Learning Analytics Dashboard – Frontend Assignment

Overview

Build a **single-page web dashboard** for a training academy that looks and behaves like the attached mockups (dark and light themes) The UI should match the look and design of the attached images.

Light theme

Dark Theme

Use the Data files found in the zip file.

The dashboard should:

1. Display key KPIs and charts for learner performance.

2. Load data from a **mock API** that reads from local JSON files.

3. Allow the user to **switch between Light and Dark themes**, and remember the choice.

4. (Optional, nice to have) Allow switching between **2024 and 2025** datasets.

You may use any modern frontend stack. **React + TypeScript** with a charting library (Recharts / Chart.js / ECharts, etc.) is preferred.

Core Requirements

1. Filters / Controls (Top bar)

Implement a top toolbar with:

- **Period selector**
- Toggle: `Monthly` / `Quarterly` (UI only – data can remain static but the control must be wired).
- **Date range display**
- Read-only text: `01 Jan, 2024 - 31 Dec 2024`.

- **District filter dropdown**
- Options: `All District`, `Ariyulur`, `Chennai`, `Coimbatore`, `Cuddalore`, `Dharmapuri`, `Dindigul`, `Erode`, `Kallakurichi`, `Karur`, `Madurai`.
- When a district is selected, either:
- Highlight that district in charts, **or**
- Filter charts to show only that district (your choice, but the effect must be visible).

> **Optional (nice)**

> - Year selector: `2024` / `2025`, switching between `dashboard_2024.json` and `dashboard_2025.json`.

---

2. Summary KPI Cards

Display the following metrics (from the JSON):

- Total Learners Enrolled
- Male
- Female
- Others
- Active Learners
- Engaged Learners

Each KPI should be a card with:

- Icon (can be simple emoji or SVG)
- Label
- Value

Layout should roughly match the mockups.

---

3. Course Progress Rate (Bar Chart)

- Grouped bar chart by **district**.
- Series:
- `Below`
- `Average`
- `Good`

- Values are percentages (0–100).
- Use the `courseProgress` array from the JSON.

---

4. Pass Percentage (Horizontal Bar Chart)

Horizontal bars for:

- `Overall Learners`
- `Assessment taken`
- `Passed`
- `Failed`

Use numeric values from `passStats`.

---

5. Average Assessment Score (Donut Chart)

Donut chart showing:

- `% Assessment completed`
- `% Assessment not completed`

Use `assessmentCompletion.completedPercent` and `notCompletedPercent`.

---

6. Learners Details Breakdown (Pie Chart)

Pie chart for grade distribution:

- A – Grade (>80)
- B – Grade (>60)
- C – Grade (>50)
- D – Grade (>30)
- E – Grade (0)

Labels and percentages come from `gradeBreakdown`.

---

7. District Ranking (Bottom Section)

Implement a **District Ranking** view using one of:

Option A – Rank cards (simpler)

Cards showing: District name + Rank (1st, 2nd, ...).

Option B – Bar/Combo chart (preferred)

Bar chart per district with multiple series, e.g.:

- `male`
- `female`
- `others`
- `passed`
- `assessmentCompleted`

Also include a `Rank by` dropdown with options:

- `Enrollment`
- `Pass %`

The selected option should change the **sorting** of districts.

Data comes from `districtRanking.districts`.

Theme Requirements

Light & Dark Mode

- Implement a **theme toggle** (switch or button) that changes the whole dashboard between:
- **Light theme** – similar to the light mock (light background, dark text).
- **Dark theme** – similar to the dark mock (dark background, light text).
- The theme should affect:
- Page background
- Cards & panels
- Text colors
- Charts (axes, labels, tooltips)
- Inputs (dropdowns, buttons)

Persistence

- Store the selected theme in `localStorage`.
- On page load, initialize the theme from `localStorage` (fall back to system preference or light).

Data & Mock API

Files

This repository includes two mock data files:

- `data/dashboard_2024.json` – base metrics.
- `data/dashboard_2025.json` – **~25% improved** metrics vs 2024 (higher enrollments, completions, pass rates, more A grades, etc.).

Expected API shape

Implement a mock API that returns one of these JSON objects. You can choose any of:

- A small Node/Express server with routes like:
- `GET /api/dashboard?year=2024`
- `GET /api/dashboard?year=2025`
- A frontend-only mock using `fetch('/data/dashboard_2024.json')` or similar.

Your frontend should assume this response schema:

```
type DashboardResponse = {

 summary: {

   totalLearners: number;

   male: number;

   female: number;

   others: number;

   activeLearners: number;

   engagedLearners: number;

 };

 courseProgress: {

   district: string;

   below: number;

   average: number;

   good: number;

 }[];

 passStats: {
   overallLearners: number;
   assessmentTaken: number;
   passed: number;
   failed: number;
 };
 assessmentCompletion: {
   completedPercent: number;
   notCompletedPercent: number;
 };
 gradeBreakdown: {
   grade: string;
   label: string;
   percent: number;
 }[];
 districtRanking: {
   rankBy: string;
   districts: {
     district: string;
     rank: number;
     enrolled: number;
     male: number;
     female: number;
     others: number;
     passed: number;
     assessmentCompleted: number;
     completionRatePercent: number;
```

```
   }[];
  };
};
```

Tech & Implementation Guidelines

- **Framework:** React (preferred) or any equivalent SPA framework.
- **Language:** TypeScript preferred, JavaScript acceptable.
- **Charts:** Recharts, Chart.js, ECharts, or similar.
- **Styling:**
- You may use CSS Modules, Tailwind, Styled Components, or standard CSS.
- Ensure themes are implemented cleanly (e.g., CSS variables, context provider, or theme library).

Structure (suggested):

- `src/components/SummaryCards.tsx`
- `src/components/CourseProgressChart.tsx`
- `src/components/PassStatsChart.tsx`
- `src/components/AssessmentDonut.tsx`
- `src/components/GradeBreakdownPie.tsx`
- `src/components/DistrictRanking.tsx`
- `src/theme/ThemeProvider.tsx`
- `src/api/dashboard.ts`

How to Run

When you implement the project:

install dependencies

npm install

start dev server

npm run dev

or:

npm start

Describe any framework-specific commands you use in your own README updates.

Evaluation Criteria

We will look at:

1. **Correctness & Completeness**

- All sections of the dashboard implemented.
- Theme toggle works and persists.
- Data is loaded from JSON (not hard-coded in components).

2. **Code Quality**

- Clear structure and naming.
- Reusable components.
- Reasonable separation of concerns.

3. **UI/UX**

- Visual fidelity to the provided mockups (doesn’t need to be pixel-perfect).
- Responsiveness on typical laptop/tablet sizes.

4. **Documentation**

- Clear README with setup instructions & assumptions.
- Comments where needed.

Bonus (Optional)

- Unit tests for key components.
- Animations on chart load or theme change.
- Year selector switching between 2024 and 2025 datasets.
- Lightweight global state management (e.g., Zustand/Context) for filters.