

# Full■Stack Screening Assignment

## Incident Tracker Mini App

### Overview

Build a small full■stack web application that allows engineers to create, browse, and manage production incidents. The goal of this assignment is to evaluate frontend engineering, backend API design, database querying, pagination logic, and overall engineering decision■making.

**Estimated Time:** 6–10 hours (guideline only). Focus on clarity, correctness, and structure.

### Functional Requirements

- 1 Create incidents with validation
- 2 Fetch incidents from a database
- 3 Display incidents in a paginated table (server■side pagination required)
- 4 Search, filter, and sort incidents server■side
- 5 View incident details and update status
- 6 Seed database with ~200 records

### Incident Data Model

id (uuid/int), title, service, severity(SEV1■SEV4), status(OPEN/MITIGATED/RESOLVED), owner(optional), summary(optional), createdAt, updatedAt

### Backend Expectations

Implement REST APIs for:

POST /api/incidents  
GET /api/incidents (pagination, filtering, sorting)  
GET /api/incidents/:id  
PATCH /api/incidents/:id

Use proper validation, indexing where appropriate, and parameterized queries/ORM safety.

### Frontend Expectations

Build a responsive UI including:

- Paginated table with loading states
  - Column sorting
  - Filters and debounced search
  - Detail page editing
  - Create incident workflow
- Focus on component organization and clean UX states.

# Reference Wireframes

The following mock wireframes illustrate expected UI layout direction:

The wireframes illustrate the user interface for an Incident Tracker application, showing three main components: Incident List, Incident Detail, and Create Incident.

**Incident List**

This screen shows a list of incidents. It includes filters for Service (dropdown), Severity (checkboxes for SEV1, SEV2, SEV3, SEV4), and Status (dropdown). A search bar and a "Filter" button are also present. The table lists incidents with columns: Title, Severity, Status, Created At, and Owner. Examples include "Login Failure" (Auth, Open, 04/15/2024, jason@team), "Payment Delay" (Payments, Mid2v1, 04/14/2024, amy@team), "API Timeout" (Backend, Resolved, 04/13/2024, dev@team), "UI Bug on Dashboard" (Frontend, Open, 04/12/2024, ...), and "Database Issue" (Database, Open, 04/11/2024, ops@team). A pagination bar at the bottom shows "Page 1 of 10" with links 1, 2, 3, 10, and Next >.

**Incident Detail**

This screen shows details for a specific incident titled "API Timeout". It includes fields for Service (Backend), Severity (SEV1), Status (Resolved), Assigned To (dev@team), and Occurred At (April 13, 2024). A summary section notes: "API requests to the backend service were timing out, causing disruptions for users." Buttons for "Save Changes" and "Cancel" are at the bottom.

**Create Incident**

This screen allows creating a new incident. It includes fields for Title (Issue Title...), Service (Select Service dropdown), Severity (radio buttons for SEV1, SEV2, SEV3, SEV4, where SEV1 is selected), Status (Select Status dropdown), Assigned To (Optional, dev@team), and Summary (Describe the incident...). Buttons for "Create Incident" and "Cancel" are at the bottom.

**Submission Instructions**

Please submit your solution as a public GitHub repository including:

/backend — backend source code

/frontend — frontend source code

README.md must include:

- Setup/run instructions
- API overview
- Design decisions & tradeoffs
- Improvements you would make with more time

Share the repository link with the hiring team upon completion.