ASHESI UNIVERSITY

TRUSTHOP: BUILDING A SOCIAL TRUST NETWORK

UNDERGRADUATE THESIS

B.Sc. Computer Science

Eugene Kwadwo Daniels

2022

ASHESI UNIVERSITY

TRUSTHOP: BUILDING A SOCIAL TRUST NETWORK

UNDERGRADUATE THESIS

Undergraduate Thesis submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Eugene Kwadwo Daniels

2022

DECLARATION

I hereby declare that this undergraduate thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

…………………………………………………………………………………………………

Candidate's Name:

…………………………………………………………………………………………………

Date:

…………………………………………………………………………………………………

I hereby declare that preparation and presentation of this undergraduate thesis were supervised in accordance with the guidelines on supervision of thesis laid down by Ashesi University.

Supervisor's Signature:

…………………………………………………………………………………………………

Supervisor's Name:

…………………………………………………………………………………………………

Date:

…………………………………………………………………………………………………

Acknowledgements

I want to express my heartfelt gratitude to Mr. Dennis Asamoah Owusu for his assistance, advice, and resources during the research and writing of this report. I am grateful to Ashesi University's CSIS Department for providing the environment, inspiration, and technological know-how, allowing me to investigate and construct an application for my project. I would also like to express my heartfelt appreciation to my friends and family for their encouragement and support. Finally, I would like to thank my family and friends for their support throughout my academic adventure at Ashesi University.

Abstract

Online social networks have increased in recent years. In many areas, such as businesses and governmental organizations, organizations of all sizes have been using social networks to thrive. As a result of this, the concept of trust comes into the picture, as individuals divulge a large amount of personal information on social networks. It is more important than ever to create and measure trust in a social network for people's privacy, safety, and livelihood. This paper aims to identify ways to use the social network structure and the trust links between them to predict how much two people who are not directly connected could trust one another. Inferring such trust values is hard to compute when person A and person B are unconnected in the social network. This paper presents a graph-based model to evaluate trust in an online social network. It focused on computing the quality assessment of person A for person B, considering that person A and B are unconnected in a given graph of quality assessment of people. The proposed model's accuracy in predicting a trust value is calculated and compared to the multiplicative strategies for trust propagation [11]. This paper also proves how this model can be used in real scenarios to answer how trustworthy someone is innate.

Table Contents

CHAPTER 1: INTRODUCTION

This chapter provides a summary of the study, describes the problem, and explains the proposed solution, as well as the study's rationale and objective. It also defines trust and online social networks.

1.1 Social Networks and Online Social Networks (OSNs)

A social network can be characterized broadly as a set of actors and relationships expressing some link (or lack thereof) among the actors [14]. A social network's actors (people, organizations, or other social entities) are linked by a series of interactions such as friendship, affiliation, financial exchanges, trading links, relations, or information exchange. According to Liebeskind et al. [1995], social networks are "a collectivity of individuals among whom interactions occur that are maintained solely by shared norms of trustworthy behavior."

Online Social Networks are web-based services that (1) allow individuals to create a public or semi-public profile for themselves within a bounded system, (2) indicate a list of other users with whom one is connected, and (3) display to users their list of connections as well as those made by other users within the system [16]. OSNs have expanded drastically over the last decade. Social media platforms such as Facebook, Twitter, WhatsApp, and Instagram are used by 3.6 billion people, or 48.3% of the world's population, with a million new members added every day [9]. By 2025, this percentage is predicted to increase to 56.7 percent of the world population [9]. They are now at the core of many people's lives, providing a platform for social interaction. Because of advancements in technology and the internet, online social networks are expanding exponentially. The internet has encouraged and facilitated online connectivity, resulting in people spending more time connecting with others on these platforms. Users join for several reasons, including networking, acquiring social information and assistance, and developing and maintaining social connections [12]. OSNs allow for self-expression, social pleasure, and the arousal of curiosity, as well as emotional and aesthetic experiences [13]. Today, a rising number of people use OSNs for daily communication or even business purposes. Such activities involve building trusting attitudes about a specific user or product.

1.2 Trust

The age of the internet has caused many changes in the way humans interact with each other. One change is that now people can connect with others from across the world, but another momentous change is meeting each other for the first time without ever telling whether they can trust their new friend’s intentions.

OSNs have evolved as a means for members of the public to engage without having to be physically present. However, this also introduces a new challenge: determining who is trustworthy within these interactions. The concept of trust is essential to OSNs. Our everyday social lives, which we take for granted, would be impossible without trust [2].

We may also define trust as confidence in integrity, strength, or ability in another person.

Trust is a measure of assurance in an entity’s ability to behave predictably in the face of a lack of ability to monitor or control the environment in which it works [17].

Golbeck [2005] defines trust in a person as “a commitment to an activity based on a confidence that that person’s future actions will result in a beneficial outcome.”

According to the psychological definition of trust, trust is the psychological condition of an individual in which he risks being susceptible to the trustee based on positive anticipation of the trustee's intentions or actions [24].

Trust is described in sociology as a "bet" on the trustee's potential future behavior [25].

Finally, there is trust in computer science, which is classified into two types: user and system. The subjective anticipation that one entity has about the future behavior of another is referred to as user trust [26]. In contrast, system trust is the expectation that a device or system will faithfully behave in a specific manner to fulfill its intended purpose. Trust in this paper refers to user trust as defined in Computer Science. There are two kinds of user trust in computer science: direct trust and recommended trust, often known as peer-to-peer trust [27]. Direct trust is founded on the member's direct experience with another party, whereas recommended trust is founded on the trust's propagative features.

1.2.1 Concepts of Trust

- **Trustor**: The trustor is a user seeking to determine another user's trustworthiness/trust degree.
- **Trustee**: The trustee is a user whose trustworthiness/trust degree is being assessed.
- **Recommender:** The recommender is an intermediate user who assists the trustor in finding the trustee's trustworthiness/trust degree.
- **Trusted/Trust Path:** A trust path is a path from the trustor to the trustee that includes:
    - i. A trustor (the source).
    - ii. One or more recommenders.
    - iii. A trustee (the target).
    - iv. Trust interactions between them (edges or links)
- **Trusted Graph:** From the trustor to the trustee, all trusted paths that begin with a trustor and end with a trustee form a trusted graph.

1.2.2 Characteristics of Trust

Different properties can be associated with trust. It could change over time or be subjective. Trust's properties significantly impact choosing the type of trust being researched and modeled [27]: The essential trust qualities are:

- **Subjective**
  - Trust, in general, can be subjective. If person A expresses an opinion on someone, the trust derived from that opinion is subjective. Because trust is subjective, trust computation is personalized, with the member's preferences directly influencing the computed trust review.
- **Propagative**
  - This is a vital aspect of trust that is discussed in several works. The following example exemplifies this. If A trusts B and B trusts C, then A, who does not know C, can place some trust in C based on how much A trusts B and how much B trusts C. Because trust is typically propagative, knowledge about it can be transmitted from one social network member to another, resulting in trust paths. Since we are investigating peer-to-peer algorithms for evaluating trust among users, this is one of the most significant trust features for this research. This type of method requires propagative features.
- **Aggregative**
  - While trust propagation in a social network states that a member can build some trust in a member who is not directly connected to it, it does not address how to respond when we have many trust links suggesting varying degrees of trust in a trustee. Data from many links must be pooled to figure out the ultimate level of trust. Golbeck [2005] suggests a trust composite function based on the trust connection pattern. Golbeck's

method is one of several that will be used and tested in this study to see whether trust derived through web user interactions has propagative and aggregative features.

1.2.3 Trust Metrics

Trust Metrics are a technique for predicting the trust levels of users in a social network. Trust metrics generate quantifiable estimations of how much trust an agent **A** should have in its peer **B** based on trust evaluations from other people in the social network [6]. They use controlled trust propagation to predict the trust score of unknown users to a specific user. In general, trust modeling follows two basic axes: local trust metric and global trust metric [22]. Local trust metrics suffer from a cold start problem, since they cannot deal with new users or users who do not have any connections or friends [23]. This issue has no bearing on global trust metrics. It is efficient even to cold start users, which increases its efficiency.

- **Global Trust Metrics:** Global trust metrics generate a global rank or trust value for the network's nodes or users [22]. They consider all the trust connections in the trust graph and provide a global rank or trust rating for the nodes. They are simply calculating the trust value based on the user's reputation in the community or network.
- **Local Trust Metrics:** Local trust metrics use trust propagation techniques to determine the customized trust value between a source node and a destination node. Local trust metrics consider the opinions of partial users, typically from the trustor's neighborhood.

1.3 Online Social Network of Trust

A social network of trust is a network of users who share the same trust signifier. These networks display genuine people with verifiable identities, skills, accomplishments, and reviews presented on a social network. According to information technology experts, “social networking looks to provide a rich backdrop for criminal activity and other crimes” [18]. According to evidence from numerous OSNs, millions of social network users do not hesitate to share their opinions, experiences, photographs, files, videos, and connections in an environment that primarily lacks security conventions. Users trust other community members with their knowledge, identification, personal information, and, in some instances, money loans [19]. Given this, social networking takes place within a (mostly unjustified) context of trust. OSNs are environments that connect their members’ various social circles, ranging from close friends and family members to strangers [20].

To put it differently, OSNs are made up of various social connections of varying strength. We looked at implementing a system where these connections are formed through mutual referrals or recommendations. Connections between people via OSNs are challenging because they entail interactions with strangers. The concept of trust is essential to OSNs. Our everyday social lives, which we take for granted, would be impossible without trust [2]. A social network of trust connects people who are related to one another. It assumes that you know and trust the individuals with whom you may interact and share resources and that you, in effect, know and trust the people they know and trust.

1.4 Problem Description

In 2019, the reported damages due to scamming totaled $134 million. However, losses from frauds rose to new highs in 2018, reaching roughly $117 million in the second quarter alone [8]. During this time, frauds on social media were often tied to shopping, romance, and other money-related frauds. It is estimated that more than 1 million people might become scamming victims each year. As social media users grow, this incidence will likely rise. How can you make an online purchase without getting ripped off? How can you trust someone who has no public profile but wants to send you a message or sell a product to you? With these concerns in mind, finding ways to build trust online became increasingly important, especially concerning greater information exposure to social media. This paper proposes a model to evaluate trust in a social network to provide secure and trusted online interactions in the face of corrupt individuals attempting to harm or steal from them.

1.5 Project Overview

A large social network site is interested in measuring the trustworthiness of members. Quality assessments are hard to compute when person A and person B are unconnected in the online social network. This study aims to find ways to use the social network structure and the trust links between them to predict how much two people who are not directly connected could trust one another. Inferring such trust values is hard to compute when person A and person B are unconnected in the social network. This paper presents a graph-based model to evaluate trust in an online social network. It focuses on the problem where given a graph of quality assessments of people; how do we compute a quality assessment about person A for person B, considering especially that person A and B are unconnected? To carry out this computation, one can first consider how the graph structure is set up. There are no edges between nodes A and B because they are not connected. So, it would help if you computed a quality assessment or trust value for A using only edges that connect with persons directly connected to node A on the graph.

In summary, if two people are not directly connected, a trust inference uses the social network paths that connect them, as well as the trust values along those paths, to decide how much one person should trust the other.

1.6 Research Questions

i. How do we compute a trust assessment about person A for person B, considering that person A and B are unconnected on a social network?

ii. Which existing models and algorithms are utilized in online social networks to infer trust between strangers or people who do not know one other?

iii. Can trust attributes be used to estimate trust among people who have never met or are unconnected on a social network?

iv. How can the proposed algorithm for inferring trust between strangers on an online social network be validated?

These questions will be investigated with the use of a trust metric algorithm and a set of survey and real-world datasets.

1.7 Objectives

1.7.1 Main Objective

This project aims to develop an accurate and efficient trust metric algorithm that can provide a trust assessment about person A for person B, who are not connected on a given social network.

1.7.2 Specific Objectives

The following are the project's specific goals:

1. Demonstrate by using an appropriate graph structure to represent the users' social network.
2. Develop an algorithm for inferring trust using the most trusted and shortest path through a graph given source (trustor) and destination (trustee).
3. To validate the algorithm for inferring trust between peers on a social network with at least 90% accuracy.

1.8. Justification of study

Due to the apparent increase in online social network interactions and information sharing, it is more important than ever to create and measure trust in a social network for people's privacy, safety, and livelihood. Individuals are conversing with and listening to other consumers, whether purchasing for business or personal use [31]. When deciding who to engage in consumer-to-consumer e-commerce and with whom, trust is very vital, especially for consumers who have not met physically or face-to-face in the physical world. A social media platform can change how often and how far a message from a user spread by assigning trust scores and making them known to users. This also incentivizes users to behave better [32]. Based on their trust score, the algorithm will limit the reach of a fraudulent seller's messages and content. As a result, the system will be utilized as a tool for individuals and organizations to engage in secure and trusted online interactions in the face of unscrupulous individuals attempting to harm or steal from them.

CHAPTER 2: LITERATURE REVIEW

This Chapter begins with a brief review of the Graph-Based Trust evaluation model. It concludes with a discussion of the existing algorithms and how they compare to the solution presented in this study.

2.1 Background

Some authors or researchers have pioneered a new field of research, Graph-Based Trust Evaluation in Online Social Networks. It presents an evaluation paradigm for social networks based on the premise that individuals' trustworthiness may be inferred from their social relationships. Social media networks like Facebook and Twitter are designed to connect us with the information we need and the people we care about. They are organized around users and symbolize a mapping from our actual life to the cyber-physical realm. What is the operation of these social media networks?

Furthermore, how does their design affect our trust in one another and society as a whole? The objective of this work is to create algorithms for inferring a trust value from one person to another in the absence of a direct connection between them in the network. Trust is defined as "a measure of confidence that an entity or entities will behave as expected [4]". Also, the computational model defines trust as "the subjective probability by which one user expects another user to do a given action. [5]" As a result, trust has various definitions and categories, and it requires a more significant grasp of the term to be studied or evaluated.

From a network viewpoint, trust models can be divided into two types: those that use a local approach that considers personal bias and those that use a global approach that considers all users' perspectives [6]. In graph-based models, the local approach is commonly utilized. Users are represented as nodes, while trust relationships (links) between users are represented as edges. The trust relationship can then be formed through many means, such as exchanging personal information such as how long you have known the user, the user's opinion about products or people, mutual friends, etc. Graph-based trust evaluation is an innovative approach for assessing user trustworthiness in online social networks. As a result, they apply to a wide range of existing graph-based models in OSNs.

2.2 Related Work

2.2.1. Tidal Trust: Tidal Trust is an algorithm proposed by Golbeck [2005]. Tidal Trust generates a recommendation based on trusted pathways for the trust degree that one person can place on the other given two people in a network. The trusted paths are discovered by conducting a breadth-first search from the trustor to the trustee. Tidal Trust employs only the shortest and most potent trusted paths. The trust computation from **s** to **d** is as follows:

$$t_{sd} = \frac{\sum_{j \in Ns,\, t_{sj} \geq max} t_{sj} t_{jd}}{\sum_{j \in Ns,\, t_{sj} \geq max} t_{sj}} \qquad (1)$$

Where Ns is s's neighbor set and max are the trustworthiness threshold (i.e., j is considered trustworthy only if $t_{sj} >$ max). The name "Tidal Trust" was chosen because the calculating process is akin to a tidal stream: calculations sweep forward from the trustor to the trustee, then pull back from the trustee to return the final value to the trustor.

According to Golbeck [2005], shorter propagation channels and paths with greater trust levels yield more accurate trust estimations. Therefore, calculation employs just the shortest paths to the sink, and the threshold is dynamically computed to favor nodes with higher trust scores. One significant disadvantage of this technique is that it is not always efficient because the longer chain path may hold essential information that should not be overlooked.

2.2.2. Mole Trust: Mole Trust is a two-step process proposed by Massa et al. [2005]. The first stage removes cycles by sorting users by their proximity to the trustors. Then it considers all users up to a maximum depth, which is provided as an input. It is vital to remember that the maximum depth is not dependent on the user. As a result, the trustworthy graph they employ is a reduced Directed Acyclic Graph (DAG). Mole Trust begins by calculating the trust degrees of users one step away from s, then two steps, three steps, and so on. Furthermore, the trust degree of a user who is k steps away from the source is only dependent on the trust degree of users who are k-1 steps away. As a result, the information provided by each user is only used once. Equation (2) is used to calculate the weighted average of all the trustful incoming neighbors' trust in the trustee d:

$$t_d = \frac{\sum_{i \in N_d^+} t_i t_d}{\sum_{i \in N_d^+} t_i} \qquad (2)$$

where N_d^+ denotes d's trustful incoming neighbors and t_i denotes i's trust degree. With a slight modification, the Mole Trust algorithm [28] overcomes the issue mentioned by the Tidal Trust algorithm. Although the Mole Trust algorithm is based on the shortest-path distance from the source user, only persons with propagative distances less than or equal to the trust propagation horizon are factored in. Horizon is the maximum distance from the source user to which trust may be expected to propagate, and it is unrelated to any specific person. Mole Trust used breadth-first search while Tidal Trust used depth-first search to find the highest trust users. Mole Trust is more efficient than Tidal Trust since it only requires a single program for each source produced instead of each source-sink pair. The substantial time complexity of this method, however, is one downside.

2.2.3. Eigen Trust: Eigen Trust is an algorithm that starts with a peer and looks for trustworthy peers based on the outlined rules [29]. It goes from one peer to another with a probability proportional to the trust score of the other peer, i.e., the greater the trust score, the higher the moving likelihood. As a result, Eigen Trust will reach more trustworthy peers than untrustworthy peers. On the other hand, this algorithm only provides trust rankings rather than absolute trust levels of peers/objects. Eigen Trust is a global trust metric compared to Mole Trust and Tidal Trust, which are both local trust metrics. The experiments show that a Local Trust Metric outperforms a Global Trust Metric in determining an individual user's trust [28].

2.2.4. Graph Convolutional Neural Networks (GCNs) - Guardian: Furthermore, some related work, termed Guardian, employs a different approach based on Graph Convolutional Neural Networks [10]. Guardian is an end-to-end approach that stacks various trust convolutional layers to find hidden and predicted latent trust variables in online social networks. Because the social trust may be represented as graph data, GCNs have a high potential for trust evaluation. Guardian is intended to measure social trust between any two users by incorporating social network structures and trust connections.

This technique tackles how to express social connections and associated trust relationships so that social trust's propagative and composable natures can be recorded concurrently without one user trusting more than the other.

2.2.5. Using Probabilistic confidence Models – SUNNY: This method uses probability models to describe trust values as probabilities and evaluate trust using probability functions. SUNNY is the name of one such technique. SUNNY is a trust inference algorithm that calculates our confidence in trust information from defined sources using a probabilistic sampling technique [3]. This technique considers the accuracy of the trust value and the measure of confidence. SUNNY was proved to outperform the well-known TIDAL TRUST inference algorithm in computed trust values accuracy.

2.2.6 Multiplicative Strategy for Trust Propagation

The multiplicative strategy is quite simple. Consider a trusted path, P, in a social network as <u1, u2, u3, …, un>, where <u1, u2, u3, ..., un> are the users along the path. The simple multiplication technique for trust prediction calculates trust Ti as it travels down the path where t(ui,uj) is the direct trust value between users ui and uj [35].

$$Ti = t(u1 \times u2) \times t(u2 \times u3) \times \ldots \times t(u_{n-1} \times u_n) \qquad (3)$$

$$\exists 0 < t(u_i, u_j) \leq 1$$

Where $t(u_i, u_j)$ is the direct trust value between u_i and u_j. The propagated trust value is the product of all the trust values along a path.

2.2.6 Summary of Related Work

As an outcome of the research, global trust represents general reputation, whereas local trust means opinion. Graphs can estimate both global and local trust, despite employing different network ranges for trust evaluation. Local trust metrics are more reliable than global trust metrics in assessing trust. The Mole Trust metric offers shorter propagation channels and pathways with higher trust levels, resulting in more accurate estimates [3]. It also features a configurable trust propagation horizon or level as an input parameter [30]. The multiplicative strategy is simple and easy to implement but gives a trust value that reflects the absolute trust on the trusted path. For example, if all the trust values on a trusted path are 1, the propagated trust value is also 1. The Eigen Trust metric gives a global trust value for all nodes in the network using the power iteration approach.

CHAPTER 3: METHODOLOGY

This chapter describes how the proposed algorithm is designed and executed to build a social trust network among users. It also describes the dataset that was used to evaluate the system. The system is divided into the following fundamental steps: (i) gathering information for trust, that is, gathering evidence to assess a user's level of trust; (ii) building a trusted graph, that is, managing trust evidence with a graph; (iii) conducting trust evaluation, that is, designing algorithms to calculate trust degree; and (iv) applying the results to other applications (e.g., security management, recommendation system).

3.1. Data Acquisition

We used three techniques to collect data: user input through a web application, real-world survey data, and an online available real-world dataset.

3.1.1. Web application data: We created a web application connected to a Neo4J graph database that could accept users' registration information and provide them with a unique ID, also known as the "TrustHop ID," to identify them in the social network. The user could make trust inferences about other users in the social network using this "TrustHop ID," assuming they have friendships, affiliations, financial exchanges, trade linkages, and relations. User interfaces of the web application can be seen in Figure 3.7, 3.8, 3.10, 3.12, 3.14, and 3.15.

3.1.2. Real-world survey data: To collect a real-world trust dataset to verify the accuracy of our algorithm, we used Google forms to conduct a survey. A questionnaire was developed and evaluated in an undergraduate class of 2022 at Ashesi University, where 134 students participated in the survey (Appendix A) and contributed to the 196 trust assertions. Students were asked to rate their close friends on a scale of 1 to 10 (Least trusted-most trusted). The results of this survey were utilized to examine the algorithm.

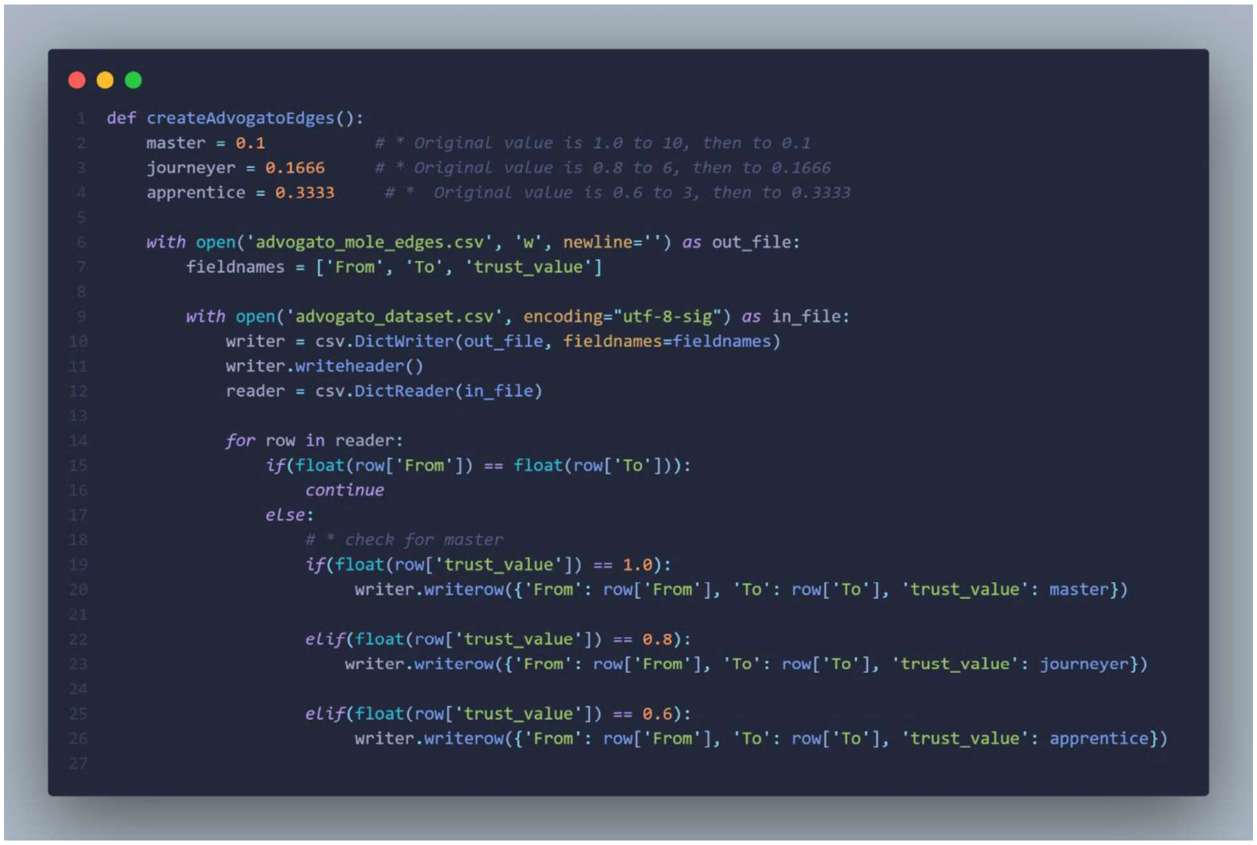
3.1.3. Online available Real-world dataset:

- **Advogato**: We used Advogato [33], an online forum for open-source software engineers, as the primary data set for our study. Advogato is a social community platform that allows members to establish explicit weighted trust connections among themselves. A list of all user-to-user linkages is included in the dataset. The site's users score each other on their level of trust. The trust values preferences are master, journeyer, and apprentice, with the master being the highest level in that category. Because it is possible to trust oneself on Advogato, there are self-loops in the dataset, but we eliminate them because they do not adhere to our model. For its three trust values, we substitute the following values: master = 1, journeyer = 0.8, and apprentice = 0.6 to values 1.0, 0.66, and 0.33, respectively which we used for the multiplicative strategy calculation. We then used updated values, thus setting master = 10, journeyer = 6.6, and apprentice = 3.3, and taking the reciprocal of these values for the trust propagation calculation. Suppose a given set of three ratings to understand the design: 3, 4, and 5.

  Ratings: 3, 4, 5 (Ascending order)

  Reciprocals: $\frac{1}{3}, \frac{1}{4}, \frac{1}{5}$ (Descending order ~ 0.33, 0.25, 0.2)

  The aim is to raise the smaller trust value while decreasing the larger value so Dijkstra's algorithm can acquire the most trusted path. The consequence of these member ratings is a rich network of trust, and after eliminating the self-loops, the final datasets include 6,551 users and 47,337 trust ratings. The trust values of the Advogato web of trust are distributed as follows: master: 17,319, journeyer: 21,349, and apprentice: 8,669. See Figure 4.2. The Advogato web of trust instance referred to in this paper was obtained from Network repository website [33]. Script used to remove self-made trust inference of users Fig. 3.1



```python
def createAdvogatoEdges():
    master = 0.1            # * Original value is 1.0 to 10, then to 0.1
    journeyer = 0.1666      # * Original value is 0.8 to 6, then to 0.1666
    apprentice = 0.3333      # *  Original value is 0.6 to 3, then to 0.3333

    with open('advogato_mole_edges.csv', 'w', newline='') as out_file:
        fieldnames = ['From', 'To', 'trust_value']

        with open('advogato_dataset.csv', encoding="utf-8-sig") as in_file:
            writer = csv.DictWriter(out_file, fieldnames=fieldnames)
            writer.writeheader()
            reader = csv.DictReader(in_file)

            for row in reader:
                if(float(row['From']) == float(row['To'])):
                    continue
                else:
                    # * check for master
                    if(float(row['trust_value']) == 1.0):
                         writer.writerow({'From': row['From'], 'To': row['To'], 'trust_value': master})

                    elif(float(row['trust_value']) == 0.8):
                        writer.writerow({'From': row['From'], 'To': row['To'], 'trust_value': journeyer})

                    elif(float(row['trust_value']) == 0.6):
                         writer.writerow({'From': row['From'], 'To': row['To'], 'trust_value': apprentice})
```

Figure 3.1: Implementation of Python script to remove self-made trust inference of users

3.2 Building a Web of Trust

After collecting the data, we developed a graph network with links (recommendations) among members. When a person registers through the web application, their information is stored in the network as a node. Then, when he makes recommendations for someone who already exists, a relationship is formed between them. The graph network develops into a web of trust due to user registration and referrals.

Furthermore, we built a trusted network using the Real-world survey data and the publicly available Real-world dataset. We created a script before utilizing the dataset that excluded all users who submitted referrals to themselves because this would not adhere to our proposed model. Then, we transformed the data such that it could be saved in a CSV file. The graph network is then created using the Neo4J desktop software, which can read data from a CSV file. Examples of the query used is given by Code 3.1, Code 3.2, and Code 3.3

```
CREATE CONSTRAINT ON (n: User) ASSERT n.user_id IS UNIQUE
```

Code 3.1 A cypher query that creates a unique constraint on the user id property.

```
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS FROM
'file:///advogato_nodes.csv' AS row WITH toInteger(row.node_id) AS
node_id CREATE (p:User {user_id: node_id}) RETURN count(p)
```

Code 3.2 A cypher query that reads a csv file to create a network.

```
:auto USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS FROM
'file:///advogato_multi_edges.csv' AS row WITH toInteger(row.From) AS
node_from, toInteger(row.To) AS node_to, toFloat(row.trust_value) AS
trust_value MATCH (p1:User {user_id: node_from}), (p2:User {user_id:
node_to}) CREATE (p1)-[r:TRUST {trust_rating: trust_value}]->(p2)
RETURN count(r)
```

Code 3.3 A cypher query to create links among nodes.

3.3 Conducting Trust Evaluation

If a person wants to know the trustworthiness of another member in any online social network with a trust rating system, our goal is to locate the most trusted and quickest path in the shortest feasible period.

3.3.1. Focused Algorithm (Trust Propagation Function)

After obtaining the quality assessment graph, we use a trust propagation function [36] to infer trust between people not directly connected in the social network. We search for the shortest and most trusted paths from the source node **s** (trustor) to the target node **t** (trustee) and then use the trust propagation function to average the propagated trust values from the trusted path. The propagated trust value for the set of selected nodes along the shortest and most trusted path from source node **s** (trustor) to target node **t** (trustee) is the average of the trust ratings from each node in P weighted by the propagative distance from the source node to each node i [36].

$$Propagated_{trust} = \frac{\sum_{i,j \in P}^{i=s} \left(d_j \times trust_{ij} \right)}{\sum d_j} \quad (4)$$

Where P denotes the trusted path, s is the source node, and t denotes the target node. A weighted average of the trust ratings from each node in P multiplied by the propagative distance di from the s to each node i in P. The pathfinding method integrated within the Neo4j Graph Data Science library was employed. The Dijkstra Source-Target Shortest Path Algorithm is one of these algorithms, and it is used to identify the shortest path between a source and a target [37]. We used the said algorithm in finding the shortest and most trusted path in the network.

3.3.2. Finding the Trusted Path

To determine the trust path in our algorithm, we implemented a methodology in which, at each level, we select the edge with the highest trust rating rather than the lowest, like in Dijkstra's algorithm. The reciprocals of all the trust rates in the trust dataset are used to replace them. Then we utilize Dijkstra's algorithm to discover the shortest path, from which we may get the most trusted path.

3.3.3. Trust calculation

We devised three techniques to calculate a user's trust rating for an unknown user in the network. We compared them to see which one was more accurate and efficient in terms of time. We utilized a Python script to eliminate all trust inferences users had generated about themselves before using these techniques.

We put this into effect by following the procedures outlined below:

i. We queried the graph database using the cypher query language in the neo4j database to find the shortest distance between the two nodes (trustor and trustee) using the Dijkstra algorithm. The cypher query is given in Code 2.4.

ii. Using the neo4j JavaScript drivers, the Neo4j graph database is connected to a Node JS server Application Programming Interface (API). The API has endpoints that accept the unique ID of the trustor and trustee and pass it to the query, which is then sent as a request to the graph database.

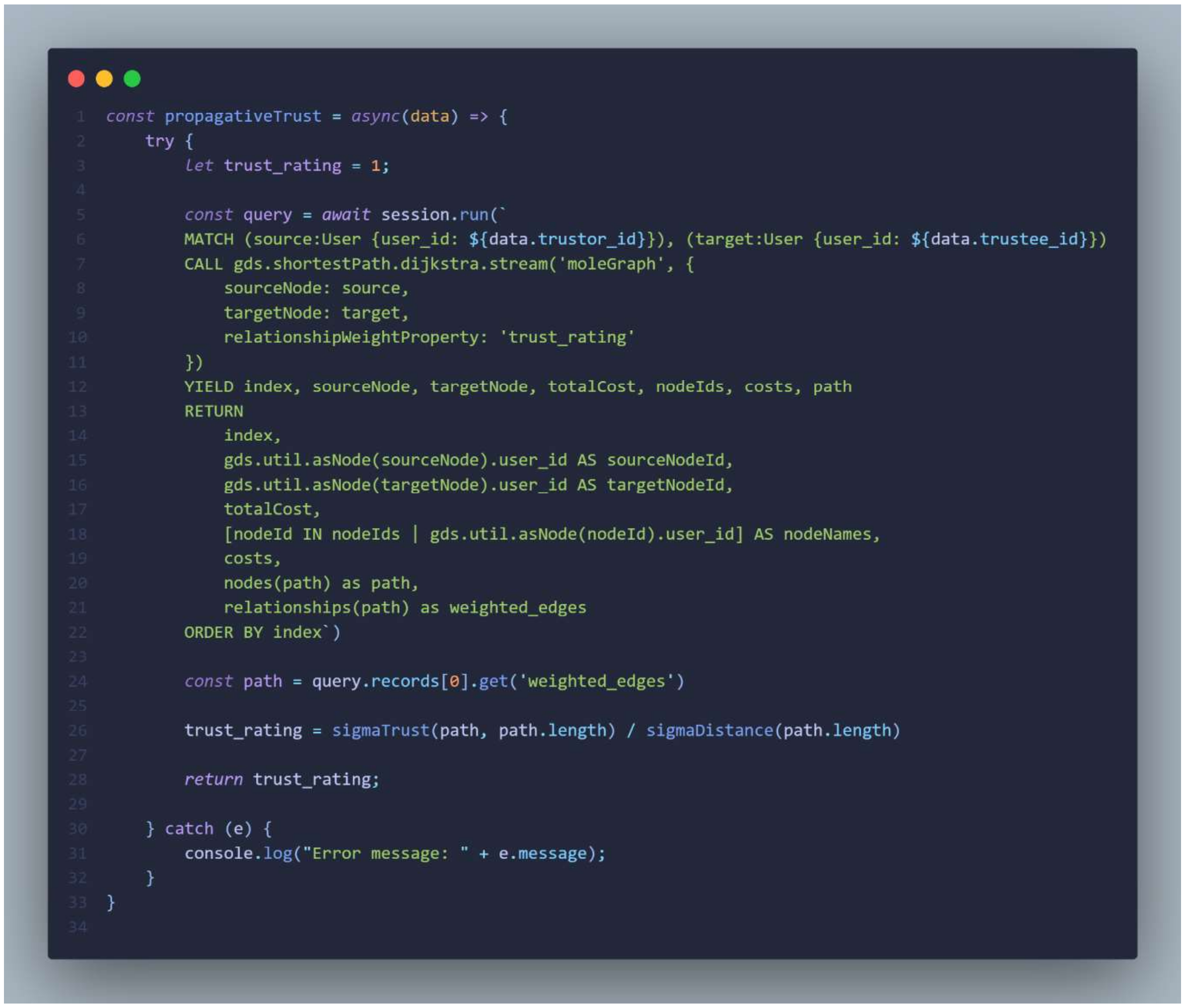
iii. The response is received once the query has been successfully completed on the database. The response is the shortest and most trusted path with the weighted edge values.

iv. The API provides a function that uses the equation (3) to determine the trust value using the Multiplicative Strategy for Trust Propagation [35]. The code can be seen in Fig. 3.3. Furthermore, another function uses equation (4) to calculate trust value based on the trust propagation function [36]. The code is shown Fig. 3.2

v. The results from the API's function calculations are then moved to an excel file for more analysis.

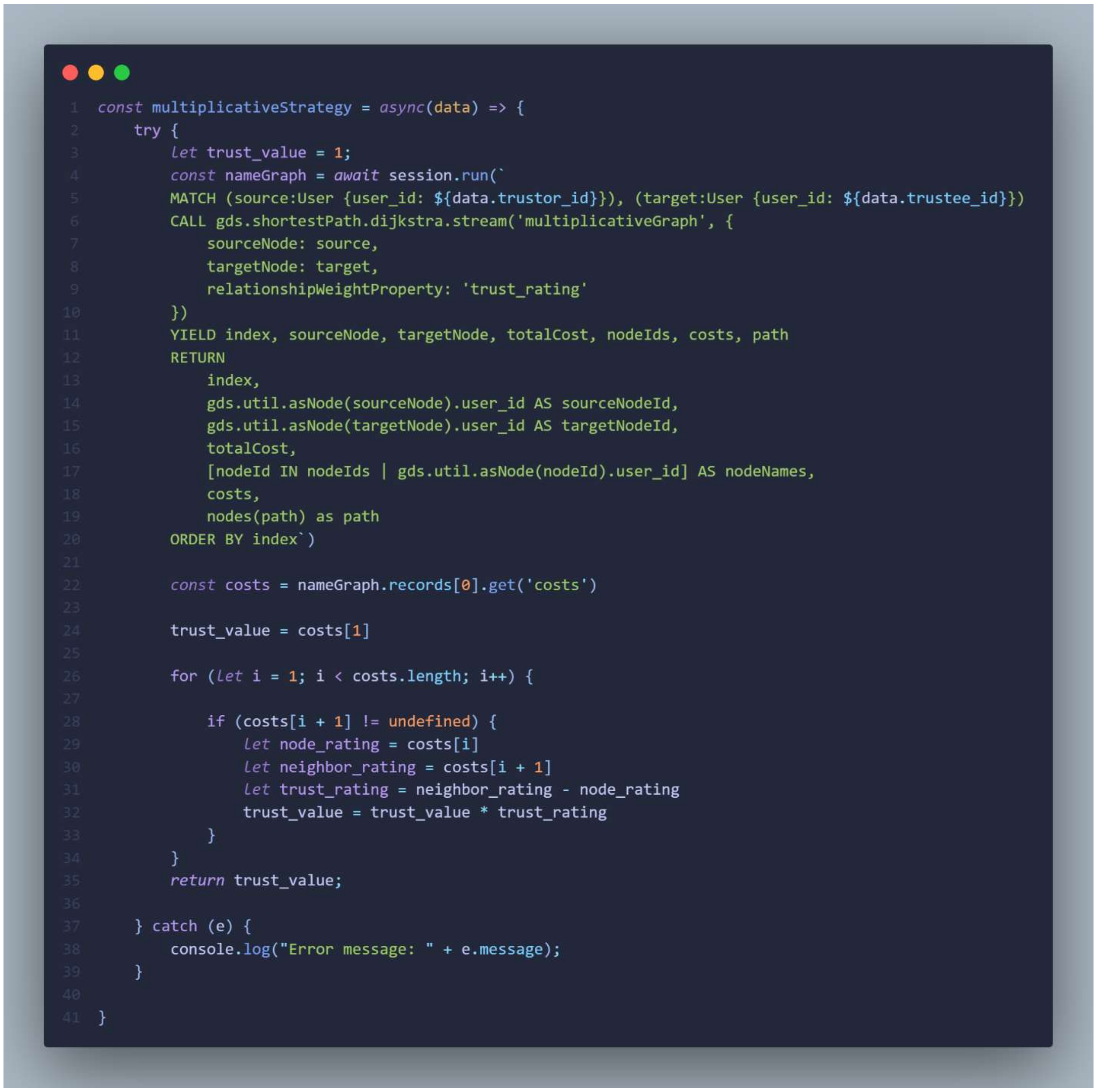
```
MATCH (source:User {user_id: 12}), (target:User {user_id: 22}) CALL
gds.shortestPath.dijkstra.stream('advogatoGraph', {
sourceNode: source,
targetNode: target,
relationshipWeightProperty: 'trust_rating'})
YIELD index, sourceNode, targetNode, totalCost, nodeId, costs, path
RETURN index, gds.util.asNode(sourceNode).user_id AS sourceNodeId,
gds.util.asNode(targetNode).user_id AS targetNodeId, totalCost,
[nodeId IN nodeIds | gds.util.asNode(nodeId).user_id] AS nodeNames,
Costs, nodes(path) as path ORDER BY index
```

Code 3.4 A cypher query that uses the Dijkstra algorithm to find the shortest path between two nodes.



```
const propagativeTrust = async(data) => {
    try {
        let trust_rating = 1;

        const query = await session.run(`
        MATCH (source:User {user_id: ${data.trustor_id}}), (target:User {user_id: ${data.trustee_id}})
        CALL gds.shortestPath.dijkstra.stream('moleGraph', {
            sourceNode: source,
            targetNode: target,
            relationshipWeightProperty: 'trust_rating'
        })
        YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
        RETURN
            index,
            gds.util.asNode(sourceNode).user_id AS sourceNodeId,
            gds.util.asNode(targetNode).user_id AS targetNodeId,
            totalCost,
            [nodeId IN nodeIds | gds.util.asNode(nodeId).user_id] AS nodeNames,
            costs,
            nodes(path) as path,
            relationships(path) as weighted_edges
        ORDER BY index`)

        const path = query.records[0].get('weighted_edges')

        trust_rating = sigmaTrust(path, path.length) / sigmaDistance(path.length)

        return trust_rating;

    } catch (e) {
        console.log("Error message: " + e.message);
    }
}
```

Figure 3.2: Implementation of the Trust Propagation



```
const multiplicativeStrategy = async(data) => {
    try {
        let trust_value = 1;
        const nameGraph = await session.run(`
        MATCH (source:User {user_id: ${data.trustor_id}}), (target:User {user_id: ${data.trustee_id}})
        CALL gds.shortestPath.dijkstra.stream('multiplicativeGraph', {
            sourceNode: source,
            targetNode: target,
            relationshipWeightProperty: 'trust_rating'
        })
        YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
        RETURN
            index,
            gds.util.asNode(sourceNode).user_id AS sourceNodeId,
            gds.util.asNode(targetNode).user_id AS targetNodeId,
            totalCost,
            [nodeId IN nodeIds | gds.util.asNode(nodeId).user_id] AS nodeNames,
            costs,
            nodes(path) as path
        ORDER BY index`)

        const costs = nameGraph.records[0].get('costs')

        trust_value = costs[1]

        for (let i = 1; i < costs.length; i++) {

            if (costs[i + 1] != undefined) {
                let node_rating = costs[i]
                let neighbor_rating = costs[i + 1]
                let trust_rating = neighbor_rating - node_rating
                trust_value = trust_value * trust_rating
            }
        }
        return trust_value;

    } catch (e) {
        console.log("Error message: " + e.message);
    }

}
```

Figure 3.3: Implementation of the Multiplicative Strategy for Trust propagation

3.4 Applying the Results

We created a front-end web application that can retrieve API results and display them to the user. A Node JS Express server will host the web application. The web application was hosted locally for this project. The web application delivers a payload including the trustor id and the trustee id to the API as a request; the API then executes and retrieves a response from the database, which is then returned to the user.

Design and Architecture: A three-tier architecture consisting of a client/presentation tier, an application layer, and a database tier (Figure 3.4).

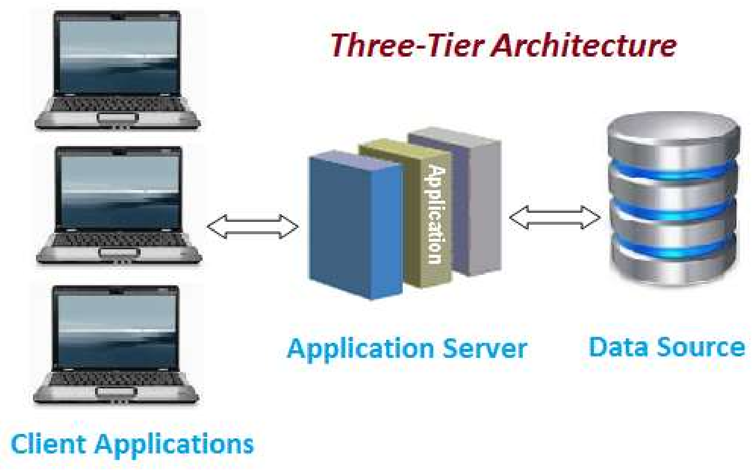


Figure 3.4: Three-tier architecture

Client/Presentation Layer: This is the application layer with which the user or client interacts. The presentation layer will explore technologies such as Bootstrap, JavaScript, CSS, and HTML.

Application Layer: This layer oversees the web application's main functionality and business logic. This layer will make use of Node JS and Express JS. PHP will also be used to handle the data.

Database Layer: This layer oversees the application's data storage and access system. The Neo4J graph database will be used in this application.

Functional requirement:

- Users should be allowed to register, and following successful registration, a unique id (TrustHop ID) should be generated for them.
- Users should be able to login to the application after registration
- Users should be able to access all recommendations made on any existing user in the social network using their unique id.
- The system should be capable of generating a trust value for the user in respect to any other user in the social network.

Non-functional requirement:

- Application interface should be both appealing and simple to utilize
- Data Storage with a Large Capacity
- Security
- Performance

Activity diagram/Use case scenario

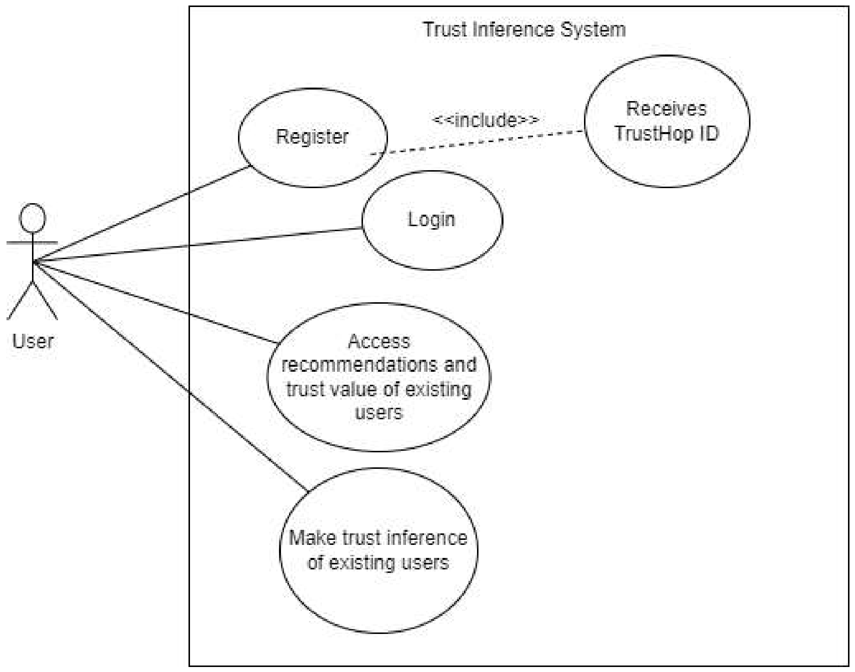

Trust Inference System
Register
<<include>>
Receives TrustHop ID
Login
User
Access recommendations and trust value of existing users
Make trust inference of existing users

Figure 3.5: A use case diagram for trust inference system

Data Model

The Neo4J graph database is used to store data in this application. By creating a data structure for the graph database, a Neo4j graph data model is meant to answer questions in the form of Cypher queries and address business and technical challenges. In a node, we store information about a user. Relationships or links between nodes provide a user's trust or quality assessment of another, with the details of the links representing the edge weight or properties. The resulting graph has weighted edges and is directed (Figure 3.6).

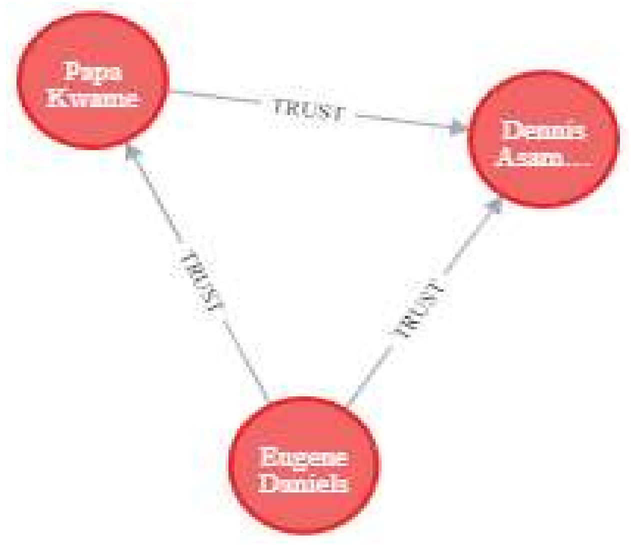


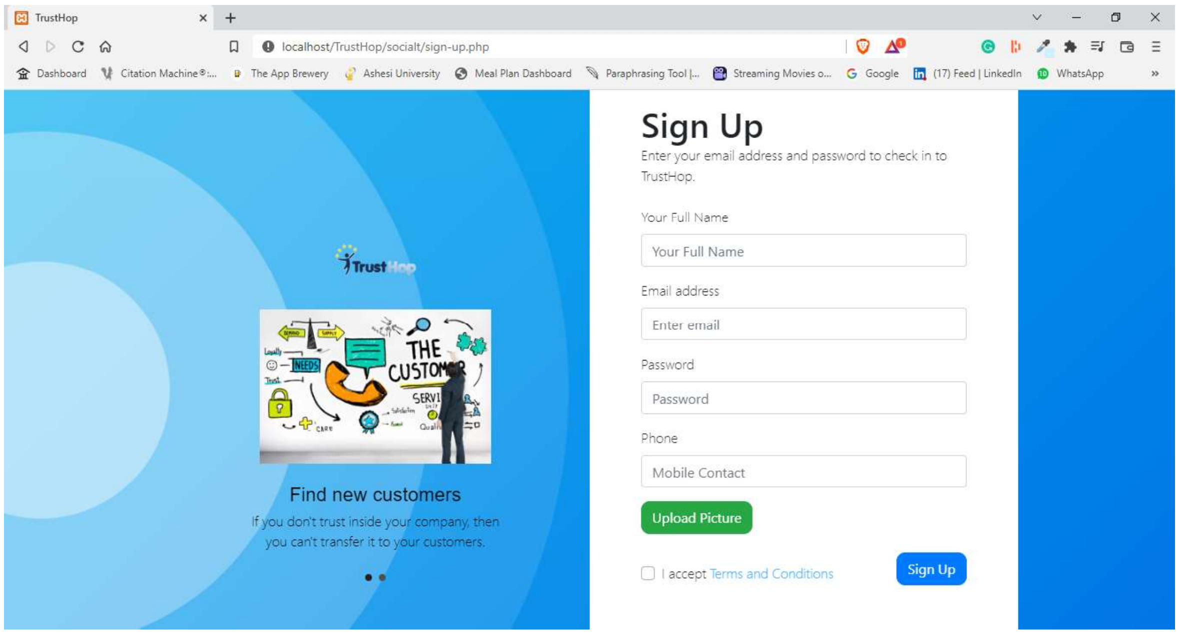
Figure 3.6: Neo4J graph data model

Implementation of Trust Hop Web Application

TrustHop's Web application is linked to or integrated with a REST API. Node JS and Express JS are used to host the REST API locally. All main application functions receive or send a response or request to the API.

Below is a snippet of the view web app and the backend implementation used for this project.

Register: This page welcomes new users to create a profile. The information provided by the user is sent to the API as a payload, and the API uses the payload to create a node in the graph database. After successful registration, a unique id is generated for the new user (Figure 3.8).



Sign Up

Enter your email address and password to check in to TrustHop.

Your Full Name: Your Full Name

Email address: Enter email

Password: Password

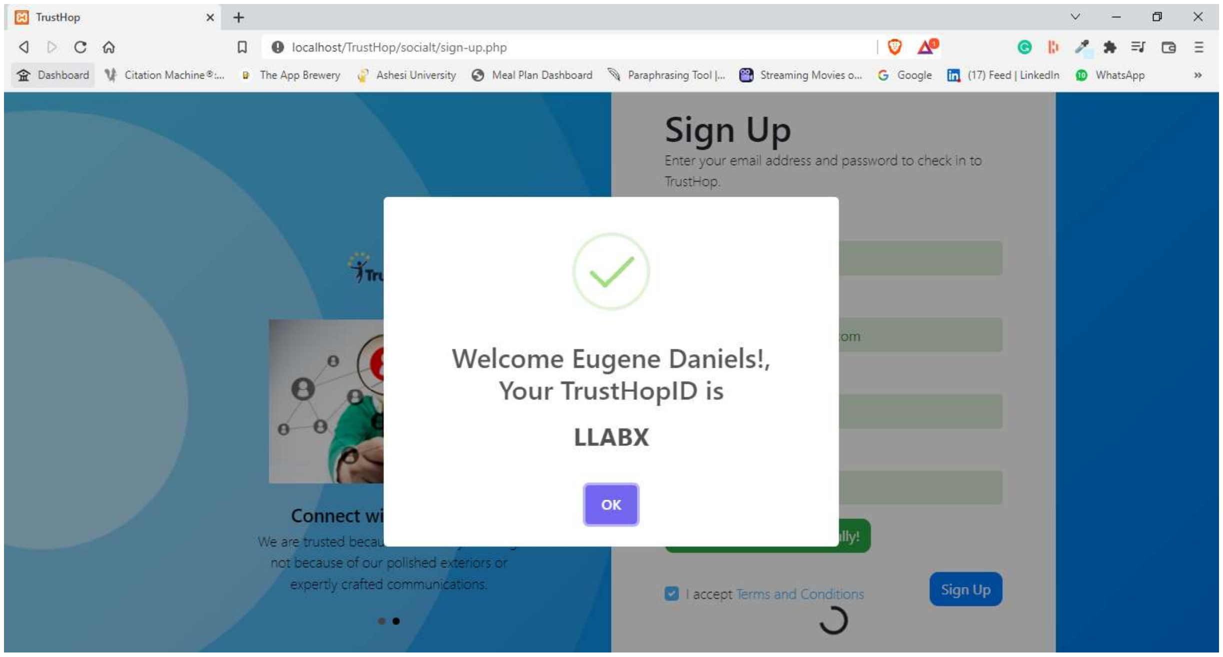
Phone: Mobile Contact

Find new customers

If you don't trust inside your company, then you can't transfer it to your customers.

localhost/TrustHop/socialt/sign-up.php
Upload Picture
I accept Terms and Conditions
Sign Up

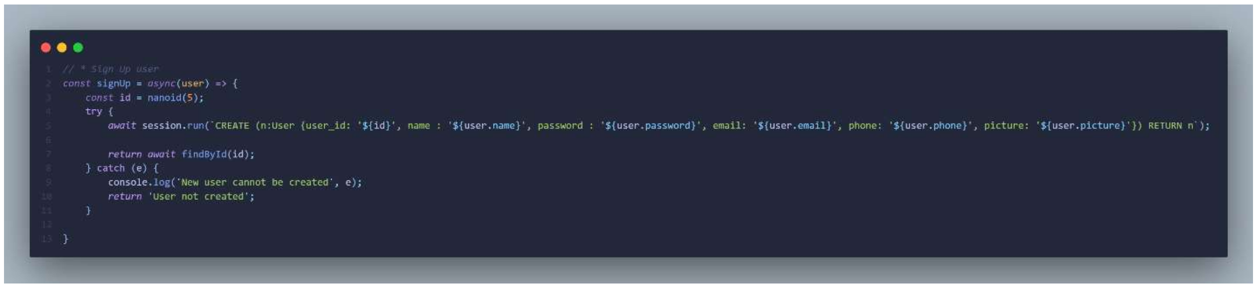
Figure 3.7: Sign up webpage



Welcome Eugene Daniels!, Your TrustHopID is

LLABX

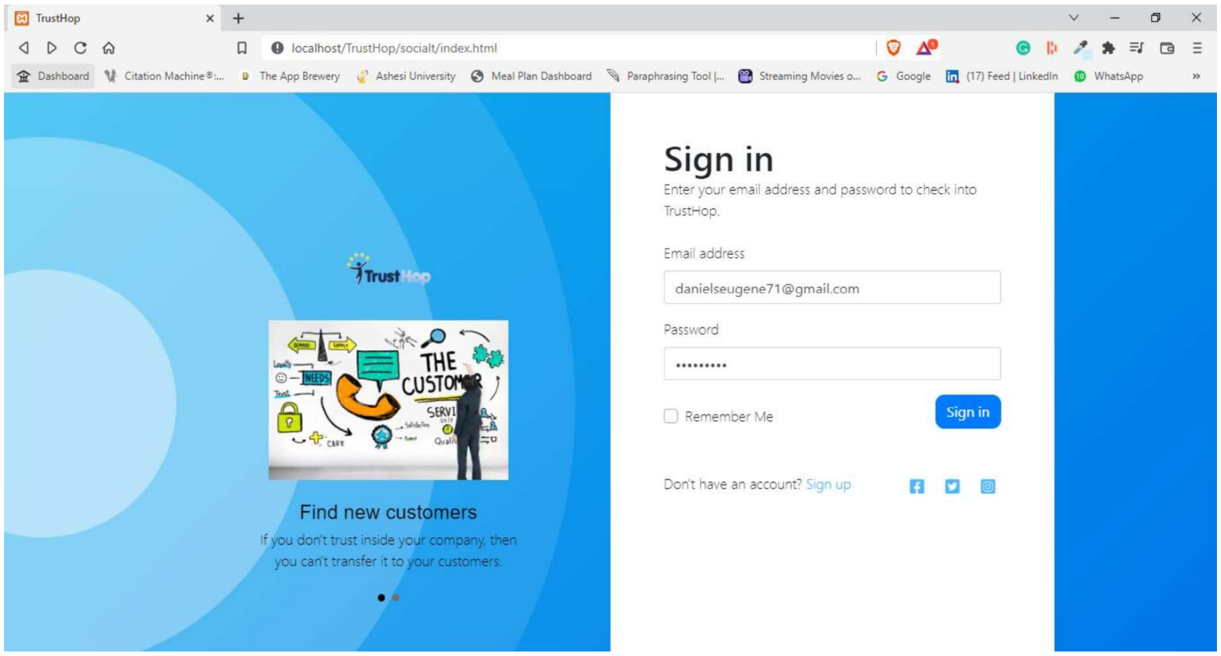
Figure 3.8: User seeing the generated unique id



```
// * Sign Up user
const signUp = async(user) => {
    const id = nanoid(5);
    try {
        await session.run(`CREATE (n:User {user_id: '${id}', name : '${user.name}', password : '${user.password}', email: '${user.email}', phone: '${user.phone}', picture: '${user.picture}'}) RETURN n`);

        return await findById(id);
    } catch (e) {
        console.log('New user cannot be created', e);
        return 'User not created';
    }

}
```

Figure 3.9: Code that implements the add new user

Login: This page serves as an interface for the user to enter his login information (email and password). The credentials are sent to the API, which searches the database to see if that user is already registered.



TrustHop

localhost/TrustHop/socialt/index.html

Sign in

Enter your email address and password to check into TrustHop.

Email address

danielseugene71@gmail.com

Password

•••••••••

Remember Me

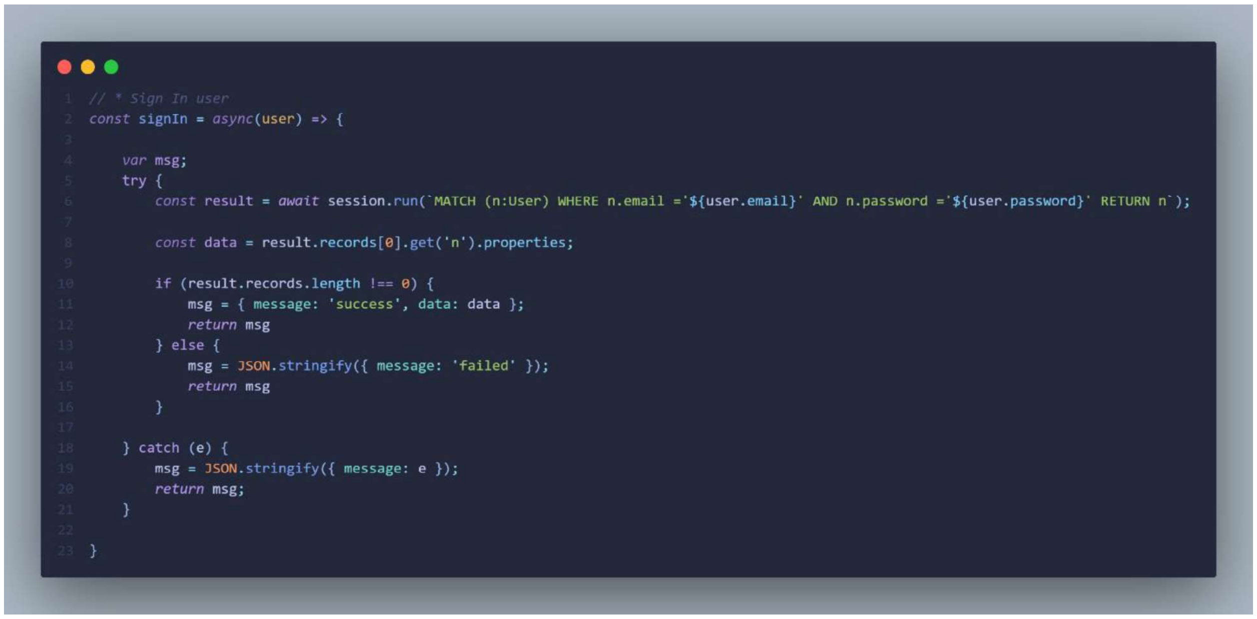
Sign in

Don't have an account? Sign up

Find new customers

If you don't trust inside your company, then you can't transfer it to your customers.

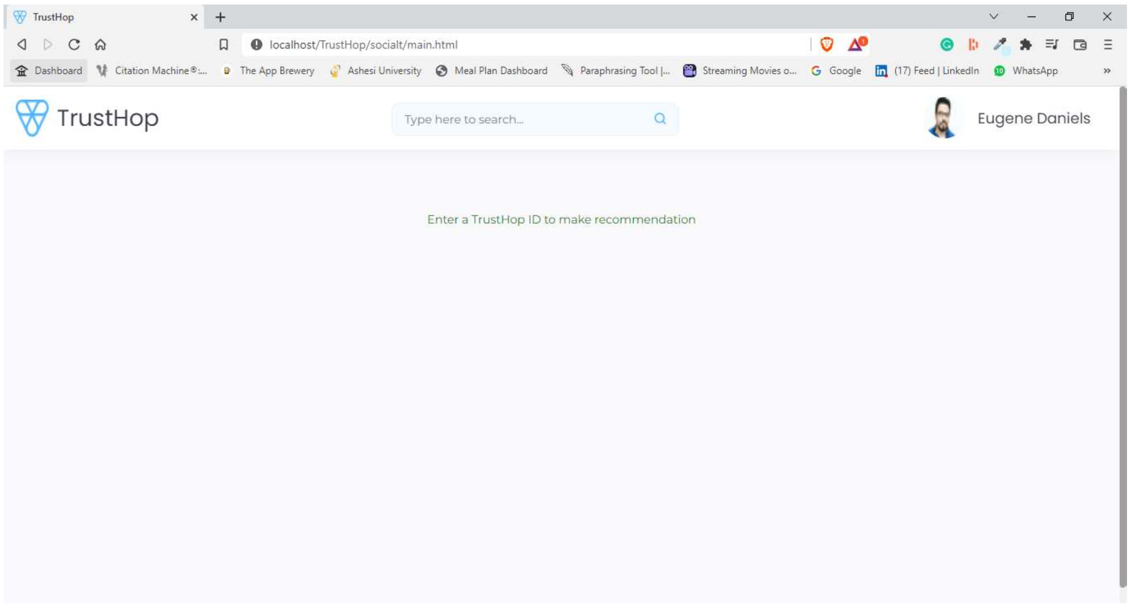
Figure 3.10: Sign in webpage



```
// * Sign In user
const signIn = async(user) => {

    var msg;
    try {
        const result = await session.run(`MATCH (n:User) WHERE n.email ='${user.email}' AND n.password ='${user.password}' RETURN n`);

        const data = result.records[0].get('n').properties;

        if (result.records.length !== 0) {
            msg = { message: 'success', data: data };
            return msg
        } else {
            msg = JSON.stringify({ message: 'failed' });
            return msg
        }

    } catch (e) {
        msg = JSON.stringify({ message: e });
        return msg;
    }

}
```

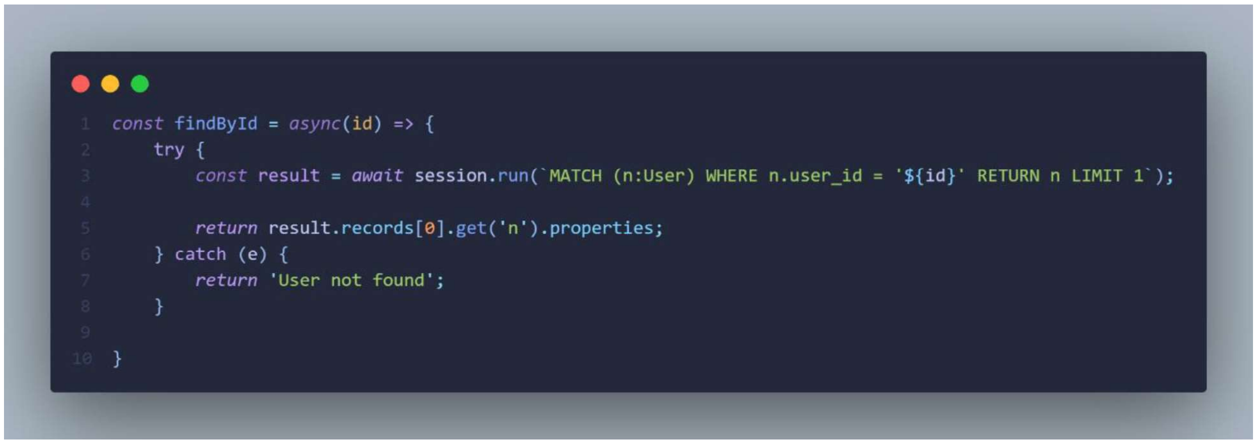
Figure 3.11: Code that implements the add new user

Home: A logged-in user can search for an existing user and assess their trustworthiness (Figure 3.12). The user searches using the "TrustHop ID" of the user.



Enter a TrustHop ID to make recommendation

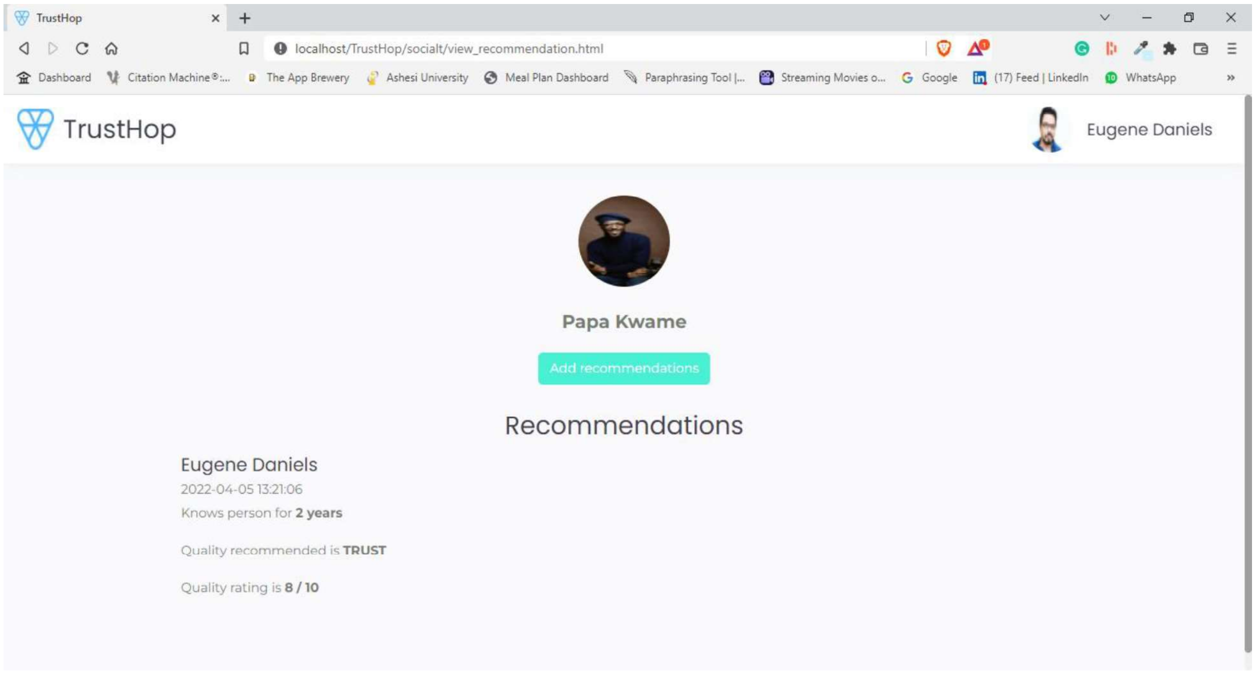
Figure 3.12: Home or search webpage



```
const findById = async(id) => {
    try {
        const result = await session.run(`MATCH (n:User) WHERE n.user_id = '${id}' RETURN n LIMIT 1`);

        return result.records[0].get('n').properties;
    } catch (e) {
        return 'User not found';
    }

}
```

Figure 3.13: Code that implements the search user

Recommendations: Users can view recommendations made on searched users and give their own recommendations or evaluations to the user (Figure 3.14 and Figure 3.15).



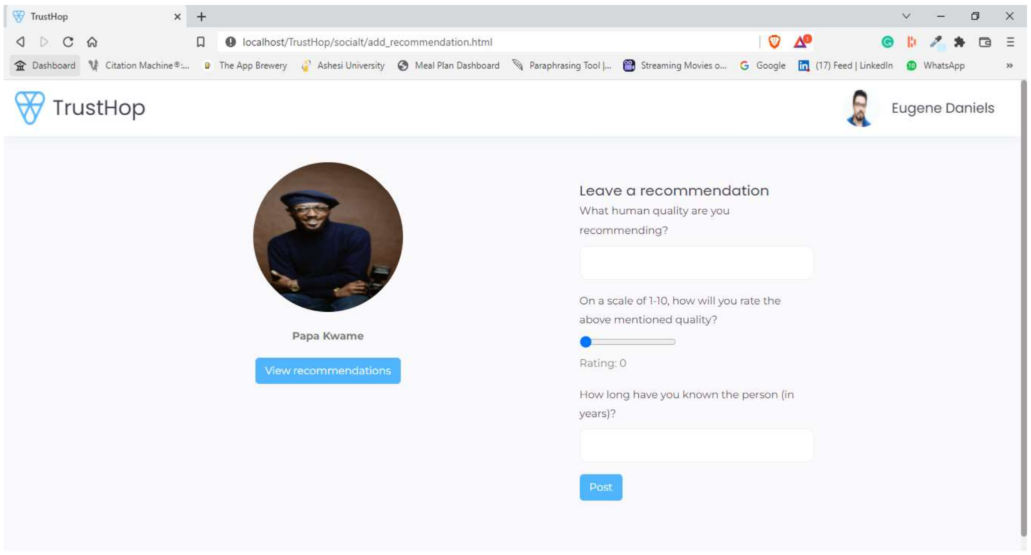
Recommendations

Eugene Daniels
2022-04-05 13:21:06
Knows person for **2 years**

Quality recommended is **TRUST**

Quality rating is **8 / 10**

Figure 3.14: View recommendation webpage



Leave a recommendation

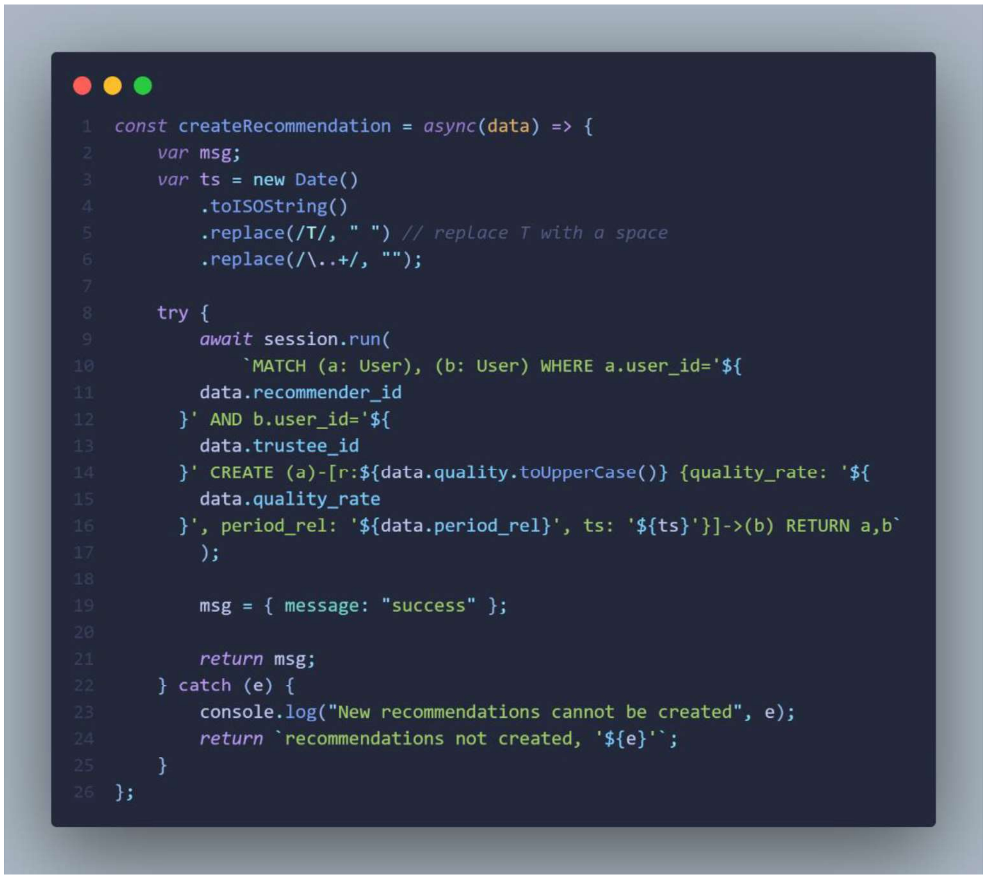
What human quality are you recommending?

On a scale of 1-10, how will you rate the above mentioned quality?

Rating: 0

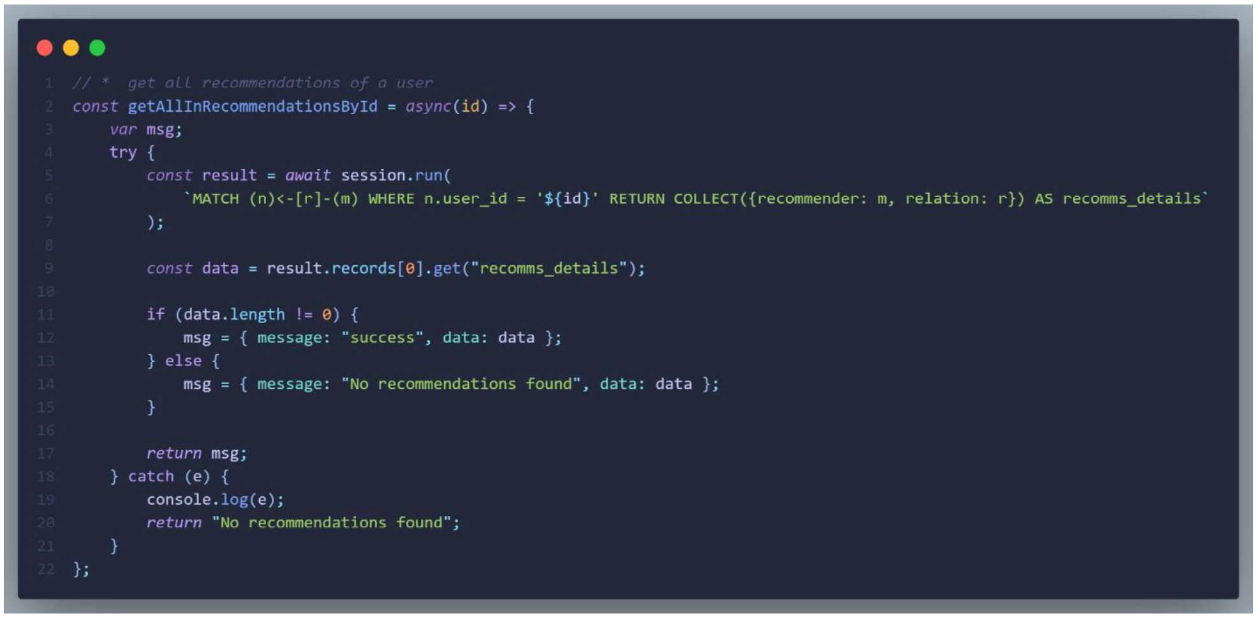
How long have you known the person (in years)?

Figure 3.15: Add recommendation webpage



```
const createRecommendation = async(data) => {
    var msg;
    var ts = new Date()
        .toISOString()
        .replace(/T/, " ") // replace T with a space
        .replace(/\..+/, "");

    try {
        await session.run(
            `MATCH (a: User), (b: User) WHERE a.user_id='${
        data.recommender_id
      }' AND b.user_id='${
        data.trustee_id
      }' CREATE (a)-[r:${data.quality.toUpperCase()} {quality_rate: '${
        data.quality_rate
      }', period_rel: '${data.period_rel}', ts: '${ts}'}]->(b) RETURN a,b`
        );

        msg = { message: "success" };

        return msg;
    } catch (e) {
        console.log("New recommendations cannot be created", e);
        return `recommendations not created, '${e}'`;
    }
};
```

Figure 3.16: Code that implements the creation of links in the network



```javascript
// *  get all recommendations of a user
const getAllInRecommendationsById = async(id) => {
    var msg;
    try {
        const result = await session.run(
            `MATCH (n)<-[r]-(m) WHERE n.user_id = '${id}' RETURN COLLECT({recommender: m, relation: r}) AS recomms_details`
        );

        const data = result.records[0].get("recomms_details");

        if (data.length != 0) {
            msg = { message: "success", data: data };
        } else {
            msg = { message: "No recommendations found", data: data };
        }

        return msg;
    } catch (e) {
        console.log(e);
        return "No recommendations found";
    }
};
```

Figure 3.17: Code that implements the returns all recommendations made to a user

Chapter 4: EXPERIMENT, IMPLEMENTATION AND RESULTS

This chapter focuses on the trust inference experiment's methodology, observations, and data analysis. The section on findings focuses on assessing the data collected throughout the experiment. The outcomes of the experiment are assessed quantitatively, and a conclusion is drawn based on them.

4.1 Trust Metric Evaluation

To evaluate the success of the various Trust Metrics, we employ a machine learning evaluation technique known as leave-one-out [30]. We remove one trust statement from the trusted network and estimate it using the local trust measure for user A to user B. We then compute the prediction error by comparing the expected trust score to the original trust declaration. The Advogato dataset contains approximately 47,337 trust assertions, and the complete leave-one-out method must be reviewed for a random selection of 300 of them. Therefore, a time-efficient local trust metric was selected. This evaluation technique yields two measures: accuracy and coverage. The mistake created when forecasting a score is represented by accuracy. We employ Mean Absolute Error, which involves measuring the absolute amount of the difference between the actual and estimated scores and averaging the errors across all estimations. The ability of the algorithms to offer a forecast is referred to as coverage. In this situation, we compute the percentage of predictable trust declarations [30].

We created a graph network using the data from our survey of the Ashesi University class of 2022 and the Advogato dataset, see Figures 4.1 and 4.2, respectively.

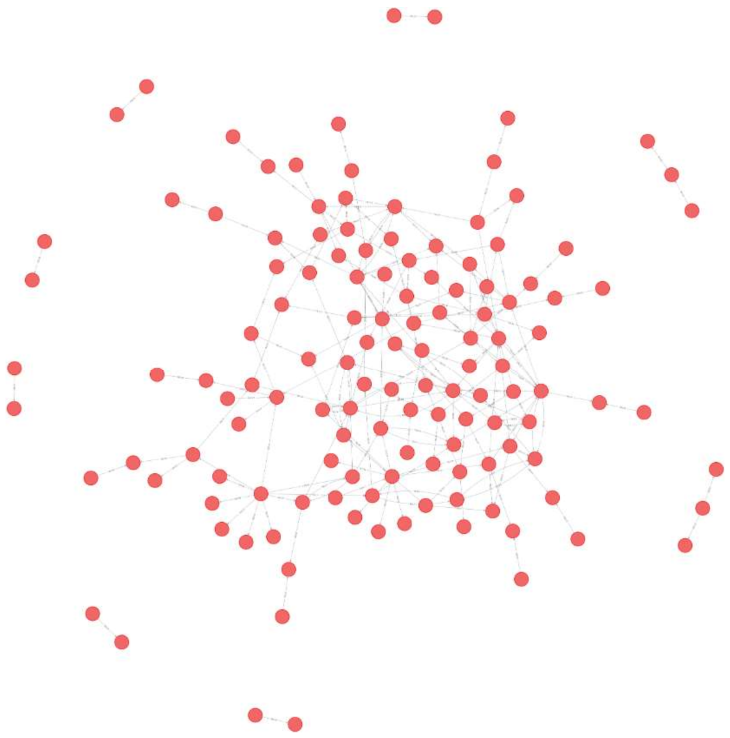


Figure 4.1: A network displaying survey data from students

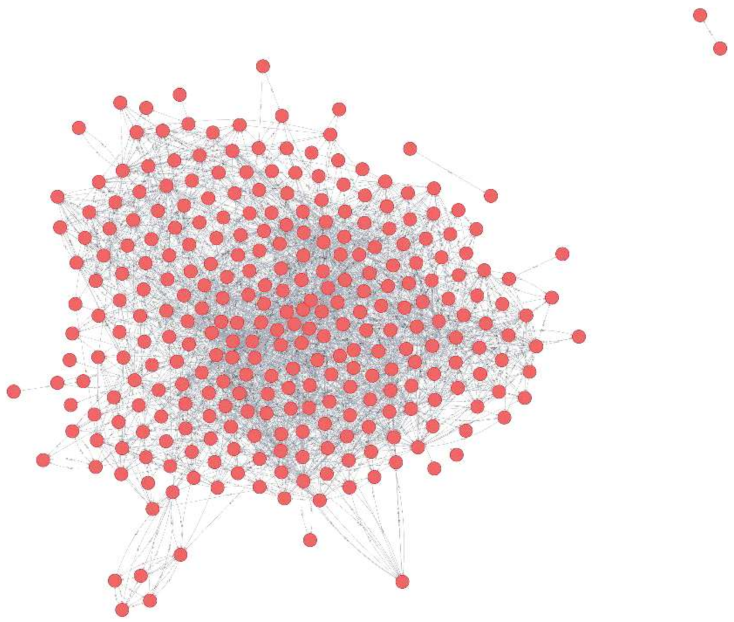
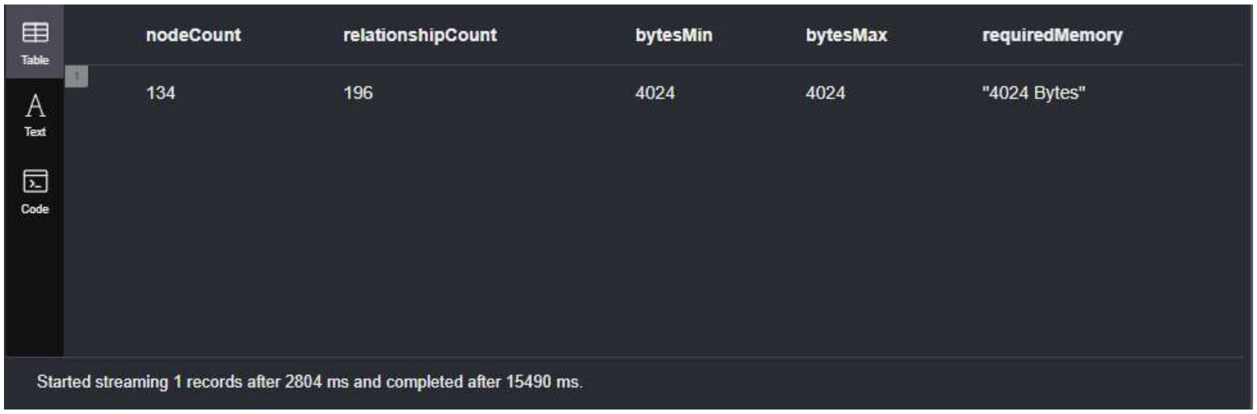


Figure 4.2: A network displaying part of the Advogato dataset's

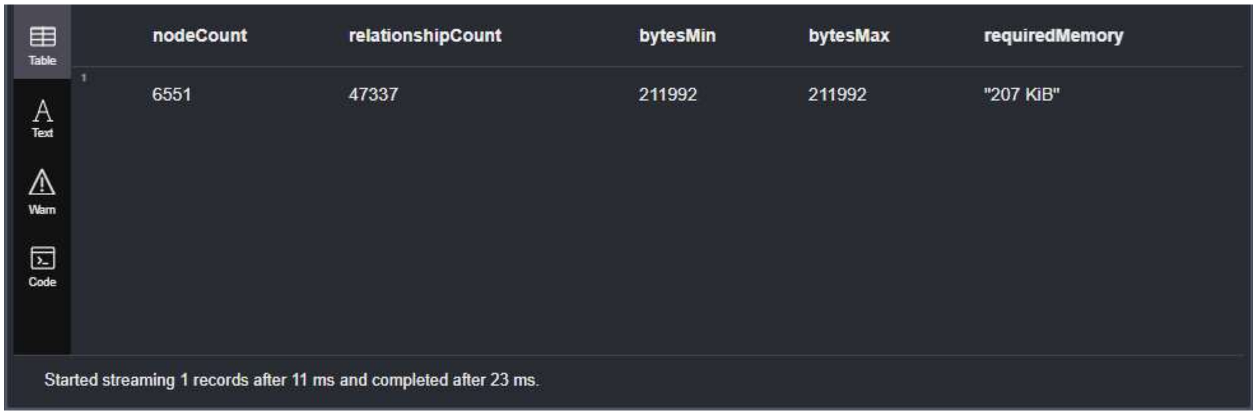
We begin by estimating the memory cost of running the global and local trust metrics algorithm (Figure 4.3 and Figure 4.4 respectively). We used eigenvector centrality, which assesses the transitive influence of nodes, to calculate global trust. Relationships formed by high-scoring nodes contribute more to a node's score than connections formed by low-scoring nodes. The approach computes the eigenvector associated with the highest absolute eigenvalue. The procedure uses the power iteration method to calculate the eigenvalue.



	nodeCount	relationshipCount	bytesMin	bytesMax	requiredMemory
1	134	196	4024	4024	"4024 Bytes"

Started streaming 1 records after 2804 ms and completed after 15490 ms.

Figure 4.3: Result of memory estimation on running the eigenvector centrality algorithm



	nodeCount	relationshipCount	bytesMin	bytesMax	requiredMemory
1	6551	47337	211992	211992	"207 KiB"

Started streaming 1 records after 11 ms and completed after 23 ms.

Fig 4.4: Memory estimation of running the Dijkstra Source-Target Shortest Path

```
CALL gds.eigenvector.write.estimate('myGraph', {
    WriteProperty: 'centrality',
    maxIterations: 20
})
YIELD nodeCount, relationshipCount, bytesMin, bytesMax,
requiredMemory
```

Code 4.1 Cypher query to estimate the cost of running the eigenvector centrality algorithm

```
CALL gds.eigenvector.stream ('myGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).user_id AS user_id, score
ORDER BY score DESC, user_id ASC
```

Code 4.2 Cypher query to compute the eigenvector score of the all the nodes

We estimated the trust value for the local trust metric using both the multiplicative method of trust propagation and the trust propagation function. We adjusted the numbers in the multiplicative strategy method by dividing the trust rating by 10, as seen below.

1, 2, and 8 are converted into 0.1, 0.2, and 0.8, respectively. Because the trust value should be $\exists 0 < trust\ value \leq 1$, this is used to evaluate the multiplicative method for the trust propagation algorithm. We calculated the trust value between two users using equation (3). The methodology discussed earlier was used for the trust propagation on the survey data as well.

4.2 Results and Analysis

To give a sufficient and fair analysis, we compared the trust propagation algorithm to the iterative multiplicative strategy [35] presented in our paper. We choose the most trusted and shortest path for trust calculation in the trust propagation algorithm, whereas in [35], the least trusted path is selected. This is the primary advantage of the trust propagation algorithm over Hasan and Brunei's iterative multiplicative strategy [35]. We know that a chain of highly trusted people is more reliable than a chain of low-trusted individuals. Furthermore, our results show a significant strong positive linear correlation of 0.36 (rounded up to two decimal places) between direct and propagated trust obtained by the trust propagation algorithm as shown in Table 4.2. Also, the Mean Absolute Error (MAE) for the trust propagation algorithm was 0.21 (rounded up to two decimal places) as shown in Table 4.1. In contrast, the iterative multiplicative strategy correlated 0.26 (rounded up to two decimal places) for the Advogato dataset [36] as shown in Table 4.2. Also, as seen in Figure 4.5. And we got an MAE of 0.48 (rounded up to two decimal places) as shown in Table 4.1.

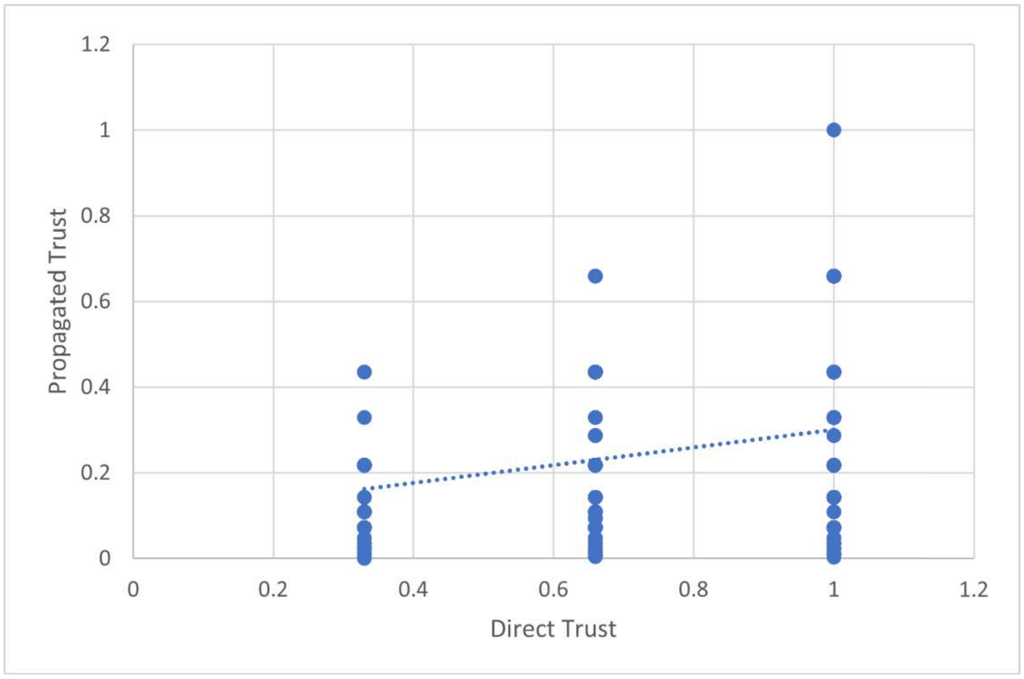


Figure 4.5: Correlation between Direct Trust and Propagated Trust

Table 4.1: Mean Absolute Error (MAE)

Dataset	Number of users	Focused Algorithm	Multiplication Strategy
Advogato	6,551	0.208141556	0.483397908

Table 4.2: Correlation between direct and propagated trust

Dataset	Number of users	Focused Algorithm	Multiplication Strategy
Advogato	6,551	0.361424776	0.263317734

Chapter 5: CONCLUSION, LIMITATION AND FUTURE WORK

5.1 Conclusion

In online social networks, recognizing trustworthy persons to form relationships is a top priority. Given the apparent rise in online social network interactions and information sharing, building, and assessing trust in a social network is critical for people's privacy, safety, and livelihood. "Trust" is vital in social network services to overcome these problems.

Implementations of three algorithms were also presented in this paper: the Trust propagation algorithm [36], the Multiplicative Strategy for Trust propagation algorithm [35], and the Eigen trust algorithm [29]. Finding the best and most trustworthy trust path in massive online social networks is usually challenging. The length of trust paths and the various measuring techniques used to decide how to integrate disparate information sources impact the accuracy of trust propagation estimates. For trust inference, this algorithm selects the most trusted and shortest path. To determine trust for the chosen path, the average of the trust ratings is weighted by the propagation distance. This method is contrasted with the multiplicative strategy proposed in [35].

By experimenting with the real and massive web of trust of the Advogato dataset, we observed a significantly strong positive linear correlation between direct trust and propagated trust acquired using the iterative multiplication technique.

We show that our strategy outperforms the other two alternatives using experimental results. The Advogato dataset was primarily utilized, containing about 47,337 trust links. In addition, student survey datasets were employed. The data was analyzed using statistical methods. Finally, the results of the experimental research were presented and interpreted.

This result provides confidence in establishing trust in an unknown user or entity by employing the aforementioned trust propagation approach. We may conclude that these algorithms yield meaningful results and can calculate trust between users and their network neighbors.

5.2 Limitations

Several constraints were encountered during the research process. The most significant limitation discovered during this investigation was a time constraint. As a result, we could not contribute to the identified trust metrics algorithm or develop our trust metric.

In addition, when employing the student survey dataset, the Trust propagation method [36] and the Multiplicative strategy algorithm [35] offered low or no accuracy. This is due to inefficiency in the network or fewer connections. These algorithms will become significantly more accurate as the user network grows. The Advogato [33] dataset did not fall under this category. Furthermore, network incompleteness does not affect the Eigen trust metric.

5.3 Future Work

This study provides a solid foundation for further research on the topic. Larger datasets may produce better results, and other trust contexts or trust algorithm combinations may be investigated. The Dijkstra algorithm is used to find the shortest path, which will only find one at a time; however, there may be more than one. We want to solve this in the future by leveraging the Neo4J graph database's "allShortestPaths" pathfinding methods and then averaging all the propagated trust values computed from each of the shortest pathways.

References

[1] Wenjun Jiang, GuojunWang, Md Zakirul Alam Bhuiyan, and Jie Wu. 2016. Understanding graph-based trust evaluation in online social networks: Methodologies and challenges. *ACM Comput. Surv.* 49, 1, Article 10 (May 2016), 35 pages.

DOI: http://dx.doi.org/10.1145/2906151

[2] D. Good. 1988. Individuals, interpersonal relations, and trust. In Trust: Making and Breaking Cooperative Relations, D. Gambetta (Ed.). Chapter 3 (1988), 31–48, New York: Basil Black.

[3] J. Golbeck. 2005. Computing and applying trust in web-based social networks. Ph.D. Dissertation, University of Maryland (2005)

[4] W. Sherchan, S. Nepal, and C. Paris. 2013. A survey of trust in social networks. Computer Surveys 45, 4 (2013), 47:1–47:33.

[5] A. Jøsang, R. Ismail, and C. Boyd. 2007. A survey of trust and reputation systems for online service provision. Decision Support Systems 43, 2 (2007), 618–644.

[6] C. Ziegler and G. Lausen. 2005. Propagation models for trust and distrust in social networks. Information Systems Frontiers 7, 4–5 (2005), 337–358.

[7] W. Jiang and G. Wang. 2011. SW Trust: Generating trusted graph for trust evaluation in online social networks. In Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (Trust Com). 320–327.

[8] Fletcher, E. (2020, October 21). *Scams starting on social media proliferate in early 2020*. Federal Trade Commission. Retrieved November 13, 2021, from https://www.ftc.gov/news-events/blogs/data-spotlight/2020/10/scams-starting-social-media-proliferate-early-2020.

[9] Statista Research Department. (2021, January 28). *Social Network Penetration Worldwide 2025*. Statista. Retrieved November 14, 2021, from https://www.statista.com/statistics/260811/social-network-penetration-worldwide/.

[10] Wanyu Lin, Zhaolin Gao, and Baochun Li. 2020. Guardian: Evaluating Trust in Online Social Networks with Graph Convolutional Networks. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 914–923. DOI:https://doi.org/10.1109/INFOCOM41043.2020.9155370

[11] Omar Hasan O, Brunie L, Pierson J Evaluation of the Iterative Multiplication Strategy for Trust Propagation in Pervasive Environments, ICPS"09 17, July 13-17, 2009, London, United Kingdom.

[12] Zhang, Y., Tang, L. S. -T., & Leung, L. (2011). Gratifications, collective self-esteem, online emotional openness, and trait like communication apprehension as predictors of Facebook uses. Cyberpsychology, Behavior, and Social Networking, 14(12), 733 – 739. doi: http://dx. doi.org/10.1089/cyber.2010.0042

[13] Hart, J., Ridley, C., Taher, F., Sas, C., & Dix, A. (2008). Exploring the Facebook experience: A new approach to usability. Paper presented at the 5th Nordic conference on Human– computer interaction: Building Bridges, Lund, Sweden

[14] Brass, D. J., Butterfield, K. D., & Skaggs, B. C. (1998). Relationships and unethical behavior: A social network perspective. The Academy of Management Review, 23(1), 14–31.

[15] Porter, L., Julia, O., Amalya, L., Zucker, L. G., & Brewer, M. B. (1995). Social networks, learning, and flexibility: Sourcing scientific knowledge in new biotechnology firms. SSRN. Retrieved from http://ssrn.com/abstract¼225384

[16] Boyd, D. M., & Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication, 13(1), 210–230.

[17] Singh, S. & BAWA, S. 2007. Privacy, trust, and policy-based authorization framework for services in distributed environments. Int. J. Comput. Sci. 2, 2, 85–92.

[18] Weir, G. R. S., Toolan, F., & Smeed, D. (2011). The threats of social networking: old wine in new bottles? Information Security Technical Report, 16(2), 38–43. doi: http://dx.doi.org/10. 1016/j.istr.2011.09.008

[19] Lai, L. S. L., & Turban, E. (2008). Groups formation and operations in the Web 2.0 environment and social networks. Group Decision and Negotiation, 17(5), 387–402. doi: http://dx.doi. org/10.1007/s10726-008-9113-2

[20] Brandtzaeg, P. B., Luders, M., & Skjetne, J. H. (2010). Too many Facebook "friends"? content sharing and sociability versus the need for privacy in social network sites. International Journal of Human–Computer Interaction, 26(11–12), 1006–1030. doi: http://dx.doi.org/10. 1080/10447318.2010.516719

[21] Haydar, C. & Roussanaly, A. & Boyer, A. (2013). Local Trust Versus Global Trust Networks in Subjective Logic. Proceedings - 2013 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2013. 1. 29-36. 10.1109/WI-IAT.2013.5.

[22] Chakraborty, P. S., & Karform, S. (2012). Designing Trust propagation algorithms based on simple multiplicative strategy for social networks. *Procedia Technology*, *6*, 534–539. https://doi.org/10.1016/j.protcy.2012.10.064

[23] Haydar, C., Boyer A., and Azim R.. Hybridising collaborative filtering and trust-aware recommender systems. In 8th International Conference on Web Information Systems and Technologies -WEBIST2012. Porto, Portugal, 2012. http://hal.inria.fr/hal-00679233.

[24] Rousseau, D. M., et al. Not so different after all: A cross-discipline view of trust. Academy Manag. Rev. 23, 3 (1998), 393- 404.

[25] Demouchel, P., Trust as an action. Euro. J. Sociol. 46, (2005), 417-428.

[26] Mui, L., Computational models of trust and reputation: Agents, evolutionary games, and social networks. Ph.D. thesis. 2003

[27] Sherchan, W., Nepal, S. and Paris, C. 2013. A Survey of trust in social networks. ACM Comput. Surv. 45, 4, Article 47 (August 2013)

[28] P. Massa, P. Avesani, and R. Tiella. 2005. A trust-enhanced recommender system application: Moleskiing. In Proceedings of the 2005 ACM Symposium on Applied Computing (SAC). 1589–1593.

[29] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, “The eigentrust algorithm for reputation management in p2p networks,” in WWW, 2003, pp. 640–651

[30] Massa, Paolo & Avesani, Paolo. (2007). Trust metrics on controversial users: Balancing between tyranny of the majority and echo chambers. International Journal on Semantic Web and Information Systems - IJSWIS. 3.

[31] Menzies, F. (2016, April). Why Consumer-to-Consumer Communication Wins. (Forbes) Retrieved January 2019 from Forbes: https://www.forbes.com/sites/gyro/2012/04/26/why-consumer-to consumercommunication-wins/#5484a9794b17

[32] Dignan, L. (2019, February). Fixing Facebook: Why it Needs a Trust Score. From Zdnet: https://www.zdnet.com/google-amp/article/fixing-facebook-why-it-needs-a-trustscore/?__twitter_impression=true

[33] Rossi, R. A., & Ahmed, N. K. (2015). *The Network Data Repository with Interactive Graph Analytics and Visualization*. Advogato. Retrieved from https://networkrepository.com

[34] M. Richardson and R. Agrawal and P. Domingos. Trust Management for the Semantic Web. ISWC, 2003.

[35] Hasan, Omar & Brunie, Lionel & Pierson, Jean-Marc. (2009). Evaluation of the Iterative Multiplication Strategy for Trust Propagation in Pervasive Environments. 10.1145/1568199.1568207.

[36] Bhattacharya, Munmun and Nesa, Nashreen. (2016). An Algorithm for Predicting Local Trust based on Trust Propagation in Online Social Networks. International Journal of Computer Applications. 156. 8-15. 10.5120/ijca2016912461.

[37] *Dijkstra source-target shortest path - neo4j graph data science*. Neo4j Graph Data Platform. (n.d.). Retrieved April 21, 2022, from https://neo4j.com/docs/graph-data-science/current/algorithms/dijkstra-source-target/

APPENDICES

Appendix A: SOCIAL TRUST SURVEY QUESTIONNAIRES

1. Name of trustor (Your name)?

   …………………………………………………………………….

2. Your phone number?

   ……………………………………………………………………

3. Name of trustee in your year group?

   …………………………………………………………………….

4. Phone number of trustee?

   …………………………………………………………………….

5. On a scale of 1-10 how well do you trust this person?

	1	2	3	4	5	6	7	8	9	10	
Strongly distrust	○	○	○	○	○	○	○	○	○	○	Strongly trust

6. How long have you known this person?

   ○ 0-6 months

   ○ 1-2 years

   ○ 3-4 years

   ○ 5 years and above