



# Who-Knows-Who

FullContact Team 3

Colten Gruchow, Jacob Ratzlaff, Joe Wilson, Daniel Zarrini

# Contents

- 1. Introduction ..... 2
- 2. Requirements:..... 2
  - 2.1 Functional Requirements ..... 2
  - 2.2 Non-Functional Requirements ..... 2
- 3. System architecture ..... 3
  - 3.1 Item Descriptions: ..... 3
  - 3.2 Database and Structure ..... 4
- 4. Technical Design..... 5
  - 4.1 Graph Database Layout..... 5
  - 4.2 Algorithm ..... 6
    - 4.2.1 Information Based Weighting ..... 6
    - 4.2.2 Behavioral Weighting ..... 7
    - 4.2.3 Introduction Pathfinding ..... 7
    - 4.2.4 Referrals ..... 7
  - 4.3 Decisions ..... 7
- 5. Results ..... 8
  - 5.1 Lessons Learned ..... 8
  - 5.2 Summary ..... 8
- Appendices..... 9
  - Appendix A: Weight Incorporation for Behavioral/Email Data ..... 9
  - Appendix B: KNOWS\_PERSON\_BACKEDGE Weight Calculations..... 9
  - Appendix C: Web Interface ..... 10

## 1. Introduction

The Denver based start-up software company, FullContact, provides a cloud based solution to manage contact data for personal, enterprise and developer use. FullContact unifies contacts from sources such as Gmail, Twitter, Microsoft and various other services and merges them together with no duplicates. The goal is to increase the usability and practicality of contact information by enhancing address books/contact data. The CTO of FullContact, Scott Brave once stated: "We are trying to do what Dropbox did with files but with contacts." FullContact currently is working on many high end technologies and is trying to expand its product to appeal to more users.

FullContact has been growing rapidly and is pursuing different ideas to incorporate into their product. The "Who knows who graph" project is such an idea. They are trying to enable the user to find introductions to other people through contacts. FullContact is also looking for a referral system to invite people outside their user base to use FullContact's products.

## 2. Requirements:

Currently FullContact has data from contact address books and other sources. The first goal of the project is to represent this data in a graph that connects people together with weighted edges representing the strength of their relationship. This graph can then be used to calculate an optimal path to allow one person to be introduced to another. For example, if Person A wants to be introduced to Person B, the process finds a path from A through contacts to find an introduction to B. FullContact would also use this system for referrals. This process finds optimal non-users to receive recommendation from users whom have stronger connections to non-users. It also takes into account non-users that are a critical part of the social graph.

### 2.1 Functional Requirements

1. Load users' contact address book data into graph database.
  - Able to accept updates to address books
2. Load users' recent email activity into graph database
  - Able to read in new emails during runtime
3. Create weights for the connections between people
  - Create algorithm for weighting edges based on address book information
  - Create algorithm to adjust connection weight through the use of email activity
4. Determine who would be the ideal person(s) to host an introduction between two contacts.
5. Determine candidates for recommendation using FullContact services.

### 2.2 Non-Functional Requirements

1. Paths suggested by the algorithm must be within a reasonable degree of accuracy
2. Algorithms must be modular and flexible
3. Removing duplicates from data
4. Must be built using Java 8
5. Use IntelliJ IDEA
6. Use the Gradle build system for dependencies
7. Queries must be able to be made from webpage
8. System should be easy to integrate into FullContact's production environment

### 3. System architecture

Our system is provided a list of address books and activity data stored in JSON formats which is parsed and then used to populate a graph in Neo4j. After initial population of the graph, our system uses algorithms to weight and add edges to represent relationships and their strength. Following this, the user/consumer can interact with the webpage and receive possible introductory paths and product referrals.

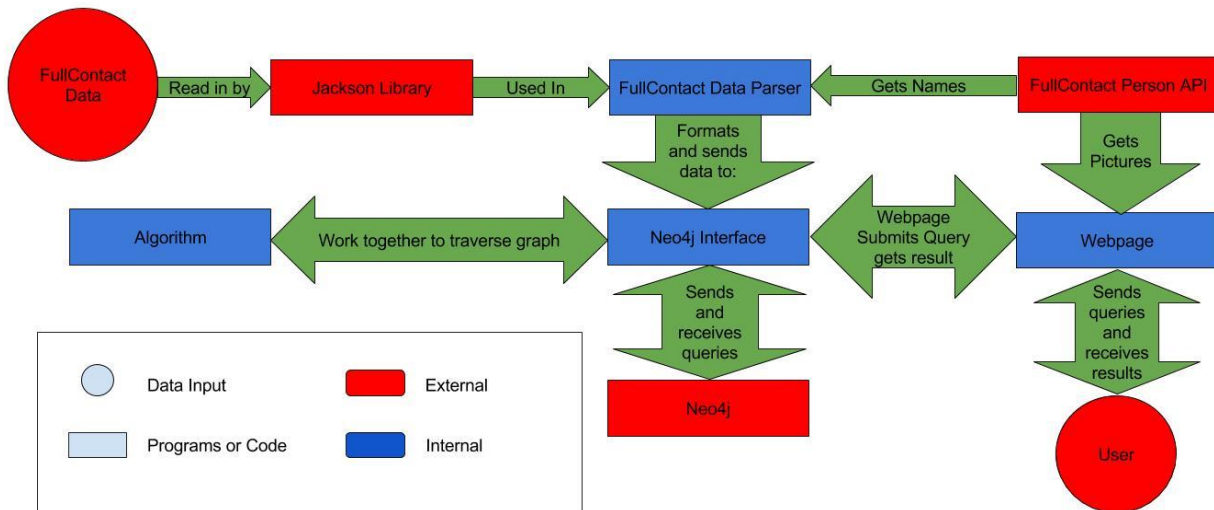


Figure 1: Architectural Structure

#### 3.1 Item Descriptions:

As seen in Figure 1: Architectural Structure:

- FullContact Data: Address books and email behavior data of all FullContact employees provided by the clients in JSON format.
- Jackson Library: Java library for JSON parsing
- FullContact Person API: FullContact API that returns additional contact information when given a single piece of information
- FullContact Data Parser: Java class that reads in all contacts and information keeping track of duplicates and calls the Neo4j interface to add them to the graph
- Neo4j Interface: Java class that handles all transactions with neo4j
- Algorithm: Java class that holds all constants and provides the equations used for weighting and traversals
- Webpage: User interface that allows a user to request and see possible introduction paths and referral options. For more information, see Appendix C: Web Interface
- Neo4j: Graph database used for storing information

### 3.2 Database and Structure

Neo4j is a NoSQL database. By the nature of the project, a graph database is essential for the use of the application. The basic structure of information stored in the database is represented in Figure 2. Initially only person nodes, information nodes, owner edges, and knows information edges are populated. The graph is weighted and has the appropriate knows person edges added by our algorithm prior to use. This is further discussed in the Technical Design section of this report.

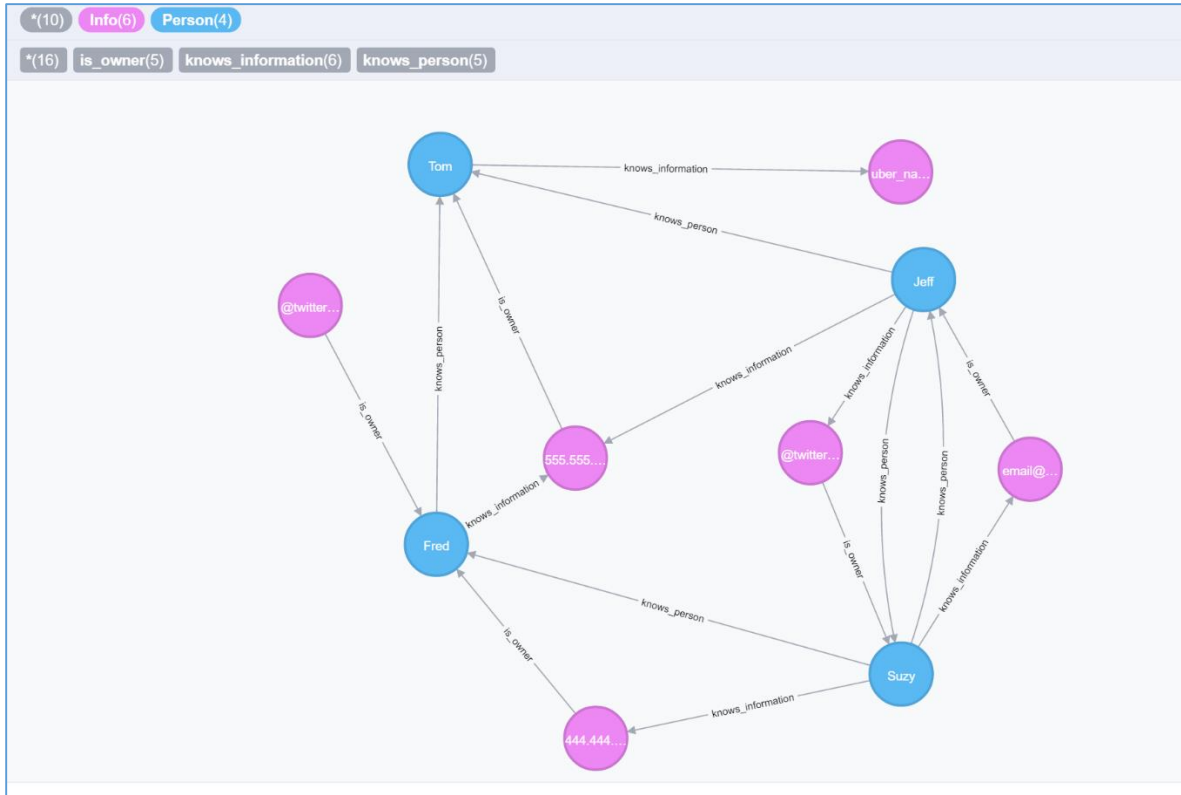


Figure 2: Neo4j Visualization

## 4. Technical Design

The key to the design of the Who-Knows-Who project lies within the graph layout. This allows the algorithm to accurately incorporate both behavior data and address book information.

### 4.1 Graph Database Layout

There are two types of nodes in the graph: PERSON and INFO nodes. When reading in an address book file each new information point creates a new info node. These info points are related to a PERSON node with a directed IS\_OWNER edge pointing from the information to the contact as seen in Figure 3.

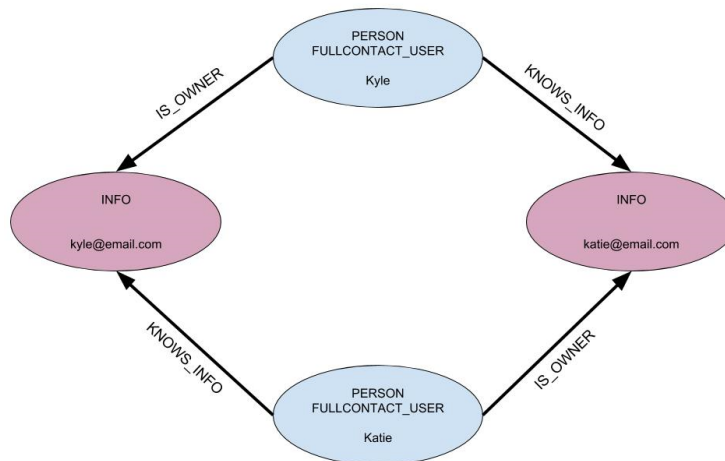


Figure 3: Info Nodes

Each time a piece of information is taken from an address book, the owner of the address book is related to the INFO node with a directed KNOWS\_PERSON edge pointing from the address owner's PERSON node to the inserted INFO node as seen in Figure 4.

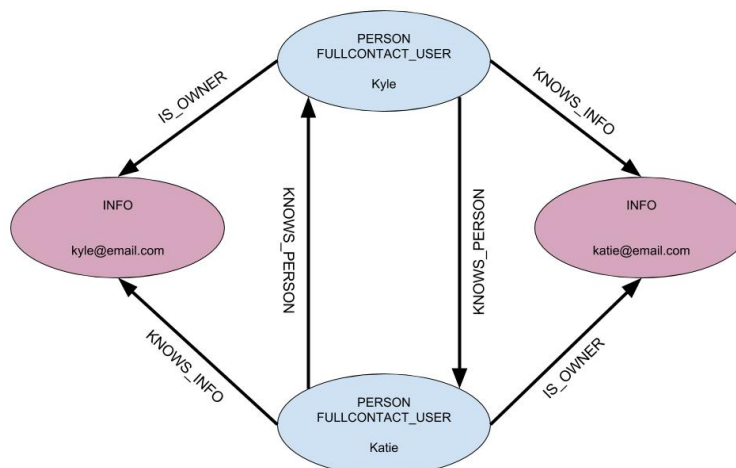


Figure 4: KNOWS\_PERSON Relations

There are two additional labels for the PERSON nodes: FULLCONTACT\_USER and NON\_USER. These labels are set at the time of loading in the address books. Using this graph, the PERSON Nodes are directly connected with a KNOWS\_PERSON edge. This is directional from the owner of the address book. If it does not exist already a KNOWS\_PERSON\_BACKEDGE is created to connect contacts who do not have an address book with FullContact as seen in Figure 5. A KNOWS\_PERSON\_BACKEDGE is also added if a person does not have the owner of a KNOWS\_PERSON edge in their address book. A KNOWS\_PERSON\_BACKEDGE is also added if a person does not have the owner of a KNOWS\_PERSON edge in their address book.

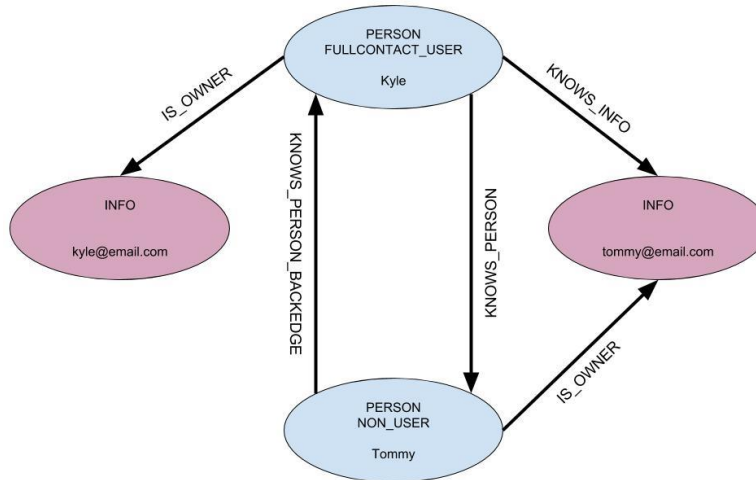


Figure 5: KNOWS\_PERSON\_BACKEDGE Relations

## 4.2 Algorithm

In order to mathematically represent the strengths of connections between people, the team used weighted graphs. The weight from one user to another was determined in the following two ways:

1. Informational Based: This method weights the connections between users using the amount of information one user knows about the other. For example, if user A knows user C's phone number and twitter handle, that might say user A knows user C better than user B since user B only has user C's twitter handle.
2. Behavioral: This method weights the connections between users using the interactions between the users such as the frequency of emails between the users that occur over a certain period.

The information given is in the form of FullContact users' Address Books and logs of email traffic between users in the form of JSON files. This information was used in the weighting of connections between users.

### 4.2.1 Information Based Weighting

In the information based weighting, the number of INFO nodes a specific user has is summed with a certain weight. For example, the phone number INFO node had a weight of 3, email weight of 2, and accounts information such as Twitter/ Facebook has a weight of 1. The initial connection strength between users is the weighted percent of information known. If a user A knows a user B but B does not

know A, a back edge is created from B to A. This is a common occurrence for non-FullContact users, they do not have address books, and in turn they do not know any information.

#### 4.2.2 Behavioral Weighting

After the weights are created using information based weighting, the behavioral weighting system is used to refine the weight along communication paths. The behavioral weighting system keeps track of recent emails sent between the users. Our implementation updates the database as email logs are received. Each PERSON node stores the total number of emails they have sent by week. In the KNOWS\_PERSON edge and KNOWS\_PERSON\_BACKEDGE there exist properties for each week of communication in our time range. For more information on the calculation for KNOWS\_PERSON\_BACKEDGE see Appendix B: KNOWS\_PERSON\_BACKEDGE Weight Calculations.

#### 4.2.3 Introduction Pathfinding

To find an introduction, the algorithm runs a Breadth First Search to find all viable (length 3 or less) paths from the requester to target. These paths are then sorted through a multiplication of the weights along each edge in the path, longer paths are penalized. The weights are a combination of the information weight and the behavior weight. For more information on the behavioral weight calculation see Appendix A: Weight Incorporation for Behavioral/Email Data. The top five paths will be returned for display.

#### 4.2.4 Referrals

The algorithm for referrals finds the non-users with the highest degree of incoming KNOWS\_PERSON edges. Once these non-users are found, the five closest users to each non-user are returned as suggestions for the recommender.

### 4.3 Decisions

Java is the main language for development at FullContact, and thus we used Java. This allowed for easy integration to existing code provided by FullContact. Java is faster when dealing with large operations with substantial amounts of data than non-compiled languages. Java is also a matured language with lots of open-source libraries to speed up development. The fact that Java is statically typed and does not support type inference limits the number of type errors. We know that user data is not clean therefore we can focus on application developing rather than data sanitization.

FullContact's libraries and APIs were used. This decision was obvious as we are using data already formatted by FullContact. When dealing with behavior data we decided to use Joda-Time. Joda provides more comprehensive functionality when dealing with date and time behavior as well as being the standard used at FullContact. Since we are developing a simple web interface we used the Dropwizard framework. This allowed us to quickly create the web portion of the interface. Dropwizard also allows for easy deployment of software on servers. Since the main focus of the product is not the web interaction this recommendation by FullContact has allowed us to focus heavily on the backend development.

The biggest technological decision came in the form of our graph management choice. We considered two options, Neo4j and Apache Spark GraphX. Both options support the scale of graphs that we would be building. The Neo4j database had strong integration with Java. In addition, Neo4j is easier to use in small scale graphs, ones that could be used for easy testing. Neo4j also has a built in visualization tool to assist in verification of results and design planning. Given these factors we decided to use Neo4j as our database for the project.



## 5. Results

### 5.1 Lessons Learned

- Daily meetings are essential to work flow
- Scrum tools like Trello help manage a large amount of tasks
- While Sprints are not necessarily hard guidelines, they help provide steady development
- Creation of Enterprise software by building from an existing codebase
- The integration of Web Development and Server incorporation
- Implementation of a graph database through Neo4j
- Usage of different APIs and their incorporation with the project
- Gradle (open source build automation system) and using it to include different projects, libraries, and packages
- Incorporation Dropwizard Web Framework with our project and display Server Output to the user
- Testing and using Amazon Cloud Servers to store the database and perform large scale calculations
- Incorporating visualization templates for the front end development such as {{ mustache }}
- Using visualization tools such as Bootstrap and JavaScript

### 5.2 Summary

For this project we implemented a graph that used contact book and activity data to create and weight connections between people. We developed a weighting system that accounts for both behavioral data and known contact information. Once the graph is created, a system to find introduction paths and candidates for referral is available. We also created a simple browser based UI for users to access our system. Although everything works well and the client's requirements were met, we did not have time to extensively tune our algorithms to return more accurate results. Future extensions of this project could include integrating it with FullContact's existing systems and implementing a machine learning system to tune our algorithm for better results.

Throughout the project, Agile/Scrum techniques were used in development. Sprints were effective and helped us monitor the status of development. Weekly client meetings and daily team meetings ensured communication between both the team and client, and led to the success of the project.

## Appendices

### Appendix A: Weight Incorporation for Behavioral/Email Data

Each path weight will be calculated using the weekly email data for each week and the total emails sent per week. These data points are used to calculate average emails sent per week for a user. Afterwards the algorithm will compare that information to the number of emails sent along the path and depending on the result, the following occurs.

If emails sent are above average:  $\frac{((1 - \text{PathWeightToUser}) * (\text{totalSentbyWeek} - \text{avg}))}{(\text{totalSent} - \text{avg})} * \text{decayFunction}$

If emails sent are below average:  $\frac{(\text{PathWeightToUser} * (\text{totalSentbyWeek} - \text{avg}))}{(\text{totalSent} - \text{avg})} * \text{decayFunction}$

This calculation will occur for every week and after the calculations are summed, the sum will be returned for use in the path weight calculation.

### Appendix B: KNOWS\_PERSON\_BACKEDGE Weight Calculations

The back edges are weighted using multiplication of the following parameters. Figure 6 shows an example in which a KNOWS\_PERSON\_BACKEDGE would be created.

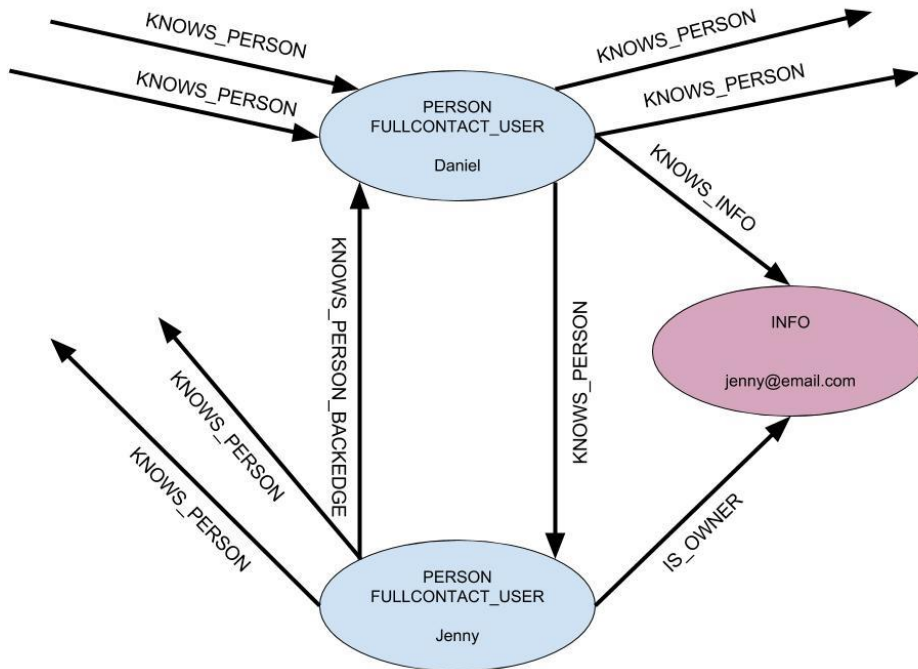


Figure 6: Algorithm Analysis

**A** = Weight of the KNOWS\_PERSON

**B** =  $(1 / k_{Out})$  where  $k_{Out}$  is the number of people Jenny knows

**C** =  $(p_{In}/(p_{In}+p_{Out}))$  where  $p_{In}$  is the number of people who know Daniel and  $p_{Out}$  is the number of people Daniel knows

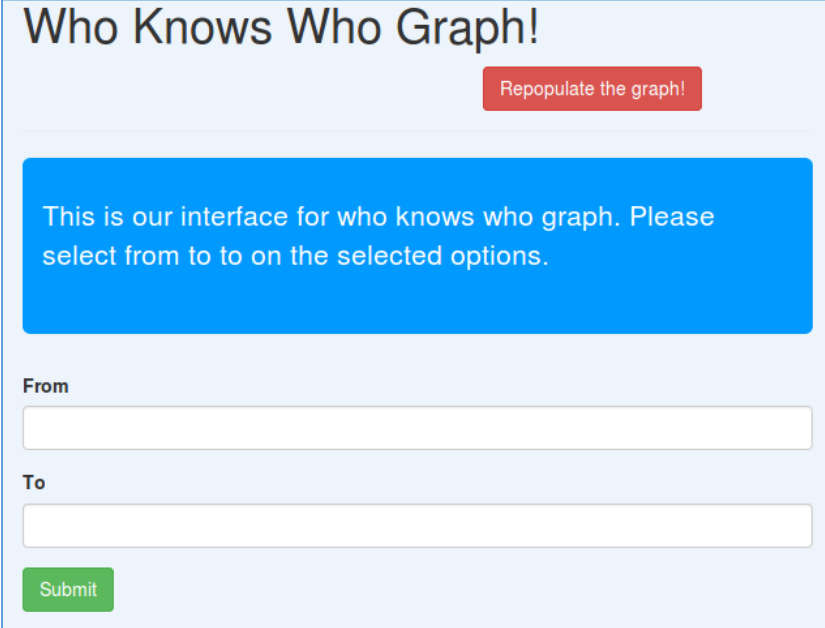
**D** =  $(1 / k_{In})$  where  $k_{In}$  is the number of people that know Jenny

$KNOWS\_PERSON\_BACKEDGE = A * B * C * D$

### Appendix C: Web Interface

A web interface is important for demoing and developing the project. Because the project focused heavily on the backend development, the web interface was not a main aspect of the design. If later implemented into the FullContact system, this web interface would be most likely removed and incorporated into the existing web app.

Figure 7 shows the initial interface for the project created in which it asks the user to type introductions "From" and "To". After the user types who they want to connect from and to, they can click the submit button and results will be displayed. The display shows five possible path options.



Who Knows Who Graph!

Repopulate the graph!

This is our interface for who knows who graph. Please select from to to on the selected options.

From

To

Submit

Figure 7: Web Interface

Figure 8 shows one of the possible results from two different people. In this example it is displaying a possible path from "Karlis" to "Bart" and the possible people Karlis has to ask to get introductions to Bart. The names displayed are clickable and will allow the user to send an email if needed.

Path Option 5

1. [Karlis Lauva](#)  

2. [Drew Lawrence](#)  

3. [Jack Wanamaker](#)  

4. [Bart Lorang](#)  


Figure 8: Path Finding

Figure 9 shows the result for Referrals. This picture shows the individuals that FullContact wants to refer to their service due to their influence on current FullContact users. It also shows who should perform the referral. In this example [chris@fullcontact.com](mailto:chris@fullcontact.com) is the best choice for referring Ben to FullContact.

# Results!

Email Address	Number of Connected FullContact Users	
Ben Deda	58	<ol style="list-style-type: none"> <li>1. <a href="mailto:chris@fullcontact.com">chris@fullcontact.com</a></li> <li>2. <a href="mailto:skylar@fullcontact.com">skylar@fullcontact.com</a></li> <li>3. <a href="mailto:mary@fullcontact.com">mary@fullcontact.com</a></li> <li>4. <a href="mailto:martins.caune@fullcontact.com">martins.caune@fullcontact.com</a></li> <li>5. <a href="mailto:ken@fullcontact.com">ken@fullcontact.com</a></li> </ol>
Daniel Williams	47	<ol style="list-style-type: none"> <li>1. <a href="mailto:jessica@fullcontact.com">jessica@fullcontact.com</a></li> <li>2. <a href="mailto:bart@fullcontact.com">bart@fullcontact.com</a></li> <li>3. <a href="mailto:alex.porter@fullcontact.com">alex.porter@fullcontact.com</a></li> <li>4. <a href="mailto:scott@fullcontact.com">scott@fullcontact.com</a></li> <li>5. <a href="mailto:jake@fullcontact.com">jake@fullcontact.com</a></li> </ol>
Eden Elder, Ph.D.	45	<ol style="list-style-type: none"> <li>1. <a href="mailto:jaclyn@fullcontact.com">jaclyn@fullcontact.com</a></li> <li>2. <a href="mailto:jessica@fullcontact.com">jessica@fullcontact.com</a></li> <li>3. <a href="mailto:bart@fullcontact.com">bart@fullcontact.com</a></li> <li>4. <a href="mailto:sage@fullcontact.com">sage@fullcontact.com</a></li> <li>5. <a href="mailto:rushtonm@gmail.com">rushtonm@gmail.com</a></li> </ol>

Figure 9: Referrals