

PawCare: Trusted Pet Discovery & Monitoring Platform

Team: Retrievers

Project: Kalvium Sprint #2 - Simulated Work Environment

Technology Stack: Flutter + Firebase

Date: January 30, 2026

Executive Summary

PawCare is a mobile application designed to solve the critical trust and transparency gap in urban pet care services. Pet owners struggle to find reliable caregivers and stay updated about their pets during walks or care sessions. PawCare addresses this by providing verified caregiver profiles, secure authentication, and real-time activity monitoring through a seamless Flutter-Firebase architecture.

This document outlines the complete product strategy, technical architecture, and implementation roadmap for the Kalvium Sprint #2 project, serving as both a development guide and presentation resource.

Problem Analysis

Current Pain Points

- **Trust Deficit:** Pet owners cannot verify caregiver credentials or background, leading to anxiety and hesitation
- **Information Blackout:** No real-time updates during pet walks or care sessions, leaving owners worried
- **Discovery Challenges:** Difficulty finding available, qualified caregivers in their locality
- **Safety Concerns:** Lack of identity verification increases risk of fraud or negligence
- **Communication Gaps:** Poor coordination between owners and caregivers regarding schedules and pet needs

Market Opportunity

Urban pet ownership is growing rapidly in India, with millions of working professionals needing reliable pet care services. The market lacks a technology-driven, trust-focused solution that combines discovery with monitoring capabilities.

App Concept & Value Proposition

Core Concept

PawCare is a **two-sided marketplace** connecting pet owners with verified caregivers, enhanced with real-time monitoring capabilities. Think of it as "Urban Company for Pet Care" with built-in trust and transparency features.

Unique Value Propositions

For Pet Owners:

- **Verified Caregivers:** Identity-verified profiles with ratings and reviews
- **Real-Time Peace of Mind:** Live updates, photos, and activity logs during care sessions
- **Easy Discovery:** Filter by location, availability, ratings, and pet specialization
- **Secure Bookings:** In-app scheduling with payment protection

For Caregivers:

- **Flexible Income:** Earn by offering pet care services on their schedule
- **Professional Profile:** Showcase experience, certifications, and client reviews
- **Trust Building:** Verification badge increases booking likelihood
- **Direct Client Access:** Connect with pet owners without intermediaries

User Personas

Persona 1: The Busy Professional (Pet Owner)

Name	Priya Sharma
Age	28 years
Occupation	Software Engineer at tech startup
Location	Pune, Maharashtra
Pet	Golden Retriever named Bruno (2 years old)
Pain Points	Works 9-7 with long commute, Bruno needs daily walks, Previous walker was unreliable, Worries about Bruno's safety during walks
Goals	Find trustworthy walker with good reviews, Receive photos and updates during walks, Flexible booking for weekdays, Budget: ₹300-500 per walk
Tech Comfort	High - Uses multiple apps daily, Comfortable with digital payments

Table 1: Pet Owner Persona

Persona 2: The Pet Care Professional (Caregiver)

Name	Rahul Verma
Age	24 years
Occupation	Part-time pet walker and dog trainer
Location	Pimpri-Chinchwad, Pune
Experience	3 years walking dogs, Certified in basic pet first aid
Pain Points	Difficult to find clients consistently, Clients don't trust new walkers without platform backing, No easy way to showcase credentials, Payment disputes with some clients
Goals	Build professional reputation with verified profile, Get 3-4 regular clients for stable income, Earn ₹25,000-35,000 monthly, Showcase certifications and experience
Tech Comfort	Moderate - Uses WhatsApp and basic apps, Willing to learn new platforms

Table 2: Caregiver Persona

Technical Architecture

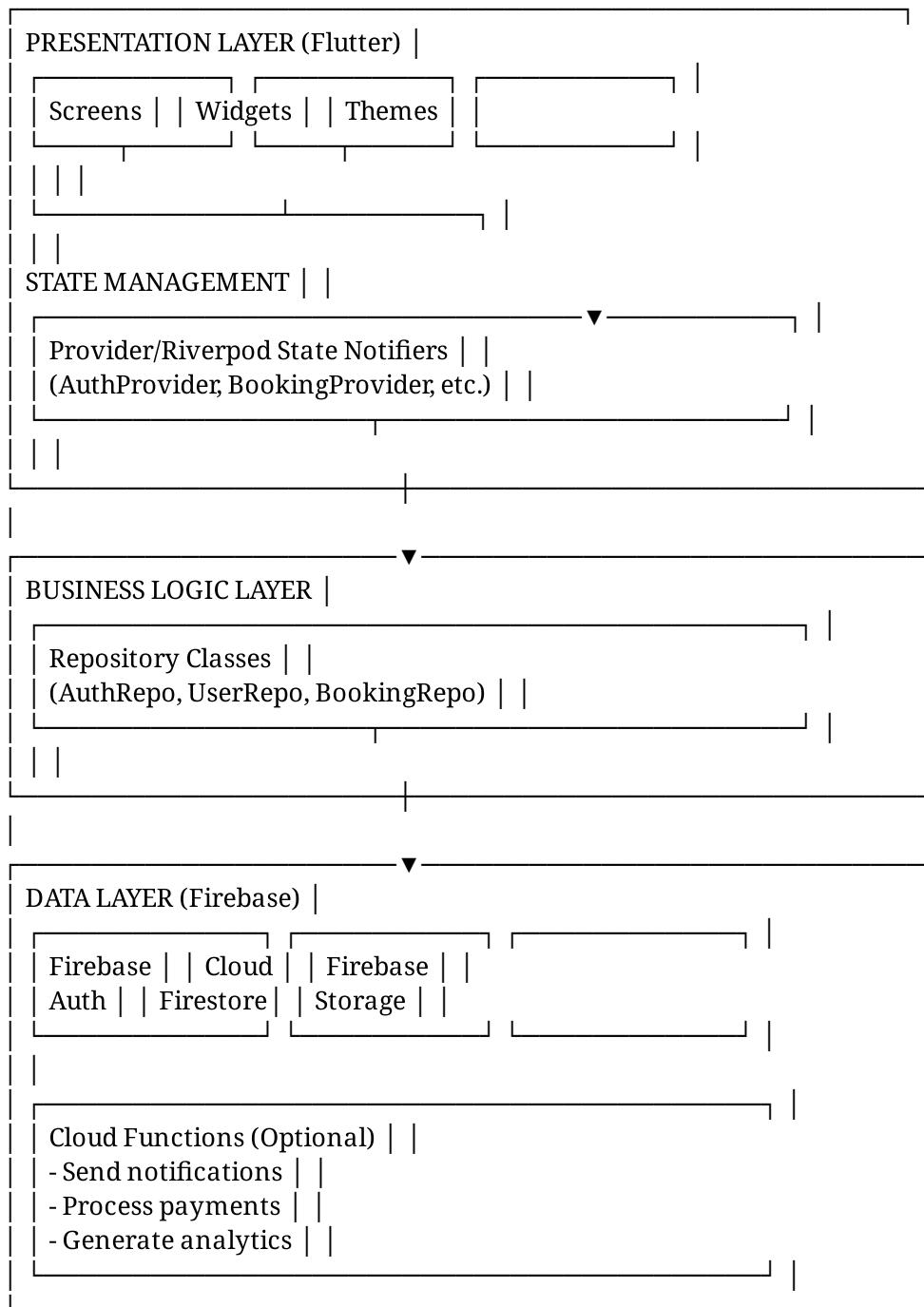
Technology Stack Overview

Layer	Technology	Purpose
Frontend	Flutter (Dart)	Cross-platform mobile UI development
Authentication	Firebase Auth	User identity verification and session management
Database	Cloud Firestore	Real-time NoSQL database for all app data
Storage	Firebase Storage	Image uploads (profiles, pet photos, activity updates)
Backend Logic	Cloud Functions	Server-side operations (notifications, calculations)
State Management	Provider/Riverpod	App state handling and data flow

Table 3: Technology Stack Breakdown

Flutter + Firebase Integration Architecture

Architecture Pattern: MVVM (Model-View-ViewModel) with Repository Pattern



Component Responsibilities

Presentation Layer (Flutter Screens & Widgets):

- Displays UI to users
- Captures user input (taps, forms, gestures)
- Listens to state changes from Providers

- Shows loading states and error messages

State Management (Provider/Riverpod):

- Holds app state (user data, bookings, caregiver lists)
- Notifies UI when data changes
- Manages loading and error states
- Caches data to reduce Firebase reads

Business Logic (Repositories):

- Handles all Firebase operations (CRUD)
- Implements business rules (validation, filtering)
- Transforms Firebase data into app models
- Error handling and retry logic

Data Layer (Firebase Services):

- Firebase Auth: User authentication and session management
- Firestore: Real-time database with offline support
- Storage: Image and file hosting
- Cloud Functions: Backend automation and triggers

Data Flow Example: Booking a Caregiver

1. User taps "Book Now" button on caregiver profile (UI)
2. BookingScreen calls BookingProvider.createBooking()
3. Provider calls BookingRepository.createBooking(data)
4. Repository validates data and writes to Firestore
5. Firestore triggers Cloud Function to notify caregiver
6. Repository returns success/failure to Provider
7. Provider updates state and notifies UI
8. UI shows success message or error dialog

Core Features for MVP

Feature 1: User Authentication & Profile Management

Owner Features:

- Sign up with email/password or Google Sign-In (Firebase Auth)
- Create profile with name, location, phone number
- Add pet profiles (name, breed, age, photo, special needs)
- Verify phone number via OTP
- Edit profile and pet information

Caregiver Features:

- Sign up with email and government ID upload
- Create professional profile (experience, certifications, hourly rate)
- Upload verification documents (Aadhaar/PAN for ID proof)
- Set availability schedule (days/hours)

- Add service areas (neighborhoods covered)

Technical Implementation:

- Firebase Auth for authentication
- Firestore for storing user profiles
- Firebase Storage for profile pictures and documents
- Provider for managing auth state globally

Feature 2: Caregiver Discovery & Search

Functionality:

- Browse verified caregivers with profile cards
- Filter by location (within 5km radius)
- Filter by rating (4+ stars)
- Filter by availability (date/time)
- Filter by pet type specialization (dogs, cats, etc.)
- Search by name or service type
- View detailed caregiver profiles with reviews

Profile Information Display:

- Profile photo and verification badge
- Name, experience years, and bio
- Average rating and total reviews
- Hourly rate and service types
- Distance from user location
- Availability calendar
- Sample photos from previous walks

Technical Implementation:

- Firestore queries with compound indexes
- GeoPoint data type for location-based filtering
- Pagination for efficient data loading
- Provider for search state management

Feature 3: Booking & Scheduling

Booking Flow:

1. Select caregiver from search results
2. Choose service type (walk, daycare, overnight stay)
3. Select date and time slot
4. Add special instructions for the caregiver
5. Confirm booking with total price calculation
6. Receive booking confirmation notification

Schedule Management:

- View upcoming bookings on calendar view
- View booking history with status

- Modify or cancel bookings (with policy)
- Track booking status (pending, confirmed, in-progress, completed)

Technical Implementation:

- Firestore transactions for booking creation
- Cloud Functions to check caregiver availability
- Provider for booking state
- Push notifications using Firebase Cloud Messaging

Feature 4: Real-Time Activity Monitoring

During Active Sessions:

- Caregiver can upload live photos during walk
- Add text updates ("Playing at park", "Had water break")
- Log walk start time and end time
- Owner receives instant notifications for updates
- View all updates in activity feed

Activity Log:

- Timestamped entries for each update
- Photo gallery from the session
- Total walk duration display
- Distance covered (future scope)
- Pet behavior notes from caregiver

Technical Implementation:

- Firestore real-time listeners for instant updates
- Firebase Storage for photo uploads with compression
- StreamBuilder widgets for reactive UI
- Provider for activity state management

Feature 5: Ratings & Reviews

Post-Session Review:

- Owner rates caregiver (1-5 stars) after session
- Written review with feedback
- Optional photo upload with review
- Caregiver can respond to reviews
- Reviews displayed on caregiver profile

Review Analytics:

- Average rating calculation
- Total review count
- Recent reviews highlighted
- Rating breakdown (5-star, 4-star, etc.)

Technical Implementation:

- Firestore subcollection for reviews
- Cloud Functions to update average ratings
- Moderation flags for inappropriate content

Feature 6: In-App Messaging

Communication:

- Chat between owner and caregiver before booking
- Share pet care instructions
- Ask questions about services
- Coordinate pickup/dropoff details
- Message history stored in Firestore

Technical Implementation:

- Firestore for real-time chat messages
- StreamBuilder for chat UI
- Push notifications for new messages
- Provider for chat state

User Flow & Screen List

Owner User Journey

Onboarding Flow:

1. **Splash Screen** - App logo and loading
2. **Welcome Screen** - Introduction with "Get Started" CTA
3. **Role Selection** - Choose "I'm a Pet Owner" or "I'm a Caregiver"
4. **Sign Up Screen** - Email/password or Google Sign-In
5. **Profile Setup** - Name, location, phone number
6. **Pet Profile Creation** - Add pet details and photo
7. **Home Screen** - Main dashboard

Core Booking Flow:

1. **Home Screen** - Search bar and featured caregivers
2. **Search Results** - List of caregivers with filters
3. **Caregiver Profile** - Detailed view with reviews and book button
4. **Booking Form** - Select service, date, time, and instructions
5. **Booking Confirmation** - Summary with "Confirm Booking" button
6. **Booking Success** - Confirmation message with details
7. **My Bookings** - List of upcoming and past bookings

Monitoring Flow:

1. **Active Session** - Real-time activity feed during walk
2. **Activity Detail** - View photos and updates from caregiver
3. **Session Complete** - Summary of session with total time
4. **Review Screen** - Rate and review the caregiver

Caregiver User Journey

Onboarding Flow:

1. **Welcome Screen** - Introduction for caregivers
2. **Sign Up Screen** - Email/password registration
3. **Profile Setup** - Professional details (experience, rate, bio)
4. **Verification Upload** - Government ID and certification documents
5. **Availability Setup** - Set service areas and schedule
6. **Verification Pending** - Waiting for admin approval (simulated)
7. **Dashboard** - Caregiver home with bookings

Service Delivery Flow:

1. **Dashboard** - View booking requests
2. **Booking Detail** - Review owner and pet details
3. **Accept/Decline** - Confirm availability
4. **Upcoming Sessions** - Calendar view of confirmed bookings
5. **Start Session** - Begin walk with "Start" button
6. **Activity Update** - Upload photos and status messages
7. **End Session** - Complete walk with "End" button
8. **Session Summary** - View session details and earnings

Complete Screen List

Category	Screen Name	Key Components
Authentication		
Auth	Splash Screen	App logo, loading indicator
Auth	Welcome Screen	Hero image, Get Started button
Auth	Role Selection	Pet Owner / Caregiver cards
Auth	Sign Up	Email, password fields, Google Sign-In button
Auth	Login	Email, password, Forgot Password link
Auth	OTP Verification	Phone number input, OTP code entry
Profile Setup		
Profile	Owner Profile Setup	Name, location, phone, profile photo
Profile	Pet Profile Form	Pet name, breed, age, photo, special needs
Profile	Caregiver Profile Setup	Experience, rate, bio, certifications
Profile	Document Upload	ID proof, certificates upload
Profile	Availability Setup	Service areas, available days/hours
Owner Screens		
Owner	Home Dashboard	Search bar, featured caregivers, quick actions
Owner	Search Results	Caregiver cards, filter options, sort options
Owner	Caregiver Profile	Photo, rating, reviews, services, availability
Owner	Booking Form	Service type, date/time picker, instructions
Owner	Booking Confirmation	Summary, total price, confirm button
Owner	My Bookings	Upcoming and past tabs, booking cards
Owner	Booking Detail	Full booking info, contact caregiver, cancel
Owner	Active Session Monitor	Real-time activity feed, photos, updates
Owner	Review Screen	Star rating, text review, photo upload
Caregiver Screens		

Caregiver	Dashboard	Booking requests, upcoming sessions, earnings
Caregiver	Booking Request Detail	Owner and pet info, accept/decline buttons
Caregiver	My Schedule	Calendar view, session list
Caregiver	Session Detail	Pet info, owner contact, start session button
Caregiver	Active Session	Update form, photo upload, notes
Caregiver	Session Summary	Duration, earnings, owner info
Caregiver	Earnings	Total earnings, transaction history
Common Screens		
Common	Chat Screen	Message list, text input, send button
Common	Notifications	List of all app notifications
Common	Profile View	User details, edit button, logout
Common	Settings	App preferences, privacy, terms

Table 4: Complete Screen List with Components

Database Structure (Firestore Collections)

Collection: users

Document ID: {userId} (auto-generated by Firebase Auth)

```
{
  "userId": "abc123",
  "email": "priya@example.com",
  "role": "owner",
  "name": "Priya Sharma",
  "phone": "+919876543210",
  "phoneVerified": true,
  "profilePhoto": "gs://bucket/users/abc123.jpg",
  "location": {
    "address": "Hinjewadi Phase 1, Pune",
    "geopoint": {
      "latitude": 18.5916,
      "longitude": 73.7331
    }
  },
  "createdAt": "2026-01-15T10:30:00Z",
  "lastActive": "2026-01-30T14:20:00Z"
}
```

Collection: caregivers

Document ID: {userId} (same as user document for caregivers)

```
{  
  "userId": "xyz789",  
  "bio": "Passionate dog lover with 3 years of experience",  
  "experienceYears": 3,  
  "hourlyRate": 400,  
  "servicesOffered": ["walking", "daycare", "training"],  
  "petTypesHandled": ["dogs", "cats"],  
  "certifications": [  
    {  
      "name": "Pet First Aid",  
      "issuedBy": "Red Cross",  
      "documentUrl": "gs://bucket/certs/xyz789_cert1.pdf"  
    }  
  ],  
  "verificationStatus": "verified",  
  "verificationDocuments": {  
    "idProof": "gs://bucket/docs/xyz789_aadhaar.jpg",  
    "idProofType": "aadhaar"  
  },  
  "serviceAreas": [  
    {  
      "name": "Hinjewadi",  
      "geopoint": {  
        "latitude": 18.5916,  
        "longitude": 73.7331  
      },  
      "radiusKm": 5  
    }  
  ],  
  "availability": {  
    "monday": ["09:00-12:00", "15:00-18:00"],  
    "tuesday": ["09:00-12:00", "15:00-18:00"],  
    "wednesday": ["09:00-12:00"],  
    "thursday": ["09:00-12:00", "15:00-18:00"],  
    "friday": ["09:00-12:00", "15:00-18:00"],  
    "saturday": ["10:00-16:00"],  
    "sunday": []  
  },  
  "stats": {  
    "totalBookings": 45,  
    "completedBookings": 42,  
    "cancelledBookings": 3,  
    "averageRating": 4.7,  
    "totalReviews": 38  
  },  
  "isActive": true,
```

```
"createdAt": "2025-10-20T08:00:00Z"  
}
```

Collection: pets

Document ID: Auto-generated

```
{  
  "petId": "pet001",  
  "ownerId": "abc123",  
  "name": "Bruno",  
  "type": "dog",  
  "breed": "Golden Retriever",  
  "age": 2,  
  "weight": 30,  
  "gender": "male",  
  "photo": "gs://bucket/pets/pet001.jpg",  
  "specialNeeds": "Scared of loud noises, prefers quiet parks",  
  "medicalInfo": "Vaccinated, no allergies",  
  "createdAt": "2026-01-15T11:00:00Z"  
}
```

Collection: bookings

Document ID: Auto-generated

```
{  
  "bookingId": "bk2026001",  
  "ownerId": "abc123",  
  "caregiverId": "xyz789",  
  "petId": "pet001",  
  "serviceType": "walking",  
  "status": "completed",  
  "scheduledDate": "2026-01-28",  
  "scheduledTime": "16:00",  
  "duration": 60,  
  "specialInstructions": "Please take Bruno to the dog park",  
  "pricing": {  
    "hourlyRate": 400,  
    "totalHours": 1,  
    "totalAmount": 400  
  },  
  "sessionData": {  
    "startTime": "2026-01-28T16:05:00Z",  
    "endTime": "2026-01-28T17:10:00Z",  
    "actualDuration": 65  
  },  
  "createdAt": "2026-01-25T12:30:00Z",  
  "updatedAt": "2026-01-28T17:10:00Z"  
}
```

Subcollection: bookings/{bookingId}/activities

Document ID: Auto-generated

```
{  
  "activityId": "act001",  
  "bookingId": "bk2026001",  
  "type": "photo",  
  "message": "Playing fetch at the park!",  
  "photoUrl": "gs://bucket/activities/act001.jpg",  
  "timestamp": "2026-01-28T16:25:00Z",  
  "uploadedBy": "xyz789"  
}
```

Collection: reviews

Document ID: Auto-generated

```
{  
  "reviewId": "rev001",  
  "bookingId": "bk2026001",  
  "caregiverId": "xyz789",  
  "ownerId": "abc123",  
  "rating": 5,  
  "comment": "Rahul was amazing with Bruno! Sent lots of photos and Bruno came home happy and tired. Highly recommend!",  
  "photos": ["gs://bucket/reviews/rev001_1.jpg"],  
  "caregiversResponse": "Thank you Priya! Bruno is a joy to walk with!",  
  "createdAt": "2026-01-28T18:00:00Z"  
}
```

Collection: chats

Document ID: Composite of userIds (e.g., "abc123_xyz789")

```
{  
  "chatId": "abc123_xyz789",  
  "participants": ["abc123", "xyz789"],  
  "lastMessage": "See you tomorrow at 4 PM!",  
  "lastMessageTime": "2026-01-27T20:15:00Z",  
  "unreadCount": {  
    "abc123": 0,  
    "xyz789": 2  
  }  
}
```

Subcollection: chats/{chatId}/messages

Document ID: Auto-generated

```
{  
  "messageId": "msg001",  
  "senderId": "abc123",  
  "content": "Hello!"  
}
```

```
"text": "Hi! Can you take Bruno to the riverside park tomorrow?",  
"timestamp": "2026-01-27T20:10:00Z",  
"read": true  
}
```

Firestore Indexes Required

- **caregivers**: Composite index on (verificationStatus, stats.averageRating, isActive)
- **bookings**: Composite index on (ownerId, status, scheduledDate)
- **bookings**: Composite index on (caregiverId, status, scheduledDate)
- **reviews**: Composite index on (caregiverId, createdAt)

Security Rules Principles

- Users can only read/write their own user documents
- Caregivers can only update their own caregiver profiles
- Pet owners can only create/edit their own pets
- Both parties can read bookings they're involved in
- Only caregivers can create activity updates for their sessions
- Reviews can only be created by booking participants
- Chat messages readable only by participants

State Management Strategy

Why Provider/Riverpod?

Provider Benefits:

- Simple to learn for Kalvium students
- Officially recommended by Flutter team
- Good performance with rebuild optimization
- Works well with Firebase real-time updates
- Less boilerplate than Bloc or Redux

Riverpod Benefits (Advanced Option):

- Compile-time safety (catches errors before runtime)
- Better testing capabilities
- No BuildContext dependency
- More flexible dependency injection

Provider Architecture Example

AuthProvider (Authentication State):

```
class AuthProvider extends ChangeNotifier {  
User? _user;  
bool _isLoading = false;  
String? _error;  
  
User? get user => _user;  
bool get isLoading => _isLoading;
```

```

bool get isAuthenticated => _user != null;

Future<void> signIn(String email, String password) async {
  _isLoading = true;
  _error = null;
  notifyListeners();

  try {
    final credential = await FirebaseAuth.instance
        .signInWithEmailAndPassword(email: email, password: password);
    _user = credential.user;
  } catch (e) {
    _error = e.toString();
  }

  _isLoading = false;
  notifyListeners();
}

Future<void> signOut() async {
  await FirebaseAuth.instance.signOut();
  _user = null;
  notifyListeners();
}

```

CaregiverProvider (Caregiver Discovery):

```

class CaregiverProvider extends ChangeNotifier {
  List<Caregiver> _caregivers = [];
  bool _isLoading = false;
  String? _error;

  List<Caregiver> get caregivers => _caregivers;
  bool get isLoading => _isLoading;

  Future<void> fetchCaregivers({String? location, double? minRating}) async {
    _isLoading = true;
    notifyListeners();
  }

```

```

try {
  Query query = FirebaseFirestore.instance.collection('caregivers')
      .where('verificationStatus', isEqualTo: 'verified')
      .where('isActive', isEqualTo: true);

```

```

        if (minRating != null) {
            query = query.where('stats.averageRating', isGreaterThanOrEqualTo: minRating);
        }

        final snapshot = await query.get();
        _caregivers = snapshot.docs
            .map((doc) => Caregiver.fromFirestore(doc))
            .toList();
    } catch (e) {
        _error = e.toString();
    }

    _isLoading = false;
    notifyListeners();
}

}

```

Using Providers in UI

```

// Provide at app root
MultiProvider(
providers: [
    ChangeNotifierProvider(create: () => AuthProvider()),
    ChangeNotifierProvider(create: () => CaregiverProvider()),
    ChangeNotifierProvider(create: (_) => BookingProvider()),
],
child: MyApp(),
)

// Consume in widgets
class HomeScreen extends StatelessWidget {
@override
Widget build(BuildContext context) {
final caregiverProvider = Provider.of<CaregiverProvider>(context);

if (caregiverProvider.isLoading) {
    return Center(child: CircularProgressIndicator());
}

return ListView.builder(
    itemCount: caregiverProvider.caregivers.length,

```

```
itemBuilder: (context, index) {
    final caregiver = caregiverProvider.caregivers[index];
    return CaregiverCard(caregiver: caregiver);
},
);

}

}
```

Implementation Roadmap

Sprint Plan (4 Weeks)

Week 1: Foundation & Authentication

- Set up Flutter project with Firebase
- Implement authentication (email/password, Google Sign-In)
- Create user profile screens (owner and caregiver)
- Set up Firestore collections and security rules
- Implement basic navigation structure

Week 2: Core Booking Features

- Build caregiver discovery and search UI
- Implement Firestore queries with filters
- Create booking form and confirmation flow
- Develop booking list screens
- Add basic chat functionality

Week 3: Real-Time Monitoring

- Implement activity update upload (photos and text)
- Create real-time activity feed UI
- Add Firebase Storage integration for photos
- Build session start/end functionality
- Implement push notifications for updates

Week 4: Polish & Testing

- Add ratings and reviews system
- Implement caregiver verification UI (simulated)
- Create dashboard widgets and statistics
- Test all user flows end-to-end
- Fix bugs and improve UI/UX
- Prepare presentation and documentation

Development Best Practices

Code Organization:

```
lib/
  └── main.dart
  └── models/
    └── user_model.dart
    └── caregiver_model.dart
    └── pet_model.dart
    └── booking_model.dart
    └── review_model.dart
  └── providers/
    └── auth_provider.dart
    └── caregiver_provider.dart
    └── booking_provider.dart
    └── activity_provider.dart
  └── repositories/
    └── auth_repository.dart
    └── user_repository.dart
    └── booking_repository.dart
    └── storage_repository.dart
  └── screens/
    └── auth/
    └── owner/
    └── caregiver/
    └── common/
  └── widgets/
    └── caregiver_card.dart
    └── booking_card.dart
    └── activity_update.dart
    └── custom_button.dart
  └── services/
    └── firebase_service.dart
    └── notification_service.dart
  └── utils/
    └── constants.dart
    └── validators.dart
  └── theme.dart
```

Firebase Integration Checklist:

- Add Firebase to Flutter project (google-services.json for Android, GoogleService-Info.plist for iOS)
 - Initialize Firebase in main.dart using FlutterFire CLI
 - Configure Firebase Authentication methods in console
 - Set up Firestore database in test mode initially
 - Create Storage buckets for photos and documents
 - Configure security rules before production
 - Set up indexes for complex queries
 - Enable Firebase Cloud Messaging for notifications
-

Future Scope & Enhancements

Phase 2 Features (Post-MVP)

GPS Live Tracking:

- Real-time location tracking during walks
- Walk route display on map (Google Maps integration)
- Distance and speed metrics
- Geofencing alerts if pet goes beyond safe zones

In-App Payments:

- Integrate Razorpay or Stripe payment gateway
- Secure in-app payment processing
- Automatic payment release after session completion
- Wallet system for caregivers
- Transaction history and invoices

Advanced Analytics:

- Pet activity dashboard (total walks, distance, time)
- Caregiver performance analytics
- Booking trends and insights
- Health tracking integration (steps, calories)
- Monthly reports for owners

Video Updates:

- 15-second video clips during walks
- Video storage optimization with compression
- Live video call option for owners to check in

Emergency Features:

- SOS button for emergencies during walks
- Emergency contact alerts
- Nearby vet finder with directions
- Pet insurance integration

Phase 3 Features (Advanced)

AI-Powered Features:

- Pet behavior analysis from photos/videos
- Automatic activity logging using ML (playing, resting, eating)
- Smart caregiver recommendations based on pet personality
- Predictive scheduling based on usage patterns

Community Features:

- Pet owner social network
- Share pet photos and stories

- Local pet events and meetups
- Pet playdate coordination

Subscription Model:

- Premium memberships with benefits (priority booking, discounts)
- Subscription plans for regular users
- Loyalty rewards program

Multi-Pet Support:

- Book multiple pets in single session
 - Group walk discounts
 - Multi-pet household management
-

Key Learning Outcomes

Technical Skills Developed

- **Flutter Development:** Building production-ready cross-platform mobile UI with Material Design
- **Firebase Integration:** Authentication, Firestore real-time database, Storage, and Cloud Functions
- **State Management:** Implementing Provider/Riverpod for scalable app architecture
- **Asynchronous Programming:** Handling Futures, Streams, and async/await patterns in Dart
- **NoSQL Database Design:** Structuring collections, subcollections, and denormalization strategies
- **Real-Time Features:** Implementing live updates with Firestore listeners and StreamBuilder
- **Image Handling:** Upload, compression, and display of images with Firebase Storage
- **User Authentication:** Implementing secure auth flows with email, social login, and phone OTP

Product & Design Skills

- **User Research:** Creating personas based on real user pain points
- **Information Architecture:** Designing intuitive navigation and user flows
- **Problem Solving:** Identifying trust and transparency gaps in marketplace
- **Feature Prioritization:** MVP vs future scope decision-making
- **Two-Sided Marketplace:** Understanding dynamics between service providers and consumers

Professional Skills

- **Technical Documentation:** Creating comprehensive project documentation
 - **System Architecture:** Designing scalable backend structures
 - **Presentation Skills:** Explaining technical concepts in simple language for viva
 - **Project Management:** Breaking down complex features into development sprints
 - **Teamwork:** Collaborating in 3-member team with defined roles
-

Viva & Presentation Tips

Expected Questions & Answers

Q1: Why did you choose Flutter over native development?

Answer: Flutter allows us to build for both iOS and Android with a single codebase, reducing development time by 50%. It provides hot reload for faster development, has rich pre-built widgets that speed up UI creation, and has excellent Firebase integration through FlutterFire packages. For a 4-week sprint project, Flutter's productivity benefits are ideal.

Q2: How does real-time monitoring work technically?

Answer: When a caregiver uploads a photo or update during a walk, it's stored in Firebase Storage and Firestore. The owner's app has a Firestore listener (StreamBuilder in Flutter) that automatically receives new updates without refreshing. This creates a live feed experience similar to Instagram Stories. Firestore's real-time capability means data syncs in milliseconds.

Q3: How do you ensure caregiver verification security?

Answer: Caregivers must upload government ID documents (Aadhaar/PAN) which are stored securely in Firebase Storage with restricted access rules. In production, this would trigger a Cloud Function to verify documents using OCR or send to admin dashboard for manual verification. The caregiver profile has a verificationStatus field that must be "verified" before they can accept bookings.

Q4: What if there's no internet during a walk?

Answer: Firebase has offline persistence enabled by default. Activity updates are queued locally on the caregiver's device and automatically sync when internet reconnects. Firestore's offline cache ensures users can view previously loaded data even without internet. For MVP, we rely on this built-in capability.

Q5: How do you prevent fake reviews?

Answer: Reviews can only be created by users who have a completed booking. Our Firestore security rules check that request.auth.uid matches the ownerId of the booking, and the booking status is "completed". This prevents anyone from posting reviews without actually using the service.

Q6: Explain your state management choice.

Answer: We use Provider because it's officially recommended by Flutter team, easy to learn for our sprint timeline, and sufficient for our app's complexity. Provider uses ChangeNotifier to notify widgets when data changes, causing automatic UI rebuilds. For example, when bookings are fetched from Firestore, BookingProvider calls notifyListeners() and all listening widgets rebuild with new data.

Q7: What are the scalability concerns?

Answer: For MVP with limited users, our current Firestore structure is sufficient. For scale, we'd need: (1) Cloud Functions to handle complex business logic instead of client-side code, (2) Firestore indexes for efficient queries with multiple filters, (3) Image compression before

upload to reduce Storage costs, (4) Pagination for loading large lists, and (5) Caching strategies to minimize Firestore reads.

Demo Script

Opening (30 seconds):

"Hi, we're Team Retrievers. Urban pet owners struggle with two problems: finding trustworthy caregivers and staying updated about their pets. PawCare solves this with verified caregiver profiles and real-time activity monitoring."

Owner Flow Demo (2 minutes):

1. Show sign up and pet profile creation
2. Search for caregivers with filters (location, rating)
3. View caregiver profile with reviews and verification badge
4. Create booking with date/time selection
5. Show real-time activity feed with photos during active session
6. Submit review after session

Caregiver Flow Demo (1.5 minutes):

1. Show caregiver profile setup with verification
2. View booking request and accept
3. Start session and upload activity updates
4. End session and view earnings

Technical Architecture (1 minute):

Show architecture diagram and explain Flutter → Provider → Repository → Firebase flow briefly.

Closing (30 seconds):

"Our MVP delivers core discovery and monitoring features. Future scope includes GPS tracking, payments, and AI-powered insights. Thank you!"

Conclusion

PawCare addresses a real market need by building trust and transparency into the pet care marketplace. The Flutter-Firebase technology stack provides rapid development capabilities while maintaining scalability for future growth. The MVP focuses on core value propositions—verified discovery and real-time monitoring—while leaving room for sophisticated features in later phases.

This project demonstrates the complete product development lifecycle from problem identification through technical implementation, preparing the team for real-world software engineering challenges.

Team Retrievers is ready to build PawCare and revolutionize urban pet care! ☺

Appendix A: Useful Resources

Flutter Documentation:

- Official Flutter Docs: <https://docs.flutter.dev>
- Flutter Codelabs: <https://docs.flutter.dev/codelabs>
- Dart Language Tour: <https://dart.dev/guides/language/language-tour>

Firebase Resources:

- FlutterFire Documentation: <https://firebase.flutter.dev>
- Firebase Console: <https://console.firebaseio.google.com>
- Firestore Data Modeling: <https://firebase.google.com/docs/firestore/data-model>

State Management:

- Provider Package: <https://pub.dev/packages/provider>
- Riverpod Documentation: <https://riverpod.dev>

UI/UX Design:

- Material Design Guidelines: <https://m3.material.io>
- Flutter Widget Catalog: <https://docs.flutterdev/development/ui/widgets>

Appendix B: Team Roles & Responsibilities

Suggested Division for 3-Member Team:

Team Member 1	Authentication & Profile Management
Responsibilities	Firebase Auth integration, Sign up/Login screens, User profile screens, Pet profile creation, Caregiver profile setup, AuthProvider implementation

Team Member 2	Discovery & Booking System
Responsibilities	Caregiver search and filters, Firestore queries and indexes, Booking form and flow, My Bookings screen, CaregiverProvider and BookingProvider, Chat functionality

Team Member 3	Real-Time Monitoring & Reviews
Responsibilities	Activity update upload, Real-time feed with StreamBuilder, Firebase Storage integration, Review and rating system, ActivityProvider, Push notifications setup

Table 5: Team Role Distribution

Shared Responsibilities:

- UI/UX design decisions
- Database structure planning
- Testing and bug fixes
- Documentation and presentation preparation