High-Level Design (HLD) Document for BankBot Project

Jaideep J.

January 26, 2025

1 Introduction

The purpose of this High-Level Design (HLD) document is to outline the architecture and design of the BankBot project, an advanced banking chatbot application. This document provides a high-level overview of the system components, their interactions, and the flow of data.

2 System Overview

BankBot is a natural language processing (NLP)-based chatbot application that answers banking-related queries and performs various banking operations. The system integrates machine learning models to provide intelligent responses to user queries, helping users stay informed about their banking activities.

3 System Architecture

The architecture of the BankBot system follows a modular approach. It consists of the following key components:

1. **Frontend Interface** - The user interacts with the chatbot through a web-based interface.
2. **Backend Server** - The backend processes user input, communicates with the pre-trained models, and returns responses to the user.

3. **Machine Learning Models** - A Random Forest model is used to classify user queries, and a TF-IDF vectorizer helps in transforming user input into a format suitable for the model.

4. **Database (Optional)** - A database may be used to store historical data, transaction information, and user interaction logs.

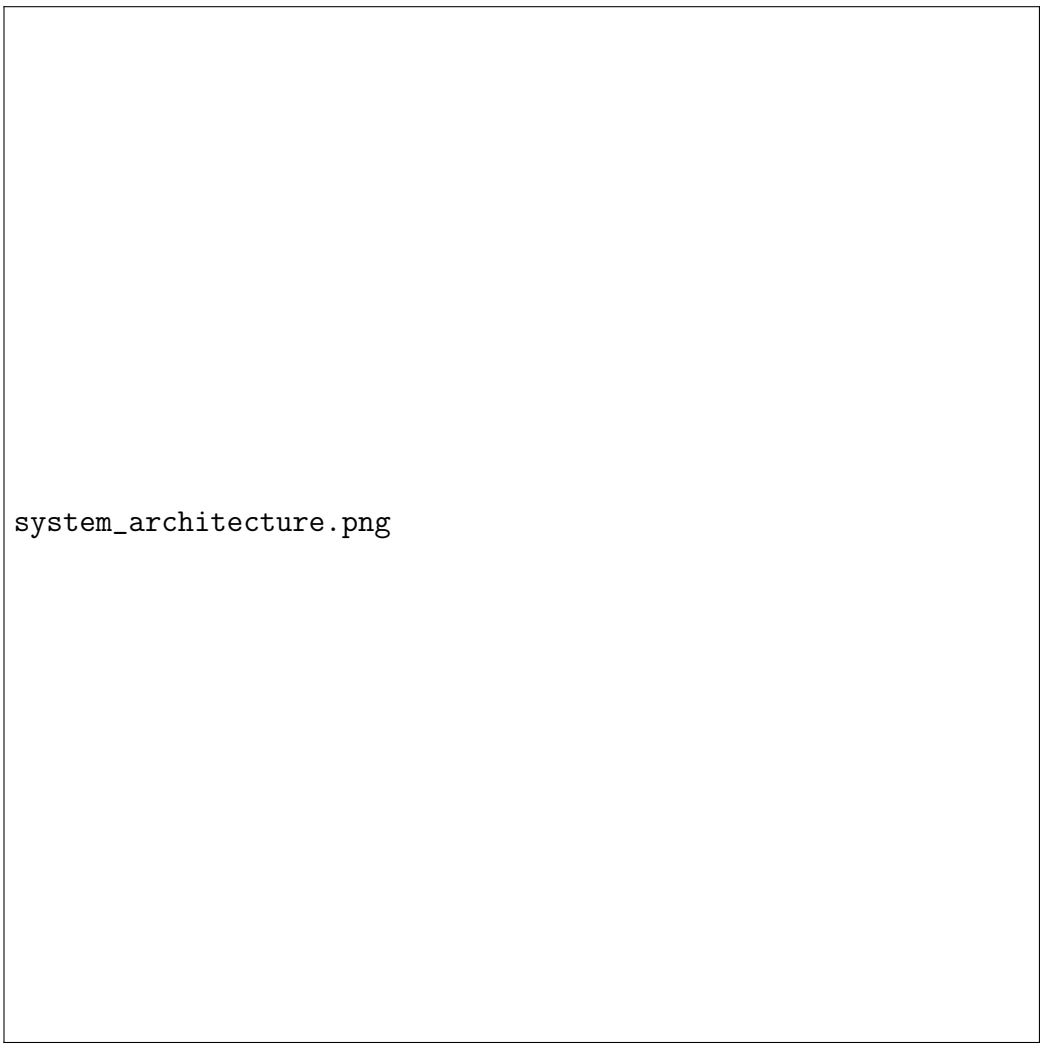

Figure 1: High-Level Architecture of BankBot

4 Components and Their Functions

4.1 Frontend Interface

The frontend interface is a web-based user interface where users can input their banking-related questions. It allows for seamless interaction with the backend and displays the responses generated by the chatbot.

- **Technology Used:** HTML, CSS, JavaScript (React.js or Angular)
- **Functions:** Collect user input, send requests to the backend, and display chatbot responses.

4.2 Backend Server

The backend is implemented using Flask, a Python web framework. The backend server receives the user input, processes it using the machine learning model, and returns the response.

- **Technology Used:** Python, Flask
- **Functions:**
  - Preprocess user input (e.g., lowercase conversion, removing punctuation).
  - Transform input into a suitable format using the TF-IDF vectorizer.
  - Predict the category of the user query using the Random Forest model.
  - Return the predicted response to the user.

4.3 Machine Learning Models

The system uses the following models:

1. **Random Forest Model:** A classifier that predicts the category of the user query.
2. **TF-IDF Vectorizer:** A tool that converts the text input into numerical features suitable for model input.
3. **Label Encoder:** Encodes categorical output into numerical format for model processing.

- **Technology Used:** Scikit-learn
- **Functions:**
  - Transform user input into a feature vector using TF-IDF.
  - Classify the query using the Random Forest model.
  - Decode the model's output into a readable response using Label Encoder.

4.4 Database (Optional)

If implemented, the database stores user interaction logs, transaction data, and system-generated logs. This enables tracking of user activity and provides historical data for the chatbot's responses.

- **Technology Used:** MySQL, PostgreSQL, or NoSQL (MongoDB)
- **Functions:**
  - Store transaction and user interaction data.
  - Retrieve stored data for future interactions.

5 Data Flow

The following diagram illustrates the flow of data in the system:

5.1 User Query Processing Flow

1. The user submits a query through the frontend interface.
2. The backend receives the input and preprocesses it (converts to lowercase, removes punctuation).
3. The preprocessed text is transformed into numerical features using the TF-IDF vectorizer.
4. The transformed input is passed through the Random Forest model for classification.
5. The predicted class is mapped back to the readable response using the Label Encoder.
6. The response is returned to the frontend for display to the user.

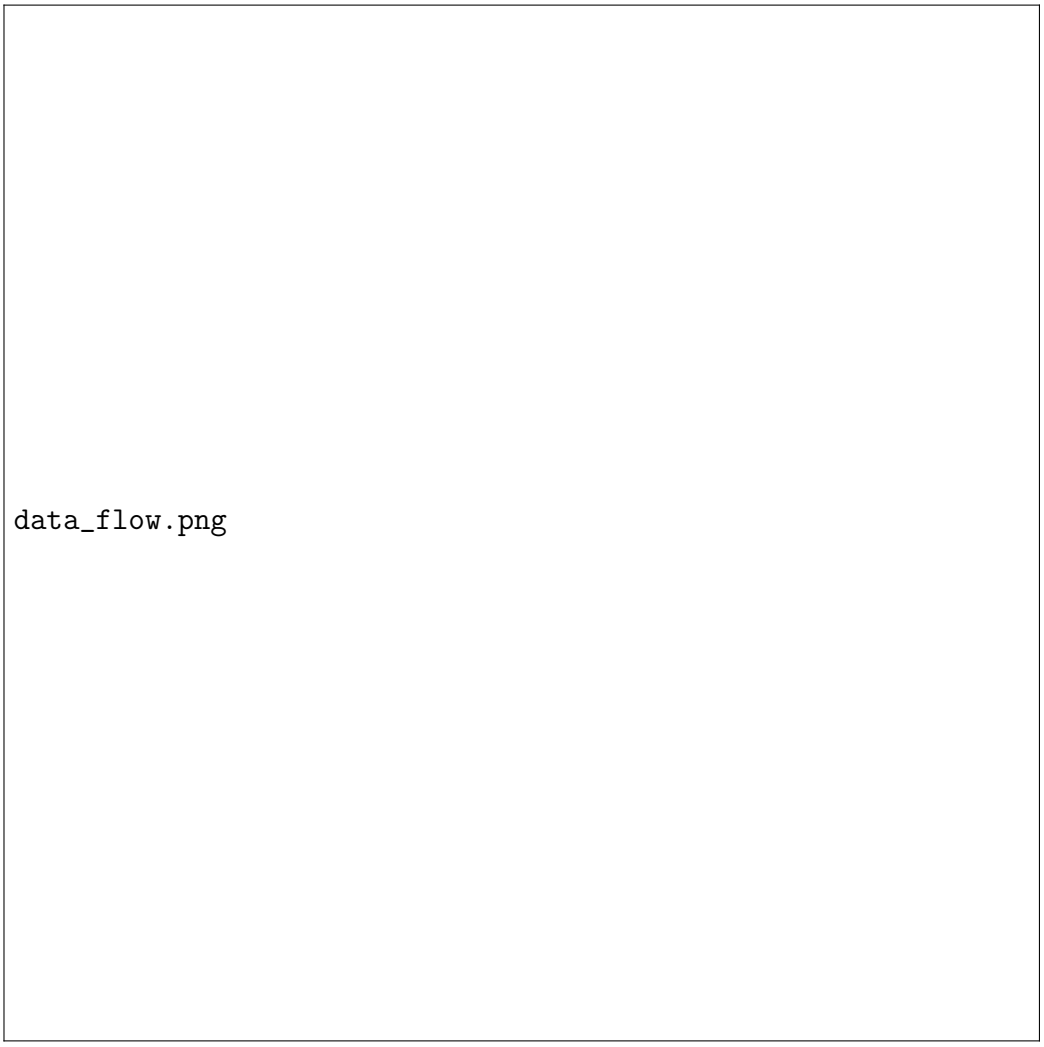

Figure 2: Data Flow in BankBot System

6 Security Considerations

Security is a key aspect of the BankBot system. The following measures should be implemented:

- **Input Validation:** Ensure that all user inputs are validated and sanitized to avoid malicious input and SQL injection attacks.
- **Encryption:** Use HTTPS to secure communication between the client and server.
- **Authentication and Authorization:** Implement user authentica-

tion for secure access to sensitive banking operations (if applicable).

7 Technology Stack

The technology stack used in the BankBot project includes:

- **Frontend:** HTML, CSS, JavaScript (React.js or Angular)
- **Backend:** Python (Flask)
- **Machine Learning:** Scikit-learn (Random Forest, TF-IDF, Label Encoder)
- **Database (Optional):** MySQL, PostgreSQL, or MongoDB
- **Deployment:** Render (for deployment of Flask app)

8 Conclusion

This High-Level Design (HLD) document provides an overview of the system architecture, components, data flow, and security considerations for the BankBot project. The next step is to move forward with the implementation, following the design principles outlined here.