Senior Design Server/Client Development for Project Matching [Phase 2]

Final Design Document

SD '23 - Team #18

Clients: Jacob Grundmeier, Akhilesh Tyagi

Advisor: Akhilesh Tyagi

Haylee Lawrence	Lead Presenter, Minute Keeper, Testing, Document Editor
MyTien Kien	Team Organization, Client Interaction
Sanjana Amatya	Individual Component Design, Report Manager, Assignment Tracker
Alec Elsbernd	Lead Researcher, Floating Help

Team Email: sdmay23-18@iastate.edu

Team Website: https://sdmay23-18.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

Since most of our product is software-based, we used the most common software practices used in the industry. This included coding in an efficient and reusable manner to save time and resources. This was followed by code reviews from everyone in the group to ensure our code was up-to-date and efficient. Lastly, documentation and/or commenting code helped everyone tremendously regarding past and future work.

Applicable Courses from Iowa State University Curriculum

Below is a brief list of courses taken at Iowa State University that have helped our team develop our project.

- COM S 228: Introduction to Data Structures
- COM S 309: Software Development Practices
- COM S 311: Introduction to Design and Analysis of Algorithms
- COM S 319: Construction of User Interfaces
- COM S 327: Advanced Programming Techniques
- COM S 363: Introduction to Database Management Systems
- COM S 409: Software Requirements Engineering

New Skills and Knowledge

There have been several learning curves throughout this project, including gaining knowledge about something we have not taken a course for. We have also continuously learned new things about the other tools we used to help build our product. Below is a list of what we have been learning how to use.

- **Figma**: A collaborative web application for interface design
- **Material UI:** A Library of React.js components allowing for a seamless and consistent design. The components also help increase our functionality, such as allowing editable data tables.
- **Lavarel**: A backend framework that provides all of the features needed to build modern web applications, such as routing, validation, caching, queues, file storage, and more
- **Project Matching Algorithm**: An algorithm from a paper cited in Section 4.1. Our modified version matches students to projects based on Project Preferences and Groupmate Preferences. Algorithm Outputs listed in Implementation.

Table of Contents

List of Figures and Tables

1 Team

1.1 TEAM MEMBERS

Haylee Lawrence (Software Engineering)
MyTien Kien (Software Engineering)
Sanjana Amatya (Software Engineering)
Alec Elsbernd (Software Engineering)

1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

Frontend Development (React.js)
Backend Development (Laravel, Routing, API)
Database Development (MySQL)
CICD knowledge
Project management skills
Client interaction skills
Teamwork and communication skills
Agile experience

1.3 SKILL SETS COVERED BY THE TEAM

Frontend Development
- Haylee Lawrence, Alec Elsbernd, Sanjana Amatya

Backend Development
- MyTien (Routing, API)

Database Development
- Haylee Lawrence, MyTien Kien, Alec Elsbernd, Sanjana Amatya

CI/CD Knowledge
- Haylee Lawrence, Sanjana Amatya

Project Management Skills
- Haylee Lawrence, MyTien Kien, Alec Elsbernd, Sanjana Amatya

Client Interaction Skills
- Haylee Lawrence, MyTien Kien

Teamwork Skills
- Haylee Lawrence, MyTien Kien, Alec Elsbernd, Sanjana Amatya

Agile Experience
- Haylee Lawrence, MyTien Kien, Alec Elsbernd, Sanjana Amatya

1.4 PROJECT MANAGEMENT STYLE

Our group planned on adopting an Agile project management style because it was more applicable to our needs, with

- 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

Each individual on our team has more experience working in an Agile environment and is more comfortable using it. Using Agile, we worked on different aspects of the project simultaneously and combined them at the end of each sprint. The end of the sprint included a meeting discussing the parts we have been working on and what we like and don't like so far, including what we could improve on before we continue onto the next sprint. We also considered the stakeholders' feedback as the project continues; this is best possible in an agile environment.

1.5 INITIAL PROJECT MANAGEMENT ROLES

Haylee Lawrence (Lead Presenter, Minute Keeper, Testing, Document Editor)
MyTien Kien (Team Organization, Client Interaction)
Sanjana Amatya (Individual Component Design, Report Manager, Assignment Tracker)
Alec Elsbernd (Lead Researcher, Floating Help)

2 Introduction

2.1 PROBLEM STATEMENT

Our project aims to capture the Senior Design project Lifecycle. Our main focus was to streamline the Senior Design process for all users involved, specifically improving the project management system.

Questions We Considered:

1. How might we allow for group selection for professors/TAs/admins so that they need to do as little work as possible along with retrieving maximal results?
2. How might we allow for easy preferences input for Senior Design students so they can quickly and easily get the right project for them?
3. How might we allow Clients to easily submit project proposals?
4. How might we allow Senior Design instructors to easily evaluate project submissions?
5. How might we streamline Faculty/Industry panel signups?

This project exists because there is room to automate the Senior Design project system. We want everyone included in this process to be satisfied with the result. Although several students are satisfied with their current project and team, there's always room for improvement. Students and clients are most affected by the problem statements above. However, Instructors, Faculty Advisors, and Faculty and Industry Board members are also stakeholders to consider.

This problem only occurs within the Senior Design classroom/course. We aim to solve this problem as we want students to enjoy what they're doing throughout the project and have them end their college careers by creating a working project that they can present to the faculty. Our solution will be to create a web-hosted system that future Senior Design students can use to accurately assign projects based on their preferences.

2.3 INTENDED USERS AND USES

There will be several users using our product. These include but are not limited to: students, clients, Senior Design professors, Faculty advisors, and Faculty and Industry Board members. Below is a quick description of how a user may interact with the product and who they are.

TABLE 1

User	Key Characteristics	Needs	Benefit and Uses
Students	A student using this product will be a senior in electrical, software, computer, or cybersecurity engineering. This student would be taking the senior design course. Their goals would likely include ending their senior year with a great project and graduating. Depending on their preferences, the student would like a great group to work with and a project they are at least interested in.	The student needs a way to state project/group/working preferences because this is how the system would match students to specific projects.	Students will fill out their preference forms. Based on their answers, our product will help to accurately pick out the best project based on their needs and others. The goal is for the group assignments to be just as effective, if not more effective, than group assignments done by hand.
Clients	A client using this product will be either from the industry or a faculty member. Their goal would be to find student help with a specific project. They would also be willing to help and teach the students over the year.	The client would need to fill out the senior design project proposal form because the students would need to know what the project is about and what is needed out of the project.	The system would result in a group of students interested in the client's project, giving the client a team to work with to help get their project idea up and running.
SD Instructors	These Senior Design Instructors are faculty members here at Iowa State. Their role in this course is to teach students skills that may aid	Senior design instructors need access to all submitted projects because they will need to go through them and	As the current process stands, the senior design instructors are the ones that are doing the manual work of

	them in their senior design project. They are also responsible for the day-to-day grades in the class.	approve the viable projects. Senior design instructors will need a way to submit student lists to the database and ensure that it matches the roster because they will need to ensure that all Senior Design students receive a project and are correctly matched to their preferred groups/project.	assigning projects. They will benefit greatly from this project because it will significantly reduce the time and effort they must put into determining and distributing project assignments. This product will also create an easy-to-access collection of all project submissions to more easily determine which projects will be viable.
Advisors	Faculty advisors are ISU faculty members assigned to advise and supervise one or more senior design teams. They will be knowledgeable about the project that the student teams are working on and will be able to provide ample guidance to students throughout the year.	The Faculty Advisors need a way to access their assigned senior design groups because they will need to determine which group(s) have been assigned to their project(s).	The faculty advisors will benefit from this product because they will be assigned groups interested in their project. This tool will allow easier access to their project groups' contact info.
Review Members	Faculty and Industry Review Board members will be ISU Faculty or Industry Members who volunteer to sit on a Review Board to evaluate Senior Design Projects. The Faculty Review Board will evaluate projects at the end of SD 491, and the Industry Board will evaluate projects at the end of SD 491. Both parties have significant knowledge of various industries and the Project Lifecycle.	The Faculty and Industry Review Members need a way to sign up for timeslots, during which they will evaluate projects. They also need a way to access information regarding the teams they are evaluating (e.g., team number(s), access to the team website, and design document(s)).	The Review Board Members will benefit from this product because it will consolidate the time signup and team information into one place.

2.4 REQUIREMENTS & CONSTRAINTS

2.4.1 Engineering Standards

The following are Engineering Standards that apply to our project.

- IEEE/ISO/IEC 26531-2015
    - This standard provides the requirements for managing content used in the software development lifecycle. We consulted this standard to ensure we manage all content throughout the product lifecycle and the user and service management documentation process.
    - https://standards.ieee.org/ieee/26531/5753/

- IEEE P2887
    - IEEE P2887 is not a standard but a recommended practice that guides Developers on how to a Zero Trust Security (ZTS) architecture. A ZTS architecture helps us ensure our project's confidentiality, integrity, and availability of the data within it.
    - ZTS is an effective way to ensure comprehensive security for a project by asserting that no user, network, or application should be trusted by default.
    - https://standards.ieee.org/ieee/2887/10278/
- ISO/IEC/IEEE 16085:2021
    - This standard outlines risk management processes for projects to reduce/mitigate risks and better handle risks as they occur.
    - https://www.iso.org/standard/74371.html
- WCAG Version 2.1 and ADA Regulations
    - These standards and regulations will guide Developers on how to make a product that is accessible to all users.

2.4.2 Engineering Constraints

1. *Time:* The main constraint for this project was time. We were given limited time as we had only two semesters to fully plan and develop this project. If given more time, we would likely include more features and make a more well-rounded algorithm and more backend connections.

2. *Technology Used & Tools:* Since our project is started in phase 2, we needed to build our project off of last semester's work to maximize efficiency. For whichever portions we reused, we needed to emulate their technology as well as possible for an easy transition. Similarly, we were limited by which technologies we were familiar with. Thus, we needed to sacrifice reusability to maximize coding efficiency.

3. *Cost:* We were not given any money to help build our site, so we were constrained to using only free resources. Tools and frameworks such as our coding languages, and database all had to be free. We also needed to be mindful of the cost of IT managers maintaining the site. To limit the money the University needs to pay them to maintain our site, we made our project fault-tolerant, easy to use, and as safe as possible. If given a budget, we would have loved to experiment with hosting our project and database using Cloud tools such as AWS.

2.4.3 Functional Requirements

1. *Web-Based Application:* This project requires a front-end interface for students, clients, and faculty members to interact with. The front end will be where students set their preferences for groups. After team formation, this interface will also be how users can see the groups and project information.

2. *Database:* Our product needed a backend database to store the data received from the users. The database is essential in ensuring that users can choose their preferred group while also reaching ABET standards for diversity and inclusion.

3. *Team Formation:* One major aspect of our application will be how it can form teams. It considers the students' preferences on which project they want to work on and who they want to work with. It also considers group constraints such as the maximum number of students on a project and the maximum number of preferred groupmates on a project.

2.4.4 Non-Functional Requirements

1. *Usability and Humanity:* This product will be usable by Students, Clients, Senior design instructors, Faculty advisors, and Faculty and Industry Board members and can be used on the first attempt with basic knowledge about websites and how the Senior Design class operates. Our product creates a server/client system for the senior design course to organize and accurately assign projects to teams and clients based on preferences.

2. *Performance:* The product collects data from the seniors in 491 using a preferences form that will assist students in choosing and differentiating what they would like to do in terms of their project or what they're looking for overall. The product uses a Project Matching algorithm (detailed in the Problem Matching Algorithm Section) to compare a student's preferences in each database column and assign projects.

3. *Security:* The product's data and functionality can only be accessed by authorized users and employees of Iowa State University. The system should protect data in the product's database from corruption and unauthorized/ accidental disclosure. Data that has been printed as a hard copy should be properly disposed of. The product will also retain all records of data processed out of the system to ensure the safety of its users.

4. *Cultural and Political:* The product uses English as its default language. It will not use any offensive wording, icons, or pictures that could displease users.

2.4.5 User Interface and Experience Requirements

We want the users to have an easy yet quick experience when choosing their preferences or submitting project proposals. Students input their information in a way that both sides (students and clients) can easily understand. The preferences form allows users to easily choose and differentiate what they would like to do regarding their project or what they're looking for overall. Everything is accessible on both the web and mobile devices. Below is a brief list of what every user may need regarding user interface and experience requirements.

TABLE 2

User	Requirements/Needs
Student	Picking project preferences, access to GitLab, website
Client	Project proposals
Senior Design Instructors	Grading system, access to GitLab, approving projects website, running project matching
Faculty Advisors	Grading system, access to GitLab, website, assigned to project proposals
Faculty & Industry Review Board Members	Signing up for review times, viewing team information and websites

2.5 DESIGN EVOLUTION

At the beginning of the semester, we were equipped with a solid Design Plan. We had a site Wireframe in Figma, a solid understanding of React.js, a minimal understanding of Laravel, and a general idea that we wanted an Auctions Algorithm.

To start the semester strong, we planned on completing as much as possible at least three weeks before our presentation data. Everything was detailed in a shared scheduling Excel where we documented all the tasks we needed to complete and when we would complete them. By the end of the first few weeks of the semester, we had completed 22 screens for the frontend in React.js and had started working on designing an algorithm. Our algorithm design process involved researching the specific heuristics we could use to maximize student satisfaction with groups and projects.

The backend team worked on creating and deploying the Laravel backend server and setting up Laravel locally with the assistance of one of the group's advisors. This process took an extended amount of time due to technical challenges that needed to be solved and remained a difficulty that our team had to deal with for the rest of the semester. Close to halfway into the semester, we realized that one of the errors that was preventing all of our progress was solved by removing all React Native dependencies from the frontend code, which was a tedious process. Doing so, however, allowed us to display the frontend on the Laravel site. The next challenge involved figuring out how to connect both frontend and backend pages on the application, and it was not completed until the end of the semester.

By mid-semester, we were also running into the issue that we had not successfully designed an algorithm yet. The difficulty lay in the complex nature of the algorithms we were designing, which made it difficult to approach coding them. Later in the semester, we found an article with an algorithm that fit our criteria for a project-matching algorithm. We coded our algorithm based

on the pseudocode, adjusting it so that the matching considered groupmates instead of lecturers. Meanwhile, the backend team had finished up backend controllers and was working on getting dummy data from our database to show up on the frontend react pages.

Overall, by the end of the semester, we ended up with a fully fleshed-out Fronted hosted on a Laravel server, few database-frontend connections, and a Project matching algorithm coded separately from our project. Due to the disparity in frontend vs. backend knowledge, our frontend team consistently got more work done faster, leading to some issues. This also caused the main interoperability issue between our site and the algorithm. The frontend team finished work far before the backend team, so they started developing the algorithm in Java. By the time the backend connections had been made and the algorithm was completed, we ran into the issue that Java and Laravel do not work well together, and we would have needed to convert the Java code into JSX code in one week. Our website is not fully functional, but it's a solid demonstration of how the project-matching site will look and behave. Future iterations of the project will have an easier time visualizing the Project Matching process and can improve upon our algorithm.

3 Testing

3.1 TESTING PROCESS

As described below, many tests can be carried out to ensure our system works as specified. We have spent some time performing user-testing our webiste but could not test as in-depth as we would have liked. With more time and a working frontend-backend connection, we would have liked to have done more rigorous and comprehensive testing.

3.2 TESTING RESULTS

3.2.2 USABILITY TESTING

Our group performed Usability Testing for our website using a Google form. We asked our testers to perform 4 tasks and recorded if they could complete the task, how long it took them to complete the task, and any comments they had. Overall, we had 6 responses, which gave us a lot of constructive feedback that we can use to improve our Website.

The Questions and the Responses are recorded in Appendix V.

The following are the big takeaways from our testing:

- Student preference forms were confusing to users - they should be more simplistic
- The site was all too similar looking, which made it hard to navigate
- The Project Matching page was either was non-responsive or quick but confusing
- Some found Approving/Declining projects to be tedious
- There is no way to create a new Proposal as a Client

There were many other useful points made by our testers that will make our site more usable in the future. Unfortunately, we ran out of time to be able to meaningfully implement the feedback we received, but we believe that this information will be useful for future iterations of the project to improve upon the site.

3.3 FUTURE TESTING PLAN

Had we had more time, we would have liked to perform more comprehensive testing on all four main components of our site: the frontend, the backend, the database, and the project's algorithm. Below is a general summary of future testing work that must be done in the future. Each summary will have a set list of requirements each component shall pass to guarantee a well-tested web application.

TABLE 3

Components	**Summary**
Frontend	The user can navigate to any page on the web application based on the role and privileges they are given. For the frontend components, we will conduct unit and integration tests that go through all performance requirements from a user's perspective and ensure user inputs and all UI functionality are set up perfectly. We also want to ensure our software matches the business and client's goals.
Backend	This component will focus more on the product's functionality and how well the features work with new code changes throughout the semester. An example is connecting to the database between the frontend and backend components and going through different user scenarios to ensure the data is transferred properly and securely when requested.
Database	All data in the database will be returned correctly to ensure the connection between all components works properly. Data return for queries should not take more than one second, and all data deleted or inserted should be put in their proper table.

Algorithm	Our team will test the Project Matching algorithm by taking in input, calculating the expected output, and comparing it to the output the algorithm gives us. Our main goal is to check if it's time efficient and not faulty. This means we need to ensure that it can take in bulk amounts of data without crashing the product.

Examples of what guidelines we will be following to write functional and quality tests:

1. Simple and transparent tests
2. Creating user test cases with the end product and client in mind
3. Avoiding repetitive test cases
4. Test cases will include a description of what is being tested, an explanation of how it will be tested, and the expected results.

4 Implementation Details

Our project is Phase 2 of the Senior Design Server/Client Development for Project Matching. As there were a lot of similarities between the goals and requirements the 1st and 2nd iterations were trying to achieve, our team decided to build off of the code they wrote for the web application and improve on the aspects they were lacking. Team 3 used Laravel and Vue.js, while Team 44 used a Laravel Backend, an Angular frontend, and Angular components and services. We think they did a good job with their database components but lacked an in-depth algorithm, which we have decided we want to focus on the most. We also wanted to improve on their basic Frontend.

To remain consistent and allow for reusability, we have decided to use Laravel for the backend, and because we were familiar with it, we chose to make a React.js frontend. We chose consistent and familiar coding languages because we wanted to be able to focus on the algorithm aspect of the project. As of May 2023, We have completed a simple algorithm that matches students to projects based on their Project and Groupmate preferences. Due to issues detailed in the Design Evolution section, the website we created is not fully functional but is up on our site (linked in Appendix IV) to show what views would look like based on the user's role.

4.1 PROJECT MATCHING ALGORITHM

The Algorithm coded for this project is a modified version of the SPAStudent Matching Algorithm proposed in “Two algorithms for the Student-Project Allocation problem” by Abraham et al. (cited below). This algorithm focuses on assigning students to projects and lecturers based on student project and lecturer preferences. This SPA algorithm is an example of a two-sided matching problem where a set of participants can be partitioned into two disjoint sets, A and B.

Our modified version takes the SPA algorithm and replaces Lecturers with preferred Groupmates. Each student has 3 Project Preferences and 3 Groupmate Preferences and is allocated to a project according to all 6 preferences. The algorithm also considers the maximum number of students in a project, if students have the right major to be in a project if their groupmate preference does not prefer to be in a group with them, and more. Detailed Pseudocode is listed below and the GitLab with the Algorithm is linked in Section 7.4.

All code was written and tested in Java.

4.1.2 Pseudocode

```
for si in all students
      Project pj = si's highest bid project preference
      Student lk = si's highest bid groupmate preference

      if si has no project preference and has a groupmate preference
            pj = a valid* project for si and lk
      if si already has a project pi
            if lk has a project pk
                  sum = si's bid for pk
            sum += si's bid for pi
            if sum > si's bid for pj
                  pj = pi
      assign si to pj

      while pj is not a valid* project for lk
            else choose lk = si's next highest groupmate bid

      while lk has a project pk
            total = lk's bid for si
            total += lk's bid for pj
            if total < si's bid for lk
                  assign lk to pj
            else choose lk = si's next highest groupmate bid

      while pj has too many students
            sr = student with the lowest bid for pj
            remove sr from pj
for si in all students with no project
      if si has a highest bid project pb and it is valid*
            pj = pb
      else if si has a highest bid groupmate lk who has a valid* project
      pk
            pj = pk
      else
            pj = the first open project that is valid*
```

*valid projects are ones where the student is in one of the required majors for that project

Matching Results

The following are the output from two separate Project matchings.

(1) Students + Preferences:

```
Student 1: major: Cyb
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 3
    Student Pref 1: Groupmate ID 3, Bid Amount 3
Student 2: major: Cyb
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 1
    Project Pref 2: Project Number 3, Bid Amount 2
    Student Pref 1: Groupmate ID 4, Bid Amount 2
Student 3: major: SE
    Project Pref 1: Project Number 3, Bid Amount 5
    Student Pref 1: Groupmate ID 1, Bid Amount 1
    Student Pref 2: Groupmate ID 6, Bid Amount 1
Student 4: major: CE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 2
    Project Pref 2: Project Number 3, Bid Amount 2
    Student Pref 1: Groupmate ID 2, Bid Amount 3
    Student Pref 2: Groupmate ID 7, Bid Amount 4
Student 5: major: SE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 2
    Project Pref 2: Project Number 3, Bid Amount 2
    Student Pref 1: Groupmate ID 2, Bid Amount 3
    Student Pref 2: Groupmate ID 7, Bid Amount 4
Student 6: major: SE
    Project Pref 1: Project Number 1, Bid Amount 4
    Student Pref 1: Groupmate ID 3, Bid Amount 3
Student 7: major: EE
```

```
Student 8: major: CE
    Project Pref 1: Project Number 2, Bid Amount 2
    Student Pref 1: Groupmate ID 2, Bid Amount 1
Student 9: major: CE
    Project Pref 0: Project Number 3, Bid Amount 1
    Project Pref 1: Project Number 2, Bid Amount 2
    Project Pref 2: Project Number 3, Bid Amount 1
    Student Pref 1: Groupmate ID 8, Bid Amount 3
Student 10: major: Cyb
    Project Pref 0: Project Number 3, Bid Amount 1
    Project Pref 1: Project Number 2, Bid Amount 2
    Project Pref 2: Project Number 3, Bid Amount 1
    Student Pref 1: Groupmate ID 8, Bid Amount 3
Student 11: major: EE
    Project Pref 0: Project Number 3, Bid Amount 1
    Project Pref 1: Project Number 1, Bid Amount 1
    Student Pref 1: Groupmate ID 12, Bid Amount 2
    Student Pref 2: Groupmate ID 13, Bid Amount 3
Student 12: major: EE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 3
    Student Pref 1: Groupmate ID 5, Bid Amount 1
    Student Pref 2: Groupmate ID 11, Bid Amount 1
Student 13: major: SE
    Project Pref 1: Project Number 2, Bid Amount 1
    Student Pref 1: Groupmate ID 10, Bid Amount 4
    Student Pref 2: Groupmate ID 11, Bid Amount 2
```

Figures 1 & 2: Example Students and Preferences #1

(1) Projects

```
Project 1
    Required Majors SE Cyb CE
Project 2
    Required Majors EE
Project 3
    Required Majors SE Cyb CE EE
```

Figures 3: Example Projects #1

(1) Results

```
Student 1: major: Cyb
    Matched Project 1
Student 2: major: Cyb
    Matched Project 1
Student 3: major: SE
    Matched Project 3
Student 4: major: CE
    Matched Project 3
Student 5: major: SE
    Matched Project 1
Student 6: major: SE
    Matched Project 1
Student 7: major: EE
    Matched Project 2
Student 8: major: CE
    Matched Project 1
Student 9: major: CE
    Matched Project 3
Student 10: major: Cyb
    Matched Project 1
Student 11: major: EE
    Matched Project 3
Student 12: major: EE
    Matched Project 2
Student 13: major: SE
    Matched Project 1
```

Figures 4: Example Project Matching Results #1

(2) Students + Preferences

```
Student 1: major: Cyb
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 3
Student 2: major: CE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 1
    Project Pref 2: Project Number 3, Bid Amount 2
    Student Pref 1: Groupmate ID 4, Bid Amount 2
Student 3: major: SE
    Project Pref 1: Project Number 3, Bid Amount 5
    Student Pref 1: Groupmate ID 1, Bid Amount 1
Student 4: major: CE
    Student Pref 1: Groupmate ID 2, Bid Amount 3
    Student Pref 2: Groupmate ID 7, Bid Amount 4
Student 5: major: SE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 2
    Student Pref 1: Groupmate ID 12, Bid Amount 1
Student 6: major: CE
    Project Pref 1: Project Number 1, Bid Amount 4
Student 7: major: EE
    Student Pref 1: Groupmate ID 4, Bid Amount 7
```

```
Student 8: major: CE
    Project Pref 1: Project Number 2, Bid Amount 2
    Student Pref 1: Groupmate ID 2, Bid Amount 1
Student 9: major: EE
    Project Pref 1: Project Number 1, Bid Amount 1
    Student Pref 1: Groupmate ID 8, Bid Amount 3
Student 10: major: Cyb
    Project Pref 0: Project Number 3, Bid Amount 1
    Project Pref 1: Project Number 2, Bid Amount 2
Student 11: major: EE
    Project Pref 0: Project Number 3, Bid Amount 1
    Project Pref 1: Project Number 1, Bid Amount 1
Student 12: major: SE
    Project Pref 0: Project Number 2, Bid Amount 2
    Project Pref 1: Project Number 1, Bid Amount 3
Student 13: major: Cyb
```

Figures 5 & 6: Example Students and Preferences #2

(2) Projects

```
Project 1
    Required Majors SE CE
Project 2
    Required Majors Cyber CE EE
Project 3
    Required Majors SE Cyb EE
```

Figures 7: Example Projects #2

(2) Results

```
Student 1: major: Cyb
    Matched Project 3
Student 2: major: CE
    Matched Project 2
Student 3: major: SE
    Matched Project 3
Student 4: major: CE
    Matched Project 2
Student 5: major: SE
    Matched Project 1
Student 6: major: CE
    Matched Project 1
Student 7: major: EE
    Matched Project 2
Student 8: major: CE
    Matched Project 2
Student 9: major: EE
    Matched Project 2
Student 10: major: Cyb
    Matched Project 3
Student 11: major: EE
    Matched Project 3
Student 12: major: SE
    Matched Project 1
Student 13: major: Cyb
    Matched Project 3
```

Figures 8: Example Project Matching Results #2

Works Cited

Abraham, D. J., Irving, R. W., & Manlove, D. F. (2007). Two algorithms for the Student-Project Allocation problem. Journal of Discrete Algorithms, 5(1), 73-90. https://doi.org/10.1016/j.jda.2006.03.006

4.2 Discussion of Algorithm Source Material

Abraham et al's matching algorithm aims to match projects using worker preferences and requirements.

Each worker is given a score based on their preferences for different types of projects, and each

project is given a score based on its requirements for different types of workers. Abraham's algorithm uses a variant of the Gale Shapley algorithm which tries to match workers to projects by assigning them to projects they score high for. The Gale Shapley algorithm and the Stable Matching algorithm both use table matching between two sets of participants, where each participant has preferences over the members of the other set. The score is produced by using a mathematical formula to calculate the optimal match between workers and projects based on their scores. The formulas in Figure 1 consider workers' preferences, the requirements of projects, and the quality of the match to ensure the matching is stable (undersubscribed or over).

The following properties of F must hold.

1. Any assignment in M involving l_k that was made after time T must involve a project from F, since s' is the worst student in $M(l_k)$.
2. Every $p \in F$ is full at time T, otherwise l_k would not have offered p' to s'.
3. $p_j \in F$, since $s_i \in B$ by Condition (b), and (s_i, p_j) is not deleted by Lemma 5.2, which implies that $(s_i, p_j) \notin M'$, since $(s_i, p_j) \notin M$.

Now since $p_j \in F$, the number of students assigned to l_k in M' is given by

$$|M'(l_k)| = \sum_{p_f \in F \setminus \{p_j\}} |M'(p_f)| + |M'(p_j)| + \sum_{p_g \in P_k \setminus F} |M'(p_g)| \leq d_k. \quad (1)$$

Similarly, the number of students assigned to l_k in M is given by

$$|M(l_k)| = \sum_{p_f \in F \setminus \{p_j\}} |M(p_f)| + |M(p_j)| + \sum_{p_g \in P_k \setminus F} |M(p_g)|.$$

Now, since all assignments in M involving l_k that were made after time T only involve projects from F (Property 1) and all projects in F are full in M' (Property 2), we have that

$$|M(l_k)| \leq \sum_{p_f \in F \setminus \{p_j\}} |M'(p_f)| + |M(p_j)| + \sum_{p_g \in P_k \setminus F} |M'(p_g)|.$$

Finally, we are given that p_j is under-subscribed at the termination of E (Condition (b)). Therefore

$$\begin{aligned} |M(l_k)| &< \sum_{p_f \in F \setminus \{p_j\}} |M'(p_f)| + |M'(p_j)| + \sum_{p_g \in P_k \setminus F} |M'(p_g)| \\ &= |M'(l_k)| \leq d_k \end{aligned}$$

by Equation 1. So, l_k is under-subscribed at the termination of E, contradicting Condition (b).

Figure 9: Formulas Used in the Abraham et al Algorithm

The algorithm uses linear programming to optimize a linear objective function subject to linear constraints. In this case, the objective function is the overall quality of matches, and the constraints ensure that each worker is assigned to only one project and each project is assigned to only one worker. The quality of a match is calculated using the dot product of the worker's preference vector and the project's requirement vector. The preference vector represents the worker's preferences for different projects, and the requirement vector represents the Project's requirements for different types of workers.

The algorithm uses constraints to ensure that each worker is assigned to only one project and each project is assigned to only one worker. These constraints are linear equations that specify that the sum of the assignments for each worker or project must equal 1. The algorithm also uses additional constraints to ensure that the number of workers assigned to each project meets its staffing requirements and that the number of projects assigned to each worker does not exceed their capacity. These constraints are linear inequalities that specify that the sum of the assignments for each worker or project must be greater than or equal to a minimum value and less than or equal to a maximum value.

The resulting linear programming problem is solved using standard optimization techniques, such as the simplex method or interior point methods, to find the optimal work assignment to projects that maximize the overall quality of matches subject to the constraints. The algorithm is iterative and therefore keeps running to improve the quality of matches over time.

5 Security Concerns and Countermeasures

5.1 PHYSICAL SECURITY

Physical Security related to our project would involve the physical security of the Servers running our site. Another security concern would be related to the physical security of the computers/laptops of the admin with access to our Server VM/Database. The physical security of the servers falls outside the scope of this Design Document since our group cannot control Campus Security. Physical security regarding the computers/laptops of either IT, Admin, or our Group is highly important and something our team can control. ISU employees must be careful with their work computers and thus ensure no one gains access to their devices. Our group members are also incredibly careful with our devices and have our Database password protected so a user that gains access to our physical computers cannot access the Database.

5.2 CYBER SECURITY

The Project Matching site deals with many users and their data. Thus we have taken measures to ensure that our site is as secure as possible for our Users.

5.2.1 Following Best Coding Practices

When creating our site, every group member has followed the best coding practices. When coding the frontend, we ensured that every user would only have access to the information they needed to see. For example, there was no page listing all Senior Design students in the Client views. We also minimized Extraneous Functionality by only coding the pages users needed to have and ensuring that we only collected necessary data. Lastly, we utilized a strict and secure

Login Function using Laravels routing capabilities. Taking these steps has ensured that we limit vulnerabilities from the outset.

5.2.2 Testing Against Common Attacks

The deliverable we currently have does not have connections from the frontend to the backend and is mostly static (not allowing changes in the frontend dynamically), which prevents most of the Vulnerabilities listed below (SQL Injection, Cross-site Scripting, Brute Forcing). We also do not have routing based on user type completed, so URL Manipulation is possible. Future project iterations should test for the following common vulnerabilities once frontend-backend functionality is completed.

Vulnerability	**Test**	**Solution**
SQL Injection	Injecting SQL statements such as *hacker' AND password = 'whatever' OR '1'='1'* Into all of the forms that access the Database (e.g., Login Form) to see if we can (a) Login without a password (b) View/edit/delete data	Sanitize inputs and use the Principle of Least Privilege.
Cross-Site Scripting	Inputting malicious HTML into inputs Ex. *“onmouseover= alert(‘hello’);”*	Sanitize and validate inputs.
Brute Forcing	Attempting to Login multiple times with common passwords.	Allow three incorrect login attempts before locking the account for five min. After nine consecutive incorrect Login attempts, IT must be contacted. Requiring Users to have complex passwords.
URL Manipulation	Attempting to access secure pages for different Users/User types by manually inputting the URL (ex. www.[site].com/User01Info)	Perform user permission checks before displaying a page.

6 Related Works Context

6.1 RELATED PROJECTS

While there are lots of educational system-related web applications that can be compared to our web application, there are none that we have found that perform automated Project Matching like ours. For example, in Canvas or Google Classroom offer project or classroom management capabilities such as our project, but as far as we have found, do not offer automated project matching based on preferences.

Other related project matching projects include the CapSource Project Matching Capstone Project (source), and the 2 teams for Phase 1 of this project. While the Phase 1 teams also considered the Senor Design lifecycle, CapSource's project and other projects we've seen do not aim to create a Project Management site as ours does.

6.2 RELATED LITERATURE

The concept of Project-Matching algorithms is not new. While the concept of using an Auctions Algorithm for Project Matching, doing a basic search for project matching algorithms returns a number of projects and papers.

For our group, the main literature reference for our Algorithm came from the"Two algorithms for the student-project allocation problem" written by Abraham, D.J., Irving, R.W. and Manlove, D.F. We used this piece as an inspiration for our algorithm. The algorithm we created and implemented in this project had similar requirements to the algorithms in this paper and ended up having a lot of influence from those algorithms.

APPENDICES

Appendix I: Operation Manual

1. **Brief descriptions of the folders for Backend, Frontend, Algorithm:**
   Backend
   The Backend has five important types of files that run in the Laravel Server and make sure the application is running smoothly.

   1. .JSX files: The first type of file are the .jsx files located in the js folder under the public folder of the repository.

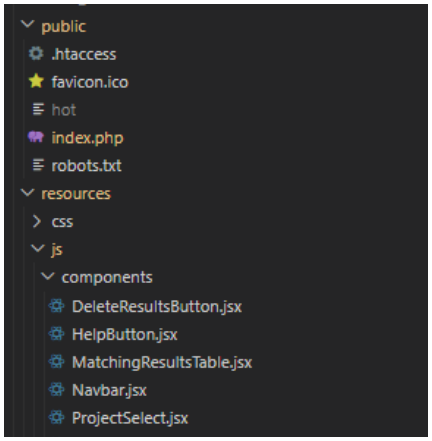


Figure 10: Public Folder

2. Controller files: The Controller files are under the Controllers folder in the app folder.

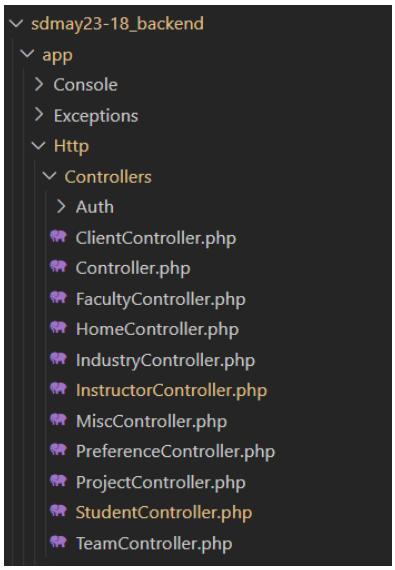


Figure 11: Backend Controllers Folder

3. Model files: These files are located in the Models folder right under the Controllers folder (in the app folder)

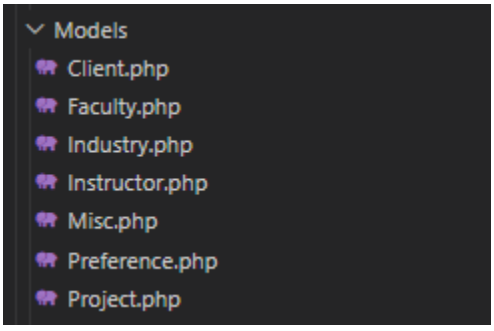


Figure 12: Backend Models Folder

4. Migration files: The Migration files can be found in the database folder under the app folder.

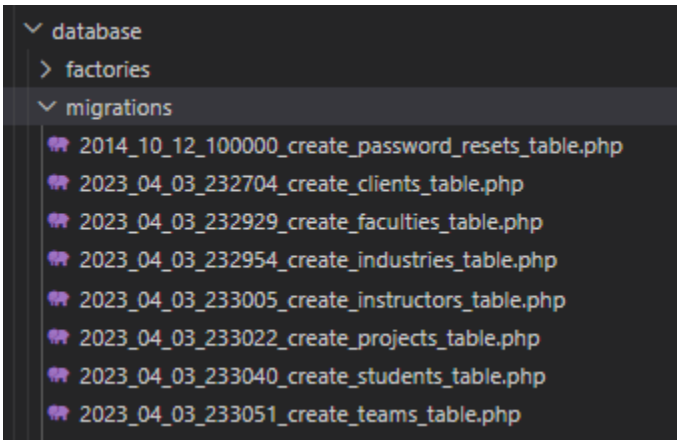


Figure 13: Database Migrations

5. Blade.php files: These files are located in the views folder under the resources folder in the repository.

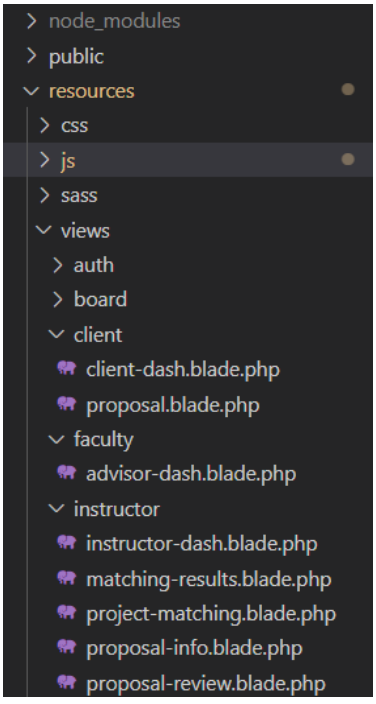


Figure 14: Backend Blade Files

How these files are connected:
The migration and model files are responsible for facilitating communication with the database. The migration files define the structure of the database tables, and the model files are an interface to manipulate the data in those tables. The Controller files have the logic determining how users interact with the website and the displayed data. They handle user requests and control how the application functions. These files call the

blade.php files that provide the structure and layout of the website and then call on the .jsx files that generate the visual elements of the website's interface.

Frontend

The frontend has five main folders that hold the code for the frontend of our website

1. The Board member folder
2. The Client folder
3. The Faculty Advisor folder
4. The Instructor folder
5. The Student folder

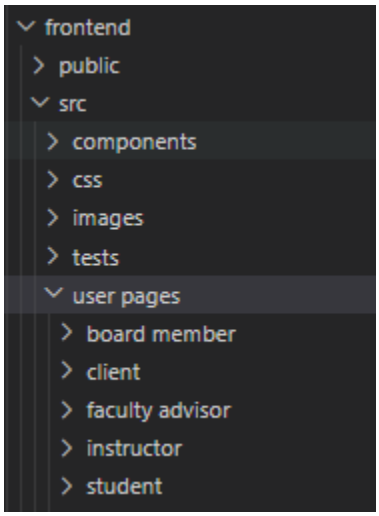


Figure 15: Frontend Folder Hierarchy

These folders contain important files like the Student Dashboard and Project Preference pages and are located under user pages folder within the src folder

Algorithm

The Algorithm is in the sdmay23-18 folder that also contains the sdmay23-18-backend folder that holds our backend code

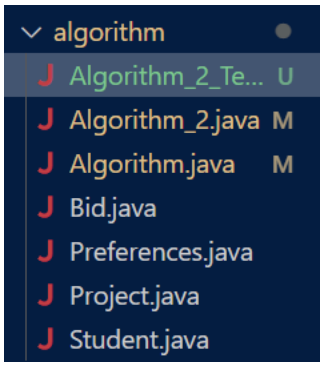


Figure 16: Algorithm Folder

1. The Bid.java, Student.java, Project.java files, etc. are all used to create student, project, and preferences objects to facilitate testing
2. The Algorithm.java file is the main driver code with printing functionality
3. The Algorithm_2.java contains our modified Algorithm code

2. Engineering Setup: Laravel Setup and Potential Errors

1. Download the repository from Gitlab
2. In a terminal, *cd* into the sdmay23-18 backend and run *php artisan serve*
    a. If this error occurs:

```
Alecs-MacBook-Pro-2:sdmay23-18_backend alec$ php artisan serve
PHP Warning:  require(/Applications/Git/sdmay23-18_backend/vendor/autoload.php)
: Failed to open stream: No such file or directory in /Applications/Git/sdmay23
-18_backend/artisan on line 18

Warning: require(/Applications/Git/sdmay23-18_backend/vendor/autoload.php): Fai
led to open stream: No such file or directory in /Applications/Git/sdmay23-18_b
ackend/artisan on line 18
PHP Fatal error:  Uncaught Error: Failed opening required '/Applications/Git/sd
may23-18_backend/vendor/autoload.php' (include_path='.:/usr/local/Cellar/php@8.
0/8.0.28/share/php@8.0/pear') in /Applications/Git/sdmay23-18_backend/artisan:1
8
Stack trace:
#0 {main}
  thrown in /Applications/Git/sdmay23-18_backend/artisan on line 18

Fatal error: Uncaught Error: Failed opening required '/Applications/Git/sdmay23
-18_backend/vendor/autoload.php' (include_path='.:/usr/local/Cellar/php@8.0/8.0
.28/share/php@8.0/pear') in /Applications/Git/sdmay23-18_backend/artisan:18
Stack trace:
#0 {main}
  thrown in /Applications/Git/sdmay23-18_backend/artisan on line 18
```

Figure 17: Laravel Possible Error

It means that the autoload.php files that are created at install/upgrade composer do not exist. To install composer run:

- *install composer*
- *upgrade composer*

This generates the necessary files to run php artisan serve which will have the server up and running.

3. Now in an IDE, open up the sdmay23-18 backend folder and run the command *npm run dev*
4. This will generate the localhost URL and APP_URL like the image below:

Figure 18: Laravel URL and APP_URL

If there is no APP_URL, go to the .env file in an IDE and change the code at the top of the file to match the code below:

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:qxsuulZOQfYw7tRgajflWYGDKRusUkrK4+VEp5IvQe8=
APP_DEBUG=true APP_URL=http://sdmay23-18.ece.iastate.edu/

And,
DB_CONNECTION=mysql
DB_HOST=sddb.ece.iastate.edu
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=sdmay23-18
DB_PASSWORD=JwU7brycJL

If there is no .env file, create one in the IDE, in a terminal or in a file explorer, add the code and save it.

Appendix II: Alternative Initial Versions of the Design

One alternative initial version included many more intended users. Besides our main users – Students, Clients, Senior Design Instructors, Faculty Advisors, and Board Members – we also had ABET Evaluators and IT managers. We removed ABET evaluators because they would

only need to access the site every few years and would therefore be better suited to being added in the future. We also decided to remove the IT Managers Frontend use case because they would not necessarily need access to our site but would instead be working with the server and database for our website.

Originally, our project would use an Auction Algorithm to facilitate project matching. Students would have a certain amount of points to bid, and they would bid on their preferred projects and group mates. At the start of the semester in SE 492, we were working to develop our own auctions algorithm, but after mid-semester, we realized that it would be infeasible for us to develop our own working algorithm, code it, and figure out how to insert it into a Laravel backend with the remaining time. We found a paper with a matching algorithm that was not an Auction Algorithm, and based on the pseudocode, we coded a working matching algorithm.

Our initial design had us implement previous teams' back ends to save time and be more efficient. Our initial reviews of their code during SE 491 made us optimistic that we could do so. After taking a closer look during SE 492, we realized that the other teams did not have enough of a backend for us to work with, and it would be easier to set up our own Laravel project and try to pick and choose from their back-end code. In the end, we ended up redoing their backend from scratch so that it would work with our React frontend. This ended up being a lot more time-consuming than we were hoping it would be. However, we have set a good foundation for future project iterations as we got our React frontend to work with our Laravel backend.

Appendix III: Other Considerations

We have learned a lot over the course of this project, mainly that while some backend and frontend frameworks can technically work together, that doesn't mean that they are easy to implement. We chose to use a React frontend because we were familiar with it, and we decided to use a Laravel backend because that was what both previous teams had implemented. After some research, we found that React apps could be implemented with Laravel. Once starting the coding process, however, we found it difficult to implement React with Laravel. We would have been better suited to use Spring Boot instead since it is another framework that we were all familiar with.

Another thing we learned is the difficulty of holding group members accountable when you do not see them regularly. While our group communicated regularly over Discord and had weekly meetings together, holding each other accountable for our work was difficult. Had we had daily or even bi-weekly in-person meetings to see each other, it would have been easier to hold each other accountable for weekly work. This lack of accountability led certain members to do less work than others. As a result, we learned the importance of meeting often to ensure every group member was pulling their weight.

Appendix IV: Code

All of our code can be accessed through our GitLab linked below:

** Please note that the GitLabs may not be Public, and viewers may need to request access

Frontend GitLab: sd / sdmay23-18_frontend · GitLab (iastate.edu)

Our Frontend repository contains the React.js app that houses all of our frontend code. This also has the original form of our algorithm.

Backend GitLab: sd / s dmay23-18_backend · GitLab (iastate.edu)

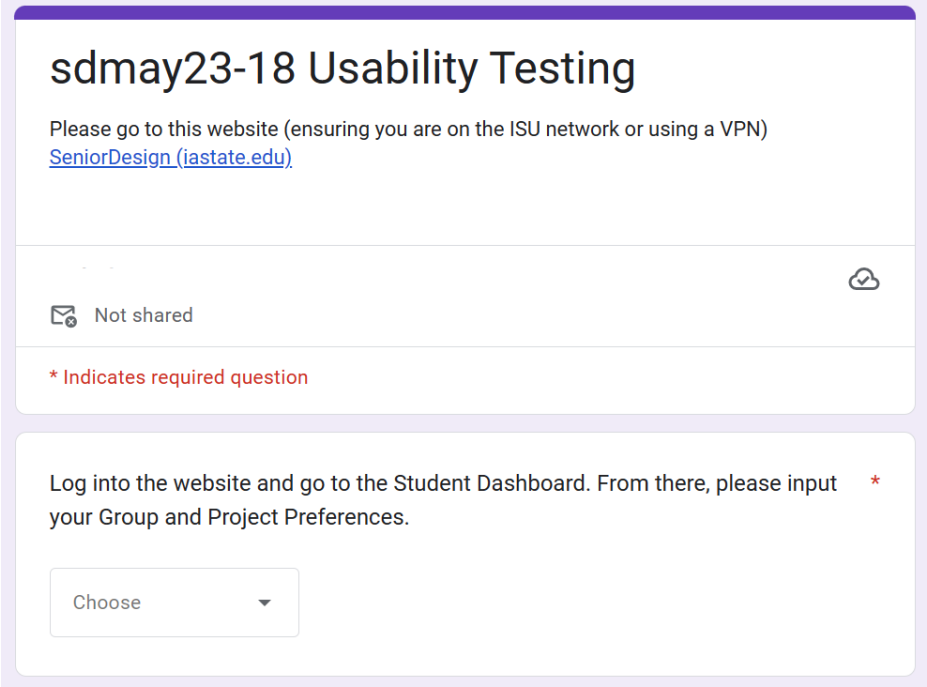
Our Backend repository contains our deliverable website code. It houses our Laravel app combined with our React frontend and Algorithm code.

Our Deliverable website can be found here: http://sdmay23-18.ece.iastate.edu/

1. Ensure you are connected to the ISU network (VPN or On Campus)
2. Open the Link
3. Navigate to the “Log In” button in the top right corner
4. Peruse the Frontend Mockup
5. For a working Frontend/Backend Use Case, please view the Demo Video linked on our Senior Design Website

Appendix V: Usability Testing Documentation

Form:



Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task? *

Your answer

Log into the website and go to the Student Dashboard. From there, please check your Group and Project Info. *

Choose

Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task? *

Your answer

Go to the Instructor Dashboard. From there, please Match Students to Projects by completing a Project Matching. *

Choose

Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task? *

Your answer

Go to the Instructor Dashboard. From there, please Review a Project Proposal and Accept or Deny the Proposal. *

Choose

Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task? *

Your answer

Go to the Client Dashboard. From there, please input a Project Proposal, and check an Existing Project Proposal. *

Choose

Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task? *

Your answer

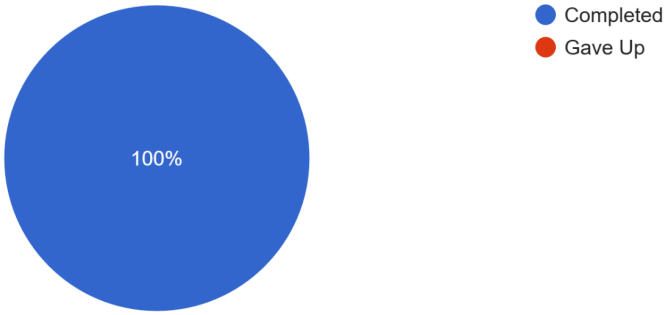
Submit

Clear form

Responses:

Log into the website and go to the Student Dashboard. From there, please input your Group and Project Preferences.

6 responses



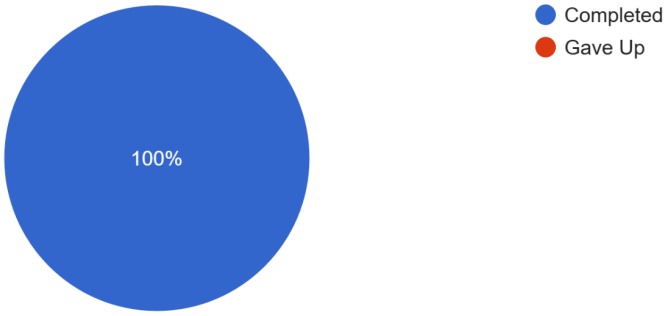
Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task?

6 responses

- It took me like 2 minutes, the points system confused me
- About 3 minutes. Why are there so many preferences options and why do we only have 15 points? I wish we had more points. The website also doesnt update if I've done things which is confusing.
- 4 min, I don't like the way the website looks, I struggled reading stuff and determining what is what because it all looks the same
- 3 minutes, I like how simple the website is
- A minute, no questions
- 2 min

Log into the website and go to the Student Dashboard. From there, please check your Group and Project Info.

6 responses



Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task?

6 responses

took me like a minute

1 minute. Was easy to find group info, project info might be too hidden. Why were there forms on the project info page that i couldnt update??

2 min, I like that it shows everyones name in columns, makes it easier to read

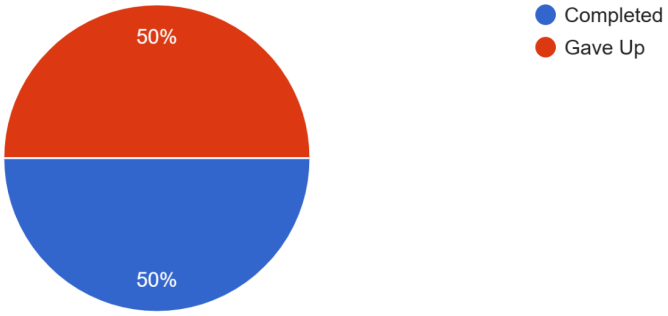
1 minute, I like being able to see everything related to my group

A minute, no questions

1 min

Go to the Instructor Dashboard. From there, please Match Students to Projects by completing a Project Matching.

6 responses



Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task?

6 responses

It did not load

1 minute. What was the point of the multiple bidding inputs? Also seems buggy, but why were there no results after project matching? Is there supposed to be a csv or something?

Would not load

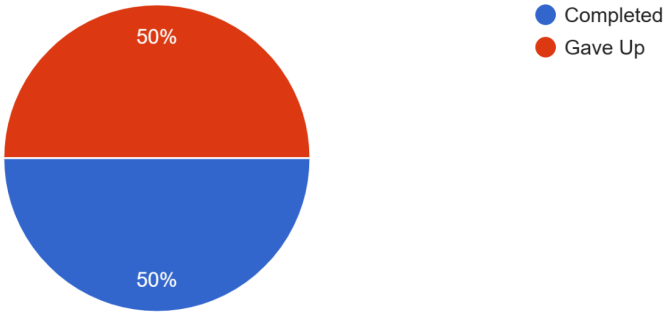
Could not access

A few seconds, no questions

Less than a minute, It was very quick, not sure what I was doing though

Go to the Instructor Dashboard. From there, please Review a Project Proposal and Accept or Deny the Proposal.

6 responses



Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task?

6 responses

It did not load

2 minutes. There was a lot of pages that I had to click through is there an easier way to do that?

Would not load

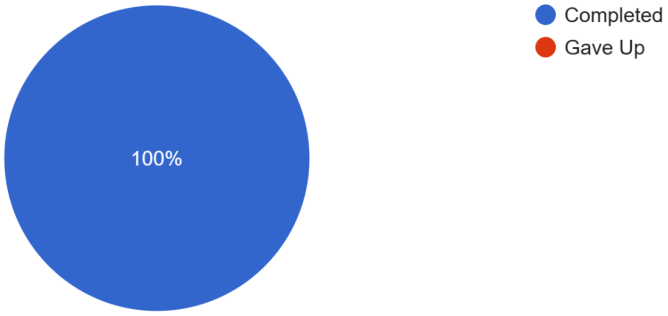
Could not access

A minute, no questions

1 minute, very easy to do

Go to the Client Dashboard. From there, please input a Project Proposal, and check an Existing Project Proposal.

6 responses



Based on the previous task, how long did it take to complete the task? Do you have any comments, questions, concerns, or suggestions for the website based on that task?

6 responses

It was unclear where I could make a new proposal

3 minutes. This was easy to do but there was a lot of inputs.

Could not submit a proposal, I could edit the existing proposals and that freaked me out.

2 minutes? I edited an existing proposal; I thought it was funny that I submitted a lot of proposals apparently.

5 minutes, no questons

1 minute