PeerMe: Enhancing Collaboration for Software Engineers through Matchmaking and Proximity Web Application

Nayaab Azim
Computer Science
Virginia Tech
Blacksburg VA USA
nayaab@vt.edu

Sadath Ullah Khan Mohammed
Computer Science
Virginia Tech
Blacksburg VA USA
msadath@vt.edu

Evan Phaup
Computer Science
Virginia Tech
Blacksburg VA USA
phaupe22@vt.edu

Adeyemi Aina
Computer Science
Virginia Tech
Blacksburg VA USA
ainababs0@vt.edu

ABSTRACT

In recent years, the field of software engineering has experienced a considerable increase in demand for competent experts, resulting in an increased demand for platforms that connect software engineers and facilitate collaboration. In response to this necessity. This paper presents a project proposal to solve the lack of a proper one-stop connection platform for software engineers and promote collaborative learning and upskilling. The necessity of the project along with the existing research and similar works in this field has been revealed. The idea of the project is to develop a web-based application (PeerMe) that would facilitate connecting and collaborating between software engineers. The application would perform algorithmic matching to suggest user connections based on their technical profiles. The users can filter profiles, discover open projects and form collaboration groups. The high-level architecture of the application and design sketch has been discussed in the later sections.

CCS CONCEPTS

• Computing methodologies • Software and its engineering

KEYWORDS

Matching algorithms, collaborative software engineering, user preferences, proximity-based social applications

1. INTRODUCTION

With rapid changes in software technology in the 21st century, Software Engineers have the task of constantly upgrading their skillset. Learning new technologies can be daunting however groups and community are a great offset. Recent studies have shown multiple benefits of collaborative learning among software engineers such as organized, time-effective learning and guidance among others (Layman, 2006). Although Software Engineers meet their peers regularly, most of them are unaware of each other's skill sets. Thus, when an engineer experiences difficulty in a project or a particular technology niche, he is left wondering.

The proposed solution PeerMe web application mitigates the above-mentioned challenges, this is a connection platform specifically designed for software developers. The users would be able to connect with other software engineers in the community. Once the user sets up his/her profile and technology interests, the application would then suggest connections by using matching algorithms on parameters such as project interests, proximity, learning outcome, and experience levels. Moreover, the users can also find and view the technical profiles of other users by applying desired filters. Discord, Stack Overflow, and other platforms exist to facilitate collaboration among software engineers. These platforms, however, cannot match users based on proximity and skill set, which PeerMe aims to provide.

1.1 Matching Algorithm

Matching algorithms are a critical component of the proposed Software Engineers' connection platform; based on their technical profiles, project interests, and other criteria such as experience levels and learning outcomes, these algorithms will allow the program to identify suitable connections for Software Engineers. The use of matching algorithms will ease connection for Software Engineers and increase the chances of successful collaboration by suggesting relevant and compatible connections. Therefore, the design and implementation of effective matching algorithms will be a key focus

1.2 Proximity preference

Proximity preference is an innovative approach that connects users with other software engineers in close proximity. This allows users to form collaborative groups with other professionals in their area, making in-person meetups and skill-sharing sessions easier to organize. The PeerMe web application uses the proximity feature to encourage software engineers to take advantage of their local network and strengthen connections with like-minded professionals within proximity

2. RELATED WORK

Matching social applications have gained interest in software development collaboration, and several related studies have explored various aspects of this topic. Liu et al. (2009) provided an overview of algorithms for rule generation and matchmaking in context-aware systems. The paper discusses various techniques for matching users with relevant content or services based on contextual information, using rule-based and machine-learning algorithms. The authors also explored techniques for combining these two approaches to create more efficient systems with better accuracy rates.

Wang and Li (n.d.) proposed an approach based on Hebbian Learning Law that can automatically group distributed e-learners

with similar interests, enhancing collaborative learning among them. This algorithm avoids the need for difficult design work required for user preference representation or user similarity calculation, making it suitable for open and distributed environments like e-learning platforms. Experimental results show that this algorithm has preferable prediction accuracy as well as improved scalability and satisfaction levels compared to existing approaches in the field of e-learning social network exploiting approach.

Other related studies have examined specific features or use cases of proximity-based social applications for software development collaboration. For example, Ali et al. (2017) proposed a dynamic self-organizing community or group mechanism with an improved matchmaking algorithm. The system can group similar agents according to their preferences and capabilities, and the authors discuss how to represent, evaluate, exchange, and matchmake during the self-organizing process. A prototype was built for verification purposes, and experiments based on real learner data show that this mechanism can organize learners properly while improving the speed and efficiency of searches for peer students owning relevant knowledge resources.

3. RESEARCH GOALS AND OBJECTIVES

This application would enhance collaborative learning and help build a community of software engineers. It is a 2-way solution: people looking to join a project can find it, on the other hand, open projects looking to add new members could do so too. Users could also discuss doubts, logic, and process flow by utilizing the text messaging feature. This would result in time-efficient work rather than a single developer being stuck on an issue. Furthermore, it would develop a sense of team spirit among the users, which is one of the key attributes to becoming a great software engineer [2](Li et al., 2020).

4. ARCHITECTURAL DESIGN

The design architecture of our site using the MERN stack typically involves a client-server architecture, with the client being built using React and the server is built using Node.js and Express. Here is a high-level overview of the different components and layers of the architecture:

1. **Client Layer** - This layer is responsible for rendering the user interface and handling user interactions. It is built using React, which allows for the creation of reusable components and efficient rendering of UI elements.
2. **API Layer** - This layer serves as an interface between the client layer and the server layer, providing a set of RESTful endpoints for the client to interact with. The AP layer is built using Express and is responsible for handling incoming requests from the client and sending back appropriate responses.
3. **Server Layer -** This layer is responsible for processing incoming requests from the API layer and interacting with the database layer to retrieve or store data. The server layer is built using Node.js and Express, and it will incorporate middleware for handling authentication, logging, error handling, and other features.
4. **Database layer -** This layer is responsible for storing and managing site data. The database will include tables for storing user profiles, chat messages, and other user-generated data. The database layer is built using MongoDB, which is a popular NoSQL database that works well with Node.js and Express.
5. **Hosting -** This layer is responsible for hosting the site on a service like Amazon Web Services (AWS) and manages the various components of the site.
6. **State Management -** To manage the state of the client layer and ensure that data is efficiently passed between different components and updates are handled consistently, we will use a tool called Redux.
7. **Security -** Implementing a third-party authentication service called Firebase to handle authentication and prevent unauthorized access or malicious attacks
8. **Message Queue -** Using a message queue or real-time communication tool like socket.io to enable real-time messaging and notifications between users.
9. **Location API -** Using a service like Google Maps API or OpenStreetMap to add the location-based search functionality.
10. **Testing -** Using a testing framework like Jest to test the site and ensure that it's working as expected.
11. **Logging -** Using logging and monitoring tools to track site performance, identify issues, and make improvements over time.

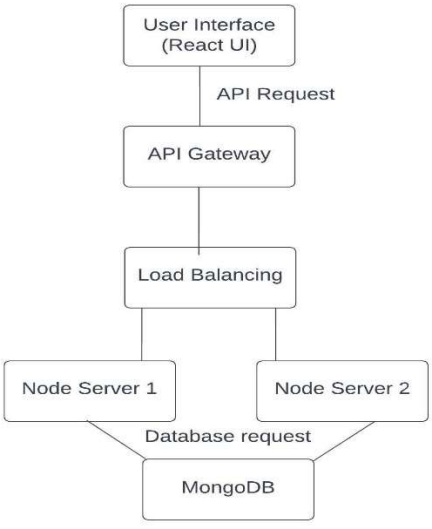

Figure 1: High-Level Architecture Flow Diagram

5. DESIGN SKETCH

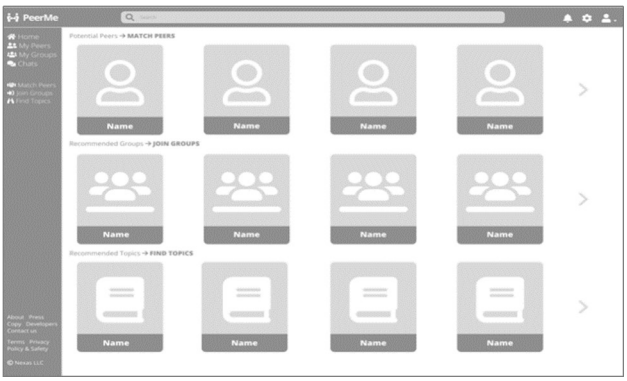

Figure 2: User's landing page with preferences to match peers, join groups find personalized topics

Figure 3: User's personalized settings menu, showing matching radius for proximity preference

REFERENCES

1. Liu, D., Meng, X. W., Chen, J. L., & Xia, Y. M. (2009). Algorithms for rule generation and matchmaking in context-aware system. Ruan Jian Xue Bao/Journal of Software, 20(10), 2655–2666. https://doi.org/10.3724/SP.J.1001.2009.03436

2. Shen, R., Yang, F., & Han, P. (2003). LNAI 2903 - A Dynamic Self-Organizing E-Learner Communities with Improved Multi-agent Matchmaking Algorithm. Springer-Verlag.

3. Wang, Z.-M., & Li, L.-N. (n.d.). Enable Collaborative Learning: An Improved E-Learning Social Network Exploiting Approach.

4. Layman, L. (2006). Changing students' perceptions: An analysis of the supplementary benefits of collaborative software development. *Software Engineering Education Conference, Proceedings*, *2006*, 159–166. https://doi.org/10.1109/CSEET.2006.10

5. Li, P. L., Ko, A. J., & Begel, A. (2020). What distinguishes great software engineers? *Empirical Software Engineering*, *25*(1), 322–352. https://doi.org/10.1007/s10664-019-09773-y