Detailed System Structure and Plan for GIMPA Digital Dashboard

1. System Overview

The digital dashboard for GIMPA is designed to monitor KPI delivery across the institution. It integrates role-based access to align with the hierarchical structure of Deputy Rector (Super Admin), Deans, Heads of Departments (HODs), and Lecturers. The system also enables data submission, performance monitoring, and feedback mechanisms for KPI tracking.

---

2. Hierarchical Role-Based Structure

1. **Deputy Rector (Super Admin)**:
   - Full access to all modules and data.
   - Oversees system-wide KPI monitoring.
   - Provides feedback to Deans and reviews overall institutional performance.
2. **Deans**:
   - Access to departmental KPIs under their school.
   - Review submissions by HODs.
   - Provide feedback to HODs.
   - Submit aggregated school-level data to the Deputy Rector.
3. **Heads of Departments (HODs)**:
   - Manage department-level KPI data.
   - Review submissions from lecturers.
   - Provide feedback to lecturers.
   - Submit departmental reports to Deans.
4. **Lecturers**:
   - Submit evidence and updates for individual KPIs they are responsible for.
   - Respond to feedback from HODs.

---

3. Functional Requirements

Frontend Features (React.js)

- **Login System**:
  - Role-based authentication (Deputy Rector, Dean, HOD, Lecturer).
  - Secure login using JWT.
- **Dashboard**:
  - Role-specific views:
    - Deputy Rector: Overview of all schools, departments, and lecturers.
    - Deans: Overview of departments within their school.
    - HODs: Overview of KPIs and lecturers within their department.
    - Lecturers: KPI submission and feedback response interface.

    - Graphs and charts to visualize KPI progress.
- **KPI Management**:
    - Submission forms for evidence uploads (files, reports, images).
    - Status indicators for KPIs (e.g., Pending, Reviewed, Approved).
    - Performance visualization with progress bars and charts.
- **Feedback Mechanism**:
    - Allow feedback at each hierarchical level.
    - Track feedback history for transparency.

Backend Features (Node.js, Express)

- **Authentication & Authorization**:
    - Implement JWT for secure token-based authentication.
    - Middleware for role-based access control.
- **Database (MongoDB)**:
    - **Users Collection**:
        - Fields: `id`, `name`, `email`, `password`, `role` (e.g., Deputy Rector, Dean, HOD, Lecturer), `school`, `department`.
    - **KPI Collection**:
        - Fields: `id`, `indicator`, `target`, `evidence`, `status` (e.g., Pending, Approved), `feedback`, `submittedBy`, `reviewedBy`.
    - **Feedback Collection**:
        - Fields: `id`, `kpiId`, `feedbackText`, `role`, `providedBy`, `timestamp`.
- **API Endpoints**:
    - **Auth Routes**:
        - POST `/login`: Authenticate user.
        - POST `/register`: Add new user (Super Admin only).
    - **KPI Routes**:
        - GET `/kpis`: Fetch KPIs based on user role.
        - POST `/kpis`: Submit/update KPI data.
        - PATCH `/kpis/:id/feedback`: Add feedback for a KPI.
    - **Feedback Routes**:
        - GET `/feedback/:kpiId`: Fetch feedback for a KPI.

---

4. System Workflow

Submission Process

1. **Lecturers**:
    - Submit KPI data through a form.
    - Attach evidence (e.g., reports, images).
    - Mark KPI as "Submitted."
2. **HODs**:
    - Review submitted KPIs from lecturers.

    - Provide feedback or mark them as "Approved."
    - Submit departmental reports to Deans.
3. **Deans**:
    - Review departmental submissions.
    - Provide feedback or escalate issues.
    - Submit school-level reports to the Deputy Rector.
4. **Deputy Rector**:
    - Review aggregated reports.
    - Provide system-wide feedback.
    - Monitor overall performance.

Feedback Process

1. Superiors (e.g., Deans, HODs) provide feedback for KPIs at their level.
2. Feedback is logged in the database and visible to the relevant users.
3. Feedback history can be reviewed to ensure accountability.

---

5. Proposed Database Schema

Users Collection

```
{
  "id": "unique_user_id",
  "name": "John Doe",
  "email": "john.doe@gimpa.edu",
  "password": "hashed_password",
  "role": "Lecturer", // or "HOD", "Dean", "Deputy Rector"
  "school": "School of Business",
  "department": "Accounting"
}
```

KPI Collection

```
{
  "id": "unique_kpi_id",
  "indicator": "Conduct capacity building workshops",
  "target": "≥ 2 workshops",
  "evidence": ["link_to_file1", "link_to_file2"],
  "status": "Pending", // "Submitted", "Approved"
  "submittedBy": "lecturer_id",
  "reviewedBy": "hod_id"
}
```

Feedback Collection

```
{
  "id": "unique_feedback_id",
  "kpiId": "linked_kpi_id",
  "feedbackText": "Good progress but improve documentation.",
  "role": "HOD", // Role of the person providing feedback
  "providedBy": "hod_id",
  "timestamp": "2025-01-16T10:00:00Z"
}
```

---

6. User Roles and Access Permissions

Role	Permissions
Deputy Rector	View, review, and provide feedback at all levels; manage users and reports.
Dean	View department-level KPIs; provide feedback to HODs; submit school reports.
HOD	View and review lecturer submissions; provide feedback to lecturers.
Lecturer	Submit and update KPI data; respond to feedback.

---

7. Implementation Phases

Phase 1: Core Infrastructure

- Backend API with user authentication and basic CRUD for KPIs.
- Basic React frontend for data submission and visualization.

Phase 2: Role-Based Features

- Implement role-specific dashboards and permissions.
- Add feedback and response mechanisms.

Phase 3: Advanced Features

- KPI performance analytics (graphs and charts).
- File upload and storage integration.
- Comprehensive testing and debugging.

Phase 4: Deployment and Training

- Deploy system on a cloud platform (e.g., AWS, Heroku).
- Conduct training sessions for GIMPA staff.

---

8. Tools and Technologies

- **Backend**: Node.js, Express.js, JWT, MongoDB.
- **Frontend**: React.js, Redux (if state management needed).
- **Deployment**: Docker, Kubernetes (optional), AWS/Heroku..