Project Design and Management

Gang_X1

Hrudith Lakshminarasimman
Jhanghir Babar
Sudeep Pasula
Rahul Gonsalves
Rahul Saravanakumar
Nicholas Vuong

Executive Summary

Build a hosted Sharetea POS frontend as a SaaS that layers on your existing PostgreSQL + POS logic. Deliver browser-based UIs for Manager, Cashier, Customer Kiosk, Kitchen Display, and Menu Board. Integrate OAuth, translation, weather, and one team-chosen API. Center development on user stories, usability tests, and WCAG-aligned accessibility. Ship an MVP in Sprint 1, iterate to a stable pilot by Sprint 3. This extends Project 2's data, reports, and flows rather than replacing them.

Personas and User Stories

Cashier "Alex"
Context: Handles peak-hour orders. Needs sub-5-second item search, keyboard shortcuts, offline queueing.
Accessibility: High-contrast theme for glare, large-target buttons.

Manager "Sam"
Context: Reviews sales, edits menu, manages inventory and staff. Needs audit logs, role controls, scheduled exports.
Accessibility: Focus order for keyboard, skip-to-content.

Accessibility persona "Jordan"
Context: Customer using self-service kiosk with low vision and limited dexterity. Needs screen reader support, voice prompts, 2.5× scalable UI, and single-switch flow with dwell select.
Accessibility: WCAG 2.1 AA text contrast, error prevention, consistent regions, ARIA labels.

User stories:

- As a Cashier, I want to add toppings and sugar/ice levels with one key so that I reduce taps during rush.
- As a Cashier, I want orders to auto-print to Kitchen Display so that prep starts immediately.
- As a Manager, I want to A/B test prices and names so that I improve conversion.
- As a Manager, I want real-time stock warnings so that I prevent selling out mid-shift.
- As a Customer, I want a kiosk with large buttons and voice so that I can order independently.
- As a Customer, I want allergen flags to surface before checkout so that I avoid unsafe items.

Proposed Work

Solution Design

Frontend: React + TypeScript + Vite. State: Redux Toolkit + RTK Query. Routing: React Router.
Styling: Tailwind + Headless UI with ARIA patterns.
Backend: Node/Express API gateway in front of the existing POS DB. Services: Auth, Orders, Menu, Inventory, Reports.
Database: PostgreSQL (reuse schema and data from Project 2).
Integrations:

- Google OAuth 2.0 for SSO.
- Google Translate API for on-demand menu localization.
- OpenWeather API to tag orders with weather context for demand analysis.
- Team-chosen API (e.g., SendGrid for receipts or Twilio for pickup SMS).
  Observability: Winston logs, OpenAPI spec, Prisma schema docs.
  Security: RBAC, parameterized queries, input validation, audit trails.
  Rationale: preserves proven data and flows from Project 2 while enabling a modern web UI.

System Diagram

Clients (Manager Web, Cashier Web, Kiosk, Kitchen Display, Menu Board) → API Gateway (Auth, Orders, Menu, Inventory, Reports) → PostgreSQL.
Event bus publishes OrderPlaced, OrderUpdated to Kitchen Display and Menu Board.
CDN serves static assets. OAuth provider issues tokens for web clients. Scheduled jobs generate nightly analytics.

Interface Diagram

Cashier: grid menu, quick modifiers, barcode hotkeys, offline cart, error-proof tendering.
Manager: dashboards, CRUD for menu and inventory, price tests, export.
Kiosk: guided steps with 44px+ targets, semantic headings, ARIA live regions, SR-only labels, voice prompts, contrast ≥ 4.5:1, zoom to 250% without reflow.

Kitchen Display: ticket queue, batching by base, timers, recall.
Menu Board: auto-rotating categories, allergen icons, dynamic 86ing.
All accessibility features map to the assigned persona needs and WCAG 2.1 AA.

Project Management

Team Roles and Qualifications

Project Manager: owns roadmap, backlogs, burn-down, release notes. Also builds Reports service.
Frontend Lead: design system, accessibility compliance, kiosk flow.
POS Frontend: cashier UI, shortcuts, offline cart.
Services Lead: Orders + Inventory APIs, event bus.
Data/Analytics: dashboards, A/B testing, forecasting hooks.
QA/UX: usability tests, persona scripts, telemetry, accessibility audits.

Development Methodology

Agile SCRUM. Two-week sprints. Daily async stand-up in chat. Three weekly SCRUM touchpoints: Mon lab, Wed class window, Fri sync. Tracking in GitHub Projects with user-story points. CI for tests and lint. Definition of Done: tests passing, a11y checks, review, deploy to staging.

Planned Scope

Baseline (MVP, Sprint 1):

- Cashier web POS with core order → pay → print.
- Kitchen Display subscribe to new orders.
- OAuth login with RBAC.
- Menu read-only from PostgreSQL.

Fallback:

- Disable A/B testing and price experiments.
- Defer translation to phase 2.
- Manager edits limited to price/availability.

Stretch:

- Customer kiosk with voice + translation.
- Menu board service with live 86ing.

- Demand-aware recommendations using weather.
- SMS/e-receipt integration.
  These extend Project 2 functionality in a non-trivial way.

Task Breakdown and Scheduling

Epics → key tasks:

- Auth & RBAC: OAuth flow, token guards, role routes.
- Orders: cart model, modifiers, payments, tickets, event publishing.
- Menu: schema adapter, fast search, a11y labels, allergen flags.
- Inventory: stock read, 86ing, low-stock alerts.
- Kiosk: stepper UI, SR support, voice prompts, large-target layout.
- Kitchen Display: queue, batching, timers, recall.
- Analytics: sales summary, hourly heatmap, weather tagging, exports.
- Ops: CI/CD, logging, feature flags, seed scripts.

Dependencies and critical path:
DB adapter → Orders API → Cashier UI → Kitchen Display.
Auth baseline precedes any protected UI.
Menu adapter precedes kiosk and cashier flows.
CI/CD precedes usability testing on staging.

Initial schedule (6 weeks, 3 sprints):

- Sprint 1: Auth, Menu read, Orders basic, KDS v1, staging deploy, MVP.
- Sprint 2: Inventory hooks, 86ing, cashier shortcuts, analytics v1, a11y audit fix list.
- Sprint 3: Kiosk v1, translations, weather tagging, price tests, polish, hardening.

Burn-down: track story points daily and actual hours weekly. Start with only story points in the initial chart.

References

[1] Google, "OAuth 2.0 Authorization Framework," *Google Developers*, 2025. [Online]. Available: https://developers.google.com/identity/protocols/oauth2

[2] Google, "Google Translate API Guide," *W3Schools*, 2025. [Online]. Available: https://www.w3schools.com/howto/howto_google_translate.asp

[3] OpenWeather, "OpenWeather API Documentation," *OpenWeather*, 2025. [Online]. Available: https://openweathermap.org/api

[4] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.1," 2018. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/

[5] S. Horton and W. Quesenbery, *A Web for Everyone: Designing Accessible User Experiences*. Brooklyn, NY: Rosenfeld Media, 2014.

[6] H. Lakshminarasimman, R. Saravanakumar, R. Gonsalves, N. Vuong, J. Babar, and S. Pasula, *Sharetea POS System – Comprehensive Documentation*. College Station, TX: Texas A&M University, CSCE 331 Project 2, 2025.

[7] React Team, "React – A JavaScript Library for Building User Interfaces," *Meta Open Source*, 2025. [Online]. Available: https://react.dev/

[8] Tailwind Labs, "Tailwind CSS Documentation," *Tailwind Labs*, 2025. [Online]. Available: https://tailwindcss.com/

[9] Redux Team, "Redux Toolkit Documentation," *Redux*, 2025. [Online]. Available: https://redux-toolkit.js.org/

[10] Node.js Foundation, "Node.js Documentation," *Node.js Foundation*, 2025. [Online]. Available: https://nodejs.org/en/docs/

[11] PostgreSQL Global Development Group, "PostgreSQL Documentation," *PostgreSQL*, 2025. [Online]. Available: https://www.postgresql.org/docs/

Appendix 1: Product Backlog

Sprint 1 – Core Web MVP

Epic: Frontend setup

- Init React project
- Set routing
- Add layout
- Theme setup

- Basic UI components

Epic: Backend API setup

- Init Express server
- PostgreSQL schema
- Auth routes
- CRUD routes
- Test endpoints

Epic: Manager dashboard

- Login screen
- Menu tab
- Inventory tab
- Reports tab
- Logout flow

Epic: Cashier POS UI

- Order builder
- Payment modal
- Receipt generation
- Discount logic
- Order history

Sprint 2 – Multi-Interface + APIs

Epic: Customer kiosk

- Welcome screen
- Item selection
- Order confirmation
- Language toggle
- Checkout screen

Epic: Menu board

- Auto-refresh view
- Category cycle
- Item availability sync
- High-contrast mode
- Price updates

Epic: Kitchen display

- Order queue
- Ready/Done buttons
- Timer alert
- Order priority
- Error recovery

Epic: External APIs

- OAuth2 login
- Translate API
- Weather API
- Team-choice API
- API documentation

Sprint 3 – Accessibility + Deployment

Epic: Accessibility improvements

- Keyboard nav
- Screen reader labels
- Color contrast fixes
- Focus indicators
- Resizable text

Epic: Persona testing

- Maria flow test
- Vishnu flow test
- Carol flow test
- Accessibility audit
- Usability feedback

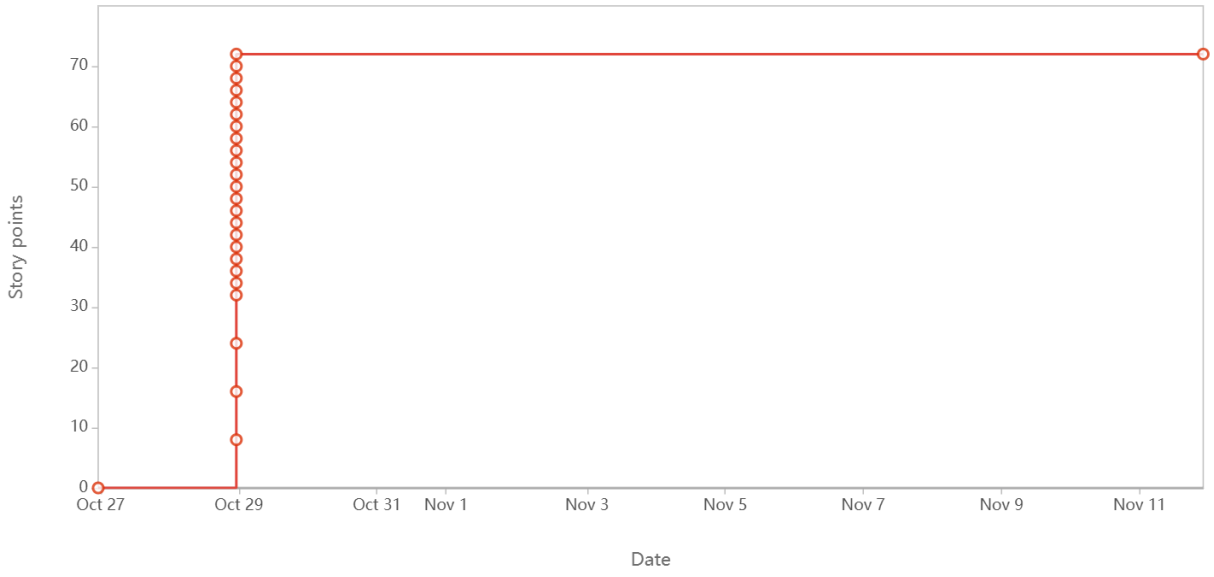
Epic: Testing and CI/CD

- Unit tests
- Integration tests
- E2E tests
- GitHub Actions pipeline
- CI monitoring

Epic: Deployment

- AWS EC2 setup
- PostgreSQL RDS config
- Domain + SSL
- Backup strategy
- Monitoring alerts

Appendix 2: Initial Product Burn-down Chart



Appendix 3: Initial Sprint Backlog

Sprint 1 27 Oct – 9 Nov (20 work items) — 0 0 0 — Complete sprint

TEAM-32	Order history	CASHIER POS UI	TO DO
TEAM-31	Discount logic	CASHIER POS UI	TO DO
TEAM-30	Receipt generation	CASHIER POS UI	TO DO
TEAM-29	Payment modal	CASHIER POS UI	TO DO
TEAM-28	Order builder	CASHIER POS UI	TO DO
TEAM-27	Logout flow	MANAGER DASHBOARD	TO DO
TEAM-26	Reports tab	MANAGER DASHBOARD	TO DO
TEAM-25	Inventory tab	MANAGER DASHBOARD	TO DO
TEAM-24	Menu tab	MANAGER DASHBOARD	TO DO
TEAM-23	Login screen	MANAGER DASHBOARD	TO DO
TEAM-22	Test endpoints	BACKEND API SETUP	TO DO
TEAM-21	CRUD routes	BACKEND API SETUP	TO DO
TEAM-20	Auth routes	BACKEND API SETUP	TO DO
TEAM-19	PostgreSQL schema	BACKEND API SETUP	TO DO
TEAM-18	Init Express server	BACKEND API SETUP	TO DO
TEAM-17	Basic UI components	FRONTEND SETUP	TO DO
TEAM-16	Theme setup	FRONTEND SETUP	TO DO
TEAM-15	Add layout	FRONTEND SETUP	TO DO
TEAM-14	Set routing	FRONTEND SETUP	TO DO
TEAM-13	Init React project	FRONTEND SETUP	TO DO