Complete Wedmantra App Development Roadmap (2.5 Months)

🎯 MIRRAW-INSPIRED FEATURES FOR MVP

Based on Mirraw analysis, here are MUST-HAVE features for your MVP:

Core Features (Week 1-4)

- ✅ User Registration/Login (Phone OTP)
- ✅ Product Catalog with Categories
- ✅ Search & Filters (Fabric, Price, Occasion)
- ✅ Shopping Cart
- ✅ Wishlist
- ✅ Order Management
- ✅ Payment Gateway (Razorpay + COD)
- ✅ Basic Admin Panel

Mobile App Essential Features (Week 5-8)

- 📱 Home Screen with Banners
- 📱 Product Grid/List View
- 📱 Product Details with Image Gallery
- 📱 Search with Voice Input
- 📱 Filter & Sort
- 📱 User Profile & Addresses
- 📱 Order Tracking
- 📱 Push Notifications

Nice-to-Have (Week 9-10)

- 🎁 Referral Program
- 📊 Basic Analytics
- 🔄 Recently Viewed
- 💭 Customer Support Chat

📅 DETAILED TIMELINE

WEEK 1-2: Backend Foundation

Goal: Fix critical issues & set up solid foundation

Backend Critical Fixes:

bash

```bash
# Day 1-3: Fix Breaking Issues
1. Fix Redis connection (new syntax)
2. Add missing model methods
3. Fix authentication middleware
4. Fix route ordering (search before :id)
5. Add input validation
```

Mobile App Setup:

bash

```bash
# Day 4-7: React Native Setup
npx react-native init WedmantraApp
# OR
npm install -g @ionic/cli
ionic start WedmantraApp tabs --type=react --capacitor
```

WEEK 3-4: Core Backend APIs

Goal: Complete all essential APIs

API Endpoints to Complete:

javascript

```javascript
// Essential APIs for MVP
GET /api/mobile/home           // Home screen data
GET /api/products/search       // Product search
GET /api/products/categories     // Category-wise products
POST /api/auth/otp-login       // OTP-based login
GET /api/user/addresses        // User addresses
POST /api/orders/create        // Order creation
GET /api/orders/track/:id      // Order tracking
```

WEEK 5-6: Mobile App Core

Goal: Build main mobile app screens

Key Screens:

1. **Splash & Onboarding**
2. **Login/Register (OTP)**
3. **Home Screen**
4. **Product Listing**
5. **Product Details**
6. **Search & Filters**
7. **Cart**
8. **Profile**

WEEK 7-8: Mobile App Advanced

Goal: Complete shopping flow

Advanced Features:

1. **Checkout Flow**
2. **Payment Integration**
3. **Order Management**
4. **Address Management**
5. **Wishlist**
6. **Push Notifications**

WEEK 9-10: Testing & Deployment

Goal: App store ready

Final Steps:

1. **Testing & Bug Fixes**
2. **Performance Optimization**
3. **App Store Assets**
4. **Backend Deployment**
5. **App Store Submission**

🔧 IMMEDIATE BACKEND FIXES (This Week)

1. Fix Redis Connection

javascript

```javascript
// src/config/redis.js - REPLACE ENTIRE FILE
const redis = require('redis');

const redisClient = redis.createClient({
  host: process.env.REDIS_HOST || 'localhost',
  port: process.env.REDIS_PORT || 6379,
  password: process.env.REDIS_PASSWORD,
  legacyMode: true
});

redisClient.on('error', (err) => console.log('Redis Client Error', err));
redisClient.on('connect', () => console.log('Redis Connected'));

// Connect
(async () => {
  await redisClient.connect();
})();

module.exports = redisClient;
```

2. Add Missing Model Methods

javascript

```javascript
// Add to src/models/productModel.js
async getBySKU(sku) {
  const result = await db.query('SELECT * FROM products WHERE sku = $1', [sku]);
  return result.rows[0];
},

async getFeatured(limit = 10) {
  const result = await db.query(
    'SELECT * FROM products WHERE featured = true AND status = \'active\' ORDER BY created_at DESC LIMIT $1',
    [limit]
  );
  return result.rows;
},

// Add to src/models/orderModel.js
async getAllOrders() {
  const result = await db.query(
    `SELECT o.*, u.first_name, u.last_name, u.email
     FROM orders o
     LEFT JOIN users u ON o.user_id = u.id
     ORDER BY o.created_at DESC`
  );
  return result.rows;
}
```

3. Mobile-Optimized APIs

javascript

```javascript
// src/routes/mobile.js - NEW FILE
const express = require('express');
const router = express.Router();
const ProductModel = require('../models/productModel');
const CategoryModel = require('../models/categoryModel');
const BannerModel = require('../models/bannerModel');

// Home screen data
router.get('/home', async (req, res) => {
  try {
    const [banners, categories, featured] = await Promise.all([
      BannerModel.getActiveBanners(),
      CategoryModel.getAll(),
      ProductModel.getFeatured(8)
    ]);

    res.json({
      success: true,
      data: {
        banners: banners.slice(0, 5),
        categories: categories.slice(0, 6),
        featuredProducts: featured,
        offers: {
          title: "Special Offers",
          subtitle: "Up to 70% Off"
        }
      }
    });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

// Product search with mobile-optimized response
router.get('/products/search', async (req, res) => {
  try {
    const { q, category, minPrice, maxPrice, page = 1, limit = 20 } = req.query;

    const result = await ProductModel.search({
      q, category_id: category, min_price: minPrice,
      max_price: maxPrice, page, limit
    });

    res.json({
      success: true
      success: true,
      data: result.products,
      pagination: {
        page: parseInt(page),
        limit: parseInt(limit),
        total: result.total,
        hasMore: (page * limit) < result.total
      }
    });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

module.exports = router;
```

📱 MOBILE APP ARCHITECTURE

Tech Stack Decision:

```bash
# RECOMMENDED: React Native with Expo (Faster Development)
npx create-expo-app WedmantraApp --template
cd WedmantraApp

# Essential Dependencies
npm install @react-navigation/native @react-navigation/stack
npm install react-native-vector-icons react-native-image-picker
npm install @reduxjs/toolkit react-redux redux-persist
npm install axios react-native-async-storage
npm install react-native-push-notification
```

Folder Structure:

```
WedmantraApp/
├── src/
│   ├── components/
│   │   ├── common/
│   │   ├── ProductCard.js
│   │   ├── CategoryCard.js
│   │   └── CartItem.js
│   ├── screens/
│   │   ├── auth/
│   │   │   ├── LoginScreen.js
│   │   │   └── OTPScreen.js
│   │   ├── home/
│   │   │   └── HomeScreen.js
│   │   ├── products/
│   │   │   ├── ProductListScreen.js
│   │   │   ├── ProductDetailScreen.js
│   │   │   └── SearchScreen.js
│   │   ├── cart/
│   │   │   └── CartScreen.js
│   │   └── profile/
│   │       └── ProfileScreen.js
│   ├── services/
│   │   ├── api.js
│   │   ├── auth.js
│   │   └── storage.js
│   ├── redux/
│   │   ├── store.js
│   │   ├── authSlice.js
│   │   ├── productSlice.js
│   │   └── cartSlice.js
│   └── utils/
│       ├── constants.js
│       └── helpers.js
└── assets/
    ├── images/
    └── icons/
```

Sample Home Screen (React Native):

javascript

```javascript
// src/screens/home/HomeScreen.js
import React, { useEffect, useState } from 'react';
import { ScrollView, View, Text, FlatList, TouchableOpacity } from 'react-native';
import { api } from '../../services/api';

const HomeScreen = ({ navigation }) => {
  const [homeData, setHomeData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    loadHomeData();
  }, []);

  const loadHomeData = async () => {
    try {
      const response = await api.get('/mobile/home');
      setHomeData(response.data.data);
    } catch (error) {
      console.error('Error loading home data:', error);
    } finally {
      setLoading(false);
    }
  };

  const renderBanner = ({ item }) => (
    <TouchableOpacity style={styles.banner}>
      <Image source={{ uri: item.image_url }} style={styles.bannerImage} />
    </TouchableOpacity>
  );

  const renderCategory = ({ item }) => (
    <TouchableOpacity
      style={styles.categoryCard}
      onPress={() => navigation.navigate('ProductList', { categoryId: item.id })}
    >
      <Image source={{ uri: item.image_url }} style={styles.categoryImage} />
      <Text style={styles.categoryName}>{item.name}</Text>
    </TouchableOpacity>
  );

  if (loading) return <LoadingScreen />;

  return (
    <ScrollView style={styles.container}>
      {/* Banners */
      {/* Banners */}
      <FlatList
        data={homeData?.banners}
        renderItem={renderBanner}
        horizontal
        showsHorizontalScrollIndicator={false}
      />

      {/* Categories */}
      <Text style={styles.sectionTitle}>Shop by Category</Text>
      <FlatList
        data={homeData?.categories}
        renderItem={renderCategory}
        numColumns={3}
      />

      {/* Featured Products */}
      <Text style={styles.sectionTitle}>Featured Products</Text>
      <FlatList
        data={homeData?.featuredProducts}
        renderItem={renderProduct}
        horizontal
        showsHorizontalScrollIndicator={false}
      />
    </ScrollView>
  );
};
```

🚀 DEPLOYMENT STRATEGY

Backend Deployment (Week 8):

bash

```bash
# Digital Ocean Droplet (Cost-effective for 1000 users)
# 2GB RAM, 1 vCPU - $12/month

# Docker deployment
docker-compose -f docker-compose.production.yml up -d

# PM2 for process management
npm install -g pm2
pm2 start ecosystem.config.js
pm2 startup
pm2 save
```

Mobile App Deployment (Week 9-10):

```bash
# Android (Google Play Console)
npx react-native run-android --variant=release

# iOS (TestFlight -> App Store)
npx react-native run-ios --configuration Release
```

📊 SUCCESS METRICS

Technical KPIs:

- ✅ API Response Time < 500ms
- ✅ App Load Time < 3 seconds
- ✅ Zero Critical Bugs
- ✅ 99.9% Uptime

Business KPIs:

- 👥 100 Users in First Month
- 📱 4.0+ App Store Rating
- 🛒 5% Cart Conversion Rate
- 💰 $10K GMV in First Quarter

🎁 BONUS FEATURES (If Time Permits)

1. **Voice Search** - "Search for red silk sarees"
2. **AR Try-On** - Basic virtual draping
3. **Social Login** - Google/Facebook
4. **Referral Program** - Earn points for invites
5. **Push Notifications** - Order updates, offers

💡 DEVELOPMENT TIPS

Time-Saving Strategies:

1. **Use UI Libraries**: NativeBase, React Native Elements
2. **Mock Data First**: Test UI before backend completion
3. **Parallel Development**: Backend + Mobile simultaneously
4. **Reuse Components**: Create once, use everywhere
5. **Focus on Core**: Polish later, functionality first

Testing Strategy:

bash

```bash
# Backend Testing
npm install --save-dev jest supertest
npm run test

# Mobile Testing
npm install --save-dev detox
detox test
```

Would you like me to start with the immediate backend fixes or help you set up the mobile app structure first?