IBM NM 5th sem 2025-26 Node js and REST API

Final Hackathon Usecases

Smart Appointment Scheduling System

1. Objective

Develop a REST API–based backend system that allows users to book, reschedule, and cancel appointments digitally.

2. Actors

- User
- Service Provider
- Admin

3. Functional Requirements

- User registration & login (JWT authentication)
- View available appointment slots
- Book appointment
- Reschedule appointment
- Cancel appointment
- Provider can create/manage time slots
- Admin can view all bookings
- Email/notification trigger (mock API allowed)

4. Non-Functional Requirements

- Secure password hashing (bcrypt)
- Token-based authentication
- Proper error handling
- RESTful API structure
- Input validation

5. Suggested Database Tables

- Users
- Providers
- AppointmentSlots
- Appointments

6. Mandatory APIs

- POST /register
- POST /login
- GET /slots
- POST /appointments
- PUT /appointments/:id
- DELETE /appointments/:id

Online Complaint & Grievance Redressal System

1. Objective

Create a backend system where users can raise complaints and track resolution status.

2. Actors

- User
- Admin
- Support Staff

3. Functional Requirements

- User registration/login
- Submit complaint (text + optional file URL)
- Track complaint status
- Admin assigns complaint to staff
- Staff updates resolution status
- Feedback submission after closure

4. Complaint Status Flow

Open → Assigned → In Progress → Resolved → Closed

5. Suggested Database Tables

- Users
- Complaints
- ComplaintUpdates
- Feedback

6. Mandatory APIs

- POST /complaints
- GET /complaints/:id
- PUT /complaints/:id/status
- POST /feedback

Digital Asset Management System

1. Objective

Build a secure backend system for managing digital files with role-based access.

2. Actors

- User
- Admin

3. Functional Requirements

- Authentication & role management
- Upload file (store file path only)
- Download file
- Create folders
- View files by folder
- Version tracking (optional advanced feature)
- Activity logs

4. Suggested Database Tables

- Users
- Roles
- Files
- Folders
- ActivityLogs

5. Mandatory APIs

- POST /files/upload
- GET /files/:id
- DELETE /files/:id
- POST /folders
- GET /folders/:id/files

API-Based Blogging Platform (Headless CMS)

1. Objective

Develop a backend-only blogging system with content management APIs.

2. Actors

- Admin
- Editor
- User

3. Functional Requirements

- Role-based authentication
- Create, edit, delete blog posts
- Draft & publish status
- Add categories & tags
- Comment system
- Like count tracking

4. Suggested Database Tables

- Users
- Posts
- Categories
- Tags
- Comments
- Likes

5. Mandatory APIs

- POST /posts
- PUT /posts/:id
- DELETE /posts/:id
- GET /posts
- POST /comments
- POST /likes

Smart Queue Management System

1. Objective

Develop a digital queue/token management backend system.

2. Actors

- User
- Admin

3. Functional Requirements

- Generate queue token
- View current token number
- View estimated waiting time
- Admin can call next token
- Admin reset queue
- Maintain queue history

4. Suggested Database Tables

- Users
- Tokens
- QueueStatus
- QueueHistory

5. Mandatory APIs

- POST /token
- GET /queue/status
- PUT /queue/next
- POST /queue/reset