Google chatbot Design Document

Vaibhav Vashisht
Email: vaibhav.vashisht94@gmail.com

1.1 Purpose

This document outlines how our chatbot system works. The chatbot is designed to help users with various tasks like managing tasks, scheduling events, and updating contacts on Apple devices. We've built the system using React for the user interface, Node.js for API setup, Integromat for routing, and Google Cloud services for the backend functions. Our chatbot also interacts with services like Dialogflow, Google Sheets, and Google Calendar to provide comprehensive assistance. This design document gives you a peek into the architecture, components, and data arrangement of our chatbot application.

1.2 Scope

The Google Chatbot serves as an AI-powered chatbot, designed to accept tasks from users, interpret them, carry out the tasks, and offer suitable responses. This is achieved by transforming English sentences into queries suitable for machine processing, seeking essential information to execute the tasks, and ultimately delivering responses in a natural and human-like manner.
The primary goal is to create an application that offers a service for handling natural language queries. Additionally, we will develop illustrative interfaces for mobile, and text messaging platforms to showcase the practical application and usability of the app.
The scope of the project extends to encompass integration with various applications utilized on the user's mobile device, offering a centralized hub for task management and information retrieval. This holistic approach aims to consolidate functionalities and streamline user interactions by providing a singular platform to manage tasks and access information across different applications.

1.3 Product Features

Natural Language Task Input: Users can input tasks using natural language sentences, allowing for a more intuitive and user-friendly interaction.

Task Understanding: The chatbot employs natural language processing (NLP) to comprehend and accurately interpret the tasks provided by users.

Task Execution:
The chatbot executes tasks on behalf of the user, leveraging its integration with various services to perform actions like scheduling events, setting reminders, or sending messages.

Information Retrieval:
The chatbot accesses and retrieves relevant information from integrated services, such as fetching contact details, events, or tasks from Google Calendar or Apple devices.

Human-Like Responses:

Responses provided by the chatbot are formulated in natural language, offering a conversational and human-like interaction experience.

Integration with Multiple Platforms:
The application integrates across various platforms, including web, mobile, and text messaging interfaces, ensuring accessibility from different devices and mediums.

Cross-App Integration:
The application serves as a central hub to manage tasks and information across multiple apps on the user's device, creating a unified user experience.

Notification and Alerts:
Users receive notifications and alerts about upcoming tasks, events, or updates based on the information processed by the chatbot.

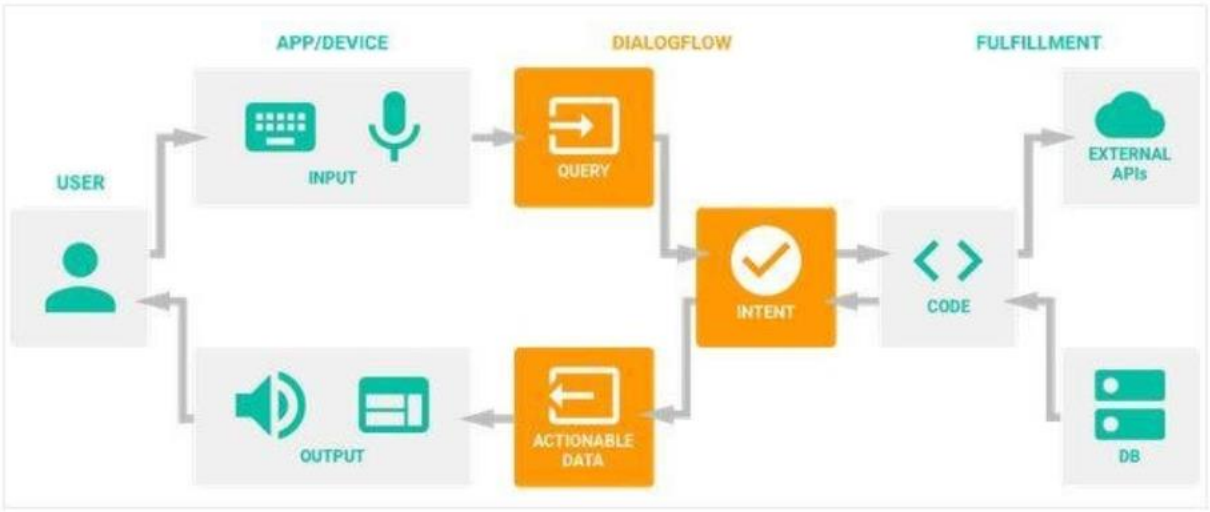
Third-Party Integrations:
The application offers the ability to integrate with additional third-party services and APIs, expanding its functionalities further.

Voice Interaction:
Voice-based interaction allows users to communicate with the chatbot using voice commands, enhancing accessibility and convenience.

These features collectively contribute to creating a robust and versatile AI chatbot application that simplifies task management and enhances user interactions across various services and platforms.

1.4 Technologies Used: React,Node, GCP, DiaglogFlow, Integeromat

1.5 Architecture
The app architecture consists of the following components:
React: To built simple interactive UI chatbot
Dialogflow: Google's natural language processing platform for chatbot conversation management.
Node.js: Backend server for handling API requests and integrating with Google services.
Google Sheets: A spreadsheet-based storage system for managing tasks and accessing google sheets data.
Google Calendar: A calendar service for scheduling and updating meetings and deleting events.
Google Cloud Console: A web-based interface for managing and configuring Google Cloud services.



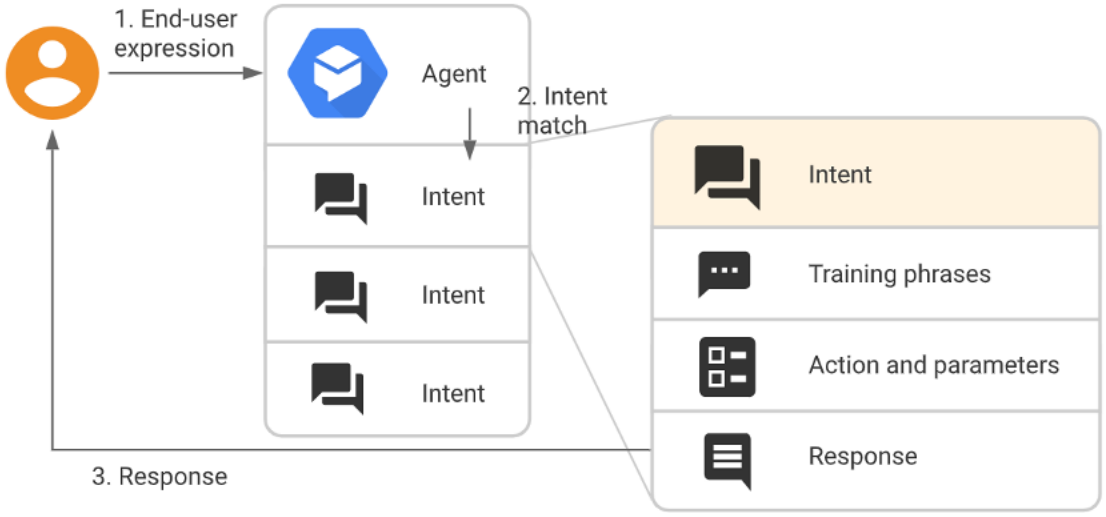
1.6 Data Design:
a. Task Data:
Each task within the application is defined as a structured data entity encompassing the subsequent attributes:
Task ID: A distinctive identifier singularly associated with the task.
Task Name: A succinct description characterizing the nature of the task.
Task Priority: A classification indicating the task's priority level, categorized as High, Medium, or Low.

b. Event Data:
The application conceptualizes each meeting as a coherent data entity, featuring the ensuing attributes:
Meeting Title: The denotation encapsulating the main focus or subject of the meeting.
Meeting Start Date: The precise date and time signifying the commencement of the meeting.
Meeting End Date: The exact date and time designating the conclusion of the meeting.
Meeting Duration: The temporal extent representing the duration span of the meeting.

c. Apple Device Integration:
The application extends compatibility to Apple devices by offering reminder and notification services that encompass the following functionalities:
Reminder Content: Notification reminders comprise comprehensive information outlining the context and relevance of the impending event.
User-Specific Queries: The chatbot facilitates personalized inquiries, tailored to update specific contacts within the user's Apple device contact list.

1.7 User Flow: The user interacts with the chatbot app using text-based conversations. The chatbot understands user intents and performs appropriate actions based on the request.

1.8
Components and Functionality:
a) Dialog flow Integration
Dialog flow is configured to handle natural language processing and intent recognition for user queries. It agent is trained with appropriate intents, entities, and responses for task management, meeting scheduling



b) Node.js Server:
Implements API endpoints to handle requests from the dialog Flow and interact with Google services.
Acts as an intermediary between the front end app and dialog Flow services, facilitating data retrieval and updates.

c) Google Sheets Integration
Allows the app to add new tasks to a Google Sheet and delete tasks based on user requests.
Server communicates with the Google Sheets API to perform read and write operations on the sheet.

d) Google Calendar Integration:
Enables users to schedule meetings and update them through the app.
Server interacts with the Google Calendar API to create, update, and delete events.

e) Apple Phone Integration:
Provides functionality to reminders, updating contacts.

f) React App Interface:
Implements the user interface using React
Sends user queries to the Node.js server and displays responses from the chatbot.

API Configuration and Webhooks:
Server sets up API endpoints for the react to send requests and receive responses.

Google Cloud Console:
The Google Cloud Console is used to configure and manage cloud services required for the app, such as Dialogflow
It provides a centralized interface for setting up authentication, enabling APIs, managing service accounts, and monitoring usage and quotas.

1.9 User Interface Design: The user interface design including the layout, navigation, and visual elements of the app.
Chatbot interaction
Text to speech conversion
Scrolling

1.10 Data Storage:
Google Sheets is used to store and manage task data.
Google Calendar stores meeting schedules and updates.
Apple Device to receive udpates

1.11 Security and Authorization
Authorization mechanisms implemented using google cloud console
Appropriate security measures should be implemented, including authentication mechanisms to protect user data and ensure secure interactions with Google services.

1.12 Deployment and Scaling
Deployment is limited to web application
Deployment considerations should include scalability, performance, and monitoring of the backend server and Google services.

1.13 Testing and Maintenance:
Unit testing completed.
Extensive testing should be conducted to ensure proper integration, functionality, and usability of the app.
Regular maintenance and updates should be performed to address bugs, security vulnerabilities, and updates to Google APIs or other dependencies.

1.14 Future Enhancements
More API calls for all the applications
Realtime database
Trained model with extensive capabilities
UI enhancement
Data modelling
API interface enhancement

1.15 Configuration and steps:
Node and react: Install package and necessary dependencies and create end point for chatbot

Dialogflow Configuration:
Create a new Dialogflow project in the Google Cloud Console.
Configure the Dialogflow agent by defining intents, entities, and training phrases.
Obtain the Dialogflow API credentials and set up authentication for the Node app.

Node.js Server Configuration
Install Node.js and set up the development environment.
And a new Node.js project for the backend server.
Install the required dependencies
Configure the API endpoints for handling requests from the web application and interacting with Google services.
Create Google Cloud Console project and enable required APIs (Google Sheets, Google Calendar, Apple).
Set up service accounts and obtain the necessary API credentials (API keys, OAuth client IDs). Configure access permissions and scopes for the APIs to be used.

Google Sheets Configuration:
Create a new Google Sheet for task management.
Share the sheet with the service account and grant necessary permissions.
Obtain the Google Sheets API credentials and set up authentication in the Node.js server.

Google Calendar Configuration:
Create a new Google Calendar for scheduling meetings.
Share the calendar with the service account and grant necessary permissions.
Obtain the Google Calendar API credentials and set up authentication in the Node.js server.

Apple API Configuration
Install make app on your device
Enable the permission to access the information and update.