Name: Tumma venkata sai

Employee ID: PIPL0385
Domain: Frontend
Date and Batch : 8th january - section 2

Hackathon Project Document

Intelligent Enterprise Operations & Decision Platform (IEODP)

1. Project Overview

The Intelligent Enterprise Operations & Decision Platform (IEODP) is a production-grade, role-driven enterprise frontend application designed to manage complex operational workflows, enable AI-assisted decision making, and enforce governance, security, and auditability. The platform is modular, scalable, and built with enterprise engineering principles.

IEODP supports multiple organizational roles (Operations, Leadership, Management, Auditors, Admin) and enables structured workflow processing of business requests (tickets), including review, action, audit, and reverify loops.

---

2. Business Objectives

- Enable structured handling of operational requests
- Provide real-time visibility into workflow status
- Enforce role-based access and responsibilities
- Support auditability and compliance
- Enable reverify loops without data loss
- Provide enterprise dashboards and analytics

---

3. User Roles & Responsibilities

3.1 Operations

- Raise new tickets
- View ticket progress

3.2 Leadership

- Review tickets
- Add strategic comments
- Forward to management

3.3 Management

- Take corrective actions
- Approve / escalate

3.4 Auditors

- Perform final review
- Approve, Reject, or Reverify tickets
- Ensure compliance

3.5 Admin

- Manage users
- Assign roles
- Activate / deactivate users

---

4. Workflow Design

Ticket lifecycle:

SUBMITTED → FORWARDED_TO_MANAGEMENT → ACTION_TAKEN → CLOSED

With reverify loop:

AUDITOR → REVERIFY → OPERATIONS / LEADERSHIP / MANAGEMENT → AUDITOR

The workflow is non-destructive and history-based. All actions are appended to history and never overwritten.

---

5. Data Model

Ticket

```
{
  "id": "string",
  "title": "string",
  "description": "string",
  "priority": "HIGH | MEDIUM | LOW",
  "status": "SUBMITTED | FORWARDED_TO_MANAGEMENT | ACTION_TAKEN |
REVERIFY | CLOSED",
  "createdAt": "ISO Date",
  "history": [
    {
      "role": "OPERATIONS | LEADERSHIP | MANAGEMENT | AUDITORS",
      "action": "string",
      "comment": "string",
      "timestamp": "ISO Date"
    }
  ]
}
```

User

```
{
  "id": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "role": "ADMIN | OPERATIONS | LEADERSHIP | MANAGEMENT | AUDITORS",
  "status": "ACTIVE | INACTIVE"
}
```

---

6. System Architecture

High level flow:

UI Components → Layout → Feature Modules → Redux Toolkit → RTK Query → Backend APIs

The application follows a feature-based folder structure with centralized state and API handling.

---

7. Frontend Architecture Principles

- Feature-based modular design
- Role-driven UI rendering
- Config-driven dashboards
- Centralized API communication via RTK Query
- Strict separation of UI, state, and business logic

---

8. Dashboard System

Each role has a config-driven dashboard. Widgets are loaded dynamically from configuration.

Widget types:

- KPI Widgets
- Pie Charts
- Bar Charts
- Line Charts

Each widget fetches its own data source and renders independently.

---

9. State Management

The application uses Redux Toolkit for global state and RTK Query for all server communication.

Global State:

- Authentication
- User session
- UI state

Server State:

- Tickets
- Users
- Audit logs

Local State:

- Forms
- Filters
- Pagination

---

10. Security & Authorization

- Role-based access control
- Protected routes
- Permission-driven menu rendering
- Component-level access restrictions

Only authorized roles can perform specific actions at each workflow stage.

---

11. Timeline & Auditability

Each ticket displays a full activity timeline showing:

- Role
- Action
- Comment
- Timestamp

This ensures full traceability and audit compliance.

---

12. Performance Engineering

- Lazy loading of dashboards
- React.memo for heavy components
- Memoized selectors
- Config-driven rendering
- Optimized re-renders

Designed to handle large datasets (10k+ rows).

---

13. Error Handling & UX

- Loading states
- Empty states
- API error handling
- User-friendly messages
- Offline detection (planned)

---

14. Accessibility & Responsiveness

- Keyboard navigable UI
- ARIA basics
- Responsive layouts
- Consistent experience across devices

---

15. Future Enhancements

- File attachments
- SLA tracking
- Auto-escalation rules
- AI-assisted insights
- Notification system
- Advanced audit reporting

---

16. Conclusion

IEODP is designed as a real enterprise system, not a demo UI. It supports complex workflows, multi-role collaboration, auditability, and scalable architecture. The system is built to simulate real-world enterprise operational platforms.