Intern Technical Assignment

Invoice Management System

Company: AINextBill Technology Private Limited

Thank you for your interest in the Intern position at **AINextBill Technology Private Limited**.

As part of our hiring process, shortlisted candidates are required to complete the following technical assignment. This exercise helps us evaluate **technical fundamentals, implementation clarity, and problem-solving ability**.

Please read all instructions carefully before starting.

Objective

Build a **basic Invoice Management System** with authentication, dashboard, and full CRUD (Create, Read, Update, Delete) functionality.

This assignment is designed at an **intern level**. We do not expect production-ready software, but we do expect **clean structure, correct fundamentals, and clear reasoning**.

Authentication

- Implement a **simple login mechanism**
- Acceptable approaches:
    - Email and password authentication
    - OR hardcoded credentials (must be clearly mentioned in README)
- Only authenticated users should be able to access the dashboard and invoice pages

Dashboard

After successful login, display a dashboard containing:

- Total number of invoices
- Total invoice amount
- Invoice table (list of invoices)

Invoice Data Model

Each invoice must contain the following fields:

- Invoice Number
- Customer / Vendor Name
- Invoice Amount
- Invoice Date
- Status (Paid / Unpaid)

All data must be stored in a **SQL database**.

CRUD Functionality (Mandatory)

Users must be able to:

- Create a new invoice
- View the list of invoices
- Update an existing invoice
- Delete an invoice

Filter and Search

The invoice table must support:

- Filter by **Status** (Paid / Unpaid)
- Filter by **Date range**
- Search by **Invoice Number or Customer/Vendor Name**

Frontend Requirements

- Framework: **React**
- Must use:
    - React Hooks
    - React Router
    - API integration using Axios or Fetch
    - Basic form validation

Required pages:

- Login
- Dashboard
- Invoice Create / Edit
- Invoice List with filters

UI design quality is not part of evaluation. A simple and functional UI is sufficient.

Backend & Database Requirements

- Runtime: **Node.js**
- ORM: **Prisma (Mandatory)**
- Database: **SQL**
  (SQLite / MySQL / PostgreSQL — candidate's choice)

Backend Responsibilities:

- Authentication APIs
- Invoice CRUD APIs
- Database access using **Prisma Client only**
- Basic request validation
- Clear and readable project structure

Prisma must be used for:

- Schema definition
- Migrations
- Database queries

Reasoning & Understanding (Mandatory)

Along with the code, candidates must submit the following documents:

A. README.md

Must include:

- Project setup instructions
- Technology stack used
- Assumptions or simplifications made
- One known limitation or issue in the implementation

B. THINKING.md

Create a file named **THINKING.md** and answer the following:

1. Describe one design decision you made and why.
2. Mention one alternative approach you considered but did not choose.
3. If a new field tax_amount is added and the total amount becomes invoice_amount + tax_amount, explain the required changes in:
    - Database (Prisma schema & migration)
    - Backend APIs
    - Frontend UI & logic

Known Limitation / Bug Declaration

Please clearly mention **one known limitation or bug** in your project and explain:

- What the issue is
- Why it occurs
- How you would fix it if given more time

This will **not negatively impact evaluation**.

Time Expectation

- Recommended time: **1-2 Day**
- Candidates are advised **not to over-engineer** the solution

Submission Instructions

Please submit:

1. **GitHub repository link**
2. **README.md**
3. **THINKING.md**

Submissions missing any of the above may not be evaluated.

13. Evaluation & Next Steps

Shortlisted candidates will be invited for a **technical interview**, where discussions will be based on the submitted assignment and the candidate's own implementation.
