TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]

THEME PARK MANAGEMENT SYSTEM

ABSTRACT IDEA

PROBLEM STATEMENT : **Theme Park Management System**

The **Theme Park Management System** is a robust and efficient software solution designed to streamline the operations of a theme park.

OBJECTIVE :

This system provides a centralized platform for managing

- Ticketing
- Ride information
- Staff Details

TECH STACK :

FRONTEND:

1. **Java Swing**:
    - For building the Graphical User Interface (GUI).
    - Provides a lightweight, platform-independent library for creating user-friendly desktop applications.

BACKEND:

1. **Java**:
    - Core language for implementing the business logic and handling backend operations.
    - Responsible for connecting the GUI to the database using JDBC.
2. **MySQL**:
    - Relational database for managing and storing data related to tickets, rides, staff, and operational metrics.
    - Ensures structured and efficient data storage with support for SQL queries.
3. **JDBC (Java Database Connectivity)**:
    - Enables seamless communication between the Java application and the MySQL database.

.

TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]

KEY FEATURES OF THE SYSTEM INCLUDES :

- **Ticket Management:** Enables customers to purchase and manage tickets, recording details such as customer names, ticket types, and prices.
- **Ride Management:** Maintains an inventory of rides, their status (operational or under maintenance), and capacity, ensuring real-time updates and better resource allocation.
- **Staff Management:** Tracks staff roles, salaries, and shifts, simplifying workforce administration.
- **Operational Insights:** Provides actionable data such as total revenue from ticket sales and visitor counts for each ride, aiding in strategic decision-making.

BENEFITS AND ADVANTAGES :

The project emphasizes modularity, scalability, and data security, making it adaptable to varying sizes of theme parks.

By automating routine processes and centralizing data management, the **Theme Park Management System** minimizes manual errors, reduces operational overhead, and improves the overall customer experience.

This project is an ideal solution for enhancing the operational efficiency of theme parks while supporting scalability and adaptability to meet future demands.

CONCLUSION :

The **Theme Park Management System** streamlines ticketing, ride operations, and staff management while providing actionable insights to enhance efficiency. Built using **Java Swing** and **MySQL**, the system ensures seamless functionality, reducing manual errors and improving decision-making with real-time analytics.

With its user-friendly interface and modular design, the system automates routine tasks, enhances operational efficiency, and delivers a superior customer experience. Scalable and adaptable, it lays a strong foundation for future enhancements like online ticketing and mobile integration, making it a reliable solution for modern theme park management.

TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]

DATABASE DESIGN

The database schema will include tables for:

1. **Staff**
2. **Rides**
3. **Tickets**
4. **Customers**
5. **Operations Insights**

STEP BY STEP IMPLEMENTATION :

1. Set up the MySQL Database

Create a MySQL database named theme_park_management and create necessary tables:

```
CREATE TABLE Staff (

    staff_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    role VARCHAR(50),
    contact VARCHAR(20),
    salary DECIMAL(10, 2)
);
CREATE TABLE Rides (
    ride_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    capacity INT,
    status VARCHAR(20) -- e.g., "Active", "Under Maintenance"
);
CREATE TABLE Tickets (
    ticket_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100),
    ticket_type VARCHAR(50), -- e.g., "Regular", "VIP"
    price DECIMAL(10, 2),
    date_of_purchase DATE
);

CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20)
);
```

TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]

2. Set up Java Project

1. Create a new Java project in your IDE.
2. Add the MySQL JDBC driver to the project dependencies (downloadable from the MySQL website).
3. Set up a connection to the MySQL database using JDBC.

3. Backend - Java Code for Database Connection

First, create a class DatabaseConnection.java to establish the connection with MySQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
   private static final String URL =
"jdbc:mysql://localhost:3306/theme_park_management";
   private static final String USER = "root";  // Change to your MySQL
username
   private static final String PASSWORD = "";  // Change to your MySQL
password

   public static Connection getConnection() throws SQLException {
      return DriverManager.getConnection(URL, USER, PASSWORD);
   }
}
```

TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]

4.MANAGEMENT SYSTEM

1. Staff Management

Create a StaffManager class to handle staff-related operations.

```
import java.sql.*;
public class StaffManager {
   public void addStaff(String name, String role, String contact, double salary) {
      String query = "INSERT INTO Staff (name, role, contact, salary) VALUES (?, ?, ?,
?)";
      try (Connection conn = DatabaseConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(query)) {
         stmt.setString(1, name);
         stmt.setString(2, role);
         stmt.setString(3, contact);
         stmt.setDouble(4, salary);
         stmt.executeUpdate();
         System.out.println("Staff added successfully.");
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
   public void viewStaff() {
      String query = "SELECT * FROM Staff";

      try (Connection conn = DatabaseConnection.getConnection();
           Statement stmt = conn.createStatement();
           ResultSet rs = stmt.executeQuery(query)) {
         while (rs.next()) {
            System.out.println("ID: " + rs.getInt("staff_id") + ", Name: " +
rs.getString("name"));
         }
      } catch (SQLException e) {
         e.printStackTrace();
}
```

2. Ride Management

Create a RideManager class for managing rides.

```
import java.sql.*;
      }
   }
public class RideManager {
   public void addRide(String name, int capacity, String status) {
      String query = "INSERT INTO Rides (name, capacity, status) VALUES (?, ?, ?)";

      try (Connection conn = DatabaseConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(query)) {
         stmt.setString(1, name);
         stmt.setInt(2, capacity);
         stmt.setString(3, status);
         stmt.executeUpdate();
         System.out.println("Ride added successfully.");
      } catch (SQLException e) {
         e.printStackTrace();
```

```
        }
    }
    public void viewRides() {
        String query = "SELECT * FROM Rides";
        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                System.out.println("Ride ID: " + rs.getInt("ride_id") + ", Name: " +
rs.getString("name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

3. Ticket Management

Create a TicketManager class to manage ticket purchases.

```
import java.sql.*;
import java.util.Scanner;

public class TicketManager {
    public void buyTicket(String customerName, String ticketType, double price) {
        String query = "INSERT INTO Tickets (customer_name, ticket_type, price,
date_of_purchase) VALUES (?, ?, ?, NOW())";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, customerName);
            stmt.setString(2, ticketType);
            stmt.setDouble(3, price);
            stmt.executeUpdate();
            System.out.println("Ticket purchased successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

4.CREATE A USER INTERFACE

```
import java.util.Scanner;
public class ThemeParkApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StaffManager staffManager = new StaffManager();
        RideManager rideManager = new RideManager();
        TicketManager ticketManager = new TicketManager();
        while (true) {
            System.out.println("Theme Park Management System");
            System.out.println("1. Add Staff");
            System.out.println("2. View Staff");
            System.out.println("3. Add Ride");
```

```
            System.out.println("4. View Rides");
            System.out.println("5. Buy Ticket");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter staff name: ");
                    String name = scanner.next();
                    System.out.print("Enter staff role: ");
                    String role = scanner.next();
                    System.out.print("Enter staff contact: ");
                    String contact = scanner.next();
                    System.out.print("Enter staff salary: ");
                    double salary = scanner.nextDouble();
                    staffManager.addStaff(name, role, contact, salary);
                    break;
                case 2:
                    staffManager.viewStaff();
                    break;
                case 3:
                    System.out.print("Enter ride name: ");
                    String rideName = scanner.next();
                    System.out.print("Enter ride capacity: ");
                    int capacity = scanner.nextInt();
                    System.out.print("Enter ride status: ");
                    String status = scanner.next();
                    rideManager.addRide(rideName, capacity, status);
                    break;
                case 4:
                    rideManager.viewRides();
                    break;
                case 5:
                    System.out.print("Enter customer name: ");
                    String customerName = scanner.next();
                    System.out.print("Enter ticket type: ");
                    String ticketType = scanner.next();
                    System.out.print("Enter ticket price: ");
                    double price = scanner.nextDouble();
                    ticketManager.buyTicket(customerName, ticketType, price);
                    break;
                case 6:
                    System.out.println("Exiting...");
                    scanner.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}
```

TEAM MEMBERS : DHARSHNA U 231901008 CSE [CYBER SECURITY]
VAISHNAVI E 231901059 CSE [CYBER SECURITY]