

Senior Design Server/Client Development for Project Matching [Phase 3]

Design Document

Team 02

Clients	Jacob Grundmeier and Akhilesh Tyagi
Advisor	Akhilesh Tyagi
Database Design	Noah Nelson
Frontend Design	Joshua Izumba Noah Nelson Evan Brummer
Algorithm Design	Robert Holeman Devin Tigges Max Kueller
Email	sdmay24-02@iastate.edu
Website	sdmay24-02.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

- **React** functional programming and component design.
- Basic branching and merging standards for **Git**.
- **Agile** software development practices.
- Weekly **stand-up meetings** to discuss progress and identify blockers.

Summary of Requirements

- The **matching algorithm** must produce an accurate matching based on several factors, including student preferences, client's desired majors and number of team members.
- **Frontend web UI** must be user friendly, intuitive, and easy to use.
- **Backend** must be able to authenticate the user and correctly respond to requests.

Applicable Courses from Iowa State University Curriculum

- **COM S 311** – Introduction to the Design and Analysis of Algorithms.
- **COM S 319** – Construction of User Interfaces.
- **COM S 309** – Software Development Practices.
- **S E 317** – Intro to Software Testing.

New Skills/Knowledge acquired that wasn't taught in courses

- **React** framework functional programming and component design.
- **React** Router-Dom navigation.
- **Laravel Eloquent** database interaction.

Table of Contents

1 Team

- 1.1 Team Members
- 1.2 Required Skill Sets for Your Project
- 1.3 Skill Sets covered by the Team
- 1.4 Project Management Style Adopted by the team
- 1.5 Initial Project Management Roles
- 1.6 Problem Statement
- 1.7 Requirements & Constraints
- 1.8 Engineering Standards
- 1.9 Intended Users and Uses

2 Project Plan

- 2.1 Task Decomposition
- 2.2 Project Management/Tracking Procedures
- 2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria
- 2.4 Project Timeline/Schedule
- 2.5 Risks And Risk Management/Mitigation
- 2.6 Personnel Effort Requirements
- 2.7 Other Resource Requirements

4 Design

- 4.1 Design Content
- 4.2 Design Complexity
- 4.3 Modern Engineering Tools
- 4.4 Design Content
- 4.5 Prior Work/Solutions
- 4.6 Design Decisions
- 4.7 Proposed Design
 - 4.7.1 Design 0 (Initial Design)
 - 4.7.2 Design 1 (Design Iteration)
- 4.8 Technology Considerations
- 4.9 Design Analysis

5 Testing

- 5.1 Unit Testing
- 5.2 Interface Testing

- 5.3 Integration Testing
- 5.4 System Testing
- 5.5 Regression Testing
- 5.6 Acceptance Testing
- 6 Implementation
- 7 Professionalism
 - 7.1 Areas of Responsibility
 - 7.2 Project Specific Professional Responsibility Areas
 - 7.3 Most Applicable Professional Responsibility Area
- 8 Closing Material
 - 8.1 Discussion
 - 8.2 Conclusion
 - 8.3 References
 - 8.4 Appendices
 - 8.4.1 Team Contract

1 Team

1.1 Team Members

Devin Tigges
Max Kueller

Evan Brummer
Noah Nelson

Joshua Izumba
Robert Holeman

1.2 Required Skill Sets for Your Project

- Frontend development.
- Backend development.
- Database development.
- CI/CD knowledge.
- Project management skills.
- Client interaction skills.
- Teamwork skills.
- Agile experience.
- Functional/performance Testing.

1.3 Skill Sets covered by the Team

Frontend Development

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer, Robert Holeman.

Backend Development

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Robert Holeman.

Database Development

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer, Robert Holeman.

CICD Knowledge

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer, Robert Holeman.

Project Management Skills

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer, Robert Holeman.

Client Interaction Skills

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer.

Teamwork Skills

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer.

Agile Experience

Max Kueller, Noah Nelson, Joshua Izumba, Devin Tigges, Evan Brummer, Robert Holeman.

1.4 Project Management Style Adopted by the team

Previous phases preferred **Agile** project management because it accommodates:

- Short-term deadlines.
- The ability to incorporate changes at any time into the project.
- Take stakeholders' feedback into account throughout the project.
- Potential to overlap work between teammates.

1.5 Initial Project Management Roles

Noah	Database Lead
Devin	Team Organization
Max	Client Interaction, Testing Lead
Joshua	Testing, Quality Assurance
Evan	UI/UX Lead
Robert	Architecture design

1.6 Problem Statement

The goal of this project is to **create a program that assists the Senior Design administrators in matching students with relevant project proposals.**

This project is currently in Phase-3, meaning significant effort has been exerted in previous semesters. Our task will be to incorporate this effort and improve it based on our client's preferences.

Specifically, the bulk of this project is composed of two parts: the UI and the matching algorithm. The project also makes use of an SQL database and additional related tools.

1.7 Requirements & Constraints

- 1) The matching algorithm should more **accurately match students to a project** based on their interests and preferences.
- 2) The web application should have **separate functions based on the user type**: proposal submission for clients, proposal selection for students, and proposal approval and assignment for administrators.
- 3) Should be ready to use in **Fall of 2024**.
- 4) The system should handle a **minimum of 100 concurrent users** without a significant increase in response time.
- 5) UI should be expanded to **allow Clients and Professors to enter projects**.
- 6) After project matching occurs, the **mass emailing process should be automated**.
- 7) The **database's design/structure** should be improved significantly.
- 8) The algorithm should be **proven to work as expected** in testing environments with large data.

1.8 Engineering Standards

React

Adhere to composition and design principles for the frontend UI.

Laravel

Use PHP for web interactions.

HTTP and HTTPS

Follow HTTP and HTTPS protocols to ensure secure and standardized communication between the client and server.

Database Management

Follow SQL best practices.

Version Control

Utilize Git and adhere to accepted practices for branching and commit messages.

Security

Develop using security best practices, such as input validation.

Issue Tracking

Keep track of issues and assign tasks ahead of time.

1.9 Intended Users and Uses

This project is intended to **benefit the future students and administrators of Iowa State's Senior Design class**. Automating much of the selection/assignment process will allow for a more efficient experience for all parties involved. Web forms for clients will ensure consistency among proposals, concise listings will help students find the projects they're interested in, and a dependable matching algorithm will allow administrators to allocate hundreds of students to their desired projects in minutes.

Industry and ISU professionals will:

- Submit project proposals.
- View the status of their proposal(s).

Students will:

- View available projects.
- Filter projects according to preferred majors and related keywords.
- Make manual modifications.
- Assess the effectiveness of the algorithm.

- Input their project preferences.
- Input their preferred teammates.
- Get notified of their assigned project.
- Administrators will.
- Approve proposed projects for selection.
- Run the matching algorithm.
- View statistics measuring the alignment of project assignments to student preferences.
- Additional Features.
- Team Communication.
- Generate Team Websites.

2 Project Plan

2.1 Task Decomposition

1. **Laravel Backend Development**
 - a. figure out Laravel routing
 - b. get REST endpoints working
 - c. test REST endpoints
2. **MariaDB Improvement**
3. **Matching Algorithm Enhancement/Alternative**
 - a. Analyze current algorithm
 - b. Define success criteria
 - c. Design new algorithm
 - d. Implement algorithm
 - e. Create test environment
 - f. test
4. **Frontend Maintenance**
 - a. Reduce duplicate code between React components
 - b. Analyze current table text alignment issues
 - c. Create constant css style for proper table text alignment
 - d. Add css style to all tables to fix text alignment issues
 - e. Fix webpage navigation (one dash shown depending on user type)
 - f. Look into login options
5. **Full System Testing**
 - a. Requirements analysis
 - b. test planning
 - c. write test cases
 - d. functional testing
 - e. usability testing
 - f. performance testing
6. **Client/User Quality Assurance**

2.2 Project Management/Tracking Procedures

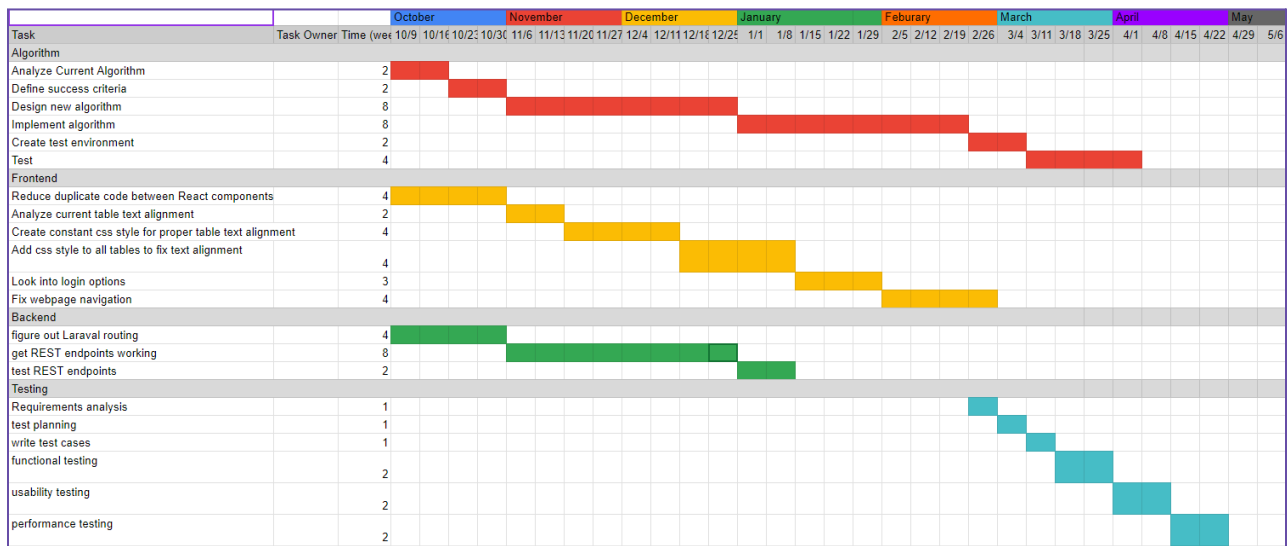
We plan on using an **agile project management** style. This will allow us to take an incremental approach to our project. This approach will allow us to get continuous feedback from stakeholders throughout the development process. An agile management style will provide the flexibility to change as our goals or requirements change. Integrated testing throughout the development process will satisfy our testing goals.

We will use **GitLab**'s project management features to track our two week sprints. Gitlab provides a central place for our repository and board to house our stories/work items.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Project matching algorithm matches **95% of students** with one of their top three projects.
- Database is reconfigured to better match requirement specifications.
- Frontend provides different functions for **three types of users** (students, clients, and administrators).
- Achieve 95% code coverage for unit testing.
- Deploy web application to live production server by the **end of spring**.
- Achieve 100% feature coverage.
- Wireframes and/or diagrams completed for new or revised features.
- Deliver final presentation.

2.4 Project Timeline/Schedule



2.5 Risks And Risk Management/Mitigation

<p>Risk 1. Description: Algorithm provides poor matching Probability: 55%. Factors increasing risk: The risk is increased by our lack of experience and thorough testing. This risk will decrease as we</p>	<p>Risk 2. Description: Web Service fails to recognize and direct users to the correct page for their use case. Probability: 15% Factors increasing risk: We will have to get information about the</p>	<p>Risk 3. Critically Poor Performance of matching algorithm probability 30% It could take a while Task can't be eliminated</p>	<p>Risk 4. Poor performance of Website Probability 15% modern web frameworks help to guide us and the webpage will be small. The main challenge will be ensuring we can</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>continue testing and gain more confidence.</p> <p>Factors Decreasing risk: we currently have an algorithm from the previous team. We should look into its viability as a point of comparison for anything we do in the future. Mitigation: expansive testing on real student data. We would want to identify if our algorithm maximized the valid matching. This task is not able to be eliminated. This task can not be replicated by an off the shelf component. We can take some inspiration from algorithms that utilize similar techniques.</p>	<p>user type from outside the application</p> <p>Factors decreasing risk: should be straightforward given we get good data. Navigation works manually. Task can't be eliminated. A very similar function can be found to assist us.</p>		<p>handle the large number data and clients at the same time</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	------------------------------------------------------------------

2.6 Personnel Effort Requirements

Task Description	Estimated Work Time (hours)
Research Algorithm	20
Run matching algorithm on testing data	7
Test data input in front end	10
Demonstrate website navigation	4
Work on Database	15

2.7 Other Resource Requirements

The web-server will need to be hosted with sufficient resources to allow all of the students to access it without crashing.

We may also need testing servers made available. These servers wouldn't require as high resource requirements.

Testing data could help us to verify and improve the algorithm.

4 Design

4.1 Design Content

Our project mostly consists of improvements and additions to the past iterations of the Project Matching system. We are expected to further the development of the React frontend, put together a well-structured database, and improve the reliability of the matching algorithm.

4.2 Design Complexity

Multiple components/Subsystems: This project involves the development of a web-based system that requires the integration of various technologies and principles Scientific, mathematical, and engineering principles:

- **Matching algorithm:** The development of a matching algorithm involves mathematical and algorithmic principles. Designing a new algorithm and analyzing its performance requires a deep understanding of algorithms and data structures. Our algorithm will also have to account for student and project sponsor preferences/skills.
- **Database Management:** Configuring and improving the current database involves principles of database design, data normalization, and query optimization. It also requires knowledge of database management like MariaDB.
- **Frontend Development:** Implementation of a web-hosted system frontend requires expertise in web technologies like React and UI design.
- **Software Architecture:** Designing a suitable client-server software architecture involves many software engineering principles and architectural patterns while maintaining performance and scalability

Challenging Requirements: Our project scope addresses many complex requirements like matching 95% of students with one of their top projects. This challenges our optimization of an algorithm that is much more complex than previous iterations now accounting for preferences and imbalance in project preferences. The focus on a diversity of skill sets and project complexity aligns with industry standards.

4.3 Modern Engineering Tools

Because our project is extremely software-oriented, our work is done with programming utilities, languages and frameworks for web, database and backend design.

Version control: We used GitLab to monitor and maintain updates to our codebase, as well as track issues and tasks among team members.

Programming IDEs: We used development applications to write, test and debug the software.

Database management: We used database management systems and query languages to preserve and manage user identities, roles, project preferences, and other attributes.

4.4 Design Content

Area	Description	Examples
Public health, safety, and welfare	We aim to enhance the student experience by ensuring everyone gets a desirable project. The efficiency of a single web application will also benefit faculty and advisors.	Our design will directly affect the well-being of the students by better matching them to a desired project. Faculty/advisors will also benefit from an improved web application to streamline the process. This will reduce the stress on both students and faculty by having a single application for the senior design class.
Global, cultural, and social	Our matching system will help clients build teams with diverse backgrounds from varying majors.	Our project we will assemble teams comprising individuals from various professional backgrounds, including software, electrical, cyber, and computer engineering majors
Environmental	The simplification of the project matching process to one service will require less resources.	This application will reduce any physical paper involved in the process of matching students to projects. Because it is a software application, we will make the system as efficient as possible
Economic	Using one platform will also allow for better efficiency in terms of time spent manually assigning projects.	This application will be cost effective as the only cost will be in the hosting of the application. Course administrators will spend less time on this process of the class saving man hours for the university.

4.5 Prior Work/Solutions

Advantages to our new design:

- **Matching algorithm:** Our new algorithm will now account for students preferences/skills when matching them to a project. This added complexity will better match students with a project they will be more successful in. We also plan on optimizing the current algorithm to account for an imbalance in the project preferences. This will make sure projects that are popular amongst students can possibly have two iterations.
- **Database Management:** Optimizing the existing database design will eliminate unused tables. This new design will be more readable which will allow course administrators to modify the database in the future.
- **Frontend Development:** Our new frontend will include new features outlined in our project plan. We will also fix issues noted in the previous iteration of this project.

Shortcomings to our new design:

- A lack of experience with developing project matching algorithms might pose potential risk

- Integration between the update backend and the pre existing front end code might cause a potential risk to our design

4.6 Design Decisions

Three Design Decisions we've made or will need to make during the design process:

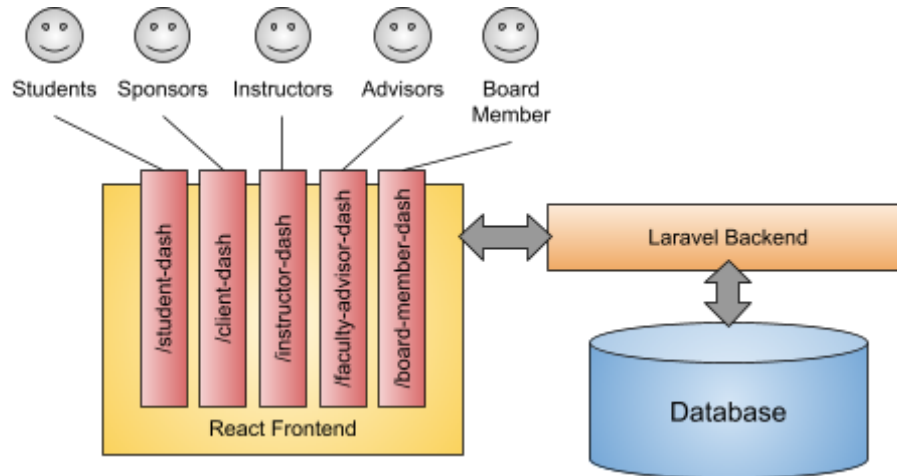
- **Continuing to utilize React on the frontend:** React's modular approach simplifies UI development, making it particularly well-suited for creating a complex interface with reusable components. Its virtual DOM will ensure responsive rendering, which is crucial for the project's performance requirements. React's abundant community support and state management options further streamline development.
- **Choice of matching algorithm:** We will need to decide on the specific matching algorithm to be used for the project. This decision involves selecting or developing an algorithm that can efficiently match students to projects while considering factors such as student preferences, project requirements, and diversity in team composition. The choice of the algorithm will significantly impact the project's success in meeting the 95% matching goal.
- **Authentication:** We will need to decide how authentication of the application will work. We'll need to design a system for user authentication and authorization, deciding on the method of user login (e.g., email, or single sign-on) and access control to various system features based on user roles.

4.7 Proposed Design

After getting familiar with the existing codebase and design decisions, we have:

- Tested instances of the frontend and backend designs.
- Designed new ER diagrams for the database.
- Tested modifications to React components to improve styling and reduce logical redundancy.
- Tried handling Laravel requests to return pages according to path.

4.7.1 Design 0 (Initial Design)

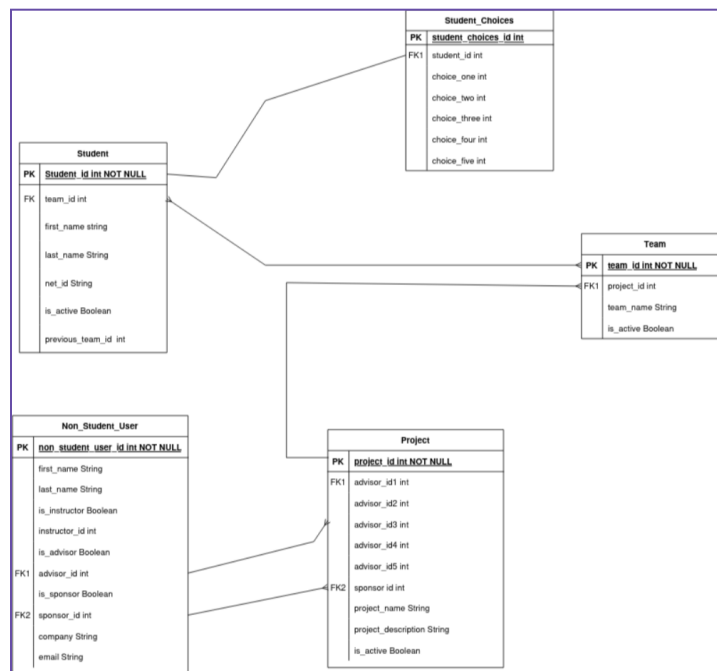


In a real world context, the design would be intended to operate by being used on individual devices which will access the website hosted on an ISU server.

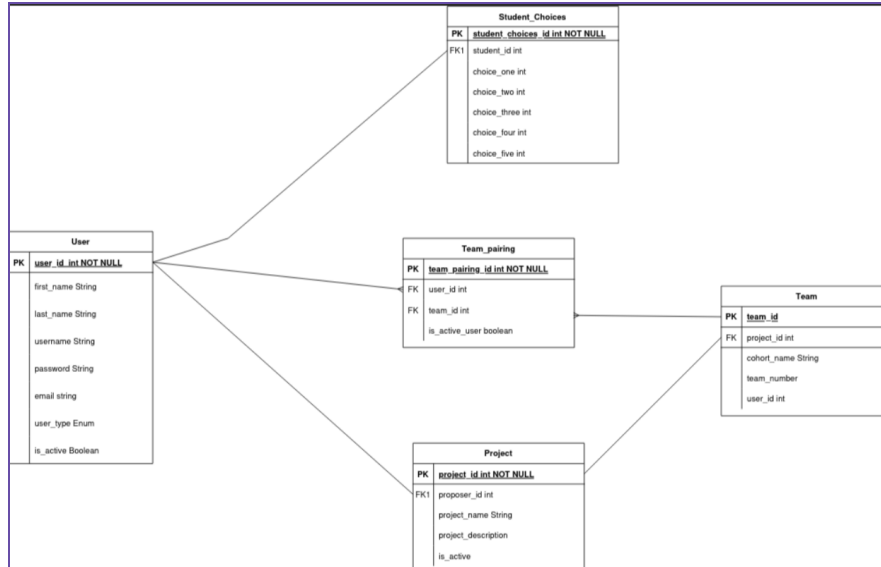
Currently neither the functional nor non-functional requirements are fully met in the design's current state. This will be something to continue working on. Currently most of the frontend functional requirements are met but almost none of the backend functional requirements are met. Some of the non-functional requirements are met for the algorithm.

4.7.2 Design 1 (Design Iteration)

Replacement for the previous team's database design:



Second iteration:



4.8 Technology Considerations

Some of the strengths and weaknesses of the technology that we chose are seen in our choice of going with React. One of the issues that it has is a lack of standardization when putting apps together. This is because React used to focus on using classes as its main way of creating components, where now it is encouraged to use functions instead. An alternative we could have used is Angular, but this would require starting over on the project and scrapping the previous team's work.

Another design solution would be our choice to use Laravel. Laravel provides a backend service without the need for a lot of dependencies, the issue with it is that it uses PHP as its language, which is not very syntax friendly. An alternative that we could have used was the Java Spring framework which has a very robust set of dependencies and derived frameworks (Spring Boot, Quarkus, etc.). However it was requested that we use laravel instead.

One consideration is the desired complexity of the matching algorithm. This is a field with many possibilities. The more complex we go, the more time testing we will need to ensure that the algorithm is working as intended. But there is also the possibility the algorithm will provide better matches. This is something that will require research to handle.

4.9 Design Analysis

Our proposed design from section 4.7 did not work. We were instructed to focus on the frontend and not worry about the backend, leading us to no longer have a heavy focus on it. As for the frontend. The design so far has worked well. Because it is still in design due to its size, we can not say for certain that it is successful.

5 Testing

5.1 Unit Testing

In our project there will be a lot of units that will need to be tested, most of this unit testing will need to occur in the backend. The tools used for this would be **PHPUnit** and **Mockery** These units will include:

- Response codes for the controllers.
- Data verification for models and controller services.
- Exception verification.

For the algorithm, we will write unit tests to verify the correctness of individual components or methods in our algorithm. We will use **JUnit** to automate the testing process. Each test will cover different scenarios and edge cases.

5.2 Interface Testing

For the frontend, we'll write tests to verify behavior. For instance, for tables with sorting functionality, we want to ensure that the tables are properly being sorted. The test will click on the sort button, then going through the columns and checking if it's sorted correctly. For Next, Back, and Submit buttons, we want to ensure that the buttons are working correctly. We will also write tests for the nav bar menu items to ensure redirection is correctly working.

Jest snapshot testing is a technique used in JavaScript and React testing. It involves capturing the output (such as rendered components or data structures) during the initial test run and saving it in a snapshot file. Subsequent test runs then compare the current output with the saved snapshot. If there are differences, the test fails, indicating potential unintended changes. Developers can review and either accept the changes by updating the snapshot or identify and fix issues if there's a bug. Snapshot testing helps ensure the stability of visual representations in your codebase.

Since we have a different dashboard for each user type, we can utilize snapshot testing for each dashboard. We can set the user type before each test and check if components are present. For instance, the Client dashboard will have a proposals section, whereas the Student dashboard won't.

5.3 Integration Testing

For the backend of this project there really is not much of a need for integration tests as the current plan will only use a single backend service. In the event this changes then they will be needed. The tool that would be used for this would be the laravel foundation testing tool.

For the algorithm, integration testing with the database is necessary to ensure that data is retrieved and stored correctly into our data structures we created. We will accomplish this by

using test data in the database and verifying it matches the correct functionality in our algorithm code. Integration testing the end points for the backend and the algorithm must be tested well.

5.4 System Testing

For the backend the closest thing to this would be the response code tests covered under unit testing; either the incoming request from the starting point worked or it didn't.

For system testing, our focus will be on validating that all components operate according to expectations. This entails ongoing testing using simulated or previous semester data. We will systematically conduct tests and address any identified issues until all functionalities operate. Multiple tests, employing test data, will be carried out across the algorithm, frontend, backend, and database.

5.5 Regression Testing

For the backend, this would be covered by the unit tests. If changes are made and a certain functionality stops working, then the test for that function will fail and since the unit tests are written in accordance with the requirements that would mean a requirement is not met.

Regression for the algorithm will include running unit tests, integration tests, and performance tests to verify new changes to the algorithm don't break old functionality and still perform as desired. Regression results will allow us to compare progress from previous iterations as well as ensure the algorithm is still functional. Our requirements contain many edge cases that will need to be continuously tested.

5.6 Acceptance Testing

Towards the end of development, we will do end-user testing to validate our project meets the expectations and requirements. We will allow users to interact with the system and provide feedback. We will also perform functional requirements testing to verify the system fulfills the pre-defined functional requirements we listed above. The data received from these tests will be analyzed and allow for the team to make adjustments to meet client expectations and requirements.

6 Implementation

The plan for next semester will be to get the backend up and running so that we can start submitting data to it. With this we will be able to create a more realistic test environment for the algorithm by actually submitting data the way a user would. Some other things that will need to be done are the instructor and advisor pages on the frontend, the pages are currently there but will need some polishing. We also intend on modifying the algorithm after finding the best way to optimize project matching.

7 Professionalism

7.1 Areas of Responsibility

Area of Responsibility	NPSE Canon	IEEE
Work Competence	Perform services only in areas of their competence; Avoid deceptive acts	<p>IEEE Code #6: Improve and maintain skills to complete tasks for others if qualified by training or experience.</p> <p>Both of these codes emphasize the importance of maintaining and improving technical skills. How competence is defined and maintained may differ between IEEE and NPSE Canon</p>
Financial Responsibility	Act for each employer of client as faithful agents or trustees	<p>IEEE Code #4: to avoid unlawful conduct in professional activities, and to reject bribery in all its forms;</p> <p>This code mentions rejecting bribery as a financial responsibility as an engineer. The NPSE canon has no mention of bribery.</p>
Communication Honesty	Issue public statements only in an objective and truthful manner	<p>IEEE Code 5: to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest, and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others</p> <p>This code stresses the importance of honest communication, acknowledgement of errors,</p>

		and truthful reporting. The emphasis on communication varies between the two codes
Health, Safety, Well-Being	Hold paramount the safety, health, and welfare of the public	<p>IEEE Code 1: to hold paramount, the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment;</p> <p>This code prioritizes the safety, health, and welfare of the public. The IEEE code goes further into depth on protecting privacy of others or anything that can harm the public specifically</p>
Property Ownership	Act for each employer or client as faithful agents or trustees	<p>IEEE Code 3: to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist</p> <p>This code advocates for respect, ideas, and information in relation to conflict of interest.</p>
Sustainability		IEEE Code 10: to support colleagues and coworkers in following this code of ethics, to strive to ensure the code is upheld, and to not retaliate against individuals reporting a violation.

		This code aligns with sustainability, because it highlights the importance of upholding the code of ethics to ensure a sustainable project.
Social Responsibility	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession	<p>IEEE Code #7: to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression</p> <p>This code highlights the importance of treat all people with respect and fairness. This differs from the NPSE canon, because it is focused less on the reputation fo the profession and more how people should be treated.</p>

7.2 Project Specific Professional Responsibility Areas

Area of Responsibility	How it relates to our project
Work Competence	It's important for us to have working knowledge of the tools we use to build the software. Working on a system like this without the necessary technical skills could be harmful to users.
Flnancial Responsibility	It will be important to use hosting/computing resources responsibly when deploying our new system.
Communication Honesty	We should be as transparent and communicative as possible when it comes to the behavior and design choices of the matching system.

Health, Safety, Well-Being	To protect the safety and well-being of all users, it will be important to protect the information provided by students, clients and advisors alike.
Property Ownership	We must respect the privacy of the information that is used in our matching system, and report any conflict of interest.
Sustainability	We must respect and report instances of security, privacy or safety violations.
Social Responsibility	The system must operate without unfair bias or discrimination.

7.3 Most Applicable Professional Responsibility Area

Health, Safety, and Well-Being will likely be the most applicable concern when building and maintaining the matching system. We must consider the necessary security measures to protect the information provided and ensure data is in the hands of the correct users.

8 Closing Material

8.1 Discussion

Our project was successful mostly in terms of planning and a little bit in terms of execution. We were able to outline our goals and tasks required to achieve these goals. Regarding the frontend, we made some progress in terms of cleanliness and alignment. We tried to improve the design by implementing the original wireframes. About half-way through the semester, we decided to forgo any backend improvements. The backend will need to be addressed to bring the project together and make it production ready.

8.2 Conclusion

For the algorithm, we have implemented and tested a simple bidding algorithm utilizing some of the resources from previous semester teams. We adjusted the algorithm to be optimized based on student preferences. In our initial testing efforts, we have found this type of algorithm to be successful based on the criteria we set early in the design phase. We received data from previous semesters to conduct testing, and found that the algorithm currently matched students better than manually project matching. Our goal for next semester is making changes to the algorithm to best match students to a project.


8.2.1 Team Contract

Team Members:

- | | |
|------------------------|--------------------------|
| 1) <u>Noah Nelson</u> | 2) <u>Joshua Izumba</u> |
| 3) <u>Devin Tigges</u> | 4) <u>Evan Brummer</u> |
| 5) <u>Max Kueller</u> | 6) <u>Robert Holeman</u> |

Team Procedures

- 1. Day, time, and location (face-to-face or virtual) for regular team meetings:**
 - 11:30 on Wednesdays, in person with TA.
 - 6:30 on Tuesdays via Discord.
- 2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):**
 - Discord.
 - E-mail.
- 3. Decision-making policy (e.g., consensus, majority vote):**
 - Majority vote of project members.
- 4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):**

 Record Keeping

Participation Expectations

- 1. Expected individual attendance, punctuality, and participation at all team meetings:**
 - Project members are expected to be in attendance of all team meetings. If an individual cannot attend or will be late to a specific meeting, a notice of 24 hours is required, except in the case of an emergency.
- 2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:**
 - Designated tasks should be completed or ready for review before or on the deadline (preferably before).
- 3. Expected level of communication with other team members:**
 - Project members should be in frequent, consistent communication. This is expected, at the very least, when a project member both starts or completes a specific task.
- 4. Expected level of commitment to team decisions and tasks:**
 - All project members should provide input when discussing a decision and assigning tasks.

Leadership

- 1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):**
 - **Noah:** Database Lead
 - **Devin:** Team Organization
 - **Max:** Client Interaction, Testing Lead
 - **Joshua:** Testing, Quality Assurance
 - **Evan:** UI/UX Lead
 - **Robert:** Architecture design
- 2. Strategies for supporting and guiding the work of all team members:**
 - Ask for help when needed.
 - Ask if other team members need help with their current tasks.
 - All team members should be familiar with all aspects of the project.
 - Setting the standard for your role. Making sure your work is a good example for other team members.
- 3. Strategies for recognizing the contributions of all team members:**
 - Documenting code authors.
 - Ensuring all tasks are listed in Gitlab with the appropriate authors.
 - Recognizing quality contributions during standup meetings.

Collaboration and Inclusion

- 1. Describe the skills, expertise, and unique perspectives each team member brings to the team.**
 - **Noah:** Industry backend experience (Springboot, Quarkus, microsoft sql server), SQL, React, linux
 - **Devin:** Industry experience in quality assurance (Selenium, Junit), UI/UX design (AdobeXD), embedded display development (C++, Qt, Linux), and team leadership through internships and class projects.
 - **Max:** Writing and communication experience (internship, Student Government, testing (Game development projects, did extensive testing for my internship, Junit)
 - **Joshua:** Backend and Frontend intern experience at Snapchat. Professional experience in React, TypeScript, Java, GraphQL, Bazel, Protobuf, and gRPC. Experience writing and presenting professional frontend and backend design docs, extensive testing, and creating sequence diagrams.
 - **Evan:** Professional experience in web/mobile development and UI remote configuration.
 - **Robert:** Full stack web development using React for front-end, Java/Kotlin for backend
- 2. Strategies for encouraging and support contributions and ideas from all team members:**
 - During team meetings, all team members will have a dedicated speaking time to share what they have been working on, or any other information they deem necessary to share.
 - Document to put ideas when not in a meeting to later be discussed

- 3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)**
- **Conflict with a specific individual(s):**
 - If a team member has a conflict with a specific individual(s) on the team, the dispute should be handled between themselves.