Yardım Yolda - Service Request Flow Implementation

Overview

This implementation adds the core service request and tracking functionality to the Yardım Yolda Flutter app, including real-time location tracking, Google Maps integration, and automatic provider matching.

What Has Been Implemented

1. New Screens Created

Customer Dashboard (`lib/features/customer/presentation/screens/dashboard_screen.dart`)

- Clean, modern dashboard with user greeting
- Large "Request Service" button to start service flow
- Quick access cards for history and support
- 7/24 service availability indicator
- Profile icon in app bar (placeholder)

Service Selection Screen (`lib/features/customer/presentation/screens/service_select_screen.dart`)

- Four service type options:
- **Çekici** (Tow Truck) - Blue
- **Akü** (Battery) - Green
- **Lastik** (Tire) - Orange
- **Yakıt** (Fuel) - Red
- Automatic location permission request on load
- Visual grid layout with colored cards
- Real-time location fetching with loading state
- Error handling for location permission denied

Request Tracking Screen (`lib/features/customer/presentation/screens/request_tracking_screen.dart`)

- Full-screen Google Maps integration
- Real-time status updates via Supabase subscriptions
- Customer location marker (red pin)
- Provider location marker (blue pin) when matched
- Status card showing:
- Current status with color coding
- Service type with icon
- Loading animation while searching
- Provider information when matched
- Camera auto-adjustment to show both markers

- Cancel service dialog confirmation
- Status messages:
- “Eşleştiriliyor…” (Searching)
- “Aracınız yola çıktı” (Matched)

2. Data Models

ServiceRequest Model (`lib/features/customer/domain/models/service_request.dart`)

- Complete service request data structure
- Maps to `service_requests` table in Supabase
- Helper methods: `isMatched`, `isSearching`
- Immutable with `copyWith` support

MockProvider Model

- Represents mock service providers
- Used for development/testing
- Contains location and name information

3. Repositories

ServiceRequestRepository (`lib/features/customer/data/repositories/service_request_repository.dart`)

- `createServiceRequest()` - Create new service request
- `getServiceRequest()` - Fetch specific request
- `getCustomerServiceRequests()` - Get all customer requests
- `subscribeToServiceRequest()` - Real-time updates via Supabase stream
- `getMockProvider()` - Fetch provider details
- `cancelServiceRequest()` - Cancel active request

4. State Management (Riverpod)

Location Provider (`lib/features/customer/providers/location_provider.dart`)

- `LocationService` class for location operations
- Permission checking and requesting
- Current position fetching with high accuracy
- Turkish error messages for permission issues
- `currentPositionProvider` - FutureProvider for current location

Service Request Provider (`lib/features/customer/providers/service_request_provider.dart`)

- `ServiceRequestNotifier` - StateNotifier for request state
- Manages service request lifecycle
- Real-time subscription to Supabase updates
- Automatic UI updates when status changes
- `mockProviderProvider` - Family provider for provider data

5. Routing Updates

Updated Routes in `app_router.dart`

```
// New customer routes
/customer/dashboard          -> DashboardScreen
/customer/service-select     -> ServiceSelectScreen
/customer/request-tracking/:id -> RequestTrackingScreen(requestId)
```

Redirect Logic Updates

- Authenticated users redirected to `/customer/dashboard`
- Old `/dashboard` route redirects to new customer dashboard
- Protected customer routes require authentication

6. Database Migration

Migration File: `supabase/migrations/0002_mock_match.sql`

- Creates `mock_providers` table
- Inserts demo provider at Istanbul coordinates (41.083, 29.012)
- Creates `match_after_insert()` trigger function
- Automatically matches requests 5 seconds after creation
- Enables Supabase realtime for `service_requests` table
- Includes documentation comments

How it works:
1. User creates service request with status='searching'
2. Trigger waits 5 seconds (`pg_sleep(5)`)
3. Updates request to status='matched' with random provider
4. Real-time subscription notifies Flutter app
5. UI updates to show provider location

7. Dependencies

All required dependencies are already in `pubspec.yaml`:
- ✅ `google_maps_flutter: ^2.6.1`
- ✅ `geolocator: ^12.0.0`
- ✅ `flutter_riverpod: ^2.5.1`
- ✅ `supabase_flutter: ^2.5.1`
- ✅ `go_router: ^14.0.2`

Complete File Structure

```
lib/features/customer/
├── data/
│   └── repositories/
│       └── service_request_repository.dart
├── domain/
│   └── models/
│       └── service_request.dart
├── presentation/
│   └── screens/
│       ├── dashboard_screen.dart
│       ├── service_select_screen.dart
│       └── request_tracking_screen.dart
└── providers/
    ├── location_provider.dart
    └── service_request_provider.dart

supabase/
└── migrations/
    └── 0002_mock_match.sql
```

How to Use

1. Run Database Migration

Apply the migration to your Supabase database:

```
# If using Supabase CLI
supabase db reset

# Or apply migration directly in Supabase Dashboard
# Copy content of 0002_mock_match.sql and run in SQL Editor
```

2. Configure Google Maps API Key

Android (`android/app/src/main/AndroidManifest.xml`):

```
<manifest>
  <application>
    <meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="YOUR_ANDROID_API_KEY"/>
  </application>
</manifest>
```

iOS (`ios/Runner/AppDelegate.swift`):

```
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]
?
  ) -> Bool {
    GMSServices.provideAPIKey("YOUR_IOS_API_KEY")
    GeneratedPluginRegistrant.register(with: self)
    return super.application(application, didFinishLaunchingWithOptions: launchOp-
tions)
  }
}
```

3. Add Location Permissions

Android (`android/app/src/main/AndroidManifest.xml`):

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

iOS (`ios/Runner/Info.plist`):

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>Bu uygulama yol yardım hizmeti sağlamak için konumunuzu kullanır</string>
<key>NSLocationAlwaysUsageDescription</key>
<string>Bu uygulama yol yardım hizmeti sağlamak için konumunuzu kullanır</string>
```

4. Run the App

```
flutter pub get
flutter run
```

User Flow

1. **Login** → User authenticates with phone number
2. **Dashboard** → User sees welcome screen with “Yardım İste” button
3. **Service Selection** → User chooses service type (Çekici, Akü, Lastik, or Yakıt)
4. **Location Request** → App requests and gets current location
5. **Request Creation** → Service request created with status=’searching’
6. **Tracking Screen** → User sees map with their location marker
7. **Auto-Match** → After 5 seconds, trigger matches with mock provider
8. **Status Update** → Real-time subscription updates UI to show provider
9. **Map Update** → Provider location marker appears, camera adjusts

Key Features

Real-time Updates

- Uses Supabase realtime subscriptions
- Automatic UI refresh when service status changes
- No polling required - push-based updates

Location Services

- High accuracy GPS positioning
- Permission handling with Turkish messages
- Error states for denied permissions
- Retry mechanism for failed location requests

Google Maps Integration

- Customer marker (red) and provider marker (blue)
- Auto-centering to show both markers when matched
- My Location button enabled
- Smooth camera animations

State Management

- Riverpod for reactive state
- AsyncValue for loading/error states
- Family providers for parameterized data
- Proper dispose handling for subscriptions

Error Handling

- Location permission denied
- Network errors
- Service not found
- User-friendly Turkish error messages
- Retry mechanisms

Future Enhancements

1. **Real Provider Matching**
   - Replace mock trigger with actual provider matching algorithm
   - Distance-based provider selection
   - Provider availability checks

2. **Provider App**
   - Accept/reject requests
   - Navigation to customer
   - Status updates (on the way, arrived, completed)

3. **Request History**
   - View past service requests
   - Ratings and reviews
   - Download invoices

4. **Notifications**
   - Push notifications for status changes
   - SMS notifications
   - In-app notification center

5. **Payment Integration**
   - Price calculation
   - Payment methods
   - Invoice generation

6. **Provider Tracking**
   - Real-time provider location updates
   - ETA calculation
   - Route display on map

Technical Notes

Supabase Realtime

- Requires `alter publication supabase_realtime add table service_requests;`
- Stream updates push changes to Flutter app
- Automatic reconnection on network issues

Google Maps Performance

- Markers updated only when data changes
- Camera animations for smooth UX
- Bounds calculation for optimal zoom level

Async/Await Pattern

- All database operations are async
- Proper error handling with try-catch
- Loading states during operations

Clean Architecture

- Domain models separate from data layer
- Repository pattern for data access
- Provider layer for state management
- Presentation layer for UI

Testing Checklist

- [ ] Location permission request works
- [ ] Location permission denied shows error
- [ ] Service type selection creates request
- [ ] Map shows customer location
- [ ] After 5 seconds, status changes to ‘matched’
- [ ] Provider marker appears on map
- [ ] Camera adjusts to show both markers
- [ ] Status card updates with provider info

- [ ] Cancel button shows confirmation dialog
- [ ] Cancel updates status in database
- [ ] Back navigation works correctly
- [ ] Real-time subscription reconnects after network loss

Troubleshooting

Location not working:
- Check permissions in device settings
- Verify location services are enabled
- Check AndroidManifest.xml / Info.plist permissions

Google Maps not showing:
- Verify API key is correct
- Check API key has Maps SDK enabled
- Ensure billing is enabled in Google Cloud

Realtime updates not working:
- Verify migration was applied
- Check Supabase realtime is enabled
- Verify table is added to publication

Service request not matching:
- Check trigger was created successfully
- Verify mock_providers table has data
- Check Supabase logs for errors

Support

For issues or questions:
1. Check implementation files
2. Review error messages in console
3. Check Supabase dashboard logs
4. Verify all migration steps completed

Implementation Date: October 23, 2025
Status: Complete and Ready for Testing
Version: 1.0.0