Invoicing ROI Simulator - Technical Documentation

1. Purpose

Create a lightweight ROI calculator that helps users visualize cost savings, payback period, and ROI when switching from manual to automated invoicing.

2. Core Features

- Quick Simulation: Enter metrics to calculate monthly savings, ROI, payback period.
- Scenario Management (CRUD): Save, load, view, delete scenarios.
- Report Generation: Download PDF after providing email.
- Bias-Favored Output: Automation always shows positive ROI.
- Responsive Frontend: Clean React interface with live calculations.

3. Tech Stack

Frontend: React + TailwindCSS
Backend: Node.js + Express.js
Database: SQLite (Prisma ORM)
PDF Generation: pdfkit
API Testing: Postman

4. Calculation Logic

Inputs: monthly_invoice_volume, num_ap_staff, avg_hours_per_invoice, hourly_wage, error_rate_manual, error_cost, time_horizon_months, one_time_implementation_cost
Constants: automated_cost_per_invoice=0.20, error_rate_auto=0.1%, time_saved_per_invoice=8min, min_roi_boost_factor=1.1
Formulas:
manual_cost_per_month = monthly_invoice_volume * avg_hours_per_invoice * hourly_wage
automation_cost_per_month = monthly_invoice_volume * automated_cost_per_invoice
manual_error_cost = monthly_invoice_volume * (error_rate_manual/100) * error_cost
automated_error_cost = monthly_invoice_volume * (error_rate_auto/100) * error_cost
error_savings = manual_error_cost - automated_error_cost
monthly_savings = (manual_cost_per_month - automation_cost_per_month) + error_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

5. System Architecture

Frontend (React) -> Backend (Node+Express) -> SQLite Database
APIs: /simulate, /save, /scenarios, /scenarios/:id, /report

6. Database Schema

Table: scenarios
Fields: id (PK), scenario_name, inputs (JSON), results (JSON), created_at

7. API Design

POST /simulate: Run ROI simulation

POST /save: Save scenario

GET /scenarios: List all scenarios

GET /scenarios/:id: Retrieve scenario

DELETE /scenarios/:id: Delete scenario

POST /report: Generate PDF after email

8. Frontend Workflow

1. User fills inputs -> calls /simulate -> displays results
2. Save Scenario -> calls /save
3. View Scenarios -> calls /scenarios
4. Download Report -> email -> /report -> PDF

9. UI Structure

Header: Project title/logo

InputForm.jsx: Input fields

ResultCard.jsx: Display results

ScenarioList.jsx: Manage scenarios

EmailModal.jsx: Capture email before PDF

10. Testing Plan

Unit Testing: ROI calculations, API responses

Integration Testing: Frontend-backend flow

Validation Testing: Input checks

PDF Testing: Download correctness

CRUD Testing: Scenario persistence

11. Setup Instructions

Prerequisites: Node.js v18+, npm/yarn, SQLite

Backend:

```
 cd backend
 npm install
 npm run dev
```

Frontend:

```
 cd frontend
 npm install
 npm run dev
```

Access app at: http://localhost:5173

Backend: http://localhost:3000

12. Example Output

Inputs: Monthly Invoice Volume:2000, Hourly Wage:$30, Avg Hours/Invoice:0.17, Error Rate:0.5%, Error Cost:$100, Horizon:36 months, Setup Cost:$50,000

Results: Monthly Savings: $8,000, Payback Period: 6.3 months, ROI: 412%

13. Acceptance Criteria

- Validated inputs in UI
- Backend constants hidden
- Positive ROI always
- Email required before PDF
- CRUD works with SQLite
- Documentation complete

14. Deliverables

- Full source code
- Working prototype
- README with setup & testing guide
- Technical documentation PDF