🎨 FRESH ROOTS - FRONTEND DEVELOPMENT ROADMAP

Project: Fresh Roots Mobile App
Platform: React Native (iOS & Android)
Target: Mauritius Fresh Vegetable Marketplace
Timeline: 2-3 Weeks
Backend API: Ready and Deployed

📱 UI/UX DESIGN SPECIFICATIONS

Design Reference

Based on uploaded UI mockup showing 3 key screens: 1. **Welcome/Splash Screen** - Hero image with vegetables, tagline, CTA button 2. **Product Listing** - Grid/list view with images, prices, add to cart 3. **Product Detail** - Large image, price, rating, description, quantity selector

Design System

Color Palette

```
const colors = {
  // Primary Colors (Green Theme - Fresh & Natural)
  primary: '#2D7A3E',        // Deep green (main brand color)
  primaryLight: '#4CAF50',   // Bright green (buttons, CTAs)
  primaryDark: '#1B5E20',    // Dark green (headers, emphasis)

  // Secondary Colors
  secondary: '#FFA726',       // Orange (accents, badges)
  secondaryLight: '#FFB74D',

  // Neutral Colors
  background: '#FFFFFF',      // White background
  surface: '#F5F5F5',        // Light gray (cards, containers)
  text: '#212121',           // Dark gray (primary text)
  textSecondary: '#757575',  // Medium gray (secondary text)
  textLight: '#BDBDBD',      // Light gray (placeholders)

  // Status Colors
  success: '#4CAF50',        // Green (success messages)
  error: '#F44336',          // Red (errors)
  warning: '#FF9800',        // Orange (warnings)
  info: '#2196F3',           // Blue (info)

  // Border & Divider
  border: '#E0E0E0',
  divider: '#EEEEEE',
}
```

Typography

```
const typography = {
  // Headings
  h1: { fontSize: 32, fontWeight: '700', lineHeight: 40 },
  h2: { fontSize: 28, fontWeight: '600', lineHeight: 36 },
  h3: { fontSize: 24, fontWeight: '600', lineHeight: 32 },
  h4: { fontSize: 20, fontWeight: '600', lineHeight: 28 },
  h5: { fontSize: 18, fontWeight: '600', lineHeight: 24 },
  h6: { fontSize: 16, fontWeight: '600', lineHeight: 22 },

  // Body Text
  body1: { fontSize: 16, fontWeight: '400', lineHeight: 24 },
  body2: { fontSize: 14, fontWeight: '400', lineHeight: 20 },

  // Special
  caption: { fontSize: 12, fontWeight: '400', lineHeight: 16 },
  button: { fontSize: 16, fontWeight: '600', textTransform:
      'uppercase' },
  price: { fontSize: 20, fontWeight: '700', color: colors.primary },

  // Mauritian Touch
  greeting: { fontSize: 24, fontWeight: '500', fontStyle: 'italic'
      },
}
```

Spacing System

```
const spacing = {
  xs: 4,
  sm: 8,
  md: 16,
  lg: 24,
  xl: 32,
  xxl: 48,
}
```

Border Radius

```
const borderRadius = {
  small: 4,
  medium: 8,
  large: 12,
  xl: 16,
  round: 999, // Fully rounded (pills, badges)
}
```

UI Components Library

1. Buttons

```
// Primary Button (Green, full width)
<Button variant="primary" size="large" fullWidth>
  Get Started
</Button>

// Secondary Button (Outlined)
<Button variant="outlined" size="medium">
  View Details
</Button>

// Icon Button (Add to cart)
<IconButton icon="add-shopping-cart" color="primary" />

// Styles
Primary: bg-primary, white text, rounded-lg, shadow
Secondary: border-primary, primary text, rounded-lg
Icon: circular, 40x40, center icon
```

2. Product Cards

```
// Grid Card (Listing View)
<ProductCard>
  <Image source={productImage} />
  <Badge>Fresh</Badge>
  <Title>Fresh Broccoli</Title>
  <Price>Rs 80/kg</Price>
  <Rating value={4.5} />
  <AddButton />
</ProductCard>

// Styles:
- White background
- Rounded corners (12px)
- Shadow (elevation 2)
- Padding: 12px
- Image: rounded-top, aspect ratio 4:3
- Badge: top-right corner, orange bg, small text
```

3. Input Fields

```
<TextInput
  label="Email"
  placeholder="Enter your email"
  icon="email"
  type="email"
/>

// Styles:
- Border: 1px solid #E0E0E0
- Border radius: 8px
- Padding: 12px 16px
- Focus: border-primary, shadow-sm
- Error: border-error, error message below
```

4. Navigation

```
// Bottom Tab Bar
<TabBar>
  <Tab icon="home" label="Home" />
  <Tab icon="category" label="Categories" />
  <Tab icon="shopping-cart" label="Cart" badge={3} />
  <Tab icon="receipt" label="Orders" />
  <Tab icon="person" label="Profile" />
</TabBar>

// Styles:
- Height: 60px
- Background: white
- Shadow: top shadow
- Active: primary color
- Inactive: gray color
- Badge: red dot with count
```

📱 SCREEN SPECIFICATIONS

1. SPLASH/WELCOME SCREEN

Purpose: First impression, brand introduction, onboarding

Layout:

```
┌─────────────────────────┐
│                         │
│   🌿 FRESH ROOTS 🌿     │  <- Logo
│                         │
│   ╔════════════════╗     │
│   ║                ║     │
│   ║   [HERO IMG]   ║     │  <- Vegetables basket
│   ║                ║     │
│   ╚════════════════╝     │
│                          │
│  Explore the World of    │  <- Headline
│   Fresh Vegetables       │
│                          │
│  "Frais ek Kalite"       │  <- Mauritian tagline
│                          │
│  [  Get Started  ]       │  <- Primary CTA
│                          │
│     Already member?      │  <- Login link
│        Sign In           │
│                          │
└──────────────────────────┘
```

Elements: - Logo: “Fresh Roots” with leaf icon - Hero Image: Colorful vegetables basket (like mockup) - Headline: “Explore the World of Fresh Vegetables” - Tagline: “Frais ek Kalite” (Fresh and Quality in Mauritian Creole) - CTA Button: “Get Started” (green, large, rounded) - Secondary: “Sign In” text link

Animations: - Fade in logo (0.5s) - Slide up hero image (0.8s) - Fade in text (1s) - Pulse CTA button (subtle, continuous)

Navigation: - Get Started → Registration/Login - Sign In → Login Screen - Auto-navigate after 3s if logged in → Home

---

2. AUTHENTICATION SCREENS

2.1 Login Screen

```
┌──────────────────────────┐
│   ← Back    LOGIN        │  <- Header
│                          │
│   Welcome Back! 👋      │  <- Greeting
│                          │
│   [Email Input]          │
│   [Password Input]       │
│                          │
│   [ ] Remember Me        │  <- Checkbox
│        Forgot Password?  │  <- Link
│                          │
│   [    Sign In    ]      │  <- Primary button
│                          │
│   ───── OR ─────         │  <- Divider
│                          │
│   [  ● Facebook  ]      │  <- Social login (future)
│   [  ● Google    ]      │
│                          │
│   New here?              │
│   Create Account         │  <- Link to register
│                          │
└──────────────────────────┘
```

API Endpoint: `POST /api/auth/login`
Request:

```json
{
  "email": "user@example.com",
  "password": "password123"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "user": {
      "id": "uuid",
      "name": "John Doe",
      "email": "user@example.com",
      "role": "user"
    },
    "accessToken": "jwt-token",
    "refreshToken": "refresh-token"
  }
}
```

Validation: - Email: Required, valid email format - Password: Required, min 6 characters

Error Handling: - Invalid credentials: "Email or password is incorrect" - Network error: "Connection failed. Please try again." - Server error: "Something went wrong. Please try later."

2.2 Registration Screen

```
┌──────────────────────────┐
│   ← Back   REGISTER      │
│                          │
│   Create Account ⚘      │
│                          │
│   [Full Name]            │
│   [Phone Number]         │
│    [Email]                 │
│    [Password]              │
│    [Confirm Password]      │
│                            │
│    [ ] I agree to Terms    │
│                            │
│    [  Create Account ]     │
│                            │
│    Already have account?   │
│         Sign In            │
│                            │
└────────────────────────────┘
```

API Endpoint: `POST /api/auth/register`
Request:

```json
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "phone": "+230 5123 4567"
}
```

---

3. HOME SCREEN (Main Dashboard)

```
┌────────────────────────────┐
│  ☰  Fresh Roots   🛒3      │  <- Header with menu, cart
│                            │
│  Bonzour, John! 👋         │  <- Personalized greeting
│  [Search vegetables...]    │  <- Search bar
│                            │
│  ┌──────────────────┐      │  <- Hero Banner/Promo
│  │ 🥕 20% OFF!      │      │
│  │ Fresh Broccoli   │      │
│  └──────────────────┘      │
│                            │
│  Categories  [See All]     │  <- Section header
│  ┌───┐  ┌───┐  ┌───┐      │
│  │🥬 │  │🍎 │  │🌿 │      │  <- Category chips
│  │Veg│  │Frt│  │Hrb│      │
│  └───┘  └───┘  └───┘      │
│                            │
│  Fresh Arrivals            │  <- Section
│  ┌──────┐ ┌──────┐        │
│  │      │ │      │        │  <- Product cards
│  │  🥦  │ │  🥬  │        │
│  │$80   │ │$45   │        │
│  │ [+]  │ │ [+]  │        │
│   └──────┘ └──────┘         │
│                             │
│  Popular Items              │
│   ┌──────┐ ┌──────┐         │
│   │      │ │      │         │
│   └──────┘ └──────┘         │
│                             │
└─────────────────────────────┘
│  [🏠] [🎁] [🛒] [🛍] [👤]  │  <- Bottom nav
└─────────────────────────────┘
```

Sections: 1. **Header** - Hamburger menu (drawer) - Logo/Title - Cart icon with badge (item count) - Notification bell

2. **Greeting**
   - "Bonzour, [Name]!" with wave emoji
   - Current location: "Moka, Mauritius" (small text)
3. **Search Bar**
   - Icon: magnifying glass
   - Placeholder: "Search vegetables, fruits…"
   - On tap: Navigate to Search Screen
4. **Promo Banner**
   - Carousel/swipeable
   - Image + text overlay
   - CTA: "Shop Now"
   - Auto-play (5s interval)
5. **Categories**
   - Horizontal scroll
   - Circular icons with labels
   - Icons: Vegetables, Fruits, Herbs, Root Vegetables, Leafy Greens
   - On tap: Filter products by category
6. **Product Sections**
   - "Fresh Arrivals" (newest 6 products)
   - "Popular Items" (most ordered)
   - "Low Stock" (urgent)
   - Grid: 2 columns on phone, 3+ on tablet

API Endpoints: - Categories: `GET /api/categories` - Fresh Arrivals: `GET /api/listings?sortBy=created_at&order=desc&limit=6` - Popular: `GET /api/listings?sortBy=order_count&order=desc&limit=6`

---

4. PRODUCT LISTING SCREEN

```
┌─────────────────────────────┐
│  ← Vegetables     🔍 🛒    │  <- Header
│                             │
│  [Filter ▼] [Sort ▼]        │  <- Filters
│                             │
│  ┌──────────┐ ┌──────────┐│
│  │ 🥦       │ │ 🥬       ││  <- Product grid
│  │Broccoli│ │Lettuce  ││
│  │Rs 80/kg│ │Rs 45/kg ││
│  │⭐4.5    │ │⭐4.8    ││
│  │  [+]   │ │  [+]    ││
│  └──────────┘ └──────────┘│
│  ┌──────────┐ ┌──────────┐│
│  │ 🥕       │ │ 🍅       ││
│  │Carrots │ │Tomatoes ││
│  │Rs 60/kg│ │Rs 70/kg ││
│  │⭐4.7    │ │⭐4.6    ││
│  │  [+]   │ │  [+]    ││
│  └──────────┘ └──────────┘│
│                          │
│  [Load More]             │  <- Pagination
│                          │
└──────────────────────────┘
```

Features: 1. **Filter Options:** - Category (Vegetables, Fruits, Herbs, etc.) - Price range (slider) - Location (Moka, Flacq, etc.) - Availability (In Stock, Low Stock) - Tags (Organic, Local, Fresh)

2. **Sort Options:**
    - Price: Low to High
    - Price: High to Low
    - Newest First
    - Most Popular
    - Rating: High to Low
3. **Product Card (Grid):**
    - Image (aspect 1:1)
    - Badge: "Fresh", "Low Stock", "New"
    - Title (2 lines max)
    - Price (bold, large)
    - Unit (/kg, /piece)
    - Rating (stars + count)
    - Add button (+ icon, primary color)
4. **Actions:**
    - Tap card: View product details
    - Tap add button: Add to cart (with quantity selector popup)
    - Pull to refresh
    - Infinite scroll / Load more button

API Endpoint: `GET /api/listings`
Query Params:

```
page=1
limit=20
search=broccoli
```

```
category=vegetables
minPrice=0
maxPrice=200
location=Moka
sortBy=price
order=asc
```

---

5. PRODUCT DETAIL SCREEN

```
┌──────────────────────────┐
│  ← Back         ♡ Share │  <- Header
│                          │
│  ┌──────────────────┐    │
│  │                  │    │  <- Image gallery
│  │   🥦 BROCCOLI    │    │  (swipeable)
│  │                  │    │
│  │  ● ○ ○           │    │  <- Indicators
│  └──────────────────┘    │
│                          │
│  Fresh Broccoli          │  <- Title (h3)
│  ⭐ 4.5 (127 reviews)    │  <- Rating
│                          │
│  Rs 80.00 / kg           │  <- Price (large, bold)
│  🏷 Save 20%             │  <- Discount badge
│                          │
│  📍 Available in Moka    │  <- Location
│  📦 Stock: 45 kg         │  <- Stock info
│                          │
│  Description             │  <- Section
│  Fresh local broccoli,   │
│  organically grown in    │
│  Mauritius. Rich in...   │
│                          │
│  Quantity                │  <- Quantity selector
│  [ - ] 1 kg [ + ]        │
│                          │
│  [ Add to Cart ]         │  <- Primary CTA
│  [ Express Interest ]    │  <- Secondary CTA
│                          │
│  Similar Products        │  <- Recommendations
│  ┌─────┐ ┌─────┐         │
│  │     │ │     │         │
│  └─────┘ └─────┘         │
│                          │
└──────────────────────────┘
```

Elements: 1. **Image Gallery** - Swipeable carousel - Zoom on tap - Indicators (dots) - Full screen mode

2. **Product Info**
   - Title (h3, bold)
   - Rating (stars, review count)
   - Price (extra large, green)
   - Discount badge (if applicable)
   - Location badge
   - Stock indicator (green if >10, orange if ≤10, red if 0)
3. **Description**
   - Expandable (“Read More” if >3 lines)
   - Tags: #organic #local #fresh
4. **Quantity Selector**
   - Minus button
   - Number input (editable)
   - Plus button
   - Unit display (/kg, /piece)
5. **Action Buttons**
   - **Add to Cart** (Primary, green, full width)
     - Shows loading state
     - Success animation
     - Updates cart badge
   - **Express Interest** (Secondary, outlined)
     - Opens modal with message input
     - Sends to admin
6. **Tabs/Sections (Optional):**
   - Overview (default)
   - Reviews (star ratings, user comments)
   - Seller Info (name, rating, contact)

API Endpoints: - Get Product: `GET /api/listings/:id` - Add to Cart: `POST /api/cart/add` (local state + backend sync) - Express Interest: `POST /api/interest`

PostHog Event:

```
posthog.capture('product_viewed', {
  product_id: productId,
  product_name: productTitle,
  price: productPrice,
  category: productCategory
});
```

---

6. CART SCREEN

```
┌──────────────────────────┐
│  My Cart        Clear    │  <- Header
│                          │
│  ┌──────────────────┐    │
│  │ [img] Broccoli   │    │  <- Cart item
│  │       Rs 80/kg   │    │
│  │       [-] 2 [+]  │    │  <- Qty selector
│  │       Rs 160  🗑│   │  <- Subtotal, delete
│  └──────────────────┘    │
│  ┌──────────────────┐    │
│  │ [img] Lettuce    │    │
│  │       Rs 45/kg   │    │
│  │       [-] 1 [+]  │    │
│  │       Rs 45   🗑│   │
│  └──────────────────┘    │
│                          │
│  [+ Add More Items]      │  <- CTA to listings
│                          │
│  Order Summary           │  <- Section
│  ┌──────────────────┐    │
│  │ Subtotal  Rs 205 │    │
│  │ Delivery  Rs 50  │    │
│  │ ───────────────  │    │
│  │ Total     Rs 255 │    │  <- Bold, large
│  └──────────────────┘    │
│                          │
│  [ Proceed to Order ]    │  <- Primary CTA
│                          │
└──────────────────────────┘
```

Features: 1. **Cart Items** - Product image (small, rounded) - Title - Price per unit - Quantity selector (inline) - Subtotal - Delete icon

2. **Actions:**
    - Update quantity (debounced API call)
    - Remove item (with confirmation)
    - Clear cart (with confirmation)
    - Add more items (navigate to listings)
3. **Order Summary**
    - Subtotal (sum of items)
    - Delivery fee (fixed Rs 50 or calculated)
    - Total (bold, highlighted)
4. **Empty State**
    - Icon (empty cart)
    - Message: "Your cart is empty"
    - CTA: "Start Shopping"

State Management: - Local state (React Context or Redux) - Sync with backend on order creation - Persist to AsyncStorage

Navigation: - Proceed to Order → Order Confirmation Screen

7. ORDER CONFIRMATION SCREEN

```
┌──────────────────────────┐
│  ← Back   Confirm Order │
│                          │
│  Delivery Information    │  <- Section
│  ┌──────────────────┐    │
│  │ John Doe         │    │
│  │ +230 5123 4567   │    │
│  │ 123 Royal Road   │    │
│  │ Moka, Mauritius  │    │
│  │         [Edit]   │    │
│  └──────────────────┘    │
│                          │
│  Payment Method          │  <- Section
│  ( ) Cash on Delivery    │  <- Radio button
│  ( ) MIPS/Juice          │  <- Disabled (future)
│                          │
│  Order Items (3)         │  <- Section
│  ┌──────────────────┐    │
│  │ Broccoli  x2     │    │
│  │         Rs 160   │    │
│  └──────────────────┘    │
│  ┌──────────────────┐    │
│  │ Lettuce   x1     │    │
│  │         Rs 45    │    │
│  └──────────────────┘    │
│                          │
│  [ ] I agree to T&C      │  <- Checkbox
│                          │
│  Total: Rs 255           │  <- Large, bold
│                          │
│  [ Place Order ]         │  <- Primary CTA
│                          │
└──────────────────────────┘
```

API Endpoint: `POST /api/orders`
Request:

```
{
  "items": [
    { "listing_id": "uuid1", "quantity": 2 },
    { "listing_id": "uuid2", "quantity": 1 }
  ],
  "delivery_address": "123 Royal Road, Moka",
  "phone": "+230 5123 4567",
  "payment_method": "cash_on_delivery",
  "notes": "Please call before delivery"
}
```

Success Response:

```
{
  "success": true,
  "data": {
    "order_number": "FR20260125001",
    "status": "pending",
    "estimated_delivery": "2026-01-27"
  }
}
```

Success Flow: 1. Show loading spinner 2. Create order (API call) 3. Clear cart 4. Navigate to Success Screen 5. Send PostHog event

---

8. ORDER SUCCESS SCREEN

```
┌──────────────────────────┐
│                          │
│       ✅                 │  <- Success icon
│                          │
│  Order Placed!           │  <- Title
│                          │
│  Mersi! Your order has   │  <- Message
│  been received and will  │  (Mauritian touch)
│  be processed soon.      │
│                          │
│  Order #FR20260125001    │  <- Order number
│                          │
│  Estimated Delivery      │
│  Jan 27, 2026            │
│                          │
│  [ View Order Details ]  │  <- Secondary button
│  [ Continue Shopping ]   │  <- Primary button
│                          │
└──────────────────────────┘
```

PostHog Event:

```
posthog.capture('order_placed', {
  order_id: orderId,
  order_number: orderNumber,
  total_amount: totalAmount,
  items_count: itemsCount,
  payment_method: 'cash_on_delivery'
});
```

---

9. ORDERS SCREEN (Order History)

```
┌──────────────────────────┐
│  My Orders      🔍       │  <- Header
│                          │
│  [All] [Pending] [...]   │  <- Filter tabs
│                          │
│   ┌──────────────────┐   │
│   │ Order #FR001     │   │  <- Order card
│   │ Jan 25, 2026     │   │
│   │                  │   │
│   │ 3 items          │   │
│   │ Rs 255           │   │
│   │                  │   │
│   │ 🟠 Pending       │   │  <- Status badge
│   │         [View]   │   │
│   └──────────────────┘   │
│   ┌──────────────────┐   │
│   │ Order #FR002     │   │
│   │ Jan 20, 2026     │   │
│   │ 2 items  Rs 180  │   │
│   │ ✅ Delivered     │   │
│   │         [View]   │   │
│   └──────────────────┘   │
│                          │
└──────────────────────────┘
```

Order Card: - Order number - Date - Items count - Total amount - Status badge (color-coded) - View button

Status Colors: - Pending: Orange - Payment Confirmed: Blue - Approved: Green - Rejected: Red - Delivered: Dark Green

API Endpoint: `GET /api/orders/my-orders`

---

10. ORDER DETAIL SCREEN

```
┌──────────────────────────┐
│  ← Orders   Order #001   │
│                          │
│  Status                  │
│   ┌──────────────────┐   │
│   │ 🟠 Pending       │   │  <- Status indicator
│   │ Waiting for      │   │
│   │ admin approval   │   │
│   └──────────────────┘   │
│                          │
│  Timeline                │  <- Order tracking
│  ✅ Order Placed         │
│  │  Jan 25, 10:30 AM   │
│  ⏳ Payment Confirmed    │
│  │  Pending...         │
│  ○  Approved            │
│  ○  Out for Delivery    │
│  ○  Delivered           │
│                         │
│  Order Items            │
│  ┌──────────────────┐   │
│  │ [img] Broccoli   │   │
│  │       x2         │   │
│  │       Rs 160     │   │
│  └──────────────────┘   │
│                         │
│  Delivery Address       │
│  123 Royal Road, Moka   │
│  +230 5123 4567         │
│                         │
│  Payment                │
│  Cash on Delivery       │
│  Total: Rs 255          │
│                         │
│  [Cancel Order]         │  <- If pending
│  [Reorder]              │  <- If delivered
│                         │
└─────────────────────────┘
```

API Endpoint: `GET /api/orders/:id`

---

11. PROFILE SCREEN

```
┌─────────────────────────┐
│  Profile                │
│                         │
│      [👤]               │  <- Avatar
│   John Doe              │
│   john@example.com      │
│   [Edit Profile]        │
│                         │
│  ┌──────────────────┐   │
│  │ 👤 My Account    │   │  <- Menu items
│  │ 📦 My Orders     │   │
│  │ ❤  Favorites     │   │
│  │ 📍 Addresses     │   │
│  │ 🔔 Notifications │   │
│  │ 💳 Payment       │   │
│  │ ℹ️ About          │   │
│  │ 🆘 Help           │   │
│  │ 🚪 Logout         │   │
│  └───────────────────┘   │
│                          │
└──────────────────────────┘
```

---

12. SEARCH SCREEN

```
┌──────────────────────────┐
│  ← [Search products...]  │  <- Search input
│                          │
│  Recent Searches         │  <- Section
│  • Broccoli        ×     │
│  • Organic tomatoes ×    │
│                          │
│  Popular Searches        │
│  🥬 Lettuce              │
│  🥕 Carrots              │
│  🍅 Tomatoes             │
│                          │
│  ─ Results (15) ──       │  <- After search
│  ┌─────────┐ ┌─────────┐│
│  │ Product │ │ Product ││
│  └─────────┘ └─────────┘│
│                          │
└──────────────────────────┘
```

Features: - Real-time search (debounced) - Search history (AsyncStorage) - Popular searches (API) - Auto-suggestions

API: `GET /api/listings?search=query`

---

13. CATEGORIES SCREEN

```
┌──────────────────────────┐
│  Categories              │
│                          │
│  ┌─────────────────┐     │
│  │ 🥬               │     │  <- Category card
│  │ Vegetables      │     │  (large, with image)
│  │ 25 items        │     │
│  └─────────────────┘     │
│  ┌─────────────────┐     │
│  │ 🍎               │     │
│  │ Fruits          │     │
│  │ 12 items        │     │
│   └──────────────────┘    │
│   ┌──────────────────┐    │
│   │ 🌿               │    │
│   │ Herbs            │    │
│   │ 8 items          │    │
│   └──────────────────┘    │
│                           │
└───────────────────────────┘
```

🛠 TECHNICAL IMPLEMENTATION

Tech Stack

```
// Core
- React Native (latest)
- TypeScript
- Expo (managed workflow)

// Navigation
- React Navigation 6
- Bottom Tabs
- Stack Navigator

// State Management
- React Context API (for simple state)
- Redux Toolkit (if complex state needed)
- React Query (API caching)

// UI Library
- React Native Paper (Material Design)
- OR React Native Elements
- Custom components

// Forms
- React Hook Form
- Yup (validation)

// API
- Axios
- React Query

// Storage
- AsyncStorage (cart, auth token)

// Analytics
- PostHog React Native SDK
```

```
// Images
- React Native Fast Image
- Expo Image Picker

// Notifications
- Expo Notifications

// Utilities
- date-fns (date formatting)
- lodash (utilities)
```

Folder Structure

```
/src
  /api
    api.ts              # Axios instance
    auth.ts             # Auth endpoints
    products.ts         # Product endpoints
    orders.ts           # Order endpoints
  /components
    /common
      Button.tsx
      Input.tsx
      Card.tsx
      LoadingSpinner.tsx
    /product
      ProductCard.tsx
      ProductList.tsx
      QuantitySelector.tsx
    /cart
      CartItem.tsx
      CartSummary.tsx
  /screens
    /auth
      LoginScreen.tsx
      RegisterScreen.tsx
    /home
      HomeScreen.tsx
    /product
      ProductListScreen.tsx
      ProductDetailScreen.tsx
    /cart
      CartScreen.tsx
      OrderConfirmationScreen.tsx
    /orders
      OrdersScreen.tsx
      OrderDetailScreen.tsx
    /profile
      ProfileScreen.tsx
  /navigation
    AppNavigator.tsx
    AuthNavigator.tsx
  /context
    AuthContext.tsx
    CartContext.tsx
  /hooks
    useAuth.ts
    useCart.ts
    useProducts.ts
  /utils
    formatters.ts       # Price, date formatting
    validators.ts       # Form validation
    constants.ts        # API URLs, colors, etc.
  /types
    index.ts            # TypeScript types
  /theme
    colors.ts
    typography.ts
    spacing.ts
```

API Integration

```
// src/api/api.ts
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-
        storage';

const API_BASE_URL = 'https://your-deployed-backend.abacusai.app';

const api = axios.create({
  baseURL: API_BASE_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Request interceptor - Add auth token
api.interceptors.request.use(
  async (config) => {
    const token = await AsyncStorage.getItem('accessToken');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

// Response interceptor - Handle refresh token
api.interceptors.response.use(
  (response) => response,
  async (error) => {
    if (error.response?.status === 401) {
      // Refresh token logic
      const refreshToken = await
        AsyncStorage.getItem('refreshToken');
      // ... refresh token flow
    }
    return Promise.reject(error);
  }
);

export default api;
```

PostHog Integration

```
// src/utils/analytics.ts
import PostHog from 'posthog-react-native';

export const posthog = new PostHog(
  'phx_wGErHN8tuYOcs8hiJsSS0zdBREjDqtBYTKApQ8i7PuGzlcR',
  { host: 'https://app.posthog.com' }
);

// Track events
export const trackEvent = (eventName: string, properties?: object)
        => {
  posthog.capture(eventName, properties);
};

// Identify user
export const identifyUser = (userId: string, userProps: object) => {
  posthog.identify(userId, userProps);
};

// Usage in components:
// trackEvent('product_viewed', { product_id: '123', product_name:
        'Broccoli' });
```

Cart State Management

```
// src/context/CartContext.tsx
import React, { createContext, useContext, useState, useEffect }
        from 'react';
```

```
import AsyncStorage from '@react-native-async-storage/async-
        storage';

interface CartItem {
  id: string;
  title: string;
  price: number;
  quantity: number;
  image: string;
}

interface CartContextType {
  items: CartItem[];
  addItem: (item: CartItem) => void;
  removeItem: (id: string) => void;
  updateQuantity: (id: string, quantity: number) => void;
  clearCart: () => void;
  totalAmount: number;
  itemCount: number;
}

const CartContext = createContext<CartContextType | undefined>
        (undefined);

export const CartProvider: React.FC = ({ children }) => {
  const [items, setItems] = useState<CartItem[]>([]);

  // Load cart from AsyncStorage on mount
  useEffect(() => {
    loadCart();
  }, []);

  // Save cart to AsyncStorage on change
  useEffect(() => {
    saveCart();
  }, [items]);

  const loadCart = async () => {
    const savedCart = await AsyncStorage.getItem('cart');
    if (savedCart) {
      setItems(JSON.parse(savedCart));
    }
  };

  const saveCart = async () => {
    await AsyncStorage.setItem('cart', JSON.stringify(items));
  };

  const addItem = (item: CartItem) => {
    const existing = items.find(i => i.id === item.id);
    if (existing) {
      updateQuantity(item.id, existing.quantity + item.quantity);
    } else {
      setItems([...items, item]);
    }
    // Track event
    trackEvent('add_to_cart', { product_id: item.id, quantity:
        item.quantity });
  };

  const removeItem = (id: string) => {
    setItems(items.filter(item => item.id !== id));
  };

  const updateQuantity = (id: string, quantity: number) => {
    if (quantity <= 0) {
      removeItem(id);
    } else {
      setItems(items.map(item =>
        item.id === id ? { ...item, quantity } : item
      ));
    }
  };

  const clearCart = () => {
    setItems([]);
  };

  const totalAmount = items.reduce((sum, item) =>
    sum + (item.price * item.quantity), 0
  );

  const itemCount = items.reduce((sum, item) => sum + item.quantity,
      0);

  return (
    <CartContext.Provider value={{
      items,
      addItem,
      removeItem,
      updateQuantity,
      clearCart,
      totalAmount,
      itemCount
    }}>
      {children}
    </CartContext.Provider>
  );
```

```
};

export const useCart = () => {
  const context = useContext(CartContext);
  if (!context) {
    throw new Error('useCart must be used within CartProvider');
  }
  return context;
};
```

📋 DEVELOPMENT PHASES

Phase 1: Setup & Authentication (Week 1)

Duration: 3-4 days

Tasks: 1. ✅ Initialize Expo project with TypeScript 2. ✅ Setup folder structure 3. ✅ Configure navigation (Stack + Bottom Tabs) 4. ✅ Setup API client (Axios) 5. ✅ Implement theme (colors, typography) 6. ✅ Create reusable components (Button, Input, Card) 7. ✅ Build Splash/Welcome Screen 8. ✅ Build Login Screen 9. ✅ Build Registration Screen 10. ✅ Implement Auth Context 11. ✅ Implement token storage (AsyncStorage) 12. ✅ Test authentication flow

Deliverables: - User can register and login - Token stored securely - Navigation between auth screens

---

Phase 2: Product Browsing (Week 1-2)

Duration: 4-5 days

Tasks: 1. ✅ Build Home Screen - Greeting with Mauritian touch - Search bar - Categories (horizontal scroll) - Product sections (Fresh Arrivals, Popular) 2. ✅ Build Product List Screen - Grid layout (2 columns) - Filter & Sort - Pagination - Pull to refresh 3. ✅ Build Product Detail Screen - Image gallery - Product info - Quantity selector - Add to cart - Express interest 4. ✅ Build Categories Screen 5. ✅ Build Search Screen 6. ✅ Implement PostHog analytics - Track screen views - Track product views - Track searches

Deliverables: - User can browse products - View product details - Search and filter products

---

Phase 3: Cart & Orders (Week 2)

Duration: 3-4 days

Tasks: 1. ✅ Implement Cart Context 2. ✅ Build Cart Screen - List cart items - Update quantity - Remove items - Show total 3. ✅ Build Order Confirmation Screen - Delivery info - Payment method selection (Cash only for now) - Order summary 4. ✅ Build Order Success Screen 5. ✅ Implement order creation API 6. ✅ Track order events in PostHog

Deliverables: - User can add items to cart - User can place orders - Orders saved to backend

---

Phase 4: Order Management & Profile (Week 2-3)

Duration: 2-3 days

Tasks: 1. ✅ Build Orders Screen (Order History) 2. ✅ Build Order Detail Screen - Status tracking - Timeline - Order items 3. ✅ Build Profile Screen - User info - Menu items 4. ✅ Build Edit Profile Screen 5. ✅ Implement logout

Deliverables: - User can view order history - User can track order status - User can manage profile

---

Phase 5: Polish & Testing (Week 3)

Duration: 3-5 days

Tasks: 1. ✅ UI/UX polish - Animations - Loading states - Error handling - Empty states 2. ✅ Add Express Interest feature 3. ✅ Implement notifications 4. ✅ Performance optimization - Image optimization - API caching - Lazy loading 5. ✅ Testing - Manual testing on iOS - Manual testing on Android - Fix bugs 6. ✅ Documentation - User guide - README

Deliverables: - Polished, production-ready app - Tested on both platforms - Ready for deployment

---

🚀 DEPLOYMENT

Build for iOS

```
eas build --platform ios
```

Build for Android

```
eas build --platform android
```

Submit to App Store

```
eas submit --platform ios
```

Submit to Play Store

```
eas submit --platform android
```

📊 SUCCESS METRICS

PostHog Events to Track

```
// User Events
- user_registered
- user_logged_in
- user_logged_out

// Product Events
- product_viewed
- product_search
- category_selected
- filter_applied
- sort_applied

// Cart Events
- add_to_cart
- remove_from_cart
- cart_viewed
- checkout_started

// Order Events
- order_placed
- order_viewed
- interest_expressed

// Engagement
- screen_view
- button_clicked
- share_product
```

Key Performance Indicators (KPIs)

1. **User Acquisition**
   - New registrations per day
   - User growth rate
2. **Engagement**
   - Daily Active Users (DAU)
   - Session duration

    - Screens per session
3. **Conversion**
    - Cart abandonment rate
    - Order completion rate
    - Average order value
4. **Retention**
    - Day 1, 7, 30 retention
    - Repeat purchase rate

✅ CHECKLIST FOR FRONTEND DEVELOPER

Before Starting

- ☐ Review this roadmap completely
- ☐ Review FRESH_ROOTS_DEVELOPER_GUIDE.md
- ☐ Review FRESH_ROOTS_BACKEND_API_DOCUMENTATION.md
- ☐ Test backend API endpoints (use Postman/Thunder Client)
- ☐ Get deployed backend URL from user
- ☐ Setup Expo account (yashhb92@gmail.com)
- ☐ Setup PostHog account access

During Development

- ☐ Follow design specifications exactly
- ☐ Use TypeScript for type safety
- ☐ Implement proper error handling
- ☐ Add loading states for all async operations
- ☐ Track all events in PostHog
- ☐ Test on both iOS and Android
- ☐ Handle offline scenarios
- ☐ Implement proper form validation
- ☐ Add empty states for all screens
- ☐ Add confirmation dialogs for destructive actions

Before Deployment

- ☐ Test all user flows end-to-end
- ☐ Test with real backend API (deployed URL)
- ☐ Verify PostHog events are firing
- ☐ Test on physical devices (not just simulator)
- ☐ Check app performance (no lag)
- ☐ Verify images load quickly
- ☐ Test offline mode
- ☐ Check API error handling

- ☐ Review app permissions
- ☐ Update app.json with correct metadata

🎯 FINAL NOTES

Mauritian Cultural Touches

Integrate these throughout the app:

1. **Language:**
   - “Bonzour” (Hello) in greetings
   - “Mersi” (Thank you) in confirmations
   - “Frais ek Kalite” (Fresh and Quality) as tagline
2. **Products:**
   - Use Mauritian vegetable names (Bredes, Lalo, Pipangaille, etc.)
   - Include local varieties
3. **Locations:**
   - Moka, Flacq, Port Louis, Curepipe, etc.
4. **Currency:**
   - Rs (Mauritian Rupee)
   - Format: Rs 80.00 or Rs 80

Design Principles

1. **Mobile-First:** Design for small screens first
2. **Touch-Friendly:** Large tap targets (min 44x44px)
3. **Fast:** Optimize images, lazy load, cache API calls
4. **Accessible:** High contrast, readable fonts, alt text
5. **Delightful:** Smooth animations, micro-interactions

Quality Standards

- **Code:** Clean, well-documented, TypeScript
- **UI:** Pixel-perfect to mockups
- **UX:** Intuitive, no confusion
- **Performance:** 60fps, fast load times
- **Reliability:** Handle errors gracefully

Backend is ready. API is deployed. Now build an amazing mobile app! 🚀

Questions? Refer to: - FRESH_ROOTS_DEVELOPER_GUIDE.md - FRESH_ROOTS_BACKEND_API_DOCUMENTATION.md - FRESH_ROOTS_HANDOFF_STATUS.md