**AI Legal Assistant Platform**

**Project Title:**

**AI-Powered Legal Assistant Suite**

**Team Members:**

* Sri Harsha
* Ramtej
* Rathna Sai
* Surya
* Ravi Varma

**Objective:**

To build a Streamlit-based AI-driven web application that assists users with various legal services, including:

1. **AI Legal Assistant Chatbot**
2. **Judgment Predictor**
3. **Legal Document Generator**

**Technologies Used:**

* **Python**
* **Streamlit** (for frontend and page navigation)
* **Natural Language Processing (NLP)** (assumed for chatbot and document understanding)
* **Machine Learning** (assumed for judgment prediction)
* **File Handling & Dynamic Execution** (for loading specific scripts based on user selection)

**Project Structure Overview:**

* **Main Controller Script (main.py)**:
  + Handles the page layout and sidebar navigation.
  + Uses selectbox for switching between different application modules.
  + Dynamically loads and executes corresponding module scripts using exec().
* **Pages:**
  + "AI Legal Assistant" → views/chatbotLegalv2.py
  + "Judgment Predictor" → views/judgmentPred.py
  + "Legal Doc Generator" → views/docGen.py

**Modules Description:**

**1. AI Legal Assistant (chatbotLegalv2.py)**

* Purpose: To assist users by answering basic legal questions using a conversational AI.
* Likely powered by a language model trained or fine-tuned on legal data.

**2. Judgment Predictor (judgmentPred.py)**

* Purpose: To predict the possible outcome of a legal case based on user-provided details.
* Uses machine learning models trained on historical court data.

**3. Legal Document Generator (docGen.py)**

* Purpose: Helps users generate legal documents like affidavits, contracts, or notices.
* Likely uses template-based generation and NLP-driven content suggestion.

**Key Features:**

* **Modular Design**: Separation of concerns via individual files for each feature.
* **Dynamic Execution**: Pages are loaded at runtime, making the app extensible and maintainable.
* **User-Friendly Interface**: Built with Streamlit, ensuring easy interaction.
* **Error Handling**: Covers file errors, Unicode decoding issues, and other runtime exceptions.

**How It Works:**

1. Users select one of the three modules via the sidebar.
2. The application reads the corresponding .py script.
3. That script is executed, and its content is rendered in the main view.
4. Page-specific headers and captions are displayed consistently.

**Conclusion:**

This project presents a practical application of AI in the legal domain. It leverages a clean modular architecture, user interactivity through Streamlit, and intelligent backend processing. The system is designed to improve accessibility and reduce time/cost for legal services.

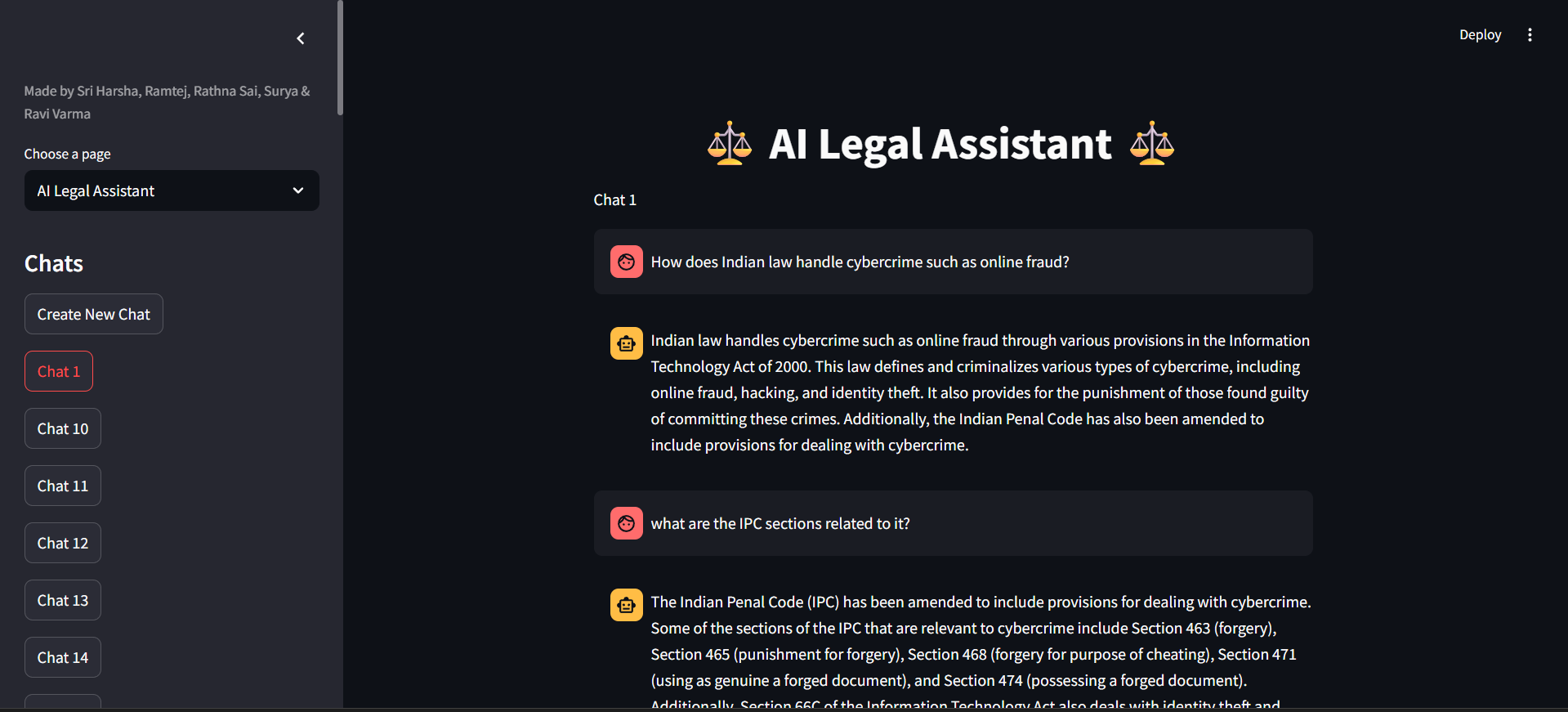
**Future Scope:**

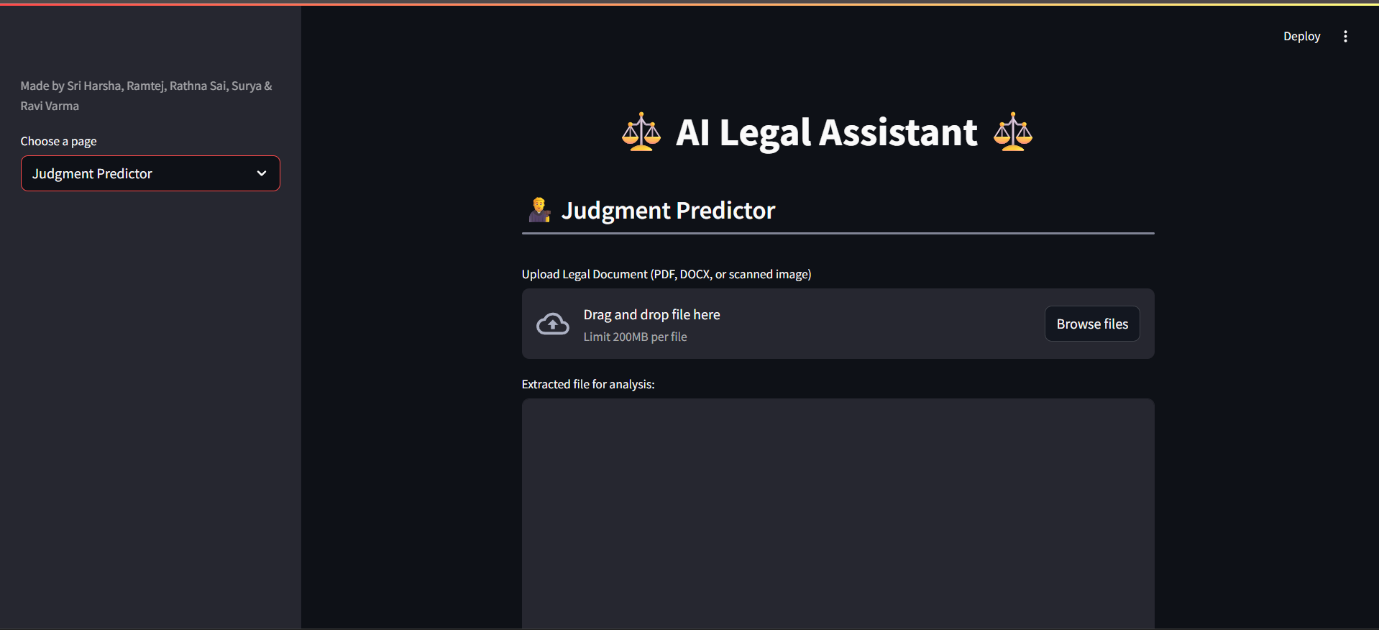
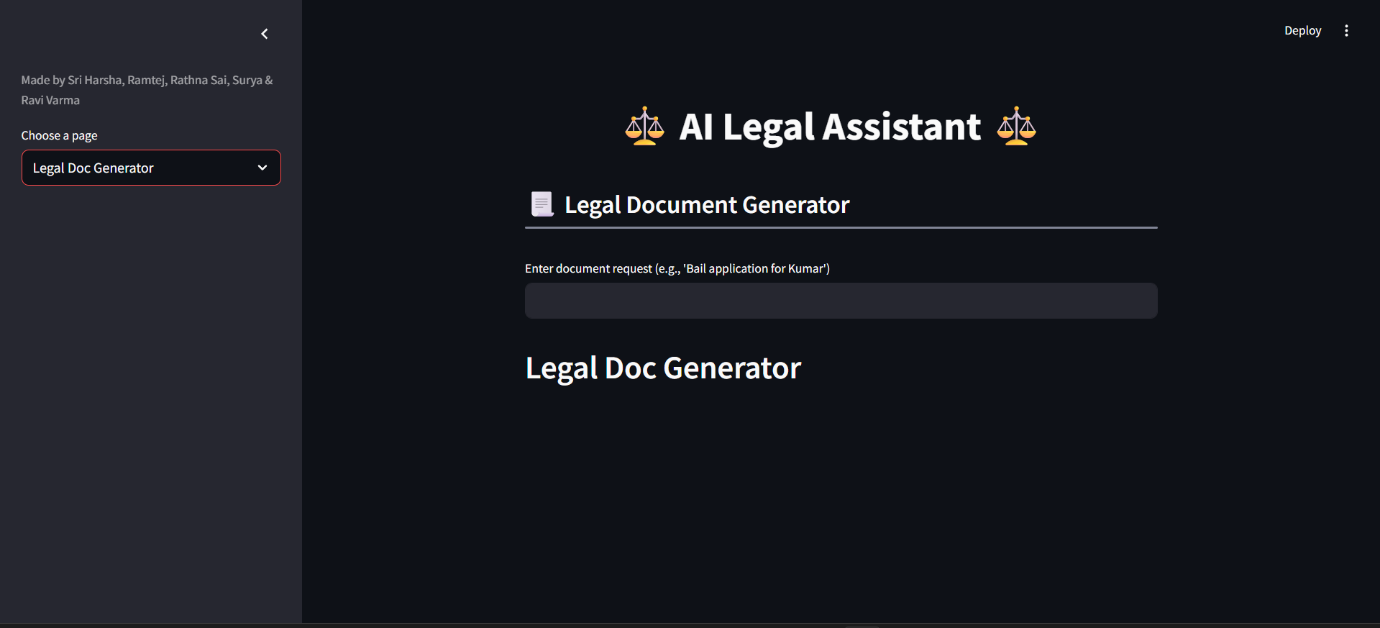
* Add **authentication** for document-sensitive operations.
* Integrate **LLMs** (like GPT-4 or fine-tuned legal BERT models).
* Expand support for **multiple jurisdictions/languages**.
* Include **document upload and analysis** features.
* Log and analyze user queries to improve chatbot performance.

**PROJECT LINK:**

<https://github.com/Harsha9009/AI--Legal-Assistance.git>

**PROJECT PICTURES:**

****

****

**SCREEN RECORDING:**

****