Software Requirements Specification

for

SNA4Slack

SER 517 – Software Factory I



Prepared by Group 3:

Abhimanyu Dogra (adogra4)

Aman Srivastava (asriva26)

Ishan Dikshit (idikshit)

Nikhil Tibrewal (ntibrewa)

Sanchit Narang (snarang2)

Shuchir Inamdar (sinamda2)

11/30/2017

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Product Scope 1

1.4 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

# Introduction

## Purpose

The purpose of this document is to serve as the requirements specification for SNA4Slack web application, which is currently under development. This SRS covers the currently developed modules of the application which are: the database, graph generation/analysis module, REST application server hosted on the Cloud and client-side home-page.

## Intended Audience and Reading Suggestions

This target readers for this document is our sponsors, course instructor and teaching assistant.

## Product Scope

SNA4Slack is a web application with the primary purpose of presenting several Social Network Analysis related data visualizations to the user. The user can be an employee of a particular team that uses Slack as their inter-team as well as intra-team conversations. These data visualizations will provide valuable insight by helping the user understand the characteristics of their communication network. Organizations will benefit from this application by understanding the vital components of their communication network and taking measures like enforcing steps to alleviate the load of densely connected components of the graph.

Any team can utilize this application by archiving their Slack communication data and providing the application with a link to their slackarchive.io archive.

## References

1. Issue tracking board: [link](https://waffle.io/aman-srivastava/SNA4Slack)
2. SCORE 2018: [link](http://score-contest.org/2018/index.php)

# Overall Description

## Product Perspective

This project is being developed for graduate level course SER 517 – Software Factory I at Arizona State University, Polytechnic. This is a mandatory course required for the completion of the Software Engineering degree and is an alternative to graduate thesis. The instructor of this course is Professor Robert Heinrichs who decided the list of projects that the students were assigned to work on. The project idea belongs to Professor Filippo Lanubile from University of Bari, Italy, who is the sponsor of this project. Professor Fabio Calefato, also from UoB, is the co-sponsor of this project.

This project is one of the contending software applications in SCORE 2018 in Gothenburg, Sweden. The SCORE Contest is aimed at promoting and fostering software engineering in universities worldwide. Student teams from all over the world participate in a competition for students from undergraduate to master's level. Each team will develop a software project chosen from a list of projects proposed and sponsored by program committee members. The final deliverable is a report and an accompanying executable system. Evaluation will based on the quality of all aspects of the software engineering process followed, as well as the resulting system. In order to accommodate a wide range of academic calendars, the SCORE 2018 Contest will run from May 2017 to January 2018. Finalist teams will be invited to ICSE 2018 in Gothenburg, Sweden.” [2]

## Product Functions C:\Users\dogra\Desktop\architecture.png

The currently developed modules perform the following functions:

**Crawler:**

* Retrieve data from slackarchive.io.
* Store data in csv format.

**Back-end logic / Graph Generator**:

* Parse .csv data generated by the crawler.
* Create a graph using the data.
* Compute Social Network Analysis metrics on the data.
* Construct a JSON format which can be understood by front-end data visualization libraries.

**Server / REST API:**

* Expose REST endpoints that provide the front-end with data analysis and visualizations

**Front-end:**

* Present a form where the user can specify what kind of visualizations do they want to view.
* Present the requested visualizations to the user.

**Cloud Platform**

* Host application server
* Store graph JSON in Cassandra database

## User Classes and Characteristics

**General public**: This class consists of users who access the web application but do not have belong to any team that uses Slack for conversations. This user may or may not have a professional background. This user will be interested in analyzing the data visualizations of teams who have publicly available Slack data. This user may be interested in observing the general characteristics of a Slack communication network.

**Team members**: This class consists of users who belong to a certain professional team which uses Slack as their communication media. This is the most important class of users as this application is intended to provide greatest benefit for this class of users. These users might not have engineering background, but they have knowledge about the components of the data visualizations. For example, an employee of the team knows who the nodes of the graph represent.

**Evaluators**: This class of users consists of the project sponsors, instructor and the teaching assistant of the course. They have a strong technical expertise and their primary aim while using the application is to evaluate its functionality and elements related to the development process.

## Operating Environment

* The server and database are hosted on the Google Cloud Platform.
* The front-end can be viewed by any machine that can run latest versions of the modern browsers like Firefox, Chrome and Safari.

**Development and Testing Hardware Requirements:**

Personal Computer/MAC with the following minimum specifications

* 1. Intel® Pentium® 4 or higher/AMD Athlon® 64 processor (2 GHz or faster)
  2. Microsoft® Windows® 7 with Service Pack 1, Windows 8, or Windows 8.1, Ubuntu (14.04 or higher) or MacOS
  3. 2 GB of RAM (8 GB recommended)
  4. 2 GB of available hard-disk space for installation; additional free space required during installation (cannot install on removable flash storage devices)
  5. 1024x768 display (1280x800 recommended) with 16-bit color and 512MB of VRAM (1 GB recommended)\*\*
  6. OpenGL 2.0–capable system
  7. Internet connection and registration are necessary for required software activation, validation of subscriptions, and access to online services.

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

* The data required for the Social Network Analysis is not easily available. The current resource for the project is slackarchive.io, as suggested by the sponsors. This website does not provide any API which can be used to retrieve the data. Hence the project has a python scrapper which retrieves data from the website by crawling through it and parsing its JSON. This implies that the data acquisition is not robust and has several issues. For example, in order to identify “@” annotated reference strings in conversation message strings, the graph generator module uses regex. Any string that starts with ‘@’ character will be considered as a reference, hence resulting in noise in the data. To avoid incorrect analysis, the application addresses this issue by only considering the users which have sent a message somewhere in the conversation, because the sender data has a separate field in the slackarchive.io JSON.
* The back-end strictly relies on Python 2.7.5 for development due to Python 3.x still being unstable for general development and compatibility issues.

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>