Sprint	User Story/Tasks ID	User Story/Tasks	Acceptance Criteria	Status
0	0.1	As a developer, I want to research and understand the immigration document requirements (I-20, I-94), so that I can build the necessary parsing functionality.	Clear understanding of I-20 and I-94 document structures. Document key fields that need to be parsed (e.g., Name, DoB, Visa Status).	Completed
0	0.2	As a developer, I want to set up the development environment and tech stack, so that the team can start building the application.	Development environment is set up (Python, Flask, AI frameworks). Shared repository for the team to collaborate.	Completed
1	1.1	As an International Student, I want to upload my I-20 document, so that I can view my basic information (Name, Date of Birth).	User can upload an I-20 PDF from the web interface. The system extracts and displays Name and Date of Birth.	Completed
1	1.2	As a developer, I want to implement the PDF upload functionality on the frontend and backend, so that students can upload their documents.	Users can upload PDFs from the frontend. The backend processes uploaded documents.	Completed
1	1.3	As a developer, I want to implement basic parsing functionality for I-20 documents, so that key fields (Name, DoB) can be extracted.	System parses I-20 documents and extracts Name and Date of Birth. Extracted information is displayed in the frontend user interface.	Completed
1	1.4	As a developer, I want to ensure the application has error handling during the document upload process, so that users receive appropriate feedback for failed uploads.	Clear error messages for upload issues (e.g., wrong file type, corrupted file). The system logs errors for troubleshooting.	Completed
1	1.5	As a developer, I want to set up the basic structure of the Flask backend, so that I can manage file uploads and communication between the frontend and backend.	Flask is set up with routes for handling file uploads. The backend communicates with the frontend for document uploads.	Completed
2	2.1	As a developer, I want to collect immigration-related documents, so that the team can use them to provide personalized advice.	USCIS documents are collected. Common immigration-related Q&A are compiled.	Completed
2	2.2	As a developer, I want to implement a tagging system, so that we have a structure to sort all the AI training materials which can train AI to provide personalized responses.	Materials are systematically identified and tagged. Document data is centralized and mapped to relevant questions accurately.	Completed
2	2.3	As a developer, I want to develop an AI model (using BERT or an RNN), so that the system can handle user queries and deliver accurate responses.	AI model is trained on a dataset of immigration-related queries and document data. Model achieves an acceptable accuracy threshold in query-response testing.	Completed
2	2.4	As a user, I want to input immigration-related questions or queries into the system, so that I can receive tailored advice.	Users can input and submit queries through the system. Queries are processed and mapped to relevant responses or document data.	Completed
2	2.5	As a user, I want a chatbox on the frontend where I can interact with the Immigration AI Advisor, so that I can easily ask questions and receive responses.	Chatbox is functional and user-friendly. Integrated with backend systems to process queries and display responses.	Completed
2	2.6	As a developer, I want to fine-tune the AI model, so that it delivers more accurate and high-quality responses to user queries.	Fine-tuned model achieves higher performance metrics compared to the baseline. Improvements are documented and validated against test cases.	Not Completed
3	3.1	Generate text responses based on query and knowledge base	The system generates accurate, coherent, and context-aware text responses. Responses are relevant to the query and align with the knowledge base. Handles queries outside the knowledge base gracefully by providing appropriate feedback or a fallback response	Not Completed
3	3.2	Improve semantic search	Semantic search returns the most contextually relevant results for user queries. Latency for search results is minimized and consistent. Improved precision and recall metrics for search functionality are documented.	Not Completed
3	3.3	Clean up database (look at "extracted_text" field/column in Firebase, clean up)	The extracted_text field/column is normalized and free from redundant or irrelevant data. All unnecessary or malformed entries in Firebase are removed. The cleaned database supports efficient queries and ensures consistency across records	Not Completed
3	3.4	Improve tokenization and segmentation methods (use something more state-of-the-art)	Tokenization and segmentation methods reduce errors in text parsing and analysis. Benchmarked against baseline methods with measurable improvements in performance. Integrated seamlessly into the existing AI pipeline without causing regressions	Not Completed
3	3.5	Support users (login/auth)	Users can register and log in securely. Authentication is implemented with industry-standard practices (e.g., OAuth, JWT). User roles (e.g., admin, student) are defined and enforced within the application	Not Completed
3	3.6	Store user documents/contents	Uploaded user documents are securely stored in a database or cloud storage. Documents are tagged and indexed for efficient retrieval. Document storage complies with relevant privacy and data protection standards (e.g., GDPR, CCPA).	Not Completed
3	3.7	Response to queries based on existing functionality, but incorporating user-uploaded documents	The system uses user-uploaded documents to generate tailored responses. Queries are processed to include relevant context from the uploaded documents. Provides accurate and personalized advice based on document data	Not Completed
3	3.8	Deploy a free, private demo version to web	A demo version of the application is deployed to a web server. Users can interact with core functionalities (e.g., upload documents, ask queries). Demo is secured to prevent unauthorized access and data leaks.	Not Completed