THE PETAL POUCHES

Complete Technical Architecture & Implementation Guide

E-Commerce Platform for Premium Personalized Gifts

Document Version: 1.0
Date: October 2025
Classification: Technical Implementation Guide
Target Audience: Development Team, Technical Stakeholders, Project Managers

---

EXECUTIVE SUMMARY

Project Overview

The Petal Pouches is a premium e-commerce platform specializing in personalized gift items with an emphasis on 3D-first, high-performance user experiences. This document provides a comprehensive technical blueprint for building a scalable, modern web application using React, Tailwind CSS, and cloud-native services while maintaining cost-effectiveness through strategic use of free-tier services.

Core Technology Stack

- **Frontend Framework:** React.js with Next.js (Server-Side Rendering)
- **Styling:** Tailwind CSS
- **3D Rendering:** Three.js via react-three-fiber
- **Database:** Supabase (PostgreSQL) or Firebase
- **Hosting:** Vercel/Netlify (Free Tier)
- **Shipping Integration:** Shiprocket/Delhivery/Blue Dart APIs
- **Payment Processing:** Razorpay/Stripe

Key Business Objectives

1. Deliver immersive 3D product visualization experiences
2. Provide accurate pin code-based delivery estimates
3. Enable seamless personalization and gift customization

4. Maintain sub-3 second page load times
5. Operate within free-tier limitations during initial launch

---

1. FRONTEND ARCHITECTURE

1.1 Core Technology Stack

Base Framework Selection

- **React.js 18+** - Component-based User Interface library for building reusable and interactive components
- **Next.js 14+** - Meta-framework providing:
  - Server-Side Rendering (SSR) for Search Engine Optimization
  - Static Site Generation (SSG) capabilities
  - Built-in image optimization
  - API routes for serverless functions
  - Automatic code splitting

Styling Architecture

- **Tailwind CSS 3.0+** - Utility-first CSS framework
  - Rapid responsive design implementation
  - Design system consistency
  - Minimal CSS bundle size
  - JIT (Just-In-Time) compilation

TypeScript Integration

- **TypeScript 5.0+** - Type-safe JavaScript
  - Enhanced code maintainability
  - Better IDE support and autocomplete
  - Compile-time error detection
  - Improved refactoring capabilities

1.2 3D Rendering & Animation Stack

Primary 3D Solution

react-three-fiber (Three.js for React)

- Most flexible and production-ready for real 3D interactions
- Required packages:
  - `@react-three/fiber` - Core React renderer
  - `@react-three/drei` - Helper components and utilities
  - `@react-three/postprocessing` - Visual effects

Alternative 3D Solutions

Spline Embeds

- Visual authoring for non-developers
- Quick prototyping capabilities
- Export to React components
- Ideal for designer-driven workflows

Animation Libraries

- **Framer Motion** - Complex UI animations and transitions
- **Lottie React** - Lightweight vector animations
- **GSAP** (GreenSock Animation Platform) - Advanced timeline animations

1.3 State Management Architecture

```
┌─────────────────────────────────────┐
│        Global State (Zustand)       │
├─────────────────────────────────────┤
│ • User Authentication State         │
│ • Shopping Cart Data                │
│ • Product Catalog Cache             │
│ • UI Theme Preferences              │
└─────────────────────────────────────┘
              │
              ▼
┌─────────────────────────────────────┐
│    Component Local State (useState) │
├─────────────────────────────────────┤
│ • Form Inputs                       │
│  • UI Toggle States          │
│  • Animation States          │
└──────────────────────────────────────────────┘
```

1.4 Performance Optimization Strategy

Code Splitting Implementation

- Dynamic imports for route-based splitting
- Component-level lazy loading
- Suspense boundaries for graceful loading states

Asset Optimization

- Next.js Image Component with automatic WebP conversion
- Lazy loading with Intersection Observer
- Progressive image loading (blur-up technique)

3D Performance Checklist

- [ ] Use lightweight GLTF/GLB formats
- [ ] Apply Draco compression to models
- [ ] Implement texture atlasing
- [ ] Reduce texture resolution for mobile
- [ ] Use instancing for repeated geometry
- [ ] Implement Level of Detail (LOD) systems
- [ ] Limit draw calls through geometry merging
- [ ] Provide mobile fallbacks (2D/video)

---

2. BACKEND ARCHITECTURE

2.1 Serverless Function Architecture

```
┌─────────────────────────────────────────────────┐
│          Client Request          │
└────────────────────┬────────────────────────────┘
       │
       ▼
┌─────────────────────────────────────────────────┐
│      API Gateway (Vercel/Netlify)      │
```

```
└─────────────────────┬─────────────────────┘
         │
         ▼
┌───────────────────────────────────────────┐
│     Serverless Functions                  │
├───────────────────────────────────────────┤
│ • /api/quote-shipping                     │
│ • /api/create-order                       │
│ • /api/webhook/payment                    │
│ • /api/create-shipment                    │
│ • /api/track/:awb                         │
│ • /api/products                           │
└─────────────────────┬─────────────────────┘
         │
         ▼
┌───────────────────────────────────────────┐
│   External Services & Database            │
├───────────────────────────────────────────┤
│ • Supabase/Firebase                       │
│ • Shipping APIs (Shiprocket)              │
│ • Payment Gateway (Razorpay)              │
└───────────────────────────────────────────┘
```

2.2 API Endpoint Specifications

POST /api/quote-shipping

Purpose: Calculate shipping rates and delivery estimates

Request Body:

json

```json
{
  "origin_pincode": "110001",
  "destination_pincode": "400001",
  "weight_grams": 500,
  "dimensions": {
    "length": 20,
    "breadth": 15,
    "height": 10
  }
}
```

Response: Courier options with rates and estimated delivery dates

POST /api/create-order

Purpose: Initialize order and payment intent
Security: JWT authentication required
Integration: Razorpay/Stripe payment gateway

POST /api/webhook/payment

Purpose: Handle payment gateway callbacks
Security: Webhook signature verification
Actions: Update order status, trigger fulfillment

2.3 Authentication & Security

JWT (JSON Web Token) Implementation

- Access token expiry: 15 minutes
- Refresh token expiry: 7 days
- Secure HTTP-only cookies for token storage
- Token rotation on refresh

API Security Measures

- Rate limiting (100 requests/minute per IP)
- CORS (Cross-Origin Resource Sharing) policy enforcement
- Input validation and sanitization
- SQL injection prevention through parameterized queries
- XSS (Cross-Site Scripting) protection

---

3. DATABASE ARCHITECTURE

3.1 Database Selection Comparison

Feature	Supabase	Firebase	Recommendation
Database Type	PostgreSQL (Relational)	Firestore (NoSQL)	Supabase for complex queries
Free Tier Limits	500 MB database, 1 GB storage	1 GB stored, 20K writes/day	Firebase for simple data
Real-time Support	Yes (WebSocket)	Yes (Built-in)	Both excellent

Feature	Supabase	Firebase	Recommendation
File Storage	1 GB free	5 GB free	Firebase more generous
Authentication	Built-in with RLS	Comprehensive auth	Firebase more mature
SQL Support	Full SQL capabilities	No SQL	Supabase for reporting
Scalability	Vertical + Read replicas	Horizontal auto-scaling	Firebase easier to scale

Final Recommendation: Supabase for SQL flexibility and relational data management

3.2 Database Schema Design

sql

```sql
-- Users Table
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  email VARCHAR(255) UNIQUE NOT NULL,
  name VARCHAR(255) NOT NULL,
  phone VARCHAR(20),
  addresses JSONB,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Products Table
CREATE TABLE products (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  title VARCHAR(255) NOT NULL,
  description TEXT,
  price DECIMAL(10,2) NOT NULL,
  sku VARCHAR(100) UNIQUE,
  category_id UUID REFERENCES categories(id),
  images TEXT[],
  variants JSONB,
  weight_grams INTEGER,
  personalization_allowed BOOLEAN DEFAULT false,
  created_at TIMESTAMP DEFAULT NOW()
);

-- Product Variants Table
CREATE TABLE product_variants (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  product_id UUID REFERENCES products(id),
  sku VARCHAR(100) UNIQUE,
  price_delta DECIMAL(10,2) DEFAULT 0,
  attributes JSONB, -- {size: "M", color: "Red"}
  inventory_count INTEGER DEFAULT 0
);

-- Orders Table
CREATE TABLE orders (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES users(id),
  status VARCHAR(50) DEFAULT 'pending',
  subtotal DECIMAL(10,2),
  shipping_cost DECIMAL(10,2),
  total DECIMAL(10,2),
  shipping_address JSONB,
  payment_status VARCHAR(50),
  created_at TIMESTAMP DEFAULT NOW()
);

-- Order Items Table
CREATE TABLE order_items (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  order_id UUID REFERENCES orders(id),
  product_variant_id UUID REFERENCES product_variants(id),
  quantity INTEGER NOT NULL,
  personalization_json JSONB,
  price_at_purchase DECIMAL(10,2)
);

-- Shipments Table
CREATE TABLE shipments (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  order_id UUID REFERENCES orders(id),
  courier VARCHAR(100),
  awb VARCHAR(100) UNIQUE,
  status VARCHAR(50),
  estimated_delivery_date DATE,
  tracking_history JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);

-- Gift Previews Table
CREATE TABLE gift_previews (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES users(id),
  preview_url TEXT,
  personalization_json JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);
```

3.3 Row-Level Security (RLS) Policies

```
sql
```

```
-- Users can only read their own data
CREATE POLICY users_read_own ON users
  FOR SELECT USING (auth.uid() = id);

-- Users can update their own profile
CREATE POLICY users_update_own ON users
  FOR UPDATE USING (auth.uid() = id);

-- Products are publicly readable
CREATE POLICY products_public_read ON products
  FOR SELECT USING (true);

-- Orders are only visible to the owner
CREATE POLICY orders_read_own ON orders
  FOR SELECT USING (auth.uid() = user_id);
```

---

4. SHIPPING & LOGISTICS INTEGRATION

4.1 Shipping Provider Comparison

Provider	Coverage	API Features	Pricing Model	Integration Complexity
Shiprocket	29000+ pincodes, Multiple couriers	Rate calculation, AWB generation, Tracking, Returns	Pay per shipment	Low - Single API for multiple couriers
Delhivery	17500+ pincodes	Direct integration, Bulk processing	Volume-based contracts	Medium - Direct courier API
Blue Dart	35000+ pincodes	Premium service, Time-definite delivery	Premium pricing	High - Complex API structure

4.2 Shipping Integration Workflow

```
┌─────────────────────────────────────────┐
│     User Enters Pincode & Products     │
└──────────────────┬──────────────────────┘
                   │
                   ▼
┌─────────────────────────────────────────┐
│   Call /api/quote-shipping Endpoint    │
└──────────────────┬──────────────────────┘
                   │
```

```
         ▼
┌──────────────────────────────────────────────┐
│  Serverless Function Calls Shiprocket API    │
│      • GET /courier/serviceability           │
│      • POST /courier/freight-charge          │
└──────────────────────┬───────────────────────┘
         │
         ▼
┌──────────────────────────────────────────────┐
│   Process & Format Response                  │
│ • Available couriers                         │
│ • Delivery estimates (days)                  │
│ • Shipping costs                             │
└──────────────────────┬───────────────────────┘
         │
         ▼
┌──────────────────────────────────────────────┐
│  Display Options to User with EDD            │
│  (Estimated Delivery Date)                   │
└──────────────────────────────────────────────┘
```

4.3 Delivery Date Calculation Algorithm

javascript

```
function calculateDeliveryDate(courierData) {
  const today = new Date();
  const processingDays = 1; // Order processing
  const transitDays = courierData.estimated_delivery_days;
  const cutoffTime = 15; // 3 PM cutoff

  let deliveryDate = new Date(today);

  // Add processing time
  if (today.getHours() >= cutoffTime) {
    deliveryDate.setDate(deliveryDate.getDate() + 1);
  }

  // Add transit days (excluding Sundays)
  let daysAdded = 0;
  while (daysAdded < processingDays + transitDays) {
    deliveryDate.setDate(deliveryDate.getDate() + 1);
    if (deliveryDate.getDay() !== 0) { // Not Sunday
      daysAdded++;
    }
  }

  return deliveryDate;
}
```

4.4 Tracking Implementation

```
┌─────────────────────────────┐
│      Order Placed           │
└──────────────┬──────────────┘
     │
     ▼
┌─────────────────────────────┐
│  Generate AWB via Shiprocket API  │
└──────────────┬──────────────┘
     │
     ▼
┌─────────────────────────────┐
│   Store AWB in Database     │
└──────────────┬──────────────┘
     │
     ▼
```

```
┌─────────────────────────────────────────────┐
│ Periodic Tracking Updates (Webhook/Cron)    │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│ Update shipments.tracking_history           │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│ Display Real-time Status to Customer        │
└─────────────────────────────────────────────┘
```

---

5. PAYMENT INTEGRATION

5.1 Payment Gateway Comparison

Feature	**Razorpay**	**Stripe**	**Recommendation**
India Support	Excellent - UPI, Wallets	Good - Limited UPI	Razorpay for India
International	Limited	Excellent	Stripe for global
Setup Fee	None	None	Both free to start
Transaction Fee	2% + GST	2.9% + 30¢	Razorpay cheaper for India
Settlement	T+3 days	T+7 days	Razorpay faster

5.2 Payment Flow Architecture

```
┌─────────────────────────────────────────────┐
│    User Clicks "Pay Now"                    │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│ Frontend Creates Order via API              │
│    POST /api/create-order                   │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│ Backend Creates Payment Intent              │
│   • Razorpay.orders.create()        │
│   • Returns order_id to frontend      │
└──────────────────────────────────────────────┬──────────────────────────────────────────────────┘
              │
              ▼
┌─────────────────────────────────────────────────────────────────────────────────────────────────┐
│   Razorpay Checkout Modal Opens         │
│   • Card/UPI/Wallet payment          │
└──────────────────────────────────────────────┬──────────────────────────────────────────────────┘
              │
              ▼
┌─────────────────────────────────────────────────────────────────────────────────────────────────┐
│     Payment Success/Failure           │
└────────────────────────────┬───────────────────────────────┬────────────────────────────────────┘
          │             │
   Success ▼              ▼ Failure
┌─────────────────────────┐  ┌─────────────────────────────┐
│ Webhook Called  │   │ Retry Payment  │
│ Order Confirmed │   │ or Cancel     │
└─────────────────────────┘  └─────────────────────────────┘
```

5.3 Security Implementation

PCI DSS Compliance

- Never store card details
- Use tokenization for recurring payments
- Implement 3D Secure authentication
- Regular security audits

Webhook Security

javascript

```
// Verify Razorpay webhook signature
function verifyWebhookSignature(body, signature, secret) {
  const expectedSignature = crypto
    .createHmac('sha256', secret)
    .update(JSON.stringify(body))
    .digest('hex');

  return expectedSignature === signature;
}
```

---

6. MEDIA & CDN ARCHITECTURE

6.1 Content Delivery Strategy

```
┌─────────────────────────────────────┐
│        Static Assets                │
├─────────────────────────────────────┤
│ • Product Images                    │
│ • 3D Models (GLTF/GLB)              │
│ • Hero Videos                       │
│ • Textures & Materials              │
└─────────────────┬───────────────────┘
                  │
                  ▼
┌─────────────────────────────────────┐
│      Object Storage (Origin)        │
├─────────────────────────────────────┤
│ • Cloudflare R2 (Free egress)       │
│ • AWS S3 (Pay per GB)               │
│ • Supabase Storage (1GB free)       │
└─────────────────┬───────────────────┘
                  │
                  ▼
┌─────────────────────────────────────┐
│        CDN Distribution             │
├─────────────────────────────────────┤
│ • Cloudflare CDN (Free tier)        │
│ • Global edge locations             │
│ • Automatic caching                 │
└─────────────────┬───────────────────┘
                  │
```

```
          ▼
┌──────────────────────────────────────────────┐
│    End User Browser    │
└──────────────────────────────────────────────┘
```

6.2 Asset Optimization Guidelines

Image Optimization

- Format: WebP with JPEG fallback
- Responsive sizes: 360px, 720px, 1080px, 2048px
- Lazy loading with intersection observer
- Compression: 85% quality for product images

3D Model Optimization

- Format: GLTF 2.0 with Draco compression
- Texture atlas: Combine multiple textures
- LOD (Level of Detail): 3 levels based on distance
- Maximum polygon count: 50,000 for hero models

Video Optimization

- Format: MP4 (H.264) with WebM fallback
- Resolution: 1080p maximum for hero videos
- Bitrate: 2-3 Mbps for quality/size balance
- Duration: 15-30 seconds maximum

---

7. DEPLOYMENT & HOSTING ARCHITECTURE

7.1 Infrastructure Overview

Component	Platform	Free Tier Limits	Scaling Strategy
Frontend	Vercel	100GB bandwidth/month, 100 deployments	Netlify as backup
API/Functions	Vercel Functions	100K requests/month	Distribute across providers
Database	Supabase	500MB, 2GB transfer	Archive old data
File Storage	Cloudflare R2	10GB storage, unlimited egress	Perfect for media

Component	Platform	Free Tier Limits	Scaling Strategy
CDN	Cloudflare	Unlimited	Use for all static assets

7.2 CI/CD Pipeline

```
┌─────────────────────────────────────────────────────────────┐
│      Developer Pushes to GitHub          │
└──────────────────────────┬──────────────────────────────────┘
         │
         ▼
┌─────────────────────────────────────────────────────────────┐
│      GitHub Actions Triggered           │
├─────────────────────────────────────────────────────────────┤
│  • Run ESLint & Prettier              │
│  • Run Unit Tests (Jest)             │
│  • Run E2E Tests (Playwright)           │
│  • Build Production Bundle            │
└──────────────────────────┬──────────────────────────────────┘
         │
         ▼
┌─────────────────────────────────────────────────────────────┐
│    Vercel Automatic Deployment             │
├─────────────────────────────────────────────────────────────┤
│  • Preview URL for PR                  │
│  • Production deploy on main branch          │
│  • Environment variables injection          │
└──────────────────────────┬──────────────────────────────────┘
         │
         ▼
┌─────────────────────────────────────────────────────────────┐
│      Post-Deployment Checks          │
├─────────────────────────────────────────────────────────────┤
│  • Lighthouse performance audit          │
│  • Security headers validation         │
│  • Uptime monitoring activation          │
└─────────────────────────────────────────────────────────────┘
```

7.3 Environment Configuration

bash

```
# .env.local (Development)
NEXT_PUBLIC_API_URL=http://localhost:3000/api
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=xxx
SUPABASE_SERVICE_KEY=xxx
RAZORPAY_KEY_ID=rzp_test_xxx
RAZORPAY_SECRET_KEY=xxx
SHIPROCKET_EMAIL=xxx
SHIPROCKET_PASSWORD=xxx

# .env.production (Production - Set in Vercel Dashboard)
NEXT_PUBLIC_API_URL=https://petalpouches.com/api
# Additional production keys...
```

---

8. PERFORMANCE OPTIMIZATION

8.1 Core Web Vitals Targets

Metric	Target	Current Best Practice
LCP (Largest Contentful Paint)	< 2.5s	Optimize hero images, use CDN
FID (First Input Delay)	< 100ms	Code splitting, lazy loading
CLS (Cumulative Layout Shift)	< 0.1	Reserve space for dynamic content
TTFB (Time to First Byte)	< 600ms	Edge caching, SSG where possible

8.2 3D Performance Optimization Strategy

Progressive Enhancement Approach

```javascript
```

```
// Detect device capabilities
const deviceTier = detectDeviceCapabilities();

switch(deviceTier) {
  case 'high':
    // Full 3D experience with shadows and post-processing
    return <Full3DExperience />;

  case 'medium':
    // Simplified 3D with reduced polygon count
    return <Simplified3DExperience />;

  case 'low':
    // Static image or video fallback
    return <StaticFallback />;
}
```

Mobile Optimization Checklist

- [ ] Detect mobile devices and reduce canvas resolution
- [ ] Disable shadows and post-processing on mobile
- [ ] Use lower-resolution textures (512x512 max)
- [ ] Implement touch controls instead of mouse
- [ ] Provide "View in 3D" button instead of auto-load
- [ ] Maximum 30 FPS frame rate cap on mobile

8.3 Bundle Size Optimization

javascript

```
// Next.js configuration for optimization
module.exports = {
  webpack: (config, { isServer }) => {
    // Tree shaking for Three.js
    config.resolve.alias = {
      ...config.resolve.alias,
      'three': path.join(__dirname, 'node_modules/three/build/three.module.js'),
    };

    // Analyze bundle size in development
    if (process.env.ANALYZE === 'true') {
      const BundleAnalyzerPlugin =
        require('webpack-bundle-analyzer').BundleAnalyzerPlugin;
      config.plugins.push(new BundleAnalyzerPlugin());
    }

    return config;
  },

  // Image optimization
  images: {
    domains: ['cdn.petalpouches.com'],
    formats: ['image/avif', 'image/webp'],
  },

  // Compression
  compress: true,

  // SWC minification
  swcMinify: true,
};
```

---

9. MONITORING & ANALYTICS

9.1 Application Monitoring Stack

Category	Tool	Purpose	Free Tier
Error Tracking	Sentry	JavaScript errors, API failures	5K errors/month
Performance	Vercel Analytics	Core Web Vitals monitoring	Included with Vercel
User Analytics	Plausible	Privacy-focused analytics	30-day trial

Category	Tool	Purpose	Free Tier
Uptime	Better Uptime	Endpoint monitoring	10 monitors free
Logs	Axiom	Centralized logging	500MB/month free

9.2 Key Performance Indicators (KPIs)

javascript

```javascript
// Custom analytics events
const trackingEvents = {
  // Conversion funnel
  VIEW_PRODUCT: 'view_product',
  ADD_TO_CART: 'add_to_cart',
  BEGIN_CHECKOUT: 'begin_checkout',
  PURCHASE: 'purchase',

  // Engagement metrics
  VIEW_3D_MODEL: 'view_3d_model',
  PERSONALIZE_PRODUCT: 'personalize_product',
  SHARE_PRODUCT: 'share_product',

  // Performance metrics
  SLOW_3D_LOAD: 'slow_3d_load', // > 3 seconds
  API_ERROR: 'api_error',
};
```

---

10. SECURITY & COMPLIANCE

10.1 Security Checklist

Application Security

- ✓ HTTPS everywhere with SSL/TLS certificates
- ✓ Content Security Policy (CSP) headers
- ✓ XSS protection through input sanitization
- ✓ SQL injection prevention via parameterized queries
- ✓ Rate limiting on all API endpoints
- ✓ JWT tokens with secure HTTP-only cookies
- ✓ Regular dependency updates and security audits

Data Protection

- [x] PCI DSS compliance for payment processing
- [x] GDPR compliance for EU customers
- [x] Data encryption at rest and in transit
- [x] Regular backups with point-in-time recovery
- [x] Access logs and audit trails

10.2 Compliance Requirements

Regulation	Requirements	Implementation
PCI DSS	Secure payment processing	Use Razorpay hosted checkout
GDPR	EU data protection	Cookie consent, data export/deletion
India IT Act	Data localization	Host data in India region
COPPA	Child protection	Age verification for accounts

---

11. IMPLEMENTATION ROADMAP

Phase 1: MVP Development (Weeks 1-4)

Objective: Basic e-commerce functionality with simple product catalog

Week 1-2: Foundation

- [ ] Setup Next.js project with TypeScript
- [ ] Configure Tailwind CSS
- [ ] Setup Supabase database and auth
- [ ] Design database schema
- [ ] Create basic component library

Week 3-4: Core Features

- [ ] Product listing and detail pages
- [ ] Shopping cart functionality
- [ ] User authentication flow
- [ ] Basic checkout process
- [ ] Razorpay payment integration

Phase 2: 3D Integration (Weeks 5-6)

Objective: Add immersive 3D product visualization

- [ ] Integrate react-three-fiber
- [ ] Create 3D product viewer component
- [ ] Optimize 3D models with Draco compression
- [ ] Implement progressive loading
- [ ] Add mobile fallbacks

Phase 3: Shipping Integration (Weeks 7-8)

Objective: Complete order fulfillment system

- [ ] Integrate Shiprocket API
- [ ] Pin code serviceability check
- [ ] Delivery date calculation
- [ ] Order tracking system
- [ ] Return/refund workflow

Phase 4: Personalization (Weeks 9-10)

Objective: Gift customization features

- [ ] Text engraving preview
- [ ] Gift message cards
- [ ] Custom packaging options
- [ ] Save personalization templates
- [ ] Share gift previews

Phase 5: Performance & Polish (Weeks 11-12)

Objective: Production-ready optimization

- [ ] Performance audit and optimization
- [ ] SEO implementation
- [ ] Analytics integration
- [ ] A/B testing setup
- [ ] Load testing

Phase 6: Launch Preparation (Weeks 13-14)

Objective: Go-live readiness

- [ ] Security audit
- [ ] Documentation completion
- [ ] Team training

- [ ] Marketing site updates
- [ ] Beta testing with real users
- [ ] Bug fixes and refinements
- [ ] Production deployment

---

12. DETAILED TECHNOLOGY STACK REFERENCE

12.1 Complete Frontend Stack

Category	Technology	Version	Purpose	NPM Package
Core Framework	React.js	18.2+	UI component library	`react`, `react-dom`
Meta-Framework	Next.js	14.0+	SSR, routing, optimization	`next`
Language	TypeScript	5.0+	Type safety	`typescript`
Styling	Tailwind CSS	3.4+	Utility-first CSS	`tailwindcss`
3D Graphics	Three.js	r158+	3D rendering engine	`three`
3D React	react-three-fiber	8.15+	React renderer for Three.js	`@react-three/fiber`
3D Helpers	drei	9.88+	Three.js utilities	`@react-three/drei`
Animation	Framer Motion	10.16+	UI animations	`framer-motion`
Vector Animation	Lottie React	2.4+	JSON animations	`lottie-react`
State Management	Zustand	4.4+	Global state	`zustand`
Forms	React Hook Form	7.47+	Form handling	`react-hook-form`
Validation	Zod	3.22+	Schema validation	`zod`
Icons	Lucide React	0.292+	Icon library	`lucide-react`
HTTP Client	Axios	1.6+	API calls	`axios`
Date Handling	date-fns	2.30+	Date utilities	`date-fns`

12.2 Backend & Infrastructure Stack

Category	Service/Tool	Purpose	Configuration
Runtime	Node.js 18 LTS	JavaScript runtime	Latest LTS version
API Framework	Next.js API Routes	Serverless endpoints	Built into Next.js
Database	Supabase (PostgreSQL)	Primary database	500MB free tier
ORM	Prisma	Database queries	`prisma` package
Authentication	Supabase Auth	User management	Built into Supabase
File Storage	Cloudflare R2	Media storage	10GB free

Category	Service/Tool	Purpose	Configuration
CDN	Cloudflare	Content delivery	Free tier
Email	Resend	Transactional emails	100 emails/day free
Monitoring	Sentry	Error tracking	5K errors/month free

12.3 Third-Party Integrations

Integration	Provider	API Endpoints	Rate Limits
Payment Gateway	Razorpay	`/v1/orders`, `/v1/payments`	No strict limit
Shipping	Shiprocket	`/v1/external/courier/serviceability`	100 requests/min
SMS	Twilio	`/2010-04-01/Accounts/{AccountSid}/Messages`	Pay per use
Analytics	Google Analytics 4	Measurement Protocol API	25M hits/property/month
Reviews	Trustpilot	`/v1/business-units/{id}/reviews`	100 requests/hour

12.4 Development Tools

Tool	Purpose	Configuration File
Version Control	Git + GitHub	`.gitignore`
Package Manager	pnpm	`pnpm-lock.yaml`
Linting	ESLint	`.eslintrc.json`
Formatting	Prettier	`.prettierrc`
Git Hooks	Husky	`.husky/`
Commit Convention	Commitizen	`.czrc`
Testing	Jest + React Testing Library	`jest.config.js`
E2E Testing	Playwright	`playwright.config.ts`
Bundle Analyzer	Webpack Bundle Analyzer	Next.js config

---

13. 3D IMPLEMENTATION GUIDE

13.1 3D Asset Pipeline

```
┌─────────────────────────────────────────────┐
│      3D Model Creation (Blender)      │
└──────────────────┬──────────────────────────┘
       │
```

```
          ▼
┌─────────────────────────────────────────────┐
│   Export as GLTF 2.0 / GLB Format          │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│     Optimization Pipeline                  │
├─────────────────────────────────────────────┤
│ • Draco Geometry Compression               │
│ • Texture Atlas Generation                 │
│ • KTX2 Texture Compression                 │
│ • Mesh Optimization (gltf-transform)       │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│   Upload to CDN (Cloudflare R2)            │
└──────────────────────┬──────────────────────┘
          │
          ▼
┌─────────────────────────────────────────────┐
│  Load in React with react-three-fiber      │
└─────────────────────────────────────────────┘
```

13.2 React Three Fiber Implementation

```javascript
```

```
// 3D Product Viewer Component
import { Canvas } from '@react-three/fiber'
import { OrbitControls, Environment, useGLTF, Preload } from '@react-three/drei'
import { Suspense } from 'react'

function ProductModel({ url }) {
  const { scene } = useGLTF(url)
  return <primitive object={scene} />
}

export function Product3DViewer({ modelUrl }) {
  return (
    <Canvas
      camera={{ position: [0, 0, 5], fov: 45 }}
      performance={{ min: 0.5 }} // Adaptive performance
    >
      <Suspense fallback={null}>
        <ambientLight intensity={0.5} />
        <spotLight position={[10, 10, 10]} angle={0.15} />
        <ProductModel url={modelUrl} />
        <Environment preset="sunset" />
        <OrbitControls
          enablePan={false}
          maxPolarAngle={Math.PI / 2}
          minPolarAngle={Math.PI / 3}
        />
        <Preload all />
      </Suspense>
    </Canvas>
  )
}
```

13.3 Performance Metrics for 3D

Metric	Target	Measurement Method
Initial Load Time	< 2 seconds	Performance.now()
Frame Rate	30-60 FPS	Stats.js monitor
Draw Calls	< 100	Spector.js
Triangle Count	< 50,000	Three.js renderer.info
Texture Memory	< 50MB	GPU profiling

14. COST OPTIMIZATION STRATEGY

14.1 Free Tier Maximization

Service	Free Tier Limit	Usage Strategy	Overflow Plan
Vercel Hosting	100GB bandwidth/month	Offload media to CDN	Upgrade to Pro ($20/month)
Supabase	500MB DB, 2GB transfer	Archive old orders	Upgrade to Pro ($25/month)
Cloudflare R2	10GB storage, unlimited egress	Primary media storage	Pay $0.015/GB after 10GB
Netlify Functions	125K requests/month	Distribute across providers	Use Vercel Functions too
Sentry	5K errors/month	Filter non-critical errors	Upgrade as needed

14.2 Monthly Cost Projection

```
Initial Phase (0-1000 orders/month):
├── Hosting: $0 (free tiers)
├── Database: $0 (Supabase free)
├── CDN: $0 (Cloudflare free)
├── Payment Gateway: ~$40 (2% of $2000 revenue)
├── Shipping API: $0 (pay per shipment to courier)
└── Total: ~$40/month

Growth Phase (1000-5000 orders/month):
├── Hosting: $20 (Vercel Pro)
├── Database: $25 (Supabase Pro)
├── CDN: $5 (additional storage)
├── Payment Gateway: ~$200 (2% of $10,000 revenue)
├── Email: $15 (Resend/Brevo)
└── Total: ~$265/month
```

---

15. TESTING STRATEGY

15.1 Testing Pyramid

```
        /\
       /E2E\      (5%)
      / Tests\     - Critical user journeys
     /        \   - Playwright
    /──────────\
   / Integration\   (15%)
```

```
     /   Tests    \   - API endpoints
    /              \  - Database operations
   /────────────────────\
  /   Unit Tests     \ (80%)
 /                    \ - Components
─────────────────────────── - Utilities
                        - Business logic
```

15.2 Test Implementation Examples

Unit Test Example

```javascript
// ProductCard.test.tsx
import { render, screen } from '@testing-library/react'
import { ProductCard } from './ProductCard'

describe('ProductCard', () => {
  it('displays product title and price', () => {
    const product = {
      title: 'Rose Gold Charm',
      price: 1299
    }

    render(<ProductCard product={product} />)

    expect(screen.getByText('Rose Gold Charm')).toBeInTheDocument()
    expect(screen.getByText('₹1,299')).toBeInTheDocument()
  })
})
```

E2E Test Example

javascript

```
// checkout.spec.ts (Playwright)
import { test, expect } from '@playwright/test'

test('complete checkout flow', async ({ page }) => {
  await page.goto('/products/rose-gold-charm')
  await page.click('button:text("Add to Cart")')
  await page.click('a:text("Checkout")')
  await page.fill('input[name="email"]', 'test@example.com')
  await page.fill('input[name="pincode"]', '110001')
  await page.click('button:text("Calculate Delivery")')
  await expect(page.locator('.delivery-date')).toBeVisible()
  await page.click('button:text("Proceed to Payment")')
  // Payment gateway mock
})
```

---

16. SEARCH ENGINE OPTIMIZATION (SEO)

16.1 Technical SEO Implementation

```
javascript
```

```
// next-seo.config.js
export default {
  titleTemplate: '%s | The Petal Pouches',
  defaultTitle: 'The Petal Pouches - Personalized Gifts',
  description: 'Premium personalized gifts with 3D preview',
  openGraph: {
    type: 'website',
    locale: 'en_IN',
    url: 'https://petalpouches.com',
    images: [
      {
        url: 'https://petalpouches.com/og-image.jpg',
        width: 1200,
        height: 630,
        alt: 'The Petal Pouches'
      }
    ]
  },
  twitter: {
    handle: '@petalpouches',
    cardType: 'summary_large_image'
  }
}
```

16.2 Schema Markup for E-commerce

```
javascript
```

```
// Product Schema
const productSchema = {
  '@context': 'https://schema.org',
  '@type': 'Product',
  name: 'Rose Gold Charm Bracelet',
  description: 'Personalized charm bracelet',
  image: 'https://cdn.petalpouches.com/products/charm-bracelet.jpg',
  brand: {
    '@type': 'Brand',
    name: 'The Petal Pouches'
  },
  offers: {
    '@type': 'Offer',
    price: '1299',
    priceCurrency: 'INR',
    availability: 'https://schema.org/InStock',
    deliveryLeadTime: {
      '@type': 'QuantitativeValue',
      minValue: 2,
      maxValue: 5,
      unitCode: 'DAY'
    }
  }
}
```

---

17. ACCESSIBILITY (A11Y) COMPLIANCE

17.1 WCAG 2.1 Level AA Checklist

✓ **Perceivable**

- Alt text for all images
- Captions for videos
- Color contrast ratio ≥ 4.5:1
- Responsive text sizing

✓ **Operable**

- Keyboard navigation support
- Skip to main content link
- Focus indicators visible

- No seizure-inducing content

✓ **Understandable**

- Clear error messages
- Consistent navigation
- Labels for form inputs
- Language declaration

✓ **Robust**

- Valid HTML markup
- ARIA labels where needed
- Screen reader compatible
- Cross-browser support

17.2 3D Accessibility Considerations

```javascript
```

```
// Accessible 3D viewer with fallback
function Accessible3DViewer({ product }) {
  const [is3DEnabled, setIs3DEnabled] = useState(true)

  return (
    <div role="img" aria-label={`3D view of ${product.name}`}>
      {is3DEnabled ? (
        <Canvas>
          {/* 3D content */}
        </Canvas>
      ) : (
        <img src={product.image} alt={product.name} />
      )}

      <button
        onClick={() => setIs3DEnabled(!is3DEnabled)}
        aria-pressed={is3DEnabled}
      >
        {is3DEnabled ? 'Switch to 2D' : 'Switch to 3D'}
      </button>

      <p className="sr-only">
        Use arrow keys to rotate the model
      </p>
    </div>
  )
}
```

---

18. DISASTER RECOVERY & BACKUP

18.1 Backup Strategy

Data Type	Backup Frequency	Retention	Storage Location
Database	Daily automated	30 days	Supabase built-in
Code	On every commit	Forever	GitHub
Media Assets	Real-time sync	Forever	R2 + S3 backup
User Uploads	Immediate	90 days	Multiple regions
Configuration	On change	Version controlled	Git

18.2 Recovery Time Objectives (RTO)

```
System Component    RTO       RPO
────────────────────────────────────────────
Frontend           < 5 min    0 (CDN cached)
API/Backend        < 15 min   1 hour
Database           < 1 hour   15 minutes
Media/CDN          < 1 min    0 (multi-region)
Payment Processing < 30 min   0 (gateway handles)
```

---

19. LEGAL & COMPLIANCE DOCUMENTATION

19.1 Required Legal Pages

1. **Privacy Policy**
   - Data collection practices
   - Third-party services disclosure
   - GDPR compliance statement
   - Cookie usage policy
2. **Terms of Service**
   - User agreements
   - Liability limitations
   - Intellectual property rights
   - Dispute resolution
3. **Return & Refund Policy**
   - 7-day return window
   - Condition requirements
   - Refund processing timeline
   - Shipping cost responsibility
4. **Shipping Policy**
   - Delivery timelines by region
   - Shipping partners disclosure

- International shipping terms
- Tracking information

19.2 Data Protection Implementation

javascript

```javascript
// GDPR Compliance - User data export
async function exportUserData(userId) {
  const userData = await db.query(`
    SELECT * FROM users WHERE id = $1
  `, [userId])

  const orders = await db.query(`
    SELECT * FROM orders WHERE user_id = $1
  `, [userId])

  return {
    profile: userData,
    orders: orders,
    exportDate: new Date().toISOString(),
    format: 'JSON'
  }
}

// Right to erasure (GDPR Article 17)
async function deleteUserData(userId) {
  // Anonymize orders for accounting
  await db.query(`
    UPDATE orders
    SET user_id = NULL,
        shipping_address = 'DELETED'
    WHERE user_id = $1
  `, [userId])

  // Delete user account
  await db.query(`
    DELETE FROM users WHERE id = $1
  `, [userId])
}
```

20. APPENDIX

20.1 Useful Resources & Documentation

Official Documentation

- **Next.js Documentation:** https://nextjs.org/docs
- **React Three Fiber:** https://docs.pmnd.rs/react-three-fiber
- **Supabase Docs:** https://supabase.com/docs
- **Tailwind CSS:** https://tailwindcss.com/docs
- **Shiprocket API:** https://apidocs.shiprocket.in
- **Razorpay Integration:** https://razorpay.com/docs

Performance Tools

- **Lighthouse CI:** https://github.com/GoogleChrome/lighthouse-ci
- **WebPageTest:** https://www.webpagetest.org
- **Bundle Analyzer:** https://bundlephobia.com
- **3D Optimization:** https://gltf.report

Learning Resources

- **Three.js Journey:** https://threejs-journey.com
- **React TypeScript Cheatsheet:** https://react-typescript-cheatsheet.netlify.app
- **Web.dev Performance:** https://web.dev/performance
- **A11y Project:** https://www.a11yproject.com

20.2 Troubleshooting Guide

Issue	Possible Cause	Solution
Slow 3D Loading	Large model files	Compress with Draco, use LOD
High Bounce Rate	Slow initial load	Implement SSG, optimize LCP
Payment Failures	Gateway timeout	Implement retry logic
Delivery Calculation Error	API rate limit	Cache pincode data
Mobile Performance	Heavy animations	Reduce motion on mobile

20.3 Contact Information

Technical Support

- Development Team: dev@petalpouches.com
- Infrastructure: ops@petalpouches.com
- Security Issues: security@petalpouches.com

Third-Party Support

- Shiprocket: support@shiprocket.com
- Razorpay: support@razorpay.com
- Supabase: support@supabase.com
- Vercel: support@vercel.com

DOCUMENT CONTROL

Version	Date	Author	Changes
1.0	Oct 2025	Technical Team	Initial document creation
-	-	-	Comprehensive architecture documentation
-	-	-	Complete implementation guide

Document Review Schedule: Quarterly **Next Review Date:** January 2026 **Document Owner:** Chief Technology Officer

---