Hotel Maintenance Management System

Enhancing Operational Efficiency and Guest Satisfaction Through Streamlined Maintenance Workflows

VISHAL JAVVAJI

HORIZON
STAY CORP

INDEX

S. No.	Topic	Page No.
1	Introduction	3
2	Project Overview	4
3	Purpose of the System	4
4	Database Design	5
5	Key Tables and Their Roles	6 – 8
6	Relationships between Tables	8
7	Data Integrity Rules	9
8	Field Design	9-10
9	View 1 – Request History	10
10	View 2 – Open or Pending Requests	11
11	View 3 – Employee Task List	11 - 12
12	Sample Data	12
13	Realistic Use Case	12
14	SQL Implementation	13
15	SQL Queries	13 - 15
16	Performance Benefits	15
17	Future Enhancements	15
18	Conclusion	16

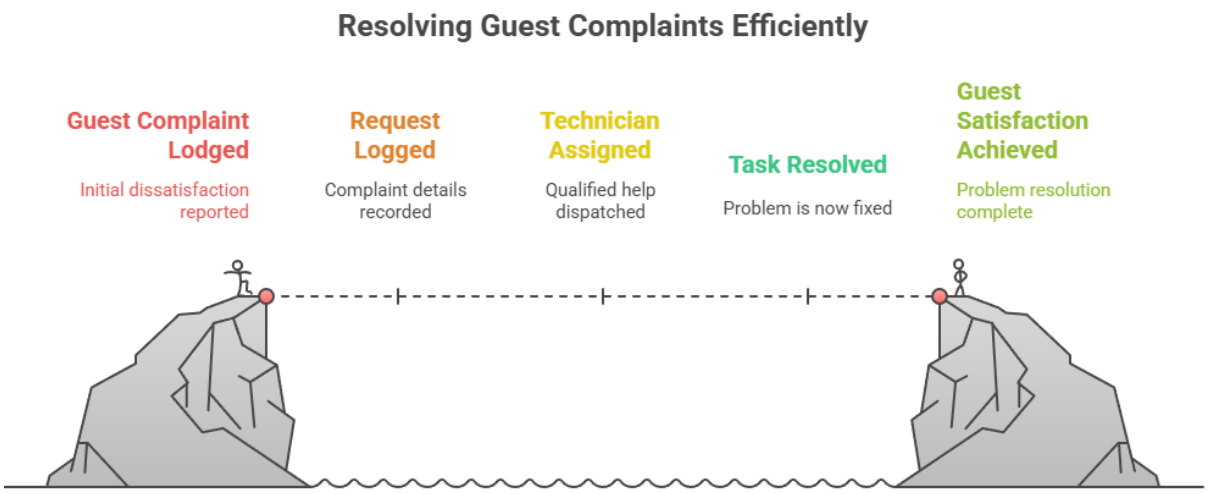
1. Introduction

In the hospitality industry, providing seamless guest experiences is key to success. One area that directly impacts satisfaction is how hotels manage room maintenance issues. This article presents a simplified and scalable Hotel Maintenance Management System. It is based on real database development practices using MySQL and includes features like structured maintenance logging, employee task tracking, and guest interaction records

2. Project Overview

The system was designed to meet the following goals:

- Log maintenance requests efficiently.
- Track the status of maintenance activities.
- Assign work to hotel staff based on roles and shifts.
- Ensure guests can raise issues and receive timely support.
- Allow managers to monitor patterns and response efficiency.



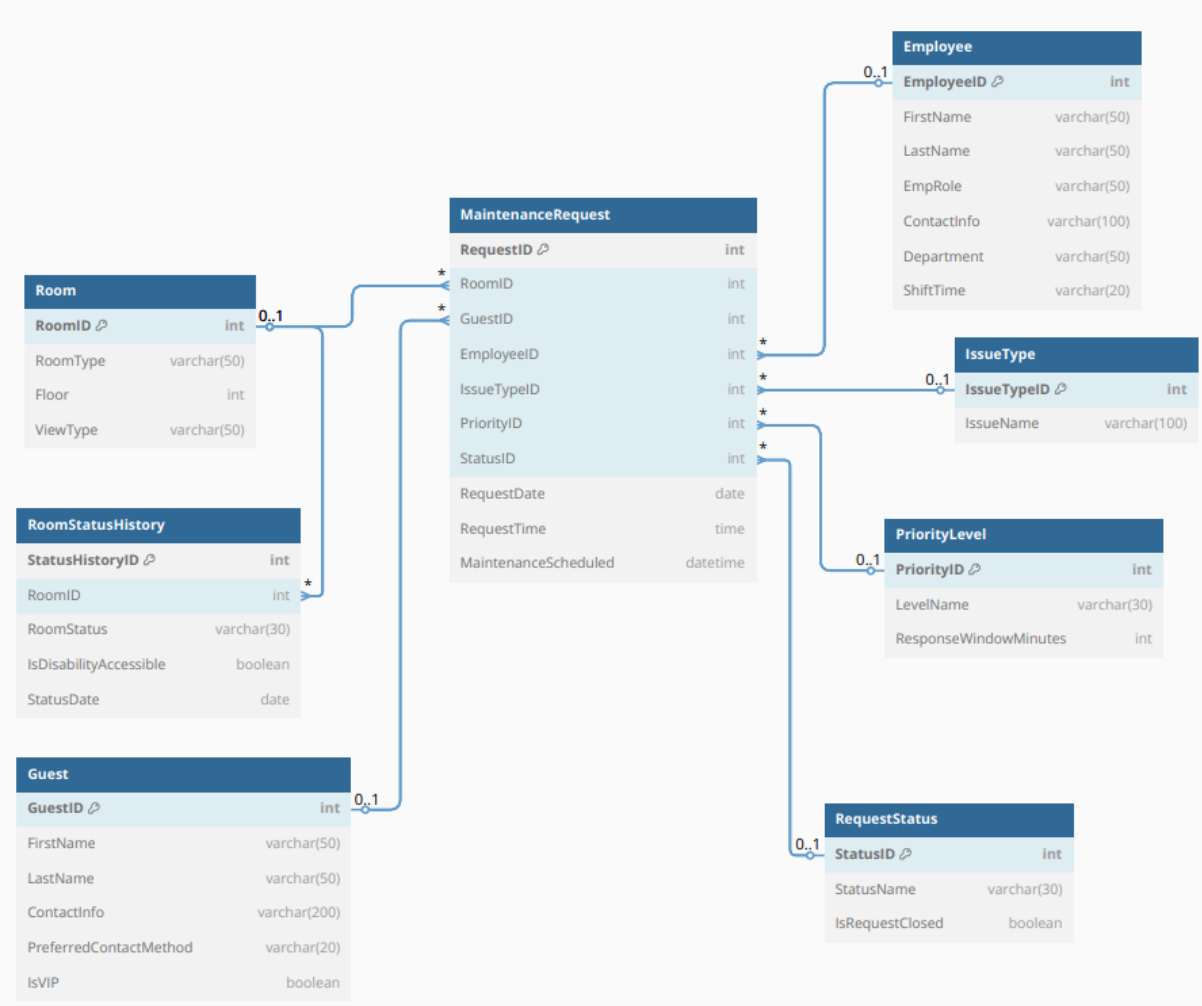
3. Purpose of the System

The core purpose of this system is to streamline room maintenance operations and improve service quality in hotels. With this system:

- Staff are assigned tasks systematically.
- Guests get faster issue resolution.
- Managers gain visibility into performance metrics.

4. Database Design

A relational database structure was used for this system. Tables are logically connected using primary and foreign keys to maintain data consistency and integrity.



5. Key Tables and Their Roles

I. Room

Field Name	Data Type	Description
RoomID	INT (PK)	Unique identifier for each room
RoomType	VARCHAR(50)	Type of room (e.g., Deluxe, Standard)
Floor	INT	Floor number where the room is located
ViewType	VARCHAR(50)	Room view type (e.g., Ocean, Garden, City)

II. Guest

Field Name	Data Type	Description
GuestID	INT (PK)	Unique identifier for each guest
FirstName	VARCHAR(50)	Guest's first name
LastName	VARCHAR(50)	Guest's last name
ContactInfo	VARCHAR(200)	Email or phone number
PreferredContactMethod	VARCHAR(20)	Preferred contact type (Email, Phone, SMS)
IsVIP	BOOLEAN	Indicates if the guest has VIP status

III. Employee

Field Name	Data Type	Description
EmployeeID	INT (PK)	Unique identifier for staff member
FirstName	VARCHAR(50)	Employee's first name
LastName	VARCHAR(50)	Employee's last name
EmpRole	VARCHAR(50)	Role (e.g., Technician, Supervisor)
ContactInfo	VARCHAR(100)	Contact number
Department	VARCHAR(50)	Department assigned (e.g., Plumbing, Cleaning)
ShiftTime	VARCHAR(20)	Work shift (Morning, Evening, Night)

IV. IssueType

Field Name	Data Type	Description
IssueTypeID	INT (PK)	Unique identifier for issue type
IssueName	VARCHAR(100)	Type of issue (e.g., AC, Plumbing)

V. PriorityLevel

Field Name	Data Type	Description
PriorityID	INT (PK)	Unique identifier for priority level
LevelName	VARCHAR(30)	Name of the priority level (High, Medium)
ResponseWindowMinutes	INT	Target resolution time in minutes

VI. RequestStatus

Field Name	Data Type	Description
StatusID	INT (PK)	Unique identifier for request status
StatusName	VARCHAR(30)	Status of the request (Open, In Progress)
IsRequestClosed	BOOLEAN	TRUE if the request is completed/resolved

VII. RoomStatusHistory

Field Name	Data Type	Description
StatusHistoryID	INT (PK, Auto)	Unique ID for each room status log
RoomID	INT (FK)	Room whose status is being logged
RoomStatus	VARCHAR(30)	Current room status (Available, Occupied, etc.)
IsDisabilityAccessible	BOOLEAN	Indicates whether room is accessible
StatusDate	DATE	Date when the status was recorded

VIII. MaintenanceRequest

Field Name	Data Type	Description
RequestID	INT (PK)	Unique ID for each maintenance request
RoomID	INT (FK)	Room where the issue was reported
GuestID	INT (FK)	Guest who raised the request
EmployeeID	INT (FK)	Staff assigned to the request
IssueTypeID	INT (FK)	Type of issue reported
PriorityID	INT (FK)	Level of urgency
StatusID	INT (FK)	Current status of the request
RequestDate	DATE	Date the request was created
RequestTime	TIME	Time the request was made
MaintenanceScheduled	DATETIME	Scheduled time for performing the maintenance

6. Relationships between tables

- **Room → RoomStatusHistory**
  One room can have many status updates over time.
- **Room → MaintenanceRequest**
  One room can have multiple maintenance requests.
- **Guest → MaintenanceRequest**
  One guest can submit multiple maintenance requests.
- **Employee → MaintenanceRequest**
  One employee can be assigned to many maintenance tasks.
- **Employee → ResolutionLog**
  One employee can update multiple resolution logs.
- **MaintenanceRequest → ResolutionLog**
  One request can be linked to multiple resolution entries.
- **IssueType → MaintenanceRequest**
  One issue type can be associated with many requests.
- **PriorityLevel → MaintenanceRequest**
  One priority level can apply to many maintenance requests.
- **RequestStatus → MaintenanceRequest**
  One status type can be shared across many requests.

7. Data Integrity Rules

- **Primary Keys**: Ensure uniqueness (e.g., GuestID, RequestID).
- **Foreign Keys**: Maintain table links (e.g., RoomID in MaintenanceRequest).
- **Data Types**: Proper types like DATE, TIME, and BOOLEAN.
- **Boolean Flags**: Used for yes/no indicators (e.g., IsVIP).

---

8. Field Design

Fields were selected based on their real-world application in hotel operations. Final fields were chosen after eliminating unrelated or redundant fields.

Preliminary Field List

- Room ID
- Room Type
- Floor
- View Type
- Room Status
- Is Disability Accessible
- Status Date
- Guest ID
- First Name
- Last Name
- Contact Info
- Preferred Contact Method
- Room Cleaning Frequency
- VIP
- Employee ID
- Role
- Contact Info
- Department
- Shift Time
- Issue ID
- Issue Name
- Priority ID
- Level Name
- Weather Condition
- Wi-Fi Speed
- Guest Feedback Score
- Response Window Minutes
- Status ID
- Status Name
- Is Request Closed
- Request ID
- Request Date
- Request Time
- Log ID
- Resolution Time
- Updated By EmployeeID
- Log Updated At
- Number Of Pillows
- Maintenance Scheduled

Final Field List

- Room ID
- Room Type
- Floor
- View Type
- Room Status
- Is Disability Accessible
- Status Date
- Guest ID
- First Name
- Last Name
- Contact Info
- Preferred Contact Method
- ~~Room Cleaning Frequency~~
- VIP
- Employee ID
- Role
- Contact Info
- Department
- Shift Time
- Issue ID
- Issue Name
- Priority ID
- Level Name
- ~~Weather Condition~~
- ~~Wi-Fi Speed~~
- ~~Guest Feedback Score~~
- Response Window Minutes
- Status ID
- Status Name
- Is Request Closed
- Request ID
- Request Date
- Request Time
- Log ID
- Resolution Time
- Updated By EmployeeID
- Log Updated At
- ~~Number Of Pillows~~
- Maintenance Scheduled

---

9. View 1 – Request History

This view shows all past maintenance requests, combining guest details, room information, and request status. It helps:

- Monitor past problems.
- Identify recurring issues.
- Plan maintenance schedules better.

```sql
-- View 1: Request History
CREATE OR REPLACE VIEW RequestHistory AS
SELECT
    mr.RequestID,
    CONCAT(g.FirstName, ' ', g.LastName) AS GuestName,
    r.RoomID AS Room_No,
    r.RoomType,
    it.IssueName,
    pl.LevelName AS Issue_Level,
    rs.StatusName,
    CAST(CONCAT(mr.RequestDate, ' ', mr.RequestTime) AS DATETIME) AS RequestedAt,
    mr.MaintenanceScheduled
FROM MaintenanceRequest mr
JOIN Guest g ON mr.GuestID = g.GuestID
JOIN Room r ON mr.RoomID = r.RoomID
JOIN IssueType it ON mr.IssueTypeID = it.IssueTypeID
JOIN PriorityLevel pl ON mr.PriorityID = pl.PriorityID
JOIN RequestStatus rs ON mr.StatusID = rs.StatusID;
SELECT * FROM RequestHistory
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

RequestID	GuestName	Room_No	RoomType	IssueName	Issue_Level	StatusName	RequestedAt	MaintenanceScheduled
9016	Isabella Clark	401	Standard	TV Malfunction	High	Resolved	2025-05-30 21:45:00	2025-05-30 22:45:00
9011	William Jackson	301	Deluxe	Internet Connection	High	In Progress	2025-05-30 18:45:00	2025-05-30 19:45:00
9001	John Doe	101	Deluxe	Air Conditioning	High	Open	2025-05-30 09:00:00	2025-05-30 10:00:00
9006	Sarah Miller	201	Standard	Mini Bar Refill	High	Open	2025-05-30 14:00:00	2025-05-30 15:00:00
9007	James Wilson	202	Deluxe	Air Conditioning	Medium	Resolved	2025-05-30 15:20:00	2025-05-30 16:20:00

10. View 2 – Open or Pending Requests

This focuses on active issues. Filters allow sorting by priority, date, or room location. Useful for:

- Managing technician workload.
- Prioritizing urgent problems.

```
-- View 2: Open/Pending Requests
CREATE OR REPLACE VIEW OpenPendingRequests AS
SELECT * FROM RequestHistory
WHERE StatusName IN ('Open', 'In Progress');

SELECT * FROM OpenPendingRequests
Limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

RequestID	GuestName	Room_No	RoomType	IssueName	Issue_Level	StatusName	RequestedAt	MaintenanceScheduled
9011	William Jackson	301	Deluxe	Internet Connection	High	In Progress	2025-05-30 18:45:00	2025-05-30 19:45:00
9001	John Doe	101	Deluxe	Air Conditioning	High	Open	2025-05-30 09:00:00	2025-05-30 10:00:00
9006	Sarah Miller	201	Standard	Mini Bar Refill	High	Open	2025-05-30 14:00:00	2025-05-30 15:00:00
9002	Jane Smith	102	Standard	Plumbing	Medium	In Progress	2025-05-30 10:30:00	2025-05-30 11:30:00
9012	Sophia White	302	Standard	Mini Bar Refill	Medium	Open	2025-05-30 19:20:00	2025-05-30 20:20:00

11. View 3 – Employee Task List

Assigns and tracks work by employee. Useful for:

- Supervisors to monitor staff activity.
- Ensuring no employee is overburdened.
- Improving accountability.

```sql
-- View 3: Employee Task List
CREATE OR REPLACE VIEW EmployeeTaskList AS
SELECT
    e.EmployeeID,
   CONCAT(e.FirstName, ' ', e.LastName) AS Employee_Name,
    mr.RequestID,
    it.IssueName,
    pl.LevelName AS Issue_Level,
    -- When the request was made
    CAST(CONCAT(mr.RequestDate, ' ', mr.RequestTime) AS DATETIME) AS RequestedAt,
    -- When the work is scheduled
    mr.MaintenanceScheduled
FROM MaintenanceRequest mr
JOIN Employee e ON mr.EmployeeID = e.EmployeeID
JOIN IssueType it ON mr.IssueTypeID = it.IssueTypeID
JOIN PriorityLevel pl ON mr.PriorityID = pl.PriorityID
WHERE mr.StatusID IN (6001, 6002);

SELECT * FROM EmployeeTaskList
Limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

EmployeeID	Employee_Name	RequestID	IssueName	Issue_Level	RequestedAt	MaintenanceScheduled
5001	Alice Brown	9001	Air Conditioning	High	2025-05-30 09:00:00	2025-05-30 10:00:00
5006	Fiona Blue	9006	Mini Bar Refill	High	2025-05-30 14:00:00	2025-05-30 15:00:00
5011	Kevin Gold	9011	Internet Connection	High	2025-05-30 18:45:00	2025-05-30 19:45:00
5002	Bob White	9002	Plumbing	Medium	2025-05-30 10:30:00	2025-05-30 11:30:00
5012	Laura Cyan	9012	Mini Bar Refill	Medium	2025-05-30 19:20:00	2025-05-30 20:20:00

---

12. Sample Data

Each table is populated with realistic sample data. For instance, the Guest table includes contact methods like phone or email, and the Room table has types like Deluxe or Suite.

---

13. Realistic Use Case

Imagine a guest named John Doe finds the air conditioner not working. He calls the front desk. The request is logged in the **MaintenanceRequest** table with the issue type "Air Conditioning". A technician is assigned. After resolving, a log entry is created in the **Log** table.

---

14. SQL Implementation

The SQL script includes:

- Creating the schema.
- Table creation with foreign key constraints.
- Data insertion.
- Creating useful views.

HotelMaintenanceDB

Limit to 1000 rows

```sql
-- Create Database
CREATE DATABASE IF NOT EXISTS HotelMaintenanceDB;
USE HotelMaintenanceDB;

-- Table Definitions

CREATE TABLE Room (
  RoomID INT PRIMARY KEY,
  RoomType VARCHAR(50),
  Floor INT,
  ViewType VARCHAR(50)
);
```

---

15. Sample Queries to Explore

Here are some example queries that help explore the system:

1. **Which guest has the highest number of unresolved requests?**

```sql
SELECT GuestID, COUNT(*) AS TotalRequests
FROM MaintenanceRequest
WHERE StatusID IN (6001, 6002)
GROUP BY GuestID
ORDER BY TotalRequests DESC;
```

```sql
SELECT GuestID, COUNT(*) AS TotalRequests
FROM MaintenanceRequest
WHERE StatusID IN (6001, 6002)
GROUP BY GuestID
ORDER BY TotalRequests DESC;
```

Results Messages

	GuestID	TotalRequests
1	3001	1
2	3002	1
3	3004	1
4	3005	1
5	3006	1
6	3008	1
7	3009	1
8	3011	1
9	3012	1
10	3014	1
11	3015	1
12	3017	1
13	3018	1
14	3020	1

2. **Which room has had the most maintenance issues in the last month?**

```sql
SELECT RoomID, COUNT(*) AS IssueCount
FROM MaintenanceRequest
WHERE RequestDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY RoomID
ORDER BY IssueCount DESC
Limit 5;
```

Results Messages

	RoomID	IssueCount
1	101	1
2	102	1
3	103	1
4	104	1
5	105	1

3. **Count of Maintenance Requests by Priority Level**

```sql
SELECT
    pl.LevelName AS Priority_Level,
    COUNT(mr.RequestID) AS TotalRequests
FROM MaintenanceRequest mr
JOIN PriorityLevel pl ON mr.PriorityID = pl.PriorityID
GROUP BY pl.LevelName
ORDER BY TotalRequests DESC;
```

Results Messages

	Priority_Level	TotalRequests
1	High	4
2	Medium	4
3	Low	4
4	Critical	4
5	Routine Check	4

These insights help identify service bottlenecks and improve overall hotel operations.

---

16. Performance Benefits

- **Faster turnaround time for resolving guest issues.**
- **Visibility into team performance.**
- **Structured way of handling requests and updates.**

---

17. Future Enhancements

- Add automated notifications.
- Develop a front-end UI for non-technical users.
- Enable mobile access for technicians.

18. Conclusion

This Hotel Maintenance Management System serves as a practical, scalable, and easy-to-use solution for managing guest complaints and room issues. It reflects real industry workflows and enforces data quality throughout.