**Multi-Agent Requirement Engineering System - Advanced Prompt Specification**

"Develop an intelligent agents powered by a Large Language Model (LLM) using Curser AI, integrated with Hugging Face's free, open-access models. The agent should be capable of understanding and performing tasks based on natural language prompts (NLP). The implementation should be done in Python and designed to automatically detect and load an external input file (containing all required data) from a user-specified path. This input file will serve as the agent's primary data source for contextual understanding, decision-making, or task execution. The overall system should be modular, easy to configure, and suitable for experimentation or integration with broader AI workflows."

**1. Agent 1: Requirement Intake & Pre-Processing and Drafting SYS.1**

**Primary Role**: Understand and extract initial requirements from raw customer communication and Derive SYS.1 requirements

**Functions & Features**:

* **Feasibility Check**: Automatically assess project feasibility using predefined business rules and domain knowledge.
* **Check SYS**.1 as per ISO/IEC/IEEE 29148:2018 or ISO 26262 or INCOSE SE Handbook
* **Multi-Source Ingestion**: Support inputs from:
  + Emails, meeting transcripts, voice-to-text (STT) data
  + PPTs, TXT, PDFs, DOCX, XLSX, CSV
  + Jira tickets, Confluence pages
  + At the same time multiple inputs can be taken
* **Output File Format**: **TXT, PDFs, DOCX, XLSX, CSV**
* **Semantic Parsing**: Use NLP to extract intents, constraints, and stakeholders.
* **Requirement Categorization**: Classify requirements into functional, non-functional, regulatory, etc.
* **Traceability Links**: Establish traceability to original source artifacts.
* **Confidence Scoring**: Assign confidence levels to each extracted requirement.
* **Duplicate Detection**: Automatically flag similar or repeated requirements.
* **Integration with External Tools**: APIs for Slack, Outlook, MS Teams, Notion.

**2. Agent 2: Requirement Drafting (SYS.2) & Structuring**

**Primary Role**: Generate structured draft requirements from parsed data of SYS.1 or manually selection in different formats (**TXT, PDFs, DOCX, XLSX, CSV**).

**Functions & Features**:

* **Template-Based Drafting**: Use configurable templates compliant with industry standards (IREB, IEEE 830).
* **Auto-Fill Attributes**: Populate requirement fields (ID, priority, rationale, etc.) automatically.
* **Requirement Rewriting**: Improve language clarity and remove ambiguity.
* **Dependency Mapping**: Identify logical dependencies and impacts across requirements.
* **Output File Format**: **TXT, PDFs, DOCX, XLSX, CSV**
* **Modularization**: Break complex requirements into manageable sub-requirements.
* **Custom Rule Engine**: Plug in domain-specific rules to refine drafting logic.
* **Feedback Loop from Agent 3**: Accept dynamic suggestions for improvement from Agent 3.
* **Suggest improvements** for any requirements that do not meet IREB standards and add button to update in front of that requirement to take the suggestion.
* **Classify each requirement** (e.g., Functional, Non-functional, Constraint, Assumption).
* Add verification method (either Sys.4 or Sys.5 and verification criteria
* Priority Assignment and Release Planning

Allow users to assign a priority level to each requirement and plan its implementation across different software releases.

**Details**:

* **Priority Assignment**:
  + Users can assign a **Priority** to each requirement using a dropdown menu.
  + Available priority levels:
    - **High**
    - **Medium**
    - **Low**
  + The priority reflects the **importance** and **impact** of the requirement.
* **Implementation Planning**:
  + Include an **Implementation Priority** dropdown menu for each requirement.
  + Available options:
    - **Release 1**
    - **Release 2**
    - **Release 3**
  + Users should be able to **select the intended release** for each requirement individually.

**User Interaction**:

* UI components (e.g., dropdowns) should be intuitive and allow **easy selection and editing** of both priority and release phase.

**3. Agent 3: System Requirement (SYS.2) Review, Compliance & Continuous Learning**

**Primary Role**: Evaluate, refine, and standardize requirements.

**Functions & Features**:

* **Check each requirement (SYS.2) against IREB (International Requirements Engineering Board), Check SYS**.2 as per ISO/IEC/IEEE 29148:2018 or ISO 26262 or INCOSE SE Handbook
* **IREB standards** for:
  + Clarity
  + Unambiguity
  + Completeness
  + Consistency
  + Verifiability
  + Modifiability
  + Traceability
* **Linguistic Analysis**: Highlight passive voice, vague terms, or conflicting statements.
* **Comment & Suggest Mode**: Provide detailed annotations and improvement suggestions.
* **App-Based Validation**: Accept/reject suggestions in real-time via web/app UI.
* **Self-Learning System**: Learn from accepted suggestions to improve future draft quality.
* **Version Control Integration**: Track changes, compare versions, and rollback if needed.
* **Automated Test Case Derivation**: Propose test cases for each functional requirement.
* Evaluates requirement clarity and completeness
* Detects ambiguous language
* Provides IREB-based improvement suggestions
* Assigns priority levels automatically

**4. Agent 4: Write System Testcases (SYS.5) from SYS.2**

**Primary Role**: Write the system testcases

Agent 4 should automatically retrieve the "sys2\_requirements\_reviewed.xlsx" file from the following path:  
**D:\AgentX\AutoTestGen\_MAPS\_Agents123\AutoTestGen\_MAPS\Inputs**.

Additionally, there should be an option to manually upload the SYS.2 file. Once the file is obtained—either automatically or manually—Agent 4 should automatically convert the SYS.2 requirements into SYS.5 test cases.

* **Generate system validation test cases** for each requirement, including:
  + SYS.2 Req. ID(which is uploaded using xlsx file)
  + SYS.2 System Requirement (which is uploaded using xlsx file)
  + Test Case ID
  + Description
  + Preconditions
  + Test Steps
  + Expected Results
  + Pass/Fail Criteria
  + Priority

**Functions & Features**:

* **Manual Edit Detection**: Detect and log manual overrides to track intent drift.
* **Auto-Correction Engine**: Improve prompt structure post-acceptance.
* **Requirement Maturity Staging**: Track lifecycle stages (Draft → Reviewed → Final → Deployed).
* **Impact Analysis**: Identify changes that affect upstream or downstream systems.
* **Custom Alerts**: Notify stakeholders on critical changes or conflict detection.
* **Link to Development Tools**: Auto-sync finalized requirements with tools like Jira, Azure DevOps, DOORS, Polarion.
* **Change Propagation**: Propagate approved changes to related requirements and regenerate impact documents.
* **Generate system validation test cases** for each requirement, including:
  + Test Case ID
  + Description
  + Preconditions
  + Test Steps
  + Expected Results
  + Pass/Fail Criteria
  + Priority
* Ensure SYS.2 to SYS.5 (**requirements-to-test-case traceability)** by mapping each test case to its corresponding requirement., Traceability Matrix: Mapping between requirement IDs and test case IDs
* Converts natural language requirements into structured test cases
* Generates logical and concrete test scenarios
* Creates detailed test steps and expected results
* Supports automotive and general software testing scenarios
* **Test Case Generation and Correction Workflow**

Ensure test cases are generated for each prompt or requirement. Allow manual corrections for any discrepancies (deltas), and implement autocorrection based on data or feedback post-review.

**Details**:

* **Test Case Generation**:
  + System should **automatically generate test cases** for each prompt or requirement.
  + If **test case generation is missing**, the system should flag it for review.
* **Delta Handling**:
  + **Delta** refers to the difference between expected and actual test case behavior/content.
  + System should detect deltas and **highlight them for manual inspection**.
* **Manual Correction**:
  + Users must be able to **manually correct test cases** when discrepancies are detected.
  + Interface should allow **inline editing or structured correction forms**.
* **Autocorrection (Post-Review)**:
  + After manual correction or feedback, the system should:
    - Learn from the inputs.
    - Apply **autocorrection** for similar future cases.
    - Optionally reprocess related test cases based on updated data.

**User Interaction**:

* Visual indicators for missing or outdated test cases.
* Editable fields or review mode for manual corrections.
* Review and feedback flow to trigger autocorrection.

**All Agents work to Together ->**

need to implement "All Agents" as well it should have feature of all agent means I'll jsut upload the customer requirement and tool should generate-> Sys.1 -> Sys.2 drafting -> Sys.2 Review -> Sys.5 Drafting

**Global Features Across All Agents**

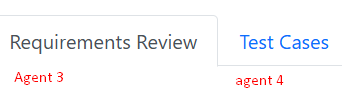
* **Audit Trail**: Maintain full traceability of who/what/when changes were made.
* **RBAC (Role-Based Access Control)**: Define permissions for different stakeholders (e.g., reviewer, author, customer).
* **Collaboration Mode**: Enable real-time editing, commenting, and review among cross-functional teams.
* **Analytics Dashboard**: KPIs for requirement quality, turnaround time, coverage, change frequency.
* **Multilingual Support**: Extract and draft requirements in multiple languages with translation and localization.
* **Integration with ML Models**: Optional integration for classification, anomaly detection, and summarization.
* **Input File Format: TXT, PDFs, DOCX, XLSX, CSV**
* Output File Format: **Provide a button to download in .xlsx,.pdf,.docx,.csv formats with all the information.**
* **Add New Section called “Dashboard” for all agents separately (Pie chart) and all details**
* **Redundancy Check (Similarity Index Sheet) Across All Agents and Individually**
  + Implement a redundancy detection system using a **Similarity Index Sheet** to identify overlapping or duplicate entries **across all agents combined** and also **within each agent individually**.

**Details**:

* **Scope of Redundancy Check**:
  + **Cross-Agent Check**:  
    Detect redundancies between system requirements, prompts, or test cases **across all agents** in the system.
    - E.g., similarities between Agent 1’s and Agent 4’s entries.
  + **Agent-Specific Check**:  
    Detect redundancies **within each agent separately**, ensuring that entries inside a single agent don’t duplicate.
* **Similarity Index Sheet**:
  + For both checks, generate a **Similarity Index Sheet** with:
    - IDs or titles of compared items.
    - Agent source of each item.
    - Similarity score (numeric percentage or score).
    - Suggested action (e.g., review, merge, ignore).
    - Reviewer comments or notes.
  + The sheet should allow **filtering** or **grouping by agent** or **cross-agent pairs**.
* **Algorithms & Techniques**:
  + Use appropriate NLP or string similarity algorithms for accurate detection.
  + Consider semantic similarity to avoid false negatives.
* **User Interface**:
  + A unified view showing:
    - All redundancy pairs across agents.
    - Ability to toggle views between “All Agents” and “Single Agent.”
  + Inline actions to:
    - Merge duplicates.
    - Flag for review.
    - Add comments.
  + History tracking for all actions taken.
* **Batch Processing & Scheduling**:
  + Should allow scheduled runs for redundancy checks.
  + Users can trigger manual re-checks for specific agents or the entire system
* **Agent Selection Dropdown Menu on Landing Page**

Add a **dropdown menu** on the landing page that allows users to select and activate a specific agent or choose to work with **all agents simultaneously**.

**Details**:

* **Dropdown Menu Options on the landing page at top right hand side**:
  + Agent 1: Customer requirements to SYS.1
  + Agent 2: SYS.2 Drafting
  + Agent 3: SYS.2 Review (Is present as of now name it as Agent 3)
  + Agent 4: SYS.2 to SYS.5 (Is present as of now name it as Agent 4)
  + **All Agents**
  + 
* **Behavior**:
  + On selection:
    - The chosen agent is **activated** for interaction (prompt handling, test case generation, etc.).
    - If **All Agents** is selected:
      * System should enable **cross-agent processing**, such as:
        + Collective prompt review
        + Combined test case generation
        + Redundancy checks across all agents
  + Persist user selection across sessions (optional).
* **UI/UX Considerations**:
  + Dropdown should be placed **prominently on the landing page**.
  + Include clear labels like: **"Select Agent"**.
  + Optionally, add agent descriptions as tooltips or subtitles in the dropdown (e.g., Agent 4 – Advanced Prompt Manager).
* **Technical Considerations**:
  + Trigger a state/context update based on the selected agent.
  + Ensure only relevant data, actions, and UI components for the selected agent(s) are displayed.
* **File Upload, Parsing, and Tabular Display in Browser**

Provide an interface in the browser to **upload and parse various file formats** specific to each agent's purpose. The parsed data should be **displayed in a structured tabular format** for review and further processing.

### ****Common Functional Requirements****

* **File Upload Interface**:
  + Allow users to **upload files or folders** directly via a browser-based UI.
  + Support **drag-and-drop** or manual file selection.
  + Include basic file validation and status indicators.
* **Supported File Types**:
  + .txt, .pdf, .docx, .xlsx, .csv
  + For Agent 1 only: Include support for **emails (.eml, .msg)**, **meeting transcripts**, **voice-to-text (STT) outputs**, and **Jira/Confluence exports**.
* **Parsing**:
  + Use appropriate parsers for each file type:
    - PDF/Text extraction tools
    - DOCX/XLSX readers (e.g., python-docx, openpyxl)
    - CSV parsers
    - Email and transcript processing tools
  + Convert parsed data into **structured rows/columns**.
* **Display Format**:
  + Show results in a **sortable, filterable table** in the browser.
  + Table should include:
    - Source File Name
    - Detected Content Type (e.g., requirement, note, ticket)
    - Extracted Text or Summary
    - Agent-specific Metadata (e.g., requirement ID, customer tag, etc.)
* **Browser UI Requirements**:
  + Tab-based or dropdown-based **agent selector**
  + Upload panel for each agent
  + Separate tab or section for **displaying parsed results**

### ****Agent-Specific Logic****

|  |  |  |
| --- | --- | --- |
| **Agent** | **File Input Types** | **Data Focus** |
| **Agent 1** | Emails, meeting transcripts, STT data, TXT, PDFs, DOCX, XLSX, CSV, Jira tickets, Confluence pages | Customer requirements (raw inputs) |
| **Agent 2** | SYS.1 in TXT, PDFs, DOCX, XLSX, CSV | SYS.2 Drafting |
| **Agent 3** | SYS.2 in TXT, PDFs, DOCX, XLSX, CSV | SYS.2 Review |
| **Agent 4** | SYS.2 to SYS.5 in TXT, PDFs, DOCX, XLSX, CSV | Full-chain processing from SYS.2–SYS.5 |

### ****Future Expansion Roadmap****

1. **Tool Integration**
   * Integrate with industry-standard tools such as:
     + **Jira** – for project and issue tracking.
     + **DOORS** – for requirements management.
     + **Polarion** – for end-to-end ALM (Application Lifecycle Management).
2. **Automated Diagram Generation**
   * Implement automatic **UML** and **flowchart** generation directly from structured or semi-structured requirements.
3. **Unit Test Generation**
   * Auto-generate **unit test cases** based on **functional requirements**.
   * Enable seamless linkage with **CI/CD pipelines** for automated validation.
4. **Multilingual Support**
   * Provide support for multiple languages to enhance collaboration across **global and diverse teams**.
5. **Agent 5: SWE.1 – Writing Phase**
   * Develop an intelligent agent capable of **authoring requirements** according to **SWE.1** of the Automotive SPICE model.
6. **Agent 6: SWE.1 Review – IREB/CPRE Aligned**
   * Introduce an AI agent for **reviewing requirements**, ensuring compliance with **IREB**/**CPRE** standards and best practices.
7. **Agent 7: SWE.5/SWE.6 – Test Case Generation**
   * Automatically generate **test cases** and **verification criteria** from refined requirements in alignment with **SWE.5** (Design) and **SWE.6** (Implementation).

## 🧠 Multi-Agent Requirement Engineering System – Refined Prompt Specification

### 🔧 Goal:

Develop a modular, intelligent, multi-agent system using **Large Language Models (LLMs)** powered by **Curser AI** and integrated with **Hugging Face’s open-access models**. The system should support **natural language understanding and task execution** for the full lifecycle of requirements engineering — from raw intake to test case generation.

* **Implementation**: Python
* **LLM Backend**: Curser AI with Hugging Face open models
* **Input**: External file auto-loaded from a user-specified path
* **Modular Architecture**: Each agent handles a specific responsibility
* **Target Use Case**: Rapid prototyping, AI-assisted engineering, and integration into broader AI/DevOps pipelines

### 🧩 Agent 1 – Requirement Intake, Pre-Processing, and SYS.1 Drafting

**Role**: Extract initial requirements from diverse raw formats and draft structured SYS.1 requirements.

#### Features:

* **Feasibility Analysis**: Leverage domain rules and heuristics
* **Multi-Source Ingestion**:
  + Emails (.eml/.msg), meeting transcripts, STT data
  + Documents: TXT, DOCX, PDF, XLSX, CSV
  + Jira, Confluence exports
* **Parsing & NLP**:
  + Intent recognition
  + Constraint detection
  + Stakeholder identification
* **Categorization**: Functional, non-functional, Information
* **Traceability**: Create source-to-SYS.1 mapping
* **Confidence Scores** for each extracted requirement
* **Duplicate Detection** (via similarity algorithms)
* **API Integrations**: Slack, Outlook, MS Teams, Notion
* **Export Formats**: TXT, PDF, DOCX, XLSX, CSV

### 🧩 Agent 2 – SYS.2 Requirement Drafting & Structuring

**Role**: Transform SYS.1 data into well-structured, standardized SYS.2 requirement sets. Agent 2 should automatically take the “sys1\_requirements\_only.xlsx” file from the path “C:\Users\Ranjit Jagtap\Desktop\AutoTestGen\_Project\AutoTestGen\_MAPS\Inputs” also there should be provision of manually uploading Sys.1 file. Agent 2 should have the same structure as Agent 1 like placements and tables.

#### Features:

* **Template-Based Generation**: IREB, IEEE 830-compliant
* **Auto-Fill Metadata**: ID, priority, rationale, type
* **Improved Clarity & Rewriting Engine**
* **Dependency Mapping**
* **Modularization** of complex requirements
* **Custom Rule Engine** (domain-specific logic)
* **Classification**: Functional, non-functional, assumption, constraint
* **Verification Mapping**: Link to Sys.4/Sys.5
* **Feedback from Agent 3**: Review integration
* **User Actions**:
  + Suggest improvement button (for non-compliant requirements)
  + Dropdowns for:
    - Priority (High, Medium, Low)
    - Release Planning (Release 1/2/3)
* **Output**: Editable, downloadable formats (TXT, DOCX, XLSX, CSV, PDF)

### 🧩 Agent 3 – SYS.2 Review, Compliance & Continuous Learning

**Role**: Analyze and enhance requirements for clarity, compliance, and traceability as per the as per ISO/IEC/IEEE 29148:2018 or ISO 26262 or INCOSE SE Handbook

Agent 3 should automatically take the “sys2\_requirements.xlsx” file from the path “C:\Users\Ranjit Jagtap\Desktop\AutoTestGen\_Project\AutoTestGen\_MAPS\Inputs” also there should be provision of manually uploading SYS.2 file.

#### Features:

* **IREB Compliance Check**:
  + Clarity, completeness, consistency, verifiability, modifiability
* **Linguistic Analysis**:
  + Detect passive voice, ambiguity, and contradictions
* **Interactive Suggest/Comment Mode**
* **App-Based Validation**:
  + Accept/reject suggestions via web UI
* **Self-Learning Engine**:
  + Adapts based on user actions
* **Version Control** Integration
* **Automated Test Case Proposals** (for functional requirements)
* **Auto Priority Assignment**

### 🧩 Agent 4 – SYS.2 to SYS.5: System Test Case Generator

**Role**: Automatically convert SYS.2 requirements into SYS.5 test cases with traceability.

Agent 4 should automatically retrieve the "sys2\_requirements\_reviewed.xlsx" file from the following path:  
**D:\AgentX\AutoTestGen\_MAPS\_Agents123\AutoTestGen\_MAPS\Inputs**.

Additionally, there should be an option to manually upload the SYS.2 file. Once the file is obtained—either automatically or manually—Agent 4 should automatically convert the SYS.2 requirements into SYS.5 test cases.

#### Features:

* **Traceability and Dashboard with Pie chart like agent 3, and go to traceability dashboard should be linked to these charts when clicked on the button it should scroll down to the charts**:
  + SYS.2 ↔ SYS.5 Mapping
* **Structured Test Case Generation**:
  + Test ID, description, steps, expected results, preconditions, criteria
* **Requirement Maturity Tracking** (Draft → Final → Deployed)
* **Delta Detection & Manual Correction Support**
* **Autocorrection Engine** (based on feedback loop)
* **Change Impact Analysis**
* **DevOps Integration**: Jira, Azure DevOps, DOORS, Polarion
* **UI Enhancements**:
  + Inline editing
  + Visual flags for outdated/missing test cases

### 🌐 Global System-Wide Features

#### 🌍 File & Data Handling:

* **Input/Output Formats**: TXT, DOCX, PDF, XLSX, CSV
* **Browser-Based UI**:
  + Drag-and-drop file upload
  + Parsing into structured, editable tables
  + Tab-based display per agent
* **Agent-Specific Data Focus**:
  + Agent 1: Raw inputs
  + Agent 2: SYS.1 → SYS.2
  + Agent 3: SYS.2 Review
  + Agent 4: SYS.2 → SYS.5

#### 📊 Analytics & Visualization:

* **Dashboard for Each Agent**:
  + Pie charts and data summaries
* **Audit Trail**: Full edit history with user/action timestamps
* **RBAC**: Reviewer, author, customer access control
* **Real-Time Collaboration**: Comments, reviews, edits
* **Multilingual Support**: Translation/localization for international projects
* **ML Integrations**:
  + Requirement classification
  + Anomaly detection
  + Text summarization

### 🧮 Redundancy Detection System

#### Features:

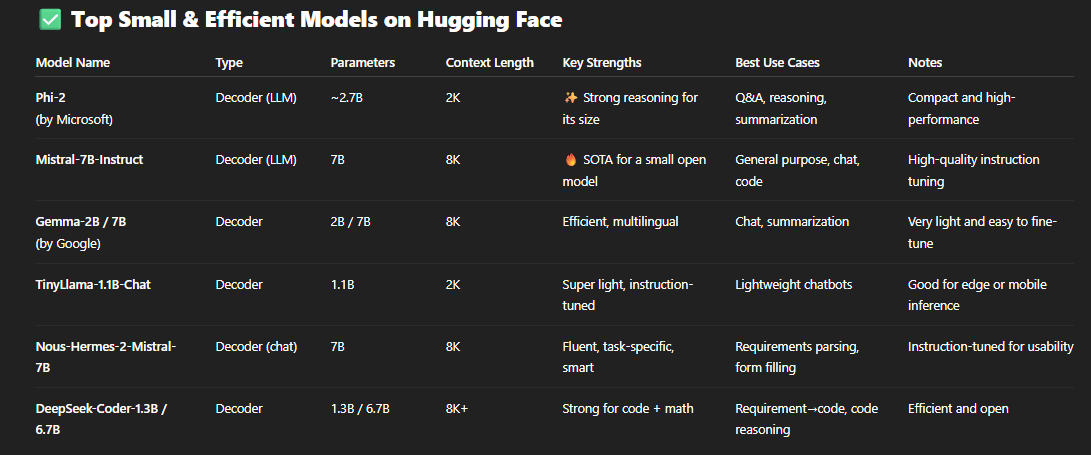
* **Similarity Index Sheet**:
  + Compare requirements/test cases across and within agents
  + Includes:
    - Similarity scores
    - Duplicate IDs
    - Recommended action (merge/review/ignore)
    - Reviewer notes
* **Toggle View**: Switch between single-agent and cross-agent views
* **Inline Actions**: Merge, flag, annotate
* **Scheduled & On-Demand Processing**
* **NLP & Semantic Matching Algorithms**

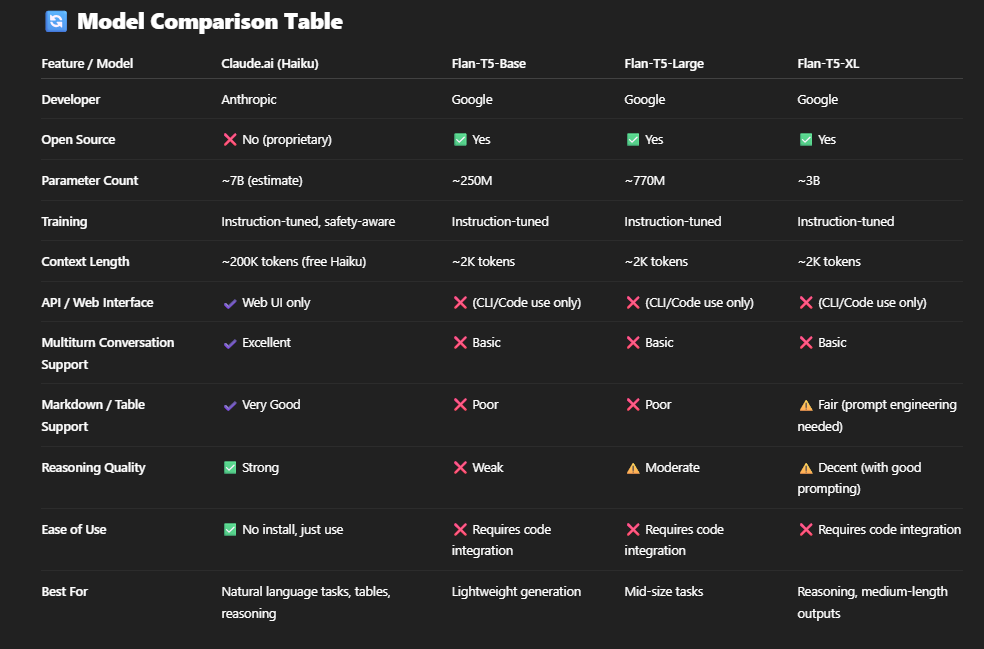
### 🧭 Agent Selector Interface (Landing Page)

* **Dropdown Menu**:
  + Agent 1: Intake
  + Agent 2: SYS.2 Drafting
  + Agent 3: SYS.2 Review
  + Agent 4: SYS.2 → SYS.5
  + All Agents
* **Behavior**:
  + Dynamically load agent-specific UI/data/actions
  + Support persistent user selection across sessions

### 🔍 Suggested Improvements (Optional/Advanced)

* **Error Handling & Logging**:
  + Graceful fallback if file types are invalid or parsing fails
* **Plugin System**:
  + Allow future agents/tools to be plugged in modularly
* **API Access Layer**:
  + Expose core agent actions as REST APIs
* **Mobile Compatibility**:
  + Allow mobile access to dashboards, feedback, and file uploads





**Similar to Agent 1, I need Traceability & dashboard for Agent 2 and agent 4**

**For Agent 2,**

**From SYS.1 to SYS.2**

**For Agent 4,**

**From SYS.2 to SYS.5**