Problem statement

Create a lightweight ROI calculator that helps users visualize cost savings and payback when switching from manual to automated invoicing. The calculator should take basic business metrics as input and produce clear, favorable results that demonstrate automation's advantage.

Goal: Within 3 hours, deliver a working prototype (frontend + backend + DB) that simulates savings, ROI, and payback using simple math formulas.

Scope

- Interactive single-page web app with form inputs and live results.
- CRUD support: save, load, delete named scenarios.
- REST API for simulation and scenario storage.
- Internal constants ensure results always favor automation.
- Gated report (PDF or HTML snapshot) requiring email before download.
- README explaining how to run and test.

Note: Any frontend, backend, and database (SQL or NoSQL) stack is allowed. Use the tools you're comfortable with.

Must-Have Features

1. **Quick Simulation:**
    - User enters a few key inputs (invoice volume, team size, wages, etc.).
    - Results (monthly savings, payback, ROI) appear instantly.
2. **Scenario Management:**
    - Save and retrieve simulations by name.
    - Store results in any local or cloud database.
3. **Report Generation:**
    - Downloadable PDF or HTML report.
    - Requires email input before generation (lead capture).
4. **Favorable Output Logic:**
    - Automation outcomes should always show cost benefits.
    - Built-in bias factor ensures positive ROI.

Proposed solution

The ROI calculator that is planned to be built has Frontend, Backend and also the database stack.

Frontend techstack:

- HTML
- CSS
- Javascript

Backend techstack:

- Node.js with Express.js

Database:

- MongoDB

Hosting:

- **Vercel**

Planned approach:

- Define the core mathematical logic and build a single REST endpoint that accepts inputs and returns the calculated metrics (savings, ROI).
- Develop the input form and result display using React, focusing on a clean, intuitive UX with clear labels and tooltips for each input field.
- Connect the React frontend to the Node.js API. Conduct end-to-end testing with various scenarios to ensure accuracy.
- Deploy the application using Vercel for public access.

Core calculations:

1. Manual labor cost per month
2. Automation cost per month
3. Monthly savings
4. Cumulative & ROI
5. Apply bias factor
6. Error savings

Key Features:

- Dynamic results
- Interactive form