Software Engineering Prototype Document

Names: Julian Zeegers (704582)
Joseph Gage (751052)
James Allingham (672732)
Nathan Haag (873666)

April 7, 2016

1 Introduction

This document describes several aspects of the prototype developed for the interconnected Facebook link data. An expanded description is provided based on the prototype's completion. The responsibilities of the back-end and front-end teams in developing the prototype are also discussed along with several a description of each member's work on the project. Finally, the testing which each team member implemented to ensure the quality of their parts of the project is discussed.

2 Expanded Description

A Model View Controller (MVC) web application was created using Django [1] which visualises Facebook user information stored in a neo4j database [2]. This application was hosted on an Apache server using mod_wsgi [3] and the neo4j database was integrated with the application using the py2neo driver [4]. The visualisations were created using d3.js [5] and bootstrap [6]. The visualisations provide data about friend networks as well as common words used in friend status updates.

The project is able to provide basic data about friends and their status updates which could be useful for market research, personal interest or social studies.

The project consisted of a front-end team and a back-end team. The front-end team created the visualisations. The two members in the front-end team were Joseph Gage and Nathan Haag. The back-end team was responsible for setting up the database, server and web application. The two members in this team were Julian Zeegers and James Allingham.

3 Group Members Responsibilities

The Friend Analyzer Project was created by four project team members divided into two main components, namely the back-end and front-end teams. This section outlines the responsibilities each member had and their respective tasks.

3.1 The Back-end Team

The back-end team was comprised of two members with each member having a respective back-end component to work on.

The database was developed and tested by James Allingham. The server and framework were developed and tested by Julian Zeegers.

3.1.1 The Server and Framework

The server and framework set-up was an element of the back-end component but was also the link between the back-end and front-end sections. A Django project was created that provided a framework for the website where web application views, URLs and templates could be stored. The Django project was also be hosted on an Apache server so the web application was able to be accessed via the local host initially and then on the Internet at a certain domain. This framework was also link to the neo4j database and requested data to the web application. The following are tasks that were required to be completed for this component:

1. Investigate and install the required prerequisite resources such as: Django, apache2, mod_wsgi, etc [1] [3] .
2. Create a Django project and Django web application: This includes setting up a home-page URL, making a homepage view and a homepage template.
3. Set-up an Apache2 server: Configure the apache.conf file to make a server host the Django project and serve static files such as css and images.
4. Write a README instructions file that explains how to set-up the project so a user can run it. Write instructions that explain the Django and Apache server set-up in particular.
5. Integrate the various components of the project. This includes storing the HTML files (created by the front-end team) in the correct places within the Django project and connect the neo4j database to the server.

3.1.2 The Database

The database used for this project is the graph database called neo4j [2]. This database stores information pertaining to data of a typical Facebook user's profile such as personal identification and relationship with other Facebook users data. Queried data is formatted into a JSON format and returned to the Django framework (as discussed in Section 3.1.1). Various neo4j set-up tasks are required which include the following:

1. Investigate neo4j and Cypher query language and gain knowledge required to use the database. Also research "py2neo" that is a driver used to run Cypher quires using python.
2. Setup a neo4j database.
3. Create python script that populates the neo4j database (with dummy data). This data is Facebook profile information obtained from the Facebook website and includes information such as the name, friends, age, occupation and a profile picture.
4. Create script to query database for data required for the front-end d3.js graphs. This data is formatted to a JSON format.
5. Write a README instructions file that explains how to set-up the project so a user can it. Write instructions that explain how to set-up neo4j in particular.

3.2 Testing

Once the back-end component was coded and set-up, a few tests were conducted to ensure the back-end worked as required. These tests included:

- Tested neo4j Set-up scripted (called "dbSetup.py") successfully created a neo4j database.
- Tested neo4j querying script (called "dbQuery.py") successfully return the correct queried data from the database as JSON.
- Test the the various Django URLs works and route to the front-end HTML views
- Tested that the entire project was successfully integrated by ensuring the web application is severed on the Apache server.

3.3 The Front-end Team

3.3.1 Description

The front-end team developed three views to be hosted by the back-end web application. These views were a home page, a friend network graph page and a friend status update word analysing bubble chart page.

The home page, since the application was in its prototype phase, consisted of only two visual links to each of the other pages. The page included a navigation bar as well.

The friend network graph page included the same navigation bar as the home page, thus it was linked to the other two pages. The page displayed a friend network using a d3.js force diagram which included friend profile pictures, links between pictures and name labels. The data used to create the force diagram was taken from the neo4j database.

The friend status update word analysing bubble chart page included the same navigation bar as the home page, thus it was linked to the other two pages. The page displayed a bubble chart generated using a d3.js pack payout with a flattened hierarchy (that is, one level of bubbles). The bubble chart was based on dummy data, which essentially listed words found in Facebook status updates and counted the frequency of those words. The dummy data was stored in a JSON format. The bubble chart visualised the dummy data by basing the size of the circles on the frequency of the word in status updates.

3.3.2 Responsibilities

The friend network graph page was developed and tested by Joseph Gage. The home page and the friend status update word analysing bubble chart page was developed and tested by Nathan Haag.

In terms of general responsibilities for the front-end team, the following was required:

1. Investigate the use of d3.js for all visualisations of Facebook data, researching the various visualisations available and implementing chosen ones that would best serve the purpose of the program.
2. Investigate the use of bootstrap in conjunction with HTML to create a desirable look-and-feel that ensures users enjoy their front-end application experience.

3. Test implementations of the chosen d3.js visualisations as well as their integration into HTML pages, stylised with bootstrap, ensuring that data was pulled correctly from the neo4j database, and where applicable, dummy JSON data.

3.3.3 Testing

General testing involved ensuring that the project workflow ran smoothly. This meant following all existing links and testing that they actually worked, in other words, redirecting users to the correct pages.

The friend network graph page was tested first by using dummy data to ensure that the graph behaved as expected. The graph's border, node characteristics and interaction capabilities was constantly adjusted until they were considered satisfactory. After this the network graph was tested with real data from the neo4j database to ensure integration worked as expected. All of these tests were successful.

Testing of the friend status update word analysing bubble chart page involved using various sets of dummy data to see how the bubble chart changed when the size and content of the data set changed.

References

[1] Django. `https://www.djangoproject.com/`, Last accessed 18 February 2016.

[2] neo4j. `http://neo4j.com/`, Last accessed 18 February 2016.

[3] mod_wsgi. `https://modwsgi.readthedocs.org/en/develop/`, Last accessed 7 April 2016.

[4] Py2Neo. `http://py2neo.org/2.0/`, Last accessed 18 February 2016.

[5] D3.js. `https://d3js.org/`, Last accessed 18 February 2016.

[6] Bootstrap. `http://getbootstrap.com/`, Last accessed 18 February 2016.