MentorMate - End of Term Report

Team 20230798

Supervisor: Professor Hamid Timorbadi

Team Members: Doaa Muhammad, Samreen Khatib Syed, Sanjana Dasadia, Sweni Shah

Due: Dec 6, 2023

1.0 Executive Summary

This report provides a comprehensive overview of MentorMate, a mentor-mentee matching algorithm, housed in a web application. The report commences with an in-depth examination of the project's current status and notable changes. Following this, we explore various potential solutions and design alternatives concerning our algorithmic choices and web design frameworks. Subsequently, the report delves into the technical design and implementation of MentorMate, offering both a high-level overview, and detailed insights into specific components such as the matching system, login system, authentication server, questionnaire server, and algorithm server, ensuring a better understanding of the project's technical intricacies. The report also outlines our strategic approach for project management, specifying task ownership distribution and milestone planning until the end of the next semester. This structured framework ensures a transparent and organized workflow throughout the project's lifecycle. Lastly, a risk assessment was conducted to identify and address potential challenges that may arise during the project's progression.

The primary adjustment in our project involves a shift in the algorithmic approach. Initially contemplating algorithms such as Hungarian, Gale-Shapley, and Hopcroft-Karp, our team ultimately opted for K-means clustering. The rationale behind this decision, including a detailed justification, is thoroughly discussed in the report, shedding light on the strategic considerations that led to this change in our project's algorithmic framework.

2.0 Project Status, Report, and Changes

In reviewing the project's current status, report structure, and changes, various components are at different stages of implementation, with ongoing efforts addressing functionalities, backend enhancements, and proactive measures to streamline component integration and ensure compatibility across React versions.

The login frontend and backend being partially implemented, awaiting the addition of mentor/mentee selection during registration and the creation of corresponding database objects. The home page is fully implemented, with minor design adjustments to be made. The profile page is partially implemented, pending the display of accurate user information, the addition of details such as matches, and the inclusion of the ability for users to edit their profiles. Additionally, there is a fully implemented role selection page that needs to be removed. In the revised process, users will directly choose their roles during registration, eliminating the need for a separate step to select a role before being redirected to the appropriate questionnaire. In the backend, there is a need to enhance session persistence capabilities to

retain user information, encompassing questionnaire responses and diverse profile customizations made by the user across previously terminated sessions. Presently, the backend solely stores login credentials, and there is a requirement to extend the session persistence functionality to capture and preserve a more extensive range of user-specific data beyond just login details. A functioning draft of the algorithm has been implemented and tested and is undergoing fine-tuning and improvements through further testing. Ongoing work includes the incorporation of corner cases like the factorization of registration time for user prioritization in matching. Additionally, there is a planned implementation to allow users to discover new matches in case they are dissatisfied with their initial pairing. The primary obstacle encountered by our group centered around integrating various versions of React and other frameworks, compounded by React's lack of backward compatibility. When combining our individually developed components, we faced significant hurdles, with the merging process proving less than smooth due to compatibility issues. To overcome this challenge, we proactively engaged in aligning React versions across all components. This effort aimed to resolve compatibility issues and ensure the cohesiveness and functionality of the overall project. Regular communication and collaborative initiatives among team members were key in promptly identifying and addressing any version discrepancies that arose during this alignment process. The team also established a streamlined process for updating and maintaining consistent React versions across all components, ensuring that future developments would not encounter similar integration hurdles. This proactive approach significantly contributed to a more seamless merging process and laid the foundation for ongoing compatibility as the project progresses.

In terms of changes, our initial proposal explored various matching algorithms such as Hungarian, Gale-Shapley, and Hopcroft-Karp across different scenarios. However, our implemented solution took a different trajectory, choosing a modified K-means algorithm based on Euclidean distance. Further insights into this decision will be elaborated upon in Section 3.0.

3.0 Possible Solutions and Design Alternatives

We explored various methods to tackle the matching algorithm. Initially, we explored using algorithms like Hungarian, Gale-Shapley, and Hopcroft-Karp for various scenarios such as solving assignment problems, stable matching, and bipartite graph matching, respectively. However, these powerful algorithms were considered suboptimal for our unique matching problem. The Hungarian algorithm, designed for job assignment to minimize costs, fell short in capturing complex mentor-mentee relationships. Gale-Shapley, tailored for stable matching with an equal number of individuals in different groups, didn't adapt well to our dynamic dataset. Lastly, Hopcroft-Karp, proficient in bipartite graph matching, didn't align with the intricate, non-linear relationships in our questionnaire responses. We also

briefly considered using basic web technologies such as HTML, CSS, and JavaScript exclusively. While this approach was straightforward and familiar to us, it lacked the advanced features and efficient state management provided by modern frameworks, such as React.

For our implementation, we found that the adapted K-means algorithm demonstrated strong performance, particularly in handling non-spherical clusters and capturing complex, nonlinear patterns of mentor-mentee relationships. The Euclidean distance metric employed in our modified K-means approach effectively quantified dissimilarity, addressing the nuanced nature of our matching problem. This approach emerged as the most suitable for us, surpassing the limitations we encountered with traditional matching algorithms. Additionally, we decided to leverage React due to its advanced features, efficient state management, and the valuable learning experience it offered. The use of React not only provided us with a robust framework but also opened doors to a robust and modern framework and component reusability, making it the preferred choice for our project.

4.0 Technical Design and Implementation

4.1 System Level Overview

MentorMate employs a dynamic and responsive frontend that communicates with servers housed in our backend (see Figure 1 for a high level system diagram). Microservices in our backend include user management, authentication, questionnaire/data storage, and the algorithm used to identify pairings. Our website can only be accessed by registered users, and hence some security measures include storing user information securely in MongoDB Cloud is essential. Some front end components including our login system, matching process webpages, and profile information webpages are developed using React for a responsive user experience. This technical framework aims for a secure, scalable, and user-friendly mentorship platform.

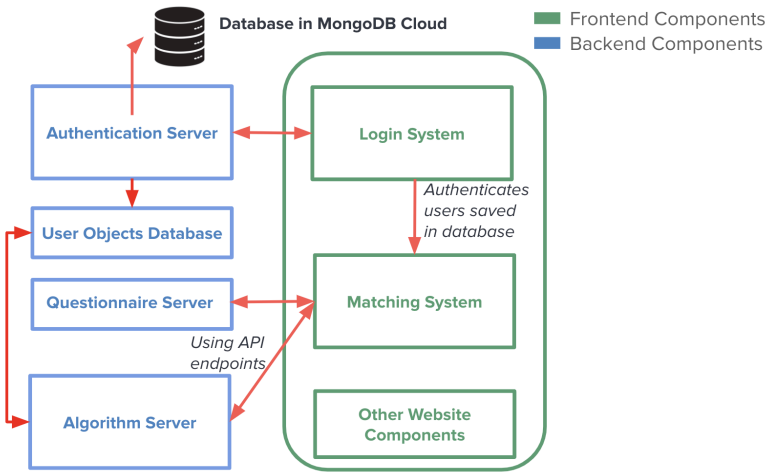

Figure 1: Shows high-level system diagram of MentorMate

4.2 Module Level Description

For each module in Figure 1, this section lists the module's functionality, inputs, outputs, and related specification from Table 1.

4.2.1 Login System

- **Functionality:** The website features two frontend web pages, Register and Login, developed using React, enabling users to access the platform.
- **Inputs:** There are user input such as email, username, or password in specified fields.
- **Outputs:** During the login process, user information is transmitted to the authentication server. Upon successful verification, the user is redirected to the website homepage. In the case of registration, user details are sent to the authentication server for storage, and subsequently, users are redirected to the login page to re-enter their credentials.
- **Specifications:** The website should allow users to register themselves and login

4.2.2 Authentication Server

- **Functionality:** The authentication of login requests and endpoints involves communication with MongoDB Cloud, which serves as the storage repository for all user information. This functionality is implemented using Passport, a Node.js authentication middleware.
- **Inputs:** User information like email, username, or password from *Login System* described in Section 4.2.1
- **Outputs:** Upon user authentication, a signal is sent back to the frontend *Login System*.

Additionally, in the event of a new user registration, a corresponding User object is created, accessible within the *User Objects Database* in Section 4.2.5 for algorithmic use.

- **Specifications:** The website should allow users to register themselves and login

4.2.3 Questionnaire Server

- **Functionality:** Stores questions for mentors and mentees. Uses JSON for question formatting and Express which allows us to create robust endpoints to communicate with the *Matching System* in Section 4.2.4.
- **Inputs:** Incoming request from *Matching System* to retrieve questions
- **Outputs:** Send questions list based on role using API endpoints to *Matching System*
- **Specifications:** The website should allow users to answer a questionnaire

4.2.4 Matching System

- **Functionality:** The frontend, which includes mentor/mentee role selection and questionnaire webpages, has been implemented using React.
- **Inputs:** The questions are obtained from *Questionnaire Server* in Section 4.2.3 using API endpoints. These questions are then displayed on the front end where the user inputs their role, and answers.
- **Outputs:** The answers and user roles are transmitted to the algorithm server through API endpoints.
- **Specifications:** The website should allow users to answer a questionnaire.

4.2.5 Algorithm Server

- **Functionality:** The system houses a K-means algorithm responsible for facilitating matching between User-mentor and mentee objects. The algorithm's code is implemented in Python and utilizes the Flask library to establish an API endpoint within the *Matching System* in Section 4.2.4.
- **Inputs:** Users and their answers are obtained from the *Matching System* using API endpoints. The algorithm is then run on unassigned users with their answers to create matches. A modification we want to make is to retrieve the user from the *User Objects Database* instead.
- **Outputs:** Match results for the user are sent back to the *Matching System* to display on the front end. This part of the module has yet to be implemented.

- **Specifications:** Website should accurately match users with a mentor based on interests selected and algorithm should be fast.

5.0 Design Specifications and Test Plan

Table 1: Outlines specifications of design and methods for testing

Specification/Measure of success	Testing
1. Website should accurately match users with a mentor based on interests selected and algorithm should be fast.	Manually test with multiple profiles, verifying that main interests match Stress test the algorithm through unit tests where we will have <1000 users, 1000 users, and beyond 1000 users to record and measure algorithm performance time.
2. The website should allow users to register themselves and login	Unit test the authentication process and also manually test
3. The website should have a way to navigate across pages to display matches - navigation bar or search bar and must be easy to navigate/read font (users should be able to spot specific features (eg: nav bar) in less than 5 seconds)	Perform manual testing on website and record number of seconds it takes to locate features
4. Handle high traffic on website and avoid crashing	Can perform load testing on website to continuously access the website 500 times during the day. The website should not crash in this case for optimal performance
5. The website should allow users to edit their profile information at any time	Register as a user on the website and test manually
6. The website should have mechanisms for error handling in the case of empty fields or wrong login information	Unit test with empty fields and manually test on website to see if front end components (error messages are displayed and returned correctly)
7. The website should allow users to answer a questionnaire	Manually tested by selecting single, multiple, and no options on webpage

In order to thoroughly evaluate the design, the specifications outlined in Table 1 must be referred to as key indicators of success. The design requires users to register, complete the questionnaire to specify their roles and interact with the site. Consequently, testing relies heavily on real users to simulate use of the final design.

Each component of the design can be tested through a diverse group of test users from various backgrounds. This group, consisting of 10 users, will require 3 users to fulfill the role of mentors and the remaining as mentees. This way, comprehensive manual testing of the user registration and questionnaire submission can be conducted. Given the varying levels of technological proficiency among users and their unfamiliarity with the design, error messages and incorrect input behaviors can be tested, as specification #6 identifies. Specification #2 will pass as users successfully register and log in, with profile information displayed accurately. As the users interact with the site, visibility and usability can be tested. Testing of specification #3 would be monitored through a survey which the user's will fill in as they navigate through the site. The survey will include a checklist of the various site features including navbar, questionnaire, log in/out button and site pages with a time constraint of 5 seconds for users to locate these elements. All users must be able to locate them within this constraint for it to pass. This will lead users to the questionnaire, where further error handling is tested. Users will be instructed on questionnaire responses so the algorithm match results can be verified manually. The generated matches will be compared with the expected result to determine the success of specification #1, passing if all match. Apart from manual user testing, unit tests will be useful in testing the security and load testing for the traffic control, monitoring any crashes. In conclusion, this detailed testing plan assures the verification of specifications for a successful and user-friendly design.

6.0 Project Management

As we progress through the project until the end of April, our focused project management plan, described in Table 2, presents a clear task ownership, key milestones, and dependencies between tasks, ensuring a streamlined and efficient execution. This strategic framework serves as our roadmap, guiding the team towards successful project completion.

Table 2: Project Management Plan highlighting tasks and due dates

Week	Tasks and Task Ownership	Specific Due Dates for Milestones
Dec 7 - Jan 15	- End to End rigorous testing (Sweni), bug fixing (Sanjana) - Work on submission for publication (All) - Improve logo, fonts - UI (Doaa) - Display more info on profile page (i.e email, name, institution, matches) - Doaa - Match result emails (Sweni) - Ask users to enroll as a mentor or mentee when they register and eliminate “role selection” leading to the questionnaire (Samreen) - Updates to algorithm, corner cases (Sanjana) - Implementation of session persistence (Samreen)	- Begin publication paper process (Dec 7) - Improve UI (Dec 29) - Testing (Dec 30) - UI fixes and profile page (Jan 15) - Match emails (Jan 15) - Eliminate role selection (Jan 15) - Session persistence (Jan 15)
Jan 15 - Jan 28	- Attend meeting #4 - Documentation (Doaa) and verification (Samreen) of match results - Create User Testing Feedback Form (Sanjana) - Find users to test the application (Sweni)	- Meeting #4 (date TBD) - Documentation and verification of match results (January 26) - Complete User Testing (January 28)
Jan 29 - Feb 16	- Attend meeting #5 (All) - Split workload for presentation practice session, create slides (Feb 5) - Improve UI components (All) - Evaluate User Match Satisfaction (Sweni) and UI experience from testing results (Doaa) - Implement improvements for algorithm (Samreen) and UI improvements based on testing results (Sanjana)	- Meeting #5 (date TBD) - Create presentation (Feb 5) - Compile and evaluate testing results (Feb 9) - Additional UI Improvements (Feb 16) - Implement Improvements from testing result evaluation (Feb 16)
Feb 17 - Mar 8	- Practice for presentation (All) - Deliver presentation (All) - Split final report workload (March 5th) - Finishing touches for backend (Doaa) - Finishing touches for frontend (Sweni)	- Practice presentation (Feb 17 - Feb 18) - Website Completed - frontend & backend (Mar 8)
Mar 9 - Mar 22	- Modify presentation based on practice presentation feedback (March 9) - Deliver graded presentation	- Graded Presentation (date TBD)
Mar 23 - Apr 6	- Work on final report - Minor bug fixing - site accessibility (All)	- Final report due (April 1-5) - Test site accessibility (March 23 - April 4)
Apr 7 - Apr 12	- Finalize and complete poster (All) - Practice presentation for design fair (All)	- Fair poster (April 5) - Design Fair Presentation dry runs (Apr 4-7) - Design Fair (Week of Apr 8)

7.0 Risk Assessment

There are a few risks that can be identified which could hinder the completion of the project. One of these risks involves performance issues due to uneven distribution of user traffic. The presence of a high user load can affect the algorithm performance as it can become computationally expensive and determining matches can become slow. The site not being able to handle a high user load will cause slow response times, affecting the user experience. Situations where most users continuously spend most time on a certain page of the site, such as the questionnaire or profile page can lead to this. The surge in user activity can degrade the database performance as well. In order to mitigate this risk, load balancing strategies can be implemented to distribute user traffic evenly across a variety of servers and ensure optimal performance [1]. Through the use of multiple servers, one server is not overloaded and adds scalability. These additional servers introduce horizontal scalability.

Another risk that can be faced when developing the project involves the differing versions of React. With technologies constantly updating, newer React versions can introduce compatibility issues and learning curves to adapt for developers. When updating to a newer version of React with significant upgrades, major changes in the code could be required which would hinder the completion. Team members would require time to familiarize with the new features and this would additionally slow down the project development. In order to mitigate this, team members can be educated and trained on the version of React for the project and collaborate with front-end developers to discuss doubts. Adequate time can be allocated for this before the start of the winter semester. Additionally, ensuring that all libraries/tools used are compatible with the project's version of React and scheduling an upgrade if needed in the break avoids unforeseen circumstances and delays. Added testing before and after any necessary upgrades will ensure risks to the project's development are eliminated.

8.0 References

[1] N. Solutions, “Load balancing strategies in Distributed Systems,” LinkedIn, https://www.linkedin.com/pulse/load-balancing-strategies-distributed-systems-netopia-solutions-vieye/ (accessed Dec. 8, 2023).