**SyntaxMatrix UI Framework Documentation**

SyntaxMatrix UI is a lightweight and flexible framework designed to simplify the creation of interactive user interfaces for your AI applications. It enables you to build modern, chat-based interfaces and dashboards quickly while abstracting away the complexities of the underlying implementation. With built‐in support for customisable themes, multiple UI layouts, and session management, SyntaxMatrix UI lets you focus on your application logic and integrations with external services.

**Key Features**

* **Rapid UI Assembly:** Easily register input widgets and interactive buttons with minimal code.
* **Customizable Themes:** Switch between light and dark modes (or provide your own custom theme) to match your brand or user preferences.
* **Multiple UI Modes:** Choose from several pre-defined presentation styles—such as "default," "bubble," "card," or "smx"—to suit different application needs.
* **Session and Chat Management:** Built-in functions allow you to maintain, update, and clear chat history and session data, ideal for conversational interfaces.
* **Widget Abstraction:** Define common UI elements like text inputs, buttons, and file uploaders without dealing with low-level UI code.
* **Easy Integration:** Seamlessly integrate with external APIs (e.g., language models, data services) to process user queries and display responses dynamically.

**Getting Started**

**Installation and Import**

To integrate SyntaxMatrix UI into your project, include the framework in your code as follows:

import syntaxmatrix as smx

This single import provides you with access to all the public API functions needed to build your application interface.

**Configuration**

Before building your UI, optionally customise the look and feel of your application by setting key properties. For example (ensure the icons a svg types, like emojis, https://getemoji.com:

# Set your site and project branding.

smx.set\_site\_title("title")

smx.set\_site\_logo("logo")

smx.set\_site\_icon("𓃑")

smx.set\_user\_icon("😮")

smx.set\_bot\_icon("🤖")

smx.set\_project\_title("SMX UI")

# Enable theme toggling (e.g., light/dark mode) and select a UI mode.

smx.enable\_theme\_toggle()

smx.set\_ui\_mode("default")

These configuration functions let you define the icons, titles, and overall aesthetic without exposing any implementation specifics.

**Widget Registration**

SyntaxMatrix UI provides simple functions to add interactive widgets to your application. The most common widgets include:

* **Text Input:**  
  Use the text input widget to capture user queries or input data.

smx.text\_input("user\_query", "Enter query:")

* Buttons:  
  Register buttons with optional callback functions to handle events such as submitting a query or clearing the chat.

smx.button("submit\_query", "Submit", callback=create\_conversation)

smx.button("clear\_chat", "Clear Chat", callback=clear\_chat)

* **File Uploader (Optional):**  
  If you need file input capabilities, register a file uploader widget.

smx.file\_uploader("upload\_file", "Upload File")

* **Retrieve Chat History:**

chat\_history = smx.get\_chat\_history()

* **Update Chat History:**  
  After processing a query, append the user query and the generated response to the chat history and update the session.

chat\_history.append(("User", query))

chat\_history.append(("Bot", answer))

smx.set\_chat\_history(chat\_history)

* **Clear Chat History:**  
  To reset the conversation:

smx.clear\_chat\_history()

**Creating Custom Callbacks**

Define functions that perform specific actions (e.g., processing a query or clearing the conversation). For example, a conversation callback might look like this:

def create\_conversation():

# Retrieve the user input from the registered text input.

query = smx.get\_text\_input\_value("user\_query")

# Get the current chat history.

chat\_history = smx.get\_chat\_history()

# Process the query using an external API integration.

answer = process\_query(query, chat\_history, retrieved\_chunks)

# Update the chat history with the new messages.

chat\_history.append(("User", query))

chat\_history.append(("Bot", answer))

smx.set\_chat\_history(chat\_history)

# Clear the text input after processing.

smx.clear\_text\_input\_value("user\_query")

Similarly, you can define other callback functions (e.g., to clear the chat) and assign them to button widgets.

**Running Your Application**

if \_\_name\_\_ == "\_\_main\_\_":

smx.run()

**Advanced Customizations**

While the default configuration and widgets cover most common use cases, SyntaxMatrix UI also allows further customization:

* **Layout Customization:**  
  Use helper functions (such as columns) to arrange components side by side.
* **Dynamic Widget Rendering:**  
  Update or add widgets dynamically based on user interactions or data state changes.
* **Session Management Enhancements:**  
  The framework manages multiple chat sessions with simple APIs, allowing users to switch between ongoing conversations seamlessly.

**Upcoming Features**

The roadmap for SyntaxMatrix UI includes several exciting enhancements that will make it even more competitive compared to contemporary frameworks:

1. **Advanced State Management:**
   * Reactive state updates and enhanced session management to ensure that UI components update seamlessly in real time.
2. **Dynamic Component Rendering:**
   * Automatic re-rendering of UI components based on changes in application state for a more responsive user experience.
3. **Expanded Widget Library:**
   * Support for additional interactive elements such as sliders, dropdowns, checkboxes, and multi-select options—features similar to what modern frameworks like Streamlit provide.
4. **Custom Layouts and Responsive Design:**
   * Improved tools for creating complex and responsive layouts, ensuring your applications look great on all devices.
5. **Integration with Data Visualization Tools:**
   * Built-in support for charts, graphs, tables, and other visual data elements to enhance analytics and reporting capabilities.
6. **Real-Time Collaboration:**
   * Features to support simultaneous multi-user interactions, enabling collaborative workflows in your applications.
7. **Enhanced Theming and Customization:**
   * More granular control over UI aesthetics, including advanced theming options, to allow deep customization without compromising performance.
8. **Performance and Scalability Optimizations:**
   * Improved rendering speed and scalability enhancements to handle high-volume data and user interactions.
9. **Interactive Debugging and Logging:**
   * Integrated debugging tools and real-time logging to monitor UI states and troubleshoot issues more effectively.
10. **Plug-and-Play Modular Components:**
    * A modular architecture that allows developers to easily add, remove, or replace UI components to tailor the framework to specific project needs.
11. **Enhanced Security Measures:**
    * Robust session and data management features to ensure that user interactions and sensitive data are securely handled.

These planned features aim to not only match but exceed the current capabilities of existing solutions, offering a more robust, flexible, and developer-friendly experience.

**Conclusion**

SyntaxMatrix UI offers a streamlined way to build interactive, modern web applications with a focus on simplicity and flexibility. By abstracting the complexities of the UI layer, it empowers developers to concentrate on core application logic and integrations. With its robust feature set and an ambitious roadmap of upcoming enhancements, SyntaxMatrix UI is poised to become a leading tool for creating dynamic and engaging user interfaces.

Feel free to experiment with the various configuration options and widget types to tailor the framework to your needs. As new features are released, the framework will continue to evolve, bringing even more power and flexibility to your application development process.