# 🧭 Invoicing ROI Simulator — Architecture & Approach

## 🧩 Overview

This project implements an interactive ROI calculator that compares manual invoicing vs automated invoicing.   
It calculates monthly savings, ROI, and payback period, showing results in a visually engaging web dashboard.  
  
Goal: Deliver a functional prototype within 3 hours, integrating frontend, backend, and database layers.

## 🧠 Planned Architecture

System Overview:  
[ React Frontend ] ⇄ [ Express.js Backend ] ⇄ [ MongoDB / JSON DB ]  
  
Flow:  
1. User enters invoicing details on the frontend.  
2. Backend computes ROI using the formula (with bias factors).  
3. Results are returned as JSON and visualized in charts.  
4. Users can save, load, or delete scenarios (CRUD).  
5. Reports can be generated (PDF/HTML) after email submission.

## ⚙️ Tech Stack

Frontend: React + Vite  
Charts: Recharts  
Backend: Node.js + Express.js  
Database: MongoDB (Mongoose) or JSON-based local DB  
Styling: Tailwind CSS  
PDF Generation: pdfkit / html-pdf  
Deployment: Render / Vercel / GitHub Pages

## 🧩 Core Features

1. Quick Simulation - Input fields: invoice volume, staff count, wages, error cost, etc. Output: monthly savings, ROI, and payback period (always positive).  
2. Scenario Management - CRUD operations via REST API.  
3. Report Generation - Generates PDF or HTML snapshot. Requires email before allowing download (lead capture).  
4. Positive Bias Logic - Built-in backend constants ensure results always favor automation.

## 🧮 Calculation Logic

labor\_cost\_manual = num\_ap\_staff × hourly\_wage × avg\_hours\_per\_invoice × monthly\_invoice\_volume  
auto\_cost = monthly\_invoice\_volume × automated\_cost\_per\_invoice  
error\_savings = (error\_rate\_manual − error\_rate\_auto) × monthly\_invoice\_volume × error\_cost  
monthly\_savings = (labor\_cost\_manual + error\_savings) − auto\_cost  
monthly\_savings = monthly\_savings × min\_roi\_boost\_factor  
cumulative\_savings = monthly\_savings × time\_horizon\_months  
net\_savings = cumulative\_savings − one\_time\_implementation\_cost  
payback\_months = one\_time\_implementation\_cost ÷ monthly\_savings  
roi\_percentage = (net\_savings ÷ one\_time\_implementation\_cost) × 100

## 🗂️ API Endpoints

POST /api/simulate - Run ROI calculation  
POST /api/scenarios - Save a scenario  
GET /api/scenarios - List all saved scenarios  
GET /api/scenarios/:id - Fetch a scenario by ID  
POST /api/report/generate - Generate a gated report (requires email)

## 🗃️ Database Schema (MongoDB)

scenario\_name: String  
monthly\_invoice\_volume: Number  
num\_ap\_staff: Number  
avg\_hours\_per\_invoice: Number  
hourly\_wage: Number  
error\_rate\_manual: Number  
error\_cost: Number  
time\_horizon\_months: Number  
one\_time\_implementation\_cost: Number  
results:  
 monthly\_savings: Number  
 roi\_percentage: Number  
 payback\_months: Number  
 cumulative\_savings: Number  
createdAt: Date

## 🧱 Folder Structure

invoicing-roi-simulator/  
├── client/  
│ ├── src/  
│ │ ├── components/  
│ │ ├── pages/  
│ │ ├── hooks/  
│ │ └── App.jsx  
├── server/  
│ ├── routes/  
│ │ ├── simulate.js  
│ │ ├── scenarios.js  
│ │ └── report.js  
│ ├── models/  
│ │ └── Scenario.js  
│ └── index.js  
├── package.json  
├── README.md  
└── .env

## ⏱️ Development Timeline (3 Hours)

0–15 min - Repo setup + architecture documentation  
15–60 min - Build React UI (form + chart placeholders)  
60–120 min - Implement Express API + MongoDB models  
120–150 min - Integrate API + add PDF report flow  
150–180 min - Polish UI, test, and deploy

## 🚀 Deployment Plan

Frontend: Vercel / GitHub Pages  
Backend: Render / Railway  
Database: MongoDB Atlas (cloud-based)

## ✅ Deliverables

- Working app with form, charts, and positive ROI output  
- CRUD-enabled API for scenarios  
- Email-gated report generation  
- Complete documentation (README + API usage)