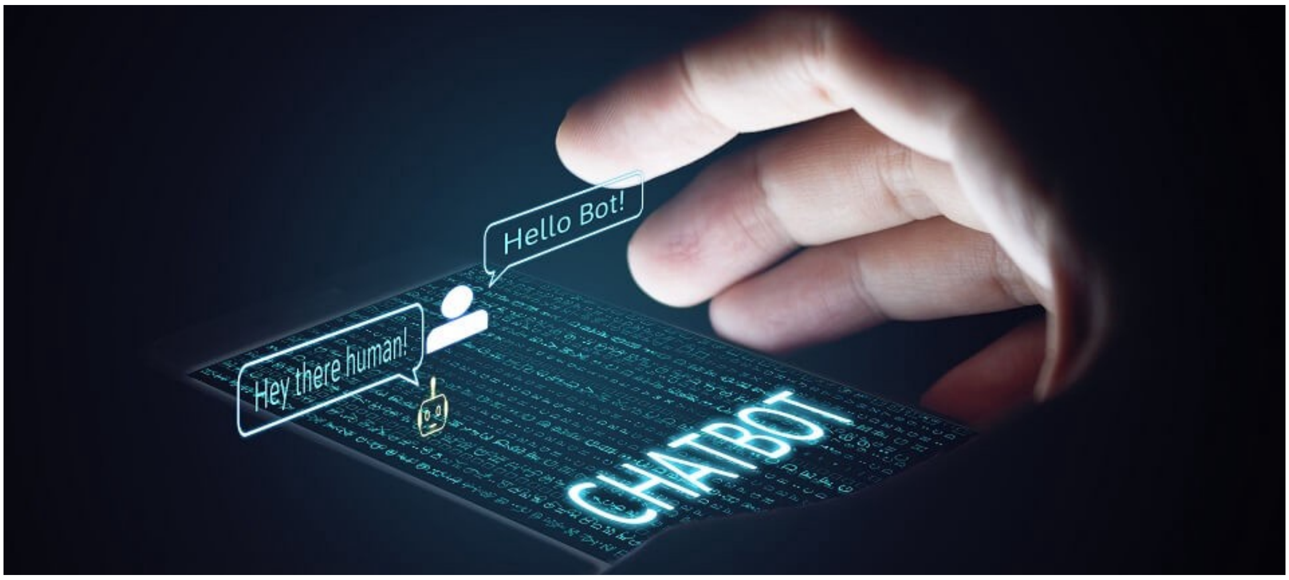
Software Requirements Specifictaion

EVOQUE: An intelligent chatbot



Submitted by:

20103120 Bharat Singh
20103121 Neeraj Kumar
20103122 Sushant Singh
20103123 Divyansh Gurtoo

Introduction

Evoque receives questions from users, tries to understand the questions, and provides appropriate answers. To do this, it converts English sentences into machine-friendly queries, searches relevant data to find the information you need, and finally returns answers in natural language sentences. In other words, answer questions like a human would, instead of providing a list of websites that might have the answer. For example, when asked "Would you like to register a complaint?", the response is "Register now!".
It aims to provide a quick and easy way for students, women, and everyone else to get their questions answered, and to give other developers the ability to integrate chatbots into their projects.

System Features

The major features for Evoque will be the following:

- **Natural Language Processing:** The system will take in questions written
in standard English.
- **Natural Language Responses**: The answer to the question will be
written in standard and understandable English.
- **Information Extraction**: There will be a database containing all the
information needed, populated using information extraction techniques.

Constraints

- **Limited Question Scope**

Creating a chatbot able to answer every single question about Bot is not possible to implement with current technology and within the duration of the project, so the system will be able to answer questions about limited topics.

- **Language**

The system will only support questions in standard English

Feasibility

1. Technical Feasibility: This includes finding the technology for the project, both hardware and software. A virtual assistant requires users to have a microphone to deliver messages and a speaker to hear when the system speaks. These are very cheap these days
and generally everyone owns them. The system also requires an internet connection. Make sure your internet connection is stable while using Evoque. And with Wi-Fi available in nearly every home and office today, that's no problem either.

2. Operational Feasibility: Ease and simplicity of operation of the proposed system. The system does not require any special skills for the user to operate. In fact, it's designed for most people to use. Children who cannot yet write can read the system's tasks and get the answers.

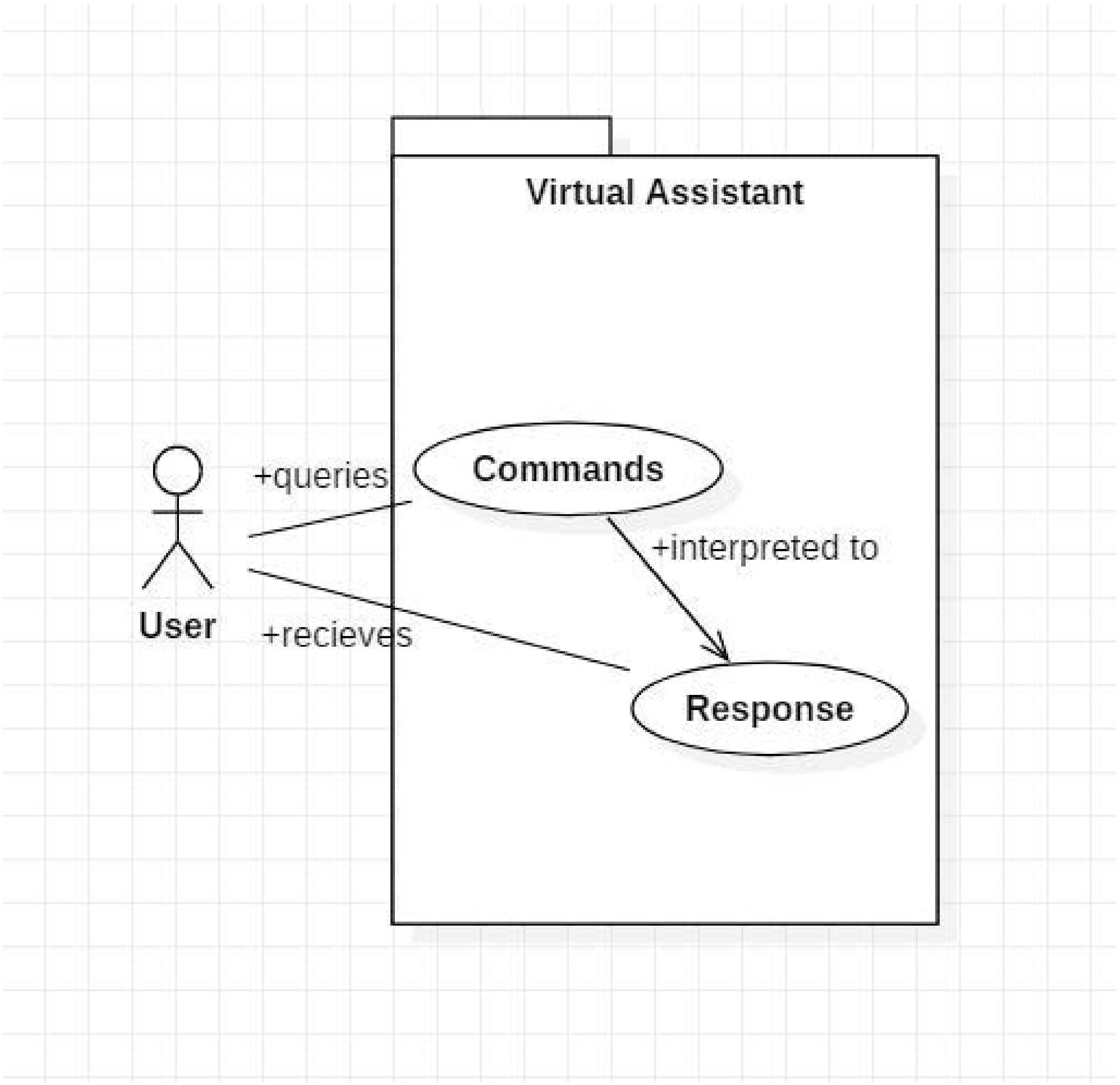
3. Economic Feasibility: Find here the overall costs and benefits of the proposed and current systems. In this project the main cost is the cost of documentation
Users also have to pay for microphones and speakers. Again, they are cheap and available. As far as maintenance goes, the Evoque doesn't cost much.

4. Organizational Feasibility: Demonstrate project management and organizational structure. This project was not created by a team. All administrative tasks are performed by one person. This avoids administrative problems and increases the feasibility of the project.

5. Cultural Feasibility: Address the compatibility of the project with the cultural context. Virtual assistants are built according to popular culture. This project is technically viable without any external hardware requirements. Plus, it's easy to use and requires no training or repairs. An overall feasibility study

of the project indicates that the objectives of the proposed system are achievable. A decision is made to proceed with the project

Use Case Diagram



Software Development Methodologies

Deciding upon an appropriate methodology is vital for the overall development of any software application to ensure a realistic timeframe is established for each stage of the project and requirements are clearly outlined. Various development methodologies will be discussed and considered for the development and design of this software. This section will highlight the development methodology that is best suited to this project.

Incremental Model

This software methodology evolved from the waterfall model. The application is designed, developed and tested using iterative incremental build stages. At the end of each build a subsystem or feature will be created. The project will progress in complexity as new requirements are likely to be discovered and implemented in each incremental build, developing on top of the functionality from the last build leading to the overall development of the application. It is very common for software to be released in stages, it is critical that component versions utilised within the software are managed throughout the entire lifecycle using version control tools such as GitHub. Each build will only last a few weeks to produce a baseline version of the application. Feedback can be given on any requirement errors or faults found in the application. Distributing the development of the project over various build cycles can lower the risks associated with development to a more manageable level as requirements are broken down into smaller functionality to be implemented at the end of each build.

The incremental model is the most suitable development methodology to implement for this project. The flexibility of the incremental model makes it ideal for this project as it is likely new requirements will be identified during the later stages of development and each iterative build makes it easy to implement new requirements throughout the the development process.

Chosen Information Gathering Technique

The most efficient way to gather information for this project would be the use of a Questionnaire, distributed online to the general public. This will reach a large user group, producing a diverse and varied result set for analysis providing detailed scope on what users from different age groups and backgrounds are expecting from an application such as this. It will give valuable insight into the age group, how often the user utilises online banking, if they would prefer assistance through an advisor on phone or using a friendly chatbot service, level of computer literacy and their expectations from the application.

Functional and Non-Functional Requirements

Functional and Non-functional requirements are identified through the analysis of the data collected from the survey. Functional requirements are the features and functionality that the system must have or be able to perform whereas non-functional requirements define the manner or characteristic the system must have such as: performance, usability, modifiability, maintainability, security, scalability, reliability, availability, configurability and design constraints .

Functional Requirements

Users will be able to converse with the Chat bot through voice or text commands and it will understand what the user is saying through natural language understandinprocessing provided through the integration of Dialogflow API.

• The chat bot should be able to maintain the conversational state when the context may be unclear through previous messages and conversations.
• Provide text and audio responses. The Chat bot will assist users with their queries and carry out appropriate actions such as scheduling appointments.

Non-Functional Requirements

• The chatbot must be efficient with very little lag in response time for instance no longer than 5 seconds to reply to a user message.
• The chatbot must be reliable with next to no faults or bugs
• The database must be scalable to adopt to a growing number of users

HARDWARE AND SOFTWARE REQUIREMENTS

The software is designed to be lightweight, so it won't overwhelm your running machine. This system has been created with compatibility of commonly available hardware and software in mind.

Here are the minimum hardware and
Software requirements for Virtual Assistant.
Hardware:
• Pentium-pro processor or later.
• RAM 512MB or more.

Software:
• Windows 7(32-bit) or above.
• Python 2.7 or later
• Chrome Driver
• Selenium Web Automation • SQLite