UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
University of Pretoria

Software Engineering
COS301 Main Project

Team CodeX

ReRoute Systems

Bondjobo, Jocelyn 13232852
Malangu, Daniel 13315120
Kirker, Tim 11152402
Hammond, Eunice NK 13222563
Burgers, Heinrich 15059538

Contents

1 Introduction

1.1 Background

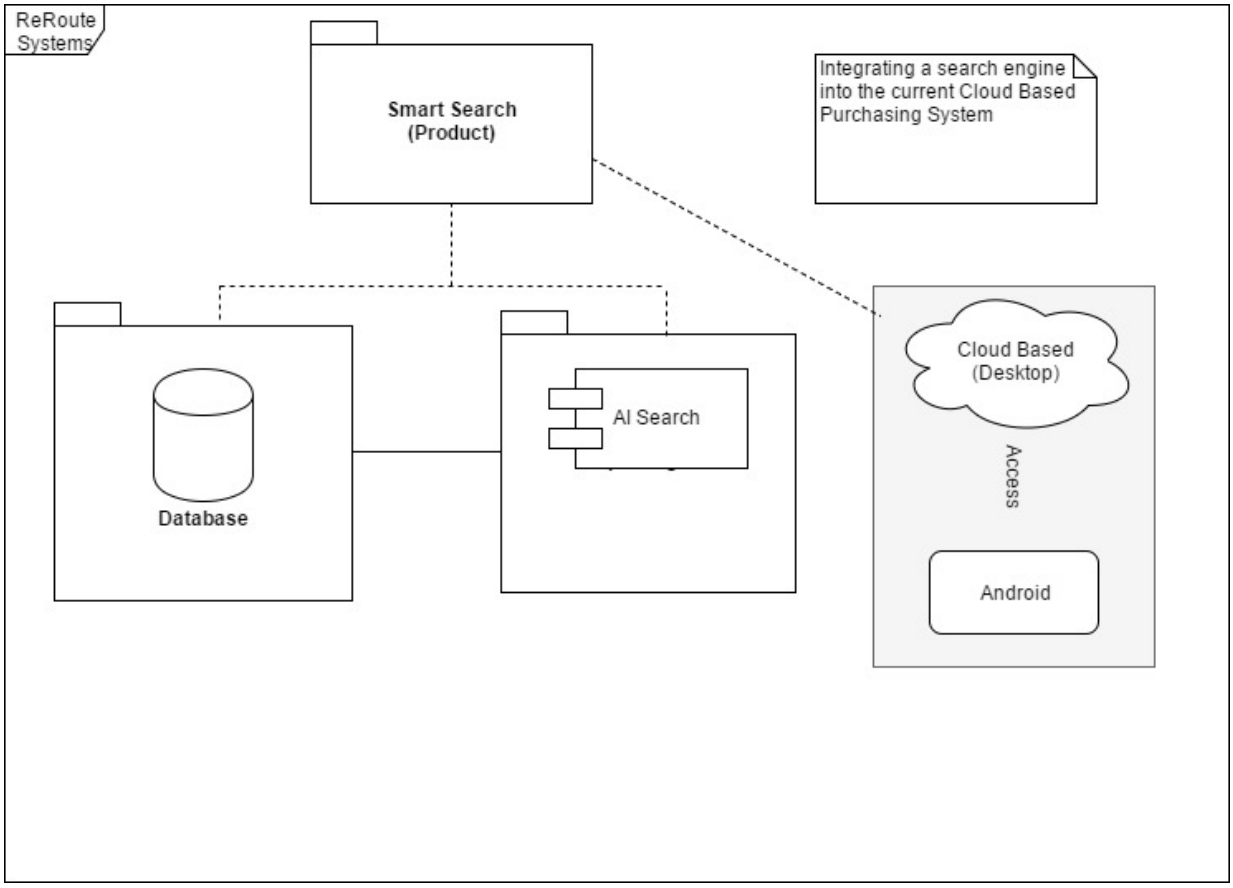
Reroute Systems is a software company with different in-house developed applications. The Purchase Management System application is the main application and is mainly active in the pharmaceutical space. The main functionality of this system is routing of requests for products from account holders to various wholesalers/suppliers, receiving the result of the order and routing the answer back to account holder.

1.2 Purpose

For an account holder to request a product, s/he must search for it against the database with all the product information. The challenge is that each wholesaler/supplier can name/describe the product in a different way/manner, and when the account holder performs the search against the master product list, the same product must be displayed across all wholesalers/suppliers, using the link between master product list and the different wholesaler's/supplier's product list.

1.3 Scope

The core of the system is a search engine which is enriched with functionality of performing some machine learning in order to retrieve the product in a master file found in a database running on the server through RESTful calls. The high level modules and their responsibilities are shown in the figure below.



1.4 Definitions, Acronyms, and Abbreviations

- Cloud-based: Refers to applications, services or resources made available to users on demand via the Internet from a cloud computing provider's servers.
- Android: A Linux based operating system developed by Google Inc. and the Open Handset Alliance for smart phones, tablets, and other mobile computing devices. Android applications are developed in Java.

- Smart Search: The use of Artificial Intelligence to search the database by implementing machine learning techniques and other algorithms to find and retrieve data.
- REST: Representational State Transfer. It is a architectural pattern consisting of guidelines and best practices for creating scalable web services.
- HTTP: HyperText Transfer Protocol, standardised by RFCs 7230-7237.

1.5 References

(To be completed)

1.6 Stakeholders

The stakeholders of the syste includes the following

- **The Client** is Diederik Mostert, Software Director of ReRoute Systems who proposed the project as a result of a challenge faced in the company, which is to search a product on master file with different combination used by each wholesaler.
- **Users in Pharmaceutical space** They will use the system to retrieve information of the product they need.

2 Overall Description

2.1 Product Perspective

2.1.1 System Interface

1. User Interface:

    - Functionality:
      The functionality of the user interface is to allow the user to interact with the system. Through this aspect of the system the user can perform all the functions necessary to use the smart search of the system. Through this, the user is able to specify the search criteria to be used by the system.

    - How functionality achieves requirements:
      The user interface allows the functionality of the application to meet the requirements. The user is able to search for any product based on any value they enter in, such as the product name or product ID. The user is able to view displayed information of products, that is based closely on keywords and values or any thing resembling the name.

2. Hardware Interface:

    - Functionality:
      The functionality of this system interface is the physical hardware that allows software operation. The hardware in this case is specifically the desktop which allows the user to run the system.

    - How functionality achieves requirements:
      This hardware will allow the achievement of the requirements by facilitating the operation of the application. The networking hardware of the components such as the Wireless Fidelity infrastructure or LAN connections between the devices which will allow for communication to the system server.

3. Software Interface:

    - Functionality:
      The functionality of the software interface is to facilitate the communication between the hardware infrastructure of the device(s) running the application's front-end and the Reroute Server. An example of this would be the operating system of the device in use. The operating system would allow the application to make use of the networking infrastructure and communicate the necessary information to the application via RESTful calls. This would enable the device to communicate with the server and perform the necessary requests.

    - How functionality achieves requirements :
      This would allow the achievement of the requirements, as the communication of the application to the system server is vital to the operation of the Smart Search. By allowing communication over WI-Fi, the user is able to communicate to the server from any location or just stick to older and more familar devices such as computers, fulfilling the requirement.

2.1.2 User Interface

All users will be able use the Smart Search system from their desktop devices when the application is launched. For the account user interface, the registered users will be able to login to their account, using their account information.

2.1.3 Communications Interface

- Users using the web will use HTTP/HTTPS protocol.
- RESTful calls from the differrent platform to the server API.
- Cloud-based server running the database.

2.1.4 Memory

The desktop application will use a minimum amount of space of roughly 300MB.
Primary memory(RAM) will use 50MB on average, based on how long the application is being used.

2.1.5 Operations

The user will require a series of usual and special case operations to be fulfilled by the system. The user will use the system in normal case operations that comprise of searching on any of the different properties of a product. On a usual case, the user will search for a product based on the name of the product of interest.

The system will provide a user the option to either save their data, such as variations of product names, locally on the device or remotely on the server.

2.1.6 Sit Adaption Requirements

(to be completed)

2.2 Product Functions

The main function of the Smart Search will be to retrieve product information based on the values given to the system. The basic functions will include:

- A search based on a specific property of a product such as the name or ID.

2.3 User Characteristics

The general user will use the system to retrieve product information. They will require minimal skills in order to utilise the application, including the ability to connect to the Wi-Fi and use it. They will require background knowledge/education in the medical or pharmaceutical field to use the application.

2.4 Constraints

The system will

- only use the data contained in the client's existing database.
- learn and evolve from user input (Machine learning). The system is subjected to working with an Internet Connection due to the linkage with the cloud. Without this in place, the system will not function.

2.5 Assumptions and Dependencies

Users will

- have some form of pharmaceutical knowledge.
- have basic computer skills.
- be connected to an internet connection.

The database will be updated regularly.

3 Architecture Requirements

This section discusses the requirements that surround the software infrastructure

3.1 Access channel requirements

3.1.1 Human Access Channels

The human access element must allow for desktop web client which will allow the user to access the Smart Search. This element will be based on the client side of the two-tier Client-Server architectural pattern.

3.1.2 System Access Channels

All system access will be made over a uniform interface which will assist in faster development time, ease of maintenance and lower overall cost associated with the system. The common interface that will be used will be a RESTful API. All client-side interaction will be done via the User Interface which will in turn use the RESTful API set of the system to accomplish the given task.

3.2 Architecture Constraints

- Representational State Transfer (REST) will be used. It is an architectural style used to design networked applications, and depends on cacheable and stateless client-server and communications protocols, using the HTTP protocol in most cases. All four CRUD operations are used in REST.
- Service Oriented Architecture (SOA). The concept of service is based on SOA. SOA makes it easier for software components on computers that are over a network to work together with no need to make changes to the underlying program itself.
- Client-Server Architecture will also be used. This architecture is a model which the server hosts, delivers and manages the resources and services to be consumed by the client. In this case, the database performs the above with the resources. A two-tier client-server architecture is in play.
- Unit Testing to automate the following tasks: Compilation of Java code, running test cases and generating product documentation.

3.3 Architectural Patterns

3.3.1 Client-Server Architectural Pattern:

The client server architecture is a network technology in which each computer connected on the network is either a server or client.

- Server(s): are powerful computers which are used to process the client requests.
- Client: is simply a computer connected on the network which is used to send request for some resources to the server.
- Requirement of a client-server: It models work in a networked environment therefore the processing of an application distributed between the client and the server.
- Function of client server: An interface/form is provided by the client to allow a user to request services of the user and to display the results the server returns.

3.3.2 Representational State Transfer (REST) Architectural Pattern:

REST is an architectural pattern for designing networked applications. Instead of using complex mechanisms such as SOAP to connect between machines or to make calls between machines, simple HTTP is used.

The Uniform interfaces is combined of resource names and HTTP, GET, DELETE, POST, PUT. A REST service is:

- Language-independent (C-sharp can communicate to Java, etc.),
- Can be used with firewalls,
- Platform-independent (MAC, Windows, UNIX) and
- Standards-based (runs on top of HTTP)

3.3.3 Services Orientated Architecture/ Microservices Architecture:

An architectural pattern in software design in which system use cases are divided into one or more service operations, and these service operations are then implemented, either individually or combined with other service operations, by a reusable SOA service. The SOA architecture itself comprises five layers:

- Consumer interface: The GUI for end users accessing application services
- Business process: The representation of system use cases
- Services: The consolidated inventory of all available services.
- Service components: The reusable components used to build the services, such as functional libraries.
- Operational system: Contains data repository, technological platforms, etc.

Not to be confused with an API, this architectural pattern provides an aggregated collection of service which implement the use cases of the system. The user or developer themselves can choose which of these available services to call on. This architectural pattern was chosen because it is built around the principles of decoupling modules. Service objects must be decoupled from one another and can be distributed on different machines and implemented using different technologies, but implement a single interface, and communicate with one another using well defined interfaces, either locally or over a network. This also lends itself well to system reliability and scalability. More than one system provider may be used, enabling the system to switch to another service provider should one fail, or instantiating more service providers if system load increases.

4 Specific Requirements

This section gives a detailed description of the system requirements. It describes all the functional as well as the quality requirements of the system.

4.1 External Interface Requirements

4.1.1 Software Interfaces

The smart search application being build will be designed to run on Desktop running Windows OS.

4.2 Requirements

4.2.1 Functional Requirements

- Fuzzy Search: This program must be able to locate and return products that are related to the searched term. It should be able to return the same product if various ways of writing the name is used.

- Fast Run-time:

  Because there are very large amounts of data to be searched, it is vital for the program to be very efficient. Even though this could be a non-functional requirement, it is vital for this program to be as fast and efficient as possible.

- Machine Learning:

  The ability to learn from a user's previous interaction and adapt to it will improve the systems capability and ability greatly. This can be done by using the dictionary provided by the ZIZO database, for creating new links between products that appear seemingly unrelated.

- Handle Concurrent Users:

  This program will be used by many people, in many different fields simultaneously. It is therefore very important for it to be able to handle multiple users at the same time. This will be achieved by the Cloud based architecture.

- Scalability:

  Since it is essential for this system to be efficient with large datasets as well as small datasets, scalability will be an essential functional requirement. The systems ability to stay efficient with very large sets of data is an essential part to the system.

4.2.2 Non-Functional Requirements

- Predictive Typing:

  Predictive typing is when the program suggests a possible solution as the user is typing. This is often based on previous searches and most common searches. This will help improve the user experience.

- Spell checker:

  The program will check the spelling as the user is typing a product name to enhance the search.

- Roubstness:

  The system will not be easily breakable as it will be error proof to a feasible extent, to avoid unnecessary fault and not wholly be affected by the hardware failures. The system should be able to recover quickly from such failures, or at least be able to hold up or return to a valid stage/state.

- Reliability:

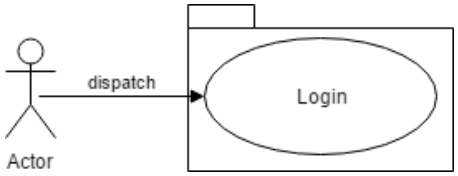
  This product should be able to adapt to different users in various fields as well as to different habits, interfaces, environments and needs of users.

- Storing and using search history:

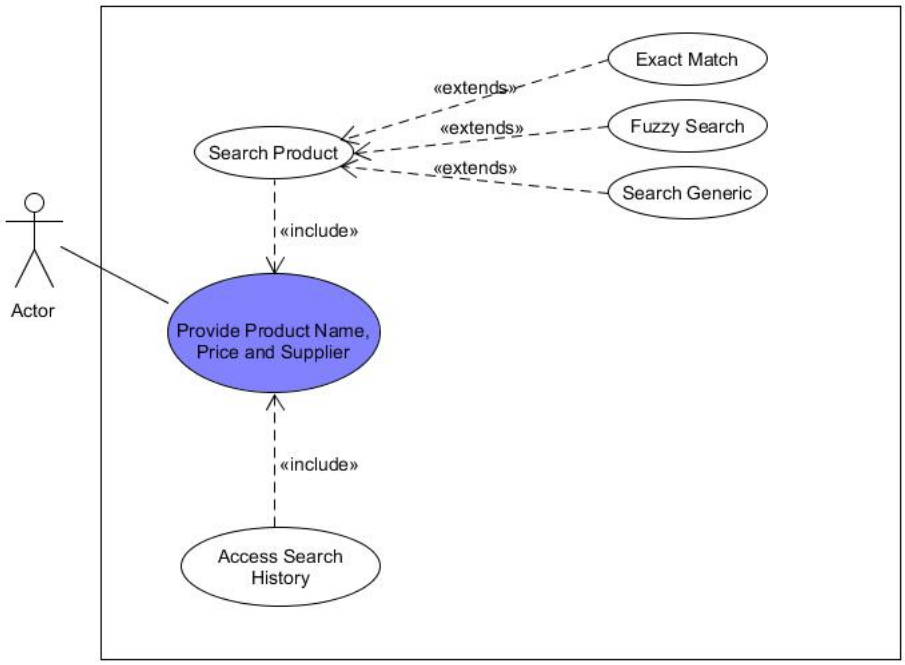
  Being able to store the search history will increase the user experience, help with predictive typing and improve the program's ability to learn and adapt.

4.2.3 Use Case Prioritization

1. **Critical**
   - Login
   - **User Interfaces**



   - Retrieve Wholesalers Accounts
   - Search Product
     Pre-condition: The user has to have been logged in successfully.
     Post-condition: System returns possible matches, or search not found.
     Use case:



UC diagram.jpg

Actor: User	**System: Purchase Management System Application** Searching Product
-	0. TUCBW System displays search bar
1. User clicks search bar, and enters product name	2. System retrieves profile information and displays it
-	4. System returns and displays all matches found (name variants inclusive) arranged according to prices and suppliers.
5. User selects appropriate match, or changes search conditions	-
7. TUCEW user makes decision by selecting preferred choice of product.	-

- Logout

2. **Important**
   - The application must search for the item the user enters for their desired search.
   - The system must return all possible matches to whatever the user searches.
   - The system must perform a smart search.
   - The application must be fast.
3. **Nice to have**
   - System using a search history, predictive typing.
   - A mobile version of the system such as Android.

4.3 Performance Requirements

The performance requirements of the smart search app. can be sub-divided into two components, of which are the following:

- Client Application:
  - The software needs to be responsive and lightweight enough to run well on desktop.

  - During User search, the Smart Search system will need to perform in real time showing results as necessary.

  - The data usage should be kept to a minimum as to not overload the whole system's Wi-Fi infrastructure being used for non-searching purposes.

- Server Performance Requirements:

  - The system should be able to handle multiple users using the smart search.

  - The system has to be able run at a high speed, returning the results of a search immediately.

  - The system should be able to handle a capacity of requests made to the database to make efficient searches and matches.

  - The system should be able to return matches of a high probability of available data in the database on the cloud.

  - The system will need to be able to concurrently handle smart searches across the board.

4.4 Design Constraints

- to be completed

4.5 Software System Attributes

The software system will have the following attributes:

- Availability: The system and its functions will be available to any registered user as long as there is an internet connection.

- Reliability: The system will provide results that are available on the database to match whenever a user performs a search/smart search.

- Portability: The system will be able to function across a multiple interfaces. Therefore, it will be able to port to Android and iOS with ease at a later time.

- Maintainability: The application will be able to be extended easily with new functionality. There will also be a good test environment to reduce and make it easy to find system errors.

- Robustness: The system will be able to stand against errors made by users in the search, returning no results and error messages as a result.

4.6 Other Requirements

4.6.1 Quality requirements

The web interface should be accessible to most modern PCs with a modern browser. The RESTful API will be implemented using the HTTP protocol as a high-level protocol. HTTP was the choice because it is a widely spread and deployed protocol, allowing a lot of flexibility.

5 Open Issues

5.1 GitHub Repository

Team CodeX Repository: `https://github.com/josephbondjobo/CodeX`
This repository contains:

- All work done by team members.