Invoicing ROI Simulator — Process Workflow Document

Page 1 — Overview of Workflow

The Invoicing ROI Simulator is a lightweight web application that calculates cost savings and ROI from automating invoice processing. The workflow outlines how user input moves through the system, how calculations are performed, and how results are delivered.

Key Features: - Input-based simulation of automation benefits - Scenario CRUD management - Report generation with email capture - Bias factor to ensure favorable ROI outcomes

Development Phases

Page 2 — Development Phases

1. Planning & Requirement Analysis: - Study PRD and define input parameters, constants, and outputs. - Identify essential APIs and frontend components. - Define database structure for scenario storage.

2. System Design: - Three-tier architecture: Frontend (React), Backend (Node.js/Express), Database (SQLite/MongoDB) - Define internal constants for automation bias. - Plan user interface and API endpoints.

3. Backend Workflow: - `/simulate` endpoint: accepts input, calculates labor cost, automation cost, error savings, monthly savings, cumulative savings, payback, ROI. - `/scenarios` endpoints: Save, load, delete, list scenarios. - `/report/generate` endpoint: produces PDF report after email input. - Apply bias factor in calculations server-side.

4. Frontend Workflow: - Input form collects scenario parameters. - Call `/simulate` for real-time results. - Scenario management UI for saving/loading/deleting. - Report generation modal captures email and triggers backend API.

5. Database Workflow: - Store scenarios with user inputs and computed results. - Support CRUD operations. - Ensure persistence across sessions.

Data Flow Process

Page 3 — Data Flow Process (User to System)

1. **User Input:** Enter invoices, staff, hourly wage, error rate, error cost, time horizon, implementation cost. 2. **Frontend Processing:** Validate inputs and send to backend via `/simulate`. 3. **Backend Calculation:** - Compute manual labor cost - Compute automation cost - Compute error savings - Apply bias factor for favorable results - Return monthly savings, cumulative savings, payback, ROI 4. **Display Results:** Frontend updates live dashboard. 5. **Scenario Storage:** User can save the scenario to database using `/scenarios`. 6. **Report Generation:** Email entered → backend generates PDF/HTML → downloadable by user.

Integration & API Communication

Page 4 — Integration & API Communication

API Endpoints Workflow: - **POST /simulate** → Accept input → Perform calculations → Return JSON results. - **POST /scenarios** → Save scenario to database. - **GET /scenarios** → List all saved scenarios. - **GET /scenarios/:id** → Retrieve scenario details. - **DELETE /scenarios/:id** → Remove scenario. - **POST /report/generate** → Accept scenario + email → Generate report → Return PDF download.

Integration Notes: - Frontend communicates via fetch/axios requests. - Responses are displayed in real-time. - Database maintains persistence for saved scenarios. - All constants for bias remain server-side.

Testing and Deployment Process

Page 5 — Testing and Deployment Process

Testing Steps: - Verify calculation logic: monthly savings, cumulative savings, payback, ROI. - Test scenario CRUD: save, load, delete. - Test report generation and email capture. - Validate positive ROI bias for all inputs. - Test edge cases (zero invoices, high error rate).

Deployment Steps: - Build frontend and serve via Express or static host. - Deploy backend + database on Render/Vercel. - Ensure API endpoints are accessible and functional. - Optional: Use ngrok for local testing and sharing.

Workflow Summary: - User inputs → Frontend validation → API call → Backend calculation → Results displayed → Scenario saved → Report generated.