JADAYU – Smart Ration Delivery: Full Documentation

This document consolidates the project’s overview, features, usage, routes, and public APIs (hooks, components, utilities, and integrations) with examples.

Overview

- Product: JADAYU – Smart Ration Delivery Service
- Audience: Developers and operators integrating or extending the app
- Stack: Vite, React 18, TypeScript, Tailwind CSS, shadcn-ui, React Router v6, TanStack Query, Supabase

Features

- Customers/Beneficiaries
  - View ration quota by card type and household size
  - Shop subsidized items, manage cart, place orders
  - Live delivery tracking map and order history
  - Profile management; upload Aadhaar and ration card documents
  - Personal QR for secure delivery verification
- Delivery Partners
  - View approved and assigned orders
  - Navigate to delivery address, scan customer QR, confirm delivery
  - Earnings summary (demo)
- Admin/Shop Owner
  - Inventory management for ration items (CRUD)
  - Review incoming orders; approve/reject; update statuses
  - Dashboard with stats (orders, revenue, customers, inventory value)

How to Use (Role-based)

- Customer
  1. Sign up/sign in (`/auth`)
  2. Complete profile and upload docs in `Profile`
  3. Check `Quota` , browse `Shop` , add items to cart
  4. Place order (identity verification dialog simulates checks)
  5. Track delivery in `Track` or see `History`
- Delivery Partner
  1. Sign in → `DeliveryDashboard`
  2. Accept orders, navigate, scan QR, confirm delivery
- Admin
  1. Sign in → `AdminDashboard`
  2. Manage inventory and approve/reject orders

Route Map

- `/auth` : authentication
- `/` : role router → `AdminDashboard` | `DeliveryDashboard` | `CustomerDashboard`

- `/shop`, `/quota`, `/track`, `/history`, `/cart`, `/payment`, `/profile`
- `/incoming-orders` (admin), `/beneficiaries` (admin), `/qr-scanner` (delivery demo)
- `/verify/:id`: delivery verification view

Hooks API

`useAuth` and `AuthProvider`

- State: `user`, `session`, `profile`, `loading`
- Actions: `signIn`, `signUp`, `signOut`, `refreshProfile`, `devSignIn`, `devSignOut`

Example

```
import { AuthProvider, useAuth } from "@/hooks/useAuth";

export function AppShell() {
  return (
    <AuthProvider>
      <MyRoutes />
    </AuthProvider>
  );
}

function ProfileButton() {
  const { user, signOut } = useAuth();
  if (!user) return null;
  return <button onClick={signOut}>Sign out</button>;
}
```

`useCart`

- State: `lines`, `totalItems`, `totalAmount`
- Actions: `add`, `remove`, `clear`

Example

```
import { CartProvider, useCart } from "@/hooks/useCart";

function Root() {
  return (
    <CartProvider>
      <Shop />
    </CartProvider>
  );
}

function AddToCart({ item }) {
  const { add } = useCart();
  return (
    <button onClick={() => add({ id: item.id, name: item.name, unit: item.unit,
price: item.price }, 1)}>
      Add
    </button>
```

```
  );
}
```

`useIsMobile`

```
import { useIsMobile } from "@/hooks/use-mobile";
const isMobile = useIsMobile();
```

`useToast` and `toast`

```
import { useToast } from "@/hooks/use-toast";
const { toast } = useToast();
toast({ title: 'Saved', description: 'Your changes were saved.' });
```

Components API

Layout and Navigation

- `MainLayout({ children })` : shell with header and mobile bottom nav
- `NavHeader({ userName?, userRole? })` : role-aware navigation
- `MobileSidebar({ userName?, userRole? })` : mobile sheet navigation

Shopping and Quota

- `RationItem({ id, name, price, image, available, unit, subsidized?, onAddToCart, onRemoveFromCart, cartQuantity? })`
- `QuotaCard({ quotaItems: { name, allocated, used, unit }[], cardNumber, validUntil })`
- `DocumentUpload({ userId, bucket, folder, label, accept?, currentUrl?, onUploaded? })`
- `VerificationDialog({ isOpen, onClose, beneficiary, onVerificationComplete })`

Examples

```
<RationItem id="rice" name="Rice" price={25.5} image={url} available={5} unit="kg"
  onAddToCart={(id,q)=>{}} onRemoveFromCart={(id,q)=>{}} cartQuantity={2} />

<QuotaCard quotaItems={[{ name:'Rice', allocated:10, used:3, unit:'kg'}]}
  cardNumber="XXXX-XXXX-1234" validUntil="Dec 2024" />

<DocumentUpload userId={uid} bucket="documents" folder="aadhaar" label="Upload
Aadhaar"
  onUploaded={(url)=>setUrl(url)} />

<VerificationDialog isOpen={open} onClose={()=>setOpen(false)} beneficiary={b}
  onVerificationComplete={()=>refetch()} />
```

UI Kit

- Import shadcn-ui components from `@/components/ui/...` (e.g., `Button`, `Card`, `Dialog`, `Tabs`, `Table`, `Progress`)

Utilities

`cn(...inputs)`

```
import { cn } from "@/lib/utils";
<div className={cn('p-4', isActive && 'bg-primary', 'text-foreground')} />
```

Integrations

Supabase

- Client: `@/integrations/supabase/client`
- Use for auth (`useAuth`), storage (`DocumentUpload`), and DB queries (orders, profiles, ration_items)
- RPCs referenced: `verify_ration_card`, `get_user_ration_quota`

```
import { supabase } from "@/integrations/supabase/client";
const { data, error } = await supabase.from('profiles').select('*');
```

Security

- Replace anon key and URL
- Set RLS policies for `profiles`, `orders`, `ration_items`

Firebase (optional)

- Scaffold: `@/integrations/firebase/client` → `getFirebase()` returns `{ app, auth, db, storage }`
- Env vars: `VITE_FIREBASE_*`

Migration tip

- Replace Supabase usage in `useAuth`, `DocumentUpload`, and DB calls with Firebase equivalents if migrating.