MATCHDAY

Flutter Developer Guide

Complete API Integration Reference
Authentication | Real-time Chat | QR Codes | Bookings | Payments

Version 1.0 | February 2026 | ~263 Endpoints | ~43 Tables
Laravel 12 + Sanctum + MySQL 8 + Reverb WebSockets

367 Tests | 2,268 Assertions | 100% Pass Rate

Table of Contents

1. Quick Start and Setup

Base URL

```
Local: http://10.0.2.2:8000/api/v1 (Android Emulator)
http://localhost:8000/api/v1 (iOS Simulator)
Staging: https://staging-api.matchday.app/api/v1
Production: https://api.matchday.app/api/v1
```

WARNING: Android emulator uses 10.0.2.2 to reach host machine localhost

Required Headers

```
Content-Type: application/json
Accept: application/json
Authorization: Bearer {token} // Only for authenticated endpoints
```

Dart HTTP Client Setup

```
class ApiClient {
static const baseUrl = 'http://10.0.2.2:8000/api/v1';
String? _token;

Map<String, String> get headers => {
'Content-Type': 'application/json',
'Accept': 'application/json',
if (_token != null) 'Authorization': 'Bearer $_token',
};

void setToken(String token) => _token = token;
void clearToken() => _token = null;
}
```

Rate Limits

Endpoint Group	Limit	Flutter Handling
Auth (login/register)	5/min per IP	Show countdown timer
Password Reset	3/min per IP	Disable button 60s
General API	60/min per user	Rarely hit normally
File Uploads	5/min per user	Queue if needed
Chat Messages	10/min per user	Throttle send button

2. Authentication Flow

Overview

Laravel Sanctum - plain bearer tokens. No OAuth complexity. No token expiry. Tokens persist until logout.

2.1 Registration (2-Step)

Step 1: Create Account

POST **/auth/register** - Create fan account

```
// REQUEST
{
"name": "Ahmed Hassan",
"email": "ahmed@example.com",
"phone": "+966512345678",
"password": "password123",
"password_confirmation": "password123"
}

// RESPONSE (201)
{
"success": true,
"data": {
"user": {
"id": 1, "name": "Ahmed Hassan", "role": "fan",
"needs_team_selection": true,
"onboarding_complete": false
},
"token": "1|abc123def456..."
}
}
```

WARNING: Save token to flutter_secure_storage immediately

Step 2: Choose Favorite Team (Skippable)

If **needs_team_selection = true**, show team selection:

GET **/teams/popular** - Popular teams list

GET **/teams/search?q=hilal** - Search teams

PUT **/profile/favorite-team** - Set team: { team_id: 1 }

2.2 Login

POST **/auth/login** - Email OR phone login

```
// REQUEST
{ "email_or_phone": "ahmed@example.com", "password": "password123" }

// SUCCESS (200) - returns user + token
// WRONG PASSWORD (401) - "The provided credentials are incorrect."
// DEACTIVATED (403) - "Your account has been deactivated."
// VALIDATION (422) - errors object with field details
```

2.3 Social Auth

POST **/auth/login/google** - { google_token: '...' }

POST **/auth/login/apple** - { apple_token: '...', name: 'Ahmed' } (name only first time)

2.4 Token Management

GET **/auth/me** - Verify token + get current user

POST **/auth/refresh** - New token (revokes old)

POST **/auth/logout** - Revoke current token

Flutter Auth Flow

```
// App Startup:
1. Read token from secure storage
2. If token: GET /auth/me
- 200: go to home
- 401: clear token, show login
3. If no token: show login

// After Login:
1. Save token to flutter_secure_storage
2. Set in HTTP client headers
3. Check needs_team_selection flag

// On ANY 401 Response:
1. Clear token
2. Navigate to login
3. Show 'Session expired'
```

2.5 Password Reset

POST **/auth/forgot-password** - { email: '...' } - sends OTP

POST **/auth/reset-password** - { email, token(OTP), password, password_confirmation }

2.6 Email Verification

POST **/auth/verify-email** - { otp: '123456' }

POST **/auth/resend-verification** - Resend OTP (1/min limit)

3. Response Format and Error Handling

Standard Response

```
{
"success": true | false,
"message": "Human-readable message",
"data": { ... } | [ ... ],
"meta": { "current_page": 1, "last_page": 5, "per_page": 15, "total": 67 },
"errors": { "field": ["Error message"] }
}
```

Error Codes

Code	Meaning	Flutter Action
200	Success	Parse data normally
201	Created	Parse data, show success
401	Unauthenticated	Clear token -> login
403	Forbidden	Show access denied
404	Not Found	Show not found or go back
422	Validation	Show errors under fields
429	Rate Limited	Show countdown (Retry-After header)
500	Server Error	Show generic error

Dart Error Handler

```
Future handleResponse(http.Response res) async {
final json = jsonDecode(res.body);
switch (res.statusCode) {
case 200: case 201:
return ApiResponse.success(json['data'], json['meta']);
case 401:
await SecureStorage.clearToken();
navigatorKey.currentState?.pushReplacementNamed('/login');
throw AuthException('Session expired');
case 422:
throw ValidationException(json['errors'] ?? {});
case 429:
final retry = res.headers['retry-after'] ?? '60';
throw RateLimitException(int.parse(retry));
default:
throw ApiException(json['message'] ?? 'Unknown error');
}
}
```

4. Complete User Journey

4.1 Home Screen

GET /home?lat=24.71&lng=46.67 - Aggregated home feed

```
// Returns:
{
"user": { "name": "Ahmed", "tier": "silver", "unread_count": 3 },
"upcoming_booking": { ... } | null,
"nearby_cafes": [ { "distance_km": 1.2, "current_status": "available" } ],
"live_matches": [ { "home_score": 2, "away_score": 1 } ],
"upcoming_matches": [ { "is_saved": false, "is_booked": false } ],
"popular_this_week": [ ... ]
}
```

TIP: Pass lat/lng for nearby cafes. Without it, nearby returns empty.

4.2 Explore

GET /explore?lat=24.71&lng=46.67 - 6 sections: featured, nearby, trending, today, popular, offers

4.3 Search

GET /search?q=champions - Returns { cafes, matches, cities }

4.4 Cafe Detail

GET /cafes/{id} - Cafe overview + branches

GET /branches/{id} - Branch: hours, amenities, status (available/busy/full/closed)

GET /branches/{id}/reviews?page=1 - Paginated reviews

POST /branches/{id}/reviews - { rating: 5, comment: '...' } - one per user

POST /saved-cafes/{id} - Save cafe

DELETE /saved-cafes/{id} - Unsave cafe

4.5 Match Detail

GET /matches/{id} - Match + teams + is_saved

GET /matches/{id}/seating - Sections with seats + is_booked flag

POST /matches/{id}/save - Toggle bookmark

Branch Status Values

Status	Color	Condition
available	Green	Open + occupancy below 70%
busy	Orange	Open + occupancy 70-90%
full	Red	Open + occupancy above 90%
closed	Gray	Currently not open

5. Real-time Chat and WebSockets

Architecture

Laravel Reverb (self-hosted, Pusher-protocol compatible). NOT Pusher cloud service.

WebSocket Connection (Flutter)

```
// pubspec.yaml
pusher_channels_flutter: ^2.0.0

// Connection
final pusher = PusherChannelsFlutter.getInstance();
await pusher.init(
apiKey: 'matchday-key',
cluster: '',
host: '10.0.2.2', // or production domain
port: 8080,
useTLS: false, // true in production
authEndpoint: '$baseUrl/broadcasting/auth',
);
await pusher.connect();
```

Two Chat Room Types

Type	Access	Endpoint
Public Fan Room	Any auth user	GET /chat/rooms/{matchId}
Cafe Room	Booking holders only	GET /chat/rooms/{matchId}/branch/{branchId}

REST API (Works Without WebSocket)

GET **/chat/rooms/{matchId}** - Get/create public room

GET **/chat/rooms/{matchId}/branch/{branchId}** - Cafe room (403 if no booking)

GET **/chat/rooms/{roomId}/messages?before=...** - Paginated (30/page)

POST **/chat/rooms/{roomId}/messages** - Send: { message(500 max), type }

POST **/chat/rooms/{roomId}/reaction** - { emoji: goal|heart|fire|clap|star }

GET **/chat/rooms/{roomId}/viewers** - Viewer count

GET **/chat/rooms/{roomId}/online-users** - Online users (max 20)

WARNING: REST API works perfectly WITHOUT WebSocket. Messages save to DB via REST.

WebSocket Events

```
// Subscribe
await pusher.subscribe(channelName: 'presence-chat.$roomId');

// Listen
pusher.onEvent = (event) {
switch (event.eventName) {
case 'message.sent': // New message from other user
case 'reaction.sent': // Floating emoji animation
case 'viewer.count.updated': // Update viewer counter
}
};
```

Complete Chat Pattern

```
// 1. Get room
final room = await api.get('/chat/rooms/$matchId');

// 2. Load messages
final msgs = await api.get('/chat/rooms/${room.id}/messages');

// 3. Subscribe WebSocket
await pusher.subscribe(channelName: 'presence-chat.${room.id}');

// 4. Send via REST (WebSocket delivers to others)
await api.post('/chat/rooms/${room.id}/messages', {'message': text});

// 5. Infinite scroll older
await api.get('/chat/rooms/${room.id}/messages?before=$oldest');

// 6. Leave
await pusher.unsubscribe(channelName: 'presence-chat.${room.id}');
```

6. QR Code System

How It Works

* QR generated server-side when booking is created
* Returned as **base64 PNG** string
* Fan shows QR at cafe entrance
* Staff scans QR - auto check-in

6.1 Fan Side - Display QR

POST **/bookings** - Create booking - QR in response

GET **/bookings/{id}/pass** - Entry pass with QR

```
// Response includes:
"qr_code": "data:image/png;base64,iVBORw0KGgo..."

// Flutter Display:
final base64Str = qrCode.split(',').last;
final bytes = base64Decode(base64Str);
Image.memory(Uint8List.fromList(bytes))
```

6.2 Staff Side - Scan QR

POST **/cafe-admin/scan-qr** - Scan QR or booking code

```
// REQUEST
{ "qr_code": "BOOK-ABC123" }
// OR
{ "booking_code": "#MD-2026-1234" }

// SUCCESS - auto check-in
{
"success": true,
"message": "Check-in successful",
"data": {
"booking": { "status": "checked_in", "customer": { "name": "Ahmed" } },
"checked_in_at": "2026-02-15T19:30:00Z"
}
}
```

Flutter QR Scanner

```
// pubspec.yaml: mobile_scanner: ^3.0.0

onDetect: (barcode) async {
final response = await api.post('/cafe-admin/scan-qr', {
'qr_code': barcode.rawValue,
});
if (response['success']) showSuccess(response['data']);
}
```

6.3 Manual Check-in

POST **/cafe-admin/bookings/{id}/check-in** - By booking ID

GET **/cafe-admin/scan-qr/recent** - Last 10 scans

GET **/cafe-admin/scan-qr/stats** - Today's stats

7. Booking Flow (Step by Step)

Complete Journey

```
Step 1: Browse
GET /matches?status=upcoming
GET /matches/live

Step 2: Select Match - View Seating
GET /matches/{id}/seating
-> sections with seats + is_booked_for_this_match flag

Step 3: Select Seats - Create Booking
POST /bookings
{ match_id: 1, seat_ids: [5,6,7], guests_count: 3 }
-> Returns QR code + payment info

Step 4: Add Payment Method (if needed)
POST /payment-methods
{ type: 'credit_card', card_last_four: '4242', ... }

Step 5: Process Payment
POST /payments/process
{ booking_id: 1, payment_method_id: 1 }
-> status: pending -> confirmed

Step 6: Confirmed!
-> Show QR from booking
-> POST /bookings/{id}/add-to-calendar (ICS file)

Step 7: Match Day
-> GET /bookings/{id}/pass (entry pass)
-> POST /bookings/{id}/enter-fan-room (join chat)
```

Booking Tabs

GET **/bookings?tab=upcoming** - Active (confirmed/pending/checked_in)

GET **/bookings?tab=past** - Completed

GET **/bookings?tab=cancelled** - Cancelled

```
// Meta includes tab counts:
"meta": { "tabs": { "upcoming": 2, "past": 5, "cancelled": 1 } }
```

UI Flags in Booking Detail

Flag	When True
can_cancel	Confirmed/pending AND match not started
can_rebook	Past or cancelled booking
can_enter_fan_room	Confirmed/checked_in AND match is live
is_today	Match is today

Cancel and Rebook

POST **/bookings/{id}/cancel** - Cancel + auto-refund + release seats

POST **/bookings/{id}/rebook** - Find next match with same teams

POST **/bookings/{id}/share** - Shareable text for social media

Players

POST **/bookings/{id}/players** - Add: { name, position, jersey_number? }

GET **/bookings/{id}/players** - List players

DELETE **/bookings/{id}/players/{pid}** - Remove

Price Breakdown

```
"subtotal": 120.00, // seat_price x count + section extras
"service_fee": 6.00, // 5% of subtotal, min 5 SAR
"total_amount": 126.00,
"currency": "SAR"
```

8. Payment Integration

Payment Methods

GET **/payment-methods** - List saved cards

POST **/payment-methods** - Add card

```
{ "type": "credit_card", "card_last_four": "4242",
"card_holder": "Ahmed Hassan", "expires_at": "2027-12", "is_primary": true }
```

PUT **/payment-methods/{id}** - Update holder/expiry

PUT **/payment-methods/{id}/set-primary** - Set as primary

DELETE **/payment-methods/{id}** - Delete (not if active payments)

WARNING: Backend uses SIMULATED gateway. Production needs Stripe/Tap integration.

Process Payment

POST **/payments/process** - Pay for booking

```
// REQUEST
{ "booking_id": 1, "payment_method_id": 1 }

// SUCCESS - booking becomes confirmed
{
"amount": "126.00", "currency": "SAR", "status": "paid",
"gateway_ref": "sim_123...", "booking": { "status": "confirmed" }
}
```

Refund

POST **/payments/{id}/refund** - Refund - cancels booking + releases seats

Payment History

GET **/payments/history?type=booking&status=paid&period=month** - Filtered history

9. Push Notifications (FCM)

Register Device Token

PUT /profile/device-token - Send FCM token after login

```
{ "device_token": "fMJK9sd8f..." }
```

TIP: Call after login AND after FCM token refresh

Notification Types

Type	Trigger	Deep Link
booking_confirmed	Payment processed	bookings/{id}
booking_cancelled	Booking cancelled	bookings/{id}
match_reminder	1hr before kick-off	matches/{id}
match_score_update	Score changes (live)	matches/{id}
achievement_unlocked	New achievement	achievements
points_earned	Points added	loyalty
welcome	After registration	home
staff_invitation	Invited to cafe	invite/{token}

FCM Payload

```
{
"notification": { "title": "Booking Confirmed!", "body": "Your booking #MD-2026-1234 is
confirmed" },
"data": { "type": "booking_confirmed", "booking_id": "123" }
}
```

Notification Settings

GET /notifications/settings - Get preferences

PUT /notifications/settings - Update

```
{ "booking_reminders": true, "match_updates": true,
"promotions": false, "chat_messages": true }
```

In-App Notifications

GET /notifications - Paginated (20/page)

GET /notifications/unread-count - { count: 5 }

PUT /notifications/{id}/read - Mark read

PUT /notifications/read-all - Mark all read

DELETE /notifications/{id} - Delete

10. Image Uploads and Multi-size

Upload Endpoints

POST **/profile/avatar** - User avatar

POST **/cafe-admin/cafe/logo** - Cafe logo

POST **/cafe-admin/offers/{id}/upload-image** - Offer image

Max: **5MB** | Formats: **JPEG, PNG, WEBP** | Multipart form-data

Response (All Uploads)

```
"avatar": {
"original": "https://api.matchday.app/storage/avatars/original/abc.jpg",
"medium": "https://api.matchday.app/storage/avatars/medium/abc.jpg", // 300x300
"thumbnail": "https://api.matchday.app/storage/avatars/thumbnail/abc.jpg" // 150x150
}
```

Flutter Upload

```
// Using http package
var request = http.MultipartRequest('POST', Uri.parse('$baseUrl/profile/avatar'));
request.headers['Authorization'] = 'Bearer $token';
request.files.add(await http.MultipartFile.fromPath('avatar', imagePath));
var response = await request.send();

// Using Dio
final formData = FormData.fromMap({
'avatar': await MultipartFile.fromFile(imagePath),
});
await dio.post('/profile/avatar', data: formData);
```

Which Size to Use

Context	Use
Chat avatar, small lists	thumbnail (150x150)
Cards, grid items	medium (300x300)
Profile, detail, hero	original (full)

11. Pagination and Filtering

Standard Pagination

```
GET /bookings?page=2&per;_page=15

"meta": { "current_page": 2, "last_page": 10, "per_page": 15, "total": 150 }
```

per_page: default 15, max 100

Common Filters

Endpoint	Filters
GET /matches	?status= &date= &team_id= &league= &branch_id=
GET /bookings	?tab=upcoming|past|cancelled
GET /cafes	?featured= &city= &search=
GET /cafes/nearby	?lat= &lng= &radius=
GET /payments/history	?type= &status= &period=
GET /cafe-admin/bookings	?status= &match_id= &date=
GET /cafe-admin/analytics/*	?period= &week_offset= &start_date= &end_date=

Chat Infinite Scroll

```
// Newest first
GET /chat/rooms/1/messages

// Load older
GET /chat/rooms/1/messages?before=2026-02-15T15:00:00Z
```

12. Loyalty and Achievements

Tiers

Tier	Min Points	Benefits
Bronze	0	Basic rewards, 10pts/booking
Silver	500	Priority booking, 15pts/booking
Gold	1000	VIP access, discounts, 20pts/booking
Platinum	2500	All benefits, dedicated support, 30pts/booking

Endpoints

GET **/loyalty/card** - Points, tier, next_tier, progress

GET **/loyalty/tiers** - All tiers with benefits

GET **/loyalty/transactions?type=earned** - Point history

GET **/loyalty/progress** - Progress to next tier

GET **/achievements** - All with unlock status

GET **/achievements/unlocked** - User's unlocked only

GET **/achievements/progress** - Progress on each

Points auto-awarded on booking confirmation. Tier upgrades are automatic.

13. Cafe Admin API (~85 Endpoints)

All require auth + spatie permission. Base: **/api/v1/cafe-admin**

Module	Count	Permission	Key Features
Cafe and Onboarding	5	manage-cafe-profile	3-step onboarding, logo upload
Branches	11	manage-branches	Multi-step create, hours, amenities
Seating	9	manage-seating	Sections, auto-label seats
Matches	9	manage-matches	Publish, live score, broadcast
Bookings	5	view-bookings	List, check-in, today summary
QR Scan	4	scan-qr	Scan, upload image, stats
Occupancy	4	view-occupancy	Live %, peak times, sections
Staff	8	manage-staff	Invite, roles, permissions
Offers	7	manage-offers	CRUD, image, auto-expire
Dashboard	3	view-bookings	Overview, upcoming, recent
Analytics	7	view-analytics	Charts, trends, peak hours
Subscription	4	manage-subscription	Plans, upgrade, cancel
Billing	4	manage-subscription	History, export CSV

Onboarding Flow

```
Step 1: POST /cafe-admin/cafe (create profile)
Step 2: POST /cafe-admin/branches (add branch)
-> PUT /cafe-admin/branches/{id}/hours (set hours)
-> POST /cafe-admin/branches/{id}/amenities/bulk (add amenities)
-> POST /cafe-admin/branches/{id}/sections/bulk (add seating)
Step 3: POST /cafe-admin/staff (invite staff - optional)

Check: GET /cafe-admin/onboarding-status
-> { step: '1'|'2'|'3'|'complete' }
```

Match Lifecycle

```
Create (draft) -> Publish -> Live -> Finished

POST /cafe-admin/matches (create unpublished)
POST /cafe-admin/matches/{id}/publish (visible to fans)
PUT /cafe-admin/matches/{id}/status (upcoming->live->finished)
PUT /cafe-admin/matches/{id}/score (update + broadcast)
```

Staff Roles

Role	Default Permissions
admin	All permissions
manager	bookings, matches, analytics, QR, occupancy

staff	view-bookings, scan-qr, check-in-customers

14. All Endpoints Quick Reference

Authentication (12)

- POST **/auth/register** -
- POST **/auth/register/cafe-owner** -
- POST **/auth/login** -
- POST **/auth/login/google** -
- POST **/auth/login/apple** -
- POST **/auth/logout** -
- POST **/auth/refresh** -
- GET **/auth/me** -
- POST **/auth/forgot-password** -
- POST **/auth/reset-password** -
- POST **/auth/verify-email** -
- POST **/auth/resend-verification** -

Profile (9)

- GET **/profile** -
- PUT **/profile** -
- POST **/profile/avatar** -
- PUT **/profile/password** -
- PUT **/profile/locale** -
- PUT **/profile/device-token** -
- PUT **/profile/favorite-team** -
- GET **/profile/activity** -
- DELETE **/profile** -

Teams (4)

- GET **/teams** -
- GET **/teams/popular** -
- GET **/teams/search** -
- GET **/teams/{id}** -

Cafes and Branches (9)

- GET **/cafes** -
- GET **/cafes/search** -
- GET **/cafes/nearby** -
- GET **/cafes/{id}** -
- GET **/cafes/{id}/branches** -
- GET **/branches/{id}** -
- GET **/branches/{id}/matches** -
- GET **/branches/{id}/reviews** -
- POST **/branches/{id}/reviews** -

Matches (8)

GET /matches -

GET /matches/live -

GET /matches/upcoming -

GET /matches/popular -

GET /matches/saved -

POST /matches/{id}/save -

GET /matches/{id} -

GET /matches/{id}/seating -

Bookings (13)

POST /bookings -

GET /bookings -

GET /bookings/{id} -

PUT /bookings/{id} -

POST /bookings/{id}/cancel -

GET /bookings/{id}/pass -

POST /bookings/{id}/share -

POST /bookings/{id}/add-to-calendar -

POST /bookings/{id}/rebook -

POST /bookings/{id}/enter-fan-room -

GET /bookings/{id}/players -

POST /bookings/{id}/players -

DELETE /bookings/{id}/players/{pid} -

Payments (8)

GET /payment-methods -

POST /payment-methods -

PUT /payment-methods/{id} -

PUT /payment-methods/{id}/set-primary -

DELETE /payment-methods/{id} -

POST /payments/process -

GET /payments/history -

POST /payments/{id}/refund -

Saved (5)

GET /saved-cafes -

POST /saved-cafes/{id} -

DELETE /saved-cafes/{id} -

GET /matches/saved -

POST /matches/{id}/save -

Loyalty (8)

GET /loyalty/tiers -

GET **/loyalty/card** -

GET **/loyalty/transactions** -

GET **/loyalty/progress** -

GET **/achievements** -

GET **/achievements/unlocked** -

GET **/achievements/progress** -

GET **/achievements/my** -

Chat (7)

GET **/chat/rooms/{matchId}** -

GET **/chat/rooms/{matchId}/branch/{branchId}** -

GET **/chat/rooms/{roomId}/messages** -

POST **/chat/rooms/{roomId}/messages** -

POST **/chat/rooms/{roomId}/reaction** -

GET **/chat/rooms/{roomId}/viewers** -

GET **/chat/rooms/{roomId}/online-users** -

Notifications (7)

GET **/notifications** -

GET **/notifications/unread-count** -

PUT **/notifications/{id}/read** -

PUT **/notifications/read-all** -

DELETE **/notifications/{id}** -

GET **/notifications/settings** -

PUT **/notifications/settings** -

Home and Explore (3)

GET **/home** -

GET **/explore** -

GET **/search** -

Support (7)

POST **/support/contact** -

POST **/support/report-issue** -

GET **/support/my-tickets** -

GET **/faqs** -

GET **/pages/{slug}** -

GET **/app/version** -

GET **/app/config** -

Offers (2)

GET **/offers** -

GET **/offers/{id}** -

Cafe Admin (~85 endpoints)

See Section 13 above. All under /cafe-admin/* prefix.

15. FAQ for Flutter Developers

Q: How do I handle token storage?

Use **flutter_secure_storage** package. Save token after login/register. Read on app start. Clear on logout or any 401 response.

Q: Does the token expire?

No. Sanctum tokens do not expire. They are only revoked on POST /auth/logout. You can optionally call POST /auth/refresh to rotate tokens.

Q: How do I know if user needs to select a team?

Check **needs_team_selection** flag in GET /auth/me response. If true, show team selection screen. User can skip it.

Q: What if WebSocket server is down?

REST API works independently. All chat messages save to DB via REST endpoints. WebSocket is only for real-time push. Users can pull-to-refresh to see new messages.

Q: How do I display the QR code?

QR comes as base64 PNG. Strip 'data:image/png;base64,' prefix, decode with base64Decode(), display with Image.memory(). See Section 6 for code.

Q: How does the seating map work?

GET /matches/{id}/seating returns sections (VIP, Premium, Standard) each with seats array. Each seat has **is_booked_for_this_match** boolean. Show booked seats as disabled/gray.

Q: What currency is used?

Default is **SAR** (Saudi Riyal). The currency field is included in booking and payment responses. Check GET /app/config for default_currency.

Q: How do I handle image URLs?

All images return 3 sizes: original, medium (300x300), thumbnail (150x150). Use thumbnail for lists/avatars, medium for cards, original for full-screen. Use **cached_network_image** package.

Q: How does booking cancellation work?

POST /bookings/{id}/cancel auto-handles everything: updates status, releases seats (increments seats_available), processes refund. Check **can_cancel** flag in booking detail.

Q: How do I implement infinite scroll for chat?

Load newest first with GET /chat/rooms/{id}/messages. For older messages, pass **?before=** with the created_at of the oldest message in your list.

Q: What's the difference between /home and /explore?

/home = personalized (upcoming booking, nearby cafes based on location). **/explore** = discovery (featured, trending, popular across platform). Both work without auth but are enhanced with it.

Q: How do I handle the cafe owner vs fan flow?

Check user.role after login. 'fan' = show fan app screens. 'cafe_owner' = show cafe admin screens. Cafe owners use /cafe-admin/* endpoints with permission-based access.

Q: What packages do I need in Flutter?

http or **dio** (API calls), **flutter_secure_storage** (token), **pusher_channels_flutter** (WebSocket), **mobile_scanner** (QR scan), **cached_network_image** (images), **firebase_messaging** (FCM).

Q: How do I test without real data?

Run **php artisan db:seed** on the backend. It creates test users, cafes, matches, bookings. Login with test accounts. Default fan: test@matchday.app / password.

Q: Does the API support Arabic?

Yes. Set locale with PUT /profile/locale { locale: 'ar' }. Error messages and content will be in Arabic where translations exist.

Q: How are push notifications structured?

FCM data payload includes **type** (e.g. booking_confirmed) and relevant IDs. Use type for routing: booking_confirmed -> navigate to booking detail screen.

Q: What about offline support?

The API requires internet. For offline UX: cache the last home feed, bookings list, and user profile locally. Show cached data with 'Last updated X min ago' indicator.

Q: How do I handle the 3-dot menu on cafe cards?

Options: View Details (navigate), Share (share cafe link), Report (POST /support/report-issue). No admin actions for fans.

MATCHDAY API

~263 Endpoints | ~43 Tables | 367 Tests | 100% Pass

Laravel 12 | Sanctum | MySQL 8 | Reverb

Postman Collection: docs/Matchday_API_Complete.postman_collection.json

February 2026 | Version 1.0