To reflect the updated app name PCS-B-LTD (replacing TradeTrack Pro), I’ll provide a revised plan to set up local and online development environments, develop the system using Bolt.new, and deploy it for daily use in a real-world business environment, ensuring all business logic, functionalities, and CRUD operations are fully implemented. The system specifications remain the same, including the local backend (PostgreSQL, Redis, Node.js/Express, TypeScript, JWT, Socket.io, Multer, Docker), Next.js PWA, React Native mobile app, and advanced features (AI, IoT, blockchain, ESG tracking), tailored for African SMEs with offline-first capabilities, mobile money integration, and multi-language support. The development will follow the 1-week timeline, leveraging Bolt.new for the PWA and backend (~70-75% of the system) and external tools (Expo, libraries) for the mobile app and advanced features.Below is the updated step-by-step process to craft PCS-B-LTD from start to deployment, ensuring it’s ready for daily business use.

PCS-B-LTD System Specifications Recap

* Backend: Local stack with PostgreSQL (database), Redis (caching), Node.js/Express (API), TypeScript, JWT (authentication), Socket.io (real-time), Multer (file storage), Docker (deployment).
* Frontend: Next.js PWA with offline-first capabilities (service workers), role-based UIs (admin, manager, worker, client), multi-language support (Swahili, French, Kinyarwanda, Luganda), gamification (badges, leaderboards).
* Mobile App: React Native (Expo) with offline sync, mobile money integration (M-Pesa, Airtel, MTN), push notifications.
* Key Features:
  + CRUD Operations: For users, projects, inventory, transactions, audit logs, ESG metrics.
  + Business Logic: Role-based access control (RBAC), validation, workflows, payment processing, real-time updates.
  + Advanced Features: AI analytics (demand forecasting, customer behavior), IoT (asset tracking), blockchain (supply chain transparency), ESG tracking (carbon emissions, social impact), marketplace, collaboration tools (messaging, Kanban).
* Market Focus: African SMEs with offline-first design, low data usage, financial inclusion.
* Deployment: Self-hosted Docker containers for VPS, on-premises, or cloud, emphasizing data ownership and offline capability.
* Objective: Set up local and online development environments, use Bolt.new to build the PWA and backend, supplement with external tools for the mobile app and advanced features, and deploy for daily use by businesses.

Step 1: Set Up Local Development EnvironmentThe local environment enables development and testing on your machine, using Docker to mirror the production setup.Prerequisites

* Hardware: Computer with 8GB RAM, 4-core CPU, 20GB free disk space.
* Software:
  + Docker Desktop: For containers (PostgreSQL, Redis, backend, frontend). Install from [docker.com](https://www.docker.com/products/docker-desktop).
  + Node.js (LTS): For Bolt.new and local tools. Download from [nodejs.org](https://nodejs.org/).
  + Git: For version control. Install from [git-scm.com](https://git-scm.com/).
  + VS Code: For editing and external tools (Expo). Install from [code.visualstudio.com](https://code.visualstudio.com/).
  + Expo CLI: For React Native. Install: npm install -g expo-cli.
* Accounts: GitHub for version control, optional Vercel/Supabase for online environment.

Local Environment Setup

1. Install Docker Desktop:
   * Download and install for your OS (Windows, macOS, Linux).
   * Verify: docker --version and docker-compose --version.
2. Set Up Project Directory:

bash

mkdir pcs-b-ltd

cd pcs-b-ltd

mkdir backend frontend mobile

* + backend: Node.js/Express API.
  + frontend: Next.js PWA.
  + mobile: React Native app.

1. Initialize Git Repository:

bash

git init

echo "node\_modules\n.env\n.next\nbuild\nuploads" > .gitignore

git add .

git commit -m "Initial commit for PCS-B-LTD"

git remote add origin <your-repo-url>

git push -u origin main

1. Set Up Docker Containers:
   * Create docker-compose.yml in the root:

yaml

version: '3.8'

services:

pcsb-postgres:

image: postgres:15

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: your\_password

POSTGRES\_DB: pcsbltd

ports:

- "5432:5432"

volumes:

- pgdata:/var/lib/postgresql/data

pcsb-redis:

image: redis:7

ports:

- "6379:6379"

volumes:

pgdata:

* + Start containers:

bash

docker-compose up -d

* + Verify: docker ps shows pcsb-postgres and pcsb-redis.

1. Configure Environment Variables:
   * Create .env in backend:

env

DATABASE\_URL=postgresql://postgres:your\_password@localhost:5432/pcsbltd

REDIS\_URL=redis://localhost:6379

JWT\_SECRET=your\_jwt\_secret

PORT=3001

UPLOADS\_DIR=/uploads

* + Add .env to .gitignore.

1. Set Up File Storage:
   * Create uploads folder:

bash

mkdir uploads

mkdir uploads/projects uploads/receipts

1. Access Bolt.new:
   * Open [bolt.new](https://bolt.new/) in Chrome.
   * Sign in with GitHub to save projects and export code.

Step 2: Set Up Online Development EnvironmentThe online environment supports collaboration, testing, and deployment previews using cloud services.Online Environment Setup

1. Supabase for PostgreSQL and Redis:
   * Sign up at [supabase.com](https://supabase.com/) with GitHub.
   * Create a project:
     + Choose a region (e.g., London for proximity to Africa).
     + Set database password and save credentials.
   * Configure PostgreSQL:
     + Use Supabase’s SQL Editor to create tables (generated by Bolt.new).
     + Copy connection string (e.g., postgresql://postgres:[password]@db.supabase.co:5432/postgres).
   * Enable Redis via Upstash:
     + Sign up at [upstash.com](https://upstash.com/).
     + Create a Redis instance and copy URL (e.g., redis://[credentials]@upstash.io:6379).
   * Create .env.online:

env

DATABASE\_URL=postgresql://postgres:[password]@db.supabase.co:5432/postgres

REDIS\_URL=redis://[credentials]@upstash.io:6379

JWT\_SECRET=your\_jwt\_secret

PORT=3001

UPLOADS\_DIR=/uploads

1. Vercel for PWA Deployment:
   * Sign up at [vercel.com](https://vercel.com/) with GitHub.
   * Create a project:
     + Import frontend from GitHub.
     + Set framework to Next.js, add environment variables from .env.online.
     + Deploy to get a preview URL (e.g., https://pcs-b-ltd.vercel.app).
   * Enable automatic redeploys on Git push.
2. GitHub for CI/CD:
   * Push code to GitHub.
   * Set up GitHub Actions for testing:

yaml

name: CI

on:

push:

branches: [main]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Set up Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

- run: npm ci

- run: npm test

Step 3: Development with Bolt.new (Days 1-4)Bolt.new will build the Next.js PWA and Node.js/Express backend, including CRUD operations and business logic. External tools will handle the mobile app and advanced features.Day 1: Backend Development

* Tasks:
  + Open Bolt.new, create project PCS-B-LTD Backend.
  + Generate database schemas:

text

Generate PostgreSQL schemas with migrations for a business management system named PCS-B-LTD with tables for users, roles, permissions, projects, tasks, assignments, inventory, stock\_movements, transactions, payments, audit\_logs, system\_logs, and esg\_metrics, including foreign key relationships and TypeScript models.

* + - Tables: users (id, email, password, role\_id), roles (id, name), permissions (id, role\_id, endpoint), projects (id, user\_id, name), etc.
  + Generate Node.js/Express API:

text

Create a Node.js/Express API with TypeScript for PCS-B-LTD, including JWT authentication, role-based access control, Socket.io for real-time updates, and RESTful CRUD endpoints for users, projects, inventory, transactions, audit logs, and ESG metrics. Include middleware for validation and business logic for payment processing, task workflows, and audit logging.

* + - Endpoints: /api/auth/login, /api/users, /api/projects, /api/inventory, /api/transactions, /api/esg, etc.
    - Middleware: JWT, RBAC, validation.
    - Socket.io: Events for task updates, payments.
  + Generate business logic:

text

Implement business logic for PCS-B-LTD, including role-based access control (admin, manager, worker, client), input validation for inventory and transactions, workflows for task assignment and payment processing, and audit logging for all actions.

* + - Example: Only managers assign tasks; payments validate amounts and log to audit\_logs.
  + Connect to local PostgreSQL/Redis:
    - Update Bolt.new’s .env with local DATABASE\_URL and REDIS\_URL.
    - Run migrations: npx prisma migrate dev (if using Prisma) or manual SQL.
  + Test APIs: Use Bolt.new’s preview (e.g., http://localhost:3001/api/users).
  + Export code to backend directory, push to GitHub.
* Output: Backend with CRUD endpoints, JWT, RBAC, Socket.io, and business logic.

Days 2-3: PWA Development

* Tasks:
  + Create Bolt.new project PCS-B-LTD Frontend.
  + Generate Next.js PWA:

text

Build a Next.js PWA for PCS-B-LTD with offline-first capabilities using service workers, role-based UI for admin, manager, worker, and client roles, multi-language support (Swahili, French, Kinyarwanda, Luganda), gamification (badges, leaderboards), and Socket.io integration for real-time updates. Include CRUD operations via API integration.

* + - Features: Role-based dashboards, offline sync, translations (next-i18next), badges/leaderboards, Socket.io notifications.
  + Generate reports:

text

Add PDF report generation to a Next.js PWA for PCS-B-LTD for project, inventory, and transaction data using pdfkit, integrated with role-based access.

* + Connect to backend (http://localhost:3001).
  + Test offline sync, role-based access, and real-time updates.
  + Export code to frontend directory, push to GitHub.
* Output: Next.js PWA with role-based UIs, offline capabilities, multi-language support, gamification, and CRUD integration.

Day 4: Testing and Integration

* Tasks:
  + Test locally:
    - CRUD operations: Create/update/delete users, projects, etc.
    - RBAC: Ensure access restrictions (e.g., non-admins blocked from /api/users).
    - Offline sync: Disconnect network, test actions, reconnect.
    - Socket.io: Verify real-time notifications.
  + Deploy PWA to Vercel:
    - Push frontend to GitHub, import to Vercel, set environment variables.
    - Test at https://pcs-b-ltd.vercel.app.
  + Test backend with Supabase:
    - Update .env with Supabase’s DATABASE\_URL and REDIS\_URL.
    - Run migrations on Supabase.
    - Test APIs using Postman.
* Output: Tested PWA and backend, with online previews.

Step 4: Mobile App and Advanced Features (Days 5-6)Bolt.new doesn’t support React Native or advanced features, so we’ll use external tools.Day 5: React Native Mobile App

* Tasks:
  + Initialize app:

bash

cd mobile

npx create-expo-app pcsb-mobile

* + Implement features:
    - Role-based UIs mirroring PWA.
    - Offline sync: AsyncStorage for local storage, queue API calls.
    - Mobile money: Integrate M-Pesa, Airtel, MTN APIs (e.g., M-Pesa Daraja).
    - Push notifications: Expo Notifications.
    - Socket.io: Connect to backend.
    - Use Cursor IDE for code:

text

Generate a React Native app with Expo for PCS-B-LTD, featuring offline-first sync with AsyncStorage, role-based UI (admin, manager, worker, client), mobile money integration (M-Pesa, Airtel, MTN), push notifications, and Socket.io for real-time updates.

* + Connect to backend (http://localhost:3001 or Supabase).
  + Test with Expo Go:

bash

npx expo start

* Output: Mobile app with offline sync, mobile money, and notifications.

Day 6: Advanced Features

* Tasks:
  + AI Analytics:
    - Use TensorFlow.js for demand forecasting, customer behavior analysis.
    - Integrate into backend:

text

Add TensorFlow.js to a Node.js/Express API for PCS-B-LTD for demand forecasting and customer behavior analysis based on transaction and inventory data.

* + IoT:
    - Use AWS IoT for asset tracking (e.g., vehicles).
    - Add backend endpoints.
  + Blockchain:
    - Use Hyperledger for supply chain transparency.
    - Set up local Hyperledger Fabric or use Infura.
  + ESG Tracking:
    - Create modules for emissions and social impact:

text

Add ESG tracking endpoints to a Node.js/Express API for PCS-B-LTD for calculating carbon emissions and social impact metrics based on inventory and transaction data.

* + Collaboration Tools:
    - Add Twilio messaging, React-based Kanban boards.
  + Marketplace:
    - Extend APIs and UI:

text

Extend a Node.js/Express API and Next.js PWA for PCS-B-LTD to include marketplace endpoints and UI for supplier-client connections, including search, listings, and transaction integration.

* + Test integrations locally and on Supabase/Vercel.
* Output: Advanced features integrated into backend, PWA, and mobile app.

Step 5: Deployment for Real-World Use (Day 7)Deploy PCS-B-LTD for daily business use with a self-hosted Docker setup.Deployment Setup

1. Create Docker Compose:
   * Update docker-compose.yml:

yaml

version: '3.8'

services:

pcsb-backend:

build:

context: ./backend

dockerfile: Dockerfile

ports:

- "3001:3001"

environment:

- DATABASE\_URL=postgresql://postgres:your\_password@pcsb-postgres:5432/pcsbltd

- REDIS\_URL=redis://pcsb-redis:6379

- JWT\_SECRET=your\_jwt\_secret

- UPLOADS\_DIR=/uploads

volumes:

- ./uploads:/uploads

depends\_on:

- pcsb-postgres

- pcsb-redis

pcsb-frontend:

build:

context: ./frontend

dockerfile: Dockerfile

ports:

- "3000:3000"

pcsb-postgres:

image: postgres:15

environment:

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=your\_password

- POSTGRES\_DB=pcsbltd

volumes:

- pgdata:/var/lib/postgresql/data

pcsb-redis:

image: redis:7

pcsb-nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

depends\_on:

- pcsb-backend

- pcsb-frontend

volumes:

pgdata:

* + Create backend/Dockerfile:

dockerfile

FROM node:18

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

EXPOSE 3001

CMD ["node", "dist/index.js"]

* + Create frontend/Dockerfile:

dockerfile

FROM node:18

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]

* + Create nginx.conf:

nginx

server {

listen 80;

location /api/ {

proxy\_pass http://pcsb-backend:3001;

}

location / {

proxy\_pass http://pcsb-frontend:3000;

}

}

* + Test locally:

bash

docker-compose up --build

1. Build Mobile Apps:
   * Generate APK/IPA:

bash

cd mobile

npx expo build:android

npx expo build:ios

* + Distribute via email or app stores.

1. Deploy to Production:
   * On-Premises/VPS:
     + Set up a server (e.g., Ubuntu on DigitalOcean).
     + Install Docker and Docker Compose.
     + Copy project files and run:

bash

docker-compose up -d

* + - Configure domain (e.g., pcsbltd.com) and SSL (Let’s Encrypt).
  + Cloud Fallback:
    - Deploy backend to Render or Supabase Edge Functions.
    - Deploy PWA to Vercel.
    - Update environment variables for cloud URLs.

1. Set Up Backups:
   * Schedule PostgreSQL backups:

bash

docker exec pcsb-postgres pg\_dump -U postgres pcsbltd > /backups/backup\_$(date +%F).sql

1. Documentation:
   * Generate API docs in Bolt.new:

text

Generate OpenAPI/Swagger documentation for a Node.js/Express API for PCS-B-LTD with endpoints for authentication, users, projects, inventory, transactions, reports, uploads, and ESG metrics.

* + Create multilingual user guides (Swahili, French, etc.) for admins, managers, workers, clients.
  + Include installation guide.

Step 6: Real-World Use and Support

* Training: Conduct workshops and provide multilingual video tutorials.
* Support: Set up 24/7 support via email/WhatsApp, monitor logs (audit\_logs, system\_logs).
* Maintenance: Update mobile money APIs, patch vulnerabilities, scale containers.
* Feedback: Collect user feedback via in-app surveys or marketplace reviews.

Ensuring Full Functionality

* CRUD Operations: Bolt.new-generated APIs cover all entities.
* Business Logic: RBAC, validation, workflows, and payment processing implemented.
* Real-time: Socket.io ensures live updates.
* African Market Needs: Offline-first, mobile money, multi-language, low data usage.
* Advanced Features: AI, IoT, blockchain, ESG, collaboration, marketplace via external tools.
* Deployment: Docker for data ownership; cloud fallback for scalability.
* Security: JWT, bcrypt, encryption, GDPR compliance.

Final Deliverables

* Local Environment: Dockerized PostgreSQL, Redis, backend, frontend.
* Online Environment: Supabase, Vercel, GitHub CI/CD.
* Source Code: Backend, PWA, mobile app.
* Deployment: Docker containers, APK/IPA, PWA at https://pcsbltd.com.
* Documentation: API docs, user guides, installation guide.
* Support: 24/7 channels, training materials.

Conclusion

Using Bolt.new,

PCS-B-LTD will be developed and deployed within 1 week, covering the PWA, backend, CRUD operations, and business logic, with external tools (Expo, TensorFlow.js, AWS IoT, Hyperledger) completing the mobile app and advanced features. The local and online environments ensure efficient development and testing, while the Docker-based deployment supports daily use by African SMEs, delivering a robust, offline-capable, and secure system.