Software Requirement Specification for Tracking Interconnected Facebook Links

ELEN4009 - Software Engineering
Julian Zeegers (704582)
Joseph Gage (751052)
James Allingham (672732)
Nathan Haag (873666)

1 Introduction

The Tracking Interconnected Facebook Links project is a project that is intended to visualize the links and connections of a Facebook user with other users. Initially, these visualization are focused on identifying the relationships of a Facebook user (and the end user of this product) with the network of friends this Facebook user has. The visuals are also intended to further illustrate the relationship connections of the users friends of friends and an overview of the user's friend network. There will be more features that could potentially be added to this tool and therefore the solution must be dynamic and flexible. This document describes the software requirements that will ensure that the end product is of the highest quality, produced most efficiently and is created as close to the requirements as possible.

1.1 requirements

The requirements for the application are that it:

1. consists of a front-end that runs client side and a back-end that runs server side
2. has user friendly friendly front-end which is easy to use for non-technologically inclined people
3. is scalable - with a back-end that supports multiple clients
4. is fast and responsive leading to a pleasant user experience
5. provides attractive and useful visualisations of Facebook data which allow the user to explore their Facebook networks in an intuitive and practical manner leading to new and interesting discoveries being made
6. is extensible - allowing for additions and modifications to be made quickly and with ease
7. is secure - keeping the users personal data safe and respecting their privacy
8. allows flexibility for data acquisition - the application should allow users to upload their own data or access their data via the Facebook API

1.2 System Overview

This project will be made up of various software systems with the back-end and front-end working together to form a dynamic visualization program. The overview of the entire software system is illustrated in Figure 1. This figure shows how the various components of the system interact with each other. Figure 1 shows how the client (using a web browser) interacts with an Apache server and the Django framework. It also shows that the Neo4j database provides data to the framework and is then sent to the web browser (through the Apache server) to create visualizations.

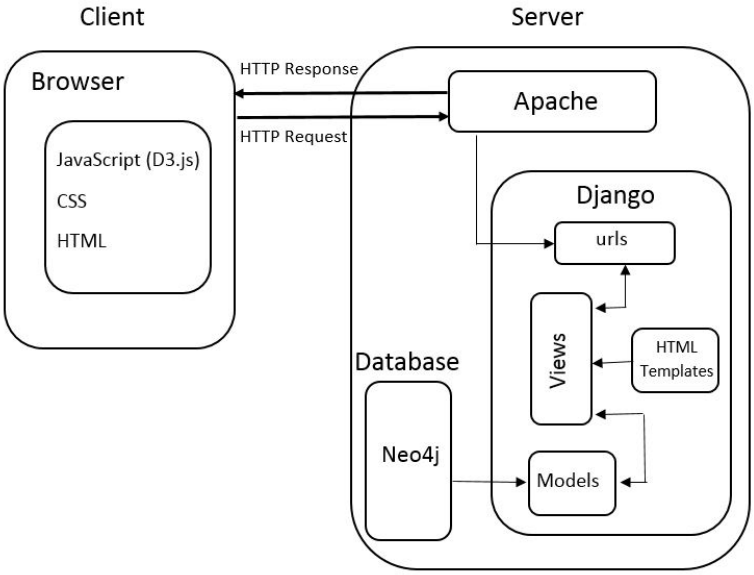

Figure 1: Diagram of System Overview

2 Software Development Life Cycle Choice

The Software Development Life Cycle (SDLC) is the process or approach the software development team adheres to throughout the project's development. Choosing the correct SDLC of the project is an important decision at the start of a software project as it could determine whether a project is successfully completed in the time given and at the required quality. There are a few SDLCs that are taken under consideration such the sequential development Waterfall approach or the Iterative and Incremental process in which features are gradually added. However, for this particular project, the fast pace method of the Agile SDLC was thought to be the best approach as the pace of producing a working solution is a priority. The requirements for this project is also not fully defined and therefore the chosen Agile SDLC must be flexible and able to deal with slight requirement changes.

The Agile SDLC also has varying methodologies that allows the achievement of an agile software development movement. These methods include the Dynamic System Development Method, the Scrum Method and the Feature-Driven Development Method. The Scrum methodology is the chosen method for this project as it allows the project progression to be agile and flexible with continuous feedback sessions to ensure the requirements are dynamical followed[1].

In the Scrum process, prioritized project tasks (referred to as sprints) are defined in short daily meetings (typically 15 minutes) with all the project team members. These meetings allow the team to communicate project progress and identify the important features that need to be added in order to increase the project progress pace. For the Tracking Interconnected Facebook Links project, these sprints will allow for a working product to be produced in the shortest amount of time while additional features can be added at a later stage. This process also allows the client to closely keep track of progress and adjust the requirements in order to produce the most high-grade

product possible. The daily meeting process occurs for up to 30 days by which time the first release of the developed software should be ready.

3 Architecture Choice

The choice of software architecture is influenced by the tools and libraries used for both the back-end service and the front-end user interface method (detailed in sections four and five), network communication and team composition. These three sections will be described further in architecture choice.

To specify the architecture choice, the team has chosen to follow a two-tier, client-server architecture. In 'client-server', client refers to any user actively interacting with (that is, drawing on the server for relevant data), the user interface provided by the system. Server refers to the storage and hosting of the integrated software solution on the 'opposite end' of the network, including business and graphing logic, as well as data drawn on from the graph database *Neo4j*. The user interface is only possible with large volumes of logic on the server, and based on user requests, the logic decides on the required data that needs to be pulled from the server. Having the logic stored on the same server as the database is an important step towards decreasing response time between receiving and fulfilling user request.

Two-tier means that any instance of a client utilizing the provided user interface will be separated from the server via a network through which all communication will occur. For the purpose of tracking interconnected Facebook links, the network between the client and the server will be the Internet. This design choice follows the need for multiple users to have access to the software simultaneously.

It is important to take note that in this architecture, the client and the server are tightly woven together, as communication between the two entities occurs directly over the network. This means that there is no middle tier to interpret and translate data correctly. In terms of team composition, this design choice is beneficial and reduces the amount of code needed to ensure compatibility. A single, four-man team is working on the software, utilizing an Agile SDLC method that promotes continuous feedback. Due to the small size of the team and the ability to closely work together and ensure quality amongst team work, direct communication between client and server is efficient and easy to achieve.

4 Front-end User Interface Method

5 Back-end Service

The back-end will be based on the *Django* web framework running on an *Apache Web Server*. The DBMS will be *Neo4j*, a graph database, that will be interacted with via the Python driver *Py2neo*.

5.1 Django

Django is an open source high-level Python Web Framework. It is based on speed, security and scalability. It is also commonly used, with companies

Figure 2: The Django Logo

such as Mozilla, Pinterest and National Geographic building their websites with it [2].

Although it has its own nomenclature, Django can be considered an MVC (Model View Controller) framework and has been compared to the popular *Ruby on Rails* web framework. An MVC framework is based on a three layer abstraction. The first layer abstracts the data access of the web application and is called the Model layer. The second layer is responsible for data display and is called the View layer. The final layer regulates the Views and is called the Controller layer. In Django's nomenclature, the view is equivalent to the standard controller, the template is equivalent to the standard view and the model is much the same as usual. For this reason Django is often referred to as a MTV or Model Template View framework [3]. It is important to note that Django is not a programming language, it is a programming pattern designed to streamline web development in the Python programming language.

Django has its own lightweight web server designed to be used for testing and development. However, for production the Django Software Foundation recommends using Apache with the *mod_wsgi* module [4].

5.2 Apache

Apache is a free HTTP web server that has been in production since 1995. It is an extremely sophisticated and flexible tool which supports the newest HTTP standards and a large number of platforms [5]. Apache has a large number of compiled modules that greatly extend it's functionality.

Apache processes any incoming HTTP requests from the clients and then sends the appropriate HTTP responses back. This will be done by interfacing with the Django web framework which will make the appropriate decisions based on the requests, and return responses containing content such as information from the Neo4j database.

Figure 3: The Apache Logo

Apache has many functions such as virtual hosting which allow one Apache installation to server multiple websites. This feature allows developers of small websites to save on server costs. Apache also provides many useful security features required when a website goes into production such as: password and digital certificate authentication, SSL and TLS support, and many more. For larger websites Apache also allows load balancing.

5.3 Neo4j

Neo4j is a graph database written in Java. It is widely used and has drivers for many languages including Python. Neo4j follows the Atomicity Consistency Isolation Durability (ACID) model which means that it is extremely reliable.

Graph databases are a type of NoSQL or Not Only SQL database. They excel at managing large complex sets of data which can be described as nodes and relationships. Figure 4 shows an example of the type of data that can be stored in a graph database.

In this application the data being stored will be that of Facebook relationships. Graph databases are orders of magnitudes faster than a RDBMS for queries on this kind of data, such as finding

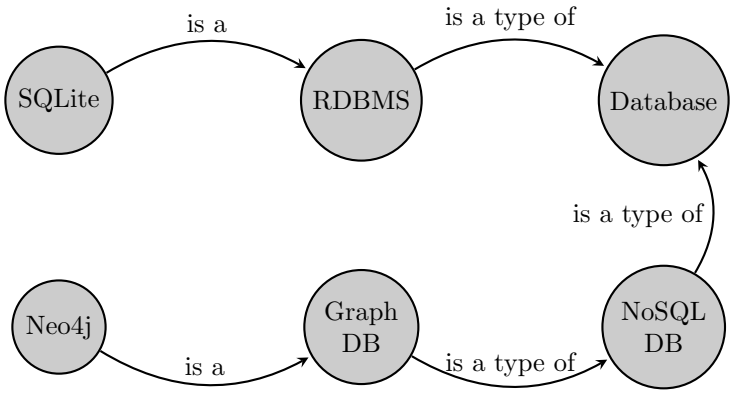

Figure 4: Example of data stored in a graph database

extended friend networks of individuals [6]. Figure 5 shows an example of the kind of data that will be stored in the database.

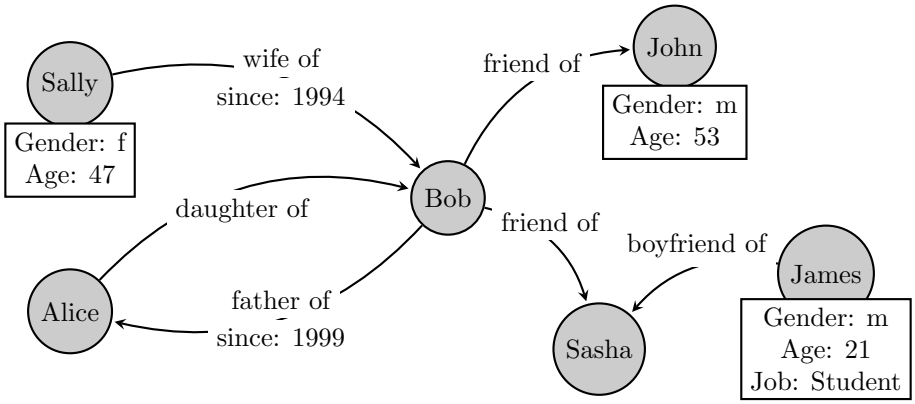

Figure 5: Example of facebook type data in a graph database

Note, from Figure 5, that both the relationships as well as the entities themselves have associated data. In face each entity acts as a key value store. Queries can be performed to find all friends since a certain date or all friends of friends (as discussed above). These queries can be performed using Neo4j's *Cypher* query language or via language drivers. The language driver that will be used for this application is Py2neo.

6 Supporting Software

The Interconnected Facebook Links project is intended to be used by a Facebook user and this user is assumed to not technically advanced. Therefore the final product needs to be user friendly and easy to use. The supporting software mentioned in this section describes the technologies that will

be used to ensure the user-friendliness and functionality of the product is achieved. The framework and internal structure of the project in terms of the architecture, front-end and back-end has been described in the previous sections. This section describes the software that will be used to support these mentioned structures.

In order to ensure that the end program is easy to use and looks professional, the visual design and theme needs to have a consistent, well designed layout and be visually attractive. Bootstrap is a HTML, CSS and JavaScript framework that allows this front-end visual design to be achieved [7]. Bootstrap provides various templates that will maintain the consistent theme throughout the final created website. It also provides navigational links for the website and the template is customizable so it can be used to create the desired theme. The Bootstrap template that will be utilized is the "dashboard template" as it has a good layout, navigational links and is visually attractive which is taken from [7].

Another technology that will support the functionally of the website is the D3.js JavaScript graphical library (assessed from [8]). This library contains JavaScript code for many different visualization graphs that can be used to visualize the data stored in the Neo4j database. Considering that the requirements for this project is to visualize the interconnections, the "force-directed graph" from the d3.js library will be used. Due to the fact that the requirements are flexible, other graphs from the d3.js library can also be used to the further visualize the database if the requirements change. These other graphs include dynamic bar and line charts, geographical heatmaps and dc.js crossfilter graphs plus others that will be considered.

References

[1] H. van Vliet, *Software Engineering: Principles and Practice* Wiley, 2007.

[2] Django Software Foundation. *Django Overview*. `https://www.djangoproject.com/start/overview/`. 2016. Last accessed: 9 March 2016.

[3] Adrian Holovaty, Jacob Kaplan-Moss, et al. *The Django Book*. `http://www.djangobook.com/en/2.0/index.html#`. Ch 3. Last accessed: 9 March 2016.

[4] Django Software Foundation. *How to install Django*. `https://docs.djangoproject.com/en/1.9/topics/install/`. 2016. Last accessed: 9 March 2016.

[5] Apache Software Foundation. *Apache - HTTP Server Project*. `https://httpd.apache.org/ABOUT_APACHE.html`. 2016. Last accessed: 9 March 2016.

[6] Robinson I, Webber J, Eifrem E. *Graph Databases* O'Reilly Media. ch 2. pp 21 - 22. June 2013.

[7] *Get Bootstrap - 3.3.6* `www.getbootstrap.com` Last accessed: 9 March 2016.

[8] *D3.js - Data Driven Documents* `www.d3.com` Last accessed: 9 March 2016.