RAG-ENABLED REAL ESTATE AI ASSISTANT
API CONTRACTS DOCUMENTATION

Version: 1.0.0
Base URL: http://localhost:8000
Documentation: http://localhost:8000/docs

TABLE OF CONTENTS

1. GENERAL INFORMATION

This API provides conversational AI capabilities with RAG (Retrieval-Augmented Generation), portfolio analysis, document management, and CRM functionality for commercial real estate applications.

Authentication: None required (development mode)
Content-Type: application/json (except file uploads: multipart/form-data)
CORS: Enabled for all origins

2. INPUT FORMATS

2.1 CHAT REQUEST

```
{
  "user_id": "string (required)",
  "message": "string (required)",
  "session_id": "string (optional)"
}
```

2.2 PORTFOLIO ANALYSIS REQUEST

```
{
  "user_id": "string (required)",
  "query": "string (required)",
  "return_chart": "boolean (optional, default: false)",
  "download_csv": "boolean (optional, default: false)"
}
```

2.3 SESSION CREATION REQUEST

```
{
  "user_id": "string (required)",
  "title": "string (optional)"
```

```
}

2.4 SESSION TITLE UPDATE REQUEST
------------------------------
{
  "title": "string (required)"
}

2.5 USER CREATION REQUEST
-----------------------
{
  "name": "string (required)",
  "email": "string (required)",
  "company": "string (optional)",
  "preferences": "string (optional)"
}

2.6 USER UPDATE REQUEST
--------------------
{
  "name": "string (optional)",
  "email": "string (optional)",
  "company": "string (optional)",
  "preferences": "string (optional)"
}

2.7 MESSAGE TAG UPDATE REQUEST
----------------------------
{
  "message_id": "string (required)",
  "tag": "string (required)"
}

2.8 DOCUMENT ADDITION REQUEST
---------------------------
{
  "documents": ["array of strings (required)"]
}

2.9 FILE UPLOAD REQUEST
--------------------
Content-Type: multipart/form-data
Field: files (List[UploadFile])
Supported formats: .txt, .pdf, .csv, .json

================================================================================
                             3. RESPONSE SCHEMAS
================================================================================

3.1 CHAT RESPONSE
---------------
{
  "response": "string",
  "session_id": "string"
}

3.2 PORTFOLIO ANALYSIS RESPONSE
-----------------------------
{
  "summary": "string",
  "matches": [
    {
      "Property Address": "string",
      "Floor": "string",
      "Suite": "string",
      "Size (SF)": "number",
      "Rent/SF/Year": "string (formatted currency)",
      "GCI On 3 Years": "string (formatted currency)",
      "Associate 1": "string",
      "Monthly Rent": "string"
    }
  ],
  "total_matches": "number",
  "query_interpretation": "string",
  "chart_url": "string (optional)",
  "csv_url": "string (optional)"
}
```

3.3 PORTFOLIO STATISTICS RESPONSE

```
{
  "total_properties": "number",
  "avg_size_sf": "number",
  "avg_rent_per_sf": "number",
  "avg_gci_3_years": "number",
  "size_range": {
    "min": "number",
    "max": "number"
  },
  "rent_range": {
    "min": "number",
    "max": "number"
  }
}
```

3.4 SESSION RESPONSE

```
{
  "id": "string",
  "title": "string",
  "created_at": "datetime",
  "updated_at": "datetime",
  "message_count": "number"
}
```

3.5 SESSION WITH CONVERSATIONS RESPONSE

```
{
  "id": "string",
  "title": "string",
  "created_at": "datetime",
  "updated_at": "datetime",
  "conversations": [
    {
      "id": "string",
      "message": "string",
      "role": "string",
      "timestamp": "datetime",
      "tag": "string"
    }
  ]
}
```

3.6 CONVERSATION RESPONSE

```
{
  "message": "string",
  "role": "string",
  "tag": "string (optional)",
  "timestamp": "string"
}

3.7 USER CREATION RESPONSE
------------------------
{
  "user_id": "string",
  "message": "string"
}

3.8 DOCUMENT UPLOAD RESPONSE
--------------------------
{
  "message": "string",
  "uploaded_files": ["array of strings"],
  "total_documents": "number"
}

3.9 DOCUMENT LIST RESPONSE
------------------------
{
  "documents": ["array of strings"],
  "total_count": "number"
}

3.10 GENERIC SUCCESS RESPONSE
--------------------------
{
  "message": "string"
}

3.11 ERROR RESPONSE
-----------------
{
  "detail": "string"
}

================================================================================
                              4. API ENDPOINTS
================================================================================

4.1 CHAT API
-----------

POST /chat/
Description: Send a message to the AI assistant with RAG capabilities
Request: ChatRequest
Response: ChatResponse
Tags: ["Chat"]

4.2 CHAT HISTORY API
------------------

GET /history/sessions/{user_id}
Description: Get all chat sessions for a user
Parameters: user_id (path)
Response: List[SessionResponse]
Tags: ["Chat History"]

GET /history/sessions/{user_id}/current
Description: Get current active session for a user
Parameters: user_id (path)
Response: SessionResponse
```

```
Tags: ["Chat History"]

GET /history/sessions/{user_id}/{session_id}
Description: Get a specific session with all conversations
Parameters: user_id (path), session_id (path)
Response: SessionWithConversations
Tags: ["Chat History"]

POST /history/sessions/create
Description: Create a new chat session
Request: CreateSessionRequest
Response: SessionResponse
Tags: ["Chat History"]

PUT /history/sessions/{session_id}/title
Description: Update session title
Parameters: session_id (path)
Request: UpdateSessionRequest
Response: Generic success message
Tags: ["Chat History"]

DELETE /history/sessions/{session_id}
Description: Delete a chat session
Parameters: session_id (path)
Response: Generic success message
Tags: ["Chat History"]

4.3 PORTFOLIO ANALYSIS API
-------------------------

POST /analyze/analyze_portfolio
Description: Analyze portfolio using natural language queries
Request: AnalyzeRequest
Response: AnalyzeResponse
Tags: ["Portfolio Analysis"]

GET /analyze/portfolio_stats
Description: Get portfolio overview statistics
Response: Portfolio statistics response
Tags: ["Portfolio Analysis"]

GET /analyze/download_chart/{filename}
Description: Download generated chart file
Parameters: filename (path)
Response: FileResponse
Tags: ["Portfolio Analysis"]

GET /analyze/download_csv/{filename}
Description: Download generated CSV file
Parameters: filename (path)
Response: FileResponse
Tags: ["Portfolio Analysis"]

4.4 DOCUMENT MANAGEMENT API
-------------------------

POST /chat/upload_docs
Description: Upload documents for RAG processing
Content-Type: multipart/form-data
Request: List[UploadFile]
Response: Document upload response
Tags: ["Upload"]

POST /chat/add-documents
```

Description: Add text documents directly to knowledge base
Request: Document addition request
Response: Generic success message
Tags: ["Upload"]

GET /chat/documents
Description: List all documents in knowledge base
Response: Document list response
Tags: ["Upload"]

DELETE /chat/documents/{document_id}
Description: Delete a specific document
Parameters: document_id (path)
Response: Generic success message
Tags: ["Upload"]

DELETE /chat/documents
Description: Clear all documents from knowledge base
Response: Generic success message
Tags: ["Upload"]

4.5 CRM API

POST /crm/create_user
Description: Create a new user
Request: UserCreate
Response: User creation response
Tags: ["CRM"]

PUT /crm/update_user/{user_id}
Description: Update user information
Parameters: user_id (path)
Request: UserUpdate
Response: Generic success message
Tags: ["CRM"]

GET /crm/conversations/{user_id}
Description: Get all conversations for a user
Parameters: user_id (path)
Response: List[ConversationResponse]
Tags: ["CRM"]

PUT /crm/tag_message
Description: Tag a specific message
Request: TagUpdate
Response: Generic success message
Tags: ["CRM"]

POST /crm/reset
Description: Reset the entire database (DANGER)
Response: Generic success message
Tags: ["CRM"]

==

5. SAMPLE API CALLS

==

5.1 CHAT WITH AI ASSISTANT

```
curl -X POST "http://localhost:8000/chat/" \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": "investor_123",
    "message": "What properties do you have available in Manhattan?"
  }'
```

Response:

```
{
  "response": "Based on our current portfolio, I found several properties in
Manhattan including...",
  "session_id": "session_456"
}
```

5.2 PORTFOLIO ANALYSIS

```
curl -X POST "http://localhost:8000/analyze/analyze_portfolio" \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": "investor_123",
    "query": "Show me properties above 15,000 SF with rent below $90/SF",
    "return_chart": true
  }'
```

Response:

```
{
  "summary": "Found 12 properties matching your criteria...",
  "matches": [
    {
      "Property Address": "1412 Broadway",
      "Suite": "500",
      "Size (SF)": 16434,
      "Rent/SF/Year": "$81.00",
      "GCI On 3 Years": "$239,608"
    }
  ],
  "total_matches": 12,
  "query_interpretation": "Applied filters: {\"Size (SF)\": {\"gt\": 15000},
\"Rent/SF/Year\": {\"lt\": 90}}",
  "chart_url": "/analyze/download_chart/chart_investor_123_abc123.png"
}
```

5.3 UPLOAD DOCUMENTS

```
curl -X POST "http://localhost:8000/chat/upload_docs" \
  -F "files=@property_list.pdf" \
  -F "files=@market_report.txt"
```

Response:

```
{
  "message": "Successfully uploaded 2 documents",
  "uploaded_files": ["property_list.pdf", "market_report.txt"],
  "total_documents": 15
}
```

5.4 GET USER SESSIONS

```
curl -X GET "http://localhost:8000/history/sessions/investor_123"
```

Response:

```
[
  {
    "id": "session_456",
    "title": "What properties do you have available...",
    "created_at": "2024-01-15T10:30:00",
    "updated_at": "2024-01-15T10:35:00",
    "message_count": 5
  }
]
```

5.5 CREATE USER

```
curl -X POST "http://localhost:8000/crm/create_user" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "John Smith",
    "email": "john.smith@realestate.com",
    "company": "Smith Properties"
  }'
```

Response:

```
{
  "user_id": "user_789",
  "message": "User created successfully."
}
```

5.6 PORTFOLIO STATISTICS

```
curl -X GET "http://localhost:8000/analyze/portfolio_stats"
```

Response:

```
{
  "total_properties": 199,
  "avg_size_sf": 14562.5,
  "avg_rent_per_sf": 94.23,
  "avg_gci_3_years": 245830.15,
  "size_range": {"min": 9010, "max": 19918},
  "rent_range": {"min": 80.00, "max": 110.00}
}
```

6. USAGE NOTES

6.1 AUTHENTICATION

- No authentication required in development mode
- Production deployments should implement proper authentication

6.2 RATE LIMITING

- No rate limiting implemented
- Consider implementing rate limits for production use

6.3 SESSION MANAGEMENT

- Sessions are automatically created if not provided
- Session titles are auto-generated from first message
- Sessions persist user conversation context

6.4 DOCUMENT PROCESSING

- Supported formats: PDF, TXT, CSV, JSON
- Maximum file size: Not specified (configure in production)
- Documents are processed with TF-IDF vectorization
- Knowledge base persists across sessions

6.5 NATURAL LANGUAGE QUERIES

- Portfolio analysis supports natural language
- Examples: "properties above 15,000 SF", "rent below $90/SF"
- Combines multiple criteria: size, rent, and GCI filters
- AI interprets and structures queries automatically

6.6 CONVERSATION TAGGING

- Messages auto-tagged as "Resolved" or "Inquiring"
- Manual tagging available via CRM endpoints
- Tags help categorize conversation types

6.7 CHART AND CSV GENERATION

- Charts generated as PNG files
- CSV exports available for filtered results
- Files accessible via download endpoints
- Temporary files cleaned up automatically

6.8 DATABASE MANAGEMENT

- SQLite database for development
- CRM tracks users, sessions, and conversations
- Reset endpoint available for testing (use with caution)

7. ERROR HANDLING

7.1 COMMON HTTP STATUS CODES

200 - Success
400 - Bad Request (invalid input)
404 - Not Found (resource doesn't exist)
422 - Validation Error (schema mismatch)
500 - Internal Server Error

7.2 ERROR RESPONSE FORMAT

```
{
  "detail": "Error description"
}
```

7.3 COMMON ERRORS

- Missing required fields in request body
- Invalid user_id or session_id
- File upload errors (unsupported format, corrupted file)
- OpenAI API errors (invalid key, rate limits)
- Database connection issues

7.4 DEBUGGING

- Check server logs for detailed error information
- Verify API endpoint URLs and HTTP methods
- Validate request body schema
- Ensure all required fields are provided

END OF DOCUMENT

For more detailed API documentation with interactive testing:
Visit: http://localhost:8000/docs

Last updated: 2024
Version: 1.0.0