B.E. PROJECT ON

Personal Assistant for Exploiting Web Services

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS OF AWARD OF
B.E. (COMPUTER ENGINEERING)
DEGREE OF UNIVERSITY OF DELHI

SUBMITTED BY:

R. Varun	331/CO/13
Puneet Grover	329/CO/13
Neeraj Kumar Sinha	307/CO/13

GUIDED BY:

Dr. Bijendra Kumar

COMPUTER ENGINEERING (COE)
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI

2013-17

नेताजी सुभाष प्रौद्योगिकी संस्थान

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)

Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi - 110 078
Telphone : 25099050; Fax : 25099025, 25099022 Website : http://www.nsit.ac.in

CERTIFICATE

The project titled "**Personal Assistant For Exploiting Web Services**" by **R. Varun (331/CO/13)**, **Puneet Grover (329/CO/13)** and **Neeraj Kumar Sinha (307/CO/13)** is a record of bonafide work carried out by them, in the **Division of Computer Engineering**, **Netaji Subhas Institute of Technology**, **New Delhi**, under the supervision and guidance of **Dr. Bijendra Kumar** in partial fulfilment of requirement for the award of the degree of Bachelor of Engineering in Computer Engineering, University of Delhi in the academic year 2016 - 2017.

Dr. Bijendra Kumar
Division of Computer Engineering
Netaji Subhas Institute of Technology
New Delhi

Dated:

CANDIDATES' DECLARATION

This is to certify that the work which is being hereby presented by us in this project titled "**Personal Assistant For Exploiting Web Services**" in partial fulfilment of the award of the Bachelor of Engineering submitted at the **Department of Computer Engineering , Netaji Subhas Institute of Technology Delhi**, is a genuine account of our work carried out during the period from January 2017 to May 2017 under the guidance of **Dr. Bijendra Kumar**, Department of Computer Engineering, Netaji Subhas Institute of Technology, Delhi. The matter embodied in the project report to the best of our knowledge has not been submitted for the award of any other degree elsewhere.

Dated:

R. Varun	Puneet Grover	Neeraj Kumar Sinha
(331/CO/13)	(329/CO/13)	(307/CO/13)

This is to certify that the above declaration by the students is true to the best of my knowledge.

Dr. Bijendra Kumar

Netaji Subhas Institute of Technology (NSIT)

Azad Hind Fauj Marg

Sector-3, Dwarka, New Delhi

PINCODE : 110078

ACKNOWLEDGEMENT

No significant achievement can be done by solo performance especially when starting a project from ground up. This B.E. Project has by no means been an exception. It took many very special people to enable it and support it. Here we would like to acknowledge their precious co-operation and express our sincere gratitude to them.

Dr. Bijendra Kumar has again been very supportive and involved in yet another student project. It was his support that helped the project to start in its earliest and most vulnerable stages. He was always found with energy and enthusiasm to make sure that we were provided everything we needed. No amount of words can express thanks to him. He was the one who backed us in providing any assistance we needed during the project work.

We are also thankful to our friends who motivated us at each and every step of this project. Without their interest in our project we could not have been gone so far.

And the most of all, we would like to thanks our wonderful parents who motivated us from day one of the project. You were the lights that lead us.

It was a great pleasure and honour to spend our time with all of them and there could be no better payment for the efforts put into completing this B.E. Project than their valuable presence. They are all very special to us.

ABSTRACT

Technology has percolated into each and every sphere of our life. What seemed impossible yesterday is implemented today and evolved tomorrow. Using a Mobile phone is now considered a necessary skill but still a major part of the world can't use it efficiently. Technology empowers the less skilled person to find his place in the society despite being at a loss than the technically.

Our project aims to exploit Service-oriented architecture (SOA) which is a proven approach that aims at producing loosely coupled, standard-based, and protocol-independent services. A compliant SOA architecture provides independent units or services, allowing users to discover, execute and compose them in their applications.

A lot of apps in the market today, provides various types of services targeting the growing mobile phone users. This adds a level of confusion among the users as to find the best service to complete a given task thus making it difficult for some to use it. Our application, aims to simplify such complexities by allowing the user to express his instruction in Natural Language without the need of choosing complex options and buttons. This facilitates the people who find it difficult to decide on which service to use for a given task.

Table of Contents

CHAPTER 1: DETAILED PROBLEM STATEMENT

1.1 Introduction

In the world of plethora of apps and services available to us, it’s becoming very complicated and hard to use and handle these many apps. There are ton of apps for different services and everyone of them has different functionality and way of using. It can be tough for someone to get accustomed to it. In the pursuit of decreasing this complication for us and for others we sought out to target the APIs provided by the services. Most of the services on the brighter end of the of number of users per service give their own API, so that other apps and services can integrate their service in their app. We thought of giving all the services at one place. So, that the user can find his most used apps and services in his day to day life at a single place.

A personal assistant (PA) or “digital butler” is a specialized intelligent artificial agent that helps users to do their activities. From a Multi-Agent Systems perspective, the PA is an intermediary between humans and other agents in a multi-agent environment. The interaction with a PA varies from the simple execution of commands to more sophisticated natural conversation with humans. The usage of PA’s is promoted by projects like the PAL Program (Personalized Assistant that Learns) proposed by DARPA, with contributions from SRI and several other laboratories with the CALO project.

1.2 Problem Statement

Many existing service platforms require the use of secondary help from other services. We aim to replace this dependency using a single platform which is easily available. The intended recipient of this tool would be a smartphone user who can seamlessly take advantage of services using just this platform.

A chat interface is used to get the command given by the user and depending on user's service request, one can access the desired service from the platform.The predefined questions helps the user to give some sort of idea as to what types of questions can be answered using this platform. Personal assistants with natural language interfaces must deal with ambiguity, uncertainty and noisy information. Building such interfaces is a difficult task that requires expertise found on different fields of research.

Inputs to the system: Text entered through chat interface

Outputs from the system: Desired services accessed output

1.3 Motivation

With the growing number of apps and services, and even many apps for the same service it's becoming difficult to maintain and decide which service to use for that service.

In an research it was shown that apple announced that the App Store now has 700,000 apps available, up from 650,000 in June. As of late June, Android had 600,000 apps available and Windows Phone had 100,000. Those numbers have presumably risen since then. Apple also noted that 90% of those 700,000 apps were downloaded at least once a month and that the average iOS user has 100 apps on his or her device.[5]

The App Store is becoming increasingly unwieldy and difficult to navigate as the number of apps available continues to explode.

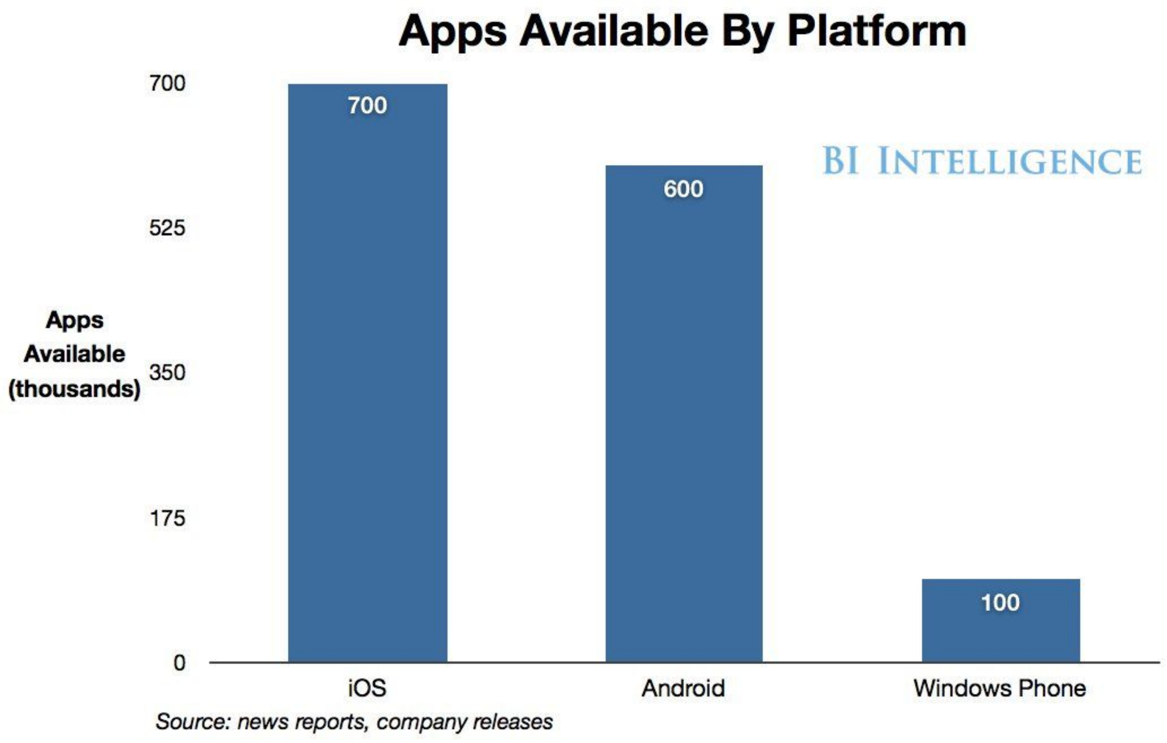


Fig. 1: Apps available in apple app store

[6] Though 20% of apps are opened only once, people are becoming more attached to apps overall. According to Localytics data, the amount of time people spend in apps has increased by 21% over the last year. People are engaging with mobile so frequently that time spent with mobile apps now exceeds time spent on traditional desktop web. Certain categories are driving this increased time spent in apps. Here is a summary of the findings:

- Overall time in app has increased by 21%

- Users open an app on average 11.5 times a month, up from 9.4 a year ago (22% increase), while app session length has remained constant at 5.7 minutes
- Social Networking experiences strong “snacking” behavior

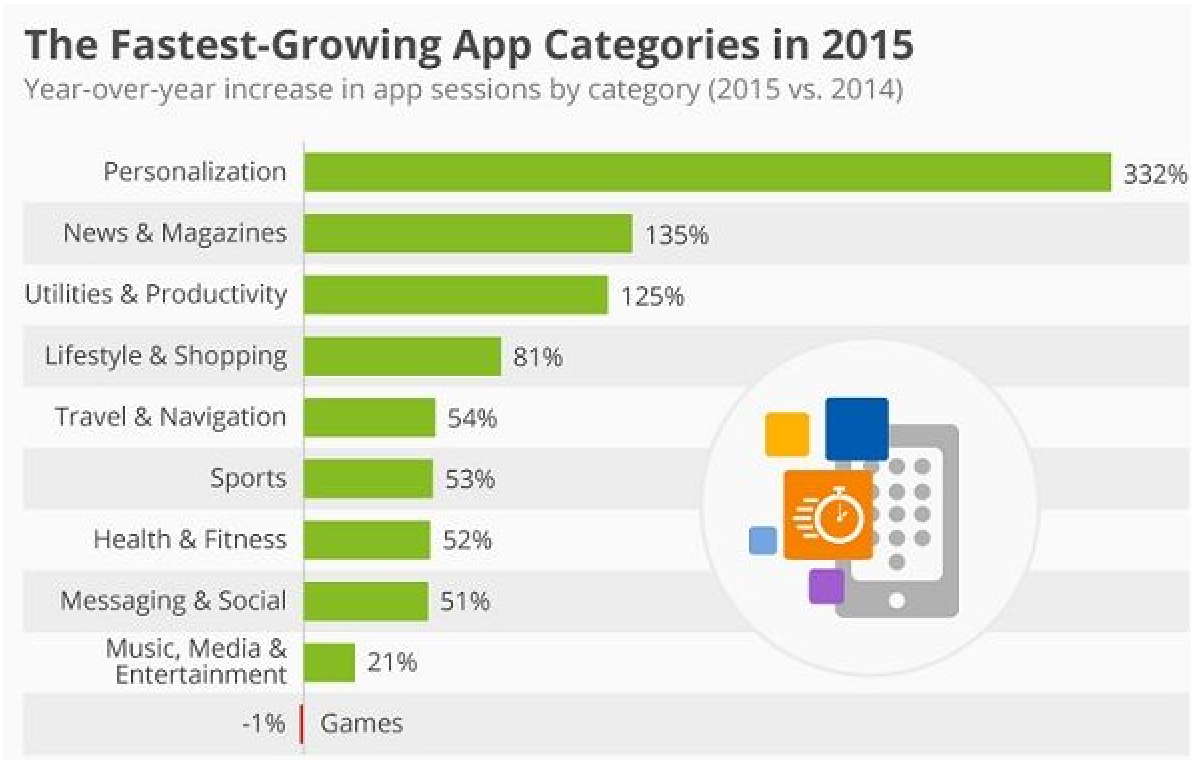


Fig. 2 : The fastest-growing App Categories in 2015 [4]

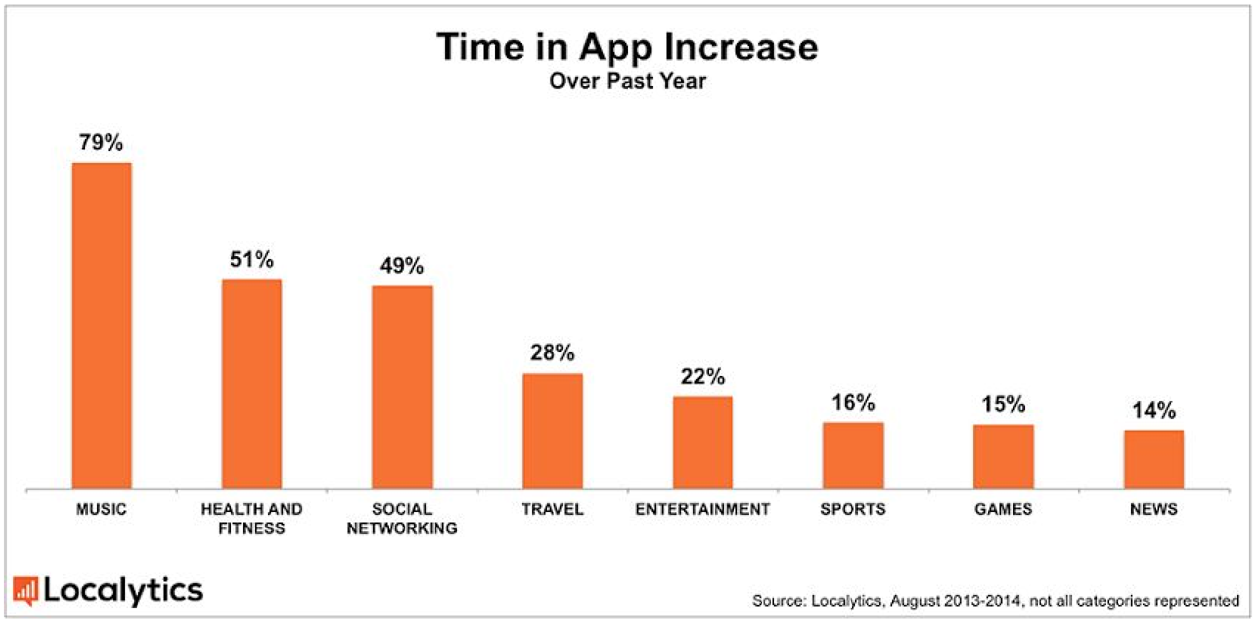


Fig. 3 : Time in App Increase [6]

Since there are so many applications to choose from, our application proposes to alleviate the problems faced by the smartphone users when they want to access any service from a large pool of available services.

Our project on **Personal Assitant for Exploiting Web Services** attempts to reduce the need to depend on various different platforms, which is often difficult to operate, and helps those having limited knowledge and even those who prefer not to use different applications. We thus decided to develop the application for the same in iOS (leading smartphone OS) and name it **PAWS** (Personal Assistant for Web Services).

Chapter 2. Literature Survey

2.1 API

In computer programming, an ***Application Programming Interface*** (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables or remote calls. [2]

Design:

The design of an API has significant impacts on its usability. The principle of information hiding describes the role of programming interfaces as enabling modular programming by hiding the implementation details of the modules so that users of modules need not understand the complexities inside the modules. Thus, the design of an API attempts to provide only the tools a user would expect. The design of programming interfaces represents an important part of software architecture, the organization of a complex piece of software. [2]

Release Policies:

APIs are one of the most common ways technology companies integrate with each other. Those that provide and use APIs are considered as being members of a business ecosystem.

The main policies for releasing an API are: [3]

- *Private*: The API is for internal company use only.
- *Partner*: Only specific business partners can use the API. For example, car service companies such as Uber and Lyft allow approved third party developers to directly order rides from within their apps. This allows the companies to exercise quality control by curating which apps have access to the API, and provides them with an additional revenue stream.
- *Public*: The API is available for use by the public. For example, Zomato makes a part of their API public, and many other public APIs are available which can be under licences for use.

2.1.1 Web APIs [2]

Web APIs are the defined interfaces through which interactions happen between an enterprise and applications that use its assets. An API approach is an architectural approach that revolves around providing programmable interfaces to a set of services to different applications serving different types of consumers.

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition

of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.
While "web API" historically has been virtually synonymous for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct *Representational State Transfer (REST)* style web resources and *Resource-Oriented Architecture (ROA)*.
Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups. In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

2.2 SDK [7]

A **Software Development Kit** (SDK or devkit) is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To enrich applications with advanced functionalities, advertisements, push notifications and more, most app developers implement specific software development kits. Some SDKs are critical for developing an iOS/Android app. For example, the development of an Android application requires an SDK with Java, for iOS apps iOS SDK with

Swift, and for MS Windows the .NET Framework SDK with .NET. There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include Google, InMobi and Facebook.

It may be something as simple as the implementation of one or more application programming interfaces (APIs) in the form of some libraries to interface to a particular programming language or to include sophisticated hardware that can communicate with a particular embedded system. Common tools include debugging facilities and other utilities, often presented in an integrated development environment (IDE). SDKs also frequently include sample code and supporting technical notes or other supporting documentation to help clarify points made by the primary reference material.

CHAPTER 3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction

This document is a Software Requirement Specification (SRS) for PAWS, an application service to help give services to people at one place. The document is prepared as per the standard IEEE standard for Software Requirement Specification.

3.1.1 Purpose

The purpose of this document is to present a detailed description of PAWS. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. Through this document, the workload needed for development, validation and verification will ease. To be specific, this document is going to describe functionality, external interfaces, performance, attributes and the design constraints of the system which is going to be developed.

3.1.2 Scope

This service system will enable us to provide simplified application interface for variety of complex services. This system will be designed to remove the hassle of handling so many service oriented apps. It can certainly help those who do not posses very little technological proficiency. With the help of this service people will be able to get everything at one place with just a few chat messages in Natural Language.

3.1.3 Technologies Used

Programming Languages:

1. **Python:** [8]

   Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

2. **Node.JS:** [9]

   Node.js is a JavaScript runtime built on *Chrome's V8 JavaScript engine*. Node.js uses an *event-driven*, *non-blocking I/O* model that makes it lightweight and efficient.

   Threading in Node:

   Node.js operates on a single thread, using *non-blocking I/O calls*, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching. The design of sharing a single thread between all the requests that uses the observer pattern is intended for building highly concurrent applications, where any function performing I/O must use a callback. In order to accommodate the single-threaded event loop, Node.js utilizes the *libuv library* that in turn uses a *fixed-sized threadpool* that is responsible for some of the non-blocking asynchronous I/O operations.

Package Management in Node:

NPM is the pre-installed package manager for the Node.js server platform. It has tons of packages available for integration. It is used to install Node.js programs from the npm registry, organizing the installation and management of third-party Node.js programs. It is not used to load code; instead, it is used to install code and manage code dependencies from the command line. The packages found in the npm registry can range from simple helper libraries like Underscore.js to task runners like Grunt.

Event loop in Node:

Node.js registers itself with the operating system in order to be notified when a connection is made, and the operating system will issue a callback. Within the Node.js runtime, each *connection is a small heap allocation*. Traditionally, relatively heavyweight OS processes or threads handled each connection. Node.js uses an event loop for scalability, instead of processes or threads. In contrast to other event-driven servers, Node.js's event loop does not need to be called explicitly. Instead *callbacks are defined*, and the server automatically enters the event loop at the end of the callback definition. Node.js exits the event loop when there are no further callbacks to be performed.

3. Swift: [25]

Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. for iOS, macOS, watchOS, tvOS, and Linux. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of extant Objective-C (ObjC) code written for Apple products. It is built with the open source LLVM compiler framework and has been included in Xcode since version 6. On platforms other than Linux, it uses the Objective-C runtime library which allows C, Objective-C, C++ and Swift code to run within one program.

Swift is intended to be more resilient to erroneous code ("safer") than Objective-C, and more concise. However it supports many core concepts that are associated with Objective-C; notably dynamic dispatch, widespread late binding, extensible programming and similar features. For safety, Swift helps address common programming errors like null pointers, and provides syntactic sugar to avoid the pyramid of doom that can otherwise result. More fundamentally, Swift adds the concept of protocol extensibility, an extensibility system that can be applied to types, structs and classes. Apple promotes this as a real change in programming paradigms they term "protocol-oriented programming".

Frameworks Used:

1. **Express.JS: [10]**

   Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.

   It is the de facto standard server framework for Node.

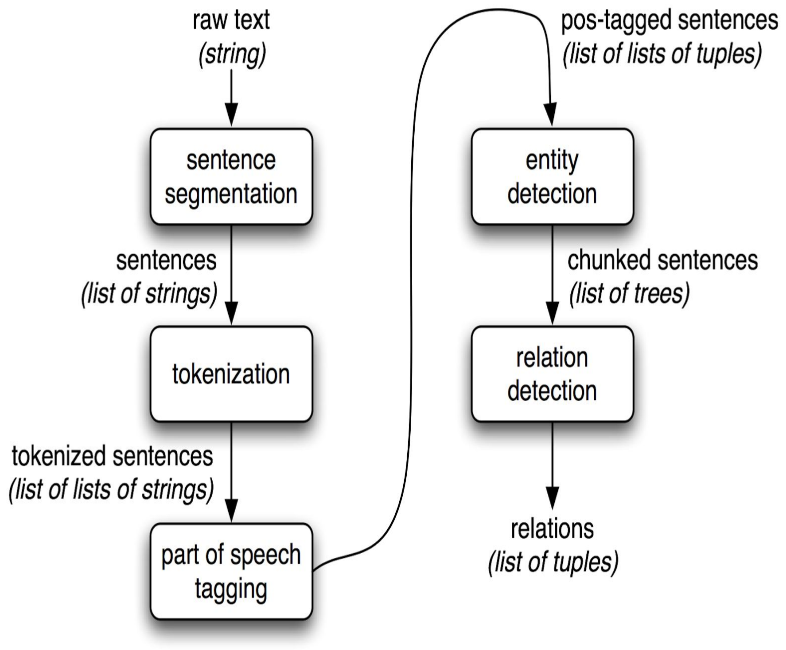
Others:

1. **NLTK: [11]**

   NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

Information Extraction

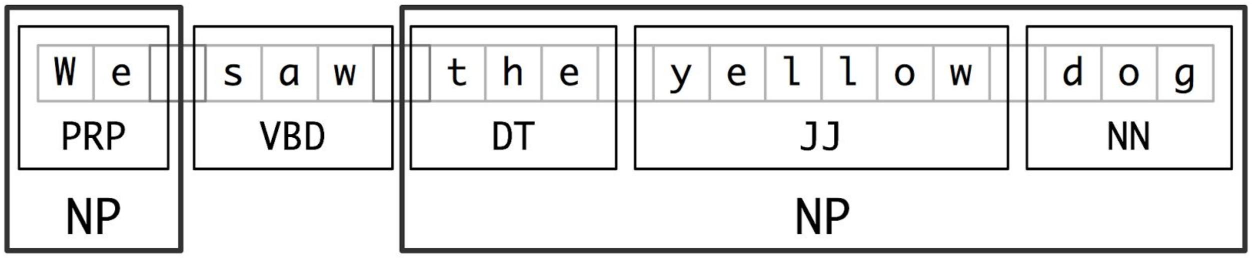
Information comes in many shapes and sizes. One important form is



structured data, where there is a regular and predictable organization of entities and relationships. For example, we might be interested in the relation between companies and locations. Given a particular company, we would like to be able to identify the locations where it does business; conversely, given a location, we would like to discover which companies do business in that location.

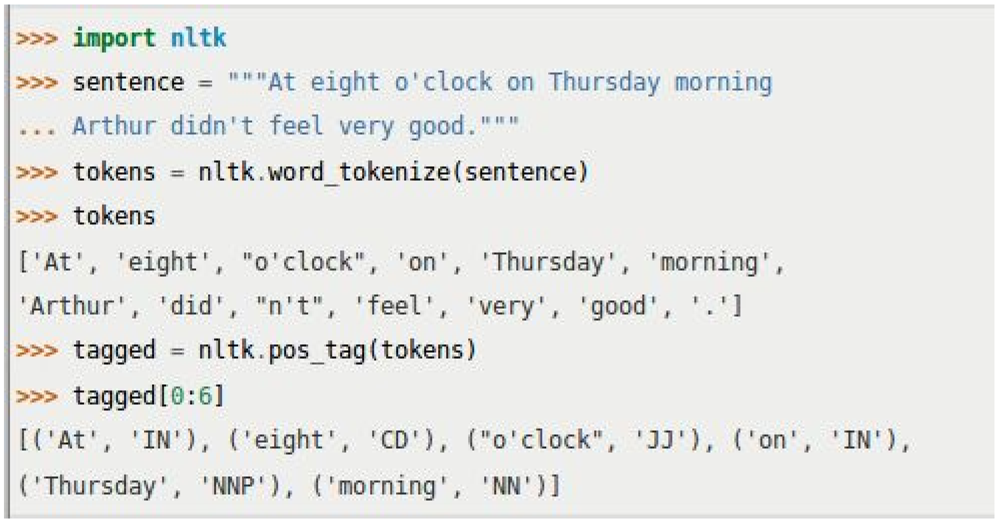
Chunking

The basic technique we will use for entity detection is **chunking**, which segments and labels multi-token.



The smaller boxes show the word-level tokenization and part-of-speech tagging, while the large boxes show higher-level chunking. Each of these larger boxes is called a **chunk**. Like tokenization, which omits whitespace, chunking usually selects a subset of the tokens. Also like tokenization, the pieces produced by a chunker do not overlap in the source text.

Tokenizing and tagging some text:



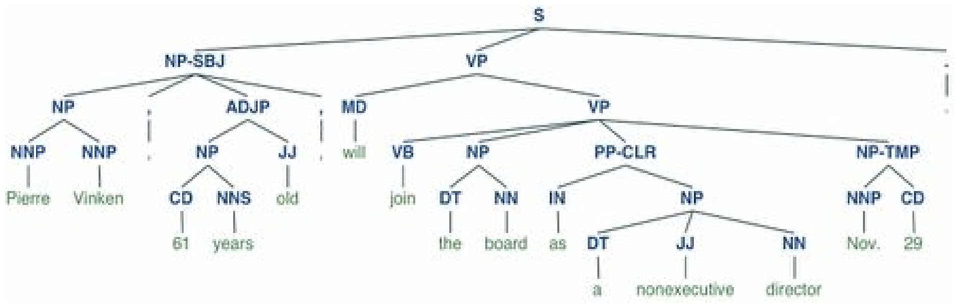
```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Identifying named entities:

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
       Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Displaying parse tree:

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```



2. Alamofire: [12]

Alamofire is an HTTP networking library written in Swift.

3. Neo4j: [13]

Neo4j is a graph database management system developed by Neo Technology, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing, Neo4j is the most popular graph database according to db-engines.com.

Neo4j is implemented in Java and accessible from software written in other languages using the Cypher Query Language through a transactional HTTP endpoint, or through the binary 'bolt' protocol.

In Neo4j, everything is stored in the form of either an edge, a node, or an attribute. Each node and edge can have any number of attributes. Both the nodes and edges can be labelled. Labels can be used to narrow searches.

4. Ngrock: [14]

Ngrok is a service with the help of which you can set up local website behind NAT or private network and make it available to the general public to use. You can run webmail, file syncing, and more securely on your hardware with full end-to-end encryption.

We can also build webhooks with ease. Webhook integration requires public address and a lot of setup to trigger hooks. Ngrok makes the process quite simple. It also gives interface to inspect the HTTP traffic flowing over your tunnel.

5. iOS SDK: [15]

The iOS SDK (Software Development Kit) (formerly iPhone SDK) is a software development kit developed by Apple Inc. and released in February 2008 to develop native applications for iOS.

The SDK was released on March 6, 2008, and allows developers to make applications for the iPhone and iPod Touch, as well as test them in an "iPhone simulator". However, loading an application onto the devices is only possible after paying an iOS Developer Program fee. Since the release of Xcode 3.1, Xcode is the development environment for the iOS SDK. iPhone applications, like macOS applications, are written in Swift and Objective-C, with some elements of an application able to be written in C or C++.

6. Postman: [16]

A powerful GUI platform to make your API development faster & easier, from building API requests through testing, documentation and sharing.

3.2 Overall Description

This section gives background information about specific requirements of PAWS to be developed. It will describe the factors that affect the final product. Although we will not describe every requirement in detail, this section will describe the factors that affect the final product.

PAWS has one active actor and one system. The end-user or the target user performs requests using chatbot or voice , which act as the input to the system. This communication between the user and the system is through a microphone or text messaging. The system processes these requests and services them accordingly.

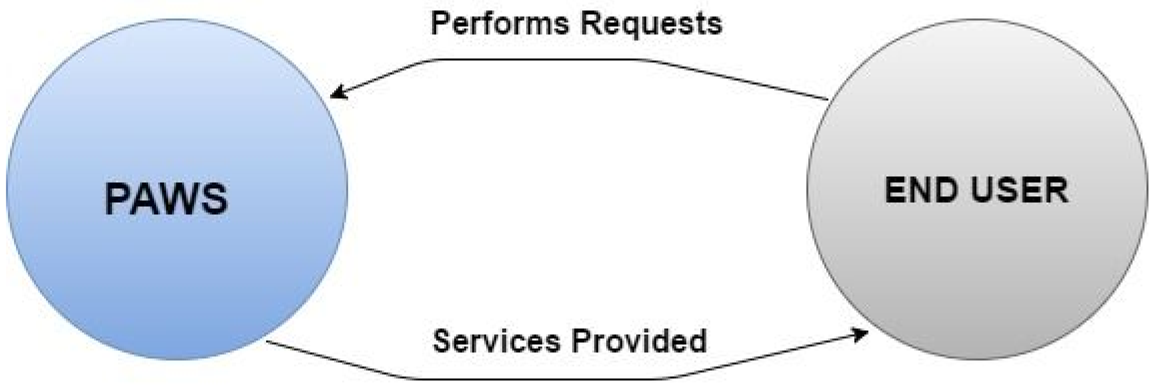


Fig 4: Interaction between user and PAWS

3.2.1 Product Perspective

This software product is eventually intended to help the smartphone users to access the services with ease. The system increases the relative ease with which the target audience can use the system as compared to their current experience with using a smartphone. The end-user will be able to select and access the various services using either a voice input or a chatbot system. The main advantage being that the instructions provided to the system can be simple command sentences in Natural Language.

3.2.2 Product Functions

This system allows users to use functionalities which have been explained above in the introduction. Required functionalities of the product can be summarized in five categories; server side functions, client side functions, API integration functions, Message parsing functions and interface functions. Overall description of the functions
can be found below:

1. **Rootfinder**: It checks whether the provided request enables the search process to enter the root node or not.

2. **CompanyFinder**: After entering the root node, the words from the request are analysed to search for the specific API service provider for which the user has requested the service.

3. **ServiceFinder**: A single API service provider can have multiple services, in order to distinguish between all these services we us the ServiceFinder functionality, which compares the user request with available services for a service provider.

4. **ParameterFilling**: The user is iteratively asked to fill in the details required for a REST API request using this function.

The following APIs are currently supported by the app:

- **Zomato**: [17] Zomato APIs give us access to the most exhaustive information for over 1.5 million restaurants across 10,000 cities globally.
  With the Zomato APIs, the user can :
  - Display detailed information including ratings, location and cuisine
  - Book a table at a restaurant.
  - Search for restaurants by name, cuisine, or location

- **Wikipedia:** [18] The MediaWiki action API is a web service that provides convenient access to wiki features, data, and metadata over HTTP. It provides direct, high-level access to the data contained in MediaWiki databases. Client programs can log in to a wiki, get data, and post changes automatically by making HTTP requests to the web service

- **Food2Fork:** [19] Food2Fork offers an API which exposes its ingredient search functionality across its database of publishers. The API gives you access to our ever expanding socially ranked recipe database and state-of-the-art ingredient search function

- **Markit:** [20] It is a pricing and valuation service focused on providing end-of-day services that aggregate valuation information on credit default swaps. The Markit Data APIs give developers access to the data collected by Markit. Currently, there are APIs available for Company Lookup and Stock Quotes

- **Open_Weather:** [21] This API is used to get the current weather and forecasts in a particular area.

- **Flipkart:** [23] This API is used for searching products for shopping and for browsing top products in a category

- **Bing News:** [24] This API provides us with latest news according to a search query or you can choose a variety of topics to get latest new on.
- **Directions:** [25] This api provides you direction to a specific place

3.2.3 User Characteristics

- Users of this system can be anyone, though it is targeted for smartphone users with internet connectivity.
- Target users have some application development knowledge to no knowledge at all.
- Low level of proficiency needed in handling smartphone.

3.2.4 Assumptions

- User has a working smartphone with internet connectivity.
- User has basic knowledge proficiency in English.

3.3 Specific Requirements

With this section and later, we will describe the requirements of the product in detail. Basically, we will categorize requirements in three which are namely external interface requirements, functional requirements and non-functional requirements. Except non-functional requirements, requirements of the product will be detailed under this section with brief information.

3.3.1 External Interface Requirements

- **User Interface Requirements**: User should have a working smartphone with working touch screen. User should be able to type instructions in in his phone.
- **Hardware Interface Requirements**: An active microphone, screen, network connectivity, GPS tracking other necessary hardware which is present in the smartphone.
- **Communication Interface Requirements**: A working internet connection is necessary for the smooth operation of the application.

3.3.2 Functional Requirements

1. **PAWS using Keyboard Input**

   Input: The user specifies the type of service required by him/her through text to the application.

   Output: The equivalent display of services is provided on the user's screen and prompted to make his/her decision.

   Eg.:

   1) For Zomato API:

      Input: Cuisine Type, Radius in which it wants to search.

      Output: List of restaurants matching query

3.3.3 Performance Requirements

- Generated application should be responsive in design.
- The application should be able to guess the user's intent to a certain degree of certainty.
- The application should be able to display the consolidated results obtained after a request to the REST API

3.3.4 Design Constraints

1) **Development Tools**

The system shall be built using NodeJS, Python, Swift, Xcode, Sublime Text, Postman, Ngrok.

2) **Database**

Neo4j graph database is used for storing information regarding various services. We used this for storing different services and functions flow.

3.3.5 Non Functional Requirements

1. **Usability** - The system shall be easy to use and understand. User will only need to give inputs to the application in the form of voice input or text. All other functions will be automated.
2. **Maintainability** – Component driven development and modular structure shall ensure maintainable code.
3. **Portability** - The end-user part is fully portable and any system should be able to use the features of the application, including any hardware platform that is available or will be available in the future.

3.4. Change Management Process

The SRS is developed in such a manner that any desired changes can be introduced by the designing party in the near future according to the suitability.

Chapter 4: Models

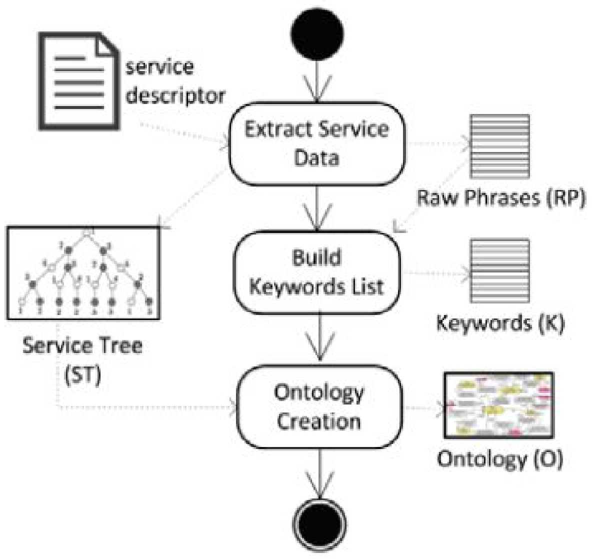
In this section we describe model and its working that has been used in the implementation of our service and app.

4.1 Model of the service [1]

Service-oriented architecture (SOA) is a proven approach that aims at producing loosely coupled, standard-based, and protocol-independent services. A compliant SOA architecture must provide independents units or services, allowing users to discover, execute and compose them in their applications. In order to follow the rapidly changing and highly competitive market, organizations have to adapt their service interfaces according to their business requirement. Thus, the design is one of the most crucial phases of service lifecycle. From the last years, the software engineering domain has made a great effort providing methods for agile development. Despite these efforts, we lack the necessary tools to use and validate the rich vocabulary presented in user stories and test cases during the service design.

4.1.1 Exploiting Service Interfaces

In this section we present a detailed description of our approach to exploit service interfaces using a personal assistant.



We could summarize it in three high-level processes:

(i) service registration,

(ii) service discovery and

(iii) service execution.

First, we present the architecture of the proposed solution. The subsequent sections describe each high-level process separately.

4.1.2 Service Registration

This process aims at building a semantic representation of the service, more specifically extracting relevant keywords and structural information of services using the descriptor as a source. Each step contributes to build a representation for service interface exploitation.

1) Extract Service Data:

This is the first step, which uses a service descriptor as an input parameter. The goals of this step are:

(i) extract service documentation; and

(ii) build a tree representing the service.

We believe that a minimum documentation and description of parameters are a kind of least common denominator for service descriptors. For example, for the WSDL specification, we use the documentation tag as a source. To retrieve the parameters, we use the input and output representation, described as XML Schema types. During (i), the process looks for service documentation in order to build a set of raw phrases RP = {rp1, rp2, ...rpn}. A raw phrase rp in this context is interpreted as the original phrase with spaces and stop words. During (ii), a service tree ST is generated for each service operation found. Finding the input and output structure is not a complex task, once the complex types are described with their respective types in service descriptors. The generated tree contains a representation of complex types found in the input and output parameters followed by the input and output branches and their respective attributes. To sum up, this step generates for each operation a set of raw phrases RP that will be used to create the keywords and a tree ST for creating a common representation and an automatic dialog.

2) Build Keywords List:

During this step we use the set of raw phrases RP = {rp1, rp2, ...rpn} as an input in order to build a list of keywords. To do so, we have used some known NLP techniques, as follows:

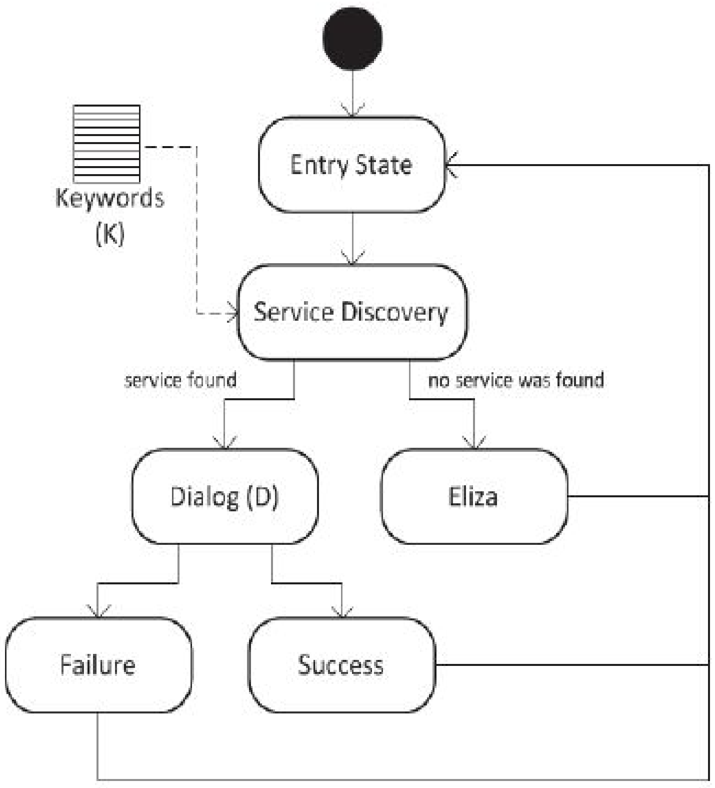
• Words are segmented and stopwords are removed for each raw phrase (rp);

• A Part of Speech algorithm (POS) is executed for each word, identifying nouns, verbs, adverbs and adjectives. This will be helpful when matching keywords during the dialog.

• A stemming algorithm is invoked, transforming the original words into root words. As a result a new set of phrases called P = {p1, p2, pn} is generated, where each p is a set of instances {(w1, t1), (w2, t2), ..., (wj, tj)}, w is the word, t is their respective type (noun, verb, adverb or adjective) and j is the quantity of words in the given phrase.

In short, at the end of this step, a set of keywords are generated for the service called K = {k1, k2, kn}, where each kn is a word and n is the number of keywords for the underlying service. In a previous work [9] we used a technique of word sense disambiguation (WSD) for keyword expansion, improving the task or service selection. For example, if an input parameter has the fragment "project risk" as a name, two new synonyms will be added: "project peril" and "project danger". We have decided to remove this step, since its usage could hide potential service design problems related to the vocabulary.

4.1.3 Service Discovery

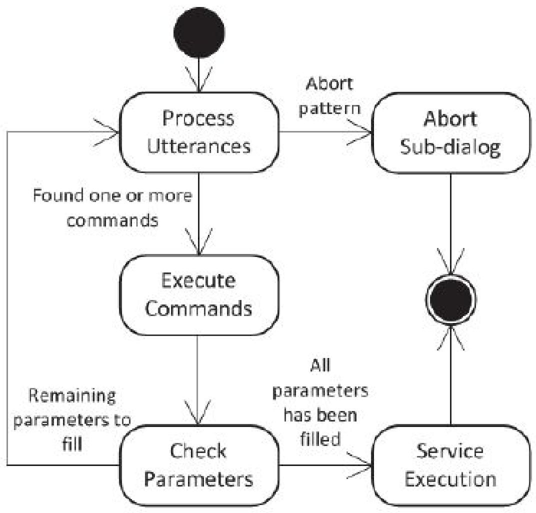


Service discovery is the process of finding a suitable service for a given request, and our selection approach is based on linguistic cues collected by our NLP service.

When the top-level dialog identifies a suitable service to attend the user, a nested sub-dialog is then created. Giving a more formal definition, a sub-dialog is a conversation graph, having a set of nodes representing states of the conversation. Each node contains a set of rules that apply to a fact base containing information obtained from the master's input or resulting from the analysis steps of the previous states. The fact base is similar to the fact base of a rule-based system. At a given node, applying the rules triggers either a transition to a new state or an action.

4.1.4 Service Execution

Once the top-level dialog has identified the target service to execute, a nested sub-dialog associated with the service is instantiated and started. The goal of this step is to build an automatic sub-dialog based on the service ontology. The ontology will be used as a source for the information extraction and the collected parameters' definition play a key role during the extraction since they indicate the presence of relevant information in the user text. The system will ignore the information typed without the corresponding keyword since our objective here is the evaluation of the expressiveness of the service, more specifically related to the choice of vocabulary.

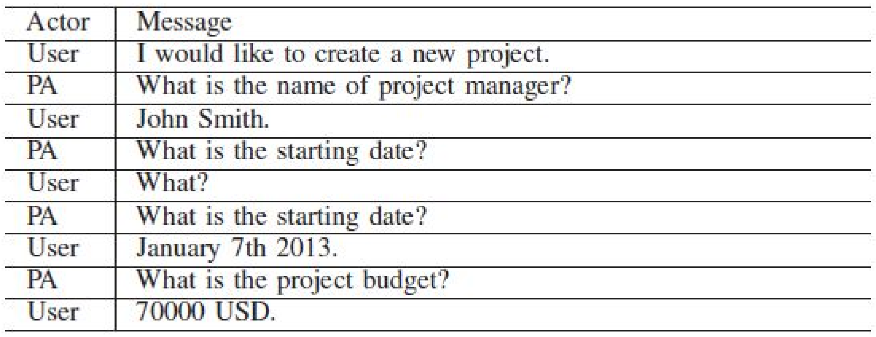


The user can ask everything at once like:

Actor	Message
User	I would like to create a new project. The project manager is John Smith, the initial budget is 70000 USD and the starting date is January 7th, 2013.

Fig. 5

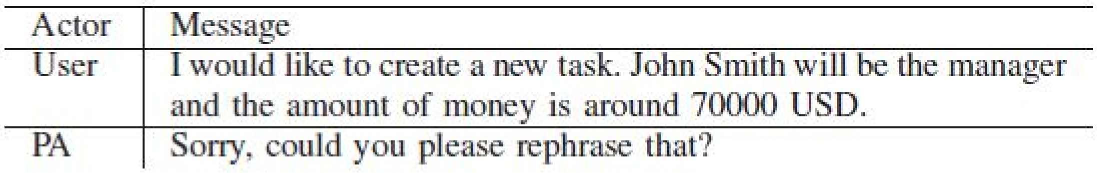
Or he can converse with our service, like:



Actor	Message
User	I would like to create a new project.
PA	What is the name of project manager?
User	John Smith.
PA	What is the starting date?
User	What?
PA	What is the starting date?
User	January 7th 2013.
PA	What is the project budget?
User	70000 USD.

Fig. 6

An unrecognized service:



Actor	Message
User	I would like to create a new task. John Smith will be the manager and the amount of money is around 70000 USD.
PA	Sorry, could you please rephrase that?

Fig. 7

4.2 Working of the model

This chapter is divided into two sections, one for server side implementation and other for application implementation.

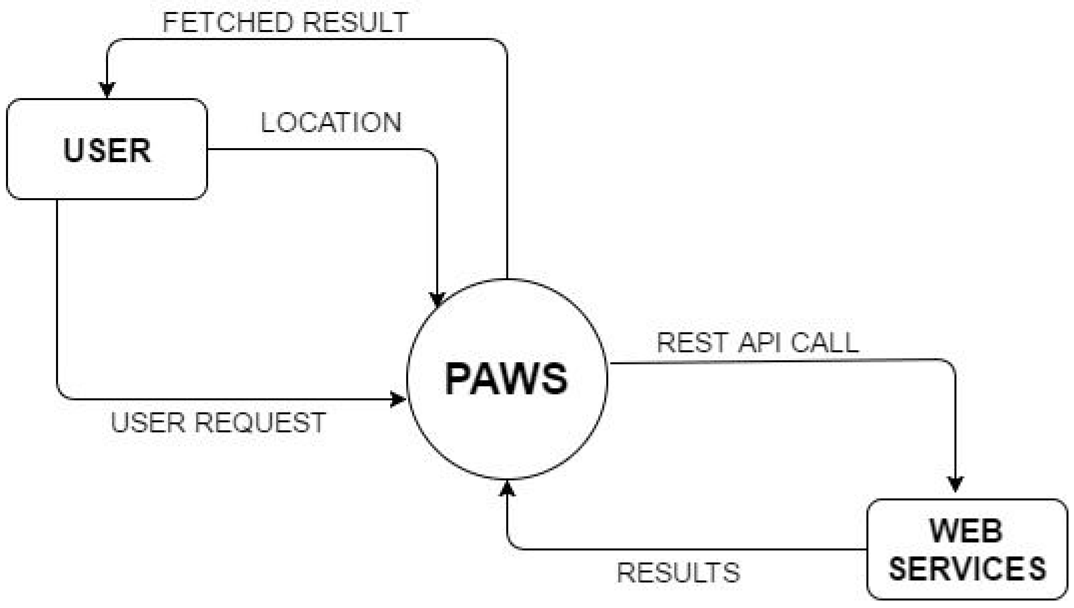


Fig. 7: Context diagram of System

4.2.1 Server Side Implementation

4.2.1.1 Introduction

PAWS's server is the workstation of our service system. It is the place where all the computation and modification of data is done, and all the request to the API's are made from here only and the application is then given the resulting message or the final output.

The user sends its query to the server. When server receives the query, it extracts useful information from it and parses a reply or final output to send back to the user.

4.2.1.2 Platform Used

PAWS's server is a Linux based desktop operating system, currently. In future it can be implemented on any linux server systems available. To transform our system into a server we used Ngrok to broadcast our localhost to everyone.
PAWS server platform has two main components having different functionality:

1. Main Server, for handling requests and parsing output,
2. Python modules for Natural Language Processing

Main server is powered by NodeJS and its framework ExpressJS for handling requests to the server and keeping a session and other NodeJs functions for calling python modules and parsing messages.

The python modules uses NLTK to extract information from the message.

The platform should have these modules pre installed:

- NLTK module,
- Stanford parser module pre downloaded,
- Neo4j and its dependencies preinstalled,
- nltk_data downloaded from NLTK downloader

Both components work hand in hand and continuously to serve every request to give an output result as soon as possible.

4.2.1.3 Technical Approach

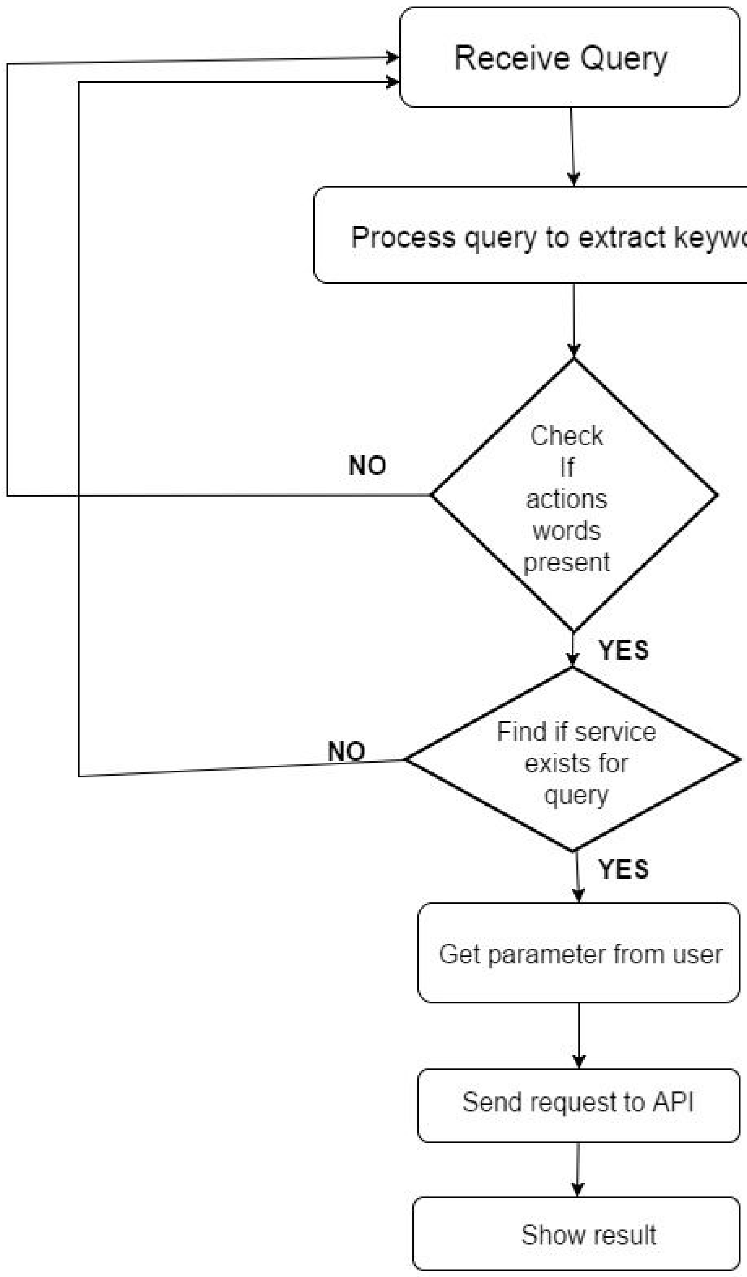
- When the server receives a request from a user. If the user is sending request for the first time, which he sends at a different end point (eg. at '/startSession'), an id is assigned to him, and an object containing his id and all the necessary variables are made which will be necessary in the future to maintain user data and identify that specific user.

- If the user is has already sent a starting request then his object with the id should already in the server database and the user can send a message request to the server at a different end point (eg. at '/requestMessage'). This request should contain the session id and the message or query.

- After receiving the message, the python module implementing NLTK is called to extract information from the message which is checked for actions words. If those action words are present then message is checked again for information about service or what function of that service does a user want to use based on keywords.

- If the service or function exists for the user then, user is prompted for parameters of that specific service or function and then a request is sent to the specific API endpoint to get the result.

- The result from the API is parsed and output message is sent to the user.


Receive Query
Process query to extract keywords
Check If actions words present
NO
YES
Find if service exists for query
NO
YES
Get parameter from user
Send request to API
Show result

Flowchart describing server side implementation

Fig. 8

4.2.1.4 Limitations

Even though we have integrated many services and tried to make it as feasible as possible, there are still many ways it can improve. Some of the limitations are:

- It understands only one language,
- It is a mobile based service only, in future it can be implemented as a WebService too,
- It can have more in depth intent extraction, to remove more possible errors

4.2.2 Client Side Implementation

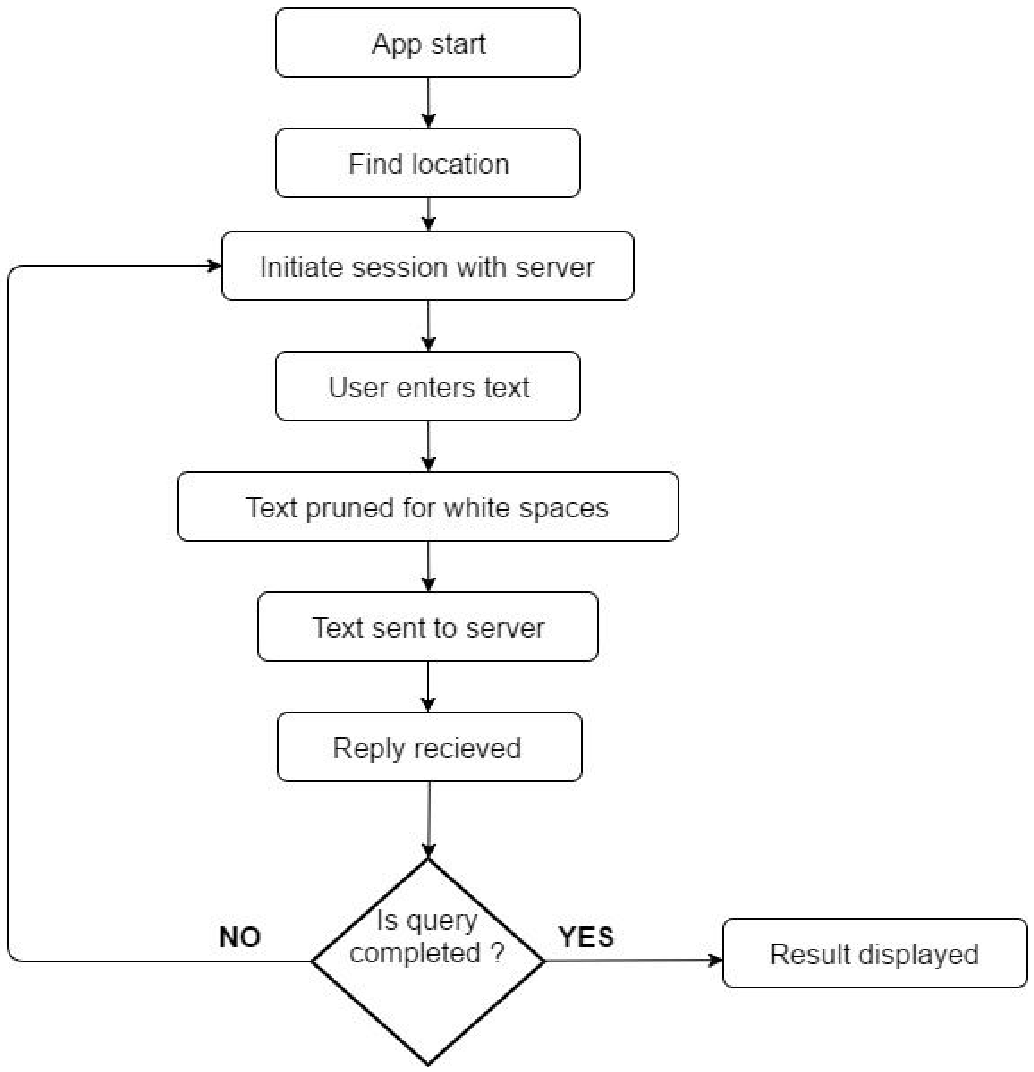


Fig. 9: Flowchart describing client side implementation

4.2.2.1 Introduction

The client side is built as an iOS application. The client app is compatible with iOS sdk 9.0.
During the startup phase of the app location is obtained using onboard GPS device. This along with session initiation request is sent to the server.
After this the user sends his queries in form of chat message. This message gets sent to the server and gets added to a message cell in UITableView. This message gets parsed by the server and in case the query is complete the result is received from server and displayed to the user or else further questioning is initiated by the server to get to the correct query.

4.2.2.2 Platform Used

The app is built using Swift programming language. The application is executed in a custom UNIX shell inside the iphone. We are using UIKit for designing the user interface of the app, Alamofire is used for the networking part of the app, it is used for sending get and post http requests to the server. Further SwiftySON (for parsing JSON) and SwiftSpinner (for displaying loading messages) were used as supporting modules.
The UI was designed using storyboard in xcode and for binding the view to the model Cocoa Touch Classes were used.
In order to build the app you would require Xcode and a simulator.

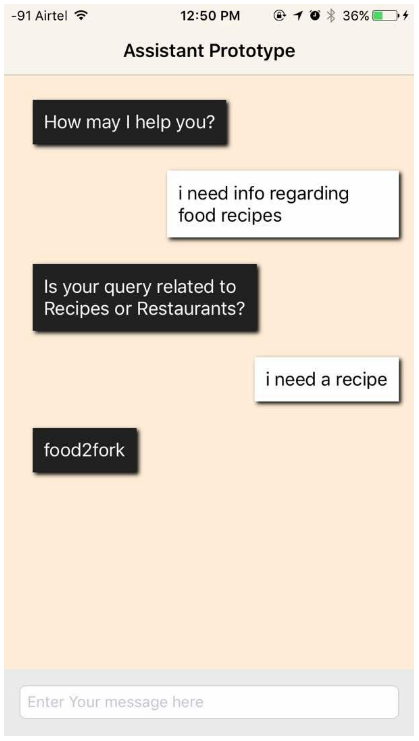
4.2.2.3 Technical Approach

1. When the app launches immediately the location of the phone is detected. For this a prompt is displayed to the user asking his permission to use his location.
2. An initiation request is sent to the server along with the location coordinates to start a new session, while this is being done the user is presented with a session loading message as a Modal.
3. After initiation the user can send a message to the server. According to the message if it is valid or invalid a reply is received from the the server and displayed to the user.
4. If a query is complete the server responds with a result JSON object. This result is then displayed on the app in a different screen.
5. After the result is shown the session i reset again.

Chapter 5. Results and Conclusion

- The system was successfully implemented in schedule, meeting the requirements in the SRS.

- We were able to integrate many APIs such as google maps, zomato, open weather, markit, wikipedia, google translate etc and were successful in getting results.



How may I help you?

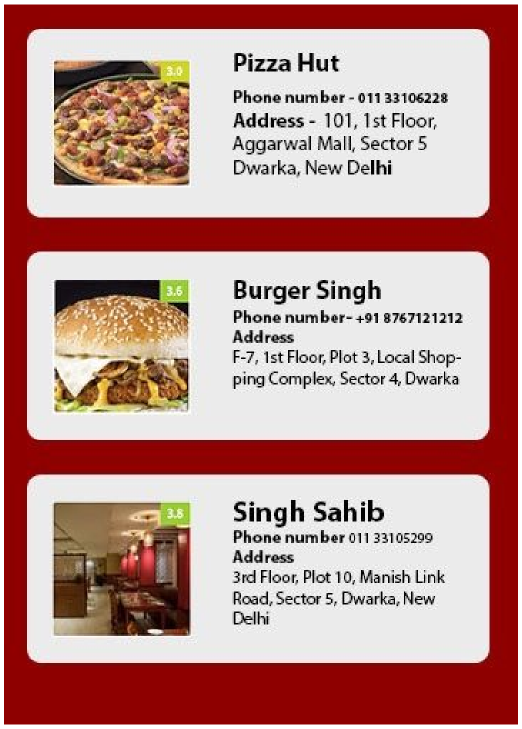
i need info regarding food recipes

Is your query related to Recipes or Restaurants?

i need a recipe

food2fork

Fig. 10: Sample Conversation

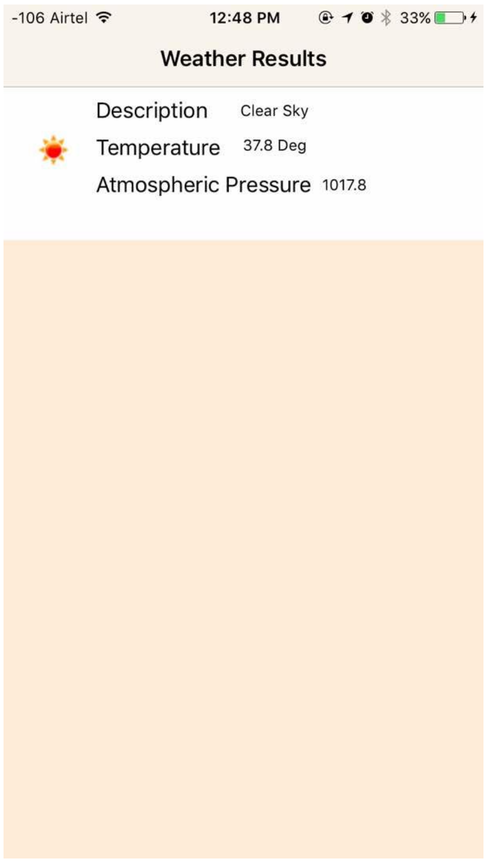


Pizza Hut
Phone number - 011 33106228
Address - 101, 1st Floor, Aggarwal Mall, Sector 5 Dwarka, New Delhi

Burger Singh
Phone number- +91 8767121212
Address
F-7, 1st Floor, Plot 3, Local Shopping Complex, Sector 4, Dwarka

Singh Sahib
Phone number 011 33105299
Address
3rd Floor, Plot 10, Manish Link Road, Sector 5, Dwarka, New Delhi

Fig. 11: Sample Output 1



Weather Results

Description	Clear Sky
Temperature	37.8 Deg
Atmospheric Pressure	1017.8

Fig. 12: Sample Output 2

- We were able to implement it without any considerable lag considering libraries used.

- We saw a considerable amount of time not being wasted which could have been seen due to tons of heavy apps.

The work proposed in this thesis based on the paper whose description is provided in the Models chapter, an approach for the exploitation of services using a personal assistant.

Its development is based on multidisciplinary techniques that come from different domains such as multi-agent systems, knowledge engineering and information extraction. It presents a discovery mechanism based on keywords extracted from service descriptors and a fulfillment process using natural language. Users validate service interfaces without previous knowledge about the location and structure of the service. The approach leverages the cooperative work between software designers and domain experts, allowing them to identify the expressiveness of their services taking into account the granularity, the vocabulary and the effort to execute the service. Metrics are generated for each session, which could be conducted by real users or even by daemons running predefined test cases. They could be used as a source for iterative process improvements.

Chapter 6. Future Scope

Our project has a lot of potential for future work. Though there are still many shortcomings in our project, we can still use it in its present forms in a lot of applications. PAWS can still integrate with many Web APIs to make itself even more powerful and diverse. There are many services and many apps for the same service. We can integrate them so that user can select services of their preference.

In future, we can use Machine Learning, so that the app will able to create a profile of the user of their interests and better predict the most compatible choice for that profile. This will help users to get results of their interests.

For better security we can add Face Detection mechanism and maintain it in the profile, so that user will be more sure of the service and have a easy and secure experience.

For better user understandability we can add suggestions to the search query to help him understand and guide him.

We can make it more secure by keeping user profile in a more secure database such as MySQL, etc.

Chapter 7: References

1. Márcio FUCKNER, Jean-Paul BARTHÈS and Edson Emílio SCALABRIN - “Using a Personal Assistant for Exploiting Service Interfaces” - IEEE 18th International Conference 2014
2. https://en.wikipedia.org/wiki/Application_programming_interface
3. Boyd, Mark. "Private, Partner or Public: Which API Strategy Is Best For Business?". *ProgrammableWeb*.
4. https://www.statista.com/topics/1002/mobile-app-usage/
5. http://www.businessinsider.com/bii-mobile-insights-sept-182-2012-9?IR=T
6. http://info.localytics.com/blog/time-in-app-increases-by-21-across-all-apps
7. https://en.wikipedia.org/wiki/Software_development_kit, SDK wiki page
8. https://en.wikipedia.org/wiki/Python_(programming_language), Python wiki page
9. https://en.wikipedia.org/wiki/Node.js, Node.JS wiki page
10. https://en.wikipedia.org/wiki/Express.js, ExpressJS wiki page
11. http://www.nltk.org/, NLTK main website
12. https://github.com/Alamofire/Alamofire, Alamofire github page
13. https://en.wikipedia.org/wiki/Neo4j, Neo4j wiki page
14. https://ngrok.com/, Ngrock main website
15. https://en.wikipedia.org/wiki/IOS_SDK, iOS wiki page
16. https://www.getpostman.com/, Postman official page
17. https://en.wikipedia.org/wiki/Zomato, Zomato wiki page
18. https://www.mediawiki.org/wiki/API:Main_page, MediaWiki Official page
19. http://food2fork.com/about/api, Food2Fork official API page

20. http://dev.markitondemand.com/MODApis/Api/Doc, Markit API official page
21. https://openweathermap.org/api, OpenWeather API official page
22. https://affiliate.flipkart.com/api-docs/affiliate_index.html, Flipkart API official page
23. https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/, Bing News API official page
24. https://cloud.google.com/apis/, Google Apis official page
25. https://en.wikipedia.org/wiki/Swift_(programming_language), Swift wiki page