Frontend Intern Take-Home Assignment: AI Safety Incident Dashboard

Goal: To assess your fundamental frontend development skills, including UI creation, user interaction, state management, and handling user input, by building a simple interactive interface relevant to HumanChain's mission in AI safety.

Context: HumanChain is a deep-tech software AI startup at the forefront of AI safety, aiming to build a safer, more trustworthy, and human-centric digital world. This assignment involves creating a frontend interface to view and log hypothetical AI safety incidents.

Instructions:

1. **Choose ONE Platform:** Select the platform you are most comfortable with:
    - **Web:** Use HTML, CSS, and TypeScript. You may use any JavaScript framework (e.g., React, Vue, Angular, Svelte, Next.js) or vanilla TypeScript.
    - **Android:** Use Kotlin and standard Android UI components.
    - **iOS:** Use Swift and standard iOS UI components (UIKit or SwiftUI).
2. **Complete the Task:** Implement the specific task outlined below for your chosen platform.
3. **Focus:** Prioritize clean, readable, well-structured code and a functional user interface. Implement the requested features, including data display, filtering/sorting, state management, user input handling, and navigation (where applicable). Apply clean styling; complex visual design is not the primary focus, but usability and layout are important.
4. **Data:** Use the provided mock data structure. All data should be handled locally within the frontend application state (in memory). Any data added or edited by the user only needs to persist for the current session. **No database connections, API calls, or other backend integration are required for this assignment.**
5. **Submission:**
    - Package your code in a zip file or provide a link to a public Git repository (e.g., GitHub, GitLab).
    - Include a README.md file with:
        - Clear instructions on how to build/install dependencies and run your project locally.
        - The language/framework choice (especially for Web).
        - (Optional) Briefly mention any design decisions or challenges.

Platform-Specific Tasks: AI Safety Incident Interface

A) Web (HTML/CSS/TypeScript + Optional Framework)

- **Task:** Create an interactive "AI Safety Incident Dashboard" component.
- **Features:**
  - Display a list of mock AI safety incidents, showing Title, Severity, and Reported Date for each.
  - Implement filtering controls (e.g., buttons or dropdown) to filter the list by Severity ("All", "Low", "Medium", "High").
  - Add sorting controls (e.g., buttons or dropdown) to sort the list by Reported Date (Newest First, Oldest First).
  - Add a "View Details" button/link for each incident. Clicking it should toggle the visibility of the full Description below the entry. Manage this expanded/collapsed state.
  - Include an "Report New Incident" form (can be always visible or toggled). The form should have inputs for Title, Description, and Severity (use dropdown or radio buttons for severity levels).
  - Submitting the form should add the new incident to the displayed list (updating the component's state). Basic input validation (e.g., non-empty fields) is a plus.
  - Ensure responsive layout (Flexbox/Grid preferred) and basic, clean styling with hover effects.
- **Mock Data Structure Example:**
  [
  { "id": 1, "title": "Biased Recommendation Algorithm", "description": "Algorithm consistently favored certain demographics...", "severity": "Medium", "reported_at": "2025-03-15T10:00:00Z" },
  { "id": 2, "title": "LLM Hallucination in Critical Info", "description": "LLM provided incorrect safety procedure information...", "severity": "High", "reported_at": "2025-04-01T14:30:00Z" },
  { "id": 3, "title": "Minor Data Leak via Chatbot", "description": "Chatbot inadvertently exposed non-sensitive user metadata...", "severity": "Low", "reported_at": "2025-03-20T09:15:00Z" }
  ]

B) Android (Kotlin)

- **Task:** Create an "AI Safety Incident Tracker" app feature.
- **Features:**

    - Display a scrollable list (RecyclerView with CardView) of mock incidents, showing Title, Severity, and Reported Date.
    - Implement filtering controls (e.g., Tabs, Spinner, Chips) to filter the list by Severity.
    - Navigate to a separate Detail Screen when an incident item is tapped, displaying all fields (Title, Severity, Date, Description). Include back navigation.
    - Add an "Report Incident" button (e.g., FAB) on the main list screen.
    - Tapping this button navigates to a new "Report Incident" screen.
    - This screen must contain input fields (EditText, etc.) for Title and Description, and a way to select Severity (Spinner, RadioGroup). The Reported Date should be set automatically (use current time).
    - A "Save" button adds the new incident to the local list data, navigates back, and the new incident should appear in the list. Basic input validation is expected.
    - Use standard Android components, layouts, and navigation patterns. Apply clean styling.
- **Mock Data Structure Example (Kotlin data class):**
  data class Incident(val id: Int, val title: String, val description: String, val severity: String, val reported_at: String) // Use String for date simplicity or Date/Timestamp
  // Use a MutableList for the data source
  val incidents = mutableListOf(
    Incident(1, "Biased Recommendation Algorithm", "Algorithm consistently favored...", "Medium", "2025-03-15T10:00:00Z"),
    // ... more incidents
)

C) iOS (Swift)

- **Task:** Create an "AI Safety Incident Log" app feature.
- **Features:**
    - Display a list (List / UITableView) of mock incidents, showing Title, Severity, and Reported Date.
    - Implement filtering controls (e.g., Picker, Segmented Control) to filter the list by Severity.
    - Navigate to a Detail View when an incident item is tapped, displaying all fields.
    - Add an "Report Incident" button (+) accessible from the main list view.
    - Tapping this button presents a new "Report Incident" view/screen.
    - This screen must have input fields (TextField, TextEditor) for Title and Description, and a way to select Severity (Picker, SegmentedControl). Set

Reported Date automatically (current time).
    - A "Save" button adds the new incident to the local data collection, dismisses the add view, and the new incident should appear in the list. Basic input validation is expected.
    - Use standard iOS components, layouts (Auto Layout / SwiftUI), and navigation. Apply clean styling.
- **Mock Data Structure Example (Swift struct):**

```
struct Incident: Identifiable {
    let id: Int
    var title: String
    var description: String
    var severity: String
    var reported_at: Date // Or String for simplicity
}
// Use a @State var array for the data source in SwiftUI, or a simple array managed by a view controller in UIKit
@State var incidents = [
    Incident(id: 1, title: "Biased Recommendation Algorithm", description: "Algorithm consistently favored...", severity: "Medium", reported_at: Date()), // Use appropriate date handling
    // ... more incidents
]
// Manage unique IDs when adding new incidents
```

Good luck! We look forward to seeing your work.