S	
E1	
E2	
For Office Use Only	

Masters Project Final Report

(MCS)

2019

Project Title	Improving the clarity of requirements by generating RCI value using Machine Learning
Student Name	L. Nijanthan
Registration No. & Index No.	2017/MCS/055 / 17440552
Supervisor's Name	Dr Thilina Halloluwa

For Office Use ONLY

Improving the clarity of requirements by generating RCI value using Machine Learning

A dissertation submitted for the Degree of Master of Computer Science

L. Nijanthan
University of Colombo School of Computing
2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: L. Nijanthan

Registration Number: 2017/MCS/055

Index Number: 17440552

_____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr. L. Nijanthan

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr Thilina Halloluwa

_____

Signature: Date:

Acknowledgements

I am using this opportunity to express my gratefulness to everyone who supported me throughout the master's individual project. I am grateful for everyone's advice, guidance and constructive criticism for the project. I would like to thank my project supervisor Dr. Thilina Halloluwa, a Lecturer of University of Colombo School of Computing who has supported with valuable knowledge and vast experience.

Abstract

Requirement Engineering is a key phase in software development, which improves quality and maintainability of the software. In requirement engineering phase, there are several ways to measure the quality of the requirement, such as reliability, performance efficiency, security, maintainability, rate of delivery, testability and usability. Requirement Clarity Index (RCI) is a quality measure, that can be implemented in a system to reflect level of clarity each stakeholder has on the project requirements. In other words, RCI used to measure of having clear understanding on what stakeholder needs is essential for a successful software system delivery. Accurate identification of RCI will help reduce the ambiguity of requirements which reduce the rework, and improve maintainability. Measuring RCI value manually requires higher human involvement, which is expensive, time consuming and subjective.

This research is intended to automate the requirement clarity index generation process using rule-based machine learning approach. Research has two main phases; (1) a text summarization phase and (2) requirement quality score analysis phase. Use of text summarization phase, natural language requirement details are summarized and key aspects of requirement details get extracted. In requirement quality score analysis phase, scoring is applied to summarized content, which is generated from text summarization phase, using identified quality factors from literature survey. Quality score for the factors returned from the quality score analysis phase. Using the generated quality score and manually computed RCI value, mapping table is created for rule based RCI generation approach. Mapping table contains range of metric scores and its related RCI value. Dataset of requirements undergoes into these phases and mapping table is generated. After generating the mapping table, requirements would be undergoes into the phases and quality score get computed. According to the quality score, particular range of quality score is mapped and respective RCI value is returned from the mapping table. With the help of large set of dataset, this research can produce more significant results for new requirements.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Overview

Requirement Engineering is a process involved in gathering, analyzing, and documenting software requirements. This is considered one of the most crucial stages in all of the software development life cycles[1]. In the traditional software development models such as waterfall, the complete requirement engineering process has to be conducted before the development[2]. However, in more recent models such as Agile[3] and RUP[4] explore a more iterative and team based approach to quickly deliver the software system without completing the entire software development tasks in sequence.

In Agile methodology, software system is divided into smaller working components. Those components are developed by system development team simultaneously on software development phases such as planning, requirement engineering, designing, implementation, testing and user acceptance testing[5].

In requirement engineering phase, the product development team will analyze the requirements taken into the iteration. Team will utilize diverse quality measures to identify the quality of the software requirement provided by the stakeholders - Requirement conformance and Requirement stability, for example[6]. Requirement clarity index (RCI) is one of such quality measures used in agile software development methodology[7].

RCI is a quality measure, that can be implemented in a system to reflect level of clarity each stakeholder has on the project requirements. Having clear understanding on what stakeholder needs is essential for a successful software system delivery. RCI is systematic way to determine the clarity of requirements. It gives responsibility for each stakeholder to determine their understanding on the software requirement. Determining the requirement gaps in the early stage of the software development will reduce the rework, and improve maintainability[7].

RCI values can vary from 1 – 5 depending on the clarity of the software requirements. Industries and experts follows this standard to measure the clarity of their requirements[8].

1	No idea (Unware of the requirement).
2	Vague idea (Aware of the requirement, but the requirement is too vague to start considering approaches).
3	Somewhat clear (Requirement is known, but details are unclear. Major assumptions are being made, which if invalid may lead to significant rework).
4	Clear (Requirement is known, and details are more or less clear. Minor assumptions are being made, which if invalid may lead to minor rework).
5	Perfectly clear (Requirement is known, and details are clear. No more clarifications are needed).

Table 1 - RCI value and description

Human involvement is higher in the current RCI value generation process followed by product development team, which is expensive and time consuming.

Main focus of this research is to propose an approach to automatically measure RCI for a given requirement with a reasonable accuracy.
The main research question I am trying to address in this study is as follows
How to determine the RCI value of a requirement with minimal human interaction ?

1.2. Motivation

Requirement engineering is a key phase of software development life cycle. Analyzing requirement requires particular skill set and experience. Measuring the requirement clarity index of requirement is one of the process followed by industries to identify the quality of the requirement. This process conducted by product team members manually. This manual process is very expensive, time consuming, and human error rate is relatively higher.

Motivation to automate this process came cause of the complexity of this process. Implementing an automated approach for this complex process will reduce human involvement, which will minimize human error. Implementing automated approach for this type of complex process requires large amount of data for accurate results. Using the previously measured RCI values, future requirement RCI value can be measured, which requires large dataset and training.

1.3. Aims and Objective

Primary aim of this research is to identify an approach to measure requirement clarity index value of a given requirement. Intention behind this research is to minimize the human involvement in measuring the RCI value of the requirement as it may involve a significant human error. Dataset of natural language requirements and manually captured RCI values of those requirements is the inputs for this proposing approach.

To achieve the aim of this research, it requires following objectives. Using named entity recognition[9] and part of speech (POS) tagging input text words are categorized. It is necessary to identify the anaphora of the sentences before doing the summarization. Anaphora is a technique to refer back an entity which has been introduced earlier in the text[10]. Anaphora is used to identify repetitive content in the given input text. Identified unique content inputted into data pre-processing phase to derive summarized content. Summarized content of the requirement use identified metrics to measure the quality score. Entire dataset undergoes into above mentioned approach and quality score would be measured, which mapped into manually captured RCI value. Basically mapping table generated using quality score range and the RCI values. Future requirement text will be summarized and RCI value will be measured using the mapping table.

Aims of this research

- Develop an automated intelligent approach to measure RCI value of a given requirement.

The main aim of the study was achieved by systematically addressing the following objectives.

- Minimize human involvement in measuring RCI value of a requirement.
- Understand how to use suitable text summarization techniques to derive necessary summarized content of natural language requirement.
- Understand how to implement a quality scoring mechanism to measure the score of summarized content for identified factors and generate the mapping table.
- Understand how to apply rule-based approach to generate RCI value using mapping table records.

1.4. Scope

This research project mainly focus on evaluating the requirement clarity index of functional requirements expressed as user stories[11][12] containing textual description in English (no other languages). This research mainly focuses on extractive summarization techniques to extract sentences, and abstractive text summarization technique will be applied on extracted sentences to build an internal semantic representation of the original content.

User story format[11]

AS A <TYPE OF USER>, I WANT <SOME GOAL> SO THAT <SOME REASON>

Real world example

As an application user, I want to read FAQs, so I can get quick answers.

2. Background

2.1. Related Work

Requirement Analysis is a relevant application area for a variety of semantic technologies related to the extraction, disambiguation, and exploitation of knowledge derived from technical requirement documents. Requirement clarity index is a metric used to identify the ambiguity of a particular requirement.

Many researchers have done researches related to measuring the quality of a requirement, identifying ambiguity of the requirement in the early stage of software product development and automating the requirement quality measuring.

There were many research works has been done related to requirement analysis using automated requirement tools, machine learning techniques, and natural language processing.

In 2011, Mohammad Ubaidullah Bokhari and Shams Tabrez Siddiqui published a research paper that proposed software requirements metrics to improve the process of managing requirements and quality of the product. These metrics identify and measure the necessary factors that affect software development. Researchers proposed the following metrics to measure quality requirement which can be used in the requirement analysis phase:- Requirement traceability metrics, Requirement completeness metrics, Requirement volatility metrics, and Size metrics. Measuring quality of requirement with these metrics manually is expensive, time consuming and prone to error. Cause of these limitations, researchers used automated requirement tools to do the measurement. In this research, researchers came up with formulas to measure the mentioned metrics[13]. These formulas can be applied to identify requirement clarity index related metrics and measurements.

In above mentioned research work, researchers have gone through all set of activities followed in the requirement engineering process and analyze them. Requirement engineering process includes activities such as problem synthesis, requirement elicitation, requirement analysis and negotiation, requirements specification, system modelling, requirement verification and validation, requirement documentation, and requirement management[14]. Researchers were analyzing software requirement specification activity that came up with two error types which will affect the quality of the software requirement. Error types known as knowledge errors

caused due to not knowing what the requirement are, and specification errors caused due to lack of knowledge or experience of specifying requirements.

Furthermore, A. M. Davis and E. E. Mills identified internal attributes, which describe how requirement should be specified and external attributes, which describe the overall or outer appearance of software requirement specification (SRS) document and how they affect other quality related attributes. Internal attributes are Unambiguous, Correct, Complete, Understandable, Verifiable, Internal consistent, Modifiable, Annotated by relative stability, Annotated by version, Precise, Traced, Traceable, Not redundant, At the right level of detail, and Organized[15]. External attributes are achievable, electronically stored, design independent and reusable. Researchers defined software metrics for these requirement document attributes to give the overall information about the development product such as cost, time, and all phase information[16].

These software metrics are defined in a general way to represent the quality requirement document. Using these extracted metrics, we will be able to derive metrics which will help to measure requirement clarity index.

Amit Mishra, A. Awal, Joseph Elijah, A. AbdulG, U. M. Gana and I. Rabiu proposed an automated software requirement analysis to close the understanding gap between user and system analysis to construct a better architecture to achieve the usability of software. In this research, researchers came up with a relational database that aimed at capturing user related information and requirements that were developed on the repository[17]. The output data repository will be very helpful to identify patterns which can be used as an input to build an intelligent system to measure the quality of the requirement.

J. Györkös proposed an approach to reduce the bad experiences in tracking, understanding and validating software requirement specifications common to mid-range software development projects. Researcher used Computer-aided Software Engineering (CASE) tools to show how extraction, utilization, and interpretation of requirements at the elicitation phase[18]. Using this approach will be able identify the factors which are not suitable to measure the quality of the software requirements.

In 2012, M. Nardini, F. Ciambra, F. Garzoli, D. Croce, D. De Cao, and R. Basili proposed a distributional method to train a kernel based learning algorithm – Support Vector Machine, for

example, as a cost effective approach to validate requirement from text support of requirement analysis in the design of a complex systems – Naval Combat Systems for example[19].

This research mainly focusing on designing a naval system. During the design, following phases need to be performed:- domain analysis, elicitation, specification assessment, negotiation, documentation, and evaluation. Generally, these phases carried out without any reuse of old analysis performed over previous system[19]. In this scenario, researchers faced a great challenge in translating user requirements and problem domain described in natural language into the consistent modelling of the target application. Vagueness and ambiguity are the main phenomena that make the natural language used to describe user requirements a challenging task.

Researchers proposed statistical learning methods embedded in a large scale natural language processing system in support of requirement analysis. Advanced techniques of natural language processing combined with machine learning such as Statistical Information Extraction and Textual Entailment is added to the model to improve the applicability on a large scale. They implemented a requirement analysis system using a machine learning technique according to their architecture. The system has been applied to a real scenario – Naval Combat System, for example. They used an empirical evaluation method to evaluate their system functionalities, such as requirement identification, information extraction, recognition textual entailment[20]. Empirical evaluation method results are derived by observation and experiment instead of theory, which is one of the suitable evaluation method to evaluate a machine language model[21]. But the annotated requirements used in this research is not sufficient to fully evaluate a machine learning model. With the large data set of annotated requirements and empirical evaluation will improve model results accurate.

In 2018, Tetsuo Tamai and Taichi Anzai proposed an approach to analyze requirements found in the software requirement specification (SRS) document in terms of their volume, balance, and structure. Natural language processing and machine learning techniques used to detect and classify quality requirement sentences[22].

In the above mentioned research, researchers proposed a method for mining quality requirements in an SRS document. Researchers collected SRS documents that are available on the web, issued by local governments, and public institutions in Japan. To train and verify their machine learning model, the labelling has been done manually by authors. Also, they used

Japanese morphological analysis since they used Japanese SRS documents as their data set. To select a suitable machine learning method, they conducted some preliminary experiments, including a comparison of the conventional multi layered perceptron and convolutional neural network (CNN). For implementation, they used existing tools called Chainer combined with Python libraries[23].

They implemented a top level classification between non-functional requirements and functional requirements. As a result of the tool, researchers introduce a way of categorizing functional requirements. This classification mechanism will be helpful to find the requirement clarity index, since RCI value calculated only for functional requirements[24].

Ahmad Mustafa, Wan M. N. Wan Kadir, and Noraini Ibrahim have identified that an effective way to minimize the ambiguity from the natural language requirement is requirement boilerplate. Requirement boilerplate is known as requirement template or pattern, have been part of requirement writing best practice[25]. They proposed an approach to automate the requirement analysis phase using a language processing tool and proposed a natural language requirement analysis model. They present an open source General Architecture for Text Engineering (GATE) framework for automatically checking natural language requirements against boilerplates for evaluation purpose[26].

An analysis of requirements lexical and syntactic analysis used to identify the vague, incomplete, and inconsistent requirements, statistical and semantic techniques used to identify similar or duplicate requirements and lexical and semantic analysis methods used to classify the non-functional and functional requirements. In the proposed model text extraction, boilerplates checking, and natural language requirement quality checking steps were taken to resolve issues of ambiguity, incompleteness, and inconsistency.

This research was using a well-known framework like General Architecture for Text Engineering (GATE) to diagnose the ambiguity. GATE is Java suite of tools used for many natural language processing tasks, including information extraction in many natural languages[27]. Rupp's boilerplates were used to natural language requirement conformance. The model proposed by the researchers is for natural requirement analysis which can be used to come up with a model to measure the quality of the requirements.

Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath did an experiment to observe the incomplete and ambiguous software requirement specification (SRS) statement which is written by the customers. Researchers proposed an approach to help the analysis phase, particularly conducting object oriented analysis by generating class diagrams and all its details from the SRS statements. They used object oriented analysis, using natural language processing (NLP) techniques to conduct an intelligent analysis[28].

In this research approach, inputted software requirement specification document will be assigned part of speech for each word using a parser. Then stemmed the noun tagged words to find the root nouns and duplicate will be removed. List of stemmed nouns with their occurrence frequency and candidates for the class name from the list will be generated. XML (Extensible Markup Language) will be generated from the list. Generated XML will be converted into XSD (XML Schema Definition). The class diagram will be generated from the XSD. In this research work measuring the quality of the requirement is not considered. Researchers assumed that the inputted requirement is well defined and structured, which is not good assumption to produce accurate results. But generating the UML diagram of the requirement will give clear understanding to some extent. Natural language processing approach used in this research can be used to do the preprocessing of a given natural language requirement.

2.2. Summary

Authors (Year)	Objective	Technology
Mohammad Ubaidullah Bokhari and Shams Tabrez Siddiqui (2011)	Propose software requirements metrics to improve the process of managing requirements and quality of the product.	Survey and Interview based
A. M. Davis and E. E. Mills	Identify internal attributes, which describe how requirement should be specified and external attributes.	Analysis and Survey based
Amit Mishra, A. Awal, Joseph Elijah, A. AbdulG, U. M. Gana and I. Rabiu	Propose an automated software requirement analysis to close the understanding gap between user and system analysis	Survey and Experiment based
J. Györkös (1994)	Propose an approach to reduce the bad experiences in tracking, understanding and validating software requirement specifications common to mid-range software development projects	Computer-aided Software Engineering (CASE)
Nardini, F. Ciambra, F. Garzoli, D. Croce, D. De Cao, and R. Basili (2012)	Propose a distributional method to train a kernel based learning algorithm – Support Vector Machine	Domain analysis, Elicitation, Specification Assessment, Negotiation, Documentation, and Evaluation
Ahmad Mustafa, Wan M. N. Wan Kadir, and Noraini Ibrahim	Identify an effective way to minimize the ambiguity from the natural language requirement using requirement boilerplate	General Architecture for Text Engineering (GATE) framework
Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath	Propose an approach to help the analysis phase, particularly conducting object oriented analysis by generating class diagrams and all its details from the SRS statements.	Object Oriented Analysis (OOA), Natural Language Processing (NLP)
Tetsuo Tamai and Taichi Anzai (2018)	Proposed an approach to analyze requirements found in the software requirement specification (SRS)	Chainer combined with Python libraries

	document in terms of their volume, balance, and structure.	

Different kind approaches, techniques have been followed and challenges faced by the researchers to achieve automatic requirement quality measuring was discussed in this chapter. Based on the previous work, identifying an optimal approach to measure requirement clarity index is undisclosed. The latter part of this chapter focuses on improving requirement quality measuring and minimize human involvement.

3. Methodology

Requirement engineering process is complex since description provided by the stakeholder about the requirement is very lengthy. Description provided by the stakeholder can have necessary and unnecessary information about the requirement. Reading and understanding lengthy text content requires time, effort and experience. Analysis process highly coupled with the individual who is doing it, which is a huge drawback for this type of process. The proposing approach tries to summarize the actual description and extract important factors from it. Below figure is a sample user story extracted from Jira instance which has three major components.

1. Title of the requirement.
2. Description of the requirement.
3. Attachments related to requirement which is optional.

Title

Daily summary Email Notification

Description

As the product owner or support team, I would like to receive a daily summary report of issues to priorities fix for the issues. Email template should be in the provided format. Email body should contain technical configurations details which is required to investigate the issue.

Attachments

Attached email template

Figure 1 - Sample user story from JIRA

This research is focused on the text in the description section. In the above example, stakeholder provided email template in separate attachment section. In that situation need to consider attachment section also.

The main objective of this research is to develop an approach to summarize and extract important details of the requirement description. This research approach used supervised learning approach to calculate RCI values. Literature review was used to identify factors which

affect quality of the requirements. Came up with this approach by analyzing several supervised techniques in the literature.

This research methodology has two main phases. Phases are:

- Text Summarization Phase
- Requirement Quality Score Analysis Phase

Following figure is the complete detailed design diagram.

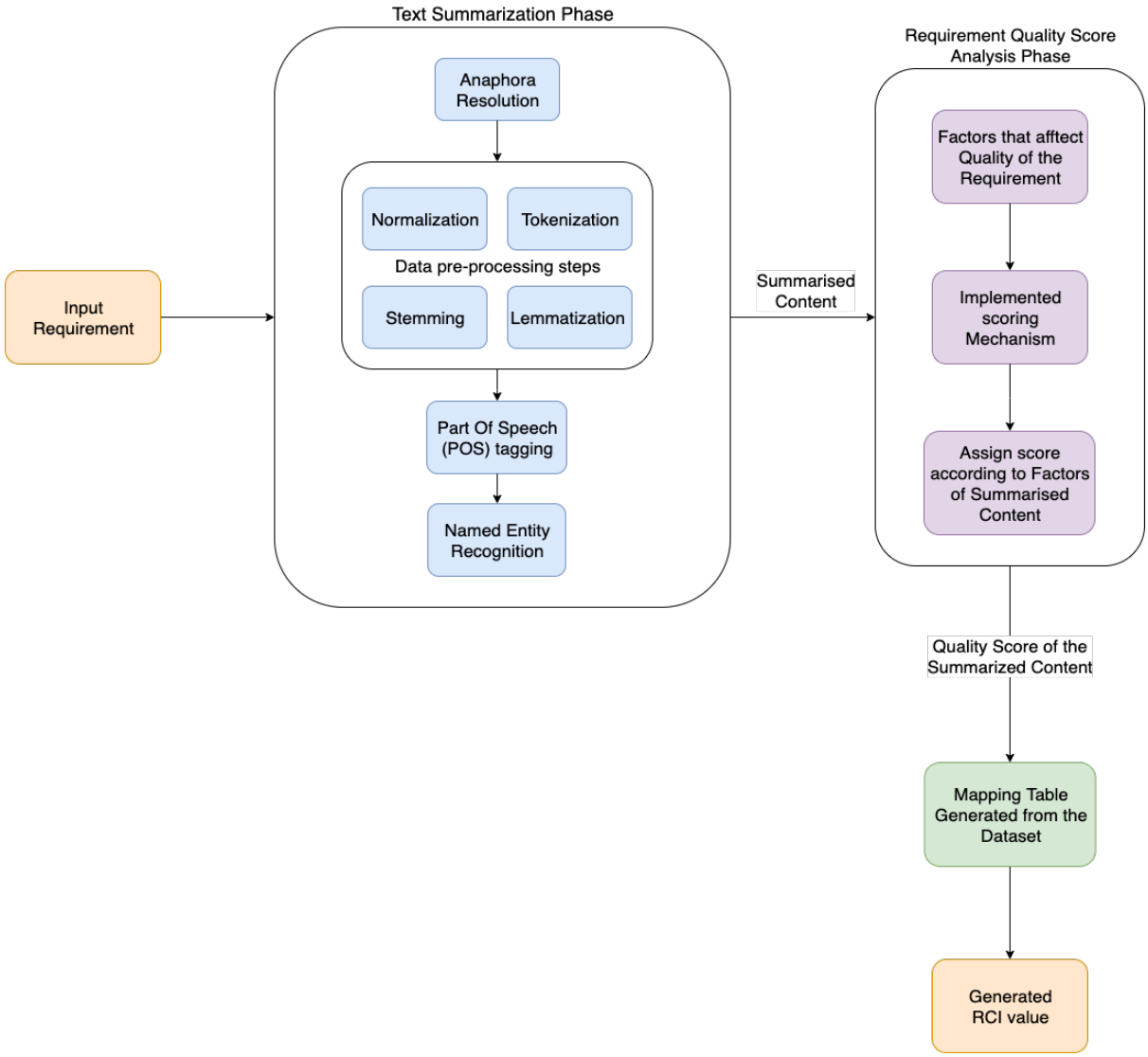


Figure 2 - Complete Detailed Design Diagram

Following figure is the high level design diagram of the system methodology.

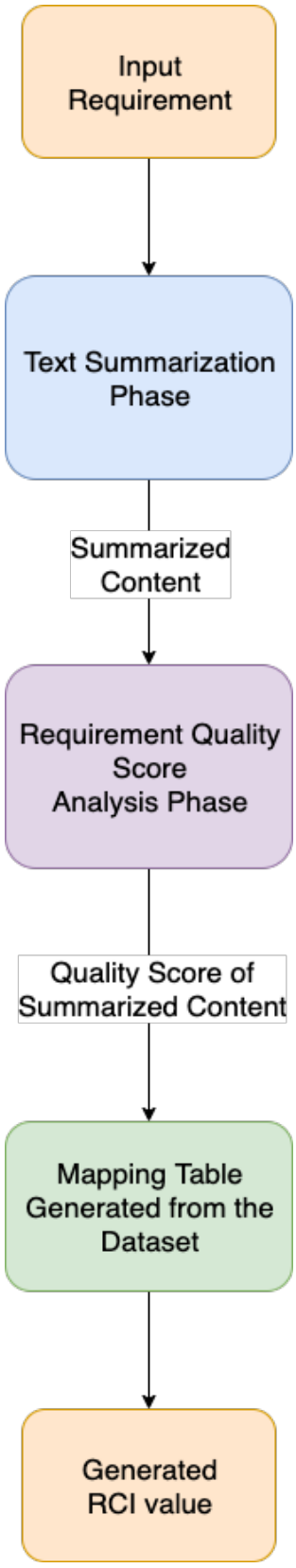


Figure 3 - High Level Design Diagram

3.1. Input Requirement

Natural language requirement provided by the stakeholder. Requirement title, requirement description, and attachments (if any) taken into the system for analysis.

3.2. Text Summarization Phase

This phase has many sub activities to generate summarized content from input requirement. Following figure is the detailed view of the Text Summarization Phase.

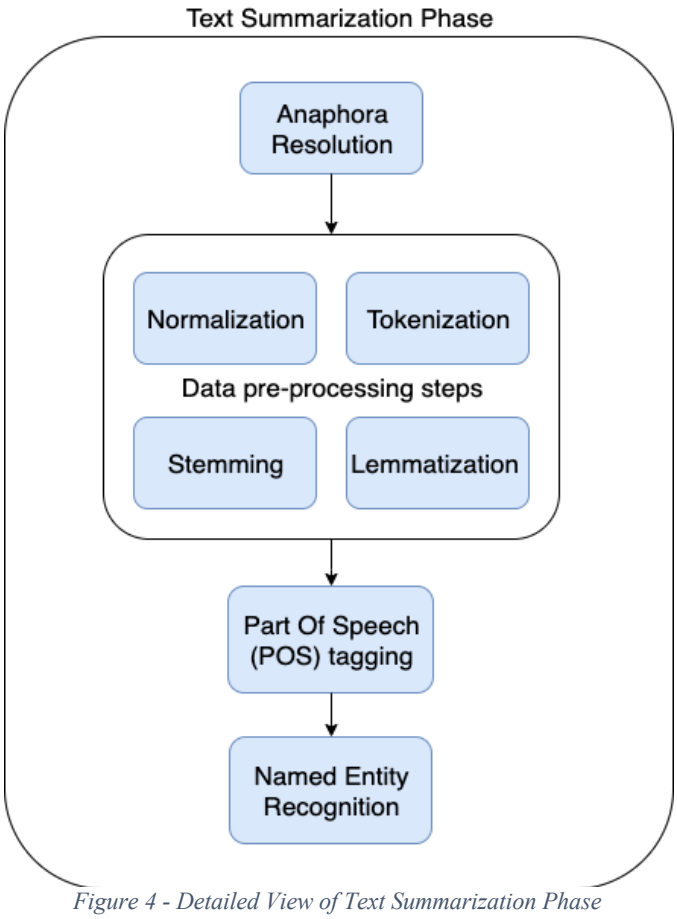


Figure 4 - Detailed View of Text Summarization Phase

Following are the activities of this phase,

Anaphora Resolution is a process of finding the antecedent (Entity to which the anaphor refers) for an anaphor (Reference that point to the previous item)[29]. Using anaphora resolution would be able to eliminate repetitive content from the inputted requirement. Anaphora resolution has different approaches such as rule based, statistical based, and machine learning

based. This research project is used rule based and machine learning based approach for the anaphora resolution.

Data pre-processing steps are used to generate summarized content from the input text. Following activities carried out to summarize the text content. **Tokenization** is a process of chopping the content into pieces called tokens. From this activity certain characters removed from the content – punctuation, for example[30]. **Normalization** is process of process of transforming content into single standard form. For example, the word tomrw, 2morrow should be transform into tomorrow[31]. Stakeholder may use some raw text in the requirement, those text should identify and normalized. **Stemming** and **Lemmatization** is process of reducing inflectional and derivational[32] form of word to a common base form. For example, requirement text can contain different form of word, those forms should be converted into base form[33]. After conducting above mentioned preprocessing step, content is clean and unnecessary data is reduced.

Part Of Speech (POS) Tagging is a process of categorizing the word in the content according to a particular part of speech, based on its definition and context[34]. Which is an important step in this phase. Identifying the category of the word according to the context improves the quality of text summarization. From the POS tagging, relationship between the words also can be identified. Some word in English has two different meaning according the place it used. In that case, identifying the category according to the context helps understand the sentence.

Named Entity Recognition is an algorithm which used extract information from unstructured text content and categorized/classified into groups. For example, United States of America is classified into Country. From named entity recognition, named entities are identified from the text content. In this research, identifying named entities gives a clear idea to summarize content in the requirement. Summarized text content is created from the input unstructured requirement.

3.3. Requirement Quality Score Analysis Phase

In this phase, summarized content is taken as input and quality score of the summarized content is outputted. Following figure is the detailed view of Requirement Quality Score Analysis Phase.

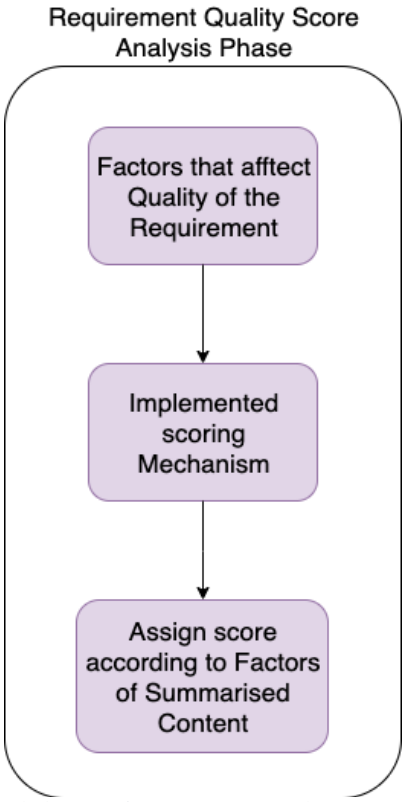


Figure 5 - Detailed View of Requirement Quality Score Analysis phase

From the literature survey identified **Factors that affect Quality of the Requirement** are used to measure the quality of the requirement. There is a **Scoring Mechanism** implemented to assign score for the factors according to the impact, those factors made to quality of the requirement. From the scoring mechanism scores are assigned to the identified factors from the summarized content. Quality score of the summarized content is outputted from the Requirement Quality Score Analysis Phase.

3.4. Mapping Table Generation from Dataset

Collected dataset contains requirement title, unstructured description and manually captured RCI value. Dataset of unstructured requirement description is passed into text summarization phase and requirement quality score analysis phase. Score generated from the quality score analysis phase is mapped according to its manually captured RCI value. Mapping table contains range of quality score and respective RCI value for that score range. Following table is a example mapping table structure,

Quality Score Range	RCI
0 – 19	1
20 – 40	2
41 – 59	3
60 – 79	4
80 – 100	5

Table 2 - Sample Mapping Table

3.5. Generate RCI value

RCI value is generated with help of mapping table according to its quality score of the input requirement.

4. Implementation

This research project implementation has three main sections such natural language text processing, metric quality score analysis and mapping table generation.

4.1. Natural Language Text Processing

Dataset of requirement contains requirement title, description and manually computed RCI values, collected from the company which follows agile methodology. In this section requirement title and description processed and summarized content is generated. Following are the text processing steps used in this research project.

4.1.1. Anaphora Resolution

In this section, all the pronouns in the title and description are resolved and resolved text replaced in the input requirement. Stanford CoreNLP library is used to achieve this processing.

Following is the example of this step

- **Input** - Amal is an undergraduate student. He is currently in second semester.
- **Output -** Amal is an undergraduate student. Amal is currently in second semester.

4.1.2. Data Pre-Processing Steps

In data pre-processing section, input text is processed and uniformed text get generated. Tag values, html string, special characters and white spaces are removed from the text. Furthermore, processed text is normalized and standard form of text is generated. Generated text is undergoes into named entity recognition step and categorized and classified into groups. Pre-processed text is outputted from this step and inputted to metric quality score analysis section.

4.2. Metric Quality Score Analysis

This step takes pre-processed unified text as an input and returns quality scores for the identified factors. Factors which affect requirements known as statement count, word count, word list count, list count and noun count. Scores and rating is assigned for each factors and

cumulated score is returned from this phase. This score used to populate mapping table which is the key aspect of this research project.

$$\text{Cumulated Quality Score} = \sum_{i}^{factors} Factor\ Score$$

- Example

$$Quality\ Score = Proper\ Sentence\ Score + Listing\ Score + Proper\ Word\ Score$$

Implemented model can be expandable to add more factors, which is very important improve the end result. With more number of factors we are able measure accurate quality score of the requirement string. Computed score used to populate the mapping table.

4.3. Mapping Table Generation

Mapping table is the key aspect of this research project. Mapping table contains range of quality score and respective RCI value of it. Quality score of the requirements with same RCI values grouped and data is formed to generate the graph. Following graph is a sample quality score of twenty requirements with RCI value 5.

Figure 6 - Quality score for twenty requirements with RCI 5

With the plotted graph, system automatically identify the metric quality score range. System skips the outliers and get the values which has frequent occurrence for the range detection. System follows this approach for all the RCI values and populate the mapping table.

After generating the mapping table with entire dataset, new requirement's quality score is mapped with mapping table and respective RCI value is returned. Rule based approach is followed in this research to identify the RCI value. With the huge dataset of requirements, we are able extract most accurate quality score range for the RCI values.

5. Evaluation

This research project contains two main phases; the text summarization phase and the requirement quality score analysis phase. In the text summarization phase, natural language requirement is converted into summarized content. Requirement quality score analysis phase, using the implemented scoring mechanism summarized content is scored for identified factors.

Evaluation plan of a natural language processing system must be designed to address the issues related to specific task. Evaluation must identify all system elements that can figure as performance factors. Partitioning of data used is widely used evaluation approach for natural language processing systems

5.1. Partitioning of Data used in Evaluations Approach

This research project follows partitioning of data used in evaluations approach for the evaluation. Key element of this approach is collected dataset of requirements with manually measured RCI values. In this approach dataset is partitioned into two disjoint subsets such as training data and test data. Training data is known as input to the system. Training dataset is used to build the model. In this research, using the training dataset of requirements are taken in text summarization phase and requirement quality score analysis phase. Natural language requirement of this dataset is used to build the scoring mechanism in the requirement quality score analysis phase. Training dataset RCI values are not taken into the output of the system. Test data is used to evaluate the system's performance after development of the system. Using test dataset RCI value, system performance is measured. For example, after completing the system implementation test dataset is inputted to the system for evaluation process. System generates the RCI value using the research process and test dataset RCI values are used to assess the result.

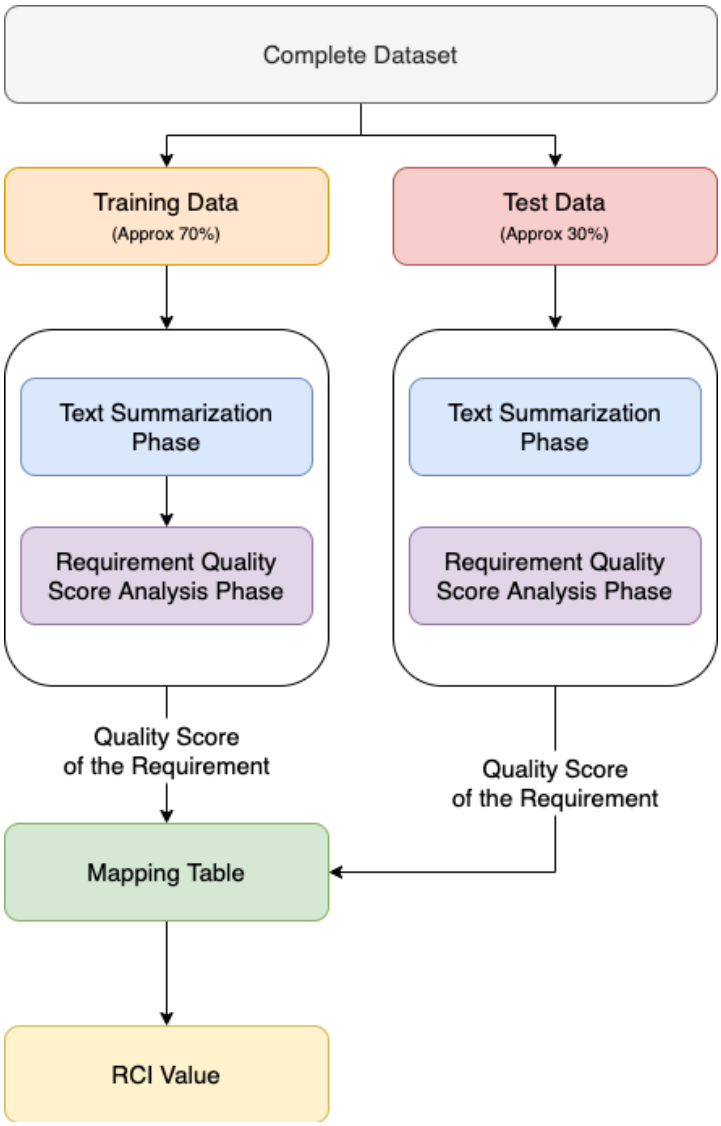


Figure 7 - Evaluation Plan Approach

Based on the partitioning data evaluation approach, dataset is partitioned into two disjoint dataset. Approximately seventy percentage of the data records considered as training data and rest of the data records considered as test data.

6. Results and Observations

This chapter briefly explains the results of the evaluation followed in the research project and observations of the results.

6.1. Results

Based on the selected evaluation approach, quality scores are calculated and plotted for range identification. Using the quality score ranges system generates the mapping table. Following table show the generated mapping table for partitioned training dataset.

Quality Score Range	RCI
23 – 46	1
47 – 58	2
59 – 69	3
70 – 78	4
79 – 100	5

Table 3 - Resulted Mapping Table from evaluation

System calculate the quality score of the new requirement and map the calculated score with mapping table RCI value and returns it. System evaluate the mapping table using partitioned dataset and verify whether RCI value returned from the system is correct or not.
Following are some sample user stories and its RCI values

User Story 1

Title

Ability to re-label the active checkbox on the study object.

Description

As a User, I should be able to re-label the 'Active' checkbox on the Study object to 'Sync to iPad', so-as to reduce confusion and increase intuitiveness of the Study functionality. In the Studies page On the Study page, there is a checkbox called 'Active'. Functionally, this is used to control which Studies get synced to the iPad. This causes some degree of confusion because there is also a 'Status' picklist on the Study page which contains a list value called 'Active'. Users wonder what the difference is between populating the Status field with a value of Active, and the checking the Active checkbox.

RCI Value - 5

Figure 8 - User story 1

In the above mentioned user story (User story 1) manually calculated RCI value is 5, which mean requirement is perfectly clear to start the implementation. Following is the output RCI value generated from the implemented system.

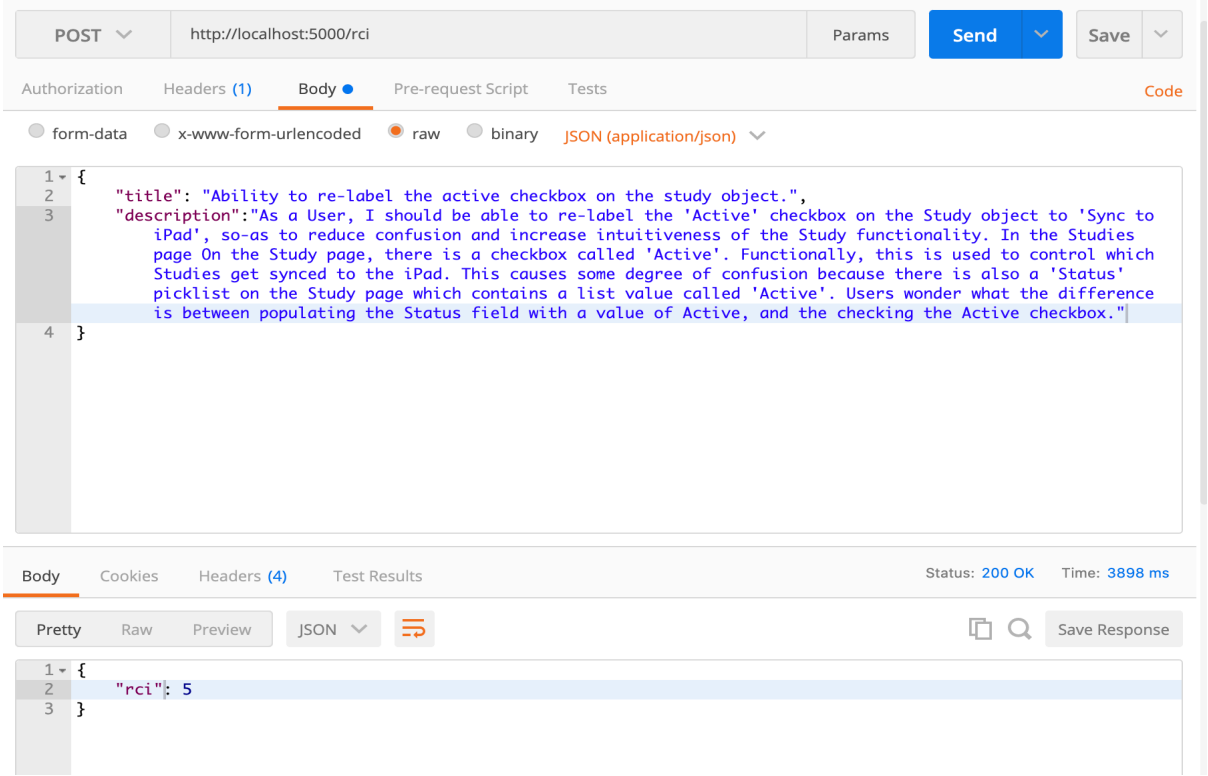


Figure 9 - System output of User story 1

In above mentioned scenario manually calculated RCI value and generated RCI value is same, since this user story has more details and clear explanation of the feature to be implemented.

User Story 2

Title

Remove the fields Category and Activity Type from the Territory Reasons Page.

Description

As a DevOps Admin I want to remove the fields Category and Activity Type from the 'Time off Territory Reasons' page as these fields are not relevant to Sales users when a new TOT Reason is created

RCI Value - 3

Figure 10 - User story 2

In this example (User story 2), RCI value is 3 which means requirement is somewhat clear. This scenario requirement is known, but details are unclear. Major assumptions are being made, which if invalid may lead to significant rework. Following the output from the system.

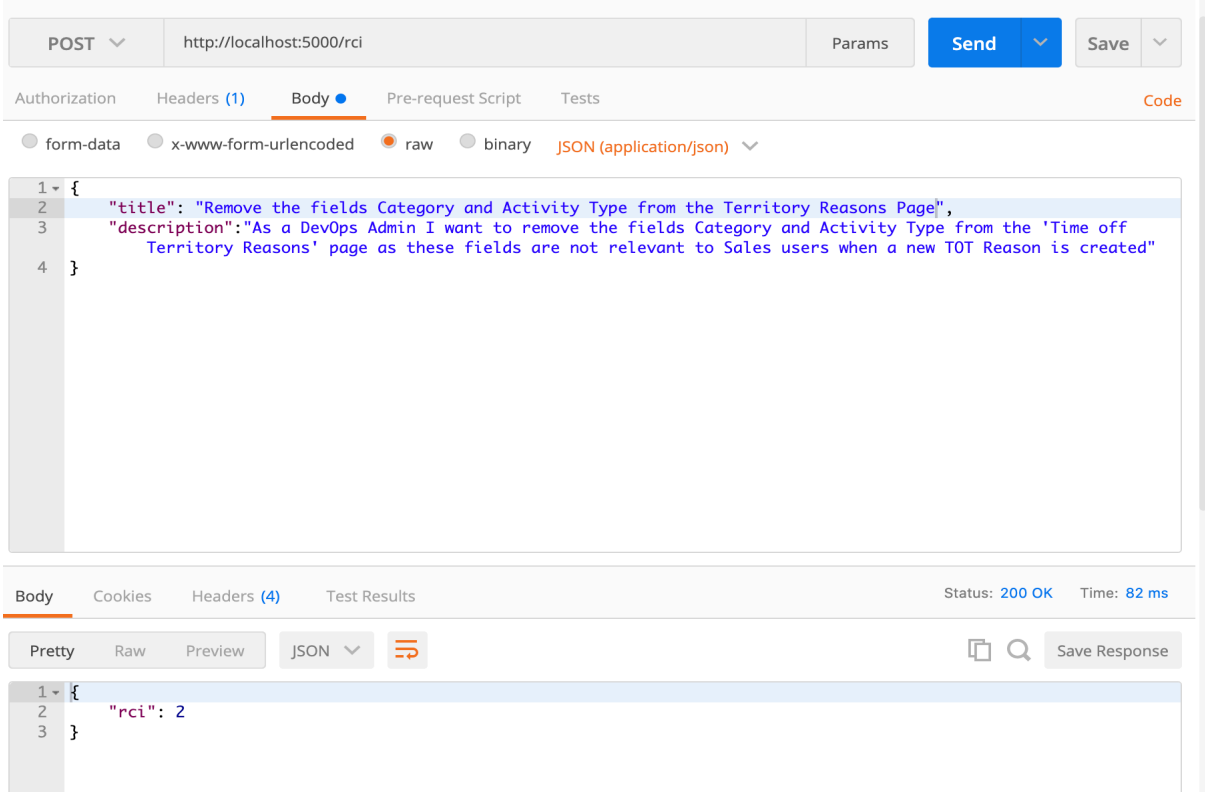


Figure 11 - System output of User story 2

In this scenario the manual RCI value and system generated RCI value is different. System identify this requirement as RCI value 2 requirement, which means requirement is an unclear idea, since this requirement has many number of keywords which is related to the domain. If a requirement contains many number of the domain specific keywords manual RCI calculation is relatively easier than automatic RCI calculation. Reducing the domain specific keywords as much as possible this the requirement can improve the RCI value of the requirement.

6.2. Observations

Following table contains some of the user stories (randomly chosen ten user stories) from data set, manually configured RCI values, system generated RCI values and percentage error. Percentage error of the value calculated with below formula.

$$Percentage\ Error = \frac{|\ \text{Approximate Value} - \text{Exact Value}\ |}{|\ \text{Exact Value}\ |} * 100\ \%$$

In this scenario Approximate value is Generated RCI and Exact value is Manual RCI

ID	User story title	Manual RCI	Generated RCI	Percentage Error
1	Update supported git version documentation	4	5	25%
2	Show bookmarks in changesets view	5	5	0%
3	When I search on the 'Parent Link' field, I want to see the sum of original estimate	2	2	0%
4	Show 'Complete Sprint' button when 'Hide menus' is enabled	4	5	25%
5	Add 'Cancel' button to add-on installation	1	1	0%
6	As a Rapid Board user I would like to be able to configure Quick Filters by shuffling through existing filters in JIRA	5	5	0%
7	Allow to order groups in the groups screen in Administration mode	2	1	50%
8	Updated Look and Feel for anonymous users	1	1	0%
9	Provide Support for Proxies using NTLM Authentication	4	4	0%
10	HipChat client for Linux doesn't correctly scale for display	2	2	0%

Table 4 - User stories and generated RCI values

From the above mentioned table we are able to see some observations on the results. Randomly took ten user stories from the data set compared the system generated RCI with manually calculated RCI. Seventy percentage (70%) (7 out of 10 user stories) of the results are accurate. Screenshot of results are in appendix section. Average error percentage of the result is 33.33%. With the large number of the results we are able calculate accurate error percentage and system success rate. Thirty percentage (30%) of user stories have different RCI values since the lack of domain knowledge in the system. When developer or a technical person manually computing the RCI, person has clear domain knowledge of the requirement and keywords in the requirement, but when we automating that type of scenario in this case returns different RCI value, since system doesn't have domain knowledge. If we think the system as a new person who has no prior knowledge of the software application or requirement, then that person won't be able understand and implement the requirement. That is the major reason of getting the thirty percentage (30%) of user stories have different RCI value.

Following are the results for the randomly chosen 200 user stories.

- Number of user stories randomly chosen – 200
- Accurate RCI value generated user stories – 174
- Different RCI value generated user stories – 26
- Success rate in percentage – (174/200) * 100 = 87%

As we can see in the results, success rate is increased from 70% to 87%. As sample size increase success rate is increasing.

7. Conclusion and Future Works

This chapter explains the outcome of the research work and future works to expand and improve the existing problems.

7.1. Conclusion

Requirement Clarity Index (RCI) is one of the quality measure to identify the requirement quality. Current approach followed in calculating RCI value requires more time and human involvement. There are several number of the researches have been done related to automating requirement quality measuring. Still there are some gaps for improvement in the area of requirement quality measuring. This research focus on identifying the research gaps in existing approaches and generating requirement clarity index automatically with improved methodology.

This research primarily focused on automating one of the software requirement quality measure known as requirement clarity index generation. So this research has fetched past understanding on requirement quality measuring and automating. In this research natural language requirement is summarized using text summarization techniques which returns summarized content with keywords are extracted. Quality score is assigned to the extracted keywords with the help of identified factors in the literature survey. The result of these approaches return a mapping table which helps to measure the requirement clarity index value for future requirements.

7.2. Future Works

There are plenty of software quality measures and metrics are available to measure quality of the requirement and which requires significant human involvement. This research was primarily focused on generating requirement clarity index which is one of the software quality measure used in the industries. Still the approach followed in this research can be extensible to cater other requirement quality measure generation. For huge set of data, there is a performance bottle neck in the mapping table generation. Improvement in the performance of the mapping table generation can be implemented in the future. Identifying more factors which affect the quality of the requirement will improve the scoring mechanism accuracy.

Current practice in the evaluation approach is to randomly split the data into approximately seventy percentage for training data and thirty percentage for test data. This practice of partitioning data leads to some issues such as class imbalance and sample representation issue. Since this approach has some issues, this research project use different variation of partitioning of data approach which semi-random data partitioning.

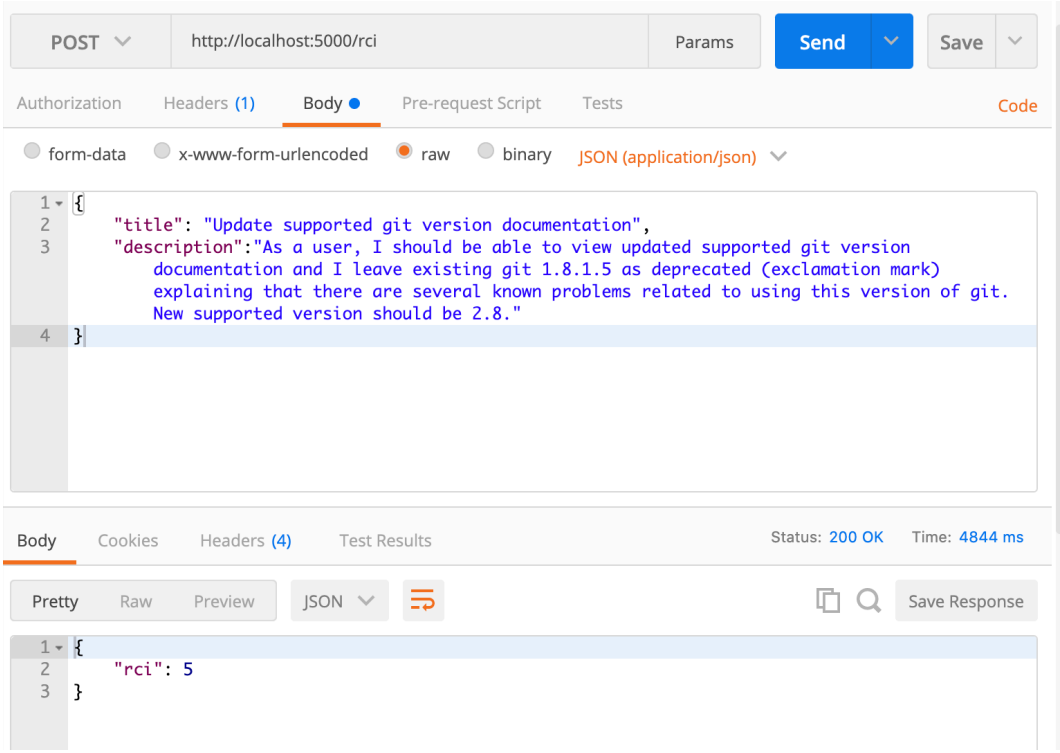
In semi-random data partitioning approach, data partitioning is done randomly on the basis of the original dataset towards getting a training dataset and a test dataset. In this approach, original dataset is divided into number of subsets, with each subset containing a class instance. Within each subset data partitioning into training and test datasets is done randomly.

So the future works are opened on additional metric enablement, current quality scoring mechanism, mapping table generation and evaluation approach.

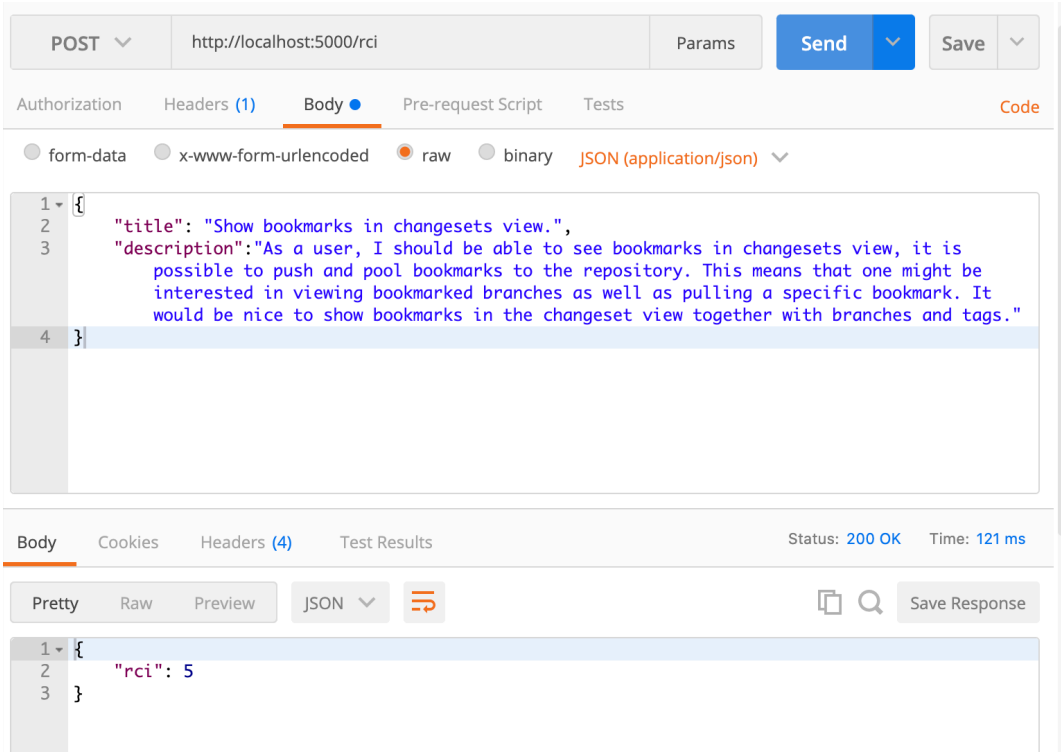
8. Appendix A

8.1. Results of user stories

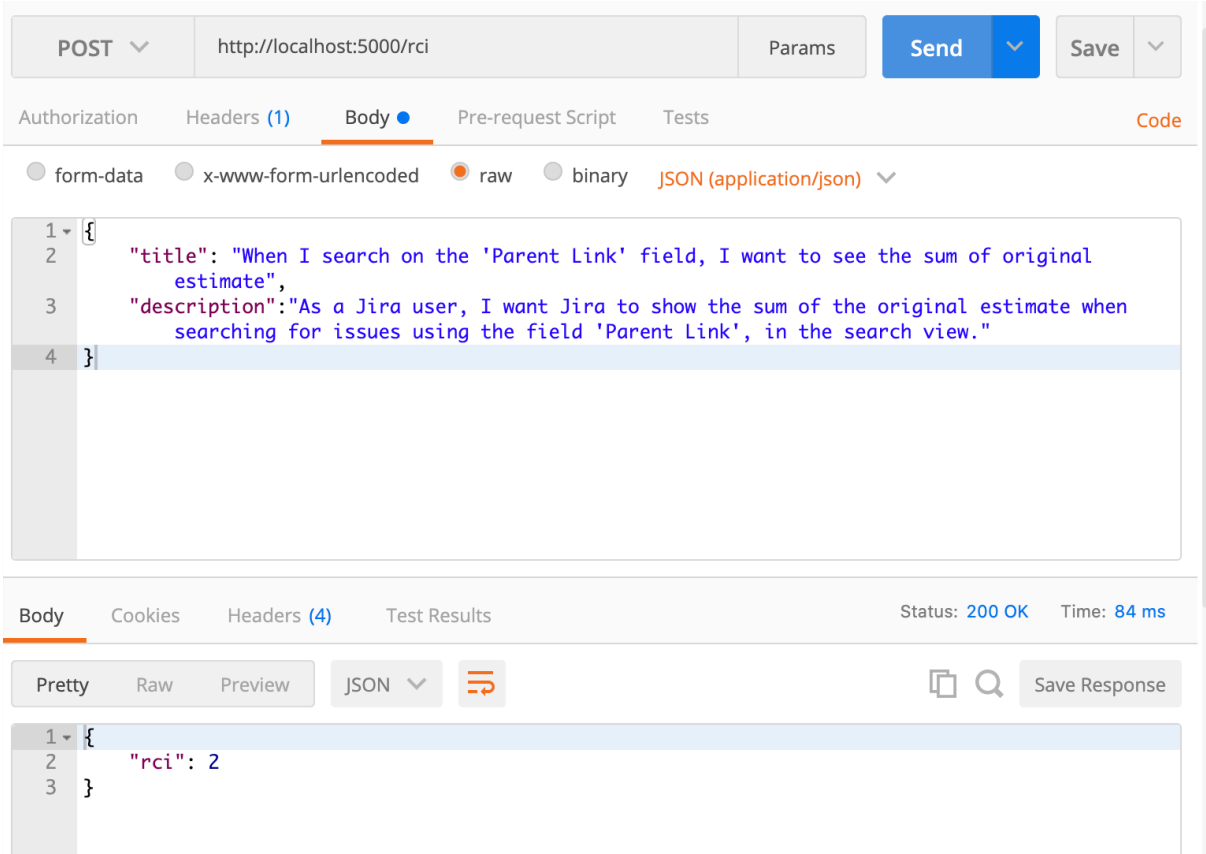
- User story 1



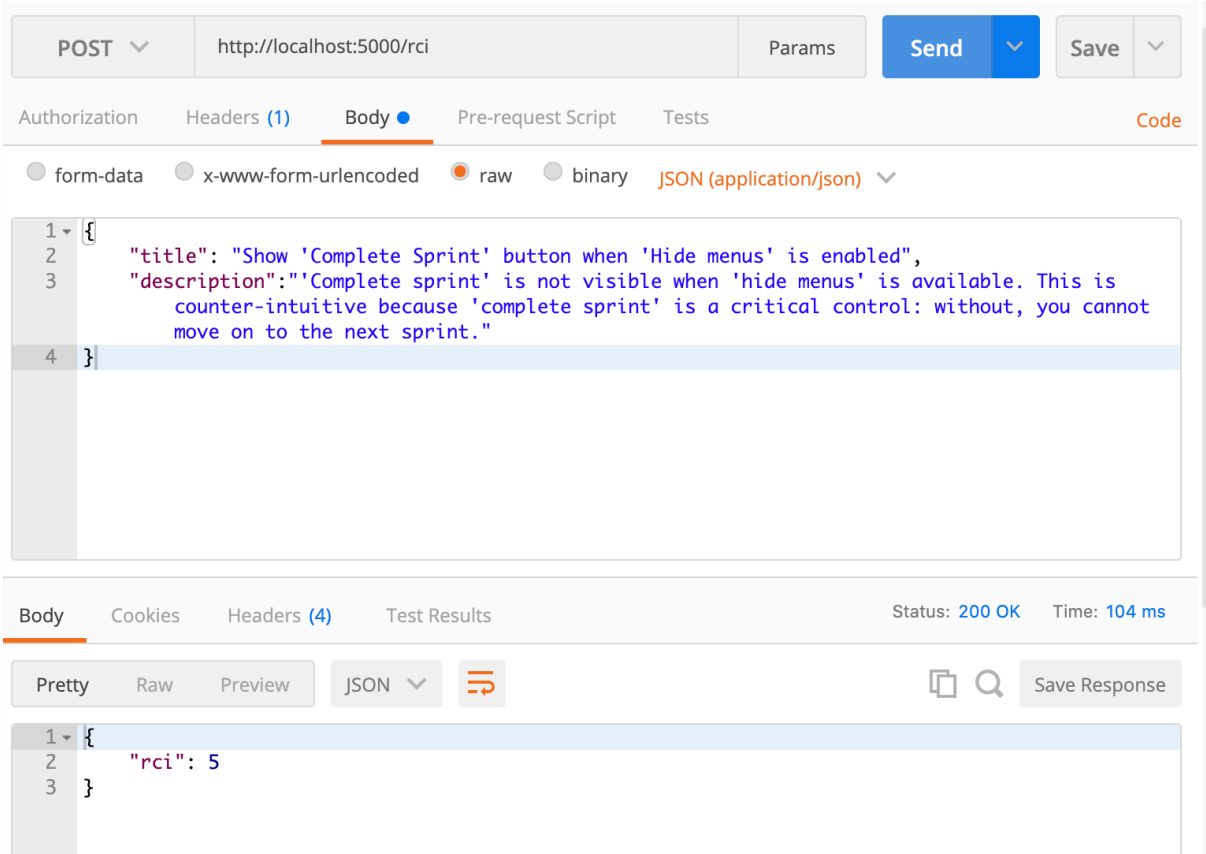
- User story 2



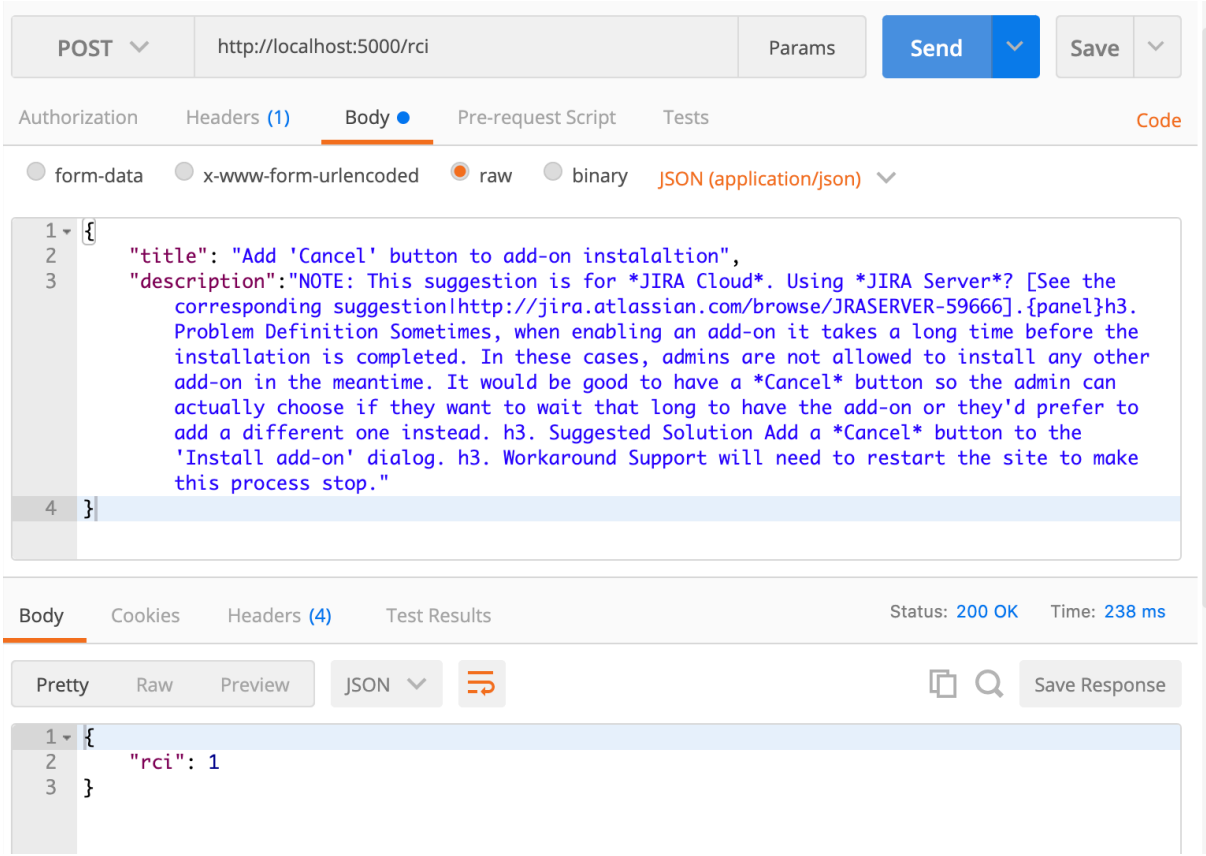
- User story 3



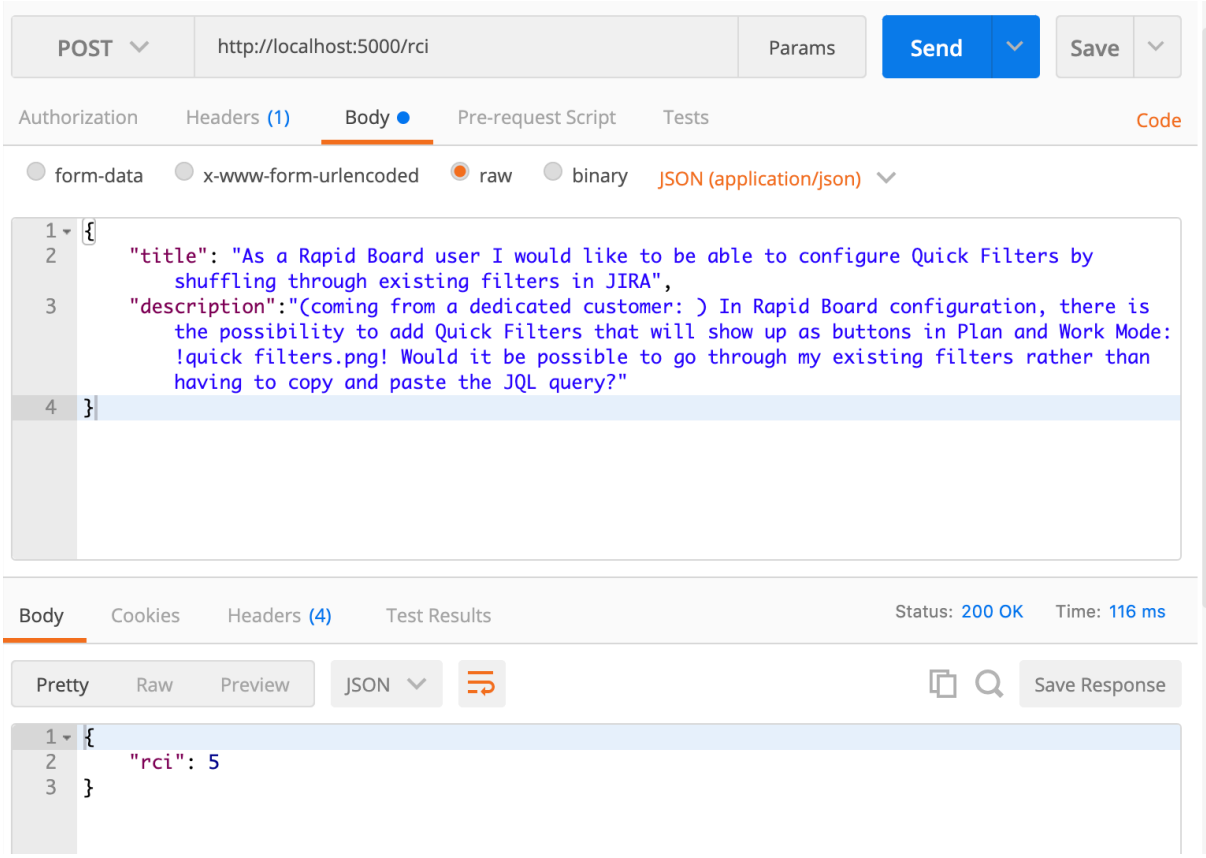
- User story 4



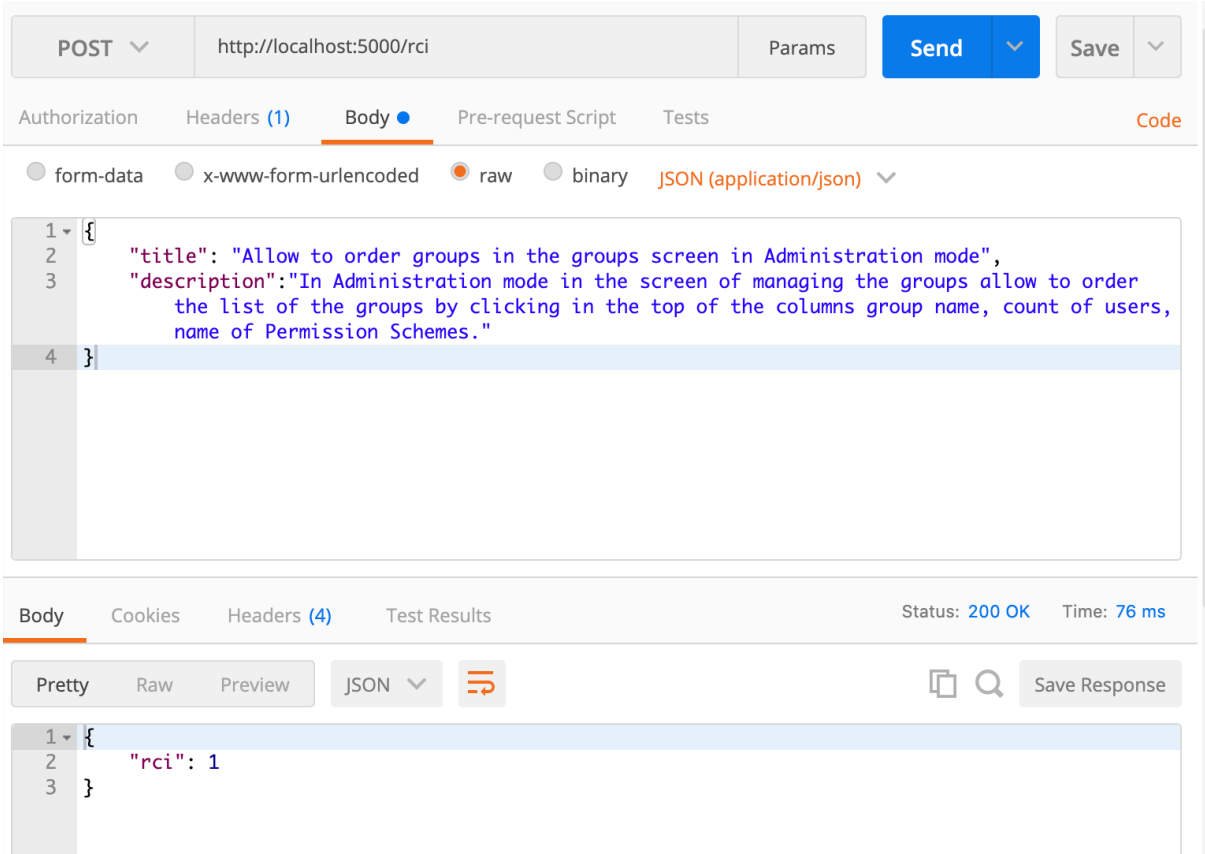
- User story 5



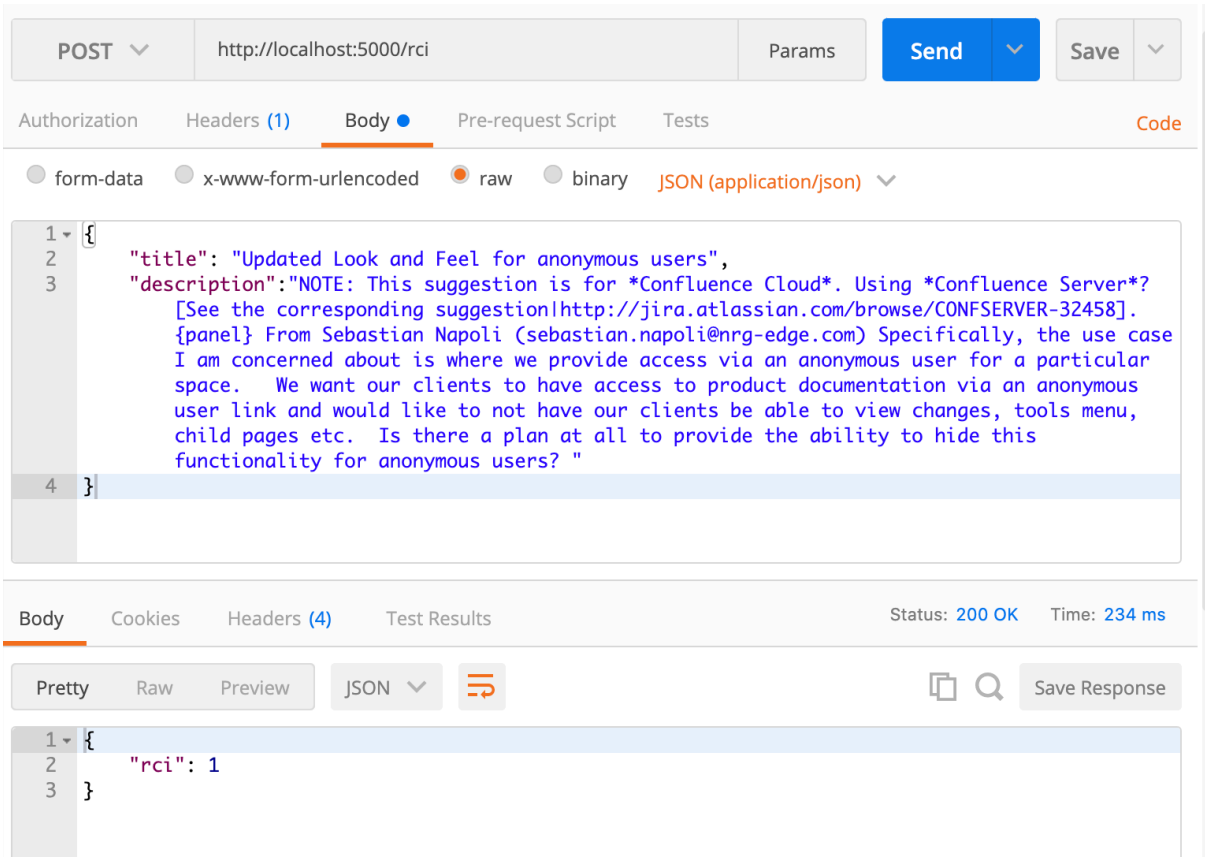
- User story 6



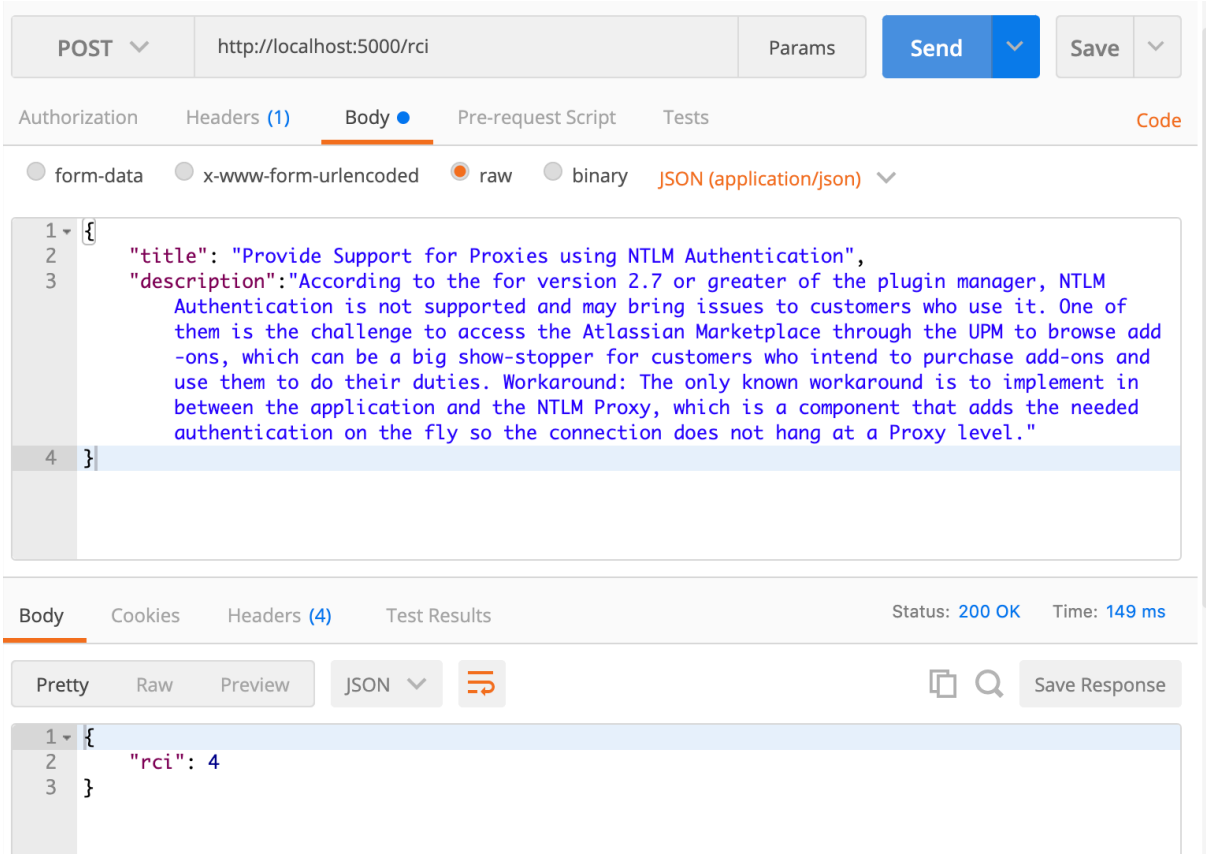
- User story 7



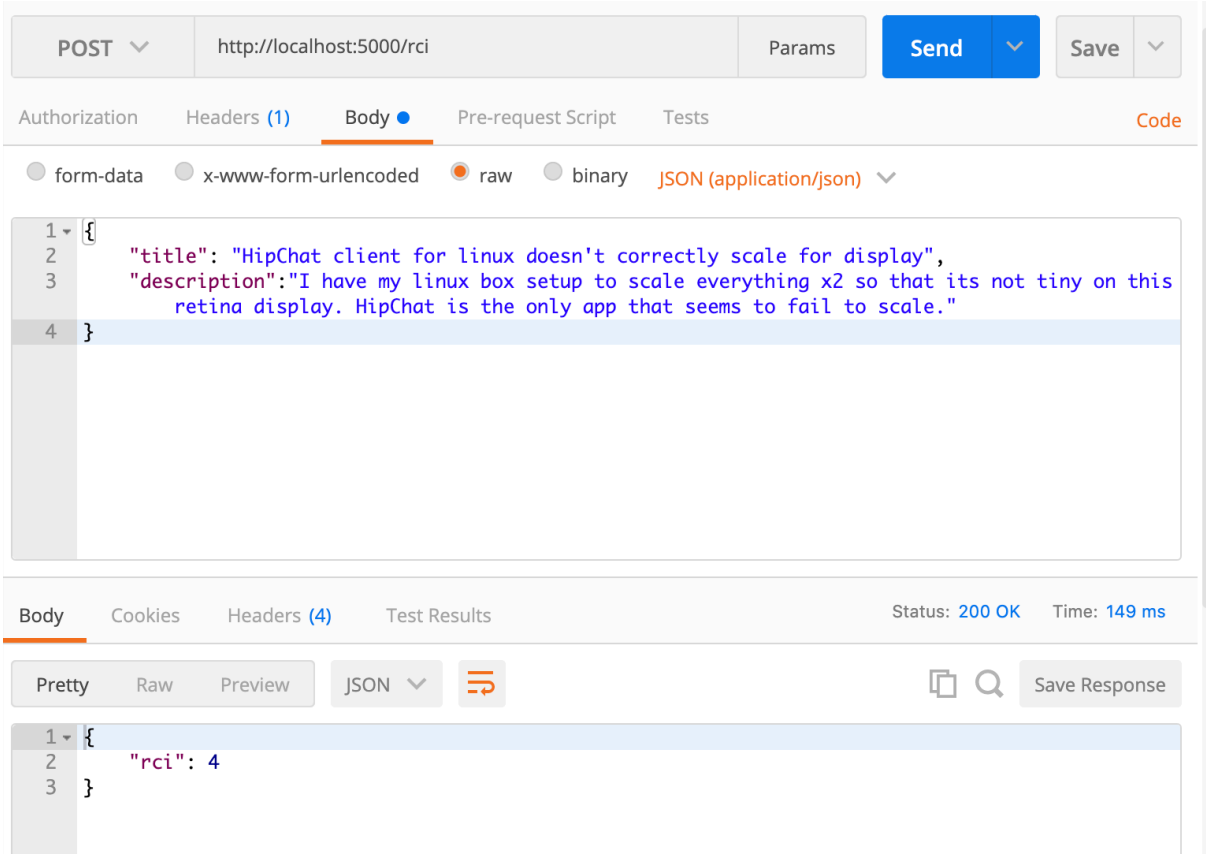
- User story 8



- User story 9



- User story 10



References

[1] "Software Engineering | Requirements Engineering Process," *GeeksforGeeks*, Jun. 17, 2018. https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/ (accessed May 15, 2020).

[2] "Traditional vs. Agile Software Development Method: Which One is Right for Your Project? - DZone Agile," *dzone.com*. https://dzone.com/articles/traditional-vs-agile-software-development-method-w (accessed Jan. 19, 2020).

[3] "What is AGILE? - What is SCRUM? - Agile FAQ's," *Cprime*. https://www.cprime.com/resources/what-is-agile-what-is-scrum/ (accessed May 15, 2020).

[4] "RUP (Rational Unified Process) Definition." https://techterms.com/definition/rup (accessed May 15, 2020).

[5] "SDLC - Agile Model - Tutorialspoint." https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm (accessed Oct. 06, 2019).

[6] S. Sedigh-Ali, A. Ghafoor, and R. A. Paul, "Software engineering metrics for COTS-based systems," *Computer*, vol. 34, no. 5, pp. 44–50, 2001, doi: 10.1109/2.920611.

[7] M. Rathnayake, "Project Management Online : RCI (Requirements Clarity Index)," *Project Management Online*, Dec. 14, 2014. http://pmonlineweb.blogspot.com/2014/12/rci-requirements-clarity-index.html (accessed Jan. 19, 2020).

[8] K. V. J. Padmini, "USE OF SOFTWARE METRICS IN THE AGILE SOFTWARE DEVELOPMENT PROCESS," p. 101.

[9] *Natural language processing of semitic languages*. New York: Springer, 2014.

[10] E. D. Liddy, "Anaphora in natural language processing and information retrieval," *Inf. Process. Manag.*, vol. 26, no. 1, pp. 39–52, Jan. 1990, doi: 10.1016/0306-4573(90)90008-P.

[11] M. Cohn, “User Stories and User Story Examples by Mike Cohn,” *Mountain Goat Software*. https://www.mountaingoatsoftware.com/agile/user-stories (accessed Jan. 18, 2020).

[12] “User Stories: An Agile Introduction.” http://www.agilemodeling.com/artifacts/userStory.htm (accessed Oct. 05, 2019).

[13] M. N. Hoda and Bharati Vidyapeeth’s Institute of Computers Applications and Management, Eds., *Proceedings of the 5th National Conference on Computing for Nation Development (10th - 11th March, 2011) INDIACom-2011: held in New Delhi*. New Delhi: Bharati Vidyapeeth’s Institute of Computer Applications and Management (BVICAM), 2011.

[14] R. R. Sud and J. D. Arthur, “Requirements Management Tools A Qualitative Assessment,” p. 19.

[15] A. M. Davis, *Software requirements: objects, functions, and states*, Rev. Englewood Cliffs, N.J: PTR Prentice Hall, 1993.

[16] E. E. Mills, “SEI Curriculum Module SEI-CM-12-1.1 December 1988,” p. 43.

[17] A. Mishra, A. Awal, J. Elijah, and A. AbdulG, “Automation of Requirement Analysis in Software Engineering,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 5, no. 5, p. 16.

[18] J. Györkös, “Measurements in software requirements specification process,” *Microprocess. Microprogramming*, vol. 40, no. 10–12, pp. 893–896, Dec. 1994, doi: 10.1016/0165-6074(94)90063-9.

[19] M. Nardini, F. Ciambra, F. Garzoli, D. Croce, D. D. Cao, and R. Basili, “3.1.2 Machine Learning technologies for the Requirements Analysis in Complex Systems,” *INCOSE Int. Symp.*, vol. 22, no. 1, pp. 371–385, Jul. 2012, doi: 10.1002/j.2334-5837.2012.tb01343.x.

[20] P. Resnik and J. Lin, “11 Evaluation of NLP Systems,” p. 26.

[21] D. N. Chin, “Empirical Evaluation of User Models and User-Adapted Systems,” p. 14.

[22] T. Tamai and T. Anzai, “Quality Requirements Analysis with Machine Learning:,” in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering*, Funchal, Madeira, Portugal, 2018, pp. 241–248, doi: 10.5220/0006694502410248.

[23] “Chainer – A flexible framework of neural networks — Chainer 7.1.0 documentation.” https://docs.chainer.org/en/stable/ (accessed Jan. 18, 2020).

[24] “Lean Agile Metrics for Scaled Agile Systems.” http://www.methodsandtools.com/archive/leanagilemetrics.php (accessed Jul. 06, 2019).

[25] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, “Requirement boilerplates: Transition from manually-enforced to automatically-verifiable natural language patterns,” in *2014 IEEE 4th International Workshop on Requirements Patterns (RePa)*, Karlskrona, Sweden, Aug. 2014, pp. 1–8, doi: 10.1109/RePa.2014.6894837.

[26] A. Mustafa, “Automated Natural Language Requirements Analysis using General Architecture for Text Engineering (GATE) Framework,” vol. 9, no. 3, p. 5.

[27] H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks, “GATE: a general architecture for text engineering,” in *Proceedings of the fifth conference on Applied natural language processing Descriptions of system demonstrations and videos -*, Washington, DC, 1997, p. 29, doi: 10.3115/974281.974299.

[28] A. Tripathy, A. Agrawal, and S. K. Rath, “Requirement Analysis using Natural Language Processing,” p. 10.

[29] K. M. Seddik and A. Farghaly, “Anaphora Resolution,” in *Natural Language Processing of Semitic Languages*, I. Zitouni, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 247–277.

[30] “Tokenization.” https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html (accessed Jan. 20, 2020).

[31] “Normalization (equivalence classing of terms).” https://nlp.stanford.edu/IR-book/html/htmledition/normalization-equivalence-classing-of-terms-1.html (accessed Jan. 20, 2020).

[32] G. Booij, “Inflection and Derivation,” in *Encyclopedia of Language & Linguistics*, Elsevier, 2006, pp. 654–661.

[33] “Stemming and lemmatization.” https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (accessed Jan. 20, 2020).

[34] “5. Categorizing and Tagging Words.” https://www.nltk.org/book/ch05.html (accessed Jan. 20, 2020).