

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) describes the functional and non-functional requirements, use case diagram, and detailed use case descriptions for the AI Lawyer application built using Streamlit. The system allows users to load constitutional texts for selected jurisdictions and interactively ask legal questions answered by an AI model.

1.2 Scope

The AI Lawyer application provides:

- Jurisdiction selection (Japan or Monaco)
- Constitutional text loading and session persistence
- Chat-based Q&A with an AI assistant using Google Gemini
- Session and chat management (new, delete, clear history)
- Question categorization and basic analytics

2. Overall Description

2.1 Product Perspective

The AI Lawyer is a standalone web application developed with Streamlit. It interacts with a cloud-hosted AI model (Google Gemini) and manages chat sessions and constitutional data files.

2.2 System Interfaces

- **File System:** Reads constitution text files from `/data`
- **SQLite (in-memory):** Tracks question statistics
- **EncryptedCookieManager:** Persists session IDs in user browser cookies
- **Google Gemini API:** Sends prompts for AI responses

2.3 User Interfaces

- Sidebar for chat selection and jurisdiction dropdown
- Main panel showing chat history and input box
- Buttons for New Chat, Delete Chat, and jurisdiction change

2.4 Assumptions and Dependencies

- Constitution files (`japan_constitution.txt`, `monaco_constitution.txt`) exist in `/data`
 - Valid Google Gemini API key configured
 - Streamlit server with cookie support
-

3. System Features and Requirements

3.1 Functional Requirements

1. **FR-1:** Initialize chat history and default jurisdiction
2. **FR-2:** Load constitutional text when jurisdiction changes
3. **FR-3:** Manage chat sessions (new, select, delete)
4. **FR-4:** Persist and retrieve session data to/from JSON files
5. **FR-5:** Send user queries and chat history to AI model and display responses
6. **FR-6:** Categorize questions into legal categories and update in-memory stats

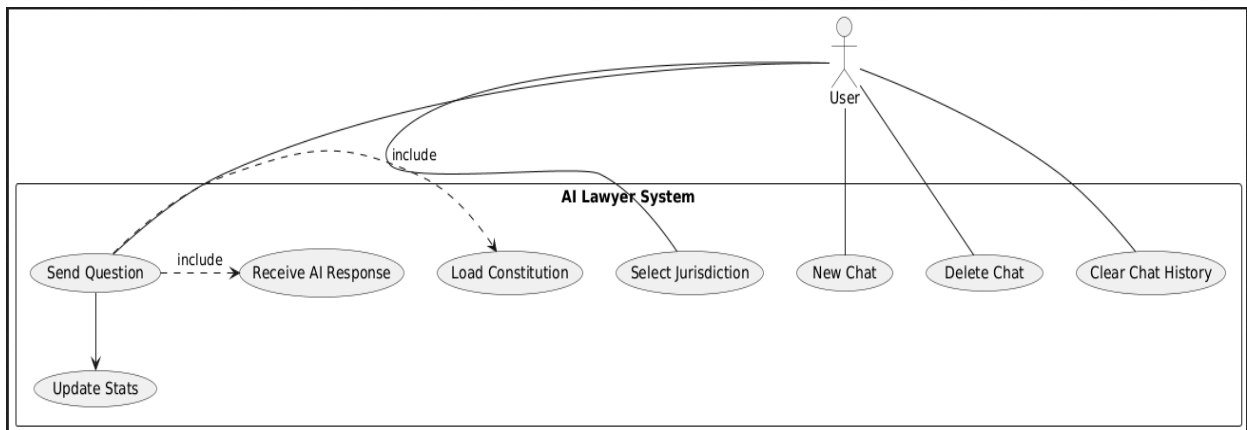
3.2 Non-Functional Requirements

- **NFR-1:** Response time for AI answers < 5 seconds under nominal load
- **NFR-2:** System availability 99.5% uptime
- **NFR-3:** Data confidentiality for cookies and session files
- **NFR-4:** Support concurrent sessions

4. Actors

- **User:** End user interacting via web UI
 - **System:** Streamlit application orchestrating AI calls and state
 - **AI Model:** External Google Gemini service
-

5. Use Case Diagram




6. Detailed Use Case Descriptions

6.1 Use Case: Select Jurisdiction


- **ID:** UC-01
- **Actor:** User
- **Precondition:** User is on the application; chat session is active
- **Trigger:** User selects a different country from the dropdown
- **Main Flow:**

1. System detects dropdown change event
 2. System clears existing chat history from session state
 3. System loads constitution text for selected country
 4. System inserts system prompts and loaded text into chat history
 5. UI refreshes to display updated chat context
- **Postcondition:** Chat history reset; new constitution loaded

6.2 Use Case: New Chat

- **ID:** UC-02
- **Actor:** User
- **Precondition:** Application loaded; optional existing chats
- **Trigger:** User clicks “ New Chat” button
- **Main Flow:**
 1. System creates a new chat entry with default country
 2. System sets the new chat as current
 3. System persists session data to JSON
 4. UI reruns to show blank chat in main panel
- **Postcondition:** New chat available in sidebar; active chat is empty

6.3 Use Case: Delete Chat

- **ID:** UC-03
- **Actor:** User
- **Precondition:** Multiple chats exist
- **Trigger:** User clicks “ Delete Chat” button
- **Main Flow:**

1. System removes selected chat entry from `session_state.chats`
 2. System updates JSON file storage
 3. System reruns and selects fallback chat
- **Postcondition:** Deleted chat removed; another chat becomes active

6.4 Use Case: Clear Chat History

- **ID:** UC-04
- **Actor:** System (implicit via jurisdiction or code logic)
- **Precondition:** Current chat has loaded constitution history
- **Trigger:** System detects jurisdiction mismatch or explicit clear
- **Main Flow:**
 1. System sets `session_state.messages` to empty list
 2. Returns confirmation message
- **Postcondition:** Chat history emptied; ready to load new context

6.5 Use Case: Send Question

- **ID:** UC-05
- **Actor:** User
- **Precondition:** Chat history initialized with constitution text
- **Trigger:** User types question and presses enter
- **Main Flow:**
 - System appends user message to history
 - System calls `get_ai_response_st()` with prompt and history
 - Inside, system compiles system instructions and filtered history

- System sends prompt to Google Gemini API
 - System receives AI-generated answer
 - System appends response to history and displays it
 - System categorizes question and updates in-memory stats
 - System persists updated session to JSON
 - **Alternate Flows:**
 - 5a. API error → System returns error message
 - **Postcondition:** User sees AI response; stats updated
-

7. Appendix

- **A. Constitution File Requirements**
 - Text files must be named `japan_constitution.txt` or `monaco_constitution.txt`
 - Stored in `/data` directory