Customer History and Rating Implementation Summary

Overview

Successfully implemented customer history and rating functionality for the yardimyolda Flutter app. Customers can now view their service request history and rate completed services.

📁 New Files Created

Rating Feature

```
lib/features/rating/
├── domain/models/
│   └── rating.dart                           # Rating model with fromJson/toJson
├── data/repositories/
│   └── rating_repository.dart                # Rating database operations
├── providers/
│   └── rating_provider.dart                  # Riverpod state management
└── presentation/screens/
    └── rating_screen.dart                    # Star rating UI with comment field
```

Customer History Feature

```
lib/features/customer/
├── presentation/screens/
│   └── history_screen.dart                   # Service request history list
└── providers/
    └── history_provider.dart                 # History data provider
```

🔧 Modified Files

1. `lib/routes/app_router.dart`

Changes:
- Added imports for `HistoryScreen` and `RatingScreen`
- Added route `/customer/history` for viewing service history
- Added route `/rating/:requestId` for rating completed services

New Routes:

```
// Customer History Route
GoRoute(
  path: '/customer/history',
  name: 'customer-history',
  builder: (context, state) => const HistoryScreen(),
),

// Rating Route
GoRoute(
  path: '/rating/:requestId',
  name: 'rating',
  builder: (context, state) {
    final requestId = state.pathParameters['requestId']!;
    return RatingScreen(requestId: requestId);
  },
),
```

2. `lib/features/customer/presentation/screens/dashboard_screen.dart`

Changes:
- Updated “Geçmiş” (History) card to navigate to history screen
- Changed from showing placeholder SnackBar to actual navigation

Modified Code:

```
_InfoCard(
  icon: Icons.history,
  title: 'Geçmiş',
  subtitle: 'İstekleriniz',
  onTap: () {
    context.push('/customer/history');  // Now functional!
  },
),
```

3. `README.md`

Changes:
- Added comprehensive “Migration’ları Uygulama” section
- Included multiple installation methods (npm, Homebrew, Scoop)
- Added step-by-step Supabase CLI setup instructions
- Documented migration commands with detailed explanations
- Added useful CLI commands reference

🎯 Feature Details

History Screen (`history_screen.dart`)

Functionality:
- ✅ Fetches all service requests for current user from `service_requests` table
- ✅ Sorts by `created_at` DESC (newest first)
- ✅ Displays service type, status, and creation date
- ✅ Shows “Değerlendir” (Rate) button for completed requests
- ✅ Checks if request already rated and shows “Değerlendirildi” badge
- ✅ Shows “Henüz talebiniz yok” message when history is empty
- ✅ Pull-to-refresh functionality with refresh button

- ✅ Loading state with CircularProgressIndicator
- ✅ Error handling with retry functionality
- ✅ Status chips with color coding (searching, matched, in_progress, completed, cancelled)
- ✅ Service type icons (çekici, akü, lastik, yakıt)
- ✅ Formatted date display (Bugün, Dün, X gün önce)

Key Features:
- Uses `serviceRequestHistoryWithRefreshProvider` for data fetching
- Implements `isRequestRatedProvider` to check rating status
- Turkish language UI throughout
- Clean card-based layout

Rating Screen (`rating_screen.dart`)

Functionality:
- ✅ Accepts requestId as route parameter
- ✅ Interactive 1-5 star rating selector with visual feedback
- ✅ Text field for optional comment (max 500 characters)
- ✅ "Gönder" (Submit) button with loading state
- ✅ Checks if request already rated on load
- ✅ Shows "Zaten Değerlendirildi" view if already rated
- ✅ Validates rating selection before submission
- ✅ Handles unique constraint error with user-friendly message
- ✅ Success feedback with SnackBar
- ✅ Auto-navigation back to history after successful submission
- ✅ Error display with inline error message container
- ✅ Rating text labels (Çok Kötü, Kötü, Orta, İyi, Mükemmel)

Validation:
- Ensures star rating is selected (1-5)
- Comment is optional but trimmed if provided
- Disables submit button while loading or if no rating selected

Rating Repository (`rating_repository.dart`)

Database Operations:
- ✅ `submitRating()` - Insert rating into database
- Handles unique constraint violation (request_id is unique)
- Proper error messages in Turkish
- ✅ `getRatingForRequest()` - Fetch existing rating
- ✅ `getProviderRatings()` - Get all ratings for a provider
- ✅ `getCustomerRatings()` - Get all ratings by a customer
- ✅ `isRequestRated()` - Check if request has been rated

Security:
- Relies on Supabase RLS policies
- Validates customer_id matches auth.uid()
- Ensures request is completed and has provider_id

Rating Provider (`rating_provider.dart`)

State Management:
- ✅ `RatingSubmissionNotifier` - Manages submission state
- ✅ `ratingSubmissionProvider` - Main provider for rating submission

- ✅ `isRequestRatedProvider` - Family provider to check rating status
- ✅ `requestRatingProvider` - Family provider to fetch rating details

State Structure:

```
class RatingSubmissionState {
  final bool isLoading;
  final String? error;
  final Rating? rating;
}
```

History Provider (`history_provider.dart`)

State Management:
- ✅ `serviceRequestHistoryProvider` - Auto-dispose future provider
- ✅ `refreshHistoryProvider` - State provider for manual refresh trigger
- ✅ `serviceRequestHistoryWithRefreshProvider` - History with refresh capability

Features:
- Auto-refresh when returning from rating screen
- Manual refresh via refresh button
- Automatic disposal to prevent memory leaks

🔐 Security & Data Access

RLS Policy Compliance

The implementation follows existing Supabase Row Level Security policies:

For service_requests:
- Customers can only read their own requests (customer_id = auth.uid())
- Auto-filtering by customer_id in queries

For ratings:
- Customers can create ratings for their completed requests
- Unique constraint on request_id prevents duplicate ratings
- Validates request is completed and has provider_id
- Anyone can view ratings (public display)

🎨 UI/UX Features

Design Consistency

- ✅ Material 3 design system
- ✅ Consistent color scheme with existing app
- ✅ Turkish language throughout
- ✅ Proper spacing and padding
- ✅ Card-based layouts
- ✅ Icon integration

User Feedback

- ✅ Loading indicators (CircularProgressIndicator)
- ✅ Error messages via SnackBar (Turkish)

- ✅ Success confirmation messages
- ✅ Empty state messaging
- ✅ Visual rating feedback (amber stars)
- ✅ Status color coding

Navigation Flow

```
Dashboard → History Screen → Rating Screen → Back to History
    ↓            ↓                 ↓              ↓
  "Geçmiş"    List of       Rate completed    Updated
   Button     Requests        service         status
```

📊 Database Schema

ratings table

```sql
CREATE TABLE public.ratings (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    request_id UUID NOT NULL UNIQUE REFERENCES service_requests(id),
    customer_id UUID NOT NULL REFERENCES user_profiles(id),
    provider_id UUID NOT NULL REFERENCES user_profiles(id),
    stars INTEGER NOT NULL CHECK (stars >= 1 AND stars <= 5),
    comment TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);
```

Constraints:
- `request_id` is UNIQUE - prevents duplicate ratings
- `stars` must be between 1 and 5
- All foreign keys with CASCADE delete

🧪 Testing Recommendations

Manual Testing Checklist

History Screen

- [ ] Navigate to history from dashboard
- [ ] Verify empty state shows when no requests
- [ ] Create a service request and verify it appears
- [ ] Check date formatting (Bugün, Dün, etc.)
- [ ] Verify status chips show correct colors
- [ ] Test refresh button functionality
- [ ] Verify “Değerlendir” button only shows for completed requests

Rating Screen

- [ ] Navigate from completed request in history
- [ ] Test star selection (all 5 stars)
- [ ] Verify rating text updates (Çok Kötü to Mükemmel)
- [ ] Submit rating without comment
- [ ] Submit rating with comment

- [ ] Try to rate same request twice (should show already rated)
- [ ] Test form validation (no star selected)
- [ ] Verify navigation back to history after submit
- [ ] Check error handling for network issues

Integration Tests

- [ ] Complete end-to-end flow: Dashboard → History → Rating → Back
- [ ] Verify rating appears as “Değerlendirildi” after submission
- [ ] Test with multiple service requests
- [ ] Verify filtering by customer_id works correctly

🚀 Deployment Notes

Prerequisites

1. Ensure all migrations are applied to Supabase:
   `bash`
   `supabase db push`

2. Verify RLS policies are active:
   `sql`
   `SELECT * FROM pg_policies WHERE tablename = 'ratings';`

3. Test with real user accounts in development

Environment Setup

No additional environment variables needed. Uses existing:
- `SUPABASE_URL`
- `SUPABASE_ANON_KEY`

📱 User Flow Examples

Example 1: First-time Rating

1. User completes a service request
2. Navigates to “Geçmiş” from dashboard
3. Sees completed request with “Değerlendir” button
4. Taps button → Rating screen opens
5. Selects 5 stars, adds comment “Harika hizmet!”
6. Taps “Gönder”
7. Success message appears
8. Returns to history → Shows “Değerlendirildi” badge

Example 2: Viewing History

1. User taps “Geçmiş” on dashboard
2. Sees list of all past requests
3. Each shows service type, status, date
4. Can see which requests are rated
5. Can tap refresh to update list

Example 3: Attempting Duplicate Rating

1. User tries to rate already-rated request
2. Rating screen immediately shows “Zaten Değerlendirildi”
3. Green checkmark icon displayed
4. “Geri Dön” button to return

🔄 State Management Architecture

Provider Hierarchy

```
authStateProvider
    ↓
currentUserProvider
    ↓
serviceRequestHistoryWithRefreshProvider
    ↓
isRequestRatedProvider (per request)
    ↓
ratingSubmissionProvider
```

Data Flow

```
UI (History Screen)
    ↓
ref.watch(serviceRequestHistoryWithRefreshProvider)
    ↓
ServiceRequestRepository.getCustomerServiceRequests()
    ↓
Supabase Query (with RLS)
    ↓
List<ServiceRequest> returned
    ↓
UI renders list with cards
```

📚 Dependencies Used

Existing Packages (No new additions needed)

- `flutter_riverpod` - State management
- `go_router` - Navigation
- `supabase_flutter` - Backend operations
- `intl` - Date formatting

✨ Key Implementation Highlights

1. Auto-Refresh After Rating

The history screen automatically refreshes when user returns from rating screen, showing updated “Değerlendirildi” status.

2. Smart Error Handling

- Unique constraint violations are caught and displayed as user-friendly Turkish messages

- Network errors are handled with retry functionality
- Loading states prevent duplicate submissions

3. Performance Optimization

- Auto-dispose providers prevent memory leaks
- Efficient queries with proper indexing
- Minimal re-renders with Riverpod state management

4. User Experience

- Visual feedback at every step
- Clear status indicators
- Intuitive navigation flow
- Turkish language consistency

🐛 Known Limitations & Future Enhancements

Current Limitations

- No editing of submitted ratings
- No photo upload with ratings
- No provider response to ratings

Potential Enhancements

1. Add rating statistics on provider profiles
2. Allow customers to view their own past ratings
3. Add filtering/sorting options in history (by service type, status, date)
4. Implement pagination for large history lists
5. Add search functionality in history
6. Show average provider rating in request tracking
7. Add rating reminder notifications

📖 Code Documentation

All files include:
- ✅ Clear class and function documentation
- ✅ Parameter descriptions
- ✅ Return type documentation
- ✅ Turkish comments for business logic
- ✅ Error handling explanations

🎓 Learning Resources

For developers working with this code:
- Riverpod Documentation (https://riverpod.dev)
- Go Router Guide (https://pub.dev/packages/go_router)
- Supabase Flutter SDK (https://supabase.com/docs/reference/dart)
- Material 3 Guidelines (https://m3.material.io)

✅ Deliverables Completed

1. ✅ New screen files created:
   - `lib/features/customer/presentation/screens/history_screen.dart`
   - `lib/features/rating/presentation/screens/rating_screen.dart`

2. ✅ New model and repository files:
   - `lib/features/rating/domain/models/rating.dart`
   - `lib/features/rating/data/repositories/rating_repository.dart`

3. ✅ New provider files:
   - `lib/features/customer/providers/history_provider.dart`
   - `lib/features/rating/providers/rating_provider.dart`

4. ✅ Updated routing:
   - `lib/routes/app_router.dart` with new routes

5. ✅ Updated dashboard:
   - `lib/features/customer/presentation/screens/dashboard_screen.dart` with functional history button

6. ✅ Updated documentation:
   - `README.md` with comprehensive migration instructions

7. ✅ Version control:
   - All changes committed to git with descriptive commit message

🎯 Success Criteria Met

- ✅ Customer can view all their service requests
- ✅ Requests sorted by date (newest first)
- ✅ Each request shows service type, status, and date
- ✅ Completed requests show rating button
- ✅ Rating screen allows 1-5 star selection
- ✅ Optional comment field with validation
- ✅ Prevents duplicate ratings
- ✅ Shows success/error feedback
- ✅ Turkish language throughout
- ✅ Consistent design with existing app
- ✅ Proper error handling
- ✅ Loading states implemented
- ✅ Follows existing code patterns
- ✅ RLS policies respected

Implementation Date: October 23, 2025
Status: ✅ Complete and ready for testing
Git Commit: `12873c0` - feat: Implement customer history and rating functionality