Orgifi

Design Document

Team 38

Client:

Adviser: Mai Zheng

Team Members: Hunter Barton, Perry Ports, Hongwei Wang, Tabe Ekpombang, Adin Huric, Dino Huric, Mohammed Abdelager

Team Email: sdmay25-38@iastate.edu

Team Website: https://sdmay25-38.sd.ece.iastate.edu

Revised: 12/07/2024

Executive Summary

Orgifi wants to solve a common challenge faced by individuals seeking to connect with others through clubs and small organizations. That is the lack of an efficient platform to discover, manage, and coordinate these groups. Whether someone is new to a city, looking for new hobbies, or managing a club. Traditional methods of finding and running these communities are often fragmented and inefficient. Orgifi provides a streamlined solution to these problems, catering to community engagement and simplifying organizational management

The platform is built around key design requirements to ensure usability, scalability, and functionality. These include secure user authentication, an intuitive user interface, real time messaging, event management, and seamless payment processing. The design emphasizes fast response times, compliance with General Data Protection Regulation (GDPR) , and the ability to support a growing user base.

The design uses a modern tech stack. The frontend is implemented in React and TypeScript for a clean modular interface, while the backend uses Node.js and MongoDB for robust data handling and scalable microservices. A monorepo architecture supports streamlined development across both the frontend and the backend with shared functionalities ensuring maintainability.

Significant progress has been made so far. Core functionalities such as user authentication, event scheduling, and messaging have been implemented. The database schema for users and organizations is also functional, and features like organization search and user onboarding are operational. The frontend integrates these features with a responsive design that ensures accessibility across all devices.

The platform has been tested against initial requirements, including functionality, speed, and security. They have been tested with tools like Jest and Selenium. Testing outcomes demonstrate that Orgifi meets user needs effectively by providing reliable performance, easy navigation, and a clear separation of role for admins and general users.

The next steps include optimizing the system for scalability to handle more concurrent users, enhancing search performance with indexing technologies, and improving the user experience for recruitment workflows. The platform will focus on implementing a robust payment processing for managing dues and transactions securely.

Orgifi is on its way to becoming a comprehensive community management tool, addressing real world needs with an innovative design and modern technologies.

Learning Summary

Development Standards & Practices Used

Standard practices for full-stack development:

- Frontend: Use of ReactJS for building dynamic, user-friendly interfaces.
- Backend: Node.js with Express for server-side logic.
- Database: MongoDB for efficient and scalable data storage.
- Version Control: Git and GitHub for collaboration and version tracking.
- Testing: Unit testing with Jest and integration testing with Cypress.

Engineering standards:

- Agile methodology for iterative development.
- RESTful API design principles.
- Use of secure authentication practices (e.g., OAuth, JWT).

Summary of Requirements:

- Create and manage clubs.
- Organize and manage events (single and joint events).
- Set up and track dues and subscriptions.
- Search for clubs based on interests.
- Join clubs and participate in events.
- Manage personal profiles and payment information.
- Secure user authentication.
- Real-time notifications for event updates.
- Admin interface for managing content.

Applicable Courses from Iowa State University Curriculum

- **CPRE 431**: Operating Systems Principles (user and group management, server architecture).
- **COM S 319**: Software Construction and User Interfaces (React and frontend principles).
- **SE 329**: Software Project Management (Agile development practices).
- **COM S 309**: Software Development Practices (version control and team collaboration).
- **CPRE 308**: Networking Basics (real-time notification systems and API development).

New Skills/Knowledge acquired that was not taught in courses

Implementation of **real-time chat and notifications** using WebSockets or Firebase.

Experience with **deploying applications** on cloud platforms like AWS or Heroku.

Hands-on **UI/UX design** practices for a polished user interface.

Advanced **security measures** such as data encryption and secure token handling.

Table of Contents

1. Introduction

1.1 Problem Statement

When a person is moving to a new city, looking for new hobbies, or looking to increase the size of their intramural sports team, they will soon encounter the issues of finding and managing a small organization. It is difficult for people seeking new clubs or organizations to find them in the first place, and it is difficult for those who run organizations to coordinate dues and find new members. People with niche interests may not even know if others in their area are interested in their hobbies. Orgifi aims to alleviate the frustration involved with finding and managing small groups. It will have features for those seeking and managing clubs, such as search, request for new group formation, and payment management.

1.2 Intended Users

Our platform is designed to cater to various targeted users' individual demands and characteristics. First and foremost, we picture students actively looking for extracurricular activity opportunities at their schools. These people probably need to set up the clubs because they are unfamiliar with the college setting. They require a straightforward approach to investigate, find, and gain additional knowledge about other groups or chapters that correspond with their hobbies. With the simple-to-use interface of our website, users may locate appropriate organizations and even submit a request to join by searching for them by name, location, or school. Fostering a sense of community and belonging contributes to the overall objective of raising student involvement.

Another critical group includes existing members of these organizations. These users range from newly joined members to those with more significant involvement within the club. They need tools to facilitate seamless communication and collaboration with other members, access to a shared calendar for event planning, and a way to interact socially within the club. The platform addresses these needs by offering features such as group chat, event management, and a list of other members. This streamlines communication and makes it easier for members to engage actively in organizational activities.

Finally, we think about these clubs' administration. These people oversee the daily operations, which include arranging events, arranging activities, and adding or removing members. The platform drastically reduces administrative workload by offering complete capabilities for managing recruitment contacts, permissions, and member information. These features free up administrators to spend more time on member experiences and less time on administrative tasks. By meeting their unique demands, our platform adds value to each user group, improving student life, building community, and guaranteeing smooth club operations.

Appendices

Fletcher, D., & Brookman, A. (2002). *Making joining easy: Case of an entertainment club website*. American Institute of Graphic Arts. Retrieved from https://dl.acm.org/doi/abs/10.1145/507752.507755?casa_token=VgTWy1jTrhwAAAAA:F8rjyHX8xAzwmJLnOoQ3N8z-jpFwM07mo-egp1QcKwJokNnQnP_bzoroe7BSxdnYtDkSuOxZNo_rISUk

This paper's intended users of the platform include students seeking to join campus clubs, current club members, and club administrators. Students need a simple way to explore and join organizations, enhancing their campus involvement. Members need communication tools and event coordination features to stay actively engaged. Administrators need efficient management tools to organize events, handle membership information, and coordinate recruitments. The platform addresses these needs by providing features like a search function for organizations, a shared calendar, chat options, and administrative tools. This supports the overarching goal of fostering community, improving engagement, and streamlining club operations.

2. Requirements, Constraints, And Standards

2.1 Requirements & Constraints

- Functional requirements
    - The platform must support the authentication(Sign in/up/out)
    - Users can search for organization by name, id or school
    - Users can move in and out of the organization very clearly
    - Users can check the description of organization very clearly
    - Members can see the list of their team members
    - Members should can easy to check their events very easy in calendar
    - Members can filter the events that are related to their club or not
    - Club leaders can invite or kick out member easily
    - Club leaders should be able to create events and filter it does only for members or everyone
    - Club leaders can view potential recruits and add new contracts to the recruits list
- Resource Requirement
    - Frontend must use React JS/Typescript to implemented
    - Backend service should use Typescript and Node.js for type safety
    - Preferred database technology is MongoDB
    - Monorepo structure should be used to store both frontend and backend services, separated by apps (services) and packages (common functionalities)
- Physical Requirements
    - Must support desktop and mobile browsers
    - Must hosted on a scalable server architecture
- Aesthetic Requirements
    - UI should be clean and easy to use
    - Color scheme and typography should enhance readability and usability
- User Experiential Requirements
    - The platform must provide the seamless onboarding experience for new users
    - User should easy to navigate easily between organization, calendar, chats, etc pages
    - User interaction should be fast, response time should be in 2s
    - Error message should be clear and concise
- Economic/Market Requirements
    - Platform should be scalability to support multiple organization and users
- UI Requirements
    - UI must support a responsive design for desktop, tablet and mobile.
    - Calendar and recruitment tabs should be easily accessible within the UI
    - A clear, visual hierarchy should differentiate between admin and non-admin users' permissions

- Security Requirements
    - Authentication should use secure password hashing
    - The platform must comply with GDPR regulations if used in regions where it applies (constraint)
    - User data should be encrypted during transmission (constraint)
- Quantitative Requirements
    - The platform should handle up to 10,000 concurrent users during peak times
    - Chat messages should be delivered in under 500 milliseconds
    - Search functionality should return results within 1 second

2.2 Engineering Standards

There are a lot of engineering standards likely for our project. For networking, the platform must follow IEEE 802.11 for Wi-Fi compatibility and use HTTP/HTTPS (RFC 2616/2818) to ensure secure web communication. OAuth 2.0 is essential for secure user authentication. Our platform's frontend must adhere to the HTML5 standard, ensuring compatibility across browsers and devices. The standard for data interchange format is especially relevant if we are working with APIs for front-end and back-end communication.

Q1)

The "IEEE Standards in Everyday Life" movie emphasizes engineering standards' importance to our day-to-day existence. Standards promote safety, dependability, and compatibility by ensuring that various technologies—from smartphones to healthcare systems—operate unison. Engineers and developers produce products that satisfy customer expectations, are interoperable, and enhance people's quality of life by adhering to standards. Standards also foster innovation by guaranteeing that new technologies may seamlessly connect with current systems across various industries. They offer a framework for reliable, safe, and expandable technology development.

Q2)

IEEE/ISO/IEC 12207-2017: This standard defines a framework for the entire software life cycle, covering processes, activities, and tasks needed to develop, maintain, and retire software systems. It ensures consistency in the terminology and methods used by software developers, making it easier to manage complex projects.

IEEE 829-1998: It describes the procedures for determining whether the program satisfies the requirements and performs as planned. Testing is divided into various integrity levels to ensure comprehensive validation at every stage of the project. It is necessary for verifying features like chat systems, event management, and member authentication.

IEEE 1012-1998: This is a standard related to Software Verification and Validation (V&V). Through stringent testing and validation procedures, it offers a framework to guarantee that software systems fulfill their intended requirements and specifications. The standard describes software evaluation procedures at different development lifecycle phases to ensure system integrity, quality,

and performance. It is appropriate for us since we are proposing the project, and to improve the results, we require stringent software review procedures.

Q3)

They are really important for our project, IEEE/ISO/IEC 12207-2017 is crucial for managing the software lifecycle, guiding the team through structured phases of development and maintenance of the platform. The IEEE 829-1998 applies to project's need for thorough software testing and validation, ensuring key features like authentication, chats, and event handling function properly. IEEE 1012-1998 is highly pertinent to the project, especially when using Verification and Validation (V&V) procedures to guarantee the platform's accuracy and dependability. V&V ensures that every component works as planned and satisfies criteria because the platform includes user data, authentication, and communication functions.

Q4)

We also choose IEEE 802.11 (Wi-Fi Standards), which ensures the platform works efficiently over wireless networks, which might be necessary if your platform needs to work seamlessly on mobile devices. Also we choose IEEE 1471 for System Architecture documentation, ensuring clear guidelines and structure for system components, useful for large-scale microservice architectures.

Q5)

We will use a defined software life cycle for distinct project phases to integrate the chosen standards, guaranteeing systematic development and maintenance. By incorporating thorough testing at each level, we will improve verification and validation and ensure all features function as intended. Test documentation will also be kept up to date to document test outcomes, monitor problems, and confirm functionality such as event handling and authentication. These changes will improve the project's quality, dependability, and adherence to industry standards.

3 Project Plan

3.1 Project Management/Tracking Procedures

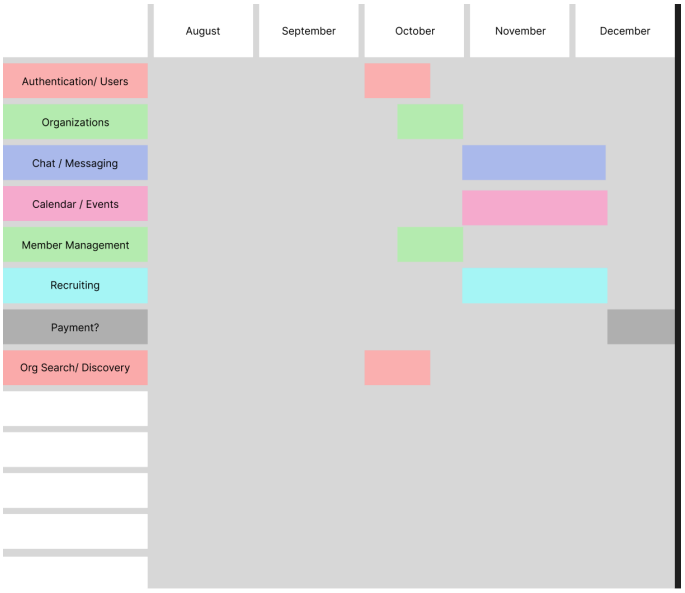
Our team will use an agile project management style. We decided to use this style because as a software project we are going to frequently encounter implementation challenges that may require us to modify our goals. Additionally, many of our goals may have dependencies that other team members are waiting on, so it is important to get constant feedback from other team members to catch blocks before they result in a large amount of wasted time. Since our project is a web application it is also never "done". We may have a version 1.0, but we can always add more features to improve it. By using agile we are able to better fit our project management style to our project and constantly be able to plan out new features.

3.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Software version 1.0 schedule:

1. Authentication
2. User database entries
3. Organization database entries
4. Organization search
5. Organization member add/delete
6. Organization calendar/events
7. Organization recruiting
8. User request organization
9. Organization chat
10. Payment processing



Our sprints will revolve around these goals. Each of these features involves changing the database or adding a new microservice into our application. These subtasks also have subtasks that constitute them as well, so the purpose of each week's sprint is to break down our high level design, engineer a solution, and assign tasks to various group members to implement each goal. Each sprint may require us to learn a new technology or to slightly modify our plans for the project to account for new information, but our main features here will be implemented in the end.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. **Project Setup:** Design the database and tables for all the milestones in section 3.2 and set up the mono repo.
    - **Metric:** 75% of the database schema has been designed, and we plan on going to 90% when we determine how we will be handling permissions.
2. **Authentication:** Users will be able to sign up, log in, reset passwords when necessary, and verify their role in a club.
    - **Metric:** Safe and successful signup/login of users under 2 seconds.
3. **User database entries:** Have a database that stores user information securely and validates their data correctly.
    - **Metric:** 100% validation of user entries like names, passwords, and email addresses.
4. **Organization search:** Design a functional search function that users use to look up clubs they want to join. Will also have a category feature.
    - **Metric:** 90% of searches will provide the desired output in under 2 seconds.
5. **Organization member add/delete:** Implement member management to be fully functional.
    - **Metric:** 90%+ success rate for adding and deleting members.
6. **Organization calendar/events:** Create a system for event management and a calendar where these events can be tracked.
    - **Metric:** 100% of events should appear accurately on the calendar.
7. **Organization recruiting:** Design and implement the organization recruitment process and feature.
    - **Metric:** 90% of users should be able to submit requests to join clubs and should be aware of the status of the requests.
8. **User request organization:** Design and create a feature that allows users to request to join an organization.
    - **Metric:** 95% of requests should be able to be processed safely and successfully.
9. **Organization chat:** Design and implement a messaging feature that allows communication between club members and also between executive members of other clubs.
    - **Metric:** Messaging should be at least 90% accurate.
10. **Payment processing:** Design and implement a system that allows users to pay their dues and allows tracking of the payments made.
    - **Metric:** Have at least 95% of successful transactions.

We will measure progress on these tasks by following strict deadlines and communicating about any issues encountered along the way so as to overcome these.

3.4 Project Timeline/Schedule

The timeline begins with essential tasks such as authentication and organization setup and progresses through communication, event management, and recruitment functions. Optional payment processing is kept for last, allowing the team to focus on key features. The chart successfully depicts Agile planning, with overlapping activities and sprints aimed at integrating and improving core functionalities over time.

3.5 Risks And Risk Management/Mitigation

- Authentication
    - **RISK:** Security Vulnerabilities: Probability = 0.6. *Mitigation*: Use a reliable authentication provider like Autho or Firebase Auth to handle security. Regularly update libraries and apply industry-standard security protocols (e.g., 2FA).
    - **Alternative Options:** By offering safe, scalable solutions, managed authentication (Autho, Firebase) may reduce risk, albeit possibly at a recurring expense.
- User Database Entries
    - **RISK:** Data Integrity Issues: Probability = 0.5. Mitigation: Use MongoDB to enforce schema validation, putting special restrictions on fields like login and email.
    - **Alternative Options:** While using a managed database with built-in scalability and data validation features may improve reliability, it may also raise running expenses.
- Organization database entries
    - **RISK:** Data Consistency Across Services: Probability = 0.6, Mitigation: To minimize read/write inconsistencies, establish a caching layer and specify a centralized service for organizational data.
    - **Alternative Options:** Employ a NoSQL database that has relationship support built in, or think about using an off-the-shelf platform that facilitates organization administration.
- Organization Search
    - **RISK:** Search Performance: Probability = 0.7. Mitigation: Test with real data early on and use an indexing technology like Elasticsearch for effective, scalable search capabilities.
    - **Alternative Options:** Despite the potential cost, think about using Elasticsearch as an add-on to increase search accuracy and speed in big datasets.
- Organization Member Add/Delete
    - **RISK:** Permission Management Complexity: Probability = 0.6. Mitigation: Clearly specify permissions and roles inside the codebase. Test role assignments with varying permissions by utilizing role-based access control libraries.
    - **Alternative Options:** For a possible ongoing fee, employ a third-party user management system to make this process simpler.
- Organization Calendar/Events
    - **RISK:** Event Synchronization: Probability = 0.5. Mitigation: Address event management and synchronization problems by using a third-party API. This lowers internal complexity.
    - **Alternative Options:** Although using an established calendar service (like Google Calendar) would lower risk, there may be fees associated with using the API.
- Organization Recruiting
    - **RISK:** Privacy and Data Compliance: Probability = 0.5. Mitigation: Limit stored personal data and put consent procedures in place to guarantee adherence to data privacy requirements.
    - **Alternative Options:** Consider using a third-party hiring tool that is compatible with the platform to cut down on development time and improve security.
- User Request Organization
    - **RISK:** Workflow Complexity: Probability = 0.4. Mitigation: Maintain a straightforward join request procedure. Establish request status monitoring and keep things simple by avoiding having several pending statuses.
    - **Alternative Options:** Think about outsourcing to an external CRM product with integration capabilities if this feature is too sophisticated.
- Organization Chat

    - **RISK:** Real-Time Performance and Message Consistency: Probability = 0.7. Mitigation: To prevent duplicates, use WebSocket-based libraries such as Socket.IO and provide message acknowledgment. Early on, test the chat feature while it's loading.
    - **Alternative Options:** For messaging, use a managed service like Pusher or Firebase Realtime Database, which offers scalability and dependability at a possible cost.
- Payment Processing
    - **RISK:** Compliance with Payment Regulations: Probability = 0.8. Mitigation: To manage transactions safely and in accordance with PCI requirements, use a third-party payment processor (Stripe, Square).
    - **Alternative Options:** Using a third-party provider like as Stripe simplifies compliance, lowers the risk of handling sensitive data, and offers safe off-the-shelf payment options.

3.6 Personnel Effort Requirements

Task	**Hours**	**Description**
Authentication	10	Implement authentication with provider integration (e.g., Autho/Firebase), including setup and testing.
User Database Entries	10	Define user schema, implement CRUD operations, and ensure validation and data integrity.
Organization Database Entries	10	Define organization schema, relationships with users, and ensure data consistency across microservices.
Search	30	Implement efficient search functionality, including indexing and testing with sample data.
Organization Member Add/Delete	20	Implement role-based access control (RBAC) and ensure permissions are correctly managed and tested.
Organization Calendar/Events	35	Build calendar integration and event CRUD operations; requires additional research and API integration.
Organization Recruiting	15	Develop logic for organization

		recruitment workflows and necessary database changes.
User Request	10	Create feature for users to request joining organizations, including notifications and response handling.
Organization Chat	40	Set up real-time chat, likely with WebSockets, and test concurrency and message persistence.

3.7 Other Resource Requirements

We might need a device like Nexus Google Device to run our app or a similar device that can be used to demo our project. Also, we might need a 32-inch or larger screen to display our final presentation slides.

4 Design

4.1 Design Context

4.1.1 Broader Context

Our project focuses on addressing the challenges of managing and engaging communities by providing a platform that simplifies club discovery, event organization, and communication. It is designed for students, organizations, and just anyone who wants to find a community of people with similar interests. By fostering better connectivity, reducing, and increasing participation, our project helps build stronger, more inclusive communities. This platform ultimately meets societal needs by improving collaboration and providing opportunities for growth and shared experiences.

Area	**Description**	**Examples**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

4.1.2 Prior Work/Solutions

Include relevant background/literature review for the project (cite at least 3 references for literature review in IEEE Format. See link: https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf)

The need for platforms to manage clubs, organizations, and events has grown as community engagement increasingly relies on digital solutions. Our project builds on existing solutions while addressing specific gaps identified in the literature and similar platforms.

[1] A. J. Coleman, "Student organization engagement: The role of leadership in fostering student retention," *Journal of Student Affairs Research and Practice*, vol. 54, no. 1, pp. 71–84, 2017

[2] CampusGroups, "Engage your students, faculty, staff, and alumni," *CampusGroups*. [Online]. Available: https://www.campusgroups.com/.

[3] N. Dabbagh and A. Kitsantas, "Personal Learning Environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning," *The Internet and Higher Education*, vol. 15, no. 1, pp. 3–8, Jan. 2012.

Several platforms like **Slack**, **Discord, and school club websites** cater to community management. **Slack** and **Discord** focus on communication but lack features like event and membership management. School club websites are usually constrained to certain schools or geographic areas.

These solutions address individual aspects but fail to integrate all functionalities seamlessly. **Orgifi** differentiates itself by offering an affordable, all-in-one platform that combines event management, membership tracking, and communication tools, specifically tailored for clubs and organizations.

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available. Thus, provide a list of pros and cons of your target solution compared to all other related products/systems.

Pros

- Affordable pricing tailored for clubs.
- Integrated event management, communication, and membership tracking.
- Accessible to all clubs and organizations, not limited to specific schools.
- User-friendly and customizable for club needs.

Cons

- May lack some advanced features offered by more specialized platforms.
- May have fewer integrations with third-party tools compared to larger platforms.

4.1.3 Technical Complexity

Provide evidence that your project is of sufficient technical complexity. Use the following metric or argue for one of your own. Justify your statements (e.g., list the components/subsystems and describe the applicable scientific, mathematical, or engineering principles)

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles –AND–
2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

4.2 Design Exploration

4.2.1 Design Decisions

Three key design decisions that we have made are:

Authentication: We decided to use Autho for authentication as it is easy to implement and essential for user security.

Database Selection: We chose to use MongoDB because of its flexible data handling as well as Firebase cloud storage for images.

Real-time Communication: We are using websockets for our chat function which will ensure real time updates.

4.2.2 Ideation

For the database selection, we considered several options such as:

- MongoDB
- MySql
- Firebase
- DynamoDB
- PostgreSQL

4.2.3 Decision-Making and Trade-Off

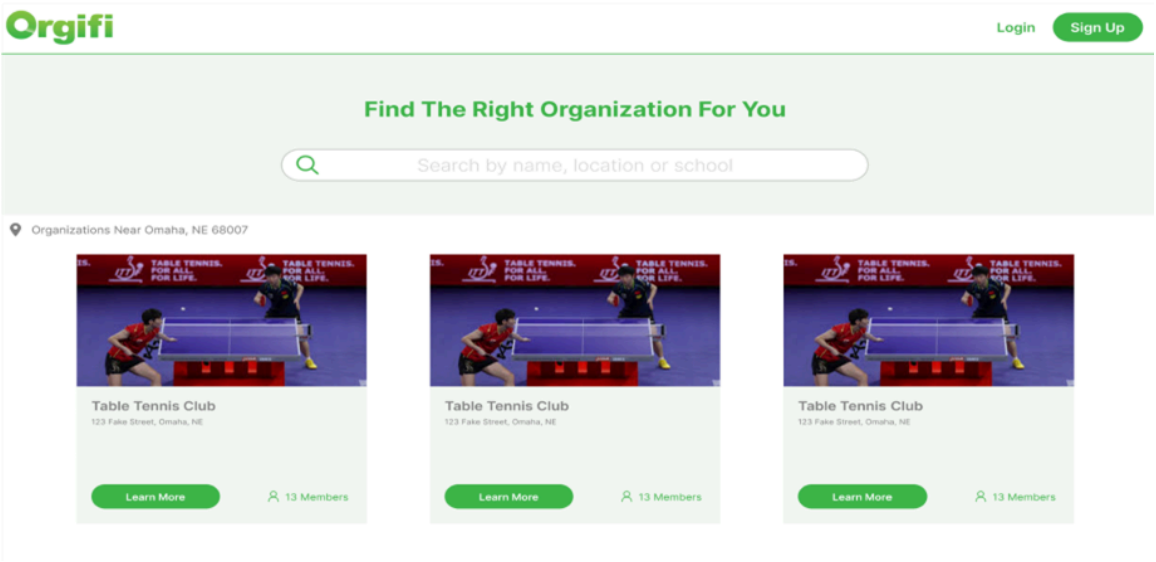
We decided to use MongoDB due to its adaptability with Orgifis data needs despite being slightly less structured than a SQL based database. For images we decided to use Firebase cloud storage due to its scalability.

4.3 Proposed Design

4.3.1 Overview

We are implementing a website that people can use to start or join communities of people that share similar interests.

- It will contain a user-friendly design user interface for users to be able to easily navigate through.
- The backend, behind the scenes work will be able to securely store and handle the user operations.
- Chat feature where club members can easily chat.
- Calendar where they can see club events and schedules.
- Organizations search where they can easily search for clubs they want to join.



4.3.2 Detailed Design and Visual(s)

Front End (Next.js): Will handle user interactions and communications with backend services via APIs. It will manage user interfaces for browsing clubs, events, managing profiles, etc.

Auth Service (Autho): Provides authentication and authorization securing access to the system.

Image Storage: Used for storing all media files like club logos and event images. This is separate from the main data storage to improve performance and scalability.

Microservices Layer (Node.js):

- Recruitment: Manages the recruitment process with club like applications and approvals.
- Calendar: Handles scheduling and tracking events integrating with user calendars if needed.
- Finances: Manages dues, subscriptions, and payments.
- Chat: Supports communication with club members between users.
- Organization: Manages club's details and settings, allows club admins to customize their organizations profile.
- Membership/Permissions: Controls user's permissions, managing roles within clubs and making sure the access is secure.
- Users: Maintains user profile data and links users with clubs they have joined.

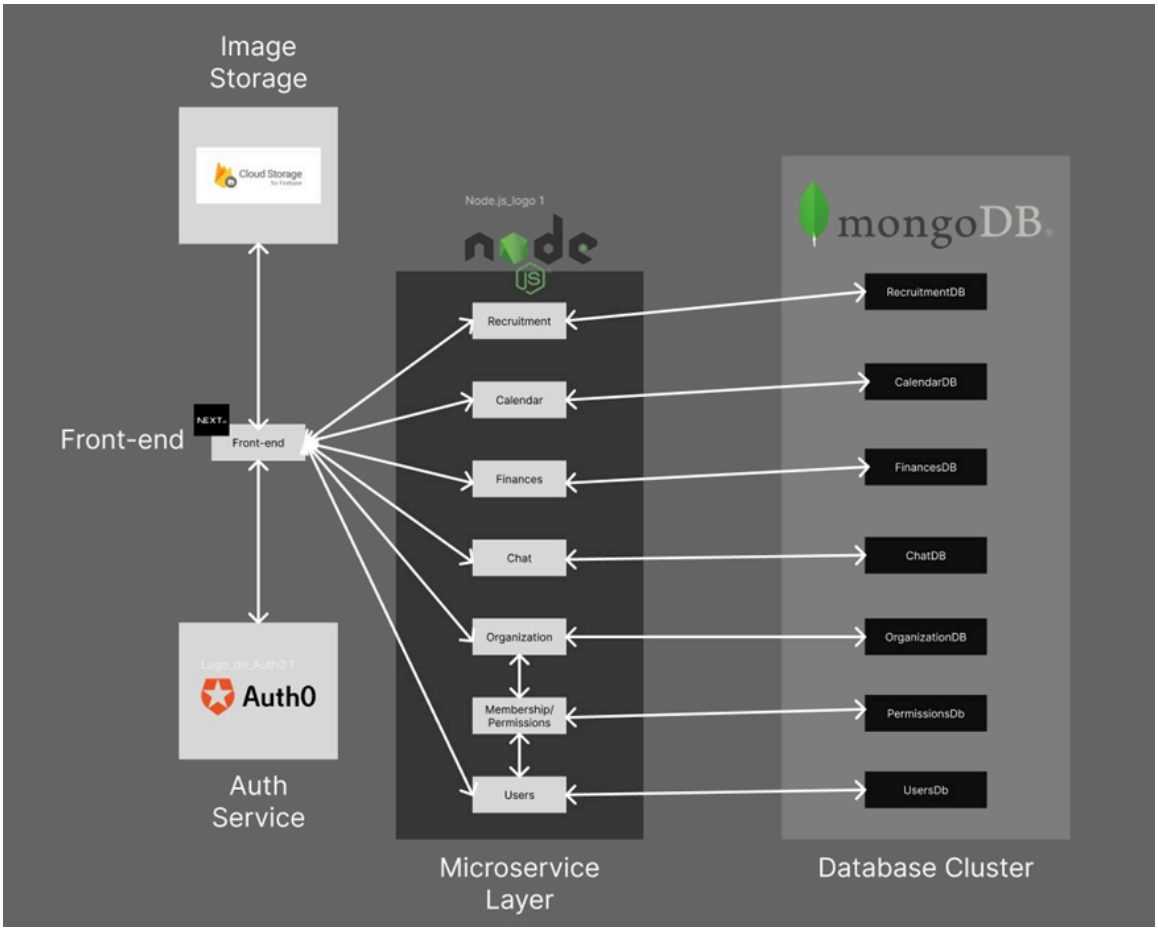


Image Storage
Cloud Storage
Front-end
NEXT.
Front-end
Auth0
Auth Service
Node.js_logo 1
Recruitment
Calendar
Finances
Chat
Organization
Membership/ Permissions
Users
Microservice Layer
mongoDB.
RecruitmentDB
CalendarDB
FinancesDB
ChatDB
OrganizationDB
PermissionsDb
UsersDb
Database Cluster

4.3.3 Functionality

4.3.4 Areas of Concern and Development

The current design of Orgifi is well structured to satisfy the core requirements, by offering features that allow club owners to create communities, events, and memberships. For students it offers an intuitive interface for discovering clubs, joining events, and managing subscriptions. This helps users meet their needs for easy access to campus events and activities.

Based on your current design, what are your primary concerns for delivering a product/system that addresses requirements and meets user and client needs?

Performance and scalability, as the user base grows the platform must handle the increased load especially during peak times like the beginning of the semester. Ensuring the system stays responsive under high traffic is crucial.

Data Security and Privacy: Since Orgifi will handle sensitive information like. personal data and payment data. We must ensure robust security practices as it is vital to protect the user's information.

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

Are there specific security standards or compliance requirements that Orgifi must adhere to?

Does our microservices approach align with best practices, or are there alternative structures that would enhance reliability and scalability?

4.4 Technology Considerations

Frontend: For the front end, we will be utilizing Next.js.

- **Strength:** Node js improves page load time.
- **Weakness:** Build times may increase significantly if the project gets too large.
- **Tradeoff:** We chose next js over react as it has better api routes.
- **Solution:** We could implement code splitting to reduce size.
- **Alternative**: We could use react instead of next.js.

Backend: We will be using Typescript with Node js for the backend

- **Strength:** Node js contains a lot of libraries that will help us be able to create multiple features easily.
- **Weakness:** Managing typescript might be hard and increase our code base size.
- **Tradeoff.:** We chose typescript over JavaScript as it is easier to manage.
- **Solution:** Modularize the code so that it is very well structured.
- **Alternative:** Utilize Django as it is also very secure.

Database: We will be using MongoDB for our database.

- **Strength:** MongoDB can handle large amounts of data sets.
- **Weakness:** Data redundancy.
- **Tradeoff:** MongoDB has better database management than firebase.
- **Solution:** Use Mongoose to avoid the duplication of data.
- **Alternative:** Utilize Firebase as it is also safe and reliable.

4.5 Design Analysis

- **Backend of Organization:** So far, this feature has been implemented. The proposed design works and the next with this feature will be to debug and test it out and make sure that it is reliable.
- **Frontend of Organization:** So far, this feature has been implemented. The proposed design works and the next with this feature will be to debug and test it out and make sure that it is reliable.
- **Chat feature:** So far, this feature is still being implemented. The proposed design works and the next with this feature will be to debug and test it out and make sure that it is reliable.
- **Authentication:** So far, this feature has been implemented and tested. The proposed design works and the next with this feature will be to ensure user data security test and make sure that it is reliable.
- **Calendar feature:** So far, this feature is still being implemented. The proposed design works and the next with this feature will be to debug and test it out and make sure that it is reliable.

5 Testing

5.1 Unit Testing

All of Orgifi's main features will be tested using Jest in Javascript. Jest is a unit testing framework that will allow regression testing of the main features, so that new features can be added without breaking the old features. There will be a test set for each main feature including but not limited to: chat, the organization page, the search page, and the calendar. In addition to the main features test suite, a database test file will be created to ensure that the MongoDb database works as expected. Finally, the website's frontend will be tested using Selenium in order to ensure that the user facing interface functions correctly.

5.2 Interface Testing

There are three main testable interfaces present in this application: the UI, the API, and the database. These sections will be tested individually by testing all user facing components after other tests confirm that the inner components are functioning correctly. The API will be tested using Postman to confirm that all basic functionality works.

5.3 Integration Testing

The goal of integration testing for Orgifi is to test the full path from user>API>database>API>user. Getting this path to work correctly will be the most difficult task of the project. Since we can't construct this system without first constructing its constituent components we will need to test each component to ensure that they function correctly. Before making them we will have specifications to ensure that each component when completed will be able to connect to the other components. After these are completed we will connect them together and write test files to ensure that these components function together.

5.4 System Testing

Our system will be tested as a whole by ignoring all internal tests and instead focusing on everything that the user would interact with. To accomplish this goal we pick specific tests from our test suite that focus on HTTP requests and UI interactions. We will move through our list of required features and write tests in Postman and Selenium that test the functionality.

5.5 Regression Testing

Our test suite will be run on all of our main components each and every time a new major or minor feature is added. We will use Jest and Selenium tests to ensure that no new feature breaks old functionality. It is critical that once a feature is added no other components features interfere with it. We aim to have a high degree of separation between components to facilitate future modifications.

5.6 Acceptance Testing

After a major feature is implemented we shall look at our list of requirements and confirm that it completes its required goal. Since our client is our professors we will confirm that our project is up to par by demonstrating progress to our advisor bi-weekly and to our professors twice a semester. These demonstrations will confirm that our project's functional requirements work as intended. Once our functional requirements work we will begin testing non-functional requirements such as

security, reliability, and scalability. Our professors and advisors will have opportunities to tell us what we should improve on several times a semester, so we should have many opportunities to pivot if need be.

5.7 Results

Currently only manual testing has been performed to ensure that the organization, database, and authentication systems work. Next we are going to write tests for each of the main components in our project to ensure that they function correctly. Since each team member works on a separate feature we will merge these features together shortly and test them together will separate integration tests. Finally we will need to write API tests and UI tests. The API tests will be written incrementally as new endpoints are added. UI tests will be written as more of the frontend is created and finalized.

6 Implementation

Frontend Implementation

- **Organization Page**:
  - Developed a responsive and intuitive interface to manage organizational details, such as name, logo, category, and location.
  - Integrated search functionality to help users locate specific organizations quickly.
  - Utilized reusable components in Next.js and TypeScript for modular and maintainable code.
- **Calendar System**:
  - Created a calendar view where users can schedule and view events for their organizations.
  - Implemented navigation between different time frames (e.g., weekly, monthly) and enhanced accessibility features.
- **Messaging Interface**:
  - Designed a chat interface to facilitate real-time communication among organization members.
  - Focused on usability by integrating message previews and notifications for incoming messages.

Backend Implementation

- **Authentication Service**:
  - Developed a user authentication system using Node.js and MongoDB.
  - Implemented user registration and login functionality with JWT-based token authentication.
  - Enforced secure password storage.
- **User Database**
  - Set up a MongoDB database for storing user details, including user IDs, roles, and permissions.
  - Integrated efficient mechanisms to retrieve and update user information dynamically.
- **Backend for Frontend**
  - Built APIs for managing the frontend functionalities of the organization, calendars, and messages.
  - Ensured consistency across services by using TypeScript and defining common types in a shared monorepo.

.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

The area that our team is performing well is honesty. Throughout the project, the team has demonstrated exceptional performance in honesty, which is a fundamental component for upholding responsibility and trust. The team has prioritized open communication from the start, making sure that all parties involved are aware of the project's status, obstacles, and reasonable expectations. This strategy is in line with the IEEE Code of Ethics, which strongly emphasizes being accurate and reasonable when stating opinions or offering estimates. Frequent team meetings and progress reports have greatly aided the development of this honest culture. The team has continuously offered data-driven updates, openly discussed technical challenges, and given reasonable delivery deadlines. For instance, the team promptly notified the adviser of any delays in creating a particular feature, explained the underlying reasons, and suggested an updated timeframe. Trust between team members and outside partners has increased due to this transparency. The group has also demonstrated a dedication to moral behavior by refraining from overstating or promising too much. Instead, they have concentrated on producing work that aligns with the project's goals and technological viability. By placing a high value on integrity, the group has fostered an atmosphere of respect and cooperation, keeping everyone informed and working together to accomplish shared objectives morally and practically soundly.

The area that we still improve is when the team increases their professional skills, especially in Node.js and MongoDB. Although the team wants to deliver high-quality work, it will face many obstacles because we sincerely try to learn these important areas. We have experienced delays in the project due or stuck on some progress because we need more knowledge to use these technologies. Team members still rely on online resources such as tutorials, manuals, and forums for self-directed learning. While this approach has been beneficial, it has not yet reached expected progress, and significant technical gaps remain. To improve it, the team needs to find a more cohesive way to fix the problem together. Having clear technical learning objectives and a timeline for achieving them, the team can ensure that each member develops the necessary skills to contribute effectively.

7.2 Four Principles

Context Area	**Beneficence**	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Environment	Promotes sustainability through reusable components.	Consumes energy/resources during development	Allows organizations to opt for greener practices through tools.	Ensures equitable resource usage and impacts.
Social	Builds strong user connections via chat and events.	Potential for misuse of the platform for harassment.	Users have control over joining and interacting in organizations.	Equal access for all users without discrimination.

Economic	Provides free access to organization tools	Maintenance could incur financial costs for admins.	Organizations can freely choose whether to adopt the platform.	Ensures all users benefit, regardless of size.
Culture	Fosters inclusion by supporting diverse organizations.	Could inadvertently marginalize underrepresented groups.	Members decide how to use the platform to suit cultural needs.	Supports diverse use cases without bias.

One critical context-principle pair for our project is **Social - Beneficence**. Our platform is designed to enhance social connections by enabling users to join organizations, engage in discussions, and participate in shared activities. This fosters a sense of belonging and collaboration within communities. To ensure we achieve this benefit, we are focusing on creating intuitive user interfaces, seamless communication tools, and accessible features that encourage meaningful interactions.

Our project is lacking in the **Environment - Nonmaleficence** area, as the platform's development and hosting consume significant energy and computational resources, potentially contributing to a higher carbon footprint.

7.3 Virtues

- **Collaboration**: The ability to work together harmoniously, leveraging each member's strengths to achieve shared goals.
    - Team Action: regular team meetings, clear task assignments, and encouraging open communication and idea-sharing.
- **Accountability:** Taking responsibility for one's actions and fulfilling commitments to the team.
    - Team Actions: Establishing deadlines, progress tracking, and addressing challenges as a group.
- **Respect**: Treating every team member with dignity, acknowledging their contributions, and valuing diverse ideas.
    - Team Action: Creating a safe space for opinions, resolving conflicts amicably, and ensuring equitable participation.

- **Hongwei Wang**: I think the respect I have demonstrated in the senior design. Respect is fundamental when doing the project. We should respect everyone and not discriminate against each member. Also, respect the member's work outcomes; we cannot plagiarize and vilify members ' teamwork. I have respected all the team members and their work. We never have any arguments with each member. The virtue that I did not perform well is accountability. It is important because it relates to our project outcome. We should do anything before the deadline to ensure the project succeeds. I have not performed well

because there is still a need to gain knowledge that will stick to the progress of our project. It makes me unable to finish the project on time sometimes.

- **Tabe Ekpombang**: Collaboration is a virtue I have greatly demonstrated during this project. Collaboration is key to success in a team environment. It allows for sharing of ideas, skills, and expertise, leading to better problem-solving. I've worked closely with my team members, ensuring clear communication between backend and frontend and making sure everyone has a voice. The virtue I have demonstrated least in is accountability. Accountability ensures that you are responsible for your tasks and commitments, contributing to trust and efficiency within the team. I will take a more proactive approach in ensuring deadlines are met, providing regular updates, and being more transparent about any challenges I face during the development process.

- **Perry Ports:** Accountability is an important part of working with a team because it allows members to acknowledge when they have made a mistake. I think this is the virtue I would most like to work on by meeting deadlines that are set and by talking with team members how I could improve my work. I think I have best exemplified the virtue of collaboration in this project by setting up team meetings and trying to keep on top of our assignments. I hope to keep collaborating well in the future while improving other aspects of my ability to work with my team.

- **Mohammed Abdalgader:** In my senior design work, the virtue that I have consistently demonstrated is respect. Respect is a foundational value when working collaboratively on any project, and I believe it is essential for maintaining a positive and productive team environment. In a diverse team setting, respect means treating every team member with dignity and recognizing their contributions, regardless of differences in background, skill levels, or opinions. It also entails respecting each member's work, ensuring that their efforts are acknowledged and that their intellectual property is protected. On the other hand, a virtue that I have not demonstrated as effectively is accountability. Accountability is critical because it directly impacts the quality and timeliness of our project deliverables. Being accountable means taking responsibility for assigned tasks, meeting deadlines, and ensuring that each contribution helps move the project forward. This virtue is essential because delays or lapses in accountability can hinder the team's overall progress and success.

- **Adin Huric:** Empathy focuses on understanding and respecting the perspectives and needs of team members, supported by open communication and a willingness to offer assistance when challenges arise. Creativity emphasizes generating innovative ideas and solutions, encouraged through brainstorming sessions and fostering a safe space for experimenting with new approaches. Reliability ensures consistent performance and trustworthiness, reinforced by meeting deadlines, being prepared, and maintaining accountability. Individually, empathy has been demonstrated by actively listening to teammates and

offering help during difficult tasks, while creativity has not yet been fully utilized but will be addressed by contributing original ideas during brainstorming and exploring alternative solutions to technical challenges.

- **Dino Huric:** Being honest and adhering to high moral standards are key components of integrity, which is reinforced by open communication, moral decision-making, and transparency in all endeavors. The ability to adapt to new situations and challenges is emphasized by adaptability, which is developed by promoting flexible thinking, problem-solving, and the acquisition of new skills when needed. Meeting deadlines, staying focused, and displaying a strong will to accomplish common objectives are all signs of commitment to the project and team. While flexibility has not yet been fully expressed, it will be addressed by adopting new tools and approaches and remaining receptive to input in order to overcome barriers. Individually, integrity has been demonstrated by giving honest updates and guaranteeing ethical procedures.

- **Hunter Barton:** Collaboration involves working effectively as a team by leveraging each member's strengths, supported by regular meetings, task delegation, and open communication. Accountability means taking responsibility for tasks and commitments, with the team using progress tracking, weekly updates, and collaborative problem-solving. Respect focuses on valuing contributions and fostering inclusivity, achieved through equal input opportunities, constructive feedback, and conflict resolution. Individually, collaboration has been demonstrated by actively participating, supporting teammates, and meeting deadlines, while initiative has not yet been demonstrated but will be addressed by proposing improvements, taking additional responsibilities, and exploring new tools.

8 Closing Material

8.1 Conclusion

Summary

So far, we have made progress in developing both the frontend and backend of our project. For the frontend, we have developed the organization page, messaging feature, and calendar feature. On the backend, we have developed the databases for users, organizations, chat functionality, calendar, and user authentication.

Goals

1. **Develop Community Platform**: The main goal of our project is to develop a community platform that enables users to create and manage communities through clubs and organizations, where they can easily manage events and communicate with each other.
2. **Frontend and Backend Integration:** We want to ensure that this platform has a seamless integration between the frontend and backend to ensure a smooth user experience and have the user's data being handled securely.

Plan of Action

1. **Key Development**: Develop key components such as user authentication, organization management, and messaging first, then build upon them as the project progresses.
2. **Microservices Development**: Implementing microservices for backend services to ensure that the backend can efficiently support the specific needs of the frontend.
3. **Seamless User Experience**: Ensuring a seamless user interface and that backend services deliver data efficiently to maximize efficiency.

8.2 References

Technical References and related work

[1] F. T. Ulrich and M. Pfeifer, *Building Microservices with Node.js and MongoDB: A Hands-on Approach*, 2nd ed. Packt Publishing, 2021.

[2] S. Sockin and M. Piotrowski, "Design and Implementation of a Secure Authentication Mechanism Using JSON Web Tokens," *IEEE Access*, vol. 8, pp. 182532-182540, Oct. 2020.

Market Survey References

[1] P. K. Nayak and V. Agarwal, "Survey on Event Management Platforms and Scheduling Tools," *Proceedings of the 2021 IEEE Conference on Innovations in Software Engineering (ICISE)*, Hyderabad, India, Feb. 2021, pp. 231-239.

[2] R. McLellan, "Digital Community Platforms: Trends and Market Insights 2023," *Global Market Research Institute*, 2023. [Online]. Available: https://www.gmri.com/digital-community-platforms.

8.3 Appendices

Below is the link to the setup guide for our monorepo:

Link: https://drive.google.com/file/d/1QIvkWNrqV66K-2RBva4mdV2uaP3tkjoY/view?usp=sharing

Summary of the idea and expected results of Orgifi

1

FRESHMAN YEAR

Bored, but looking for something to do.

2

DECISION TO FIND A CLUB

Wants to find a club.

3

USING ORGIFI

Explores the app & features. System suggests popular clubs

4

FINDING A CLUB

System Confirms and saves preferences.

5

JOINING ANOTHER CLUB

6

HAPPY WITH RESULTS

System sends event updates and reminders

9 Team

9.1 Team Members

- Adin Huric
- Dino Huric
- Tabe Ekpombang
- Perry Ports
- Hongwei Wang
- Mohammed Abdelgader
- Hunter Barton

9.2 Required Skill Sets for Your Project

- Backend Development.
- UX/UI Design
- Frontend Development
- Project Management
- Testing
- Version Control(Git)
- Effective Communication

9.3 Skill Sets covered by the Team

- Adin Huric - Backend.
- Dino Huric- Frontend.
- Tabe Ekpombang - Backend.
- Perry Ports - Backend.
- Hongwei Wang- Frontend, Communication with Advisor.
- Hunter Barton - UX/UI Design, Backend.
- Mohammed Abdalgader - Backend, Frontend.

9.4 Project Management Style Adopted by the team

The team has adopted an **Agile** project management style to allow flexibility and iterative development, ensuring continuous improvement and adaptation

9.5 Initial Project Management Roles

- Adin Huric - Backend Developer
- Dino Huric- Frontend Developer
- Tabe Ekpombang - Project Manager
- Perry Ports - Project Manager
- Hongwei Wang- Communication Lead
- Hunter Barton - UX/UI Design Lead
- Mohammed Abdalgader - Full stack developer

9.6 Team Contract

Team Members:

1) Adin Huric 2) Perry Ports

3) Tabe Ekpombang 4) Dino Huric

5) Hongwei Wang 6) Mohammed Abdelgader

7) Hunter Barton

Team Procedures

1. **Day, time, and location for regular team meetings:** Weekly meetings via on Discord every Wednesday at 5pm and in-person meetings when necessary.

2. **Preferred method of communication updates, reminders, issues, and scheduling** : Discord

3. **Decision-making policy:** Majority vote.

Participation Expectations

1. **Expected individual attendance, punctuality, and participation at all team meetings:** All team members are expected to attend discord meetings unless a circumstance arises. Advanced notice must be provided.

2. **Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:** Each member is expected to do their assigned tasks on time and communicate if there are any issues in doing so.

3. **Expected level of communication with other team members:** Open communication is required and each team member must provide updates on their progress.

4. **Expected level of commitment to team decisions and tasks**: Team members must respect group decisions to ensure unity.

Leadership

1. **Leadership roles:**

- Tabe Ekpombang: Individual component design and testing, Team organization.

- Adin Huric: Individual component design and testing.

- Dino Huric: Individual component design and testing.

- Mohammed Abdelager: Individual component design and testing.

- Hunter Barton: Individual component design and testing.

- Perry Ports: Individual component design and testing, team organization.

- Hongwei Wang: Client/Advisor Interactionw, Individual component design and testing.

2. **Strategies for supporting and guiding the work of all team members:** Regular check-ins to ensure progress and address any issues or obstacles.

3. **Strategies for recognizing the contributions of all team members:** Acknowledge individual contributions during meetings as well as using Discord to highlight any milestone achieved by team members.

Collaboration and Inclusion

1. **Skills and Expertise:**

- Adin Huric: Backend expertise

- Dino Huric: Frontend expertise.

- Tabe Ekpombang: Backend expertise

- Perry Ports: Backend expertise

- Mohammed Abdalgader : Full stack expertise

- Hongwei Wang: Frontend expertise.

- Hunter Barton : Full stack expertise and project innovation.

2. **Strategies for encouraging and supporting contributions and ideas from all team members**: Foster inclusivity, encourage participation, provide feedback, and celebrate ideas to support team contributions.

3. **Procedures for identifying and resolving collaboration or inclusion issues:** Goal-Setting, Planning, and Execution

1. **Team goals for this semester**: Lay the groundwork for the project, ensure steady progress, and prepare for full implementation by next semester's end.

2. **Strategies for planning and assigning individual and team work:** Divide tasks based on skills and interests, set clear deadlines, and use tools like Trello or Asana for tracking.

3. **Strategies for keeping on task**: Hold regular check-ins, set milestones, and use reminders to ensure accountability and steady progress.

Consequences for Not Adhering to Team Contract

1. **Handling infractions of any of the obligations of this team contract:** Address infractions through open discussion, clarify expectations, and provide an opportunity for improvement.

2. **What will your team do if the infractions continue:** Escalate to involve the instructor or supervisor and reassign responsibilities as needed to ensure project progress.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Adin Huric DATE 12/6/2024

2) Tabe Ekpombang DATE 12/7/2024

3) Perry Ports DATE 12/7/2024

4) Hongwei Wang DATE 12/7/2024

5) Dino Huric DATE 12/7/2024

6) Mohammed Abdelgader DATE 12/7/2024

7) Hunter Barton DATE 12/7/2024