**Invoicing ROI Simulator — Assignment Round 3 Documentation**

**Date:** 07 October 2025  
**Duration:** 2 Hours  
**Author:** *[Your Name]*

**Project Overview**

The **Invoicing ROI Simulator** is a lightweight, interactive web application that allows businesses to visualize **cost savings**, **ROI**, and **payback period** when switching from manual invoicing to automation.

The tool uses a simple mathematical model with a **positive bias** towards automation to show favorable financial outcomes.  
It’s designed to be a **functional MVP** that can be built and deployed within **2 hours**.

**Goal**

Deliver a working prototype that:

* Accepts user inputs for key invoicing metrics.
* Calculates savings and ROI instantly.
* Allows users to **save, load, and delete** simulation scenarios.
* Generates a **report** (HTML or PDF) gated by an **email input**.
* Persists scenarios locally using a lightweight database.

**Planned Tech Stack**

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Frontend** | React + Tailwind CSS | Single-page, responsive UI |
| **Backend** | Node.js + Express | REST API and business logic |
| **Database** | lowdb (JSON-based) | Lightweight storage without setup |
| **Report Generation** | html-pdf / jsPDF | HTML or PDF report export |
| **Deployment** | Vercel (Frontend), Render or Railway (Backend) | Quick and free hosting |
| **Version Control** | GitHub | Code and documentation hosting |

**Architecture Overview**

**Frontend (React SPA)**

* Interactive form to input business parameters.
* Real-time simulation results (monthly savings, ROI, payback).
* Scenario list (save/load/delete).
* “Generate Report” feature with email modal.
* Simple, minimal, and mobile-friendly UI.

**Backend (Express API)**

* Performs all financial calculations using **server-side constants**.
* Exposes endpoints for:
  + /simulate — Calculate ROI and savings.
  + /scenarios — CRUD operations for saved scenarios.
  + /report/generate — Generates downloadable report (requires email).
* Uses lowdb for persistence in a local db.json file.

**Internal Constants (Hidden on Server)**

| **Constant** | **Description** | **Value** |
| --- | --- | --- |
| automated\_cost\_per\_invoice | Fixed automation cost | 0.20 |
| error\_rate\_auto | Automated error rate | 0.1% |
| time\_saved\_per\_invoice | Reduced time per invoice | 8 min |
| min\_roi\_boost\_factor | Bias factor for positive ROI | 1.1 |

**⚙️ Calculation Logic**

The ROI logic follows the PRD formulas:

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

**🚀 Implementation Plan (2 Hours Total)**

| **Phase** | **Time** | **Tasks** |
| --- | --- | --- |
| **Phase 1** | 0–10 min | Setup repo, create client (React) and server (Express) folders |
| **Phase 2** | 10–50 min | Implement backend (/simulate, /scenarios, /report) using lowdb |
| **Phase 3** | 50–105 min | Build frontend form + results + scenario management + email modal |
| **Phase 4** | 105–120 min | Test end-to-end, style UI, finalize README, and deploy |

**Fallback plan:**  
If time is short, skip PDF generation and render an HTML “report view” instead.  
If deployment takes too long, submit local run instructions with screenshots.

**🧠 Data Model (lowdb Example)**

{

"scenarios": [

{

"id": "uuid-1",

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

"inputs": {

"monthly\_invoice\_volume": 2000,

"num\_ap\_staff": 3,

"avg\_hours\_per\_invoice": 0.17,

"hourly\_wage": 30,

"error\_rate\_manual": 0.5,

"error\_cost": 100,

"time\_horizon\_months": 36,

"one\_time\_implementation\_cost": 50000

},

"results": {

"monthly\_savings": 8000,

"payback\_months": 6.3,

"roi\_percentage": 400

}

}

]

}

**🔗 API Endpoints**

| **Method** | **Endpoint** | **Purpose** |
| --- | --- | --- |
| **POST** | /simulate | Run ROI simulation |
| **POST** | /scenarios | Save a simulation scenario |
| **GET** | /scenarios | List all saved scenarios |
| **GET** | /scenarios/:id | Retrieve scenario details |
| **DELETE** | /scenarios/:id | Delete scenario |
| **POST** | /report/generate | Generate email-gated report |

**🖥️ Frontend UI Overview**

**Components:**

1. App.jsx – main wrapper with form and result display
2. FormSection.jsx – inputs for simulation parameters
3. ResultCard.jsx – displays monthly savings, payback, ROI
4. ScenarioList.jsx – view, load, and delete scenarios
5. ReportModal.jsx – email prompt before generating report

**Layout:**  
Two-column responsive layout →  
**Left:** Inputs  
**Right:** Results + Actions

**Deployment Plan**

* **Frontend:** Deploy using **Vercel** (https://roi-simulator.vercel.app)
* **Backend:** Deploy using **Render** (https://roi-simulator-api.onrender.com)
* **Database:** Local JSON file persisted in Render service (lowdb).

If deployment isn’t possible within the window, app can be tested locally using:

npm run dev # React frontend

node index.js # Express backend