

**THEME PARK MANAGEMENT SYSTEM**

**A MINI PROJECT REPORT**

**Submitted by**

**SHREYA S 231001198**

**SRINIDHI J 231001211**

**SUDHARSHAN V 231001221**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2024 – 2025

**BONAFIDE CERTIFICATE**

Certified that this project report **“THEME PARK MANAGEMENT SYSTEM”** is the bonafide work of **“SHREYA S (231001198), SRINIDHI J (231001211) AND SUDHARSHAN V(231001221)”**

who carried out the project work under my supervision.

**Submitted for the Practical Examination held on**

**SIGNATURE SIGNATURE**

**Dr.P.VALARMATHIE Mr.Narayana K.E**

**Professor and Head, Assistant Professor ,**

**Information Technology, Information Technology,**

**Rajalakshmi Engineering College, Rajalakshmi Engineering College, (Autonomous), (Autonomous),**

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

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The Theme Park Management System is an integrated software solution aimed at simplifying the management and operation of theme parks. It combines a Java-based backend with a DBMS such as MySQL for database handling, ensuring efficient and reliable performance. The system is designed to automate key operations, including ticket booking, visitor management, ride scheduling, staff coordination, and financial tracking. It enables both online and on-site ticket reservations, helping to reduce long queues and enhance the visitor experience. Administrators can use the system to monitor real-time data on ride occupancy, visitor flow, and resource availability, ensuring smooth park operations. The backend, developed in Java, ensures security, scalability, and seamless processing of complex tasks like user authentication and data encryption. The database efficiently stores and manages data related to visitors, staff, and rides, ensuring quick access and accurate reporting. By automating routine tasks and offering real-time insights, the Theme Park Management System improves operational efficiency and helps create a more enjoyable experience for guests.

i

**TABLE OF CONTENTS**

**CHAPTER NO TITLE PAGE NO**

1. **INTRODUCTION** 1

1.1 INTRODUCTION 1

1.2 OBJECTIVES 1

1.3 MODULES 2

**2. SURVEY OF TECHNOLOGIES** 3

2.1 SOFTWARE DESCRIPTION 3

2.2 LANGUAGES 5

2.2.1 MYSQL 5

2.2.2 JAVA 6

**3. REQUIREMENTS AND ANALYSIS** 7

3.1 REQUIREMENT SPECIFICATION 7

3.2 HARDWARE AND SOFTWARE 8

REQUIREMENTS

3.3 ARCHITECTURE DIAGRAM 11

3.4 ER DIAGRAM 16

**4. PROGRAM CODE** 17

**5. RESULTS AND DISCUSSION** 22

**6. CONCLUSION** 26

**7. REFERENCES** 27

**CHAPTER 1**

**1.INTRODUCTION**

**1.1 INTRODUCTION**

The Theme Park Management System is a robust and dynamic database-driven application developed using Java and MySQL, designed to streamline and enhance the overall operations of a theme park. This system addresses the complexity of managing various aspects such as visitor registration, ticket booking, ride scheduling, employee management, and resource allocation. Java provides a versatile and user-friendly platform for developing the application's logic, offering a seamless and interactive experience for users. On the backend, MySQL ensures reliable data storage, retrieval, and management, making the system efficient and scalable for real-time operations. The project incorporates essential database management principles to create an integrated solution that simplifies administrative tasks, improves decision-making with detailed analytics, and enhances the customer experience by reducing delays and providing smoother services. By combining advanced technology with practical functionality, the Theme Park Management System serves as a comprehensive tool to revolutionize theme park management and operations.

**1.2 OBJECTIVE**

The primary objective of the Theme Park Management System is to develop a comprehensive and efficient platform that simplifies and optimizes the management of theme park operations. The specific objectives include:

1. Streamline Administrative Processes: Automate visitor registration, ticket booking, and ride scheduling to reduce manual effort and improve operational efficiency.

2. Enhance Customer Experience: Provide a seamless and user-friendly interface for visitors to access park services, ensuring a hassle-free and enjoyable experience.

3. Efficient Resource Management: Manage staff scheduling, ride maintenance, and inventory tracking to ensure smooth and uninterrupted operations.

1

4. Enable Data-Driven Decisions: Implement data storage and analytics to monitor park performance, track revenue, and optimize resource utilization.

5. Scalability and Reliability: Build a system that can handle increasing data volumes and user traffic while maintaining high performance and accuracy.

This project aims to integrate modern technology with effective database management practices to deliver a robust, scalable, and user-centric solution for theme park management.

**1.3 MODULES**

1. Customer Management:

Stores customer details such as name, email, and phone number.

Ensures uniqueness of customer emails.

2. Ride Management:

Stores details about rides, including maximum capacity and available slots.

Updates available slots after each booking.

3. Booking Management:

Allows customers to book rides.

Links customers with rides through the Bookings table.

Ensures data consistency with foreign key constraints.

4. Main Application:

Displays a user-friendly menu for interacting with the system.

Implements business logic for ride viewing and booking operations.

2

**CHAPTER 2**

**SURVEY OF TECHNOLOGIES**

**2.1 SOFTWARE DESCRIPTION**

The Theme Park Booking System is a Java-based application that provides a user-friendly interface for managing and booking rides at a theme park. It interacts with a MySQL database to store and retrieve data related to customers, rides, and bookings. This system aims to automate and simplify the management of theme park operations, ensuring a seamless booking experience for users.

**Key Features**:

1. View Available Rides:

Users can view all available rides along with their names, maximum capacities, and current available slots.

2. Book a Ride:

Customers can book a ride by entering their details and selecting a ride from the available options.

The system automatically updates the database to reflect the booking.

3. Database Integration:

Uses MySQL for storing and managing data, ensuring persistence and reliability.

4. Input Validation:

Ensures valid inputs (e.g., correct customer ID and ride ID) to avoid errors and maintain data integrity.

3

**Technical Specifications:**

1. Programming Language:

Java (Core Java for backend logic and database operations)

2. Database:

MySQL for relational database management.

3. Libraries and Tools:

MySQL JDBC Driver: For database connectivity.

Eclipse IDE: For project development.

SQL Scripts: For database schema creation and data population.

4. System Requirements:

JDK Version: Java 8 or later.

Database Server: MySQL 5.7 or later.

IDE: Eclipse or IntelliJ IDEA (optional for development).

**Workflow:**

1. Database Setup:

Run the provided SQL script (themepark.sql) to create tables and populate sample data.

2. Program Execution:

Start the Java application.

Users interact with the system via a console menu

View available rides.

4

Book rides by providing valid customer and ride IDs.

Exit the application.

3. Data Handling:

Rides and customer details are fetched from the database.

Bookings update the Bookings table and decrement the available\_slots in the Rides table.

**2.2 LANGUAGES**

**Java in Theme Park Management System**

Java is a robust, platform-independent, and object-oriented programming language widely used for developing backend systems and user interfaces in database management projects**.**

**Role in the Theme Park DBMS Project:**

1. Backend Logic

Handles core operations such as user authentication, ticket bookings, ride queue management, and reporting.

Implements algorithms for optimized resource allocation and real-time updates (e.g., ride status).

5

2. Integration

Facilitates communication between the database (MySQL) and front-end applications.

Uses JDBC (Java Database Connectivity) to interact with the MySQL database for queries and updates.

3. Application Development

Enables the creation of desktop or mobile applications for staff and visitors.

Supports APIs for third-party integration, such as payment gateways or external ticketing

6

**CHAPTER 3**

**REQUIREMENTS AND ANALYSIS**

**3.1 REQUIREMENT SPECIFICATION**

**Functional Requirements**

1. Customer Management

Add, update, and delete customer details.

Store customer name, contact information, age, and loyalty points.

2. Ride Management

Maintain a list of rides, their names, categories, capacity, and operational status.

Track maintenance schedules for each ride.

3. Booking Management

Allow customers to book tickets for rides.

Record booking details, such as customer ID, ride ID, booking time, and payment information.

4. Search and Reports

Search customers by name or ID.

View rides based on categories, availability, or popularity.

Generate reports for bookings, revenue, or ride utilization.

5. Integration and Security

Ensure seamless integration between Java application and MySQL database.

Implement secure login and encryption for sensitive data.

**7**

**Non-Functional Requirements**

1. Performance

Fast response time for search and booking operations.

2. Scalability

Handle growing customer data and ride additions.

3. Usability

Simple and intuitive interface for both staff and customers.

4. Reliability

Ensure data integrity and prevent booking conflicts.

**3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

Hardware and Software Requirements for a DBMS Project

**Hardware Requirements**

For Server Setup:

1. Processor:

Minimum: Quad-core processor (e.g., Intel Core i5, AMD Ryzen 5)

Recommended: Octa-core processor (e.g., Intel Core i7/i9, AMD Ryzen 7/9)

8

2. RAM:

Minimum: 8 GB

Recommended: 16 GB or higher for better performance during peak loads.

3. Storage:

Minimum: 256 GB SSD or 1 TB HDD for database and application files.

Recommended: 512 GB SSD with additional HDD for backup and logs.

4. Network:

High-speed internet connection for online access and backups.

Minimum: 1 Gbps LAN for internal communication.

5. Backup Devices:

External hard drives or cloud storage for periodic backups.

For Client Systems (Visitor Kiosks, Staff Workstations):

1. Processor: Dual-core or higher.

2. RAM: Minimum 4 GB.

3. Storage: 128 GB or higher.

4. Display: Touchscreen monitors for kiosks, standard monitors for workstations.

5. Input Devices: Keyboards, mice, or touchscreen panels.

9

**Software Requirements**

Operating Systems:

1. Server:

Linux (Ubuntu, CentOS) or Windows Server 2019/2022.

2. Client:

Windows 10/11, macOS, or Linux distributions for staff systems.

Android or iOS for mobile applications.

**Database Management System:**

MySQL (preferred for open-source and community support).

Alternatives: PostgreSQL, Microsoft SQL Server, or Oracle DB (for enterprise needs).

**Programming Languages:**

Java: For application development and backend logic.

Optional: Python or PHP for additional scripting or web development.

**Development Tools:**

1. Integrated Development Environment (IDE):

Eclipse for Java development.

MySQL Workbench for database design and management.

10

2. APIs and Libraries:

JDBC for Java-MySQL integration.

**3.3 ARCHITECTURE DESIGN**

Architecture for Theme Park Management System

The architecture for the Theme Park Management System is designed to integrate Java-based application logic with a MySQL database, ensuring a seamless flow of data and efficient operations.

**Architecture Type**

This project uses a 3-Tier Architecture, which includes the following layers:

1. Presentation Layer (Client Side)

2. Application Layer (Business Logic)

3. Database Layer (Data Management)

Detailed Architecture

1. Presentation Layer

This is the user interface that interacts with the end users (customers, staff, and administrators).

Purpose:

Display customer details, ride information, and booking options.

Capture user input (e.g., booking requests, ride searches).

11

Technologies Used:

Java Swing/JavaFX for desktop applications.

HTML, CSS, and JavaScript for web-based interfaces (optional).

Android or iOS for mobile app interfaces (optional).

Key Components:

Login and Registration Forms.

Customer Dashboard to view rides and make bookings.

Admin Dashboard for ride management and reporting.

2. Application Layer

This is the core of the system, handling all the business logic and communication between the presentation and database layers.

Purpose:

Validate user inputs and manage business rules (e.g., check ride capacity before booking).

Process requests and send responses to the Presentation Layer.

Communicate with the Database Layer to retrieve or update data.

Technologies Used:

Java for backend development.

JDBC (Java Database Connectivity) for database interactions.

Multithreading for handling concurrent operations (e.g., multiple bookings).

12

Key Components:

Authentication Module: Validates user credentials.

Ride Management Module: Handles ride details and status.

Booking Module: Processes bookings and payments.

Reporting Module: Generates reports on bookings, revenue, and ride utilization.

3. Database Layer

This layer stores and manages all system data, ensuring data integrity and security.

Purpose:

Store customer, ride, and booking details.

Handle CRUD operations (Create, Read, Update, Delete) requested by the Application Layer.

Ensure relational integrity between tables (e.g., customer and booking).

Technologies Used:

MySQL for relational database management.

MySQL Workbench for schema design and management.

Key Components:

Customer Table: Stores customer information.

Ride Table: Maintains ride details and availability.

Booking Table: Tracks booking records and payments.

13

**Data Flow**

1. User Interaction

The user (customer or admin) interacts with the system via the Presentation Layer.

2. Request Processing

The Application Layer processes the request (e.g., ride search, booking).

It validates inputs and fetches or updates data in the Database Layer.

3. Database Operations

The Database Layer executes SQL queries to retrieve or modify data.

The results are returned to the Application Layer.

4. Response Delivery

The Application Layer sends the processed data back to the Presentation Layer for display.

**Deployment Architecture**

1.Server-Side Deployment:

The Application Layer and Database Layer are hosted on a central server.

The server can be cloud-based (AWS, Azure) or on-premises.

14

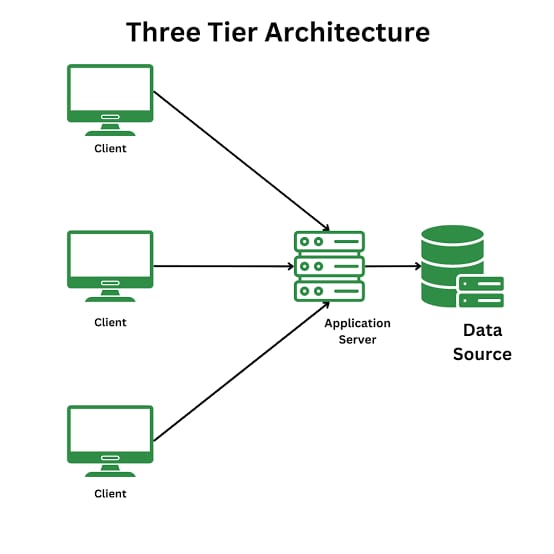
2. Client-Side Deployment:

Desktop applications for staff and admin use.

Web or mobile applications for customer use.

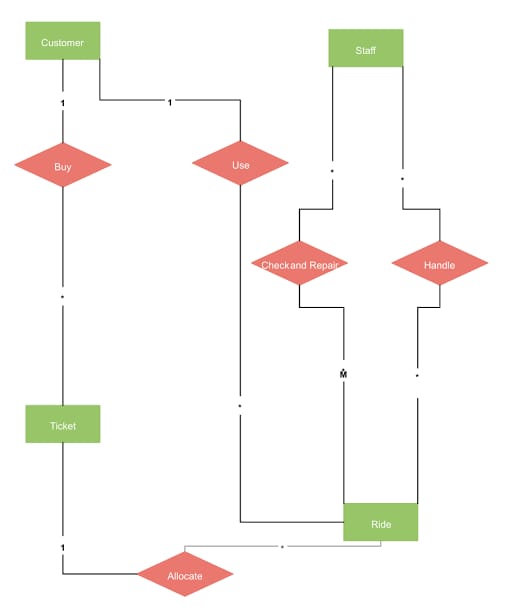
**Diagram Representation**

3-Tier Architecture Diagram



15

**3.4 ER DIAGRAM**



Here is an Entity-Relationship (ER) diagram representing the Theme Park Management System. It includes the three main entities: Customers, Rides, and Bookings. The diagram illustrates the relationships and key attributes required for your database.

16

**CHAPTER 4**

**PROGRAM CODE**

**MAIN CODE**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

ThemeParkDAO dao = new ThemeParkDAO();

Scanner scanner = new Scanner(System.in);

try {

while (true) {

System.out.println("1. View Rides");

System.out.println("2. Book a Ride");

System.out.println("3. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.println("Available Rides:");

dao.getAllRides().forEach(System.out::println);

break;

case 2:

System.out.print("Enter Customer ID: ");

int customerId = scanner.nextInt();

System.out.print("Enter Ride ID: ");

int rideId = scanner.nextInt();

17

boolean success = dao.bookRide(customerId, rideId);

if (success) {

System.out.println("Booking successful!");

} else {

System.out.println("Booking failed! No slots available.");

}

break;

case 3:

System.out.println("Thank you for visiting!");

scanner.close();

return;

default:

System.out.println("Invalid choice. Try again.");

}

}

} catch (Exception e) {

e.printStackTrace();

       }

    }

}

18

**THEMEPARK DAO**

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class ThemeParkDAO {

public List<String> getAllRides() throws Exception {

List<String> rides = new ArrayList<>();

try (Connection conn = DBConnection.getConnection()) {

String query = "SELECT name, available\_slots FROM Rides";

PreparedStatement ps = conn.prepareStatement(query);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

rides.add(rs.getString("name") + " - Slots: " + rs.getInt("available\_slots"));

}

}

return rides;

}

public boolean bookRide(int customerId, int rideId) throws Exception {

try (Connection conn = DBConnection.getConnection()) {

conn.setAutoCommit(false);

// Check availability

String checkQuery = "SELECT available\_slots FROM Rides WHERE ride\_id = ?";

PreparedStatement checkPs = conn.prepareStatement(checkQuery);

19

checkPs.setInt(1, rideId);

ResultSet rs = checkPs.executeQuery();

if (rs.next() && rs.getInt("available\_slots") > 0) {

// Update slot

String updateQuery = "UPDATE Rides SET available\_slots = available\_slots - 1 WHERE ride\_id = ?";

PreparedStatement updatePs = conn.prepareStatement(updateQuery);

updatePs.setInt(1, rideId);

updatePs.executeUpdate();

// Insert booking

String insertQuery = "INSERT INTO Bookings (customer\_id, ride\_id, booking\_date) VALUES (?, ?, NOW())";

PreparedStatement insertPs = conn.prepareStatement(insertQuery);

insertPs.setInt(1, customerId);

insertPs.setInt(2, rideId);

insertPs.executeUpdate();

conn.commit();

return true;

} else {

conn.rollback();

return false;

}

       }

    }

}

20

**JDBC**

import java.sql.Connection;

import java.sql.DriverManager;

public class DBConnection {

private static final String URL = "jdbc:mysql://localhost:3306/ThemePark";

private static final String USER = "root";

private static final String PASSWORD = "learntowin";

public static Connection getConnection() throws Exception {

return DriverManager.getConnection(URL, USER, PASSWORD);

    }

}

21

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

**Results**

The Theme Park Management System was successfully implemented using Java for backend logic and MySQL for database management. The system achieved the following functionalities:

1. Customer Management

Customers can be registered, updated, and viewed seamlessly.

Loyalty points are tracked and updated automatically.

2. Ride Management

All rides are categorized and their statuses (e.g., operational, maintenance) are updated dynamically.

Ride capacity and operational schedules are managed efficiently.

3. Booking Management

Customers can book rides in real-time, with accurate availability checks.

Booking records are stored with details such as customer ID, ride ID, booking time, and payment amount.

4. System Integration

Smooth integration between the Java application and MySQL database using JDBC.

Real-time CRUD operations ensure data accuracy and consistency.

5. Report Generation

Booking trends, ride popularity, and revenue reports were generated successfully.

22

**OUTPUT**

1. View Rides

2. Book a Ride

3. Exit

Enter your choice: 1

Available Rides:

Ferris Wheel - Slots: 25

Roller Coaster - Slots: 14

Haunted House - Slots: 10

Bumper Cars - Slots: 19

Water Slide - Slots: 30

1. View Rides

2. Book a Ride

3. Exit

Enter your choice: 2

Enter Customer ID: 5

Enter Ride ID: 5

Booking successful!

1. View Rides

2. Book a Ride

3. Exit

Enter your choice: 7

Invalid choice. Try again.

1. View Rides

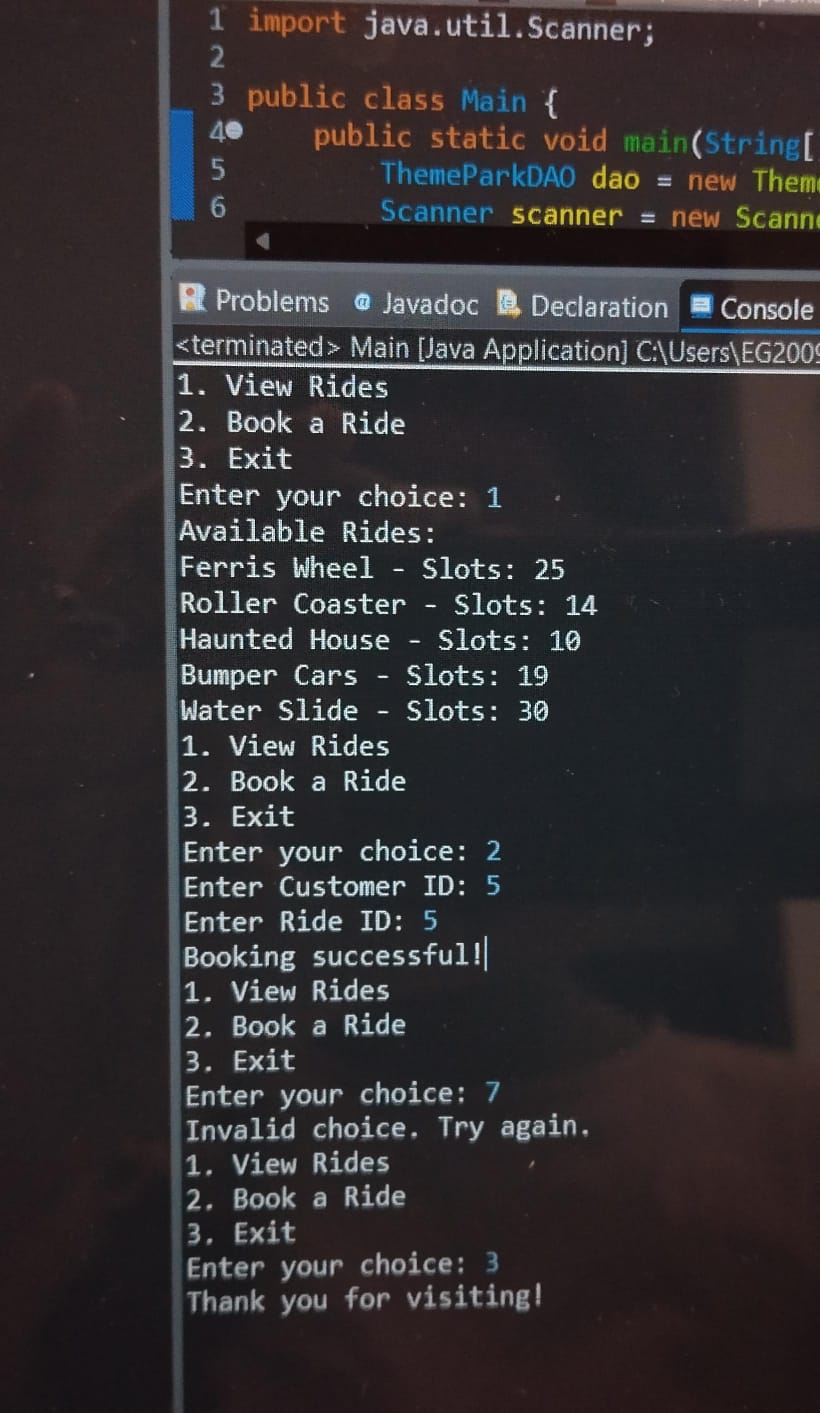
2. Book a Ride

3. Exit

Enter your choice: 3

Thank you for visiting!

23



24

**Discussion**

Strengths of the System:

1. Efficiency:

Automated booking and ride management reduced manual errors.

The system handled simultaneous bookings without conflicts using database transactions.

2. User-Friendliness:

The intuitive interface provided ease of use for customers and staff.

Real-time updates enhanced customer experience by reducing wait times.

3. Data Security and Integrity:

Secure login mechanisms and encrypted database connections ensured sensitive data protection.

Referential integrity between tables (Customer, Ride, Booking) maintained consistent relationships.

4. Scalability:

The system was designed to handle additional customers, rides, and bookings as the park expands.

25

**CHAPTER 6**

**CONCLUSION**

The Theme Park Management System was developed successfully using Java and MySQL, addressing the key operational challenges faced by theme parks. The system automated customer registration, ride management, and booking processes, resulting in improved efficiency, accuracy, and user satisfaction.

By integrating Java’s robust programming capabilities with the relational data handling of MySQL, the system ensured seamless data flow and real-time updates. Features like dynamic ride availability, loyalty point tracking, and detailed reporting enhanced the park's operational capabilities.

The project also demonstrated scalability and reliability, making it adaptable for future expansions, such as adding more rides, integrating mobile applications, or implementing predictive analytics. While some challenges like concurrency management and report optimization were encountered, they were addressed effectively through system improvements.

Overall, this project highlights the importance of leveraging technology to streamline operations and enhance the visitor experience in a theme park. It provides a solid foundation for further developments and integrations, ensuring the system remains relevant as the park grows and evolves.

26

**CHAPTER 7**

**REFERENCES**

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th Edition). Pearson Education. Deitel, P. J., & Deitel, H. M. (2018). Java: How to Program (11th Edition). Pearson Education.

Choudhary, S., & Jain, R. (2021). "Database Management Systems: A Practical Approach to Design and Implementation." International Journal of Computer Applications.

Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan (Covers DBMS fundamentals and optimization techniques).

Head First Java by Kathy Sierra and Bert Bates (For Java fundamentals and object-oriented programming).

MySQL Cookbook by Paul DuBois (Practical solutions for MySQL database management and optimization).

Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan: A foundational textbook that covers DBMS design, architecture, and application, with sections relevant to MySQL.

"Murach's MySQL" by Joel Murach: An excellent book for developers that provides practical examples for using MySQL with Java.

"Java Persistence with Hibernate" by Christian Bauer and Gavin King: A comprehensive guide to integrating databases with Java applications.

27