TourPal

Technical Report

Course: Intro to Mobile Development (ICM)

Project: Travel Guide Platform

Authors:

José Cerqueira (76758)
Panos Lekos (128625)

Date: June 13, 2025

Contents

1 Overview

TourPal is a Flutter mobile app that connects travelers with local tour guides through real-time GPS coordination and live tour experiences. The app uses a dual-role system where users can be both travelers and guides, featuring smart booking, real-time tracking, and digital journaling.

Built with Flutter 3.7.2+ and Firebase, the app demonstrates Clean Architecture principles, BLoC state management, and comprehensive real-time features. The main innovation is live tour coordination where guides and travelers can track each other's locations and progress through tour stops in real-time.

2 Technical Architecture

2.1 Architecture Pattern

The app follows Clean Architecture with three layers:

```
1
2       Presentation (BLoC + UI)
3
4       Domain (Business Logic)
5
6       Data (Firebase + Repositories)
7
```

Each feature is organized as data/domain/presentation modules. The presentation layer uses BLoC for state management, domain contains business logic and repository interfaces, and data implements Firebase integration and repository patterns.

2.2 State Management

We use BLoC pattern with flutter_bloc 8.1.6 for predictable state management. The app also leverages Provider 6.1.1 for dependency injection and GetIt 8.0.0 for service location. Dartz 0.10.1 provides functional programming utilities for error handling.

3 Technology Stack

3.1 Core Technologies

Technology	Version	Purpose
Flutter	3.7.2+	Cross-platform framework
Dart	3.7.2+	Programming language
Firebase Suite	Latest	Backend services
Google Maps Flutter	2.5.0	Location services

3.2 Key Dependencies

State Management & Architecture:

- flutter_bloc 8.1.6 - BLoC pattern implementation
- provider 6.1.1 - Dependency injection
- get_it 8.0.0 - Service locator
- dartz 0.10.1 - Functional programming utilities
- equatable 2.0.5 - Value equality

Firebase Integration:

- firebase_core 3.13.1 - Core Firebase functionality
- firebase_auth 5.5.4 - Authentication
- cloud_firestore 5.6.8 - Real-time database
- firebase_storage 12.4.6 - File storage
- firebase_messaging 15.2.6 - Push notifications
- firebase_app_check 0.3.2+6 - App verification

Maps & Location:

- google_maps_flutter 2.5.0 - Map integration
- geolocator 14.0.1 - GPS location services
- geocoding 3.0.0 - Address conversion
- flutter_polyline_points 2.0.0 - Route drawing
- google_places_flutter 2.0.9 - Places API

Media & Storage:

- image_picker 1.0.7 - Camera and gallery access
- flutter_image_compress 2.1.0 - Image optimization
- cached_network_image 3.4.1 - Image caching
- path_provider 2.1.1 - File system access

4 Database Design and Security

4.1 Firestore Collections

The database has thirteen main collections optimized for real-time tour coordination:

- **users**: User profiles with dual-role support (travelers/guides)
- **guides**: Guide-specific information with availability subcollections
- **tourPlans**: Reusable tour templates with places subcollection
- **tourInstances**: Scheduled tour sessions with participants
- **bookings**: Reservation management and tour lifecycle
- **tourSessions**: Live tour coordination with participant locations subcollection
- **tourProgress**: Tour navigation and progress tracking
- **userLocations**: Real-time location sharing during active tours
- **tourJournals**: Traveler experiences with entries subcollection
- **conversations**: Guide-traveler communication with messages subcollection
- **reviews**: Tour feedback and rating system

4.1.1 Subcollections

Several collections include subcollections for hierarchical data organization:

- **guides/availableTimes**: Guide availability schedules
- **guides/unavailableSlots**: Guide blocked time periods
- **tourPlans/places**: Tour location details and itineraries
- **tourSessions/participantLocations**: Real-time location data
- **tourJournals/entries**: Individual journal entries with photos
- **conversations/messages**: Chat message history

4.2 Security Rules

Firestore security implements comprehensive role-based access with helper functions for authentication and ownership verification:

```
function isAuthenticated() {
  return request.auth != null;
}

function isOwner(userId) {
  return request.auth.uid == userId;
}

function isGuide() {
  return isAuthenticated() &&
         exists(/databases/$(database)/documents/users/$(request.auth.
    uid)) &&
         get(/databases/$(database)/documents/users/$(request.auth.uid))
    .data.isGuide == true;
}

function isSessionParticipant(sessionData) {
  return isAuthenticated() &&
         (request.auth.uid == sessionData.guideId ||
          request.auth.uid == sessionData.travelerId);
}
```

Listing 1: Core Security Functions

4.2.1 Access Control Patterns

The security model implements different access patterns for various use cases:

Public Discovery: Collections like `users`, `guides`, `tourPlans`, and `reviews` allow public read access to enable tour discovery and guide selection.

Participant-Only Access: `tourSessions`, `conversations`, and location tracking collections restrict access to authenticated tour participants.

Owner-Controlled: `bookings` and `tourJournals` allow only owners to create content, with flexible read permissions for tour coordination.

Draft Support: Tour creation supports draft status with relaxed validation, allowing iterative tour development before publication.

The security rules balance open discovery with strict privacy controls, enabling collaborative tour experiences while protecting sensitive location and personal data.

5 Key Features

5.1 Real-Time Tour Coordination

The most innovative feature is live tour tracking with custom map markers. Guides and travelers share GPS locations during active tours, with numbered markers showing visit order.

Guides can advance tour progression and handle group coordination scenarios through real-time Firebase synchronization.

5.2 Dual-Role Architecture

Single accounts support both traveler and guide modes with seamless role switching. Users can explore as travelers while also offering guide services, creating a community-driven platform. The guide registration system includes availability management and specialized UI components.

6 Security and Privacy

API keys and sensitive configuration use flutter_dotenv 5.2.1 with .env files excluded from version control. Multi-layer input validation prevents malicious input. Location tracking uses granular permissions through permission_handler 12.0.0 with clear user consent.

Firebase provides transport encryption while the app implements secure token storage through shared_preferences 2.2.2 and session management.

7 Technical Achievements

7.1 Key Accomplishments

- Clean Architecture implementation with feature-based organization
- Real-time GPS coordination with conflict resolution
- Comprehensive Firebase integration across 6 services
- Complex state management for live tour experiences
- Custom map marker system with Canvas-based rendering
- Smart booking system with availability management
- Dual-role user system with seamless switching

7.2 Innovation Areas

The real-time tour coordination moves beyond static booking platforms to dynamic, collaborative experiences. Custom map markers provide intuitive visual feedback about tour progress. The dual-role system recognizes that users often want both traveler and guide capabilities.

Digital journaling transforms passive tour consumption into active memory creation with location-aware entries and real-time photo sharing.

8 Project Structure

8.1 Directory Organization

The project follows Flutter's standard structure with feature-based organization for scalability and maintainability:

```
tourpal/
 lib/                          # Main application code
    app.dart                   # App configuration and theming
    main.dart                  # Application entry point
    core/                      # Shared utilities and services
    features/                  # Feature-based modules
       auth/                  # Authentication system
       bookings/              # Booking management
       dashboard/             # Main application screens
       navigation/            # App navigation system
       profile/               # User profile management
       tours/                 # Tour management & live coordination
    models/                    # Shared data models
    repositories/              # Data access abstractions
 .env                          # Environment variables
 test/                         # Test files
 android/                      # Android platform files
 assets/                       # Static assets
 firestore.rules               # Database security rules
 storage.rules                 # Storage security rules
 pubspec.yaml                  # Dependencies and metadata
 README.txt                    # Setup instructions
```

8.2 Feature Module Structure

Each feature follows Clean Architecture principles with consistent organization:

```
features/feature_name/
 data/                         # Data layer
    datasources/               # Remote/local data sources
    models/                    # Data transfer objects
    repositories/              # Repository implementations
 domain/                       # Business logic layer
    repositories/              # Repository interfaces
    usecases/                  # Business use cases
 presentation/                 # UI layer
     bloc/                     # BLoC state management
     screens/                  # UI screens
     widgets/                  # Feature-specific widgets
```

This modular approach enables independent development, testing, and maintenance of features while maintaining clear separation between layers and dependencies.

9 Results and Learning

TourPal demonstrates modern Flutter development with advanced state management, real-time features, and comprehensive backend integration. The project showcases Clean Architecture benefits, Firebase real-time capabilities, and complex UI coordination.

Key learning areas include real-time application architecture, NoSQL database design, location services integration, and production-quality error handling. The custom map marker system demonstrates advanced Canvas manipulation and bitmap generation.

The project proves ability to integrate multiple complex systems (Firebase, Google Maps, real-time coordination) into a cohesive user experience while maintaining clean architecture principles.

10 Platform Support

Currently, TourPal supports Android and Windows platforms. The project structure includes comprehensive Android configuration with proper Firebase integration and Google Services setup. Windows support enables desktop development and testing workflows.

Future expansion could include iOS support with minimal architectural changes due to Flutter's cross-platform nature.

11 Conclusion

TourPal successfully combines modern mobile development practices with innovative travel technology. The real-time coordination features create new possibilities for collaborative travel experiences while the technical architecture provides a solid foundation for scaling.

The project demonstrates proficiency in Flutter development, Firebase integration, custom UI rendering, and real-time application design within a clean, maintainable architecture. The comprehensive feature set and robust technical implementation showcase advanced mobile development capabilities.