ClearKeep — Frontend Plan (MVP → V1)

This is the working plan for building the **web frontend** that sits on top of the backend we just stabilized (Parishioners → Sacraments → Transactions + Calendar). Use this as our single source of truth in the new chat.

---

0) Quickstart (local dev)

- **Location**: `C:\ckchurch1\clearkeep-frontend-starter`
- **Node**: v22.x OK (you have 22.17.0)
- **Install**: `npm i`
- **Env**: create `.env.local` with:

```
NEXT_PUBLIC_API_BASE=http://127.0.0.1:8000
NEXT_PUBLIC_TZ=Asia/Manila
```

- **Run**: `npm run dev` → http://localhost:3000
- **Build**: `npm run build && npm run start`
- **Lint**: `npm run lint`

The app assumes the backend is running locally on 127.0.0.1:8000 and that sacraments auto-sync to Transaction + Calendar (as per the continuity pack).

---

1) Goals & success criteria

Primary goal: A clean, reliable UI where office staff can:

- Create/search **Parishioners**
- Create/update **Sacraments**
- See the **linked Transaction** and the **Calendar event** for each Sacrament
- Browse the **Calendar** and **Ledger** (transactions) with filtering

Success:

- Creating a Sacrament from the UI shows a success toast, then the detail page with a link to the generated Transaction and Calendar event (ref `SAC-{id}`).
- Changing fee/time/date updates the linked Transaction/Calendar (after save) and the UI reflects it without refresh.
- SLA-quality performance on local hardware and clear error messages for failures.

---

2) Tech choices (why these)

- **Next.js 14 (App Router) + TypeScript** – stable, batteries-included SSR/ISR if needed.
- **Tailwind + shadcn/ui + lucide-react** – consistent design system, fast iteration.
- **TanStack Query (React Query)** – caching, revalidation, optimistic updates.
- **React Hook Form + Zod** – accessible forms with strong validation.
- **date-fns + date-fns-tz** – small, reliable timezone/formatting; default `Asia/Manila`.
- **TanStack Table** – flexible tables for lists/ledgers.
- **FullCalendar (react)** – calendar view (monthly/weekly) with event fetching from API.
- **Playwright** for E2E; **Vitest + RTL** for component tests; **MSW** for API mocks.

> Keep global state light; data lives in React Query. Use a tiny store (Zustand) only for UI app state (e.g., sidebar open, theme).

---

3) Information architecture & routes

```
/
  ├─ /dashboard              → quick stats, shortcuts
  ├─ /parishioners
  │    ├─ page.tsx           → list + filters + create button
  │    └─ [id]/page.tsx      → details + sacraments for person
  ├─ /sacraments
  │    ├─ page.tsx           → list + filters (type/date)
  │    ├─ new/page.tsx       → create form (baptism first)
  │    └─ [id]/page.tsx      → details (shows linked Tx + Calendar link)
  ├─ /transactions
  │    └─ page.tsx           → ledger view (filter by reference_no)
  └─ /calendar
       └─ page.tsx           → FullCalendar month view → event drawer
```

Navigation: left sidebar with primary sections; topbar with search/global actions.

---

4) Data contracts (frontend expectations)

Base URL: `${process.env.NEXT_PUBLIC_API_BASE}`

- **Parishioners**

- `GET /parishioners/?q=&limit=...` → list
- `POST /parishioners/` → `{ first_name, last_name, contact_number }`
- `GET /parishioners/{id}` → detail

- **Sacraments**

- `GET /sacraments/?limit=...` → list

- `POST /sacraments/` → `{ parishioner_id, date, fee, notes?, sacrament_type, details? }`
- `GET /sacraments/{id}` → returns read DTO including `{ id, sacrament_type, date, fee, notes, parishioner:{id}, details, created_at }`
- `PATCH /sacraments/{id}` → partial update
- `DELETE /sacraments/{id}` → removes linked Tx, deactivates Calendar event

- **Transactions**

- `GET /transactions/?limit=...` → list of items with `{ id, date, description, amount, type, reference_no, ... }`

- **Calendar**

- `GET /calendar/events?expand=false` → active events. Each has `{ id, title, start_at, end_at, timezone, external_ref, location, description, origin, meta }`

Invariant displayed in UI: `reference_no == external_ref == "SAC-{sacrament.id}"`.

---

5) UI/UX flows

A. Create Baptism

1. From `/sacraments/new` fill form (select existing parishioner by search/autocomplete).
2. On submit → `POST /sacraments/`.
3. Success toast with link `View sacrament` → `/sacraments/{id}`.
4. Detail page fetches:
5. Sacrament
6. Transactions list and finds `reference_no = SAC-{id}`
7. Calendar events and finds `external_ref = SAC-{id}`
8. Show linked cards (Transaction summary, Calendar slot with open-in-calendar action).

B. Edit Sacrament

- Inline edit or edit page; on save, refetch dependent queries → Transaction + Calendar reflect changes.

C. Delete Sacrament

- Confirm dialog → `DELETE /sacraments/{id}` → optimistic UI removal; verify Transaction removal and Calendar deactivation by refetch.

D. Calendar browsing

- `/calendar` shows month view, time zone Asia/Manila. Click event opens drawer with details + link to related Sacrament.

E. Transactions browsing

- `/transactions` shows ledger with filtering by date range, type, and `reference_no` (type `SAC-` to see sacrament-related entries).

---

6) Components (atomic → composite)

- **Form primitives**: Input, Select, DatePicker (with time), MoneyInput, Textarea, Toggle, FormRow
- **Data primitives**: DataTable (TanStack Table), EmptyState, Badge, StatusPill
- **Feedback**: Toast, Spinner, ErrorBanner, ConfirmDialog
- **Domain composites**: ParishionerPicker (search + create inline), SacramentForm (baptism first), TransactionCard, CalendarCard
- **Layout**: AppShell (sidebar, topbar), PageHeader, FilterBar

  All components are accessible (labels, keyboard nav), responsive, and theme-friendly.

---

7) API client & error handling

- `` `: tiny wrapper around `` fetch` with JSON, auth header placeholder, and base URL from env.
- Map backend errors → `{ title, message, hint? }` for toasts.
- **React Query**: keys like `['sacraments', {page, filters}]`, `['transactions', {range}]`, `['calendar', {start,end}]`.
- Retry policy: small (e.g., 1–2 retries) for GET; no retry for POST/PATCH/DELETE.
- Idempotency: not needed client-side (server-side optional); prevent double submit with form state.

---

8) Validation & forms

- **Zod schemas** mirror create/update DTOs.
- Pre-fill time `10:00` & timezone `Asia/Manila` in the baptism form.
- Friendly error messages (e.g., "Please select a parishioner").

---

9) Calendar specifics

- Library: **@fullcalendar/react** with dayGrid & timeGrid plugins.
- Data adapter maps `CalendarEvent` → `{ id, title, start, end, extendedProps }`.
- Timezone handling: convert API ISO strings to `Asia/Manila` for display; show local time with suffix if user not in Manila (later).

---

10) Styling & theming

- Tailwind + shadcn/ui tokens.
- Soft shadows, rounded-2xl, readable font sizes.
- Dark mode (optional toggle).

---

11) Testing strategy

- **Unit**: Vitest + RTL for components (forms, table cells, cards).
- **Integration**: React Query flows with MSW mocking.
- **E2E**: Playwright script: create sacrament → confirm linked Tx + Calendar visible.
- **Smoke against real backend**: small Playwright fixture or PowerShell script (already available via continuity pack).

---

12) Accessibility & i18n

- Use semantic HTML, ARIA where appropriate, and keyboard traps avoided.
- Screen-reader-friendly labels and form hints.
- `en-PH` locale; date formatting via date-fns; copy kept simple.

---

13) Observability & quality gates (optional for MVP)

- Sentry (browser) for runtime errors.
- ESLint + Prettier in CI; TypeScript strict mode.
- Bundle analysis if needed.

---

14) Security (now and later)

- For MVP, no auth (local network). Abstract headers in `lib/api.ts` so adding JWT/session later is easy.
- Escape/encode user-provided content; avoid dangerouslySetInnerHTML.

---

15) Project structure (starter matches this)

```
/clearkeep-frontend-starter
  ├─ app/                 # Next.js app router pages
  ├─ components/          # UI building blocks
  ├─ lib/                 # api.ts, date.ts, utils/
  ├─ styles/              # globals.css, tailwind layers
  ├─ public/              # static assets
  ├─ tests/               # unit/integration
  └─ package.json         # scripts
```

---

16) Execution plan (3 short sprints)

Sprint 1 — Foundations & Sacraments (2–3 days)

-

Sprint 2 — Lists & Calendar (2–3 days)

-

Sprint 3 — Polish & hardening (2–3 days)

-

---

17) Definition of Done (MVP)

- ✅Create/Update/Delete sacrament works end-to-end; UI shows linked Tx & Calendar
- ✅Lists are filterable and fast; URL state encoded in query params
- ✅Clear error toasts; forms validated with Zod
- ✅Basic tests pass; lint/tsc clean

---

18) Risks & mitigations

- **Backend variance** (contract changes): lock a small contract mirror in `lib/types.ts` and integration tests to catch drift.
- **Timezone confusion**: centralize helpers in `lib/date.ts` and show timezone indicator in UI.
- **API latency**: use React Query caching and optimistic updates for edits.

---

19) Open questions (track here)

- Do we need staff logins before go-live? If yes, which provider?
- Calendar: should users drag to reschedule (would PATCH sacrament date/time)?
- Printing/exports for ledger or baptism certificates?

---

20) Next step right now

1. Ensure backend is running.
2. In `C:\ckchurch1\clearkeep-frontend-starter`: `npm i` → `npm run dev`.
3. We'll confirm the starter loads, then implement **/sacraments/new** first.

   In the new chat, we'll work step-by-step (one command at a time) and commit after each green milestone.