


Project Name: Yoshi
Project Requirements

Version 1.0

Revision History

Date	Version	Description	Author
2014-11-10	1.0	Initial Draft	Martin Anev and Yuxing Chen

Table of Contents

1. Introduction

1.1 Purpose of this document

The purpose of this document is to introduce general functionalities of the project delivered by Team Yoshi in the Distributed Software Development done simultaneously in 'Politecnico di Milano' situated in Milan, Italy and 'Mälardalen University' situated in Västerås, Sweden.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction,* describes contents of this guide, used documentation during developing process etc.
- Section 2, *Overall Description*, describes different interfaces and high-level description of the domain
- Section 3, *Goals and Requirements*
- *Section 4, Use cases*

1.3 Intended Audience

The intended audience is:

- The customer of the project
- The supervisors of the project
- Yoshi Team
- All related stakeholders
- Any developer with interest to continue or improve the project

1.4 Scope

The document addresses the requirements of the Yoshi Project. The project is addressing problems of evaluating and supporting community types.

1.5 Definitions and acronyms

1.5.1 Definitions

Keyword	Definitions
Community	Social unit of any size that shares common values.
Open Source Community	Community that develops open source software.

1.5.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
NTR	Nothing to Report. There is no information to a specific topic available or necessary.
API	Application Programming Interface
G	Goal
FR	Functional Requirement

2. Overall description

This section describes the general factors that affect the software product and its requirements, providing background for specifying concrete requirements in the next sections of the document.

2.1 Product descriptive

The software product is completely independent from other systems, making the software product self-contained. However, in the future it may offer external interfaces to other products.

2.1.1 Interfaces

- External interfaces - The software product does not provide any external interfaces.
- *User interfaces* - The software product should have user interface that is graphical and interactive. The user should be able to visually understand and recon characteristics of communities as well as visualize them. The user should be able to focus on one characteristic or inspect multiple communities and their visualization at the same time, in the same screen.
- *Hardware interfaces* - The software product does not provide any hardware interfaces.
- *Software interfaces*- The software product should connect to Database Management System. It should be able to connect to Application Server. The system borrows the API of Grimoire[2] and the supplement VizGrimoire[3] registered under GPL v2[4] for legal use and extending it for the needs of Yoshi

2.2 High level description of the domain

The software product that will be delivered is Yoshi. Yoshi is a web application intended for helping people to measure, compute and study social community types. For instance, it will allow users to measure social and organizational characteristics of open-source communities.

2.2.1 General functionalities

Yoshi should provide general functionalities for managing:

- Profile
  Yoshi should manage basic data for registered users.
- Users
  Yoshi should manage registering, logging in/out of users.
- Connection
  Yoshi should be able to connect to instances from which it should gather the data.
- Measuring
  Yoshi should be able to measure metrics from the instances to which it connects
- Visualization
  Yoshi should be able to outputs the measured data.

2.2.2 General limitations

Yoshi will have the following limitations (probably developed in future versions):

- Statistics and Auditing for usage - The system will not provide any kind of statistics and analyses based on user interactions.
- External interfaces to social networks or sharing options - The system will not offer any kind of sharing (by the mean of social networks and email) options.
- Community and reviews - Comment, recommendations, reviews will not be supported on this stage of the system.

3. Goals and Requirements

This section describes the goals, functional and non-functional requirements that affect the software, providing background for specifying concrete use cases and description of the desired functionality.

3.1 Goals

We are giving the general goals of the project following.
[G1] Allow users to measure community social and organizational characteristics
[G2] Allow users to compute social network that makes characteristics explicit
[G3] Allow users to study type and characteristics against performance metrics to evaluate fitness
[G4] Allow users to understand the characteristics
[G5] Allow users to visualize the characteristics
[G6] Allow users to focus on one characteristic or inspect multiple communities and their visualization at the same time, in the same screen.

3.2 Requirements

3.2.1 Functional requirements

3.2.1.1 Managing Profiles

[FR1] The system should distinguish profiles in two categories: non-registered users, registered users.
[FR2] The non-registered users should be able to register to the system
[FR3] The registered users should be able to Login.
[FR4] The registered users should be able to Logout.
[FR5] The registered users should be able to modify personal information.
[FR6] The users should be able to view his/her saved results from computation

3.2.1.2 Managing Non-registered users

[FR7] Non-registered users should be able to interact with the system - view already computed metrics that are presented from the product

3.2.1.3 Managing registered users

[FR8] Registered users should be able to compute communities' type by offered APIs from the system
[FR9] Registered users should be able to compute communities' type by offered DB
[FR10] Registered users should be able to understand what is the organisational type of the community that he/she is observing
[FR11] Registered user should be able to generate a report with all relevant characteristics for open source community
[FR12] Registered user should be able to correlate community type and characteristics with known performance metrics for open-source

3.2.1.4 Managing Connection

[FR13] The product should be able to connect to instances from which it should gather the data

3.2.1.5 Managing Computation

[FR14] The product should be able to grasp the data from the community that is researched by the user
[FR15] The product should be able to compute the grasped data
[FR16] The product should be able to store the computed data
[FR17] The product should be able to visualize the stored data
[FR18] The product should be able to visualize data provided by the user
[FR19] The product should be able to compare characteristics on multiple communities
[FR20] The product should be able to visualize the comparison of characteristic on multiple communities and at the same time, in the same screen

3.2.2 Priorities of the Functional Requirements

The Functional Requirements are assigned with different priority levels as follows:
A-Highest priority
B-High
C-Medium
D-Low Priority

[FR1] The system should distinguish profiles in two categories: non-registered users, registered users.	C
[FR2] The non-registered users should be able to register to the system	C
[FR3] The registered users should be able to Login.	C
[FR4] The registered users should be able to Logout.	C
[FR5] The registered users should be able to modify personal information.	D
[FR6] The users should be able to view his/her saved results from computation	C
[FR7] Non-registered users should be able to interact with the system - view already computed metrics that are presented from the product	A
[FR8] Registered users should be able to compute communities' type by offered APIs from the system	A*
[FR9] Registered users should be able to compute communities' type by offered DB	A*
[FR10] Registered users should be able to understand what is the organisational type of the community that he/she is observing	A
[FR11] Registered user should be able to generate a report with all relevant characteristics for open source community	A
[FR12] Registered user should be able to correlate community type and characteristics with known performance metrics for open-source	A
[FR13] The product should be able to connect to instances from which it should gather the data	A
[FR14] The product should be able to grasp the data from the community that is researched by the user	A
[FR15] The product should be able to compute the grasped data	A
[FR16] The product should be able to store the computed data	B
[FR17] The product should be able to visualize the stored data	B
[FR18] The product should be able to visualize data provided by the user	C

[FR19] The product should be able to compare characteristics on multiple communities	C
[FR20] The product should be able to visualize the comparison of characteristic on multiple communities and at the same time, in the same screen	C

4. Use Cases

4.1 Actors

4.1.1 Guest:

A person that has not registered and can only exploit basic functionalities such as looking the demo for famous known communities analysis provided by system. The guest is limited for some function such as not able to operate the function of computing communities.

4.1.2 User

A person that has fully functions provided by the system such as select a community to analyse and report the relevant characteristics. The registration is necessary and the system may record the habits (data) of the users.

4.1.3 Administrator

A person that is responsible to update the contents of the system such as updating the demo of computed communities.

4.2 Use cases

4.2.1 Specific use case

Procedure 6,7 and 8 are optional methods to compute the communities, in other words, at least one of them will be implemented. We also giving the priority to each use case
1 register (D)
2 log in (D)
3 log out (D)
4 update profile (D)
5 view demo (C)
6 computing communities' type by offered APIs (A)
7 computing communities' type by offered DB (C)
8 computing communities' type by scratching (C)
9 compare communities' characteristics (choose different communities with the same (common) characteristics) (B)
10 visualize computed characteristics (A)

4.2.2 Use case diagram

The following part discusses the use case diagram. Note that one of the actors (administrator) is not included for two reasons:
1 Higher priority is considered to be the computation and metric functionalities, not the support of different type of accounts. Profile functionalities could be introduced as extension of this product.
2 The functions and requirement the administrator needs does not depend on the functionality of the product.

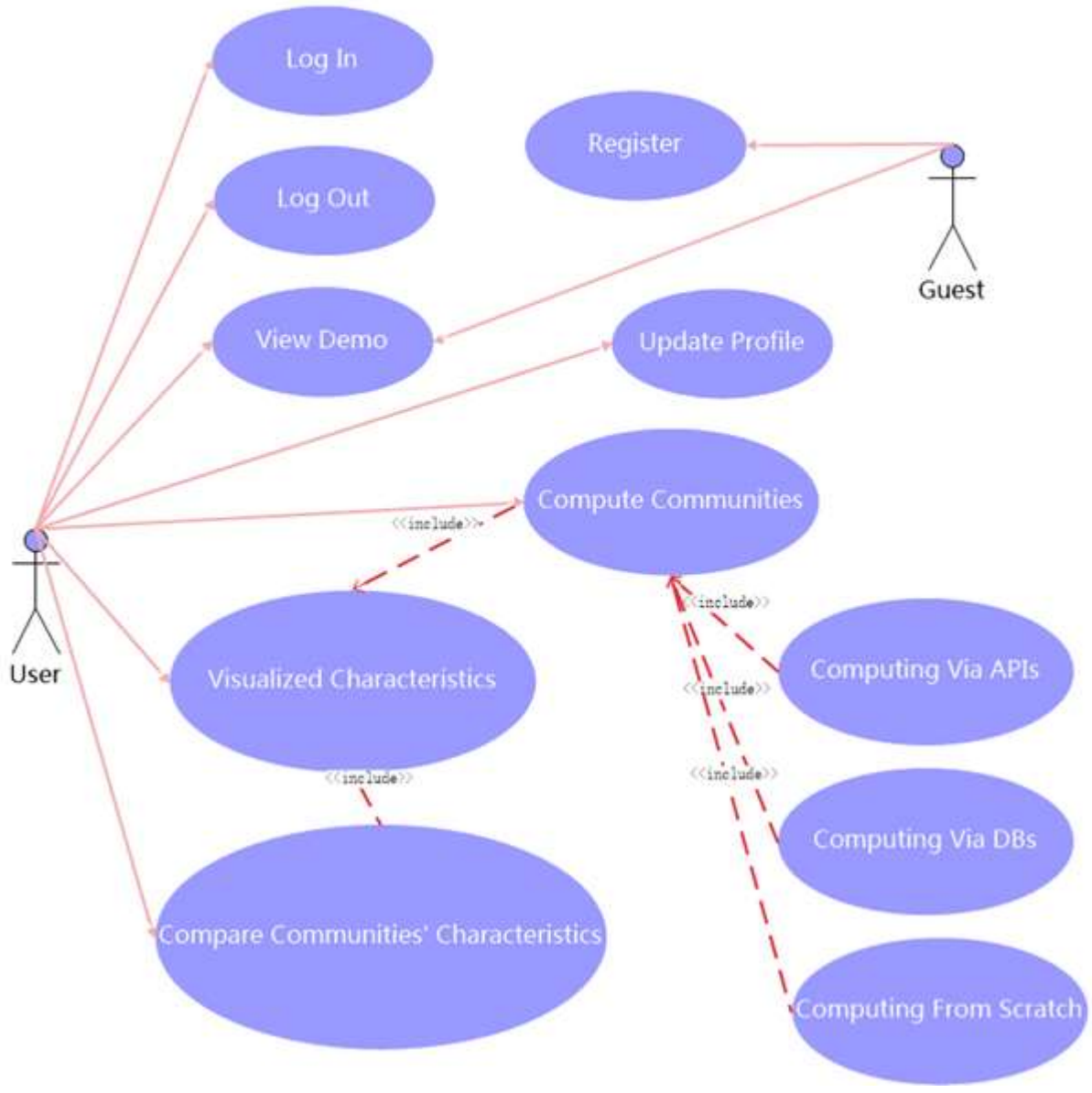

Figure 1 General Use Case Diagram

4.2.3 Description of use-case

In this section is discussed the detailed description with sequence diagram to the various use-cases related to the main use-case diagram defined above. For simplicity and readability, here are presented sequence diagram of the more complex use cases and the more simple ones are only described in textual form.

4.2.4 Lists of Tables

Table 1. Use Case: Register

Name	Register
Actors	Guest
Entry conditions	The guest opens the register page
Flow of events	The guest opens the home page and clicks "register" The system shows to the guest the register page The guest fills the form of the needed information (e.g. E-mail address; password; name; etc.) for registering. The guest finishes the form and click "register" The system shows guest whether he succeeds or not
Exit conditions	No Exit condition
Exceptions	The form will not be submitted if required information remains empty The form will not be submitted if the e-mail address is already registered.

Table 2 Use Case: Log in

Name	Log in
Actors	User
Entry conditions	The user has successfully registered to the system
Flow of events	The user opens the home page The system shows to the user the page The user enters the E-mail address and password in the provided input form. The user clicks "log in" The system shows the personal homepage

Exit conditions	User clicks “log out”
Exceptions	The form will not be submitted if the required information remains empty The system will not jump to homepage if the password is not corresponding

Table 3 Use Case: Log out

Name	Log out
Actors	User
Entry conditions	The users have successfully logged into the system
Flow of events	The user opens any page The user clicks “log out” button which is usually provided in the top-right side of the page
Exit conditions	No Exit condition
Exceptions	The system still has the cookie if user simply closes the page

Table 4 Use Case: Edit profile

Name	Edit profile
Actors	User
Entry conditions	The users have successfully logged into the system and in the home page
Flow of events	The user clicks “edit profile” The system shows the page to the user The user edits the profile and saves The system saves and jumps to home page
Exit conditions	The user quits the activity The user finishes the page and clicks Save. The system saves the changes jumps to the home page.
Exceptions	The data source are not able to get

Table 5 Use Case: View Demo

Name	View demo
Actors	User and guest
Entry conditions	The users have successfully logged into the system and in the home page
Flow of events	The user clicks “view demo” The system shows the list of demo The user chooses one to see the visualized result The system jumps to specific demo page
Exit conditions	The user quits the activity The user clicks to continue and the system jumps to the specific page
Exceptions	No exceptions

Table 6 Use Case: Computing communities' type by offered API

Name	Computing communities' type by offered APIs
Actors	User
Entry conditions	The users have successfully logged into the system and the computing methods page is opened
Flow of events	The user clicks "APIs" method to compute the communities The system shows the page to user The user needs to fills the form of needed information (which API method to be used e.g. Grimoire(provided by system); which communities to be computed e.g. github(provided by system); etc) The system returns the text(raw) results and ask user if they are needed to visualize.
Exit conditions	The user quits the activity The user finishes the page and clicks to continue and the system jumps to the result page
Exceptions	The data source are not able to get

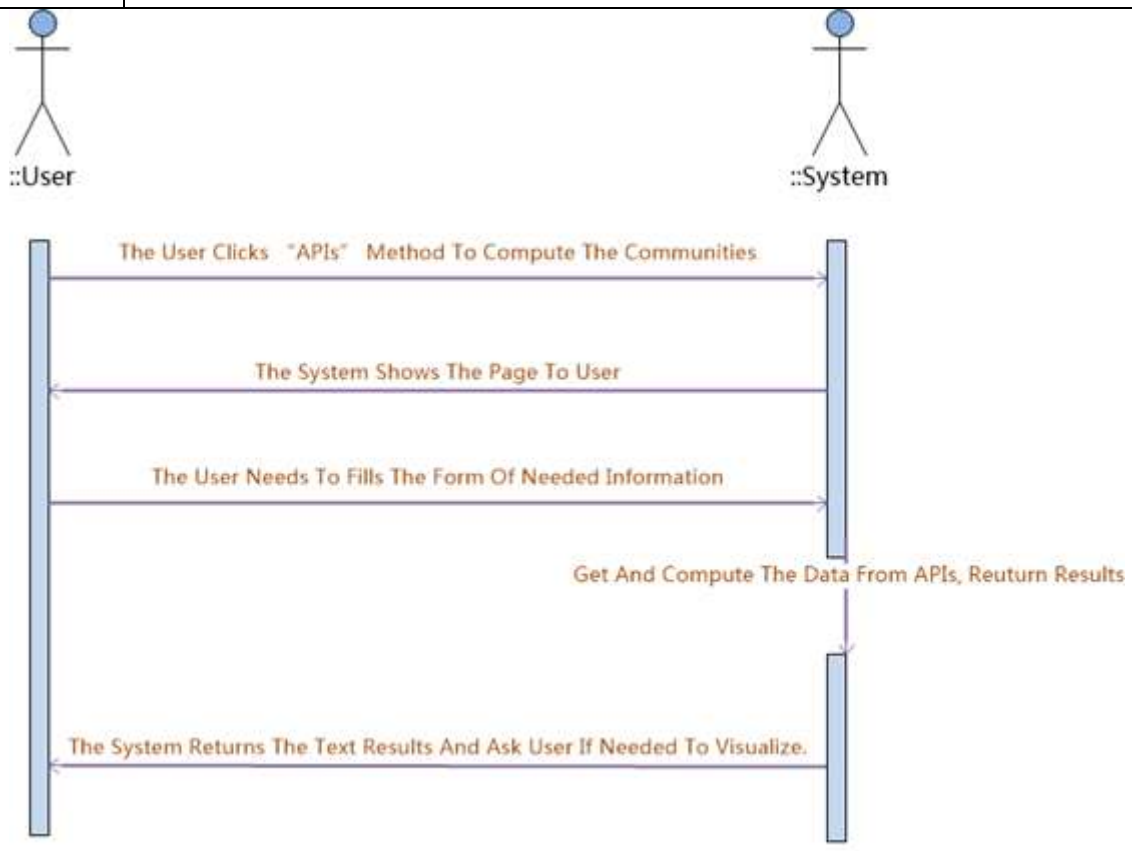

Figure 2 Sequence diagram computing communities' type by offered APIs

Table 7 Use Case: Computing communities' type by offered DB

Name	Computing communities' type by offered DB
Actors	User

Entry conditions	The user has successfully logged into the system and the computing methods page is opened
Flow of events	The user clicks “DB” method to compute the communities The system shows the page to user The user needs to fill the form of needed information (which open-source DB to be used (provided by system); which communities to be computed e.g. Github (provided by system); etc.) The system returns the text (raw) results and ask user if they are needed to visualize.
Exit conditions	The user quits the activity The user finishes the page and click to continue and the system jumps to the result page
Exceptions	The data source are not able to get

Table 8 Use Case: Computing communities' type from scratch

Name	Computing communities' type from scratch
Actors	User
Entry conditions	The user has successfully logged into the system and the computing methods page is opened
Flow of events	The user clicks "scratch" method to compute the communities The system shows the page to user The user need to fills the form of needed information (which scratch tools to be used (provided by system); which communities to be computed e.g. Github(provided by system); etc.) The system return the text (raw) results and ask user if they are needed to visualize.
Exit conditions	The user quits the activity The user finishes the page and click to continue and the system jumps to the result page
Exceptions	The scratch tools are not able to get the data

Table 9 Use Case: Compare communities characteristics

Name	Compare communities characteristics
Actors	User
Entry conditions	The user has successfully logged into the system and the computing methods page is opened
Flow of events	The user clicks "Compare Communities" to compute the communities The system shows the page to user The user needs to fill the form of needed information (which computed method to be used (provided by system); which communities to be compared; which common characteristics to be compared; etc.) The system returns the text (raw) results and ask user if they are needed to visualize.
Exit conditions	The user quits the activity The user finishes the page and click to continue and the system jumps to the result page
Exceptions	The data source are not able to get

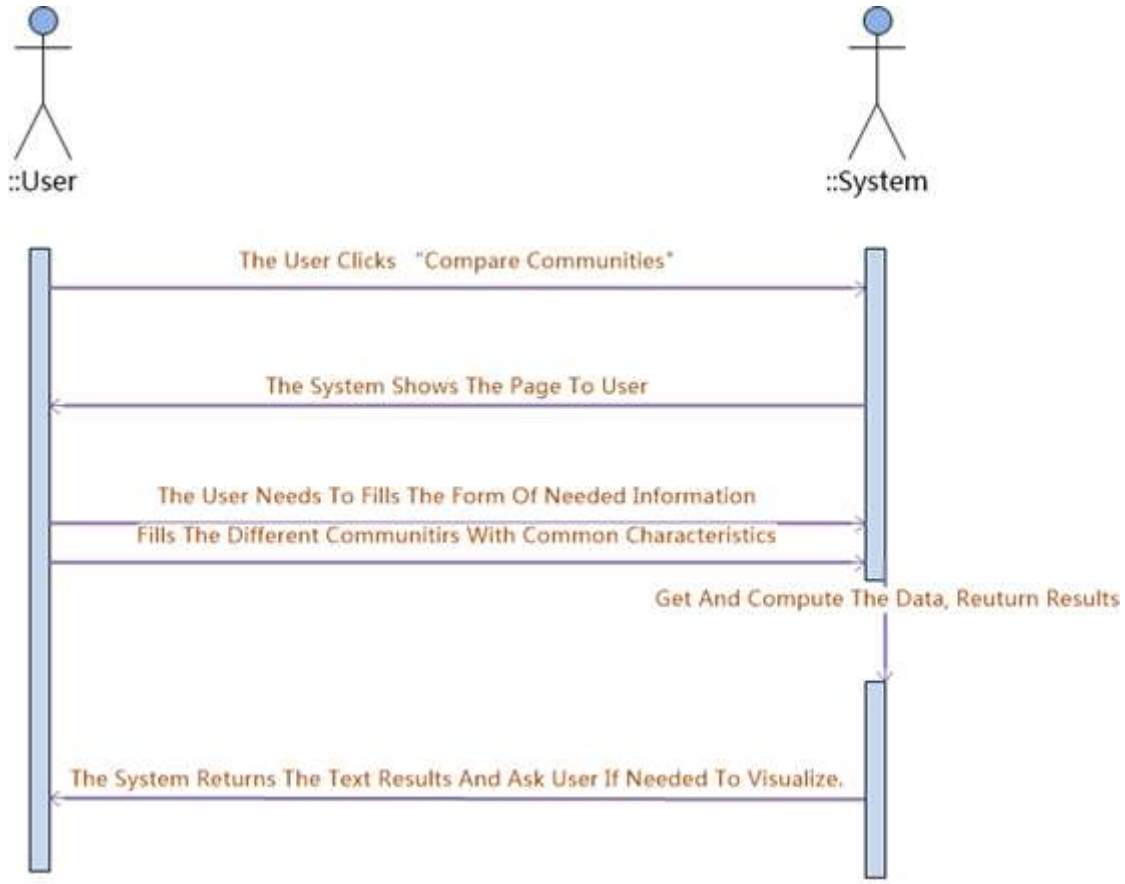

Figure 3 Sequence diagram Compare communities characteristics

Table 10 Use Case: Visualize computed characteristics

Name	Visualize computed characteristics
Actors	User
Entry conditions	After the page computed the system shows the result page
Flow of events	The system shows the result page The user clicks “visualize the result” The system shows the visualize result
Exit conditions	The user quits the activity The user finishes the page and clicks to continue and the system jumps to the visualization page
Exceptions	No exception

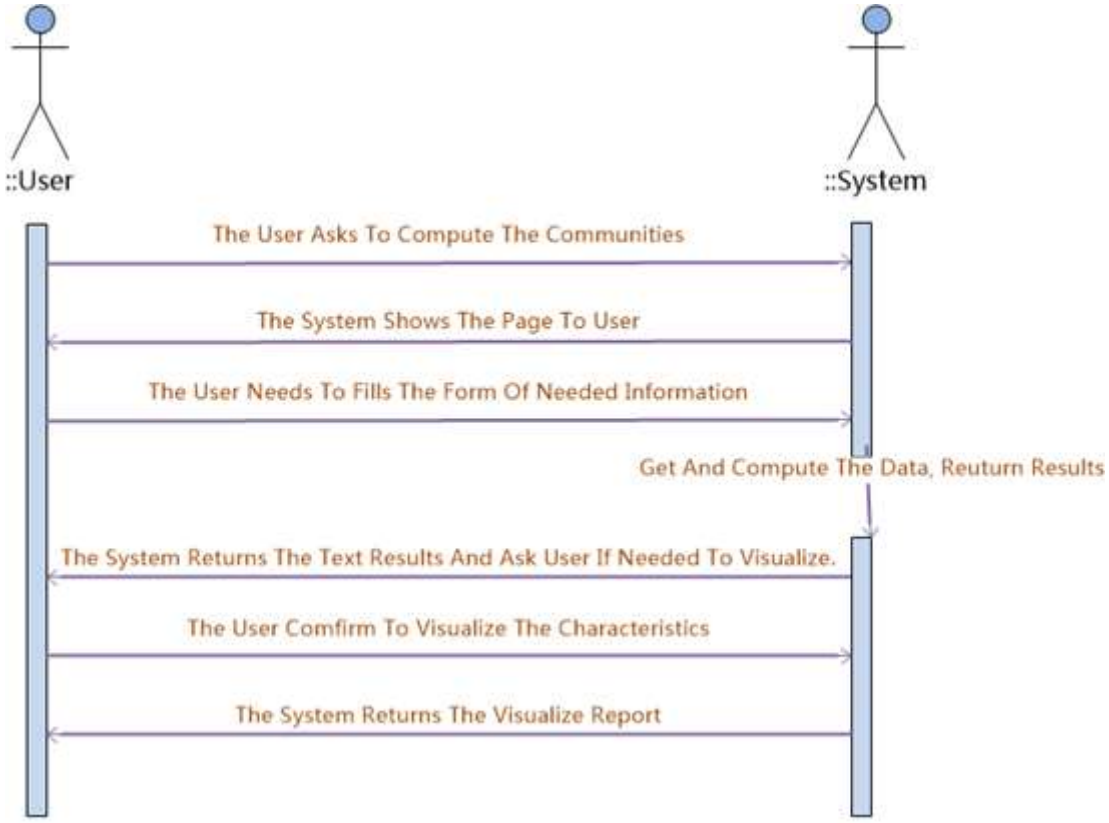

Figure 4 Sequence diagram: Visualize computed characteristics

5. References

1. IEEE Recommended Practice for Software Engineering Requirements Specifications
2. [2] MetricsGrimoire - https://metricsgrimoire.github.io/
3. [3] VizGrimoire - http://vizgrimoire.bitergia.org/
4. [4] GPL v2 - http://www.gnu.org/licenses/gpl-2.0.html