Invoicing ROI Simulator — Technical Documentation

# Planned Approach & Architecture

Our approach is to build a lightweight, full-stack ROI simulator as a single-page web application. The solution will consist of three core layers:

1. Frontend (React SPA):

- Provides an interactive form for users to input invoicing metrics.

- Displays live simulation results (monthly savings, payback, ROI).

- Supports scenario save/load/delete operations.

- Handles gated report download (email required).

2. Backend (Node.js + Express):

- Implements REST APIs for simulation, scenario persistence, and report generation.

- Encapsulates business logic, including bias-favoring formulas and internal constants (kept server-side).

- Generates downloadable PDF/HTML reports.

- Validates inputs and applies favorable bias factor before returning results.

3. Database (MongoDB):

- Stores saved scenarios with user-provided names.

- Supports CRUD operations for scenario management.

- Schema example:

{

"scenario\_name": "Q4\_Pilot",

"inputs": {...},

"results": {...},

"createdAt": "2025-10-07T10:00:00Z"

}

4. Hosting & Deployment:

- Frontend: Vercel

- Backend: Render

- Database: MongoDB Atlas (cloud-hosted)

# System Architecture Diagram

React Frontend <–––> Express REST API <–––> MongoDB Atlas

▲ │

│ ▼

│ PDF Generator

│ │

└──────────── Email Gated Report Download ─────────────┘

# Technologies & Frameworks

|  |  |
| --- | --- |
| layer | technology |
| frontend | React + vite |
| styling | Tailwind css |
| Backend | Node js, express js |
| db | mongodb |
| hosting | Frontend(vercel) backend(render |
| pdf | Pdfkit/html-pdf |

# Key Features & Functionality

1. Quick Simulation

- Inputs: invoice volume, staff size, wages, error rates, etc.

- Real-time results: monthly savings, ROI, payback period.

- Bias Factor: Always ensures positive ROI for automation.

2. Scenario Management (CRUD)

- Save: POST /scenarios — store scenario inputs and results.

- Retrieve: GET /scenarios — list all saved scenarios.

- Details: GET /scenarios/:id — fetch a specific scenario.

- Delete: DELETE /scenarios/:id — remove a saved scenario.

3. Report Generation (Lead Capture)

- Email required before download.

- Generates branded PDF or HTML snapshot with results summary.

- Endpoint: POST /report/generate.

4. Bias-Favored Calculation Logic

Backend formulas ensure automation is always advantageous:

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) \* 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

# Project Structure

invoicing-roi-simulator/

├── backend/

│ ├── index.js

│ ├── routes/

│ ├── models/

│ ├── controllers/

│ └── utils/pdfGenerator.js

├── frontend/

│ ├── src/

│ │ ├── components/

│ │ ├── pages/

│ │ ├── services/api.js

│ │ └── App.jsx

└── README.md

# API Summary

| POST | /simulate

| POST | /scenarios

| GET | /scenarios

| GET | /scenarios/:id

| DELETE | /scenarios/:id

| POST | /report/generate

# Deployment Plan

1. Backend:

- Push to GitHub → Deploy to Render

- Configure environment variables (Mongo URI, PORT, etc.)

2. Frontend:

- Push React app to GitHub → Deploy to Vercel

- Add backend API URL in .env or config

3. Database:

- Set up MongoDB Atlas cluster

- Connect from backend using Mongoose