Flutter-Django Integration Guide: TaskLink Accounts Module

Overview

You are an AI agent tasked with integrating a Django REST Framework backend with a Flutter mobile application. Your immediate focus is implementing the **Accounts section only**. This document provides comprehensive guidance on folder structure, data models, API integration patterns, and error handling.

Backend API Information

Base Configuration

- **Base URL**: `http://localhost:8000/api/v1/`
- **Authentication**: JWT (JSON Web Tokens)
- **Response Format**: Standardized JSON structure

Standard Response Format

All API responses follow a consistent structure:

Success Response

```
{
  "success": true,
  "message": "Success message here",
  "data": {
    // Response data here
  }
}
```

Error Response

```
{
  "success": false,
  "message": "Error message here",
  "errors": {
    "field_name": ["Error detail 1", "Error detail 2"]
  }
}
```

Paginated Response

```
{
  "success": true,
  "message": "Data retrieved successfully",
  "data": {
    "count": 100,
    "next": "http://localhost:8000/api/v1/endpoint/?page=2",
    "previous": null,
    "results": [
      // Array of items
    ]
  }
}
```

Flutter Folder Structure

Create a modular, feature-based architecture with clear separation of concerns:

```
lib/
├── core/
│   ├── constants/
│   │   ├── api_constants.dart          # API endpoints, base URLs
│   │   └── app_constants.dart          # App-wide constants
│   ├── network/
│   │   ├── api_client.dart             # Dio/HTTP client configuration
│   │   ├── api_interceptor.dart        # JWT token interceptor
│   │   └── api_response.dart           # Generic API response wrapper
│   ├── errors/
│   │   ├── exceptions.dart             # Custom exception classes
│   │   └── failures.dart               # Failure handling
│   └── utils/
│       ├── validators.dart             # Input validation utilities
│       └── storage_helper.dart         # Secure storage for tokens
│
├── features/
│   └── accounts/
│       ├── data/
│       │   ├── models/
│       │   │   ├── user_model.dart
│       │   │   ├── user_profile_model.dart
│       │   │   └── wallet_model.dart
│       │   ├── repositories/
│       │   │   └── accounts_repository.dart
│       │   └── data_sources/
│       │       └── accounts_remote_data_source.dart
│       ├── domain/
│       │   ├── entities/
│       │   │   ├── user.dart
│       │   │   ├── user_profile.dart
│       │   │   └── wallet.dart
```

```
│       │   └── repositories/
│       │       └── i_accounts_repository.dart
│       ├── presentation/
│       │   ├── controllers/
│       │   │   ├── auth_controller.dart
│       │   │   ├── profile_controller.dart
│       │   │   └── wallet_controller.dart
│       │   ├── screens/
│       │   │   ├── register_screen.dart
│       │   │   ├── login_screen.dart
│       │   │   ├── otp_verification_screen.dart
│       │   │   ├── profile_screen.dart
│       │   │   ├── edit_profile_screen.dart
│       │   │   ├── change_password_screen.dart
│       │   │   ├── forgot_password_screen.dart
│       │   │   └── wallet_screen.dart
│       │   └── widgets/
│       │       ├── profile_image_picker.dart
│       │       └── custom_text_field.dart
│       └── accounts_module.dart
│
└── main.dart
```

📋 Data Models to Create

1. User Model (`user_model.dart`)

```dart
class UserModel {
  final String id;
  final String phoneNumber;
  final String email;
  final bool isPhoneVerified;
  final bool isEmailVerified;
  final String role; // 'POSTER', 'TASKER', 'BOTH'
  final double ratingAvg;
  final int totalTasksPosted;
  final int totalTasksCompleted;
  final DateTime createdAt;
  final UserProfileModel? profile;

  UserModel({
    required this.id,
    required this.phoneNumber,
    required this.email,
    required this.isPhoneVerified,
    required this.isEmailVerified,
    required this.role,
    required this.ratingAvg,
    required this.totalTasksPosted,
    required this.totalTasksCompleted,
```

```
    required this.createdAt,
    this.profile,
  });

  factory UserModel.fromJson(Map<String, dynamic> json) {
    return UserModel(
      id: json['id'],
      phoneNumber: json['phone_number'],
      email: json['email'],
      isPhoneVerified: json['is_phone_verified'],
      isEmailVerified: json['is_email_verified'],
      role: json['role'],
      ratingAvg: (json['rating_avg'] as num).toDouble(),
      totalTasksPosted: json['total_tasks_posted'],
      totalTasksCompleted: json['total_tasks_completed'],
      createdAt: DateTime.parse(json['created_at']),
      profile: json['profile'] != null
          ? UserProfileModel.fromJson(json['profile'])
          : null,
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'phone_number': phoneNumber,
      'email': email,
      'is_phone_verified': isPhoneVerified,
      'is_email_verified': isEmailVerified,
      'role': role,
      'rating_avg': ratingAvg,
      'total_tasks_posted': totalTasksPosted,
      'total_tasks_completed': totalTasksCompleted,
      'created_at': createdAt.toIso8601String(),
      'profile': profile?.toJson(),
    };
  }
}
```

2. User Profile Model (`user_profile_model.dart`)

```
class UserProfileModel {
  final String fullName;
  final String? profileImage;
  final String? bio;
  final DateTime? dateOfBirth;
  final String? gender; // 'M', 'F', 'O', 'N'

  UserProfileModel({
    required this.fullName,
    this.profileImage,
```

```
    this.bio,
    this.dateOfBirth,
    this.gender,
  });

  factory UserProfileModel.fromJson(Map<String, dynamic> json) {
    return UserProfileModel(
      fullName: json['full_name'] ?? '',
      profileImage: json['profile_image'],
      bio: json['bio'],
      dateOfBirth: json['date_of_birth'] != null
          ? DateTime.parse(json['date_of_birth'])
          : null,
      gender: json['gender'],
    );
  }

  Map<String, dynamic> toJson() {
    return {
      'full_name': fullName,
      'profile_image': profileImage,
      'bio': bio,
      'date_of_birth': dateOfBirth?.toIso8601String(),
      'gender': gender,
    };
  }
}
```

3. Wallet Model (`wallet_model.dart`)

```
class WalletModel {
  final String id;
  final double balance;
  final DateTime createdAt;
  final DateTime updatedAt;

  WalletModel({
    required this.id,
    required this.balance,
    required this.createdAt,
    required this.updatedAt,
  });

  factory WalletModel.fromJson(Map<String, dynamic> json) {
    return WalletModel(
      id: json['id'],
      balance: (json['balance'] as num).toDouble(),
      createdAt: DateTime.parse(json['created_at']),
      updatedAt: DateTime.parse(json['updated_at']),
    );
```

```
  }
}
```

4. Auth Response Model (`auth_response_model.dart`)

```dart
class AuthResponseModel {
  final UserModel user;
  final TokensModel tokens;

  AuthResponseModel({
    required this.user,
    required this.tokens,
  });

  factory AuthResponseModel.fromJson(Map<String, dynamic> json) {
    return AuthResponseModel(
      user: UserModel.fromJson(json['user']),
      tokens: TokensModel.fromJson(json['tokens']),
    );
  }
}

class TokensModel {
  final String access;
  final String refresh;

  TokensModel({
    required this.access,
    required this.refresh,
  });

  factory TokensModel.fromJson(Map<String, dynamic> json) {
    return TokensModel(
      access: json['access'],
      refresh: json['refresh'],
    );
  }
}
```

API Endpoints for Accounts Module

Authentication Endpoints

Endpoint	Method	Auth Required	Description
`/auth/register/`	POST	❌	Register new user
`/auth/login/`	POST	❌	Login and get JWT tokens

Endpoint	Method	Auth Required	Description
`/auth/refresh/`	POST	❌	Refresh access token
`/auth/send-otp/`	POST	☑	Send OTP (email/phone)
`/auth/verify-otp/`	POST	☑	Verify OTP

User Profile Endpoints

Endpoint	Method	Auth Required	Description
`/users/me/`	GET	☑	Get current user profile
`/users/update_me/`	PUT	☑	Update user profile
`/users/change_password/`	POST	☑	Change password
`/users/update_location/`	POST	☑	Update location

Password Reset Endpoints

Endpoint	Method	Auth Required	Description
`/password-reset/send_otp/`	POST	❌	Request password reset OTP
`/password-reset/reset/`	POST	❌	Reset password with OTP

Wallet Endpoints

Endpoint	Method	Auth Required	Description
`/wallet/`	GET	☑	Get wallet information

📡 API Request/Response Examples

1. Registration

Request:

```
POST /api/v1/auth/register/
Content-Type: application/json

{
  "phone_number": "923001234567",
  "email": "user@example.com",
  "password": "SecurePass123",
  "password_confirm": "SecurePass123",
  "role": "POSTER",
  "profile": {
    "full_name": "John Doe"
```

```
  }
}
```

Success Response (201 Created):

```
{
  "success": true,
  "message": "Registration successful",
  "data": {
    "user": {
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "phone_number": "923001234567",
      "email": "user@example.com",
      "is_phone_verified": false,
      "is_email_verified": false,
      "role": "POSTER",
      "rating_avg": 0.0,
      "total_tasks_posted": 0,
      "total_tasks_completed": 0,
      "created_at": "2026-02-05T12:00:00Z",
      "profile": {
        "full_name": "John Doe",
        "profile_image": null,
        "bio": "",
        "date_of_birth": null,
        "gender": ""
      }
    },
    "tokens": {
      "access": "eyJ0eXAiOiJKV1QiLCJhbGc...",
      "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
    }
  }
}
```

Error Response (400 Bad Request):

```
{
  "success": false,
  "message": "Validation failed",
  "errors": {
    "phone_number": ["Phone number must be exactly 12 characters long."],
    "password": ["Passwords do not match."]
  }
}
```

2. Login

Request:

```
POST /api/v1/auth/login/
Content-Type: application/json

{
  "username": "923001234567",  // Can be phone or email
  "password": "SecurePass123"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "user": {
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "phone_number": "923001234567",
      "email": "user@example.com",
      "is_phone_verified": true,
      "is_email_verified": true,
      "role": "POSTER",
      "rating_avg": 4.5,
      "total_tasks_posted": 10,
      "total_tasks_completed": 25,
      "created_at": "2026-01-01T12:00:00Z",
      "profile": {
        "full_name": "John Doe",
        "profile_image":
"http://localhost:8000/media/profiles/550e8400.../profile.jpg",
        "bio": "Professional tasker",
        "date_of_birth": "1990-01-01",
        "gender": "M"
      }
    },
    "tokens": {
      "access": "eyJ0eXAiOiJKV1QiLCJhbGc...",
      "refresh": "eyJ0eXAiOiJKV1QiLCJhbGc..."
    }
  }
}
```

Error Response (401 Unauthorized):

```
{
  "success": false,
  "message": "Invalid credentials",
```

```
  "errors": {
    "non_field_errors": ["No active account found with the given credentials"]
  }
}
```

3. Send OTP

Request:

```
POST /api/v1/auth/send-otp/
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "type": "email"  // or "phone"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "OTP sent successfully to your email",
  "data": {}
}
```

4. Verify OTP

Request:

```
POST /api/v1/auth/verify-otp/
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "type": "email",  // or "phone"
  "otp": "123456"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Email verified successfully",
  "data": {}
}
```

Error Response (400 Bad Request):

```
{
  "success": false,
  "message": "Invalid or expired OTP",
  "errors": {
    "otp": ["Invalid OTP"]
  }
}
```

5. Get Profile

Request:

```
GET /api/v1/users/me/
Authorization: Bearer <access_token>
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Profile retrieved successfully",
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "phone_number": "923001234567",
    "email": "user@example.com",
    "is_phone_verified": true,
    "is_email_verified": true,
    "role": "BOTH",
    "rating_avg": 4.75,
    "total_tasks_posted": 15,
    "total_tasks_completed": 30,
    "created_at": "2026-01-01T12:00:00Z",
    "profile": {
      "full_name": "John Doe",
      "profile_image":
"http://localhost:8000/media/profiles/550e8400.../profile.jpg",
      "bio": "Experienced professional",
      "date_of_birth": "1990-01-01",
      "gender": "M"
    }
  }
}
```

6. Update Profile (with Image)

Request:

```
PUT /api/v1/users/update_me/
Authorization: Bearer <access_token>
Content-Type: multipart/form-data

{
  "email": "newemail@example.com",
  "role": "BOTH",
  "profile.full_name": "John Updated Doe",
  "profile.bio": "Updated professional bio",
  "profile.profile_image": <binary_image_data>
}
```

Success Response (200 OK):


```
{
  "success": true,
  "message": "Profile updated successfully",
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "phone_number": "923001234567",
    "email": "newemail@example.com",
    "is_phone_verified": true,
    "is_email_verified": false,
    "role": "BOTH",
    "rating_avg": 4.75,
    "total_tasks_posted": 15,
    "total_tasks_completed": 30,
    "created_at": "2026-01-01T12:00:00Z",
    "profile": {
      "full_name": "John Updated Doe",
      "profile_image":
"http://localhost:8000/media/profiles/550e8400.../profile.jpg",
      "bio": "Updated professional bio",
      "date_of_birth": "1990-01-01",
      "gender": "M"
    }
  }
}
```


7. Change Password

Request:

```
POST /api/v1/users/change_password/
Authorization: Bearer <access_token>
Content-Type: application/json
```

```
{
  "old_password": "OldPass123",
  "new_password": "NewSecurePass456",
  "new_password_confirm": "NewSecurePass456"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Password changed successfully",
  "data": {}
}
```

Error Response (400 Bad Request):

```
{
  "success": false,
  "message": "Password change failed",
  "errors": {
    "old_password": ["Old password is incorrect."]
  }
}
```

8. Password Reset Flow

Step 1: Request OTP

```
POST /api/v1/password-reset/send_otp/
Content-Type: application/json

{
  "identifier": "923001234567"  // Can be phone or email
}
```

Response:

```
{
  "success": true,
  "message": "OTP sent to 923001234567",
  "data": {
    "message": "OTP sent successfully"
  }
}
```

Step 2: Reset Password

```
POST /api/v1/password-reset/reset/
Content-Type: application/json

{
  "identifier": "923001234567",
  "otp": "123456",
  "new_password": "NewSecurePass789",
  "new_password_confirm": "NewSecurePass789"
}
```

Response:

```
{
  "success": true,
  "message": "Password reset successful",
  "data": {}
}
```

9. Get Wallet

Request:

```
GET /api/v1/wallet/
Authorization: Bearer <access_token>
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Wallet retrieved successfully",
  "data": {
    "id": "wallet-uuid",
    "balance": 1500.50,
    "created_at": "2026-01-01T12:00:00Z",
    "updated_at": "2026-02-05T12:00:00Z"
  }
}
```

🛠 Implementation Guidelines

1. API Client Setup (`api_client.dart`)

```dart
import 'package:dio/dio.dart';

class ApiClient {
  static const String baseUrl = 'http://localhost:8000/api/v1/';
  late Dio _dio;

  ApiClient() {
    _dio = Dio(BaseOptions(
      baseUrl: baseUrl,
      connectTimeout: const Duration(seconds: 30),
      receiveTimeout: const Duration(seconds: 30),
      headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
      },
    ));

    _dio.interceptors.add(ApiInterceptor());
  }

  Dio get dio => _dio;
}
```

2. API Interceptor for JWT (`api_interceptor.dart`)

```dart
import 'package:dio/dio.dart';
import '../utils/storage_helper.dart';

class ApiInterceptor extends Interceptor {
  @override
  void onRequest(RequestOptions options, RequestInterceptorHandler handler) async
{
    final token = await StorageHelper.getAccessToken();
    if (token != null) {
      options.headers['Authorization'] = 'Bearer $token';
    }
    super.onRequest(options, handler);
  }

  @override
  void onError(DioException err, ErrorInterceptorHandler handler) async {
    if (err.response?.statusCode == 401) {
      // Token expired, attempt refresh
      final refreshed = await _refreshToken();
      if (refreshed) {
        // Retry original request
        final response = await _retry(err.requestOptions);
        return handler.resolve(response);
```

```
      }
    }
    super.onError(err, handler);
  }

  Future<bool> _refreshToken() async {
    // Implement token refresh logic
    return false;
  }

  Future<Response> _retry(RequestOptions requestOptions) async {
    final options = Options(
      method: requestOptions.method,
      headers: requestOptions.headers,
    );
    return Dio().request(
      requestOptions.path,
      data: requestOptions.data,
      queryParameters: requestOptions.queryParameters,
      options: options,
    );
  }
}
```

3. Generic API Response Wrapper (`api_response.dart`)

```
class ApiResponse<T> {
  final bool success;
  final String message;
  final T? data;
  final Map<String, dynamic>? errors;

  ApiResponse({
    required this.success,
    required this.message,
    this.data,
    this.errors,
  });

  factory ApiResponse.fromJson(
    Map<String, dynamic> json,
    T Function(dynamic)? dataParser,
  ) {
    return ApiResponse(
      success: json['success'] ?? false,
      message: json['message'] ?? '',
      data: json['data'] != null && dataParser != null
          ? dataParser(json['data'])
          : null,
      errors: json['errors'],
    );
```

```
  }

  bool get isSuccess => success;
  bool get hasErrors => errors != null && errors!.isNotEmpty;
}
```

4. Error Handling

Custom Exception Classes:

```
class ServerException implements Exception {
  final String message;
  ServerException(this.message);
}

class NetworkException implements Exception {
  final String message;
  NetworkException(this.message);
}

class ValidationException implements Exception {
  final Map<String, dynamic> errors;
  ValidationException(this.errors);
}

class UnauthorizedException implements Exception {
  final String message;
  UnauthorizedException(this.message);
}
```

Response Handler:

```
Future<ApiResponse<T>> handleApiCall<T>(
  Future<Response> Function() apiCall,
  T Function(dynamic) dataParser,
) async {
  try {
    final response = await apiCall();
    return ApiResponse.fromJson(response.data, dataParser);
  } on DioException catch (e) {
    if (e.response != null) {
      return ApiResponse.fromJson(e.response!.data, null);
    }
    throw NetworkException('Network error occurred');
  } catch (e) {
    throw ServerException('Unexpected error occurred');
  }
}
```

5. Secure Token Storage (`storage_helper.dart`)

```dart
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

class StorageHelper {
  static const _storage = FlutterSecureStorage();

  static Future<void> saveAccessToken(String token) async {
    await _storage.write(key: 'access_token', value: token);
  }

  static Future<void> saveRefreshToken(String token) async {
    await _storage.write(key: 'refresh_token', value: token);
  }

  static Future<String?> getAccessToken() async {
    return await _storage.read(key: 'access_token');
  }

  static Future<String?> getRefreshToken() async {
    return await _storage.read(key: 'refresh_token');
  }

  static Future<void> clearTokens() async {
    await _storage.delete(key: 'access_token');
    await _storage.delete(key: 'refresh_token');
  }
}
```

☑ Validation Rules

Phone Number Validation

- Must be exactly **12 digits**
- Must start with country code **92** (Pakistan)
- Example: `923001234567`

```dart
String? validatePhoneNumber(String? value) {
  if (value == null || value.isEmpty) {
    return 'Phone number is required';
  }
  if (!RegExp(r'^\d{12}$').hasMatch(value)) {
    return 'Phone number must be 12 digits';
  }
  if (!value.startsWith('92')) {
    return 'Phone number must start with 92';
  }
  return null;
}
```

Email Validation

```
String? validateEmail(String? value) {
  if (value == null || value.isEmpty) {
    return 'Email is required';
  }
  if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$').hasMatch(value)) {
    return 'Enter a valid email';
  }
  return null;
}
```

Password Validation

```
String? validatePassword(String? value) {
  if (value == null || value.isEmpty) {
    return 'Password is required';
  }
  if (value.length < 8) {
    return 'Password must be at least 8 characters';
  }
  return null;
}
```

Key Implementation Notes

Authentication Flow

1. **Register** → Receive JWT tokens → Store securely
2. **Login** → Receive JWT tokens → Store securely
3. **Auto-login** → Check stored tokens on app start
4. **Token Refresh** → Automatically refresh when access token expires

Profile Image Upload

- Use `multipart/form-data` for image uploads
- Field name: `profile.profile_image`
- Backend automatically overwrites old images
- Returns full URL in response

OTP Verification

- Two types: `email` and `phone`
- OTP is 6 digits

- Must be verified for certain operations
- Rate limited on backend

Role Management

- Three roles: `POSTER`, `TASKER`, `BOTH`
- Can be changed via profile update
- Affects available features in app

Error Display

- Extract field-specific errors from `errors` object
- Show general message from `message` field
- Handle network errors gracefully
- Implement retry mechanisms

🎯 Implementation Checklist

- [ ] Set up folder structure as specified
- [ ] Create all data models with proper JSON serialization
- [ ] Implement API client with Dio/HTTP
- [ ] Add JWT interceptor for automatic token injection
- [ ] Implement secure storage for tokens
- [ ] Create repository pattern for data layer
- [ ] Build all authentication screens (register, login, OTP)
- [ ] Build profile management screens
- [ ] Build password reset flow
- [ ] Implement form validation
- [ ] Add error handling and user feedback
- [ ] Test all API integrations
- [ ] Implement loading states
- [ ] Add image picker for profile pictures
- [ ] Test token refresh mechanism

🚀 Next Steps

After completing the accounts module:

1. Implement **Tasks** module (task creation, bidding)
2. Implement **Chat** module (real-time messaging)
3. Implement **Payments** module (wallet, escrow)
4. Implement **Reviews** module
5. Implement **Disputes** module
6. Implement **Notifications** module

Each module will follow the same architectural pattern established in the accounts module.

> **Note**: This guide focuses exclusively on the **Accounts** module. Do not implement features from other modules until explicitly instructed.