Mock HCL hackathon problem statements and guidelines :

Note: The purpose of this document is to help students practice for hackathon by solving and developing a mock project

Tech Expectations

- Angular: Forms, routing, reusable components
- .NET Web API: CRUD for leave requests
- Database: SQL Server / SQLite

1. Smart Leave Management System

Problem Statement

Organizations still manage employee leaves using emails and spreadsheets, leading to conflicts, delays, and lack of visibility.

Design and develop a Leave Management Web Application for small teams.

Core Features (MVP)

- User login (Employee / Manager role)
- Apply for leave (date range + reason)
- Manager can approve/reject
- Leave status tracking
- Dashboard with leave summary

Evaluation Focus

- Clean UI (calendar view preferred)
- API design
- Unit tests for:
    - Leave validation logic
    - Approval flow

2. Complaint & Issue Tracking Portal

Problem Statement

Large campuses and organizations lack a centralized system to track issues raised by users.

Build a Complaint Tracking System where users can raise and track issues.

Core Features

- Register complaint (category, priority)
- Admin dashboard to update status
- Status lifecycle: Open → In Progress → Resolved
- Comment history on complaints

Extra Points

- Search & filter complaints
- Status badges in UI

Testing Scope

- Unit tests for status transitions
- API response validation

3. Smart Appointment Booking System

Problem Statement

Manual appointment booking leads to overlapping schedules and confusion.

Create an Appointment Booking System for service providers.

Core Features

- View available slots
- Book appointment
- Prevent double booking

- Admin can manage slots

Frontend Focus

- Time slot UI
- Confirmation screens

Backend Focus

- Slot availability logic
- Conflict detection

4. Inventory & Asset Management System

Problem Statement

Small businesses struggle to track assets and inventory efficiently.

Develop a Lightweight Inventory Management System.

Core Features

- Add / update inventory items
- Stock in / stock out
- Low stock alerts
- Inventory dashboard

Unit Test Focus

- Stock update logic
- Alert triggering logic

5. Student Feedback & Survey Platform

Problem Statement

Institutions lack real-time feedback collection mechanisms.

Build a Feedback Collection System for students.

Core Features

- Create survey (Admin)

- Submit feedback anonymously
- View analytics (basic charts)

UI Focus

- Clean form UX
- Simple chart visualization

6. Task & Sprint Tracker (Mini Jira)

Problem Statement

Small teams need a simplified task tracking tool.

Create a Mini Project Tracking Tool.

Core Features

- Create tasks
- Task status: To Do / In Progress / Done
- Assign tasks to users
- Kanban-style board (simple)

Evaluation Focus

- Component reusability
- State management
- Status movement logic

7. Expense Tracking & Approval System

Problem Statement

Manual expense approvals are time-consuming and error-prone.

Develop an Expense Management System.

Core Features

- Submit expense
- Upload receipt (optional)

- Manager approval/rejection
- Monthly summary

Testing Focus

- Expense validation rules
- Approval flow tests

Mandatory Hackathon Deliverables / Submission:

Note: All deliverables should be uploaded to a single public github repo , along with guide to the project in the readme file along with deployment links

✅ 1. Design

- Low-fidelity wireframe (Figma or paper)
- Basic UX flow diagram

✅ 2. Prototype

- Angular frontend
- .NET API
- Database schema

✅ 3. Hosting

- Frontend: Netlify / Azure Static Web Apps
- Backend: Azure App Service / Railway

✅ 4. Documentation

- README.md
- API endpoints
- Setup steps
- Assumptions & limitations

✅ 5. Testing

- Unit tests (xUnit / NUnit)
- Code coverage screenshot or report