### **CoXPRT Application: Comprehensive Function & API Documentation**

This document provides a detailed, function-by-function breakdown of the entire CoXPRT codebase. It is organized by module to reflect the application's architecture.

### **1. Main Application (app.py)**

This is the main entry point and controller for the Streamlit application. It manages the overall UI structure, navigation, and state flow between the different tabs.

| **Function Name** | **Input Parameters** | **Output** | **Purpose & Key Interactions** |
| --- | --- | --- | --- |
| app.py (script) | None | Renders Streamlit UI | **Main Application Controller.** Initializes the app, sets up logging and session state, and renders the tab-based navigation. It acts as the central orchestrator. |
| setup\_logging | None | Logger object | Configures a session-specific logger, creating a new log file for each user session to aid in debugging. |
| navigate\_to\_tab | target\_tab\_index: int | None | **Core Navigation Logic.** Handles all tab-switching logic. It calls the appropriate validation function for the current tab before allowing forward navigation and triggers confirmation popups (show\_lock\_confirmation\_popup) to lock a tab's state. |
| is\_tab\_accessible | tab\_index: int | bool | A validation helper that checks if a user is allowed to navigate to a specific tab based on whether preceding tabs have their mandatory data filled out. |
| handle\_validation\_popups | None | Renders UI | Checks the session state for any pending validation errors and calls show\_validation\_popup to display the appropriate error modal to the user. |
| refresh\_all\_data | None | None | A utility function tied to the "Refresh" button that clears the entire st.session\_state to reset the application to its initial state. |

### **2. Data Collection Tabs (Client/ & Seller/)**

These modules handle the UI and state for the "Client Information" and "Seller Information" tabs. Their structures are nearly identical.

| **Module / File** | **Function Name** | **Input Parameters** | **Output** | **Purpose & Key Interactions** |
| --- | --- | --- | --- | --- |
| client.py | client\_tab | st, logger, is\_locked: bool | ClientTabState object | **Client Data Entry UI.** Renders all UI components for the "Client Information" tab and orchestrates calls to backend services. Returns the final state object for this tab. |
| seller.py | seller\_tab | is\_locked: bool | SellerTabState object | **Seller Data Entry UI.** Renders all UI components for the "Seller Information" tab. Returns the final state object for this tab. |
| client\_dataclass.py | ClientTabState | Dataclass fields | ClientTabState instance | **Client State Model.** A dataclass that defines the data schema for the client tab. Includes methods for saving to (to\_session\_state) and loading from (from\_session\_state) Streamlit's session state. |
| seller.py | SellerTabState | Dataclass fields | SellerTabState instance | **Seller State Model.** A dataclass defining the data schema for the seller tab, with methods for session state persistence. |
| client.py | render\_client\_name\_section | logger, client\_data, is\_locked | str | Renders the UI for entering the client's name and the "Find Website" button, which triggers get\_urls\_list. |
| client.py | render\_spoc\_name\_section | logger, client\_data, is\_locked | str | Renders the UI for entering the SPOC's name and triggers the LinkedIn search agent (get\_linkedin). |
| client.py | doc\_upload\_section | logger, client\_data, is\_locked | None | Renders the file uploader for RFI documents and the "Get pain points" button, which triggers the RAG pipeline (get\_pain\_points). |
| seller.py | \_process\_all\_documents | seller\_state, seller\_documents\_upload | None | Orchestrates the analysis of uploaded seller documents. It iterates through files, saves them, and calls get\_services to extract capabilities. |

### **3. Backend Services & AI Agents**

These modules contain the core logic for data processing, web scraping, and AI-powered generation.

| **Module / File** | **Function Name** | **Input Parameters** | **Output** | **Purpose & Key Interactions** |
| --- | --- | --- | --- | --- |
| **common\_utils.py** | get\_scraped\_data | url: str, scrape\_technique: str | User object or str | **Web Scraping Service.** Scrapes a website for company details using either BeautifulSoup or a crawl4ai agent. |
| **common\_utils.py** | get\_urls\_list | company\_name: str | List[str] | **Website Search Agent.** A synchronous wrapper that runs the asynchronous get\_urls agent to find official company websites. |
| **Search/Linkedin/linkedin\_agent\_runner.py** | get\_linkedin | user\_name: str | dict or None | **LinkedIn Search Agent.** A synchronous wrapper that runs the get\_linkedin\_p agent to find LinkedIn profiles and infer job priorities. |
| **doc\_vectorizer.py** | vectorize | filepath: str, company\_name: str | Chroma VectorStore | **Document Ingestion Pipeline (RAG).** The main entry point for document processing. It routes files to the correct content extractor, chunks the text, creates vector embeddings, and stores them in a Chroma database. |
| **doc\_vectorizer.py** | image\_summarize | model, base64\_image: str, prompt: str | str | **Multimodal AI Service.** Uses the Gemini vision model to generate a textual summary of an image's content. |
| **recommendation\_utils.py** | get\_ai\_proj\_sepc\_recommendations | prompts: str, client\_data, seller\_data | dict | **AI Suggestion Service.** Generates structured AI suggestions for project specifications (scope, timeline, etc.) based on client and seller context. |
| **SalesProposalWriting\_Agent/** | get\_presentation | client, seller, project\_specs | (str, str, str) | **Core Proposal Generation Entry Point.** Orchestrates the final document creation by calling the HTML generation and PDF conversion functions. |
| **SalesProposalWriting\_Agent/** | generate\_modern\_presentation | filename: str, logo\_url: str, theme details | (str, str) | **HTML Templating Engine.** Reads AI-generated text content and injects it into a sophisticated, dynamically styled HTML template. |
| **SalesProposalWriting\_Agent/** | generate\_pdf\_from\_html | html\_content: str, output\_dir: str | str (file path) | **PDF Conversion Service.** Converts the final HTML string into a PDF file using WeasyPrint with a fallback to pdfkit. |

### **4. Project Specification & Proposal Generation Tabs**

These modules handle the final stages of the user workflow.

| **Module / File** | **Function Name** | **Input Parameters** | **Output** | **Purpose & Key Interactions** |
| --- | --- | --- | --- | --- |
| **project\_spec.py** | proj\_specification\_tab | client\_data, seller\_data, is\_locked: bool | list | **Project Specification UI.** Renders the UI for the third tab. It calls load\_all\_data to pre-fetch AI suggestions and uses render\_two\_column\_selector to build the UI. |
| **project\_spec.py** | load\_all\_data | client\_data, seller\_data | None | **AI Recommendation Orchestrator.** Runs once to pre-fetch all AI suggestions for the project specification sections by calling get\_ai\_proj\_sepc\_recommendations multiple times. |
| **proposal\_generator.py** | generate\_tab | client\_data, seller\_data, additional\_specs | Renders Streamlit UI | **Proposal Generation UI.** Renders the final tab, which calls render\_preview\_tab to manage the generation process and display the download button. |
| **proposal\_generator.py** | render\_preview\_tab | client\_data, seller\_data, project\_specs | None | Manages the user-facing generation process, displaying a progress bar and status updates while calling the backend get\_presentation function. |