Nice Places App - Architecture Summary

🏗️ Overall Architecture

App Type: SwiftUI iOS location tracking app with Spotify-inspired dark theme
Architecture Pattern: MVVM with Observable managers
Key Technologies: Core Location, Photos Framework, MapKit, Contacts Framework, MessageUI

📁 File Structure & Feature Mapping

🎯 Core App Files (Always needed for context)

- **`LocationTrackerApp.swift`** - App entry point, scene configuration
- **`Color+Spotify.swift`** - UI theme colors (spotifyGreen, darkGray, etc.)

🧠 Managers Layer (Business Logic)

📍 Location Tracking & Management

- **`LocationManager.swift`** - Core Location integration, GPS tracking, geocoding
  - *When to upload*: Location permission issues, GPS accuracy, address formatting, Norwegian address format bugs
  - *Dependencies*: None (standalone)
- **`DataManager.swift`** - Location data persistence, CRUD operations
  - *When to upload*: Saving/loading locations, data persistence bugs, UserDefaults issues
  - *Dependencies*: `LocationData.swift`

📸 Photo & Media Management

- **`PhotoManager.swift`** - Photos framework integration, album management, thumbnail loading
  - *When to upload*: Photo saving bugs, album creation, thumbnail loading issues, PHAsset problems
  - *Dependencies*: None (standalone)

👤 User Profile & Emergency Contacts

- **`ProfileManager.swift`** - User profile data, emergency contact management
  - *When to upload*: Profile saving, emergency contact features, name extraction from device
  - *Dependencies*: `UserProfile.swift`

🗺️ Trip Organization & Auto-Save

- `TripManager.swift` - Trip creation, auto-save logic, Norwegian street detection
  - *When to upload*: Trip bugs, auto-save not working, street change detection, Norwegian address parsing
  - *Dependencies*: `Trip.swift`, `LocationData.swift`

📊 Models Layer (Data Structures)

🏷️ Core Data Models

- `LocationData.swift` - Location data structure with photos, comments, coordinates
  - *When to upload*: Location data structure changes, encoding/decoding issues
- `Trip.swift` - Trip data structure, auto-save configuration, trip statistics
  - *When to upload*: Trip data bugs, auto-save config issues, trip type problems
- `UserProfile.swift` - User profile structure, validation logic
  - *When to upload*: Profile data structure changes, validation bugs

---

🎨 Views Layer (UI Components)

📱 Main Application Views

- `ContentView.swift` - Main app screen, location tracking UI, auto-save integration
  - *When to upload*: Main screen bugs, auto-save UI issues, emergency button problems
  - *Dependencies*: ALL managers, `SpotifyLocationCard.swift`, `TripComponents.swift`
- `LocationDetailView.swift` - Individual location details, photo gallery
  - *When to upload*: Location detail bugs, photo gallery issues
  - *Dependencies*: `DataManager.swift`, `PhotoManager.swift`, most photo-related components

🧩 Reusable Components (`/Views/Components/`)

📷 Camera & Photo Components

- `CameraView.swift` - Camera integration, photo/video capture
- `PhotoGalleryView.swift` - Photo grid, thumbnail display, full-screen viewer
- `MediaLibraryPicker.swift` - Photo library access, PHPickerViewController wrapper
- *Upload for*: Camera bugs, photo capture issues, library access problems

🗺️ Map Components

- `MapView.swift` - Location maps, trip route visualization, map overlays
- *Upload for*: Map display bugs, route drawing issues, annotation problems

✏️ Location Editing Components

- `EditLocationSheet.swift` - Edit location comments, update location data
- `SaveLocationSheet.swift` - Save new locations with comments
- `LocationSelectionSheet.swift` - Select location for adding media
- *Upload for*: Location editing bugs, save sheet issues

📱 UI Display Components

- `SpotifyLocationCard.swift` - Main location display card with actions
- `SpotifyLocationRow.swift` - Location list item with swipe actions, photo thumbnails
- *Upload for*: Location display bugs, action button issues, thumbnail problems

👥 Contact & Profile Components

- `ContactPicker.swift` - Contacts framework integration, emergency contact selection
- *Upload for*: Contact picker bugs, permission issues, phone number formatting

🏠 Feature-Specific Views

👤 Profile Management (`/Views/Profile/`)

- `ProfileView.swift` - User profile editing, emergency contact setup, settings
- *Upload for*: Profile bugs, emergency contact setup, validation issues
- *Dependencies*: `ProfileManager.swift`, `ContactPicker.swift`

💾 Saved Locations (`/Views/SavedLocations/`)

- `SpotifySavedLocationsView.swift` - Saved locations list, bulk operations
- *Upload for*: Saved locations list bugs, mass operations, media adding
- *Dependencies*: `DataManager.swift`, `PhotoManager.swift`, location row components

🗺️ Trip Management (`/Views/Trips/`)

- `TripsView.swift` - Trip list, active trip management
- `TripDetailView.swift` - Individual trip details, route display, location management
- `TripSuggestionsView.swift` - Smart trip creation from existing locations
- `TripComponents.swift` - Trip UI components, auto-save configuration, trip creation
- *Upload for*: Trip features, auto-save settings, trip creation, route visualization
- *Dependencies*: `TripManager.swift`, `Trip.swift`, map components

🎯 Feature Implementation Guide

📍 For Location Tracking Issues:

```
Essential: LocationManager.swift, LocationData.swift
UI: ContentView.swift, SpotifyLocationCard.swift
Optional: Color+Spotify.swift (for theme)
```

📸 For Photo/Camera Features:

```
Essential: PhotoManager.swift, CameraView.swift, PhotoGalleryView.swift
Data: LocationData.swift (for photo identifiers)
UI: MediaLibraryPicker.swift, LocationDetailView.swift
```

🗺️ For Trip Features:

```
Essential: TripManager.swift, Trip.swift
UI: TripsView.swift, TripDetailView.swift, TripComponents.swift
Maps: MapView.swift (for route visualization)
Dependencies: LocationData.swift, DataManager.swift
```

🤖 For Auto-Save Functionality:

```
Essential: TripManager.swift, Trip.swift (AutoSaveConfiguration)
UI: TripComponents.swift (auto-save settings), ContentView.swift (auto-save logic)
Dependencies: LocationManager.swift (for location changes)
```

👥 For Profile/Emergency Features:

```
Essential: ProfileManager.swift, UserProfile.swift
UI: ProfileView.swift, ContactPicker.swift
Integration: ContentView.swift (emergency button logic)
```

🎨 For UI/Theme Issues:

```
Essential: Color+Spotify.swift
Components: Any specific UI component file
Theme: All files use Spotify color scheme
```

🐛 For Norwegian Address Formatting:

```
Essential: LocationManager.swift (formatAddress method)
Integration: TripManager.swift (extractNorwegianStreetName method)
```

🔄 Common Dependency Patterns

High-Impact Files (Changes affect multiple features):

1. `LocationData.swift` - Used by almost all location features
2. `ContentView.swift` - Main coordinator, integrates all managers
3. `Color+Spotify.swift` - UI theme used everywhere
4. **Manager files** - Core business logic for their respective domains

Standalone Components (Can be modified independently):

- Individual UI components in `/Views/Components/`
- Specific feature views (ProfileView, TripsView, etc.)
- Photo/camera related components

Cross-Feature Dependencies:

- **Trip ↔ Location**: `TripManager.swift` + `DataManager.swift` + `LocationData.swift`
- **Profile ↔ Sharing**: `ProfileManager.swift` affects sharing functionality across the app
- **Photo ↔ Location**: `PhotoManager.swift` + `LocationData.swift` for photo attachments

💡 Quick Reference for Common Tasks

Task	Essential Files	Optional Context
Fix location accuracy	`LocationManager.swift`	`ContentView.swift`
Add photo feature	`PhotoManager.swift`, `CameraView.swift`	`LocationData.swift`
Fix auto-save	`TripManager.swift`, `Trip.swift`	`ContentView.swift`
UI styling issues	`Color+Spotify.swift`, specific component	-
Trip route bugs	`TripManager.swift`, `MapView.swift`	`Trip.swift`
Profile problems	`ProfileManager.swift`, `ProfileView.swift`	`UserProfile.swift`
Norwegian addresses	`LocationManager.swift`, `TripManager.swift`	-
Emergency features	`ProfileManager.swift`, `ContentView.swift`	`ContactPicker.swift`

🚀 Development Tips

1. **Start with Managers**: When debugging features, upload the relevant manager first

2. **Add Models**: Include the data model if the issue involves data structure
3. **Include UI Last**: Add view files only if the bug is UI-specific
4. **Check Dependencies**: Use the dependency info above to avoid missing critical files
5. **Theme Context**: Include `Color+Spotify.swift` for any UI-related questions

This architecture follows a clean separation where **Managers** handle business logic, **Models** define data structures, and **Views** handle presentation—making it easy to isolate issues to specific layers.