ReqEx

1.0 Version

Design Document

October 2021

Zachary Bruggen zbruggen2016@my.fit.edu
Nicholas Epler nepler2018@my.fit.edu
Ivan Hernandez ihernandez2018@my.fit.edu
Thomas Morrison tmorrison2017@my.fit.edu

Faculty Advisor: Dr. Khaled Slhoub kslhoub@fit.edu

Table of Contents

1. Introduction

1.1 Purpose

This document describes the layout of the user interface and the algorithms used to analyze code and extract requirements. The algorithms will analyze variable names, function names, conditional statements, input and output, and English comments. From this it will output functional requirements in an English format as a file type of the user's choice.

1.2 Scope

This system is intended to perform an automatic code analysis in order to extract functional requirements from a file. This system takes in either source code from a High Level Language, or a text file, and will generate a list of functional requirements in the user's chosen file type. From there, that list will be displayed in a separate panel, and the user can perform normal file operations on it. It is not designed to fix source code that is not meeting requirements, but rather to help the user understand what their program is doing line by line automatically to save the hassle of manually analyzing the source code file. Along with this, the system is also useful for debugging and maintenance of complex software, especially programs that aren't well documented.

1.3 Overview

This document describes the design behind the implementation of the ReqEx system. It will show step-by-step what happens between the user and the application interface, as well as what happens behind the scenes to analyze the code and output requirements.

1.4 Reference Material

The design for this system is based on the Systems Requirements Specification, and the Test Plan for this project. These can be found included in the GitHub repository for this project.
[Link]

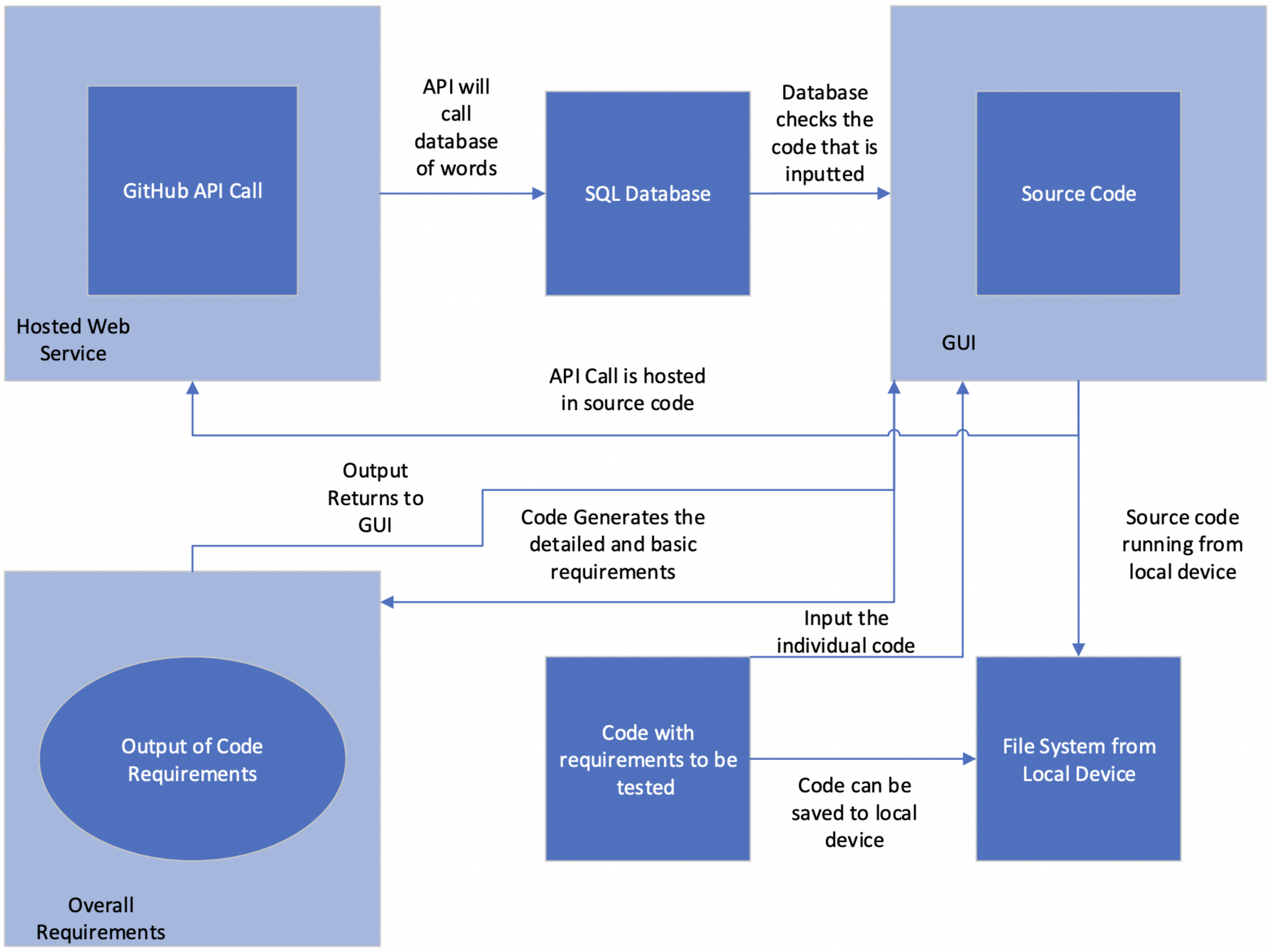
1.5 Definitions and Acronyms

GUI	Graphical User Interface
API	Application Program Interface
SQL	Structured Query Language
Requirements	Description of what the program is doing by

	looking at the source code file line by line

2. System Overview

2.1 UML Diagram


GitHub API Call
Hosted Web Service
API will call database of words
SQL Database
Database checks the code that is inputted
Source Code
GUI
API Call is hosted in source code
Output Returns to GUI
Code Generates the detailed and basic requirements
Source code running from local device
Input the individual code
Output of Code Requirements
Overall Requirements
Code with requirements to be tested
Code can be saved to local device
File System from Local Device

2.2 Design Overview

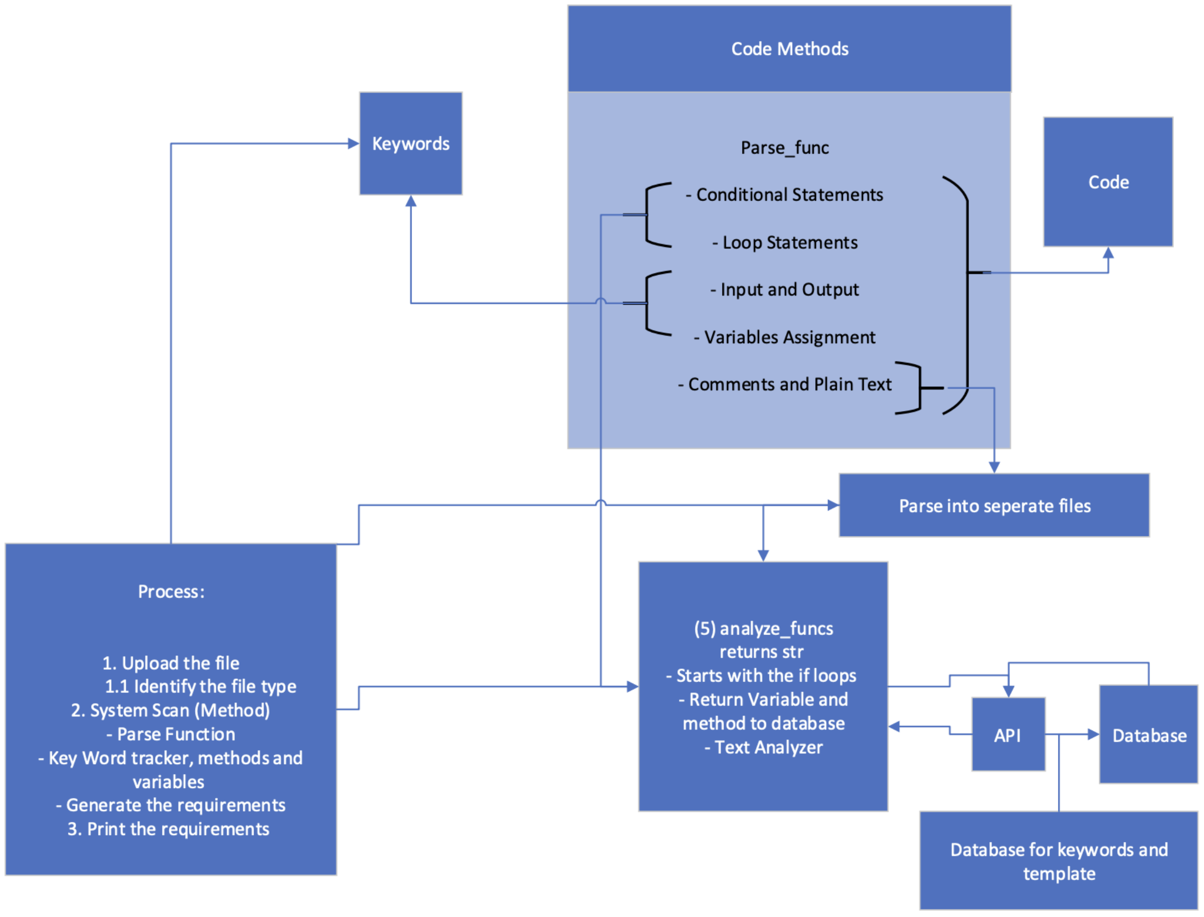
The software will be a standalone program where the user is able to generate a list of requirements from source code. This will be done in a three step process.

1. File Input
    a. During this step, the user will upload a file into the system that they are trying to extract requirements from. This file can be either source code from a High Level Language, header files for Object Oriented Languages, or a text file of English language bullet points
2. File Parsing
    a. In this step, the system will parse through the input file and separate the file into 5 distinct items. These 5 items will have templates associated with them, which will then extract keywords from each of those templates. Those keywords will then be sent to a database that holds templates for generating requirements, and requirements will be created using the keywords.
3. Generated Requirement Display
    a. After the system matches the input in the database, a list of requirements is generated. This list is then displayed in the appropriate section of the GUI, and the user will be able to add or remove text to this display if the system generated inaccurate information. Finally, the system will allow the user to select a file type to save it to the file system on the machine.

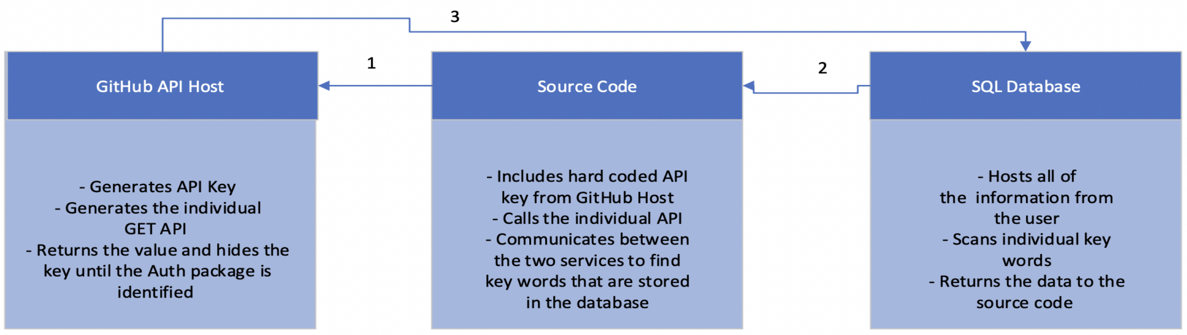
3. System Architecture

3.1 Architectural Design

UML Diagram


Code Methods
Parse_func
- Conditional Statements
- Loop Statements
- Input and Output
- Variables Assignment
- Comments and Plain Text
Keywords
Code
Parse into seperate files
Process:
1. Upload the file
1.1 Identify the file type
2. System Scan (Method)
- Parse Function
- Key Word tracker, methods and variables
- Generate the requirements
3. Print the requirements
(5) analyze_funcs returns str
- Starts with the if loops
- Return Variable and method to database
- Text Analyzer
API
Database
Database for keywords and template

API Call Chart


3
1
2
GitHub API Host
- Generates API Key
- Generates the individual GET API
- Returns the value and hides the key until the Auth package is identified
Source Code
- Includes hard coded API key from GitHub Host
- Calls the individual API
- Communicates between the two services to find key words that are stored in the database
SQL Database
- Hosts all of the information from the user
- Scans individual key words
- Returns the data to the source code

3.2 Decomposition Description

The code initializes with the input file being read by the computer, and initializes the standard input/output from the source code. The code process begins by tracking and generating the methods and variables in the code, this is done in the "analyze_func" function, parsing out all of the data from the source code. Once the file is being read it looks for keywords that are in the code itself and will parse these words. The code then generates an API call to the SQL database, in the database, key words are stored and linked to the corresponding value that they have, generating a meaning to the words that are parsed from the original code. When these words are linked to their meaning in the database, then the code will continue to run, extracting the meanings that originated from the code. This will loop until all of the values and key words are parsed in the original source code. This will then push the requirements that originated from the source code.

3.3 Design Rationale

The rationale for the design is breaking down the input file into 5 distinct objects. These five objects are:

1. Conditional Statements (If...Then...Else statements)
2. Loop States (For Loops, While Loops)
3. Input and Output (Function parameters)
4. Variables and Assignments (Variable names and the data they contain)
5. English Comments (Comments that follow an English language syntax)

These five items will each have a function assigned to them in order to pull out relevant data in the text through the format of templating. The data extracted from these function will be called the keywords, and the API will then call to the SQL database, which will hold templates for functional requirements based on templates with keywords to fill. The database will then return the requirements of the code, which will be listed out.

4. Data Design

4.1 Data Description

The majority of the data in this project will be represented by the extractor based code, all of the data is passed in the source code, to the new, extractor based code. The data passses and allows the user to extract the requirements from the code, passing the data between the source code, to the extractor code, to the SQL database, and then returning to the original extracting code that will pull the data in order to retrieve the requirements.

4.2 Data Dictionary

The data dictionary is broken up into separate categories:

1. Source Code
   - 1.1. Object Oriented language
     - 1.1.1. Internal files
       - 1.1.1.1. Methods
         - 1.1.1.1.1. Parsed Data
       - 1.1.1.2. Functions
         - 1.1.1.2.1. Variables
       - 1.1.1.3. Return Types
2. Extractor Code
   - 2.1. Object Oriented Language
     - 2.1.1. Internal Files
     - 2.1.2. SQL Database
       - 2.1.2.1. Individual parsed words from source code
       - 2.1.2.2. API Call from the code
     - 2.1.3. Methods
       - 2.1.3.1. Method to extract data
         - 2.1.3.1.1. Extracted data loops for the final requirements testing
       - 2.1.3.2. Function
         - 2.1.3.2.1. Variable testing and extracting
           - 2.1.3.2.1.1. Looping the extracted data until the final requirements are extracted

5. Component Design

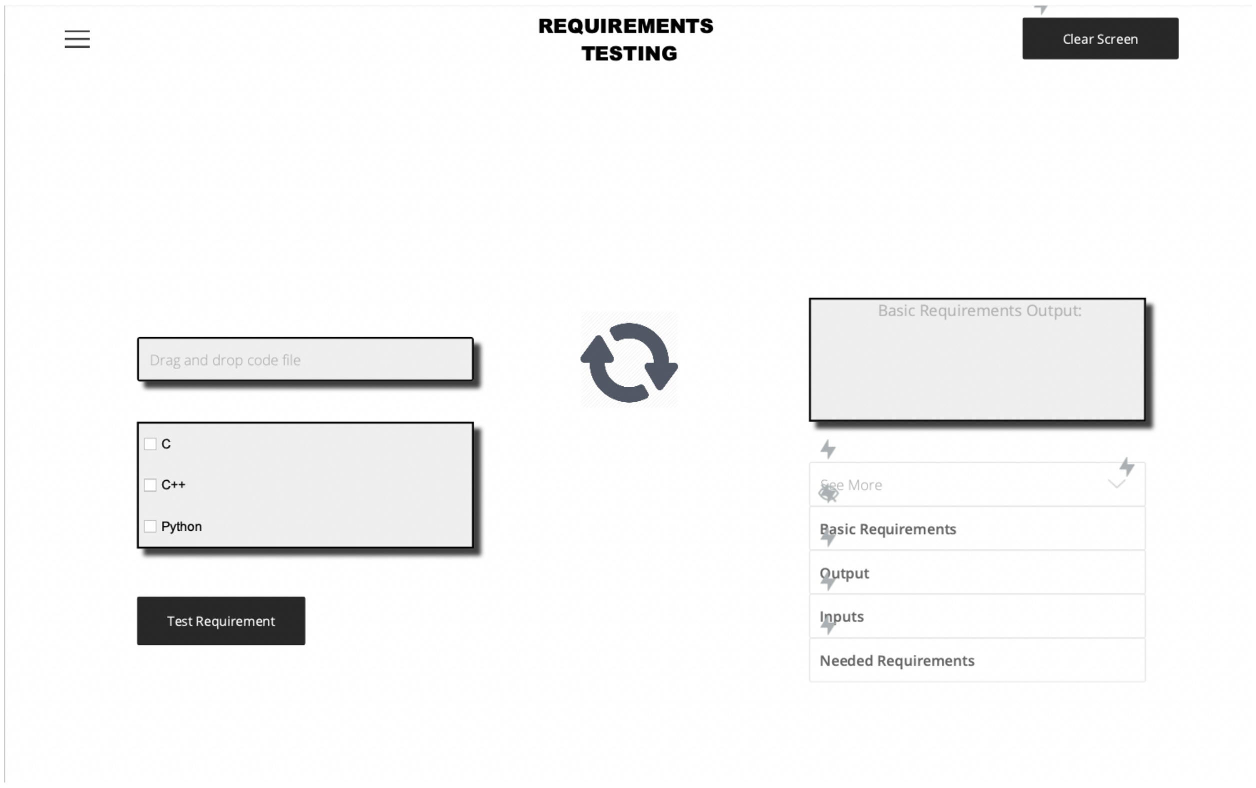
There will be parsing functions to analyze function names, conditional statements, loop statements, input and output, variable and assignment statements, and English comments. It will reference a database of standard function names associated with an English description of the function. It will then output the function description as well as the parameters taken in and return values. It will also state what is happening to the variables, i.e. mathematical calculations and assignments to values. Lastly, the comments will be parsed for the verb and noun to simplify the description to the user. Comments that are unused segments of code will be ignored.

6. User Interface Design

6.1 Overview of User Interface

The basic design for the user inference is to allow the user to view 2 files on the left and right sides of the GUI. On the left side of the GUI, the user will be able to view their uploaded file, and any sort of files that were previously uploaded. The file will exist in a viewing panel that will render out the text of the file. The generated requirements report will be on the right side of the GUI. Once the user asks the system to perform a requirement extraction, the generated report will be displayed in this viewing panel. The user will also be able to interact with the text in this file, if they feel that something needs to be added or removed. Along with this, the user will be able to perform normal file operations on the generated report, which will be buttons on the right side of the GUI.

6.2 Screen Images



6.3 Screen Objects and Actions

The GUI includes a basic prototyped UI that allows the user to add the individual file to the program, this file is read from the computer and can be included in the requirements testing. Then the object-oriented language of the original code is selected. The user then would click on the “Test Requirement” button, the output will return the basic requirements of the code, as well as an option to print them, then there is a drop down “See More” that will allow the user to select a more detailed view of the requirements, Outputs, Inputs, and more needed requirements for the original code. Each individual drop down will allow the user to see the details that are more applicable to the needs of the code.

7. Requirements Matrix

Not applicable at this time

8. Appendices

Not applicable at this time