University of Glasgow | School of Computing Science

Are matrix-based or node-linked graphs more readable when representing causal relationships for social and health data?

Kristina Lazarova

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March, 2017

Abstract

We show how to produce a level 4 project report using latex and pdflatex using the style file l4proj.cls

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Contents

Chapter 1

Literature Review

This is the first chapter where I will introduce data visualisation and and explain how I came up with the idea of this research

Chapter 2

Literature Review

Chapter 3

Introduction

Introduction to the specific area of my research

Chapter 4

Implementation details

4.1 Software tools and technologies

Given the opportunity to chose any tools and technologies for the development of this web application was a very exciting task. However, I had to be certain that the right decisions are made. After a research period followed by a trial-error week it was decided that the backend of the application will built with Node.js and JavaScript combined with the web application framework Express. Node.js was chosen on the grounds of being event-driven, non-blocking I/O model which contributes to a very efficient and lightweight software. A Node.js JavaScript engine is also used in the Google Chrome browser. JavaScript servers have incredible performance due to their asynchronous I/O. Node.js appears to be single-threaded from a developer's point of view, as there is no thread management involved in the development process. However, behind the scenes Node.js handles threading, file system events, implements the event loop, feature thread pooling etc. Coming from a Java background, the Maven equivalent in Node.js is NPM. By using NPM commands the developer is able to install variety of different modules to help the implementation process. NPM executes the function of a package manager. Express is the standard server framework for Node.js. It is usually described as a minimal and flexible Node.js web application framework. Many popular frameworks such as KeystoneJS, Kraken and Sails, are built on Express.

AngularJS 1 was chosen for management of frontend functionality. Even though there is a newer version of the product, the lack of documentation and support online, was a sufficient reason for using the older AngularJS. It uses HTML as a template and enables the developers to extend it to express the application's components clearly. AngularJS supports features such as data binding and dependency injection which decreases the amount of code that a developer would usually write to implement them.

The database system chosen for the project is PostgreSQL. Considering the size of the project an object-relational database was chosen. In addition, it decided on PostgreSQL in particular because it is open source and has gained a reputation of a reliable database system. Also previous experience with PosgreSQL from developers point of view made the decision easier.

* maybe database schema will be added here *

4.2 Development Process

4.2.1 User Stories

The first step of the implementation was understanding the requirements and creating user stories. Some of the most important user stories are:

```
As a researcher, I want to be able to see participant's answers, so that I can anal
As a researcher, I want to keep participant's scores anonymous, so that my experime
As a participant, I want to be able to see graphs and associated questions, so that
As a participant, I want to be unable to go the next question, before completing th
```

4.2.2 Wire-frames

After the requirements gathering analysis, development of wire-frames followed. Balsamiq Mockups 3 is the software used for the creation of wire-frames. An example of the research question page can be found in figure 4.1.

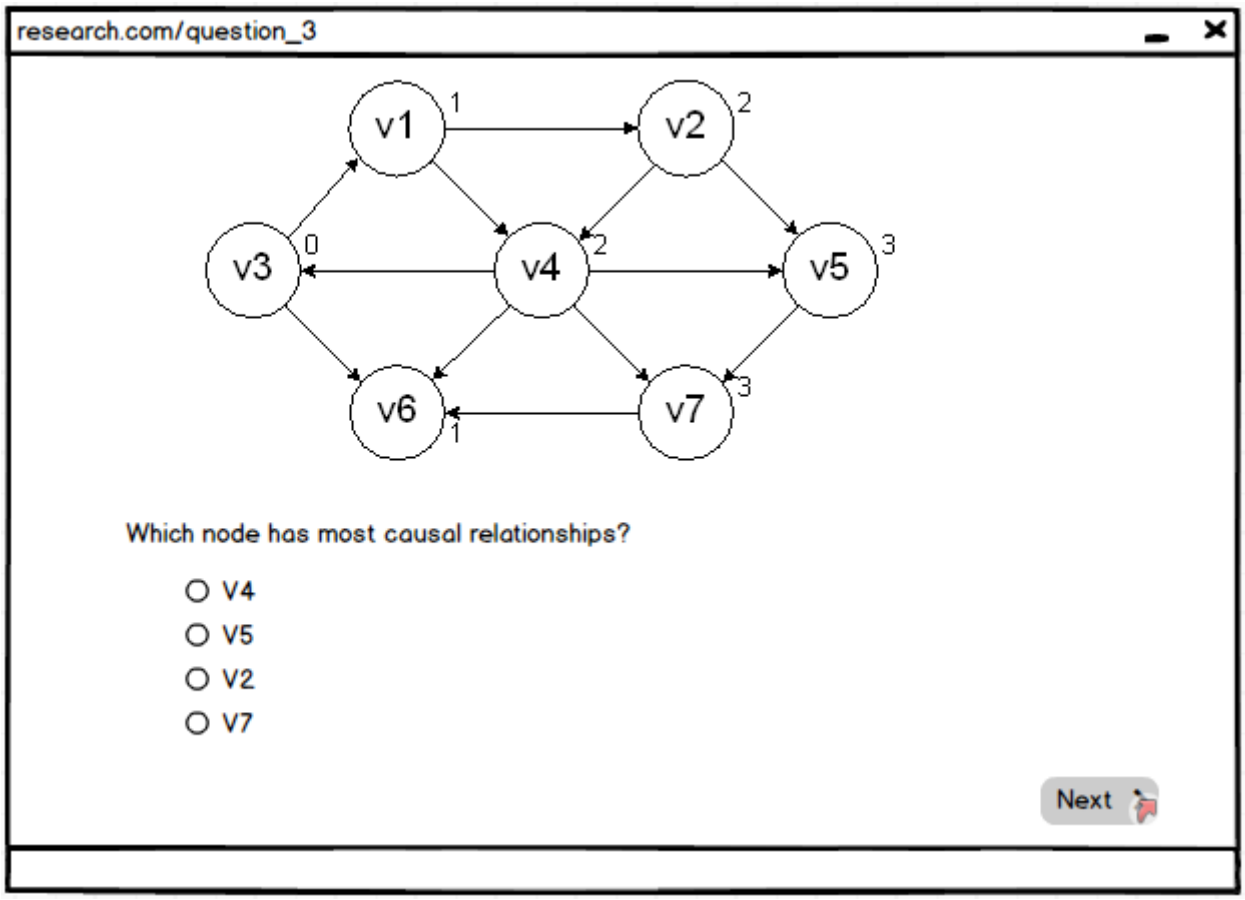

Figure 4.1: Research Question Wire-frame

After discussing the wire-frames it appeared that some important features are missing. One of those features was a participants training session. The aim of the experiment is to test which graph is more readable for people

who do not have regular exposure to such data visualisation. Therefore, it is important to make the participants aware of how to read each graph before the actual experiment. This way, the requirements specification became an iterative process during which a better understanding of the product evolved.

4.2.3 System Design

Designing the system was the next stage in the process.

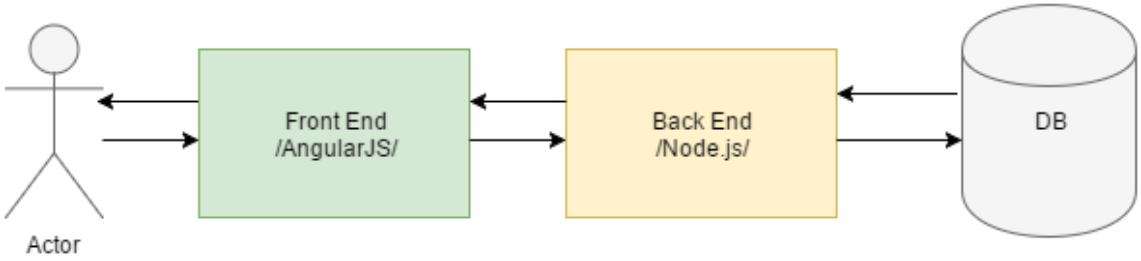

Figure 4.2: An abstract representation of the system design

Figure 4.2 shows an abstract view of the system design. There is an Actor who will either be a participant in the study or a researcher. They will interact with the front-end which will be in the form of a web application in a browser. The front end will communicate with the back-end which will be implemented in Node.js. The back-end will make requests to the database to retrieve and send information.

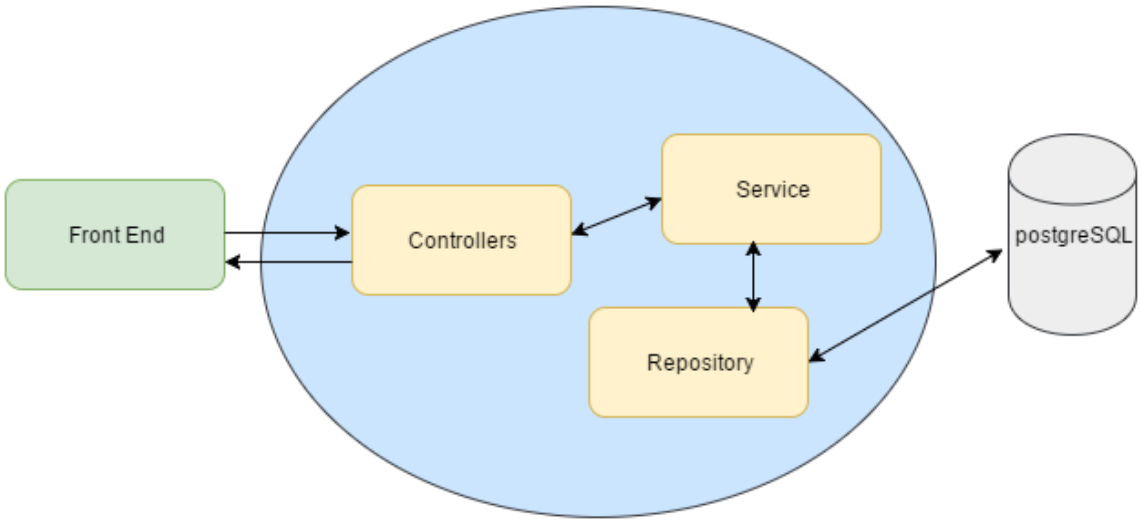

Figure 4.3: A more specific representation of the system design

Figure 4.3 displays a more detailed version of the system design. This particular design has been implemented to separate the different concerns in this specific system. When the Actor interacts with the application, the front-end will send information to the Controllers. There are many controllers because there is a controller for each page with front-end functionality. The Controller component decides what should the next action be according to the user input. It has control over the front-end logic and sending requests to the service if information from the database or the server is requested. For example, the answers to all questions are kept in the front-end until a ”Submit” button is clicked on. This action triggers a request to the Service. The Service component works with the back-end logic. It can send and retrieve data from the database and keep the information in the Repository.

The Service also deals with the requests for the different web-pages. Also it ensures that the project dependencies are loaded.

4.2.4 Implementation

The development process was split into front-end and back-end. Without using any frameworks, the front-end Hhtml pages were created following the wire-frames. Bootstrap was added to the html to improve the UI design and make it look more appealing for the participant.

By this time, it was clear that Node.js will be used to create a server so the next step was to implement it. The decision to use Express as a framework with Node.js followed and the html pages were mapped to a handlebars or hbs files.

The following couple of weeks were dedicate on work on the database system: creating database schema, and tables, and work on connecting it with the server.

4.3 Challenges

```
Spring idea failed
changed to Node.js
Angular compatibility with Node.js
```

In the beginning of this project the Java framework Spring was going to be used in the implementation as it is among the most widely used frameworks in industry [**?**]. This decision was supported by extensive previous experience with Java from developer's point of view and the applicability of the skills to be acquired. However, one of the reasons why Spring is used in industry is because of the large and complex systems that exist there. The Spring framework works on a very high level of abstraction where you can easily write configuration files to add dependencies from different project. Therefore, it is considered rather unfriendly for small independent projects and developers with limited Spring experience. The reasoning behind this conclusion was provoked after a couple of unsuccessful attempts to set relative paths to different CSS and JavaScript files. The issue was found to be in the web application configuration file. This is how the very simple task of reading a css file turned to be a long tedious debugging process after which the realisation that Spring is unnecessary abstract and complex for this project occurred.

A new research for web-application frameworks followed. Node.js backend was chosen because of its event-driven, non-blocking I/O model which creates an efficient and lightweight server-side of the application. Another challenge appeared when trying to incorporate AngularJS with Node.js. Usually in AngularJS one uses curly braces to reference data structure from the AngularJS controller. However, Node.js also uses curly brackets to reference information from the backend in the frontend. After a long research it was found that Node.js overrides the use of curly braces and the application is not displaying Angular data as it expects it come from the backend. Unfortunately, an appropriate error message does exist and it all had to be discovered during the development process. Instead of using curly brackets one can also use "ng-bind" and achieve the same result. This approached solved the issue until "ng-bind" information was need in "ng-src" to display the appropriate graph image. It is not possible to use "ng-bind" inside "ng-src" so the present solution at the time was no longer solving the problem. Therefore, the Angular configurations had be altered to use a different symbol. Implementing this solved the problem entirely.

4.4 Software reliability testing

Chapter 5

Evaluation

5.1 Design

5.2 Participants

5.3 Procedure

5.4 Results

5.5 Discussion

Chapter 6

6.1 First Section in Chapter

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog [**?**]. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

6.1.1 A subsection

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox [**?**] jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

Chapter 7

The Fox and Dog

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

7.1 The Fox Jumps Over

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over Uroborus (Figure 7.1). The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over [**?**] the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

7.2 The Lazy Dog

The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

The quick brown fox jumped over the lazy dog. The quick brown fox [**?**] jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog. The quick brown fox jumped over the lazy dog.

Figure 7.1: An alternative hierarchy of the algorithms.

Appendices

Appendix A

Database Schema

```
participants_answers(question_id, participant_id, answer, time)
participants_info(participant_id,participant_name, email, uni_degree, age)
questions(question_id, question, one, two, three, four, correct, image)
```

Appendix B

Running the Programs

An example of running from the command line is as follows:

```
> java MaxClique BBMC1 brock200_1.clq 14400
```

This will apply $BBMC$ with $style = 1$ to the first brock200 DIMACS instance allowing 14400 seconds of cpu time.

Appendix C

Generating Random Graphs

We generate Erdós-Rënyi random graphs $G(n, p)$ where n is the number of vertices and each edge is included in the graph with probability p independent from every other edge. It produces a random graph in DIMACS format with vertices numbered 1 to n inclusive. It can be run from the command line as follows to produce a clq file

```
> java RandomGraph 100 0.9 > 100-90-00.clq
```