Custom Form Builder with Live Analytics

Overview:

The goal of this project is to build a dynamic, customizable form builder application that allows users to create forms, collect responses, and view live analytics about the responses in real time. The project is designed to test your full-stack development skills, including your ability to:

- Build a dynamic frontend using Next.js and TailwindCSS
- Implement a Go Fiber API to handle form submissions and data processing
- Use MongoDB to store flexible form data and responses
- Build a real-time analytics dashboard to visualize user responses

Features:

- **Form Builder:**
  - Create forms with text, multiple choice, checkboxes, and rating fields.
  - Support drag-and-drop reordering, field validation, and saving drafts.
- **Feedback Form:**
  - Generate a unique shareable link per form.
  - Users fill and submit responses linked to that form.
- **Analytics Dashboard:**
  - Real-time updates (Socket.IO or polling).
  - Show trends (e.g. answer distribution, average ratings). Include visual breakdowns per field.
- **Backend (Go Fiber):**
  - APIs to create/update forms, submit responses, and fetch analytics.
  - Validate submitted data.
- **Database (MongoDB):**
  - Store flexible form schemas, responses, and metadata.

Key Requirements

- **Custom Form Logic**: Create your own form logic without relying on third-party libraries like Formik or React Hook Form. This tests your ability to manage form state, validation, and field updates.
- **State Management**: Use React hooks for managing form state and backend interactions. Implement an efficient state management strategy to handle dynamic form inputs.
- **Custom Dashboard**: The real-time analytics dashboard should be built from scratch. Focus on creating **dynamic visualizations** that update live as new feedback is received.
- **Real-Time Updates**: You must implement **live data updates** (either via WebSocket's or long-polling). No basic reloading: data should appear on the dashboard as it's submitted.

Additional Features (Optional, for extra credit):

1. **Export Responses**: Allow users to download form responses as a CSV file or a PDF report.
2. **Conditional Fields**: Add logic to show/hide specific fields based on previous answers (e.g., "If question 3 = 'Yes', show question 4").
3. **User Authentication**: Implement basic **JWT authentication** for users to save their form creations and responses. This ensures a personalized experience (e.g., users can view only their forms).
4. **Survey Trends**: Implement trend analysis on form responses. For example, show insights like the average rating for a form, most common responses, or most frequently skipped questions.
5. **Dark Mode**: Implement a **dark mode toggle** to give the app a more modern feel and improve usability.
6. **Unit Tests**: Have unit tests for frontend and backend aspect of the project.

Instructions to Submit:

1. **Git Repository**: Push your code to a GitHub repository. Make sure your commits are meaningful and show progression over time.
2. **Demo**: Provide a link to a live demo. Ensure the demo is functional and accessible for review.
3. **Documentation**: Include a **README.md** with the following:
    - A brief description of your project
    - How to set up the project locally
    - Any assumptions you made or challenges you faced during the development
    - How to test the real-time analytics feature (if applicable)