■■■ PROJECT DOCUMENTATION ■■■

SME Transaction & Receipt Manager

A Full-Stack Web Application for Small & Medium Enterprises

Document Type	Project Features, User Stories & Roadmap
Target Users	SME Owners, Managers & Daily Workers
Scope	Transaction Entry · Receipt Generation · Expense Tracking · Dashboard · Exports · RBAC
Timeline	Approx. 12–14 Weeks (Solo Developer)

Table of Contents

1. Project Overview

This application is designed to give small and medium enterprises a single, easy-to-use platform for recording daily sales transactions, printing or downloading professional receipts, tracking business expenses, and viewing a real-time summary dashboard. It also lets SME owners invite workers and assign them specific roles so that every team member only sees and does what they are supposed to.

What the App Solves

- ✓ Replaces manual notebooks or loose spreadsheets with structured daily records
- ✓ Auto-generates professional receipts the moment a transaction is saved
- ✓ Gives owners a clear picture of money coming in and going out each day
- ✓ Lets owners control exactly what each worker can access or change
- ✓ Allows data export so owners can keep backups or do offline analysis

2. Core Features

Transaction Entry

Workers log each sale: item name, quantity, unit price, and payment method. The system calculates totals automatically and timestamps every record.

Receipt Generation

Every saved transaction instantly produces a clean, printable receipt showing the business name, items sold, totals, and a unique receipt number.

Expense Tracking

A dedicated section lets owners or managers record business expenses (rent, stock, utilities, etc.) with dates, categories, and amounts.

Dashboard

A summary page that shows today's total sales count, total revenue, total expenses, and net profit — all updated live as transactions are added.

Data Export

Owners can download transaction and expense records as a PDF report or as a CSV file (which opens neatly in Excel or Google Sheets).

Role-Based Access

The owner registers the business and adds workers. Each worker is assigned a role (Owner / Manager / Cashier) that controls their permissions.

3. User Stories

User stories describe the app from the perspective of each type of person using it. Each story follows the standard format: **As a** [role], **I want to** [action], **so that** [benefit]. These will guide what you build in each feature.

Owner Stories

Register & Set Up	As an owner, I want to sign up and create my business profile, so that the app knows my business name, contact info, and branding for receipts.
Add Workers	As an owner, I want to invite team members by email and assign them a role, so that everyone can log in with the right level of access.
View Full Dashboard	As an owner, I want to see daily sales, expenses and net profit on one screen, so that I can make quick decisions about the business.
Export All Data	As an owner, I want to download all transactions and expenses as a PDF or CSV, so that I have a backup and can do further analysis offline.

Manager Stories

Enter Transactions	As a manager, I want to log a sale with item details and payment method, so that every transaction is recorded accurately and on time.
Track Expenses	As a manager, I want to add an expense record with a category and amount, so that the owner can see exactly where money is being spent.
Generate Receipts	As a manager, I want receipts to appear automatically after saving a transaction, so that I can print or share them with the customer instantly.

Cashier Stories

Log a Sale	As a cashier, I want a simple form to enter what was sold and the amount paid, so that I can finish each transaction quickly during a busy day.
Print a Receipt	As a cashier, I want to print or download the receipt right after a sale, so that the customer gets their proof of purchase on the spot.
View Today's Summary	As a cashier, I want to see how many sales have been made today, so that I know the shop is on track without needing to count manually.

4. Role-Based Access Control (RBAC)

RBAC is the system that decides what each user can and cannot do. The SME owner is always the top-level user. They can add workers and assign one of three roles. The table below shows exactly which actions are allowed for each role.

Owner	Full control over everything — business settings, workers, data, and exports.
Manager	Can handle transactions, expenses, receipts, and exports. Cannot change business settings or manage workers.
Cashier	Day-to-day role. Can log sales and print receipts only. No access to expenses or exports.

Permission Matrix

Permission	Owner	Manager	Cashier
View Dashboard	✓	✓	✓
Enter Transactions	✓	✓	✓
Generate Receipts	✓	✓	✓
Print / Download Receipt	✓	✓	✓
Enter Expenses	✓	✓	✗
View Expense Records	✓	✓	✗
Export PDF / CSV	✓	✓	✗
Add / Remove Workers	✓	✗	✗
Edit Business Profile	✓	✗	✗
Delete Any Record	✓	✓	✗

5. Development Roadmap

This roadmap breaks the project into 6 sequential phases. Each phase builds on the one before it. Follow this order so that you always have a working foundation before adding the next feature.

PHASE 1 — Weeks 1–2 — Project Setup & Planning

1. Set up your development environment (VS Code, Node.js, Git)
2. Initialize the project with your chosen framework (e.g., React + Node.js)
3. Set up the database (PostgreSQL or MongoDB) and connect it
4. Create the project folder structure and install core dependencies
5. Set up Git with a remote repository (GitHub) and make your first commit
6. Plan out your file/folder structure before writing feature code

PHASE 2 — Weeks 3–4 — Authentication & RBAC

1. Build the sign-up page for SME owners to register their business
2. Build the login page with email + password (use bcrypt for password hashing)
3. Implement JSON Web Tokens (JWT) so logged-in users stay authenticated
4. Create the worker invitation flow (owner enters email, assigns a role)
5. Build the middleware that checks a user's role before allowing access to a page
6. Test that each role can only see what they are supposed to see

PHASE 3 — Weeks 5–7 — Transactions & Receipts

1. Build the transaction entry form (item, quantity, price, payment method)
2. Connect the form to the database so transactions are saved

3. Auto-generate a receipt object every time a transaction is saved
4. Build a receipt display page with clean formatting (business name, items, total)
5. Add a print button and a download-as-PDF button for each receipt
6. Build a transaction list/history page so users can view past records

PHASE 4 — Expenses & Dashboard

Weeks 8–9

1. Build the expense entry form (description, category, amount, date)
2. Save expenses to the database and show a list of all recorded expenses
3. Build the dashboard page layout with summary cards
4. Pull live data: total transactions today, total revenue, total expenses, net profit
5. Add simple visual indicators (e.g., up/down arrows or color coding for profit vs. loss)
6. Make the dashboard the default landing page after login

PHASE 5 — Data Export & Polish

Weeks 10–11

1. Implement PDF export: generate a report of transactions/expenses using a PDF library
2. Implement CSV export: convert transaction and expense data into downloadable .csv files
3. Add date-range filtering so users can export data for a specific period
4. Polish the UI: consistent colors, spacing, responsive layout for mobile
5. Add loading states, error messages, and empty-state screens
6. Make sure all forms have proper validation (no empty fields, valid numbers, etc.)

PHASE 6 — Testing & Deployment

Weeks 12–14

1. Write manual test cases for every feature and walk through them as each role
2. Fix any bugs you find — pay special attention to RBAC and data accuracy
3. Test on different browsers (Chrome, Firefox, Safari) and screen sizes
4. Choose a hosting platform (Vercel, Railway, or Render) and deploy the app

5. Set up environment variables for your database and secrets on the hosting platform
6. Do a final end-to-end walkthrough before sharing the app with real users

6. Recommended Tech Stack

This is a beginner-friendly stack. Every tool listed here has excellent documentation, large communities, and plenty of tutorials available online. You do not need to know all of these before you start — learn each one as you reach the relevant phase.

Category	Technology	What It Does
Frontend	**React.js**	UI library — builds the pages the user sees
CSS	**Tailwind CSS**	Utility-first styling — fast and flexible
Backend	**Node.js + Express**	Runs the server and handles API requests
Database	**PostgreSQL**	Stores all data reliably (relational)
ORM	**Prisma**	Lets you talk to the database using JavaScript
Auth	**JWT + bcrypt**	Keeps users logged in and passwords safe
PDF Gen	**jsPDF or PDFKit**	Creates downloadable PDF reports & receipts
Hosting	**Vercel (frontend)**	Deploys your React app instantly
Hosting	**Railway (backend)**	Hosts your Node.js server + database
Version Ctrl	**Git + GitHub**	Saves your work and tracks every change

7. Database Schema Overview

These are the main data tables your app will need. You do not need to design them perfectly from day one — start with these core tables and add fields as you go.

Table: businesses

Columns: id · name · address · phone · logo_url · created_at

Stores each SME's profile information used on receipts.

Table: users

Columns: id · email · password_hash · role · business_id · created_at

Each person (owner, manager, cashier). Role field controls permissions.

Table: transactions

Columns: id · business_id · user_id · items (JSON) · total · payment_method · created_at

Every sale record. Items can be stored as a JSON array of objects.

Table: receipts

Columns: id · transaction_id · receipt_number · generated_at

Tracks each receipt linked to a transaction.

Table: expenses

Columns: id · business_id · user_id · description · category · amount · date · created_at

Each expense entry with a category for filtering and reporting.

8. Tips & Best Practices

Commit Often

Push your code to GitHub at least once every day, or even after each small feature. This way you always have a safe point to go back to if something breaks.

Build One Thing at a Time

Finish one feature completely before starting the next. Do not try to build everything at once — it leads to bugs that are very hard to find.

Test as You Go

After adding each feature, manually open the app and click through it as if you were a real user. Catch problems early before they stack up.

Use Environment Variables

Never hardcode passwords or database URLs in your code. Put them in a .env file and add that file to .gitignore so it never gets pushed to GitHub.

Keep It Simple First

Get the basic version working before adding nice-to-have features like animations or advanced filters. A working app always beats a beautiful but broken one.

Read Error Messages

When something breaks, read the error message fully before searching for help. Most errors tell you exactly what went wrong and even which line of code caused it.