FRONTEND_HANDOFF_v1
Prime Tech Services – Frontend Implementation Specification

STATUS: APPROVED FOR IMPLEMENTATION
VERSION: v1
DATE: 24 January 2026
SCOPE: Next.js (App Router) frontend for Prime Tech Services, aligned to CODEX_HANDOFF_v1 backend contracts.

Purpose

This document defines the frontend scope, routes, UI behaviour, design system rules, integration rules, and Codex execution constraints. It must be followed exactly. The backend remains the source of truth for data and enforcement.

Table of Contents

1. Design Principles

The site must be simple, professional, and consistent. Design must prioritise clarity, speed, accessibility, and trust. No clutter, no decorative excess, and no inconsistent styles between pages.

• Mobile-first responsive layout; desktop enhancements only after mobile is correct.

• Consistent typography hierarchy across all pages.

• Consistent spacing scale; avoid ad-hoc padding/margins.

• Clear call-to-action (CTA) placement; primary CTA per section.

• All forms must have clear labels, inline validation, and accessible error states.

• Avoid unnecessary animations; prefer subtle transitions only (hover/focus).

2. Information Architecture (Routes)

Routes are defined below. Any additional route requires explicit approval. Public routes must not leak privileged information.

Public

• **/** — Homepage: value proposition, services overview, trust signals, CTA to booking.

• **/services** — Services catalogue: list of service cards; filter/search optional.

• **/services/[slug]** — Service detail: description, inclusions, price guidance, CTA to /book?service=slug.

• **/book** — Booking form: collects customer details and service selection; creates booking.

• **/track** — Booking tracking: accepts booking reference (BKG-...) + email; shows status timeline.

• **/articles** — Article index: cards listing articles with category and read time.

• **/articles/[slug]** — Article detail: content, metadata, canonical URL, related articles.

• **/contact** — Contact details and enquiry form (non-auth).

• **/legal/privacy** — Privacy policy.

• **/legal/terms** — Terms of service.

Account (Authenticated)

• **/account** — Account overview: profile basics, orders, licences, downloads.

• **/account/orders** — Order list with ORD/INV/RCP identifiers.

• **/account/orders/[ordId]** — Order detail: line items, invoice/receipt links, status.

• **/account/licences** — Licence list; status; activation count; offline grace info.

• **/account/downloads** — Available downloads for entitled releases; uses signed URLs.

• **/account/settings** — Change password, email preferences, security info.

Admin (Authenticated + Role)

• **/admin** — Admin dashboard: metrics tiles and quick links.

• **/admin/bookings** — Bookings table: filter by status, date; view details; update status.

• **/admin/orders** — Orders table: filter/search; view invoice/receipt links.

• **/admin/releases** — Releases list: upload metadata (no direct S3 public access).

• **/admin/users** — User lookup: view account status; disable/enable if supported by backend.

3. Shared UI Components

The following components must be implemented once and reused across the site:

- **LayoutShell** — Container widths, header/footer, consistent padding, and page title slot.
- **SiteHeader** — Logo, primary nav, account/admin links based on session.
- **SiteFooter** — Contact, legal links, social icons if provided, copyright.
- **Card** — Service/article/order cards: consistent border, padding, hover.
- **Button** — Variants: primary, secondary, ghost, destructive, link; consistent sizes.
- **Input / Select / Textarea** — Accessible labels, descriptions, and error rendering.
- **Alert** — Success, warning, error messages; used for API errors and confirmations.
- **Badge** — Status badges for booking/order/licence states.
- **Table** — Admin and account lists with sticky header optional.
- **Modal/Drawer** — For confirmations and details; must be keyboard accessible.
- **LoadingSkeleton** — For data fetch states; avoids layout shift.

4. Design System

Use a minimal, neutral palette and a clear typographic hierarchy. Implementation should be via Tailwind configuration and reusable components.

Typography

Define semantic styles and apply consistently: H1, H2, H3, body, small, caption. Avoid inline font sizes except within component definitions.

Spacing scale

Use a consistent spacing scale (e.g., 4/8/12/16/24/32). Avoid arbitrary values unless required for alignment.

Colour

Use a restrained set: background, surface, border, text, muted text, primary, primary-hover, error, success. Do not introduce additional colours without approval.

Elevation

Cards may use subtle shadow. Avoid heavy shadows. Prefer borders and whitespace.

5. Forms, Validation, and Error Handling

All forms must validate on the client (schema-based) and display server validation errors using the canonical error contract. Do not display raw stack traces or provider messages.

Client validation

Use Zod schemas that mirror API contracts. Validate on submit and on blur for critical fields (email, reference IDs).

Error mapping

All API errors must render via a single error presenter component that understands the canonical error shape (code, message, details). Show a human-readable message and optionally field-level errors.

Identifier validation

```
ORD: ^ORD-[0-9]{8}-[A-Z0-9]{4}$
INV: ^INV-[0-9]{4}-[0-9]{6}$
RCP: ^RCP-[0-9]{4}-[0-9]{6}$
LIC: ^LIC-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}$
BKG: ^BKG-[A-Z0-9]{8}$ (or project-defined exact pattern)
```

6. Authentication and Session Handling

Authentication is session-based using HTTP-only cookies. The frontend must never store tokens in localStorage.

• Use server actions or route handlers sparingly; prefer calling backend API endpoints from server components where appropriate.

• Session detection should be performed via backend session endpoint (e.g., /api/auth/me) and cached appropriately.

• Logout must call backend logout endpoint and then refresh UI state.

• Admin routes must be protected by role check; never rely solely on client-side gating.

7. Frontend ↔ Backend Integration Rules

The frontend must conform to backend API contracts. No ad-hoc payloads or undocumented query parameters.

• Base URL comes from NEXT_PUBLIC_API_BASE. No hard-coded localhost in production.

• All requests must send credentials (cookies).

• Use canonical error contract; treat non-2xx as structured error.

• Implement request timeouts and user-facing retry guidance for network failures.

• Never assume payment success on redirect; always verify via backend order/receipt endpoints.

8. Downloads and Signed URL Flow

Downloads must always use short-lived signed URLs. The frontend must not expose S3 keys beyond what is required.

Flow

• User navigates to /account/downloads.

• Frontend requests list of entitled releases (metadata only).

• On user click Download, frontend calls backend to mint a signed URL for that release objectKey.

• Frontend opens the returned URL (new tab or direct navigation) immediately; do not cache for reuse beyond TTL.

Rules

• Do not attempt to download if licence validation fails.

• Handle expiry errors by re-requesting a new signed URL.

• Display file size/version and release notes where provided by backend.

9. Admin UI Requirements

Admin UI must be functional, consistent, and safe. All admin actions must confirm intent and show audit-friendly outcomes.

• Bookings: list view, filter by status/date, detail view, status update workflow.

• Orders: list view, detail view, links to invoice and receipt PDFs (signed URLs).

• Releases: list view; create/edit metadata; upload is via backend-mediated process (no public S3).

• Users: lookup by email; show verification status; show role; disable/enable only if backend supports.

Admin authentication

Admin routes require session + role. If backend supports an admin header token for specific routes, it must be used only server-side and never exposed to the browser.

10. Environment Variables (Frontend)

```
Required:
- NEXT_PUBLIC_SITE_URL
- NEXT_PUBLIC_API_BASE

Optional (if used by backend/public pages):
- NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY (only if frontend hosts Stripe Elements; otherwise omit)

Rules:
- All NEXT_PUBLIC_* values are safe to expose.
- Never expose admin secrets or server-only credentials in the frontend.
```

11. SEO, Metadata, and Accessibility

• All public pages must have title + description metadata.

• Articles must have canonical URLs and structured metadata where applicable.

• Generate sitemap and robots rules consistent with deployment policy.

• All interactive elements must be keyboard accessible and have visible focus states.

• Images must include alt text; decorative images use empty alt.

12. Execution Rules for Codex

• Do not change backend contracts or reinterpret requirements.

• Do not introduce new routes beyond those listed without approval.

• Implement shared components first, then public routes, then account, then admin.

• All identifiers must remain uppercase and pattern-validated.

• No localStorage tokens; cookie sessions only.

• Prefer reusable components; avoid duplicated UI logic between pages.

13. Acceptance Criteria

- All listed routes exist and render correctly on mobile and desktop.
- Forms validate and show field-level and page-level errors correctly.
- Session-based authentication works; protected routes do not leak content.
- Downloads use signed URL flow and handle expiry gracefully.
- Admin tables support filtering and status updates with confirmations.
- No hard-coded localhost URLs in production builds.
- Lighthouse-style basics: no major layout shift, acceptable LCP, accessible labels.