Community Sports Ground Slot Booking & Usage Tracking System JDBC

Submitted By

717823P352

717823P247

Environmental Waste Collection Route Planning & Pickup Assignment System JDBC

Overview

Create a console-based Java application that allows a local community center or resident welfare association to manage bookings of a common sports ground (for example, for football, cricket, volleyball, or casual play) and track basic usage details after each session.

- The console application should support 5–6 core operations:
- 1. Register New Slot Booking
- 2. View Booking Details / List Bookings by Date & Status
- 3. Confirm / Cancel Booking (Transactional)
- 4. Record Post-Usage Details for a Booking (Transactional)
- 5. Adjust / Correct a Usage Record (Transactional)
- 6. View Usage Reports by Date / Sport / Team

Short description of operations:

• Register New Slot Booking – Captures booking party details, date, time window, purpose (sport), and user type; calculates a provisional booking fee using simple rules and stores the booking with status PENDING.

• View Booking Details / List – Retrieves a booking by ID or lists bookings filtered by date and status (PENDING, CONFIRMED, CANCELLED, COMPLETED) so that coordinators can review daily schedules.

• Confirm / Cancel Booking – Confirms a booking after checking that no existing CONFIRMED booking overlaps the same date and time window; or cancels the booking with a reason. Confirmation and cancellation are updated transactionally.Remove Zone / Vehicle (only when safe)

• Record Post-Usage Details – After the slot is used, the ground caretaker records sport played, number of players, condition of ground, any minor damage, and whether the group overran the time; this inserts a row in SLOT_USAGE_TBL and sets the booking to COMPLETED in one transaction.

• Adjust / Correct a Usage Record – Allows correction of basic data like number of players or comments, with validation. • View Usage Reports – Retrieves usage records by date, sport, or team name to understand ground utilization and problem areas

A. Database Design

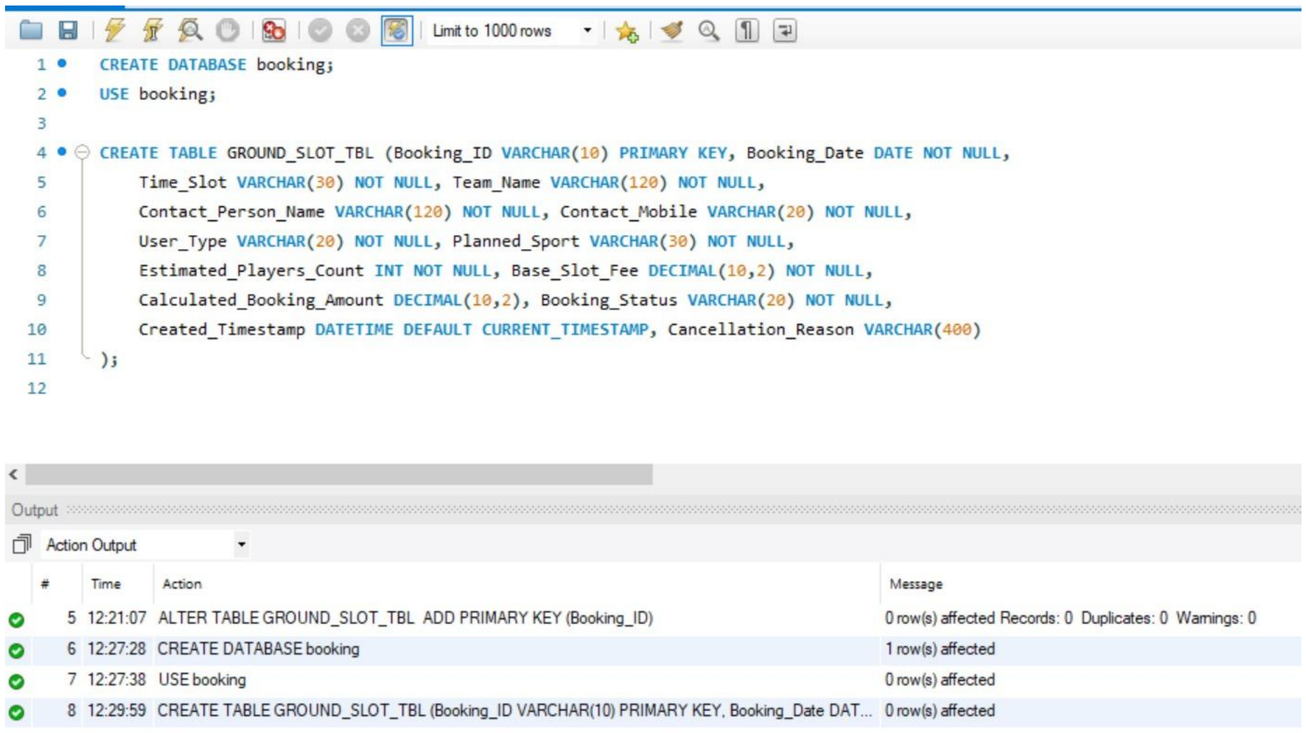
Create a dedicated database user and grant privileges using SQL commands. Then, create the required tables under the new schema to store booking and usage details for the sports ground management system.

Table: ZONE_TBL

Column	Datatype	Description
Booking ID	Varchar2(10)	Primary key uniquely identifying a ground booking.
Booking Date	Date	Date for which the ground slot is booked.
Time Slot	Varchar2(30)	Time window of the slot (e.g., 06:00-08:00).
Team / Group Name	Varchar2(120)	Name of the team or group booking the ground.
Contact Person Name	Varchar2(120)	Responsible person for the booking.
Contact Mobile	Varchar2(20)	Contact number for coordination.
User Type	Varchar2(10)	RESIDENT, GUEST_OF_RESIDENT, or EXTERNAL_TEAM
Planned Sport	Varchar2(30)	Planned activity such as FOOTBALL, CRICKET, etc.
Estimated Players Count	Number(5)	Approximate number of players expected.
Base Slot Fee	Number(10,2)	Base fee for resident users.
Calculated Booking Amount	Number(10,2)	Final booking amount after applying multipliers.
Booking Status	Varchar2(20)	PENDING, CONFIRMED, CANCELLED, or COMPLETED.
Created Timestamp	Date	Date and time of booking creation.
Cancellation Reason	Varchar2(400)	Reason for cancellation, if applicable.

Sample Records

Booking ID	Date	Time Slot	Team Name	User Type	Sport	Players	Amount	Status
GB1001	2025-04-10	06:00-08:00	Green Street FC	RESIDENT	FOOTBALL	14	300.00	CONFIRMED
GB1002	2025-04-10	16:00-18:00	Sunrise Cricket Group	EXTERNAL_TEAM	CRICKET	18	600.00	PENDING
GB1003	2025-04-11	06:00-08:00	Block-B Residents	RESIDENT	MIXED	20	300.00	CANCELLED



```
CREATE DATABASE booking;
USE booking;

CREATE TABLE GROUND_SLOT_TBL (Booking_ID VARCHAR(10) PRIMARY KEY, Booking_Date DATE NOT NULL,
    Time_Slot VARCHAR(30) NOT NULL, Team_Name VARCHAR(120) NOT NULL,
    Contact_Person_Name VARCHAR(120) NOT NULL, Contact_Mobile VARCHAR(20) NOT NULL,
    User_Type VARCHAR(20) NOT NULL, Planned_Sport VARCHAR(30) NOT NULL,
    Estimated_Players_Count INT NOT NULL, Base_Slot_Fee DECIMAL(10,2) NOT NULL,
    Calculated_Booking_Amount DECIMAL(10,2), Booking_Status VARCHAR(20) NOT NULL,
    Created_Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP, Cancellation_Reason VARCHAR(400)
);
```

```
#  Time      Action                                                                          Message
5  12:21:07  ALTER TABLE GROUND_SLOT_TBL ADD PRIMARY KEY (Booking_ID)                        0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
6  12:27:28  CREATE DATABASE booking                                                         1 row(s) affected
7  12:27:38  USE booking                                                                     0 row(s) affected
8  12:29:59  CREATE TABLE GROUND_SLOT_TBL (Booking_ID VARCHAR(10) PRIMARY KEY, Booking_Date DAT...  0 row(s) affected
```

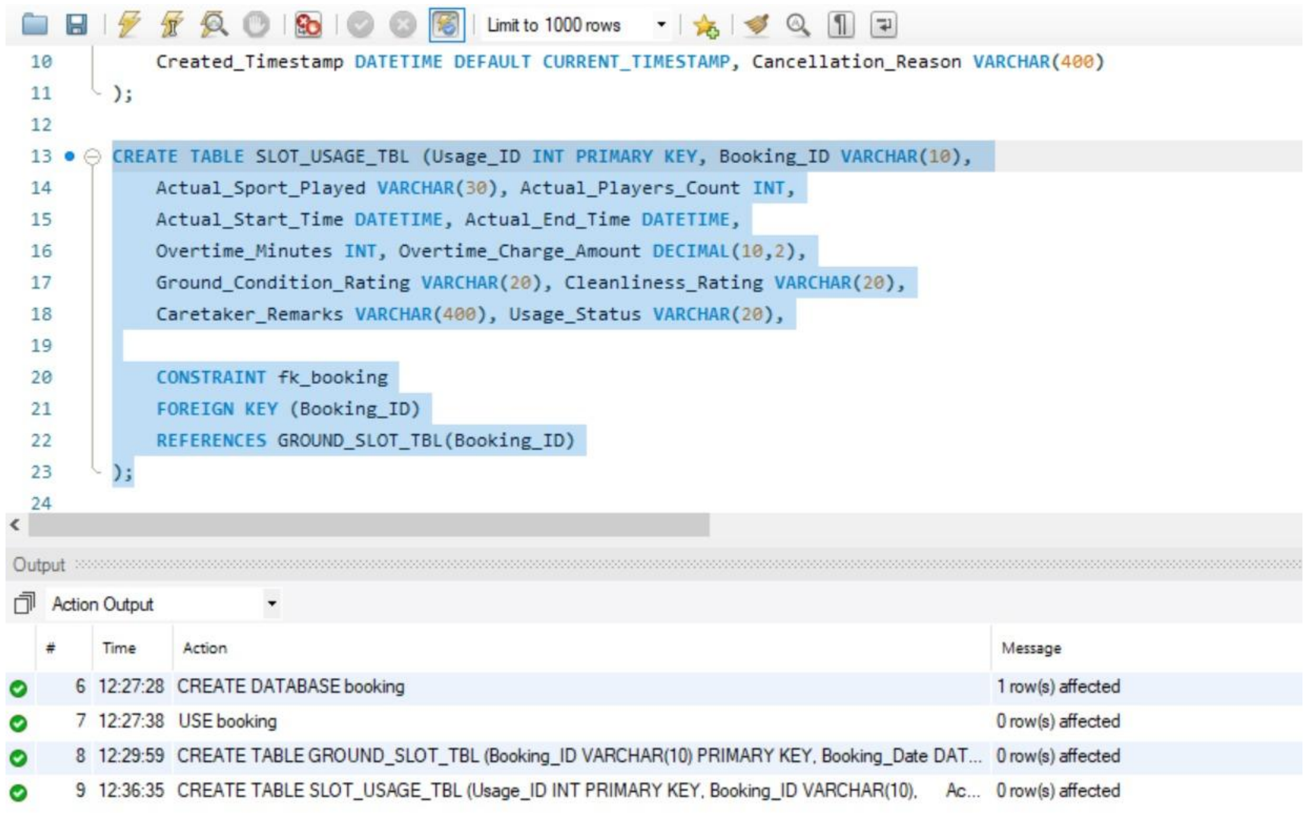
Table: SLOT_USAGE_TBL

Column Name	Datatype	Description
Usage ID	Number(10)	Primary key uniquely identifying each usage record.
Booking ID	Varchar2(10)	Foreign key referencing GROUND_SLOT_TBL.
Actual Sport Played	Varchar2(30)	Sport actually played.
Actual Players Count	Number(5)	Actual number of players participated.

Actual Start Time	Date	Actual start date and time.
Actual End Time	Date	Actual end date and time.
Overtime Minutes	Number(5)	Extra minutes used beyond booked slot.
Overtime Charge Amount	Number(10,2)	Additional charge for overtime usage.
Ground Condition Rating	Varchar2(20)	GOOD, MINOR_DAMAGE, or MAJOR_DAMAGE.
Cleanliness Rating	Varchar2(20)	CLEAN, MINOR_LITTER, or HEAVY_LITTER.
Caretaker Remarks	Varchar2(400)	Remarks recorded after usage.
Usage Status	Varchar2(20)	SUBMITTED or ADJUSTED.

Sample Records

Usage ID	Booking ID	Actual Sport Played	Actual Players Count	Actual Start Time	Actual End Time	Overtime Minutes	Overtime Charge Amount	1Ground Condition Rating
970001	GB1001	FOOTBALL	15	2025-04-10 06:05	2025-04-10 08:05	5	50.00	MINOR_DAMAGE
970002	GB1002	CRICKET	18	2025-04-10 16:00	2025-04-10 18:10	10	100.00	GOOD
970003	GB1003	MIXED	20	2025-04-11 06:00	2025-04-11 08:00	0	0.00	GOOD



```
    Created_Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP, Cancellation_Reason VARCHAR(400)
);

CREATE TABLE SLOT_USAGE_TBL (Usage_ID INT PRIMARY KEY, Booking_ID VARCHAR(10),
    Actual_Sport_Played VARCHAR(30), Actual_Players_Count INT,
    Actual_Start_Time DATETIME, Actual_End_Time DATETIME,
    Overtime_Minutes INT, Overtime_Charge_Amount DECIMAL(10,2),
    Ground_Condition_Rating VARCHAR(20), Cleanliness_Rating VARCHAR(20),
    Caretaker_Remarks VARCHAR(400), Usage_Status VARCHAR(20),

    CONSTRAINT fk_booking
    FOREIGN KEY (Booking_ID)
    REFERENCES GROUND_SLOT_TBL(Booking_ID)
);
```

```
#  Time      Action                                                                                  Message
6  12:27:28  CREATE DATABASE booking                                                                 1 row(s) affected
7  12:27:38  USE booking                                                                             0 row(s) affected
8  12:29:59  CREATE TABLE GROUND_SLOT_TBL (Booking_ID VARCHAR(10) PRIMARY KEY, Booking_Date DAT...    0 row(s) affected
9  12:36:35  CREATE TABLE SLOT_USAGE_TBL (Usage_ID INT PRIMARY KEY, Booking_ID VARCHAR(10),    Ac...  0 row(s) affected
```

DBUtil.java

```
package com.ground.util;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBUtil {
    private static final String URL = "jdbc:mysql://localhost:3306/ground_db";
    private static final String USER = "ground_user";
    private static final String PASSWORD = "ground_pwd";
    public static Connection getDBConnection() throws Exception {
        Class.forName("com.mysql.cj.jdbc.Driver");
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

ValidationException.java

```
package com.ground.util;

public class ValidationException extends Exception {
    public ValidationException(String msg) {
        super(msg);
    }
    @Override
    public String toString() {
        return "Validation Error: " + getMessage();
    }
}
```

UsageAlreadyExistsException.java

```
package com.ground.util;
public class UsageAlreadyExistsException extends Exception {
    public UsageAlreadyExistsException(String msg) {
        super(msg);
    }
    @Override
    public String toString() {
        return "Usage Already Exists: " + getMessage();
    }
}
```

SlotOverlapException.java

```
package com.ground.util;
public class SlotOverlapException extends Exception {
    public SlotOverlapException(String msg) {
        super(msg);
    }
    @Override
    public String toString() {
        return "Slot Overlap Error: " + getMessage();
    }
}
```

Package: com.ground.bean

GroundSlot.java

```
package com.ground.bean;
import java.sql.Date;
import java.sql.Timestamp;
public class GroundSlot {
    private String bookingID;
    private Date bookingDate;
    private String timeSlot;
    private String teamOrGroupName;
    private String contactPersonName;
    private String contactMobile;
    private String userType;
    private String plannedSport;
    private int estimatedPlayersCount;
    private double baseSlotFee;
    private double calculatedBookingAmount;
    private String bookingStatus;
    private Timestamp createdTimestamp;
    private String cancellationReason;
    public String getBookingID() { return bookingID; }
    public void setBookingID(String bookingID) {
 this.bookingID = bookingID; }
    public Date getBookingDate() {
return bookingDate; }
    public void setBookingDate(Date bookingDate) {
 this.bookingDate = bookingDate; }
    public String getTimeSlot() {
return timeSlot; }
    public void setTimeSlot(String timeSlot) {
this.timeSlot = timeSlot; }
    public String getTeamOrGroupName() {
 return teamOrGroupName; }
```

```
    public void setTeamOrGroupName(String teamOrGroupName) {
this.teamOrGroupName = teamOrGroupName; }
    public String getContactPersonName() {
 return contactPersonName; }
    public void setContactPersonName(String contactPersonName) {
this.contactPersonName = contactPersonName; }
    public String getContactMobile() {
 return contactMobile; }
    public void setContactMobile(String contactMobile) {
this.contactMobile = contactMobile; }
    public String getUserType() {
return userType; }
    public void setUserType(String userType) {
this.userType = userType; }
    public String getPlannedSport() {
return plannedSport; }
    public void setPlannedSport(String plannedSport) {
this.plannedSport = plannedSport; }
    public int getEstimatedPlayersCount() {
return estimatedPlayersCount; }
    public void setEstimatedPlayersCount(int estimatedPlayersCount) {
this.estimatedPlayersCount = estimatedPlayersCount; }
    public double getBaseSlotFee() {
 return baseSlotFee; }
    public void setBaseSlotFee(double baseSlotFee) {
 this.baseSlotFee = baseSlotFee; }
    public double getCalculatedBookingAmount() {
 return calculatedBookingAmount; }
    public void setCalculatedBookingAmount(double calculatedBookingAmount) {
        this.calculatedBookingAmount = calculatedBookingAmount;
    }
    public String getBookingStatus() {
return bookingStatus; }
    public void setBookingStatus(String bookingStatus) {
this.bookingStatus = bookingStatus; }
```

```
    public Timestamp getCreatedTimestamp() {
return createdTimestamp; }
    public void setCreatedTimestamp(Timestamp createdTimestamp) {
        this.createdTimestamp = createdTimestamp;
    }
    public String getCancellationReason() {
 return cancellationReason; }
    public void setCancellationReason(String cancellationReason) {
        this.cancellationReason = cancellationReason;
    }
  }
```

SlotUsage.java

```
package com.ground.bean;

import java.sql.Timestamp;

public class SlotUsage {

    private int usageID;
    private String bookingID;
    private String actualSportPlayed;
    private int actualPlayersCount;
    private Timestamp actualStartTime;
    private Timestamp actualEndTime;
    private int overtimeMinutes;
    private double overtimeChargeAmount;
    private String groundConditionRating;
    private String cleanlinessRating;
    private String caretakerRemarks;
    private String usageStatus;
    public int getUsageID() {
return usageID; }
```

```
    public void setUsageID(int usageID) {
  this.usageID = usageID; }
    public String getBookingID() {
  return bookingID; }
    public void setBookingID(String bookingID) {
this.bookingID = bookingID; }
    public String getActualSportPlayed() {
  return actualSportPlayed; }
    public void setActualSportPlayed(String actualSportPlayed) {
 this.actualSportPlayed = actualSportPlayed;
    }

    public int getActualPlayersCount() {
  return actualPlayersCount; }
    public void setActualPlayersCount(int actualPlayersCount) {
  this.actualPlayersCount = actualPlayersCount;
    }

    public Timestamp getActualStartTime() {
  return actualStartTime; }
    public void setActualStartTime(Timestamp actualStartTime) {
  this.actualStartTime = actualStartTime;
    }

    public Timestamp getActualEndTime() {
  return actualEndTime; }
    public void setActualEndTime(Timestamp actualEndTime) {
 this.actualEndTime = actualEndTime;
    }

    public int getOvertimeMinutes() {
   return overtimeMinutes; }
    public void setOvertimeMinutes(int overtimeMinutes) {
 this.overtimeMinutes = overtimeMinutes;
    }
    public double getOvertimeChargeAmount() {
  return overtimeChargeAmount; }
    public void setOvertimeChargeAmount(double overtimeChargeAmount) {
 this.overtimeChargeAmount = overtimeChargeAmount;
    }

    public String getGroundConditionRating() {
  return groundConditionRating; }
    public void setGroundConditionRating(String groundConditionRating) {
  this.groundConditionRating = groundConditionRating;
    }

    public String getCleanlinessRating() {
  return cleanlinessRating; }
    public void setCleanlinessRating(String cleanlinessRating) {
  this.cleanlinessRating = cleanlinessRating;
    }

    public String getCaretakerRemarks() {
  return caretakerRemarks; }
    public void setCaretakerRemarks(String caretakerRemarks) {
this.caretakerRemarks = caretakerRemarks;
    }

    public String getUsageStatus() { return usageStatus; }
    public void setUsageStatus(String usageStatus) {
this.usageStatus = usageStatus;
    }
 }
```

Package: com.ground.dao

GroundSlotDAO.java

```
package com.ground.dao;

import com.ground.bean.GroundSlot;
import com.ground.util.DBUtil;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class GroundSlotDAO {

    public GroundSlot findBooking(String bookingID) throws Exception {
        Connection con = DBUtil.getDBConnection();
        PreparedStatement ps = con.prepareStatement(
                "SELECT * FROM GROUND_SLOT_TBL WHERE booking_id=?");
        ps.setString(1, bookingID);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            GroundSlot g = new GroundSlot();
            g.setBookingID(rs.getString("booking_id"));
            g.setBookingDate(rs.getDate("booking_date"));
            g.setTimeSlot(rs.getString("time_slot"));
            g.setBookingStatus(rs.getString("booking_status"));
            return g;
        }
        return null;
    }

    public boolean insertBooking(GroundSlot slot) throws Exception {
        Connection con = DBUtil.getDBConnection();
        PreparedStatement ps = con.prepareStatement(
            "INSERT INTO GROUND_SLOT_TBL VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?)");

        ps.setString(1, slot.getBookingID());
        ps.setDate(2, slot.getBookingDate());
        ps.setString(3, slot.getTimeSlot());
        ps.setString(4, slot.getTeamOrGroupName());
        ps.setString(5, slot.getContactPersonName());
        ps.setString(6, slot.getContactMobile());
        ps.setString(7, slot.getUserType());
        ps.setString(8, slot.getPlannedSport());
        ps.setInt(9, slot.getEstimatedPlayersCount());
        ps.setDouble(10, slot.getBaseSlotFee());
        ps.setDouble(11, slot.getCalculatedBookingAmount());
        ps.setString(12, slot.getBookingStatus());
        ps.setTimestamp(13, slot.getCreatedTimestamp());

        return ps.executeUpdate() == 1;
    }

    public boolean updateBookingStatusAndCancellation(
            String bookingID, String status, String reason) throws Exception {

        Connection con = DBUtil.getDBConnection();
        PreparedStatement ps = con.prepareStatement(
            "UPDATE GROUND_SLOT_TBL SET booking_status=?, cancellation_reason=? WHERE
booking_id=?");

        ps.setString(1, status);
        ps.setString(2, reason);
        ps.setString(3, bookingID);

        return ps.executeUpdate() == 1;
    }
```

SlotUageDAO.java

```
package com.ground.dao;
import com.ground.bean.SlotUsage;
import com.ground.util.DBUtil;
import java.sql.*;
public class SlotUsageDAO {

    public int generateUsageID() throws Exception {
        Connection con = DBUtil.getDBConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT MAX(usage_id) FROM SLOT_USAGE_TBL");
        if (rs.next()) return rs.getInt(1) + 1;
        return 1;
    }
    public boolean insertSlotUsage(SlotUsage usage) throws Exception {
        Connection con = DBUtil.getDBConnection();
        PreparedStatement ps = con.prepareStatement(
                "INSERT INTO SLOT_USAGE_TBL VALUES (?,?,?,?,?,?,?,?,?,?,?,?)");
        ps.setInt(1, usage.getUsageID());
        ps.setString(2, usage.getBookingID());
        ps.setString(3, usage.getActualSportPlayed());
        ps.setInt(4, usage.getActualPlayersCount());
        ps.setTimestamp(5, usage.getActualStartTime());
        ps.setTimestamp(6, usage.getActualEndTime());
        ps.setInt(7, usage.getOvertimeMinutes());
        ps.setDouble(8, usage.getOvertimeChargeAmount());
        ps.setString(9, usage.getGroundConditionRating());
        ps.setString(10, usage.getCleanlinessRating());
        ps.setString(11, usage.getCaretakerRemarks());
        ps.setString(12, usage.getUsageStatus());
        return ps.executeUpdate() == 1;
    }
}
```

Package: com.ground.service

GroundService.java

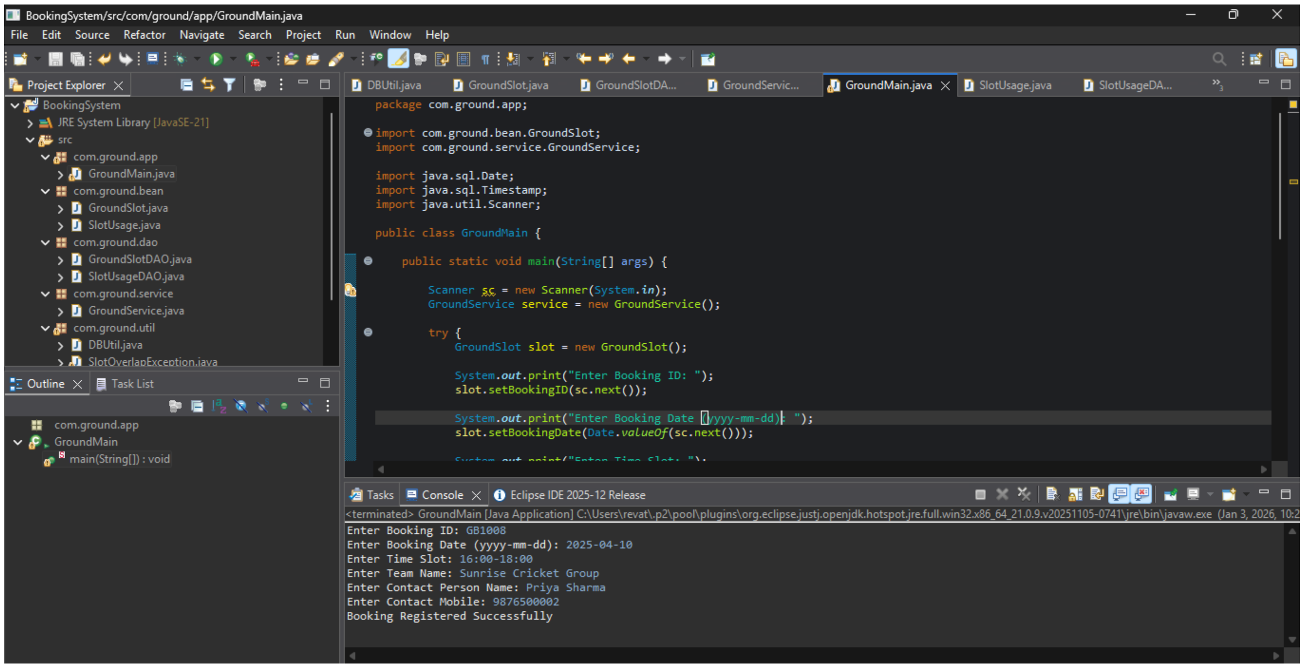
```
package com.ground.service;
import com.ground.bean.*;
import com.ground.dao.*;
import com.ground.util.*;
import java.sql.Timestamp;
public class GroundService {
   GroundSlotDAO slotDAO = new GroundSlotDAO();
   SlotUsageDAO usageDAO = new SlotUsageDAO();
   public boolean registerNewBooking(GroundSlot slot,
                                    double residentMul,
                                    double guestMul,
                                    double externalMul) throws Exception {

      if (slot.getBookingID() == null || slot.getBookingID().isEmpty())
         throw new ValidationException("Booking ID required");
      double amount = slot.getBaseSlotFee();
      if (slot.getUserType().equals("RESIDENT"))
         amount *= residentMul;
      else if (slot.getUserType().equals("GUEST_OF_RESIDENT"))
         amount *= guestMul;
      else
         amount *= externalMul;
      slot.setCalculatedBookingAmount(amount);
      slot.setBookingStatus("PENDING");
      slot.setCreatedTimestamp(new Timestamp(System.currentTimeMillis()));
      return slotDAO.insertBooking(slot);
   }
}
```

Package: com.ground.app

Demo Main Code

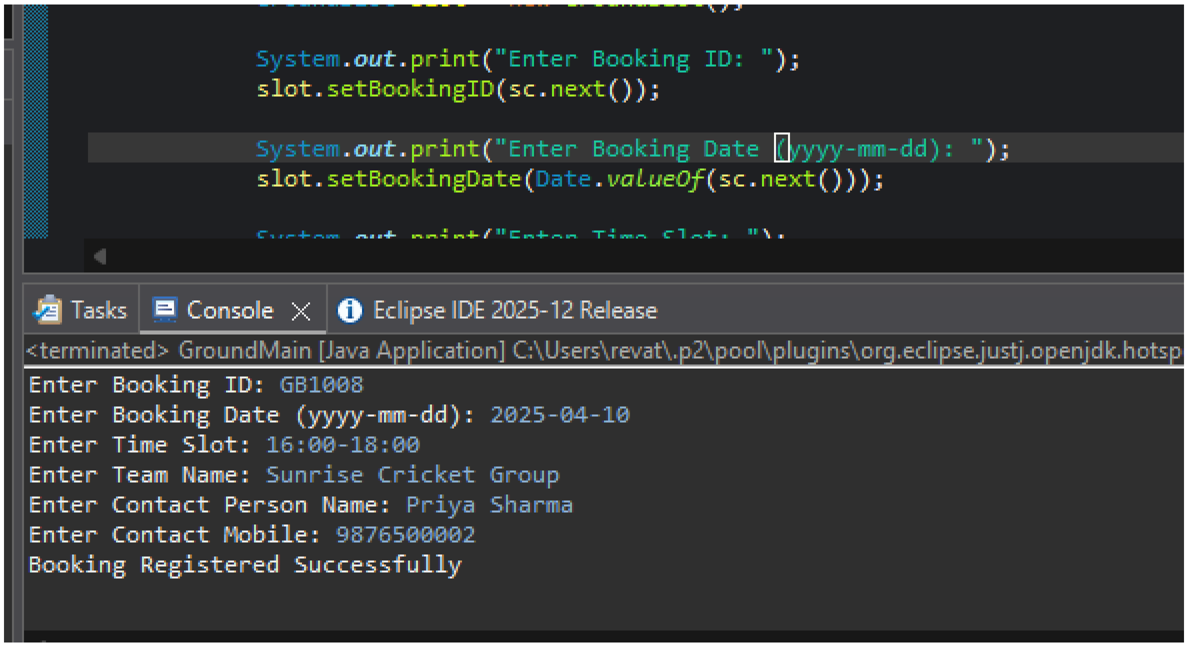
```
package com.ground.app;
import com.ground.bean.GroundSlot;
import com.ground.service.GroundService;
import java.sql.Date;
import java.util.Scanner;
public class GroundMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        GroundService service = new GroundService();
        try {
            GroundSlot slot = new GroundSlot();
            System.out.print("Enter Booking ID: ");
            slot.setBookingID(sc.next());
            System.out.print("Enter Booking Date (yyyy-mm-dd): ");
            slot.setBookingDate(Date.valueOf(sc.next()));
            System.out.print("Enter Time Slot: ");
            slot.setTimeSlot(sc.next());
            System.out.print("Enter Team Name: ");
            slot.setTeamOrGroupName(sc.next());
            slot.setUserType("RESIDENT");
            slot.setPlannedSport("Football");
            slot.setEstimatedPlayersCount(10);
            slot.setBaseSlotFee(1000);
            boolean result = service.registerNewBooking(slot, 1.0, 1.2, 1.5);
            System.out.println(result ? "Booking Registered" : "Failed");
        } catch (Exception e) {
            System.out.println(e);
        }
    }}
```

OUTPUT:



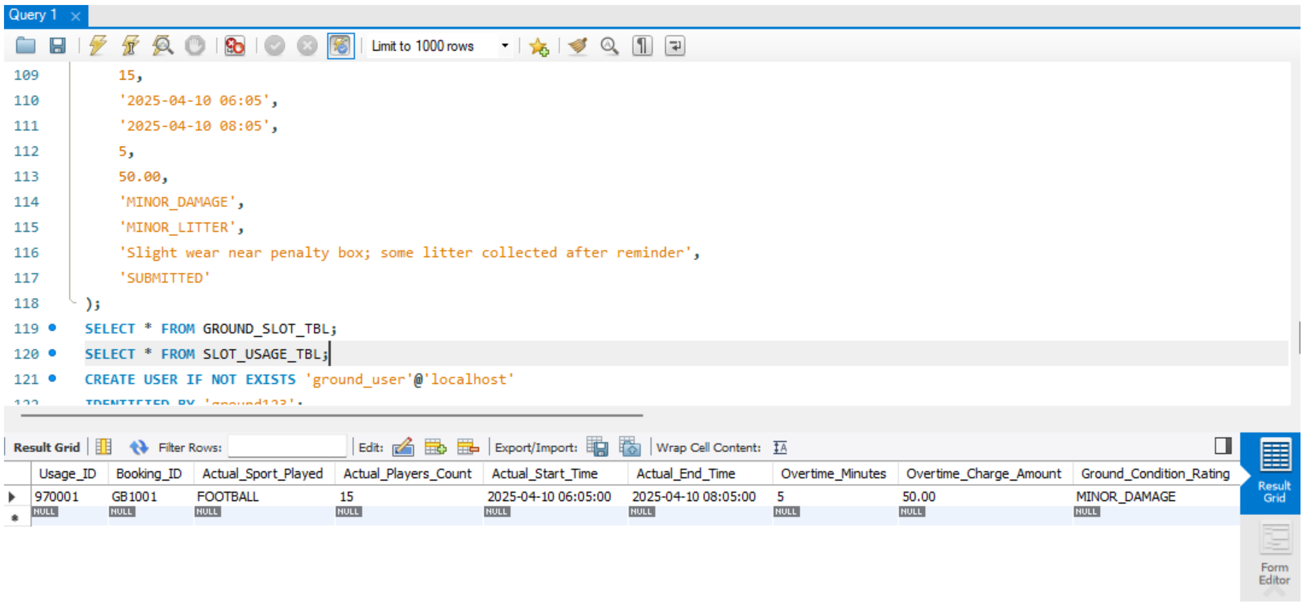
```java
package com.ground.app;

import com.ground.bean.GroundSlot;
import com.ground.service.GroundService;

import java.sql.Date;
import java.sql.Timestamp;
import java.util.Scanner;

public class GroundMain {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        GroundService service = new GroundService();

        try {
            GroundSlot slot = new GroundSlot();

            System.out.print("Enter Booking ID: ");
            slot.setBookingID(sc.next());

            System.out.print("Enter Booking Date (yyyy-mm-dd): ");
            slot.setBookingDate(Date.valueOf(sc.next()));
```

```
<terminated> GroundMain [Java Application] C:\Users\revat\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.9.v20251105-0741\jre\bin\javaw.exe (Jan 3, 2026, 10:2
Enter Booking ID: GB1008
Enter Booking Date (yyyy-mm-dd): 2025-04-10
Enter Time Slot: 16:00-18:00
Enter Team Name: Sunrise Cricket Group
Enter Contact Person Name: Priya Sharma
Enter Contact Mobile: 9876500002
Booking Registered Successfully
```



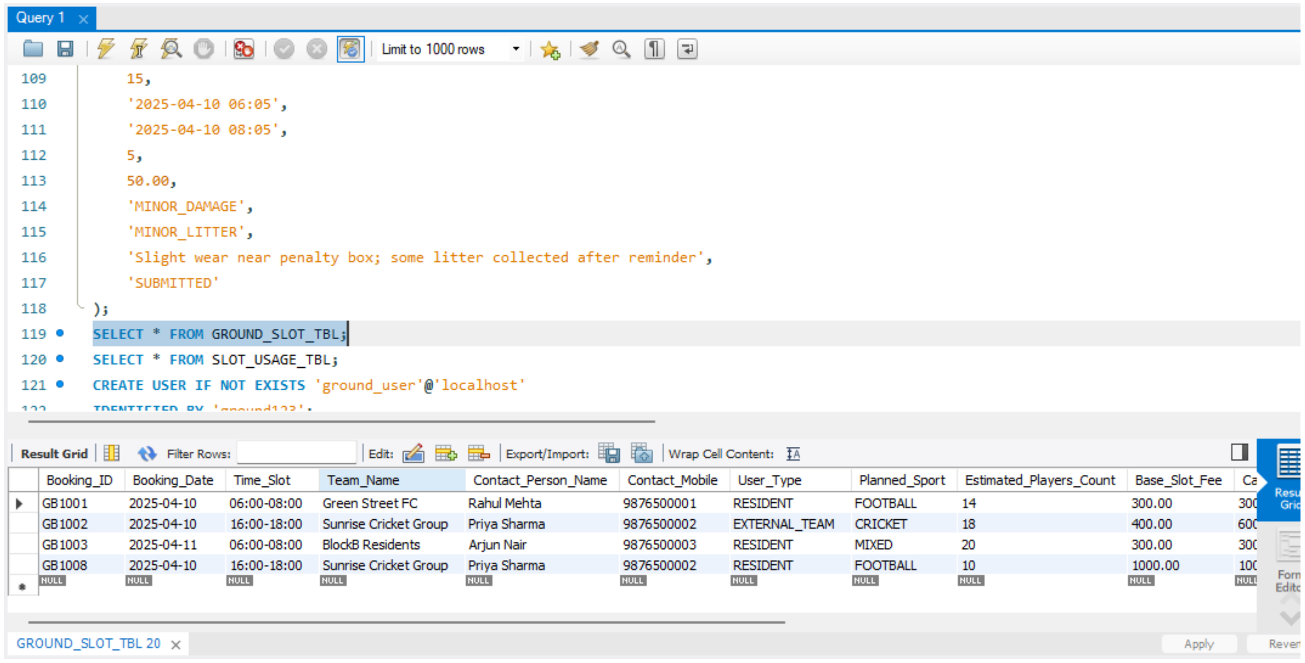
```java
            System.out.print("Enter Booking ID: ");
            slot.setBookingID(sc.next());

            System.out.print("Enter Booking Date (yyyy-mm-dd): ");
            slot.setBookingDate(Date.valueOf(sc.next()));
```

```
<terminated> GroundMain [Java Application] C:\Users\revat\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
Enter Booking ID: GB1008
Enter Booking Date (yyyy-mm-dd): 2025-04-10
Enter Time Slot: 16:00-18:00
Enter Team Name: Sunrise Cricket Group
Enter Contact Person Name: Priya Sharma
Enter Contact Mobile: 9876500002
Booking Registered Successfully
```



```
    15,
    '2025-04-10 06:05',
    '2025-04-10 08:05',
    5,
    50.00,
    'MINOR_DAMAGE',
    'MINOR_LITTER',
    'Slight wear near penalty box; some litter collected after reminder',
    'SUBMITTED'
);
SELECT * FROM GROUND_SLOT_TBL;
SELECT * FROM SLOT_USAGE_TBL;
CREATE USER IF NOT EXISTS 'ground_user'@'localhost'
```

Usage_ID	Booking_ID	Actual_Sport_Played	Actual_Players_Count	Actual_Start_Time	Actual_End_Time	Overtime_Minutes	Overtime_Charge_Amount	Ground_Condition_Rating
970001	GB1001	FOOTBALL	15	2025-04-10 06:05:00	2025-04-10 08:05:00	5	50.00	MINOR_DAMAGE
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
    15,
    '2025-04-10 06:05',
    '2025-04-10 08:05',
    5,
    50.00,
    'MINOR_DAMAGE',
    'MINOR_LITTER',
    'Slight wear near penalty box; some litter collected after reminder',
    'SUBMITTED'
);
SELECT * FROM GROUND_SLOT_TBL;
SELECT * FROM SLOT_USAGE_TBL;
CREATE USER IF NOT EXISTS 'ground_user'@'localhost'
```

Booking_ID	Booking_Date	Time_Slot	Team_Name	Contact_Person_Name	Contact_Mobile	User_Type	Planned_Sport	Estimated_Players_Count	Base_Slot_Fee	Ca
GB1001	2025-04-10	06:00-08:00	Green Street FC	Rahul Mehta	9876500001	RESIDENT	FOOTBALL	14	300.00	300
GB1002	2025-04-10	16:00-18:00	Sunrise Cricket Group	Priya Sharma	9876500002	EXTERNAL_TEAM	CRICKET	18	400.00	600
GB1003	2025-04-11	06:00-08:00	BlockB Residents	Arjun Nair	9876500003	RESIDENT	MIXED	20	300.00	300
GB1008	2025-04-10	16:00-18:00	Sunrise Cricket Group	Priya Sharma	9876500002	RESIDENT	FOOTBALL	10	1000.00	100
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL