Smart Loan App - Project Report

Executive Summary

The Smart Loan App is an innovative multilingual financial assistant designed to bridge the language gap in financial services in India. By leveraging advanced AI technologies, the application provides loan-related information, financial advice, and guidance in multiple Indian languages, making financial services more accessible to a diverse population.

Problem Statement

Financial literacy and access to financial information are significant challenges in India, particularly for non-English speakers. Many potential borrowers face language barriers when trying to understand loan eligibility criteria, application processes, and financial best practices. This project addresses these challenges by creating a multilingual conversational AI system that can provide financial guidance in the user's preferred language.

Solution Architecture

System Overview

The Smart Loan App employs a modular architecture with the following key components:

1. **User Interface Layer**:
    - Streamlit-based web interface for quick deployment
    - React-based frontend for enhanced user experience
    - Supports both text and voice input methods
2. **Natural Language Processing Layer**:
    - Intent classification using Groq LLM
    - Query processing and response generation
    - Multilingual support for English, Hindi, Tamil, Telugu, and Kannada
3. **Knowledge Retrieval Layer**:
    - Retrieval Augmented Generation (RAG) for accurate information extraction
    - PDF document processing for bank-specific loan information
    - Vector database for efficient information retrieval
4. **Speech Processing Layer**:
    - Speech-to-Text conversion for voice input
    - Text-to-Speech conversion for audio responses
    - Language-specific voice models
5. **Translation Layer**:
    - Translation services for converting responses to the user's preferred language
    - Support for multiple Indian languages

Technical Implementation

Backend Components

1. **Intent Classification (intent.py)**:
    - Uses a mission control system to classify user queries into categories
    - Categories include loan eligibility, financial tips, loan application guidance
    - Routes queries to appropriate specialized agents
2. **RAG System (rag.py)**:
    - Extracts text from PDF documents containing bank-specific loan information
    - Creates vector embeddings using HuggingFace models
    - Implements a retrieval system to find relevant information for user queries
3. **Speech-to-Text (stt.py)**:
    - Captures audio input from users
    - Processes audio using Sarvam AI's speech recognition API
    - Supports multiple Indian languages
4. **Text-to-Speech (tts.py, tts1.py)**:
    - Converts text responses to speech
    - Uses Sarvam AI's text-to-speech API
    - Handles chunking for long responses
5. **Translation Services (translate.py, translate2.py, translate3.py)**:
    - Translates responses to the user's preferred language
    - Implements sentence splitting for better translation quality
    - Uses Sarvam AI's translation API

6. **Web Search Integration (serp.py)**:
    - Provides up-to-date financial information from the web
    - Uses Serper API for web search capabilities
    - Extracts relevant financial tips and advice

Frontend Components

1. **User Interface**:
    - Language selection dropdown
    - Input method toggle (voice/text)
    - Response display area
    - Audio playback controls
2. **API Integration**:
    - Communicates with backend services
    - Handles file uploads for document processing
    - Manages audio recording and playback

Implementation Details

Data Flow

1. **User Input Processing**:
    - User selects language and input method
    - For voice input, audio is recorded and sent to the speech-to-text service
    - For text input, the text is captured directly
2. **Query Processing**:
    - The input is sent to the intent classification system
    - Based on the identified intent, the query is routed to the appropriate agent
    - The agent retrieves relevant information using the RAG system or web search
3. **Response Generation**:
    - The system generates a response based on the retrieved information
    - The response is translated to the user's preferred language if needed
    - The translated response is converted to speech
4. **Output Delivery**:
    - The text response is displayed in the UI
    - The audio response is played through the user's device

Key Technologies

1. **LangChain & LangGraph**:
    - Used for building the LLM application framework
    - Enables the creation of agent workflows and chains
2. **Groq LLM**:
    - Provides fast inference for natural language understanding
    - Powers the intent classification and response generation
3. **Sarvam AI**:
    - Provides Indian language speech-to-text and text-to-speech capabilities
    - Enables translation between Indian languages
4. **FAISS Vector Database**:
    - Enables efficient similarity search for the RAG system
    - Stores and retrieves document embeddings
5. **HuggingFace Embeddings**:
    - Creates vector representations of text for semantic search
    - Uses the all-mpnet-base-v2 model for high-quality embeddings
6. **Streamlit & React**:
    - Provides the user interface framework
    - Enables responsive and interactive user experience

Challenges and Solutions

Challenges

1. **Multilingual Support**:

    - Handling various Indian languages with different scripts and grammatical structures
    - Ensuring accurate translation and natural-sounding speech

2. **Voice Input Quality**:
    - Dealing with background noise and accent variations
    - Ensuring accurate transcription in different languages

3. **Information Accuracy**:
    - Providing up-to-date and accurate financial information
    - Ensuring compliance with financial regulations

Solutions

1. **Specialized Language Models**:
    - Using language-specific models for better accuracy
    - Implementing sentence-level translation for improved quality

2. **Audio Processing Techniques**:
    - Implementing noise reduction and audio normalization
    - Using language-specific speech recognition models

3. **RAG System Implementation**:
    - Using bank-specific documents for accurate information
    - Supplementing with web search for up-to-date information

Future Enhancements

1. **Additional Languages**:
    - Expand support to more Indian languages
    - Improve translation quality for existing languages

2. **Enhanced User Interface**:
    - Develop a mobile application for better accessibility
    - Implement a chat history feature for continuity

3. **Advanced Features**:
    - Integrate with bank APIs for real-time eligibility checks
    - Implement document upload for personalized advice
    - Add comparison tools for different loan products

4. **Performance Optimization**:
    - Improve response time through caching and optimization
    - Enhance voice recognition accuracy

Conclusion

The Smart Loan App successfully addresses the language barrier in financial services by providing a multilingual conversational AI system. By combining advanced NLP, speech processing, and information retrieval technologies, the application makes financial information more accessible to a diverse population. The modular architecture allows for easy expansion and enhancement, making it a scalable solution for improving financial literacy and access to financial services in India.

Appendix

API Documentation

Sarvam AI APIs

1. **Speech-to-Text API**:
    - Endpoint: `https://api.sarvam.ai/speech-to-text`
    - Parameters: language_code, model, with_timestamps
    - Returns: Transcribed text

2. **Text-to-Speech API**:
    - Endpoint: `https://api.sarvam.ai/text-to-speech`
    - Parameters: inputs, target_language_code, speaker, model, pitch, pace, loudness
    - Returns: Base64-encoded audio data

3. **Translation API**:
    - Endpoint: `https://api.sarvam.ai/translate`
    - Parameters: source_language_code, target_language_code, speaker_gender, mode, model, input
    - Returns: Translated text

Backend APIs

1. **Query Processing API**:
    - Endpoint: `/api/query`
    - Method: POST
    - Parameters: query, language_code
    - Returns: Response text, audio URL
2. **Document Upload API**:
    - Endpoint: `/api/upload`
    - Method: POST
    - Parameters: file
    - Returns: Success status, document ID

Environment Setup

Required Environment Variables

```
# API Keys
SARVAM_API_KEY=your_sarvam_api_key
GROQ_API_KEY=your_groq_api_key
SERPAPI_API_KEY=your_serpapi_api_key
FINANCIAL_API_KEY=your_financial_api_key
HF_API_TOKEN=your_huggingface_token
```

Development Environment

- Python 3.8+
- Node.js 14+
- Required Python packages in requirements.txt
- React dependencies in package.json