# Project Title: Digital Maintenance Tracker for Apartments / Buildings

## Objective

Develop a full-stack web application where residents can raise maintenance requests, track technician visits, and view service history using **Angular**, **Node.js (TypeScript)**, and **MySQL**.

---

# Functional Requirements

## 1. User Roles

- **Resident:**
  Can raise maintenance requests, track status, and provide feedback.
- **Technician:**
  Can view assigned requests, update job progress, and add notes.
- **Admin (Optional):**
  Can assign technicians, manage requests, and view analytics (optional).

---

## 2. Maintenance Request Management

Residents can submit maintenance requests for:

- Plumbing, Electrical, Painting, or Other issues
- Upload **optional** photos/videos (store file path or URL)
- View request history and status updates

### UI Suggestions

- **Request Submission Page:** Form to submit new request
- **Request History Page:** View all past requests with status and optional media

---

## 3. Technician Assignment & Tracking

- Admin assigns technicians based on request category.
- Request status flows: **New** → **Assigned** → **In-Progress** → **Resolved**
- Technician can update status, add notes, and upload **optional** media.

### UI Suggestions

- **Admin Dashboard:** Assign technicians, track ongoing work
- **Technician Dashboard:** View assigned requests & update status

---

## 4. Resident Feedback & Rating

- Residents can submit feedback and **optional** comments after a request is resolved.
- Ratings contribute to optional technician performance analytics.

### UI Suggestions

- **Feedback Page:** Submit rating & optional comments

---

# 5. Frontend Requirements (Angular 16+)

- Use **Angular Material** for UI design.
- Maintain modular component structure.

## Suggested Components

- **MaintenanceRequestComponent** → Submit new requests
- **RequestHistoryComponent** → View past requests
- **TechnicianDashboardComponent** → Technician's assigned tasks
- **AdminDashboardComponent** → Assign technicians, view analytics (optional)

## Suggested Routes

| Path | Component | Description |
|---|---|---|
| `/maintenance/new` | MaintenanceRequestComponent | Submit new maintenance request |
| `/maintenance/history` | RequestHistoryComponent | View past requests |
| `/technician/dashboard` | TechnicianDashboardComponent | Technician task management |
| `/admin/dashboard` | AdminDashboardComponent | Technician assignment & analytics (optional) |

---

# 6. Backend Requirements (Node.js + TypeScript + Express)

- Use **MySQL** as the database.
- Implement REST APIs for:
    - Users
    - Maintenance requests
    - Technician assignment
- Include validation and error handling.
- Optional analytics endpoints.

---

# 7. Database Structure (MySQL)

**Maximum 2 tables—simple relation.**

### Users Table

- id
- name
- role (Resident / Technician / Admin (optional))
- contact_info
- created_at

### Requests Table

- id

- resident_id (foreign key → Users)
- technician_id **(nullable, optional)**
- category (Plumbing/Electrical/Painting/Other)
- description
- media **(optional file path or URL)**
- status (New / Assigned / In-Progress / Resolved)
- feedback_rating **(nullable, optional)**
- created_at

---

# 8. Validation Outline

- Category and description must NOT be empty.
- Status must be valid and in proper flow.
- Feedback rating (optional) must be in range (1–5).
- Optional fields (media, feedback_rating, technician_id) must support null values.

---

# 9. Exception Handling

- Handle errors gracefully in frontend & backend.
- UI must display user-friendly messages.
- Server logs errors for debugging.

- Optional global error-handling middleware.

---

# 10. Role-Based Access (Using Angular Guards)

### Resident:

- Submit request
- View request history
- Provide feedback (optional)

### Technician:

- View assigned tasks
- Update job status & notes
- Upload optional media

### Admin (Optional):

- Assign technicians
- View dashboards & analytics
- Manage all maintenance requests

Unauthorized users → Redirect to **Login** or **Not Authorized** page.

Guards ensure:

- Secure access

- Proper workflow
- Separation of roles