Sanbot Android App - API Contract

Project: TripAndEvent Sanbot Interactive Assistant

Document Version: 1.0

Date: November 26, 2025

1. Overview

This document defines the API endpoints, request/response formats, and authentication methods for the Sanbot Android application. All endpoints use HTTPS and return JSON responses.

Base URL (Production):

Plain Text

```
https://bot.tripandevent.com/api
```

Note: This is the actual production API. For testing during development, use test API tokens that will be provided separately.

2. Authentication

All API requests require an API key in the request header.

Header:

Plain Text

```
Authorization: Bearer YOUR_API_KEY_HERE
Content-Type: application/json
```

Test API Key:

Plain Text

```
test_sanbot_key_abc123xyz789
```

Note: Production keys will be provided separately. Do NOT hardcode keys in the APK.

3. Voice Interaction API

3.1 Speech to Text (Transcription)

Endpoint: `POST /voice.transcribe`

Purpose: Convert customer speech to text using Whisper AI.

Request:

```JSON
{
  "audio": "BASE64_ENCODED_AUDIO_DATA",
  "format": "wav",
  "language": "en"
}
```

Response:

```JSON
{
  "success": true,
  "text": "Show me Dubai packages",
  "confidence": 0.95,
  "language": "en",
  "duration_seconds": 2.3
}
```

Error Response:

```JSON
{
  "success": false,
  "error": {
    "code": "AUDIO_TOO_LARGE",
    "message": "Audio file exceeds 5MB limit"
  }
}
```

Status Codes:

- `200` - Success
- `400` - Bad request (invalid audio format)

- `401` - Unauthorized (invalid API key)
- `413` - Payload too large
- `500` - Server error

3.2 Text to Speech (Speech Generation)

Endpoint: `POST /voice.generateSpeech`

Purpose: Convert text to speech using OpenAI or ElevenLabs.

Request:

```json
{
  "text": "Hello! How can I help you today?",
  "voice": "alloy",
  "language": "en"
}
```

Response:

```json
{
  "success": true,
  "audio_url": "https://bot.tripandevent.com/audio/response_123.mp3",
  "duration_seconds": 3.5,
  "format": "mp3"
}
```

Available Voices:

- `alloy` - Neutral, balanced voice
- `echo` - Male, clear voice
- `fable` - Warm, expressive voice
- `onyx` - Deep, authoritative voice
- `nova` - Female, energetic voice
- `shimmer` - Soft, gentle voice

3.3 Complete Voice Conversation

Endpoint: `POST /voice.conversation`

Purpose: Complete voice flow (Speech-to-Text → AI Processing → Text-to-Speech).

Request:

JSON

```json
{
  "audio": "BASE64_ENCODED_AUDIO_DATA",
  "format": "wav",
  "session_id": "unique_session_identifier",
  "language": "en",
  "voice": "alloy"
}
```

Response:

JSON

```json
{
  "success": true,
  "session_id": "unique_session_identifier",
  "transcript": "Show me Dubai packages",
  "response": {
    "text": "Here are our top Dubai packages: Desert Safari, Burj Khalifa Tour, and Dubai Marina Cruise.",
    "audio_url": "https://bot.tripandevent.com/audio/response_456.mp3",
    "duration_seconds": 5.2
  },
  "intent": "package_inquiry",
  "confidence": 0.92
}
```

Error Response:

JSON

```json
{
  "success": false,
  "error": {
    "code": "AUDIO_TOO_LARGE",
    "message": "Audio file exceeds 5MB limit"
  }
}
```

Status Codes:

- `200` - Success
- `400` - Bad request (invalid audio format)
- `401` - Unauthorized (invalid API key)
- `413` - Payload too large
- `500` - Server error

3.4 Session Management

Session ID Format: Generate a unique session ID for each customer interaction using UUID v4 format:

Plain Text

```
session_id = "sanbot_" + UUID.randomUUID().toString()
// Example: "sanbot_123e4567-e89b-12d3-a456-426614174000"
```

Session Persistence:

- Sessions should persist for the duration of the customer interaction
- Reset session when customer returns to Welcome Screen
- Include session_id in all voice API calls for conversation continuity

4. Package Management API

4.1 Get Package List

Endpoint: `GET /packages`

Purpose: Fetch all available tour packages.

Query Parameters:

- `category` (optional): Filter by category (e.g., "dubai", "europe")
- `min_price` (optional): Minimum price filter
- `max_price` (optional): Maximum price filter
- `limit` (optional): Number of results (default: 20)

Request Example:

Plain Text

```
GET /packages?category=dubai&limit=10
```

Response:

JSON

```json
{
  "success": true,
  "total_count": 45,
  "packages": [
    {
      "id": "PKG001",
      "name": "Dubai Desert Safari",
      "category": "dubai",
      "price": 299,
      "currency": "AED",
      "duration": "6 hours",
      "image_url": "https://cdn.tripandevent.com/images/pkg001.jpg",
      "thumbnail_url":
"https://cdn.tripandevent.com/images/pkg001_thumb.jpg",
      "highlights": [
        "Dune bashing",
        "Camel ride",
        "BBQ dinner",
        "Traditional entertainment"
      ],
      "description": "Experience the thrill of desert adventure with our
premium safari package",
      "rating": 4.8,
      "reviews_count": 234,
      "available": true
    }
  ]
}
```

4.2 Get Package Details

Endpoint: `GET /packages/{package_id}`

Purpose: Get detailed information about a specific package.

Response:

JSON

```json
{
  "success": true,
  "package": {
    "id": "PKG001",
    "name": "Dubai Desert Safari",
    "category": "dubai",
    "price": 299,
    "currency": "AED",
    "duration": "6 hours",
    "images": [
      "https://cdn.tripandevent.com/images/pkg001_1.jpg",
      "https://cdn.tripandevent.com/images/pkg001_2.jpg",
      "https://cdn.tripandevent.com/images/pkg001_3.jpg"
    ],
    "video_url": "https://cdn.tripandevent.com/videos/pkg001.mp4",
    "highlights": [
      "Dune bashing",
      "Camel ride",
      "BBQ dinner"
    ],
    "description": "Full description here...",
    "itinerary": [
      {
        "time": "15:00",
        "activity": "Hotel pickup"
      },
      {
        "time": "16:00",
        "activity": "Dune bashing"
      }
    ],
    "inclusions": [
      "Hotel pickup and drop-off",
      "BBQ dinner",
      "Soft drinks"
    ],
    "exclusions": [
      "Alcoholic beverages",
      "Personal expenses"
    ],
    "rating": 4.8,
    "reviews_count": 234,
    "available": true
  }
}
```

5. CRM API (Lead Management)

5.1 Create Lead

Endpoint: `POST /crm/leads`

Purpose: Create a new lead in the CRM system.

Request:

JSON

```json
{
  "name": "Ahmed Ali",
  "phone": "+971501234567",
  "email": "ahmed@example.com",
  "source": "Sanbot",
  "location": "Dubai Office",
  "interested_packages": ["PKG001", "PKG003"],
  "notes": "Interested in family packages for December",
  "preferred_contact": "whatsapp"
}
```

Response:

JSON

```json
{
  "success": true,
  "lead_id": "LEAD_20251126_001",
  "message": "Lead created successfully",
  "created_at": "2025-11-26T10:30:00Z"
}
```

Error Response:

JSON

```json
{
  "success": false,
  "error": {
    "code": "INVALID_PHONE",
    "message": "Phone number format is invalid",
    "field": "phone"
  }
}
```

Validation Rules:

- `name` : Required, 2-100 characters
- `phone` : Required, valid international format
- `email` : Optional, valid email format
- `source` : Required, must be "Sanbot"
- `interested_packages` : Optional, array of package IDs

5.2 Update Lead

Endpoint: `PUT /crm/leads/{lead_id}`

Purpose: Update an existing lead with additional information.

Request:

```JSON
{
  "notes": "Customer requested callback at 3 PM",
  "status": "contacted",
  "interested_packages": ["PKG001", "PKG002", "PKG005"]
}
```

Response:

```JSON
{
  "success": true,
  "lead_id": "LEAD_20251126_001",
  "message": "Lead updated successfully",
  "updated_at": "2025-11-26T11:00:00Z"
}
```

5.3 Add Note to Lead

Endpoint: `POST /crm/leads/{lead_id}/notes`

Purpose: Add a note/remark to an existing lead.

Request:

JSON

```
{
  "note": "Customer showed interest in luxury packages",
  "author": "Sanbot"
}
```

Response:

```JSON
{
  "success": true,
  "note_id": "NOTE_123",
  "message": "Note added successfully"
}
```

---

6. Customer Actions API

6.1 Send SMS

Endpoint: `POST /actions/send-sms`

Purpose: Send package details to customer via SMS.

Request:

```JSON
{
  "phone": "+971501234567",
  "package_id": "PKG001",
  "template": "package_details",
  "lead_id": "LEAD_20251126_001"
}
```

Response:

```JSON
{
  "success": true,
  "message_id": "SMS_123456",
  "status": "sent",
  "message": "SMS sent successfully"
}
```

6.2 Send WhatsApp Message

Endpoint: `POST /actions/send-whatsapp`

Purpose: Send package details to customer via WhatsApp.

Request:

```JSON
{
  "phone": "+971501234567",
  "package_id": "PKG001",
  "template": "package_details",
  "lead_id": "LEAD_20251126_001"
}
```

Response:

```JSON
{
  "success": true,
  "message_id": "WA_123456",
  "status": "sent",
  "message": "WhatsApp message sent successfully"
}
```

6.3 Send Email Quote

Endpoint: `POST /actions/send-email`

Purpose: Send detailed quote to customer via email.

Request:

```JSON
{
  "email": "ahmed@example.com",
  "package_ids": ["PKG001", "PKG003"],
  "template": "quote",
  "lead_id": "LEAD_20251126_001",
  "customer_name": "Ahmed Ali"
}
```

Response:

```json
{
  "success": true,
  "email_id": "EMAIL_123456",
  "status": "sent",
  "message": "Email sent successfully"
}
```

6.4 Create Booking Request

Endpoint: `POST /actions/book-now`

Purpose: Create a booking request for a package.

Request:

```json
{
  "lead_id": "LEAD_20251126_001",
  "package_id": "PKG001",
  "travel_date": "2025-12-15",
  "number_of_people": 4,
  "special_requests": "Need vegetarian meals"
}
```

Response:

```json
{
  "success": true,
  "booking_id": "BOOK_123456",
  "status": "pending",
  "message": "Booking request created. Our team will contact you soon."
}
```

7. Media API

7.1 Get Media Library

Endpoint: `GET /media`

Purpose: Fetch promotional videos and images.

Query Parameters:

- `type` : "video" or "image"
- `category` : Filter by category

Response:

```json
{
  "success": true,
  "media": [
    {
      "id": "MEDIA_001",
      "type": "video",
      "title": "Dubai Highlights",
      "url": "https://cdn.tripandevent.com/videos/dubai_highlights.mp4",
      "thumbnail": "https://cdn.tripandevent.com/images/dubai_thumb.jpg",
      "duration_seconds": 120,
      "category": "dubai"
    },
    {
      "id": "MEDIA_002",
      "type": "image",
      "title": "Burj Khalifa",
      "url": "https://cdn.tripandevent.com/images/burj_khalifa.jpg",
      "category": "dubai"
    }
  ]
}
```

8. Configuration API

8.1 Get App Configuration

Endpoint: `GET /config`

Purpose: Fetch app configuration settings.

Response:

JSON

```
{
  "success": true,
  "config": {
    "app_version": "1.0.0",
    "min_supported_version": "1.0.0",
    "welcome_message": "Welcome to TripAndEvent",
    "idle_timeout_seconds": 120,
    "default_language": "en",
    "supported_languages": ["en", "ar"],
    "features": {
      "voice_enabled": true,
      "video_enabled": true,
      "whatsapp_enabled": true,
      "sms_enabled": true,
      "email_enabled": true
    }
  }
}
```

9. Health Check API

9.1 Check API Status

Endpoint: `GET /health`

Purpose: Check if API is accessible.

Response:

JSON

```
{
  "status": "healthy",
  "timestamp": "2025-11-26T10:30:00Z",
  "version": "1.0.0"
}
```

10. Error Handling

Standard Error Response Format

All errors follow this format:

```json
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "field": "field_name_if_applicable",
    "details": {}
  }
}
```

Common Error Codes

Code	Description	HTTP Status
`INVALID_API_KEY`	API key is missing or invalid	401
`INVALID_REQUEST`	Request format is incorrect	400
`MISSING_FIELD`	Required field is missing	400
`INVALID_PHONE`	Phone number format is invalid	400
`INVALID_EMAIL`	Email format is invalid	400
`PACKAGE_NOT_FOUND`	Package ID does not exist	404
`LEAD_NOT_FOUND`	Lead ID does not exist	404
`RATE_LIMIT_EXCEEDED`	Too many requests	429
`SERVER_ERROR`	Internal server error	500
`SERVICE_UNAVAILABLE`	Service temporarily unavailable	503

---

11. Rate Limiting

Limits:

- 100 requests per minute per API key
- 1000 requests per hour per API key

Rate Limit Headers:

Plain Text

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1732612800
```

Rate Limit Exceeded Response:

JSON

```json
{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Too many requests. Please try again in 60 seconds."
  }
}
```

---

12. Testing

Test Data

Test Phone Numbers:

- `+971501111111` - Always succeeds
- `+971502222222` - Always fails (for error testing)

Test Package IDs:

- `PKG_TEST_001` - Dubai Desert Safari
- `PKG_TEST_002` - Burj Khalifa Tour
- `PKG_TEST_003` - Dubai Marina Cruise

Test Lead ID:

- `LEAD_TEST_001`

Sample cURL Commands

Get Packages:

```bash
curl -X GET "https://api-test.tripandevent.com/sanbot/v1/packages" \
  -H "Authorization: Bearer test_sanbot_key_abc123xyz789" \
  -H "Content-Type: application/json"
```

Create Lead:

```bash
curl -X POST "https://api-test.tripandevent.com/sanbot/v1/crm/leads" \
  -H "Authorization: Bearer test_sanbot_key_abc123xyz789" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Test Customer",
    "phone": "+971501111111",
    "source": "Sanbot",
    "location": "Dubai Office"
  }'
```

---

13. Best Practices

1. **Always use HTTPS** - Never use HTTP in production
2. **Store API keys securely** - Use Android Keystore or encrypted SharedPreferences
3. **Implement retry logic** - Retry failed requests with exponential backoff
4. **Cache responses** - Cache package data to improve performance
5. **Handle errors gracefully** - Show user-friendly error messages
6. **Validate input** - Validate all user input before sending to API
7. **Use timeouts** - Set reasonable timeouts for all API calls (30 seconds recommended)
8. **Log errors** - Log API errors for debugging (but don't log sensitive data)

Next Document: UI/UX Requirements