**SmartSheba - Sequential Feature Implementation Roadmap**

**Complete Feature Breakdown for Full App Development**

**PHASE 1: FOUNDATION FEATURES (Week 1-4)**

**Week 1: Basic App Infrastructure**

**Day 1-2: Project Setup**

1. **Flutter Project Initialization**
2. flutter create smartshebacd smartshebaflutter pub get
3. **Folder Structure Creation**
4. lib/├── core/├── data/├── domain/├── presentation/└── main.dart
5. **Dependencies Setup**
6. dependencies: flutter\_bloc: ^8.1.3 http: ^1.1.0 shared\_preferences: ^2.2.2 go\_router: ^12.1.1

**Day 3-4: Basic Navigation & Theme**

1. **App Theme Configuration**
   * Light/Dark theme setup
   * Bangla font integration
   * Color palette definition
2. **Basic Navigation Setup**
   * Route definitions
   * Navigation wrapper
   * Bottom navigation bar structure

**Day 5-7: Authentication Foundation**

1. **Login/Register UI Screens**
   * Phone number input screen
   * OTP verification screen
   * Basic form validation
2. **Basic State Management**
   * BLoC pattern setup
   * Authentication BLoC
   * Loading/error states

**Week 2: User Authentication System**

**Day 8-10: Authentication Logic**

1. **Phone Number Validation**
2. bool isValidBangladeshiPhone(String phone) {
3. RegExp regex = RegExp(r'^(\+8801|8801|01)[3-9]\d{8}$');
4. return regex.hasMatch(phone);
5. }
6. **OTP Integration**
   * Fake OTP service (for development)
   * Timer countdown
   * Resend OTP functionality
7. **Local Storage Setup**
8. class AuthLocalStorage {
9. static const String \_tokenKey = 'auth\_token';
11. static Future<void> saveToken(String token) async {
12. final prefs = await SharedPreferences.getInstance();
13. await prefs.setString(\_tokenKey, token);
14. }
15. }

**Day 11-14: User Profile**

1. **Profile Creation Screen**
   * Basic user information form
   * Profile picture placeholder
   * Location selection
2. **Profile Management**
   * Edit profile functionality
   * Data persistence
   * Profile validation

**Week 3: Service Catalog Foundation**

**Day 15-17: Service Categories**

1. **Service Category Model**
2. class ServiceCategory {
3. final String id;
4. final String name;
5. final String nameBangla;
6. final String icon;
7. final List<String> subCategories;
8. }
9. **Home Screen Layout**
   * App bar with location
   * Service category grid
   * Search bar placeholder
10. **Category Display**
    * Grid view of services
    * Category icons
    * Bangla text support

**Day 18-21: Basic Service Discovery**

1. **Service List Screen**
   * List of services in category
   * Basic service information
   * Simple navigation
2. **Service Detail Screen**
   * Service description
   * Basic pricing information
   * Contact provider button
3. **Search Functionality**
   * Basic text search
   * Filter by category
   * Search history

**Week 4: Provider Management**

**Day 22-24: Provider Profiles**

1. **Provider Model Structure**
2. class ServiceProvider {
3. final String id;
4. final String name;
5. final String phone;
6. final double rating;
7. final String location;
8. final List<String> services;
9. final bool isVerified;
10. }
11. **Provider List Display**
    * Provider cards
    * Rating display
    * Basic information layout
12. **Provider Detail Screen**
    * Full provider information
    * Service offerings
    * Contact options

**Day 25-28: Provider Registration**

1. **Provider Registration Form**
   * Multi-step form
   * Service selection
   * Basic validation
2. **Provider Dashboard Structure**
   * Empty dashboard layout
   * Navigation setup
   * Basic statistics placeholders
3. **Role-Based Navigation**
   * Customer vs Provider views
   * Conditional navigation
   * Role switching (for testing)

**PHASE 2: CORE BOOKING SYSTEM (Week 5-8)**

**Week 5: Basic Booking Creation**

**Day 29-31: Booking Foundation**

1. **Booking Model**
2. class Booking {
3. final String id;
4. final String customerId;
5. final String providerId;
6. final String serviceCategory;
7. final DateTime scheduledAt;
8. final BookingStatus status;
9. final double price;
10. }
11. **Book Service Screen**
    * Service selection
    * Date/time picker
    * Basic booking form
12. **Booking Confirmation**
    * Review booking details
    * Terms and conditions
    * Confirm booking action

**Day 32-35: Booking Management**

1. **My Bookings Screen**
   * List of user bookings
   * Status indicators
   * Basic sorting
2. **Booking Detail Screen**
   * Complete booking information
   * Status timeline
   * Action buttons
3. **Booking Status Updates**
   * Status change logic
   * UI updates
   * Simple notifications

**Week 6: Provider Booking Management**

**Day 36-38: Provider Booking Views**

1. **Incoming Requests Screen**
   * New booking requests
   * Accept/Decline actions
   * Request details
2. **Provider Schedule View**
   * Calendar view
   * Booked appointments
   * Availability management
3. **Booking Response System**
   * Accept booking logic
   * Decline with reason
   * Auto-decline expired requests

**Day 39-42: Basic Communication**

1. **In-App Chat Foundation**
2. class ChatMessage {
3. final String id;
4. final String senderId;
5. final String message;
6. final DateTime timestamp;
7. final MessageType type;
8. }
9. **Chat Screen UI**
   * Message list view
   * Input field
   * Send button
10. **Chat Integration**
    * Link chat to bookings
    * Message storage
    * Real-time updates (mock)

**Week 7: Payment Integration**

**Day 43-45: Payment Foundation**

1. **Payment Method Selection**
   * Cash on delivery option
   * Mobile banking options (UI only)
   * Payment method storage
2. **Payment Status Tracking**
   * Payment pending/completed states
   * Payment history screen
   * Transaction records
3. **Basic Pricing System**
   * Service base pricing
   * Dynamic pricing calculations
   * Price estimation

**Day 46-49: Booking Completion**

1. **Service Completion Flow**
   * Mark service as completed
   * Payment confirmation
   * Completion notifications
2. **Rating and Review System**
3. class Review {
4. final String id;
5. final String bookingId;
6. final int rating;
7. final String comment;
8. final DateTime createdAt;
9. }
10. **Review Submission**
    * Star rating widget
    * Comment input
    * Review display

**Week 8: Basic Location Services**

**Day 50-52: Location Foundation**

1. **Location Permission Handling**
2. Future<Position> getCurrentLocation() async {
3. bool serviceEnabled = await Geolocator.isLocationServiceEnabled();
4. LocationPermission permission = await Geolocator.checkPermission();
5. // Handle permissions and get location
6. }
7. **Address Input System**
   * Manual address entry
   * Address validation
   * Address book functionality
8. **Basic Map Integration**
   * Static map display
   * Location markers
   * Address confirmation

**Day 53-56: Service Area Management**

1. **Provider Location Setup**
   * Service area definition
   * Location-based filtering
   * Distance calculations
2. **Nearby Services**
   * Filter by distance
   * Sort by proximity
   * Location-based recommendations
3. **Basic Tracking UI**
   * Provider location display
   * Simple tracking screen
   * ETA estimation (static)

**PHASE 3: ADVANCED FEATURES (Week 9-12)**

**Week 9: Search and Filtering**

**Day 57-59: Advanced Search**

1. **Smart Search Implementation**
2. class SearchService {
3. List<ServiceProvider> searchProviders(String query) {
4. return providers.where((provider) =>
5. provider.name.toLowerCase().contains(query.toLowerCase()) ||
6. provider.services.any((service) =>
7. service.toLowerCase().contains(query.toLowerCase()))
8. ).toList();
9. }
10. }
11. **Search Suggestions**
    * Auto-complete functionality
    * Popular searches
    * Search history
12. **Voice Search (Basic)**
    * Voice input button
    * Speech to text
    * Search query processing

**Day 60-63: Filtering System**

1. **Advanced Filters**
   * Price range filter
   * Rating filter
   * Distance filter
   * Availability filter
2. **Filter UI Components**
   * Range sliders
   * Checkbox filters
   * Filter chips
3. **Sorting Options**
   * Sort by rating
   * Sort by price
   * Sort by distance
   * Sort by availability

**Week 10: Notifications System**

**Day 64-66: Push Notifications**

1. **Firebase Setup**
2. dependencies:
3. firebase\_messaging: ^14.7.10
4. firebase\_core: ^2.24.2
5. **Notification Types**
   * Booking confirmations
   * Status updates
   * Chat messages
   * Promotional notifications
6. **Notification Handling**
   * Foreground notifications
   * Background notifications
   * Notification navigation

**Day 67-70: In-App Notifications**

1. **Notification Center**
   * Notification list
   * Mark as read/unread
   * Notification categories
2. **Real-time Updates**
   * Status change notifications
   * New message alerts
   * Booking reminders
3. **Notification Settings**
   * Enable/disable notifications
   * Notification preferences
   * Quiet hours setting

**Week 11: User Experience Enhancements**

**Day 71-73: Onboarding**

1. **App Introduction Screens**
   * Welcome carousel
   * Feature highlights
   * Permission requests
2. **Tutorial System**
   * First-time user guide
   * Interactive tooltips
   * Help overlay
3. **User Preferences**
   * Language selection
   * Theme preferences
   * Default locations

**Day 74-77: Performance Optimization**

1. **Image Optimization**
2. Widget optimizedImage(String url) {
3. return CachedNetworkImage(
4. imageUrl: url,
5. placeholder: (context, url) => CircularProgressIndicator(),
6. errorWidget: (context, url, error) => Icon(Icons.error),
7. memCacheWidth: 300,
8. memCacheHeight: 300,
9. );
10. }
11. **List Performance**
    * Lazy loading
    * Pagination
    * Image caching
12. **App Responsiveness**
    * Loading indicators
    * Skeleton screens
    * Error handling

**Week 12: Basic AI Features**

**Day 78-80: Simple Chatbot**

1. **Rule-Based Chatbot**
2. class SimpleChatbot {
3. String getResponse(String userMessage) {
4. if (userMessage.contains('plumber') || userMessage.contains('পানির')) {
5. return 'I can help you find a plumber. What specific issue do you have?';
6. }
7. // Add more rules
8. return 'I can help you find services. What do you need?';
9. }
10. }
11. **Chat Interface**
    * Bot chat screen
    * Quick reply buttons
    * Service suggestions
12. **Intent Recognition**
    * Basic keyword matching
    * Service category mapping
    * Fallback responses

**Day 81-84: Service Recommendations**

1. **Basic Recommendation Engine**
2. List<ServiceProvider> getRecommendations(User user) {
3. // Simple popularity-based recommendations
4. return providers
5. .where((p) => p.rating >= 4.0)
6. .take(10)
7. .toList();
8. }
9. **Personalized Suggestions**
   * Based on booking history
   * Popular in area
   * Highly rated providers
10. **Recommendation Display**
    * "Recommended for you" section
    * Service suggestions
    * Provider highlights

**PHASE 4: SAFETY & EMERGENCY (Week 13-16)**

**Week 13: Emergency Features**

**Day 85-87: SOS System Foundation**

1. **Emergency Contacts Setup**
2. class EmergencyContact {
3. final String name;
4. final String phone;
5. final ContactType type; // family, police, medical
6. }
7. **SOS Button Implementation**
   * Prominent SOS button
   * Confirmation dialog
   * Emergency activation
8. **Emergency Alert System**
   * SMS to emergency contacts
   * Location sharing
   * Emergency status

**Day 88-91: Safety Features**

1. **Provider Verification Display**
   * Verification badges
   * ID verification status
   * Background check indicators
2. **Safety Tips and Guidelines**
   * Safety information screens
   * Best practices
   * Emergency procedures
3. **Incident Reporting**
   * Report inappropriate behavior
   * Safety concerns
   * Issue tracking

**Week 14: Real-Time Tracking**

**Day 92-94: Live Location**

1. **Real-Time Location Sharing**
2. Stream<Position> trackLocation() {
3. return Geolocator.getPositionStream(
4. locationSettings: LocationSettings(
5. accuracy: LocationAccuracy.high,
6. distanceFilter: 10,
7. ),
8. );
9. }
10. **Provider Tracking Screen**
    * Live map with provider location
    * Route visualization
    * ETA updates
11. **Location Privacy Controls**
    * Start/stop sharing
    * Privacy settings
    * Tracking notifications

**Day 95-98: Advanced Tracking**

1. **Geofencing**
   * Service area boundaries
   * Arrival notifications
   * Location-based triggers
2. **Route Optimization**
   * Best route suggestions
   * Traffic integration
   * Alternative routes
3. **Tracking History**
   * Service route history
   * Time tracking
   * Distance calculations

**Week 15: Payment Gateway Integration**

**Day 99-101: Mobile Banking**

1. **bKash Integration**
2. class BKashPayment {
3. Future<PaymentResult> initiatePayment(double amount) async {
4. // bKash payment integration
5. }
6. }
7. **Nagad Integration**
   * Payment gateway setup
   * Transaction handling
   * Payment confirmation
8. **Payment Method Management**
   * Add/remove payment methods
   * Default payment selection
   * Payment method validation

**Day 102-105: Payment Processing**

1. **Transaction Management**
   * Payment processing states
   * Transaction history
   * Payment receipts
2. **Refund System**
   * Refund requests
   * Refund processing
   * Refund status tracking
3. **Payment Security**
   * Secure payment flow
   * Transaction encryption
   * Fraud prevention

**Week 16: Admin and Analytics**

**Day 106-108: Basic Admin Panel**

1. **Admin Authentication**
   * Admin login system
   * Role-based access
   * Admin permissions
2. **User Management**
   * View users
   * Block/unblock users
   * User activity monitoring
3. **Provider Management**
   * Provider approval
   * Verification management
   * Provider monitoring

**Day 109-112: Analytics and Reporting**

1. **Basic Analytics**
2. class AnalyticsService {
3. static void trackEvent(String eventName, Map<String, dynamic> parameters) {
4. // Firebase Analytics or custom implementation
5. }
6. }
7. **Usage Statistics**
   * Booking statistics
   * User engagement metrics
   * Popular services tracking
8. **Performance Monitoring**
   * App performance metrics
   * Crash reporting
   * Error tracking

**PHASE 5: AI/ML INTEGRATION (Week 17-20)**

**Week 17: Image Recognition**

**Day 113-115: Camera Integration**

1. **Camera/Gallery Access**
2. Future<File?> pickImage(ImageSource source) async {
3. final ImagePicker picker = ImagePicker();
4. final XFile? image = await picker.pickImage(source: source);
5. return image != null ? File(image.path) : null;
6. }
7. **Image Upload System**
   * Image compression
   * Cloud storage upload
   * Upload progress tracking
8. **Image Preview and Editing**
   * Image preview screen
   * Basic editing tools
   * Image annotation

**Day 116-119: AI Image Classification**

1. **TensorFlow Lite Integration**
2. class ImageClassifier {
3. late Interpreter interpreter;
5. Future<void> loadModel() async {
6. interpreter = await Interpreter.fromAsset('model.tflite');
7. }
9. List<double> classify(File image) {
10. // Image classification logic
11. }
12. }
13. **Problem Detection Model**
    * Pre-trained model integration
    * Classification results processing
    * Confidence score handling
14. **Service Suggestion from Image**
    * Map classification to services
    * Suggest relevant providers
    * Display recommendations

**Week 18: Advanced AI Chatbot**

**Day 120-122: Natural Language Processing**

1. **Improved Chatbot Logic**
2. class AdvancedChatbot {
3. Future<String> processMessage(String message) async {
4. // NLP processing
5. final intent = await detectIntent(message);
6. final entities = await extractEntities(message);
7. return generateResponse(intent, entities);
8. }
9. }
10. **Intent Recognition**
    * Service inquiry handling
    * Booking assistance
    * Problem troubleshooting
11. **Context Awareness**
    * Conversation history
    * User preferences
    * Previous interactions

**Day 123-126: Chatbot Integration**

1. **Multi-turn Conversations**
   * Context maintenance
   * Follow-up questions
   * Conversation flow
2. **Chatbot Training Interface**
   * Feedback collection
   * Response rating
   * Continuous improvement
3. **Voice Integration**
   * Speech-to-text
   * Text-to-speech
   * Voice commands

**Week 19: Recommendation System**

**Day 127-129: User Behavior Tracking**

1. **User Activity Logging**
2. class UserActivityTracker {
3. void trackServiceView(String serviceId) {
4. // Log service viewing behavior
5. }
7. void trackBookingComplete(String serviceCategory) {
8. // Log successful bookings
9. }
10. }
11. **Preference Learning**
    * Service preferences
    * Provider preferences
    * Time preferences
12. **Behavioral Analytics**
    * Usage patterns
    * Preference analysis
    * Trend identification

**Day 130-133: Smart Recommendations**

1. **Personalized Service Suggestions**
2. class RecommendationEngine {
3. List<ServiceProvider> getPersonalizedRecommendations(String userId) {
4. final userPreferences = getUserPreferences(userId);
5. final behaviorData = getUserBehavior(userId);
6. return calculateRecommendations(userPreferences, behaviorData);
7. }
8. }
9. **Location-based Recommendations**
   * Nearby popular services
   * Area-specific suggestions
   * Weather-based recommendations
10. **Time-sensitive Recommendations**
    * Seasonal services
    * Emergency services
    * Peak time recommendations

**Week 20: Final Integration and Polish**

**Day 134-136: System Integration**

1. **End-to-End Testing**
   * Complete user journeys
   * Integration testing
   * Performance testing
2. **Bug Fixes and Optimization**
   * Critical bug fixes
   * Performance improvements
   * Memory optimization
3. **UI/UX Polish**
   * Animation improvements
   * Visual consistency
   * User experience refinements

**Day 137-140: Production Preparation**

1. **App Store Preparation**
   * App metadata
   * Screenshots
   * Privacy policy
2. **Production Configuration**
   * Environment variables
   * API endpoints
   * Security configurations
3. **Launch Preparation**
   * Beta testing
   * Soft launch preparation
   * Marketing integration

**DEVELOPMENT CHECKLIST SUMMARY**

**Essential Features Implemented (120 Features)**

**Phase 1 - Foundation (24 features)**

* ✅ Project setup and infrastructure
* ✅ Authentication system
* ✅ Basic navigation and UI
* ✅ User profile management

**Phase 2 - Core Booking (24 features)**

* ✅ Service catalog and discovery
* ✅ Provider management
* ✅ Booking system
* ✅ Basic communication

**Phase 3 - Advanced Features (24 features)**

* ✅ Search and filtering
* ✅ Notifications
* ✅ User experience enhancements
* ✅ Basic AI integration

**Phase 4 - Safety & Emergency (24 features)**

* ✅ Emergency features
* ✅ Real-time tracking
* ✅ Payment integration
* ✅ Admin panel basics

**Phase 5 - AI/ML Integration (24 features)**

* ✅ Image recognition
* ✅ Advanced chatbot
* ✅ Recommendation system
* ✅ Final polish and integration

**Technical Debt Management**

**Weekly Reviews:**

* Code quality assessment
* Performance monitoring
* Security vulnerability checks
* Architecture compliance review

**Continuous Improvement:**

* Refactoring sessions
* Documentation updates
* Test coverage improvement
* Performance optimization

**Quality Assurance Throughout**

**Testing Strategy:**

* Unit tests for business logic
* Widget tests for UI components
* Integration tests for complete flows
* Performance tests for optimization

**Code Review Process:**

* Feature branch development
* Pull request reviews
* Code quality checks
* Documentation updates

This sequential implementation plan provides a clear roadmap to build SmartSheba from basic infrastructure to a fully-featured, AI-powered local service marketplace, with each feature building upon the previous ones to create a robust and scalable application.