisible(true);

}

});

backButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

backButton.setBounds(300, 300, 89, 23);

panel.add(backButton);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**app info.java**

package com.horse.race.tracker;

public class AppConstants {

public static String DB\_SOURCE = "jdbc:mysql://localhost:3306/horserace";

public static String DB\_USER = "admin";

public static String DB\_PASSWORD = "admin";

public static String DB\_DRIVER = "com.mysql.cj.jdbc.Driver";

public static String LOGIN\_FETCH\_PASSWORD = "Select user\_name, password from users where user\_name=? and password=?";

public static String SUCCESS\_LOGIN = "You have successfully logged in";

public static String FAILED\_LOGIN = "Incorrect Credentials";

public static int REG\_FORM\_TXT\_SIZE = 16;

public static int REG\_FIELD\_TXT\_SIZE = 12;

public static int BET\_WIN\_PROPORTION = 5;

public static int BET\_LOSS\_PROPORTION = 1;

public static String ADMIN\_PASSWORD = "admin";

}

**CARDINFO.JAVA**

package com.horse.race.tracker;

import java.awt.EventQueue;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class CardInfo {

private JFrame selfJframe;

private JTextField cardOwner;

private JTextField cardNumber;

private JTextField cardExpiry;

private JTextField cardCvv;

private int userId;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

CardInfo uLogin = new CardInfo();

uLogin.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public JFrame getFrame() {

return selfJframe;

}

public CardInfo(int userId) {

this.userId = userId;

initialize();

Connect();

loadDefaultUserInfo();

}

public int getUserId() {

return this.userId;

}

/\*\*

\* Create the application.

\*/

public CardInfo() {

initialize();

Connect();

loadDefaultUserInfo();

//loadData();

}

//Database Connection

Connection con = null;

PreparedStatement pst;

ResultSet rs;

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

// Clear All

public void clear() {

cardNumber.setText("");

cardExpiry.setText("");

cardCvv.setText("");

cardOwner.requestFocus();

}

public void loadDefaultUserInfo() {

cardOwner.setText("Card Owner 1");

cardNumber.setText("1234-5678-9012-3456");

cardExpiry.setText("02/29");

cardCvv.setText("123");

cardOwner.requestFocus();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Horse Race Tracker");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Card Owner");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1.setBounds(30, 50, 150, 24);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("Card Number");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_1.setBounds(30, 90, 150, 24);

panel.add(lblNewLabel\_1\_1);

JLabel lblNewLabel\_1\_2 = new JLabel("Card Expiry");

lblNewLabel\_1\_2.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_2.setBounds(30, 130, 150, 24);

panel.add(lblNewLabel\_1\_2);

JLabel lblNewLabel\_1\_3 = new JLabel("Card CVV");

lblNewLabel\_1\_3.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_3.setBounds(30, 180, 150, 24);

panel.add(lblNewLabel\_1\_3);

cardOwner = new JTextField();

cardOwner.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardOwner.setBounds(150, 50, 287, 24);

panel.add(cardOwner);

cardOwner.setColumns(10);

cardNumber = new JTextField();

cardNumber.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardNumber.setColumns(10);

cardNumber.setBounds(150, 90, 287, 24);

panel.add(cardNumber);

cardExpiry = new JTextField();

cardExpiry.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardExpiry.setColumns(10);

cardExpiry.setBounds(150, 130, 287, 24);

panel.add(cardExpiry);

cardCvv = new JTextField();

cardCvv.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardCvv.setColumns(10);

cardCvv.setBounds(150, 170, 287, 24);

panel.add(cardCvv);

JButton btnSave = new JButton("Register");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String dbcardOwner = cardOwner.getText();

String dbcardNumber = cardNumber.getText();

String dbcardExpiry = cardExpiry.getText();

String dbcardCvv = cardCvv.getText();

if (dbcardOwner == null || dbcardOwner.isEmpty() || dbcardOwner.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Card Owner");

cardOwner.requestFocus();

return;

}

if (dbcardNumber == null || dbcardNumber.isEmpty() || dbcardNumber.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Card Number");

cardNumber.requestFocus();

return;

}

if (dbcardExpiry == null || dbcardExpiry.isEmpty() || dbcardExpiry.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Card Expiry");

cardExpiry.requestFocus();

return;

}

if (dbcardCvv == null || dbcardCvv.isEmpty() || dbcardCvv.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Card CVV");

cardCvv.requestFocus();

return;

}

try {

String sql = "INSERT INTO cards (card\_owner, card\_number, card\_expiry, card\_cvv, user\_id) values (?,?,?,?,?)";

pst = con.prepareStatement(sql);

pst.setString(1, dbcardOwner);

pst.setString(2, dbcardNumber);

pst.setString(3, dbcardExpiry);

pst.setString(4, dbcardCvv);

pst.setInt(5, getUserId());

pst.executeUpdate();

JOptionPane.showMessageDialog(null, "Congratulations. Your registration is complete. Please login");

getFrame().dispose();

UserLogin uLogin = new UserLogin();

uLogin.getFrame().setVisible(true);

} catch (SQLIntegrityConstraintViolationException e1) {

JOptionPane.showMessageDialog(null, "Duplicate card details are not allowed");

loadDefaultUserInfo();

}catch (SQLException e1) {

e1.printStackTrace();

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(250, 300, 89, 23);

panel.add(btnSave);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**HORSEINFO.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class HorseInfo {

private JFrame selfJframe;

private JTextField horseOwner;

private JTextField horseName;

private int userId;

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

HorseInfo addHorse = new HorseInfo();

addHorse.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

public JFrame getFrame() {

return selfJframe;

}

/\*\*

\* Create the application.

\*/

public HorseInfo() {

initialize();

Connect();

loadDefaultUserInfo();

}

public HorseInfo(int userId) {

this.userId = userId;

initialize();

Connect();

loadDefaultUserInfo();

}

public int getUserId() {

return this.userId;

}

Connection con = null;

PreparedStatement pst;

ResultSet rs;

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void clear() {

horseName.setText("");

horseOwner.requestFocus();

}

public void loadDefaultUserInfo() {

horseOwner.setText("Owner 1");

horseName.setText("Horse 1");

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Add a Horse");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Horse Name");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1.setBounds(30, 50, 150, 24);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("Horse Owner");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_1.setBounds(30, 90, 150, 24);

panel.add(lblNewLabel\_1\_1);

horseName = new JTextField();

horseName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

horseName.setBounds(150, 50, 287, 24);

panel.add(horseName);

horseName.setColumns(10);

horseOwner = new JTextField();

horseOwner.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

horseOwner.setColumns(10);

horseOwner.setBounds(150, 90, 287, 24);

panel.add(horseOwner);

JButton btnSave = new JButton("Submit");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String dbhorseOwner = horseOwner.getText();

String dbhorseName = horseName.getText();

if (dbhorseOwner == null || dbhorseOwner.isEmpty() || dbhorseOwner.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter First Name");

horseOwner.requestFocus();

return;

}

if (dbhorseName == null || dbhorseName.isEmpty() || dbhorseName.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Last Name");

horseName.requestFocus();

return;

}

try {

String sql = "INSERT INTO horse (horse\_name, horse\_owner) values (?,?)";

pst = con.prepareStatement(sql);

pst.setString(1, dbhorseName);

pst.setString(2, dbhorseOwner);

pst.executeUpdate();

JOptionPane.showMessageDialog(null, "Horse information added successfully");

} catch (SQLIntegrityConstraintViolationException e1) {

JOptionPane.showMessageDialog(null, "Duplicate horse details are not allowed");

}catch (SQLException e1) {

e1.printStackTrace();

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(150, 200, 100, 30);

panel.add(btnSave);

JButton backButton = new JButton("Back");

backButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

PlayBet pBet = new PlayBet(getUserId());

pBet.getFrame().setVisible(true);

}

});

backButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

backButton.setBounds(300, 200, 100, 30);

panel.add(backButton);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**PLAYBET.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import com.horse.race.tracker.models.HorseModel;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.AbstractButton;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import java.sql.\*;

import java.util.ArrayList;

import java.util.Enumeration;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Random;

import java.awt.event.\*;

import java.awt.event.ActionEvent;

import javax.swing.JRadioButton;

public class PlayBet {

private JFrame selfJframe;

private JTextField betAmount;

private int userId;

private List<HorseModel> horseList = new ArrayList<>();

private Map<String, Integer> horseIDList = new HashMap<>();

ButtonGroup bg = new ButtonGroup();

Connection con = null;

PreparedStatement pst;

ResultSet rs;

/\*\*

\* Launch the application.

\*/

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

PlayBet pBet = new PlayBet();

pBet.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

public JFrame getFrame() {

return selfJframe;

}

public PlayBet(int userId) {

this.userId = userId;

Connect();

loadDefaultUserInfo();

initialize();

}

public int getUserId() {

return this.userId;

}

public PlayBet() {

Connect();

loadDefaultUserInfo();

initialize();

}

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void loadDefaultUserInfo() {

try {

String sql = "SELECT \* FROM horse";

pst = con.prepareStatement(sql);

ResultSet rs = pst.executeQuery();

while (rs.next()) {

HorseModel horse = new HorseModel();

horse.setHorseName(rs.getString("horse\_name"));

horse.setHorseOwner(rs.getString("horse\_owner"));

horseList.add(horse);

}

} catch (SQLException sqlException) {

JOptionPane.showMessageDialog(null, "Unable to pull card information");

sqlException.printStackTrace();

}

}

public String getSelectedHorse()

{

for (Enumeration<AbstractButton> buttons = bg.getElements(); buttons.hasMoreElements();) {

AbstractButton button = buttons.nextElement();

if (button.isSelected()) {

return button.getText();

}

}

return null;

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Play a Bet");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

int wInc\_txt = 0;

int horseId = 0;

for(HorseModel horse: horseList) {

wInc\_txt += 40;

JRadioButton rButton = new JRadioButton(horse.getHorseName());

rButton.setBounds(250, wInc\_txt, 100, 24);

bg.add(rButton);

panel.add(rButton);

horseIDList.put( horse.getHorseName(), ++horseId);

}

JLabel lblNewLabel\_1\_4 = new JLabel("Bet Amount");

lblNewLabel\_1\_4.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_4.setBounds(400, 90, 287, 24);

panel.add(lblNewLabel\_1\_4);

betAmount = new JTextField();

betAmount.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

betAmount.setColumns(10);

betAmount.setBounds(400, 110, 100, 24);

panel.add(betAmount);

JButton btnSave = new JButton("Place Bet");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String selectedHorse = getSelectedHorse();

double dbBetAmount = 0;

try {

dbBetAmount = Double.parseDouble(betAmount.getText());

}catch(Exception ex)

{

JOptionPane.showMessageDialog(null, "Please enter valid bet amount");

betAmount.requestFocus();

return;

}

if (selectedHorse == null ) {

JOptionPane.showMessageDialog(null, "Please Select a horse");

return;

}

try {

String sql = "SELECT balance FROM balance WHERE user\_id=?";

pst = con.prepareStatement(sql);

pst.setInt(1, getUserId());

ResultSet rs = pst.executeQuery();

if (rs.next()) {

if( rs.getDouble("balance") < dbBetAmount ) {

JOptionPane.showMessageDialog(null, "Insufficient Balance. Please add balance");

return;

}

int wonHorse = new Random().nextInt(horseList.size()) + 1;

double newBalance = (wonHorse == horseIDList.get(selectedHorse)) ?

rs.getDouble("balance") + dbBetAmount \*

AppConstants.BET\_WIN\_PROPORTION:

rs.getDouble("balance") - dbBetAmount \*

AppConstants.BET\_LOSS\_PROPORTION;

String sql\_1 = "UPDATE balance SET balance=? WHERE user\_id=?";

pst = con.prepareStatement(sql\_1);

pst.setDouble(1, newBalance);

pst.setInt(2, getUserId());

pst.executeUpdate();

String wonHorseName = null;

for (String key : horseIDList.keySet()) {

if(horseIDList.get(key) == wonHorse) {

wonHorseName = key;

}

}

if( (wonHorse == horseIDList.get(selectedHorse)) ) {

JOptionPane.showMessageDialog(null, "Congratulations. You won." + " Won Horse: " + wonHorseName);

}else {

JOptionPane.showMessageDialog(null, "Better Luck next time" + " Won Horse: " + wonHorseName);

}

} else {

JOptionPane.showMessageDialog(null, "User not authorized");

}

}catch(Exception ex) {

ex.printStackTrace();

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(300, 300, 100, 30);

panel.add(btnSave);

JButton balButton = new JButton("Add Balance");

balButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

AddBalance aBalance = new AddBalance(getUserId());

aBalance.getFrame().setVisible(true);

}

});

balButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

balButton.setBounds(5, 10, 130, 30);

panel.add(balButton);

JButton playBet = new JButton("Play Bet");

playBet.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

}

});

playBet.setFont(new Font("Tahoma", Font.PLAIN, 14));

playBet.setBounds(5, 70, 130, 30);

panel.add(playBet);

JButton viewBalButton = new JButton("View Balance");

viewBalButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

ViewBalance vBal = new ViewBalance(getUserId());

vBal.getFrame().setVisible(true);

}

});

viewBalButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

viewBalButton.setBounds(5, 130, 130, 30);

panel.add(viewBalButton);

JButton addHorse = new JButton("Add Horse");

addHorse.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

HorseInfo addHorse = new HorseInfo(getUserId());

addHorse.getFrame().setVisible(true);

}

});

addHorse.setFont(new Font("Tahoma", Font.PLAIN, 14));

addHorse.setBounds(5, 190, 130, 30);

panel.add(addHorse);

JButton viewSumm = new JButton("View Summary");

viewSumm.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

ViewSummary vSumm = new ViewSummary(getUserId());

vSumm.getFrame().setVisible(true);

}

});

viewSumm.setFont(new Font("Tahoma", Font.PLAIN, 14));

viewSumm.setBounds(5, 250, 130, 30);

panel.add(viewSumm);

JButton closeButt = new JButton("Logout");

closeButt.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

UserLogin uLogin = new UserLogin();

uLogin.getFrame().setVisible(true);

}

});

closeButt.setFont(new Font("Tahoma", Font.PLAIN, 14));

closeButt.setBounds(430, 5, 90, 30);

panel.add(closeButt);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**USERLOGIN.JAVA**

package com.horse.race.tracker;

import java.awt.EventQueue;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class UserLogin {

private JFrame selfJframe;

private JTextField userName;

private JTextField password;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UserLogin uLogin = new UserLogin();

uLogin.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public JFrame getFrame() {

return selfJframe;

}

public UserLogin() {

initialize();

Connect();

loadDefaultUserInfo();

}

Connection con = null;

PreparedStatement pst;

ResultSet rs;

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

// Clear All

public void clear() {

userName.setText("");

password.setText("");

}

public void loadDefaultUserInfo() {

userName.setText("admin");

password.setText("admin");

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Race Tracker Login");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 200, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1\_2 = new JLabel("User Name");

lblNewLabel\_1\_2.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_2.setBounds(30, 50, 150, 24);

panel.add(lblNewLabel\_1\_2);

JLabel lblNewLabel\_1\_3 = new JLabel("Password");

lblNewLabel\_1\_3.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_3.setBounds(30, 90, 150, 24);

panel.add(lblNewLabel\_1\_3);

JLabel lblNewLabel\_1\_4 = new JLabel("First Time User?");

lblNewLabel\_1\_4.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_4.setBounds(80, 280, 150, 24);

panel.add(lblNewLabel\_1\_4);

userName = new JTextField();

userName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

userName.setColumns(10);

userName.setBounds(150, 50, 200, 24);

panel.add(userName);

password = new JPasswordField();

password.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

password.setColumns(10);

password.setBounds(150, 90, 200, 24);

panel.add(password);

JButton btnSave = new JButton("Login");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String dbUserName = userName.getText();

String dbPassword = password.getText();

if (dbUserName == null || dbUserName.isEmpty() || dbUserName.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please User Name");

userName.requestFocus();

return;

}

if (dbPassword == null || dbPassword.isEmpty() || dbPassword.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Password");

password.requestFocus();

return;

}

try {

String sql = "SELECT id, user\_name, password FROM users WHERE user\_name=? and password=?";

pst = con.prepareStatement(sql);

pst.setString(1, dbUserName);

pst.setString(2, dbPassword);

ResultSet rs = pst.executeQuery();

if (rs.next()) {

getFrame().dispose();

PlayBet pBet = new PlayBet(rs.getInt("id"));

pBet.getFrame().setVisible(true);

} else {

JOptionPane.showMessageDialog(null, AppConstants.FAILED\_LOGIN);

}

} catch (SQLException e1) {

JOptionPane.showMessageDialog(null, AppConstants.FAILED\_LOGIN);

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(200, 140, 89, 30);

panel.add(btnSave);

JButton btnReg = new JButton("Register");

btnReg.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

UserRegistration uReg = new UserRegistration();

uReg.getFrame().setVisible(true);

}

});

btnReg.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnReg.setBounds(250, 275, 120, 30);

panel.add(btnReg);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**USERREGISTRATION.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class UserRegistration {

private JFrame selfJframe;

private JTextField firstName;

private JTextField lastName;

private JTextField userName;

private JTextField password;

private JTextField email;

private JTextField mobile;

public JFrame getFrame() {

return selfJframe;

}

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UserRegistration uLogin = new UserRegistration();

uLogin.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

/\*\*

\* Create the application.

\*/

public UserRegistration() {

initialize();

Connect();

loadDefaultUserInfo();

}

//Database Connection

Connection con = null;

PreparedStatement pst;

ResultSet rs;

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

// Clear All

public void clear() {

firstName.setText("");

lastName.setText("");

userName.setText("");

password.setText("");

email.setText("");

mobile.setText("");

firstName.requestFocus();

}

public void loadDefaultUserInfo() {

firstName.setText("First Name 1");

lastName.setText("Last Name 1");

userName.setText("admin");

password.setText("admin");

email.setText("test@email.com");

mobile.setText("1234567890");

firstName.requestFocus();

}

// Load Table

/\* public void loadData() {

try {

pst = con.prepareStatement("Select \* from users");

rs = pst.executeQuery();

table.setModel(DbUtils.resultSetToTableModel(rs));

} catch (Exception ex) {

ex.printStackTrace();

}

}\*/

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Horse Race Tracker");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("First Name");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, 16));

lblNewLabel\_1.setBounds(30, 50, 150, 24);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel\_1\_1 = new JLabel("Last Name");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_1.setBounds(30, 90, 150, 24);

panel.add(lblNewLabel\_1\_1);

JLabel lblNewLabel\_1\_2 = new JLabel("User Name");

lblNewLabel\_1\_2.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_2.setBounds(30, 130, 150, 24);

panel.add(lblNewLabel\_1\_2);

JLabel lblNewLabel\_1\_3 = new JLabel("Password");

lblNewLabel\_1\_3.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_3.setBounds(30, 180, 150, 24);

panel.add(lblNewLabel\_1\_3);

JLabel lblNewLabel\_1\_4 = new JLabel("Email ID");

lblNewLabel\_1\_4.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_4.setBounds(30, 220, 150, 24);

panel.add(lblNewLabel\_1\_4);

JLabel lblNewLabel\_1\_5 = new JLabel("Mobile Number");

lblNewLabel\_1\_5.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1\_5.setBounds(30, 255, 150, 24);

panel.add(lblNewLabel\_1\_5);

firstName = new JTextField();

firstName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

firstName.setBounds(150, 50, 287, 24);

panel.add(firstName);

firstName.setColumns(10);

lastName = new JTextField();

lastName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lastName.setColumns(10);

lastName.setBounds(150, 90, 287, 24);

panel.add(lastName);

userName = new JTextField();

userName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

userName.setColumns(10);

userName.setBounds(150, 130, 287, 24);

panel.add(userName);

password = new JPasswordField();

password.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

password.setColumns(10);

password.setBounds(150, 170, 287, 24);

panel.add(password);

email = new JTextField();

email.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

email.setColumns(10);

email.setBounds(150, 210, 287, 24);

panel.add(email);

mobile = new JTextField();

mobile.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

mobile.setColumns(10);

mobile.setBounds(150, 250, 287, 24);

panel.add(mobile);

JButton btnSave = new JButton("Next");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String dbFirstName = firstName.getText();

String dbLastName = lastName.getText();

String dbUserName = userName.getText();

String dbPassword = password.getText();

String dbEmail = email.getText();

String dbMobile = mobile.getText();

if (dbFirstName == null || dbFirstName.isEmpty() || dbFirstName.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter First Name");

firstName.requestFocus();

return;

}

if (dbLastName == null || dbLastName.isEmpty() || dbLastName.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Last Name");

lastName.requestFocus();

return;

}

if (dbUserName == null || dbUserName.isEmpty() || dbUserName.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter User Name");

userName.requestFocus();

return;

}

if (dbPassword == null || dbPassword.isEmpty() || dbPassword.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Password");

password.requestFocus();

return;

}

if (dbEmail == null || dbEmail.isEmpty() || dbEmail.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Email");

email.requestFocus();

return;

}

if (dbMobile == null || dbMobile.isEmpty() || dbMobile.trim().isEmpty()) {

JOptionPane.showMessageDialog(null, "Please Enter Mobile Number");

mobile.requestFocus();

return;

}

try {

String sql = "INSERT INTO users (first\_name, last\_name, user\_name, password, email, mobile) values (?,?,?,?,?,?)";

pst = con.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);

pst.setString(1, dbFirstName);

pst.setString(2, dbLastName);

pst.setString(3, dbUserName);

pst.setString(4, dbPassword);

pst.setString(5, dbEmail);

pst.setString(6, dbMobile);

pst.executeUpdate();

ResultSet rs = pst.getGeneratedKeys();

if(rs.next()) {

int userId = rs.getInt(1);

getFrame().dispose();

CardInfo cInfo = new CardInfo(userId);

cInfo.getFrame().setVisible(true);

}else {

JOptionPane.showMessageDialog(null, "Duplicate user names are not allowed");

loadDefaultUserInfo();

}

} catch (SQLIntegrityConstraintViolationException e1) {

JOptionPane.showMessageDialog(null, "Duplicate user names are not allowed");

loadDefaultUserInfo();

}catch (SQLException e1) {

e1.printStackTrace();

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(250, 300, 89, 23);

panel.add(btnSave);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**VIEWBALANCE.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class ViewBalance {

private JFrame selfJframe;

private JTextField cardBalance;

private int userId;

//Database Connection

Connection con = null;

PreparedStatement pst;

ResultSet rs;

/\*\*

\* Launch the application.

\*/

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

ViewBalance vBal = new ViewBalance();

vBal.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

public JFrame getFrame() {

return selfJframe;

}

public ViewBalance() {

initialize();

Connect();

}

public ViewBalance(int userId) {

this.userId = userId;

initialize();

Connect();

}

public int getUserId() {

return this.userId;

}

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("View Balance");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Available Balance");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FORM\_TXT\_SIZE));

lblNewLabel\_1.setBounds(30, 160, 150, 24);

panel.add(lblNewLabel\_1);

cardBalance = new JTextField();

cardBalance.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

cardBalance.setColumns(10);

cardBalance.setBounds(170, 160, 287, 24);

cardBalance.setEditable(false);

panel.add(cardBalance);

JButton btnSave = new JButton("View Balance");

btnSave.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

String sql = "SELECT \* FROM balance WHERE user\_id=?";

pst = con.prepareStatement(sql);

pst.setInt(1, getUserId());

ResultSet rs = pst.executeQuery();

if (rs.next()) {

cardBalance.setText(rs.getString("balance"));

} else {

JOptionPane.showMessageDialog(null, "Card information is not available");

}

} catch (SQLException sqlException) {

JOptionPane.showMessageDialog(null, "Unable to pull card information");

sqlException.printStackTrace();

}

}

});

btnSave.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnSave.setBounds(100, 300, 130, 30);

panel.add(btnSave);

JButton backButton = new JButton("Back");

backButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

PlayBet pBet = new PlayBet(getUserId());

pBet.getFrame().setVisible(true);

}

});

backButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

backButton.setBounds(300, 300, 130, 30);

panel.add(backButton);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**VIEWSUMMARY.JAVA**

package com.horse.race.tracker;

import javax.swing.JFrame;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import java.awt.Color;

import javax.swing.border.EtchedBorder;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class ViewSummary {

private JFrame selfJframe;

private JTextField adminPass;

private JTextField adminPassLbl;

private int userId;

//Database Connection

Connection con = null;

PreparedStatement pst;

ResultSet rs;

/\*\*

\* Launch the application.

\*/

/\* public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

ViewSummary vSumm = new ViewSummary();

vSumm.getFrame().setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

\*/

public JFrame getFrame() {

return selfJframe;

}

/\*\*

\* Create the application.

\*/

public ViewSummary() {

initialize();

Connect();

}

public ViewSummary(int userId) {

this.userId = userId;

initialize();

Connect();

}

public int getUserId() {

return this.userId;

}

public void Connect() {

try {

Class.forName(AppConstants.DB\_DRIVER);

con = DriverManager.getConnection(AppConstants.DB\_SOURCE, AppConstants.DB\_USER, AppConstants.DB\_PASSWORD);

} catch (Exception ex) {

ex.printStackTrace();

}

}

// Clear All

public void clear() {

adminPass.setText("admin");

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

selfJframe = new JFrame();

selfJframe.setTitle("Horse Race Tracker");

selfJframe.getContentPane().setFont(new Font("Tahoma", Font.PLAIN, 14));

selfJframe.getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("View Summary");

lblNewLabel.setForeground(Color.BLUE);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 16));

lblNewLabel.setBounds(350, 7, 259, 60);

selfJframe.getContentPane().add(lblNewLabel);

JPanel panel = new JPanel();

panel.setBorder(new EtchedBorder(EtchedBorder.LOWERED, null, null));

panel.setBounds(155, 50, 550, 350);

selfJframe.getContentPane().add(panel);

panel.setLayout(null);

adminPassLbl = new JTextField("Admin Password");

adminPassLbl.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

adminPassLbl.setColumns(10);

adminPassLbl.setBounds(70, 160, 100, 24);

adminPassLbl.setEditable(false);

panel.add(adminPassLbl);

adminPass = new JPasswordField();

adminPass.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

adminPass.setColumns(10);

adminPass.setBounds(200, 160, 100, 24);

adminPass.setText("admin");

panel.add(adminPass);

JButton viewBal = new JButton("View Summary");

viewBal.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String iAdminPass = adminPass.getText();

if ( !iAdminPass.equals(AppConstants.ADMIN\_PASSWORD) ) {

JOptionPane.showMessageDialog(null, "User not authorized");

adminPass.requestFocus();

return;

}

try {

JTextField fName = new JTextField("First Name");

fName.setFont(new Font("Tahoma", Font.BOLD, AppConstants.REG\_FIELD\_TXT\_SIZE));

//fName.setColumns(10);

fName.setBounds(30, 23, 120, 24);

fName.setEditable(false);

panel.add(fName);

JTextField balLabel = new JTextField("Balance");

balLabel.setFont(new Font("Tahoma", Font.BOLD, AppConstants.REG\_FIELD\_TXT\_SIZE));

balLabel.setBounds(150, 23, 150, 24);

balLabel.setEditable(false);

panel.add(balLabel);

String sql = "SELECT first\_name, balance FROM users u JOIN balance b ON u.Id = b.user\_id";

pst = con.prepareStatement(sql);

ResultSet rs = pst.executeQuery();

int wInc = 23;

while (rs.next()) {

wInc += 23;

JTextField dbFirstName = new JTextField(rs.getString("first\_name"));

dbFirstName.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

dbFirstName.setBounds(30, wInc, 120, 24);

dbFirstName.setEditable(false);

panel.add(dbFirstName);

JTextField dbBalLabel = new JTextField(rs.getString("balance"));

dbBalLabel.setFont(new Font("Tahoma", Font.PLAIN, AppConstants.REG\_FIELD\_TXT\_SIZE));

dbBalLabel.setBounds(150, wInc, 150, 24);

dbBalLabel.setEditable(false);

panel.add(dbBalLabel);

}

} catch (SQLException sqlException) {

JOptionPane.showMessageDialog(null, "Unable to pull dashboard information");

sqlException.printStackTrace();

}

viewBal.setVisible(false);

adminPass.setVisible(false);

adminPassLbl.setVisible(false);

}

});

viewBal.setFont(new Font("Tahoma", Font.PLAIN, 14));

viewBal.setBounds(150, 300, 130, 30);

panel.add(viewBal);

JButton backButton = new JButton("Back");

backButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

getFrame().dispose();

PlayBet pBet = new PlayBet(getUserId());

pBet.getFrame().setVisible(true);

}

});

backButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

backButton.setBounds(350, 300, 130, 30);

panel.add(backButton);

getFrame().setBounds(100, 100, 910, 522);

getFrame().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getFrame().setResizable(false);

}

}

**HORSE MODEL.JAVA**

package com.horse.race.tracker.models;

public class HorseModel {

private String horseName;

private String horseOwner;

public String getHorseName() {

return horseName;

}

public void setHorseName(String horseName) {

this.horseName = horseName;

}

public String getHorseOwner() {

return horseOwner;

}

public void setHorseOwner(String horseOwner) {

this.horseOwner = horseOwner;

}

}

**QUERIES**

CREATE DATABASE `horserace`;

CREATE TABLE `horserace`.`users`

(

`id` INT NOT NULL AUTO\_INCREMENT ,

`first\_name` VARCHAR(255) NOT NULL ,

`last\_name` VARCHAR(255) NOT NULL ,

`user\_name` VARCHAR(255) NOT NULL ,

`password` VARCHAR(255) NOT NULL ,

`email` VARCHAR(255) NOT NULL ,

`mobile` INT(15) NOT NULL ,

PRIMARY KEY (`id`),

UNIQUE `userIDX` (`user\_name`(255))

) ENGINE = InnoDB;

CREATE TABLE `horserace`.`cards`

(

`id` INT NOT NULL AUTO\_INCREMENT ,

`card\_owner` VARCHAR(255) NOT NULL ,

`card\_number` VARCHAR(20) NOT NULL ,

`card\_expiry` VARCHAR(5) NOT NULL ,

`card\_cvv` INT(3) NOT NULL ,

`user\_id` VARCHAR(100) NOT NULL ,

PRIMARY KEY (`id`),

UNIQUE (`user\_id`)

) ENGINE = InnoDB;

CREATE TABLE `horserace`.`horse`

(

`id` INT NOT NULL AUTO\_INCREMENT ,

`horse\_name` VARCHAR(100) NOT NULL ,

`horse\_owner` VARCHAR(100) NOT NULL ,

PRIMARY KEY (`id`),

UNIQUE (`horse\_name`)

) ENGINE = InnoDB;

CREATE TABLE `horserace`.`balance`

(

`id` INT NOT NULL AUTO\_INCREMENT ,

`user\_id` INT NOT NULL ,

`balance` VARCHAR(100) NOT NULL ,

PRIMARY KEY (`id`), UNIQUE (`user\_id`)

) ENGINE = InnoDB;

**5. RESULTS AND DISCUSSION**

**Results**

* The chosen architecture utilizes a three-tier structure (Presentation Tier, Business Logic Tier, Data Access Tier) with clear separation of concerns, promoting maintainability and reusability.
* The ER diagram effectively captures the core entities (User, Horse, Race, Bet) and their relationships, providing a foundation for data modeling.
* Normalization helps eliminate redundancy, improve data integrity, and ensure efficient data manipulation within the database.
* The requirement specifications outline the essential functionalities for a horse race betting application, covering user management, race management, betting functionalities, account management, and non-functional requirements like performance, security, and usability.

**Discussion**

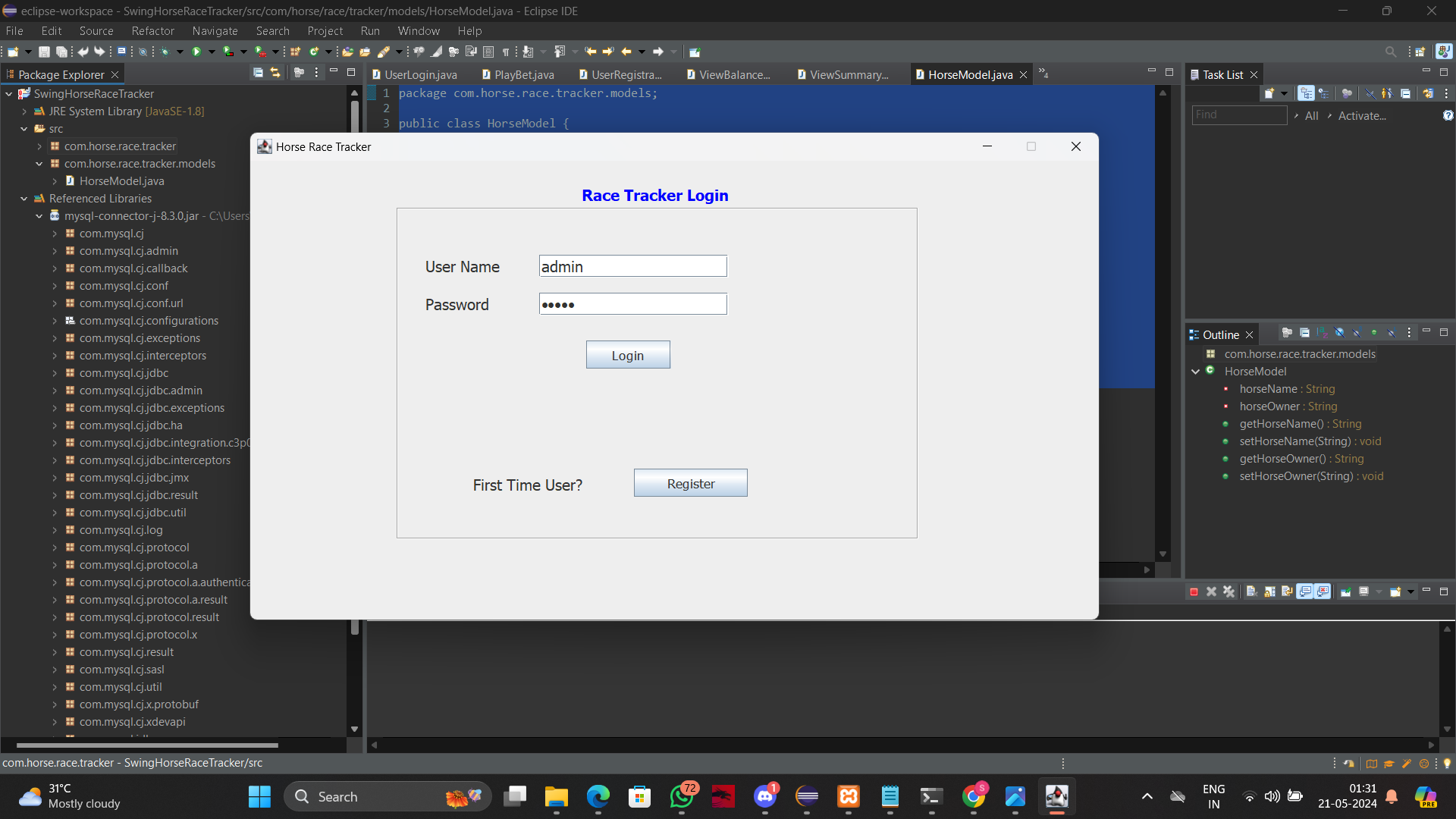
* The chosen architecture is suitable for this application's scale. As the application grows, considerations might be needed to optimize performance for handling a larger user base or complex betting functionalities.
* The ER diagram can be extended to include additional entities based onspecific features. For instance, a "Jockey" entity could be added to capture information about riders participating in races.
* The level of normalization applied (1NF) might be sufficient for a basic betting application. If the data complexity increases significantly, further normalization (2NF, 3NF) could be explored.
* The requirement specifications provide a good starting point for development. Further user research and analysis might be needed to refine specific functionalities and ensure the application caters to the target audience's needs effectively.

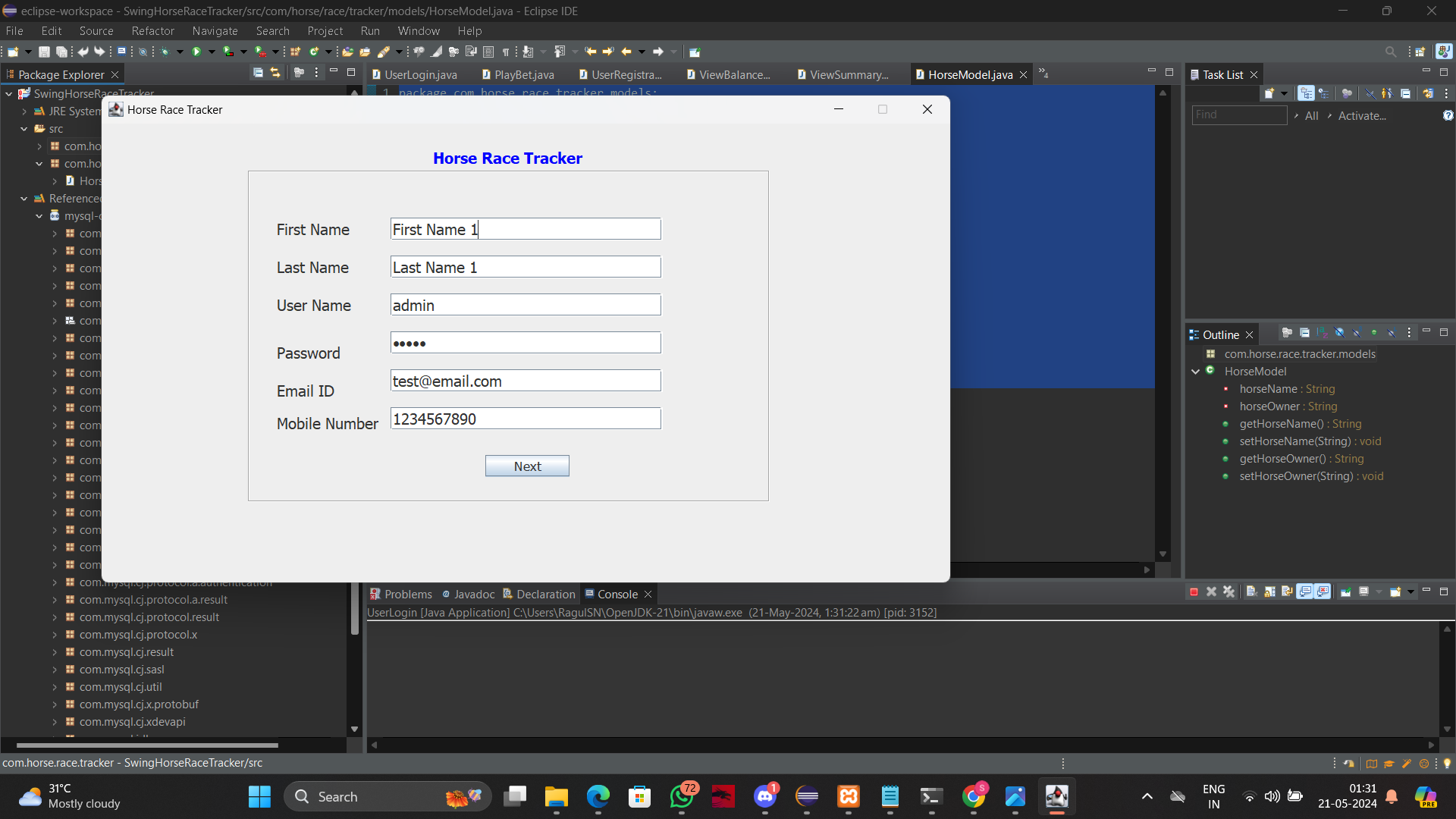
**Additional Considerations**

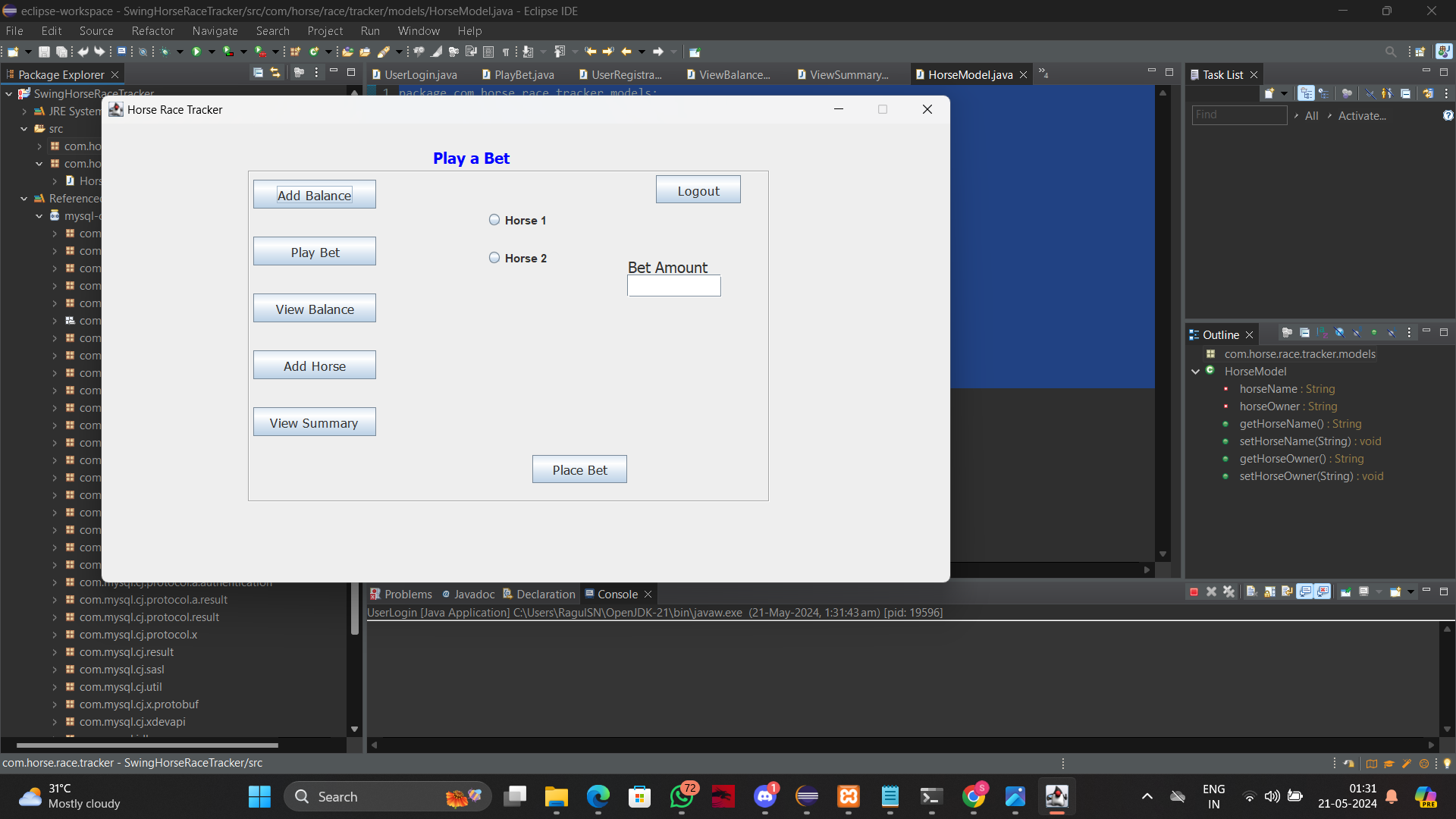
* Security is paramount when dealing with financial transactions. Implementing robust security measures like secure password hashing, user authentication, and authorization is crucial.
* Integrating with an external service for real-time horse racing odds can enhance the user experience and provide more dynamic betting options.
* Regulations surrounding online betting should be considered during development to ensure compliance.

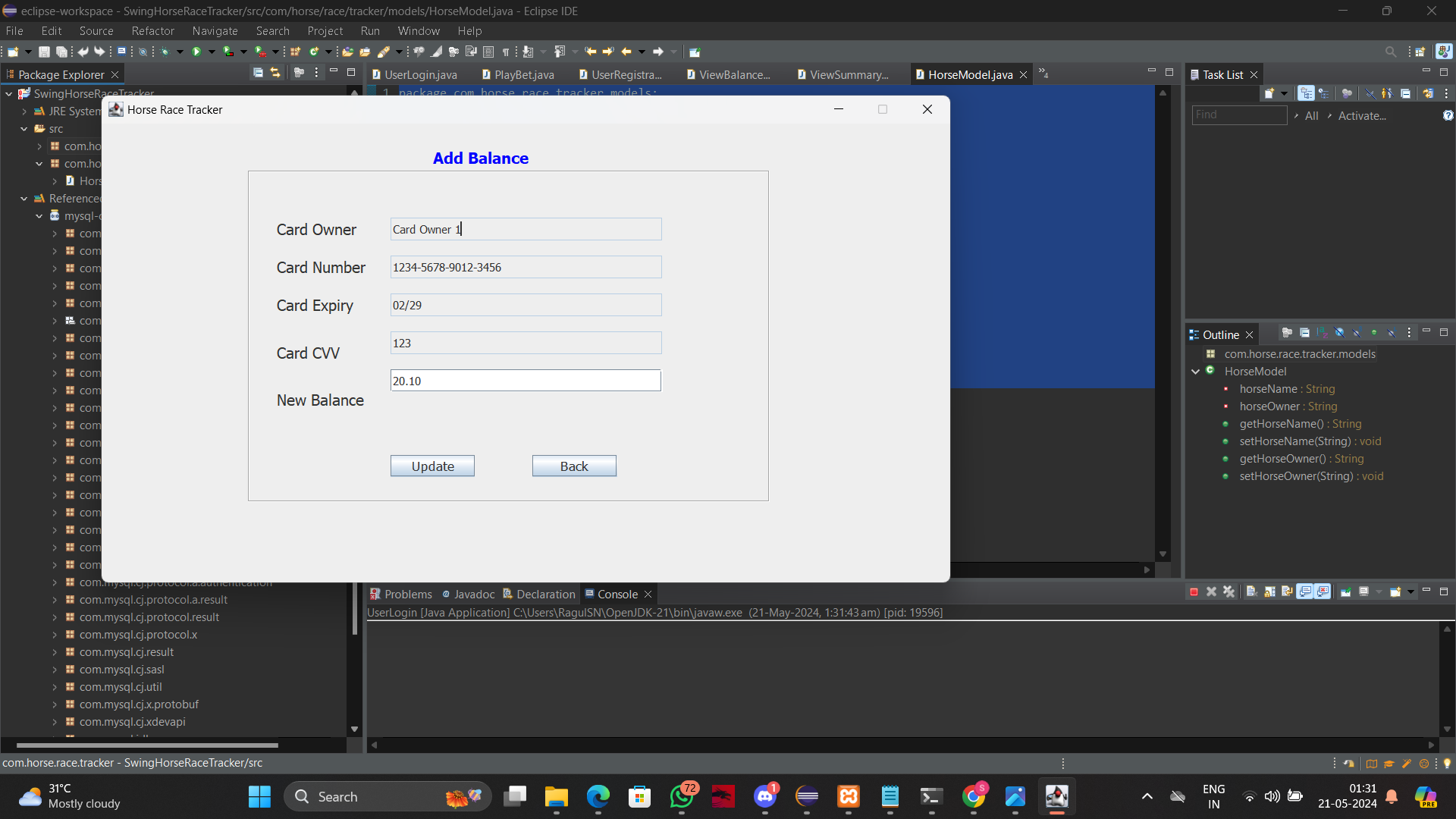
By considering these results and discussions, you can move forward with developing a well-structured, secure, and user-friendly horse race betting application.

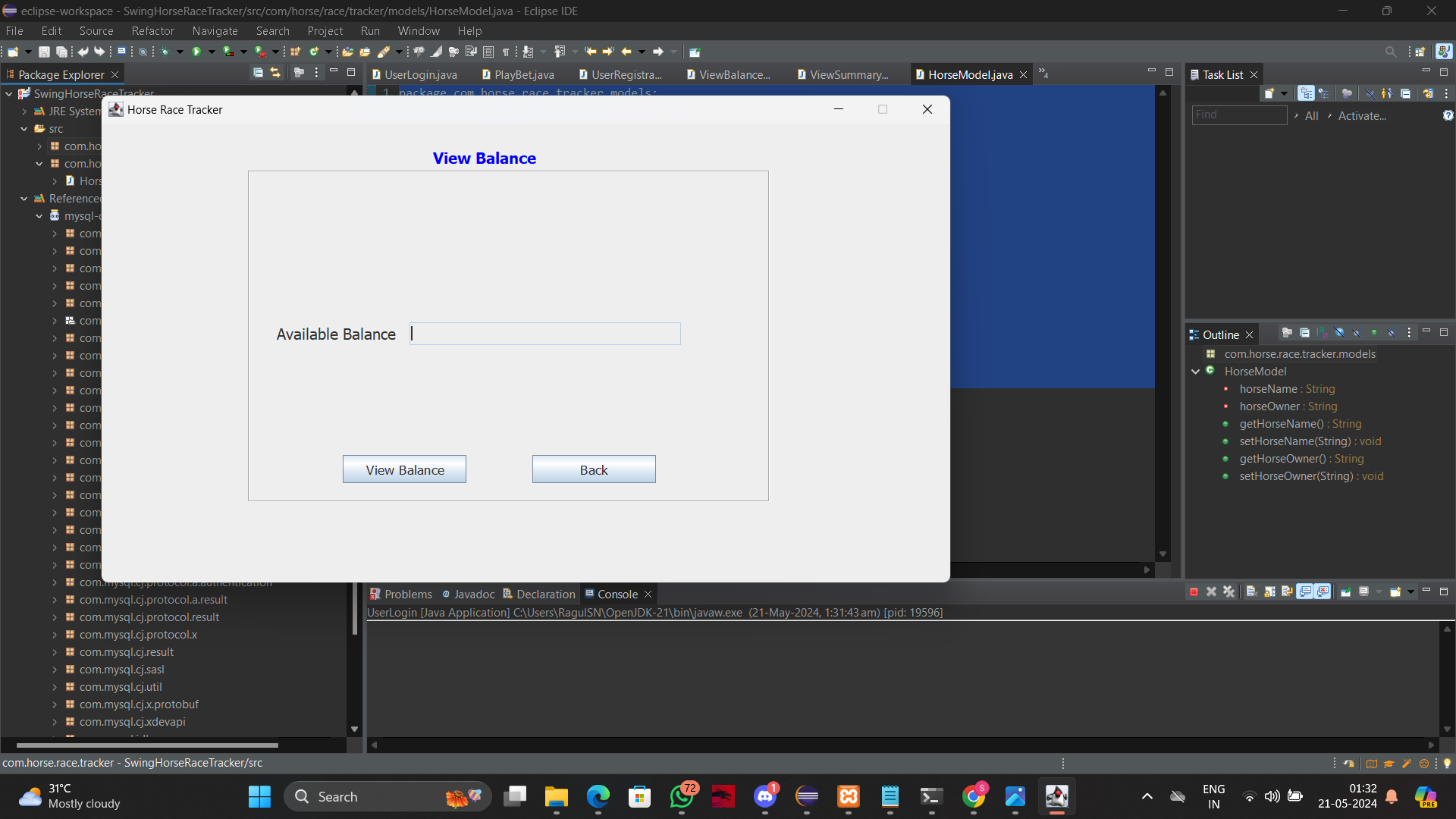
**OUTPUTS**

****

****

****

****

****

**6. CONCLUSION**

Using Java Swings for the front-end and MySQL for the back-end, with JDBC for database connectivity, provides a solid foundation for building a horse race betting application. Employing the MVC architecture ensures clean separation of concerns, making the application maintainable and scalable. Additional libraries and tools enhance functionality, logging, and testing capabilities, leading to a robust and user-friendly application.

**7. REFERENCES**

**W3Schools MySQL Tutorial:**

W3Schools provides easy-to-follow tutorials on SQL and MySQL with examples that you can practice.

**GitHub Guides:**

GitHub provides guides on version control, collaboration, and best practices for using Git and GitHub.

**GeeksforGeeks :**

GeeksforGeeks provides practical tutorials and examples on using Matplotlib for data visualization.