CEN 4010 Principles of Software Engineering,

Spring and 2023

Milestone 3

Project Proposal and High-level description

Team name: T.E.A.M

Project name: ModuLearn

Group 5

Names and Emails

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ethan Curtis - [ecurtis2019@fau.edu](mailto:ecurtis2019@fau.edu)

Kevin Horta - [hortak2022@fau.edu](mailto:hortak2022@fau.edu)

Bryan Cooke - [bcooke2020@fau.edu](mailto:bcooke2020@fau.edu)

Marco Parucho - [mparucho2021@fau.edu](mailto:mparucho2021@fau.edu)

Oliver Pennanen - [opennanen2018@fau.edu](mailto:opennanen2018@fau.edu)

History Revision dates

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Created: 3/27/2023

Edited: 3/31/2023

Executive Summary

**Product Name: ModuLearn**

ModuLearn is an e-learning website that allows for a unique, modular approach. The new modular design allows for a higher level of customizability and personalization. Each module can be created by institutions, teachers, and learning providers, and allow for students to understand different concepts without all the unnecessary baggage of other online learning platforms. The platform allows for students and teachers to customize their learning and teaching experience in an easy to use system.

With the new modular system we developed, students can learn at their own pace. Without the pressure of deadlines or assignments, the student can focus on learning what they are interested in. Students can explore, search, and select various different topics and select which they want to learn about, very easily.

ModuLearn is perfect for busy students and curious individuals who don’t have time for bulky, unnecessary courses, but still want to learn more about various subjects. No matter what age, or grade you are in, you can get a customizable learning experience without having to sit through boring lectures.

In conclusion, ModuLearn is an alternative and unique educational website that offers a modular, flexible, and different experience to online learning. The modularity features, and customization allows for a very pleasant and productive experience. If you are curious and interested in various subjects, but don’t have the time to commit, ModuLearn is a great platform for you.

**Competitive analysis**

| **Feature** | **ModuLearn** | **Khan Academy** | **Canvas** |
| --- | --- | --- | --- |
| Modular Topic Design | ✓ | ✓ | ✓ |
| Customizability | ✓ |  | ✓ |
| Variety of Topics | ✓ | ✓ | ✓ |
| Topic Experts | ✓ | ✓ | Varies |
| Flexibility | ✓ |  |  |
| User Cost | Free | Free | Higher |

**Data Definition**

Teacher: A teacher is an instructor who is responsible for creating and delivering course content, as well as assessing student performance and providing feedback. Teachers have unique identifiers, usernames, and passwords, and are associated with specific courses.

Student: A student is an individual who is enrolled in a course and is responsible for completing assignments, assessments, and other course requirements. Students have unique identifiers, usernames, and passwords, and their performance is tracked through the gradebook.

Lesson: A lesson is a unit of instruction that is taught within a course. It typically has a title and may have optional content associated with it.

Assessment: An assessment is an evaluation of a student's performance in a course. It may be optional or required, and it has a due date and a title. Assessments may have associated files, such as study materials or grading rubrics.

Assignment: An assignment is a specific task or project that a student is assigned to complete as part of a course. It may be optional or required and has a due date, a title, and a value or weight that contributes to the student's overall grade. Assignments may have associated files, such as instructions or templates.

Module: A module is a self-contained unit of content within a course that typically covers a specific topic or objective. It may have associated files, such as readings, videos, or quizzes. Modules may be linked together sequentially to form a course curriculum.

Course: A course is a structured program of study that is typically composed of multiple modules, lessons, assignments, and assessments. It has a unique identifier, a title, and is associated with a teacher and a gradebook. Courses may have prerequisites and requirements that must be met in order to enroll.

Grade cell: A grade cell is a unit of measurement used in a gradebook to record a student's performance on a specific assignment, assessment, or other course requirement. It is associated with a gradebook, a student, and a module, and includes both the total points available and the earned points.

**Overview, scenarios, & use cases**

**Scenario 1:** Busy Students

Meet John, a busy college student who has to juggle work, school, and family responsibilities. John wants to excel in his classes, but he doesn't have time for bulky, unnecessary courses. With ModuLearn, John can easily search and select modules that cover the specific topics he needs to learn for his assignments and tests. He can learn at his own pace without feeling the pressure of deadlines or assignments. ModuLearn's modular approach makes it easy for John to focus on what he needs to know and skip over the material he's already familiar with.

**Scenario 2:** Curious Individuals

Meet Sarah, a curious person who loves to learn new things. She's always eager to explore different subjects, but she doesn't want to sit through boring lectures or take full-length courses. With ModuLearn, Sarah can browse through various modules covering a wide range of topics, and select the ones that interest her. She can explore and learn at her own pace, without feeling overwhelmed. ModuLearn's modular design allows Sarah to customize her learning experience and delve deeper into the subjects that fascinate her.

**Scenario 3:** Teachers

Meet Ms. Jackson, a high school teacher who wants to create a more personalized learning experience for her students. With ModuLearn, Ms. Jackson can create and upload modules that cover the topics she teaches in class. She can customize the material to match her students' learning styles and pace. ModuLearn's tracking feature allows Ms. Jackson to monitor her students' progress and identify areas where they need more help. By using ModuLearn in her class, Ms. Jackson can make learning more engaging and effective for her students.

In conclusion, ModuLearn is a flexible and customizable e-learning platform that offers a unique and productive experience for students and teachers alike. Its modularity features allow for a more personalized learning experience, making it a great choice for busy students, curious individuals, and educators.

**High-level functional requirements**

1.0 General Account:

1.1 The login screen allows users to log in as either a student or an instructor, or create a new account if they don't already have one. It provides a secure way for users to access the system and their respective courses and materials. The login screen is the first point of entry to the system and is essential for providing users with access to their respective functionalities. (Priority 1)

1.2 Access the Profile site to see a picture of themselves or other users, the courses they're enrolled in, a bio box, and contact email. They can also choose to log out, change passwords, or go back home. (Priority 1)

2.0 Instructor:

2.1 Manage Order: Set the order of the list of topics. Some topics can’t be skipped and must be completed to move on to the next section. (Priority 2)

2.2 Manage Task: Create courses and topics for groups of students. Each course can have optional assignments, which the instructor can toggle on/off. (Priority 1)

2.3 View their own gradebook, which shows a summary of each student's grades for each assignment. (Priority 1)

2.4 Have the gradebook in the form of a set of tables for each course, showing various assignments, quizzes, etc. (Priority 3)

2.5 Use the queueing feature of the modules, which ensures that courses with closer due dates are ordered above those with later due dates. (Priority 2)

3.0 Student:

3.1 View courses and topics set by the instructor. The course with the soonest due date is ordered above courses with later due dates. (Priority 1)

3.2 View gradebook, which is a summary of assignment grades in classes the student is currently enrolled in. Each grade has feedback from the instructor and the amount of points received in total. (Priority 1)

3.3 Have a pop-up that displays the list of enrolled courses with clickable links to assignments due that current day. This pop-up can be disabled. (Priority 1)

3.4 Use the queueing feature of the modules, which ensures that courses with closer due dates are ordered above those with later due dates. (Priority 3)

4.0 Course Content:

4.1 View course content: Both instructors and students should be able to view course content, including modules, lessons, assignments, and assessments. Course content should be presented in a clear and organized manner to make it easy for users to navigate and access the material they need. (Priority 1)

4.2 Create and edit course content: Instructors should be able to create and edit course content, including modules, lessons, assignments, and assessments. They should be able to upload files, add multimedia content, and customize the appearance of the course content. (Priority 1)

4.3 Submit assignments and assessments: Students should be able to submit assignments and assessments through the system. They should be able to upload files and enter text responses, and receive feedback from the instructor on their submissions. (Priority 1)

4.4 Track progress: Both instructors and students should be able to track their progress through the course content. This can include tracking completed modules, lessons, assignments, and assessments, as well as monitoring their grades and overall performance. (Priority 2)

**Non-functional requirements**

1. Scalability: The system should be able to accommodate growth in user demand and data volume by allowing for the addition or removal of modules without affecting the overall system performance.
2. Reliability: ModuLearn should be reliable and available 24/7, with minimal downtime for maintenance or additions to ModuLearn. Data backups should be performed to not lose data on our classes.
3. Security: The system should have robust security features, including data encryption, and protection against hacking of our users login information.
4. Usability: ModuLearn should be user-friendly and intuitive, with a simple and clear interface that allows easy navigation and use of features.
5. Compatibility: ModuLearn should be compatible with various web browsers, operating systems, and devices, ensuring a consistent user experience across different platforms.

**High-level system architecture**

DB Organization:

Grade\_cell

Id: Unique identifier for the grade cell (integer)

Gradebook\_id: Identifier for the gradebook associated with the grade cell(integer)

Student\_id: Identifier for the student associated with the grade cell (string)

Module\_id: Identifier for the module associated with the grade cell (integer)

Total\_points: Total points available for the grade cell (integer)

Earned\_points: Points earned for the grade cell (integer)

Student

Username: Unique identifier for the student (string)

Hashed\_pw: Hashed password for the student (string)

First\_name: First name of the student (string)

Last\_name: Last name of the student (string)

Teacher

Username: Unique identifier for the teacher (string)

Hashed\_pw: Hashed password for the teacher (string)

First\_name: First name of the teacher (string)

Last\_name: Last name of the teacher (string)

Course

Id: Unique identifier for the course (integer)

First\_module\_id: Identifier for the first module in the course (integer)

Latest\_module\_id: Identifier for the latest module in the course (integer)

Title: Title of the course (string)

Teacher\_id: Identifier for the teacher associated with the course (string)

Gradebook\_id: Identifier for the gradebook associated with the course (integer)

Module

Id: Unique identifier for the module (integer)

Next\_module\_id: Identifier for the next module in the course (integer)

Module\_type: Type of the module (enum)

Content\_id: Identifier for the content associated with the module (integer)

Assignment

Id: Unique identifier for the assignment (integer)

Optional: Indicates if the assignment is optional (boolean)

DueDate: Due date of the assignment (timestamp)

Title: Title of the assignment (string)

File location: Data associated with the assignment (nvarchar)

Assessment

Id: Unique identifier for the assessment (integer)

Optional: Indicates if the assessment is optional (boolean)

DueDate: Due date of the assessment (timestamp)

Title: Title of the assessment (nvarchar)

File location: Data associated with the assessment (nvarchar)

Lesson

Id: Unique identifier for the lesson (integer)

Optional: Indicates if the lesson is optional (boolean)

DueDate: Due date of the assignment (timestamp)

Title: Title of the lesson (nvarchar)

File location: Data associated with the lesson (nvarchar)

Main API’s:

* GET /grades: Returns a list of all grade cells in the database
* POST /grades: Creates a new grade cell in the database
* GET /students/:id: Returns a specific student by their ID
* POST /students: Creates a new student account in the database
* PUT /students/:id: Updates an existing student account by their ID
* GET /teachers/:id: Returns a specific teacher by their ID
* POST /teachers: Creates a new teacher account in the database
* PUT /teachers/:id: Updates an existing teacher account by their ID
* GET /courses: Returns a list of all courses in the database
* GET /courses/:id: Returns a specific course by its ID
* POST /courses: Creates a new course in the database
* PUT /courses/:id: Updates an existing course by its ID
* GET /modules/:id: Returns a specific module by its ID
* POST /modules: Creates a new module in the database
* PUT /modules/:id: Updates an existing module by its ID
* GET /assignments/:id: Returns a specific assignment by its ID
* POST /assignments: Creates a new assignment in the database
* PUT /assignments/:id: Updates an existing assignment by its ID
* GET /assessments/:id: Returns a specific assessment by its ID
* POST /assessments: Creates a new assessment in the database
* PUT /assessments/:id: Updates an existing assessment by its ID
* GET /lessons/:id: Returns a specific lesson by its ID
* POST /lessons: Creates a new lesson in the database
* PUT /lessons/:id: Updates an existing lesson by its ID

Media Storage:

We have decided to store our images and HTML information in our database rather than in a file system. We will be storing HTML information in the database. This means that any text-based content displayed on our application, such as articles or user-generated comments, will be stored as text data in the database

Search/filter architecture and implementation:

We are using SQL queries to sort and filter data coming from our database.

In order to provide a search and filter functionality for our database, we will use SQL queries to retrieve data based on specific criteria. The algorithm for the search functionality will involve using the "SELECT" statement with appropriate "WHERE" clauses to narrow down the search results based on the desired search terms.

For example, if we wanted to search for all the grades of a specific student, we would use a query like:

SELECT \* FROM grade\_cell WHERE student\_id = 'student\_id'

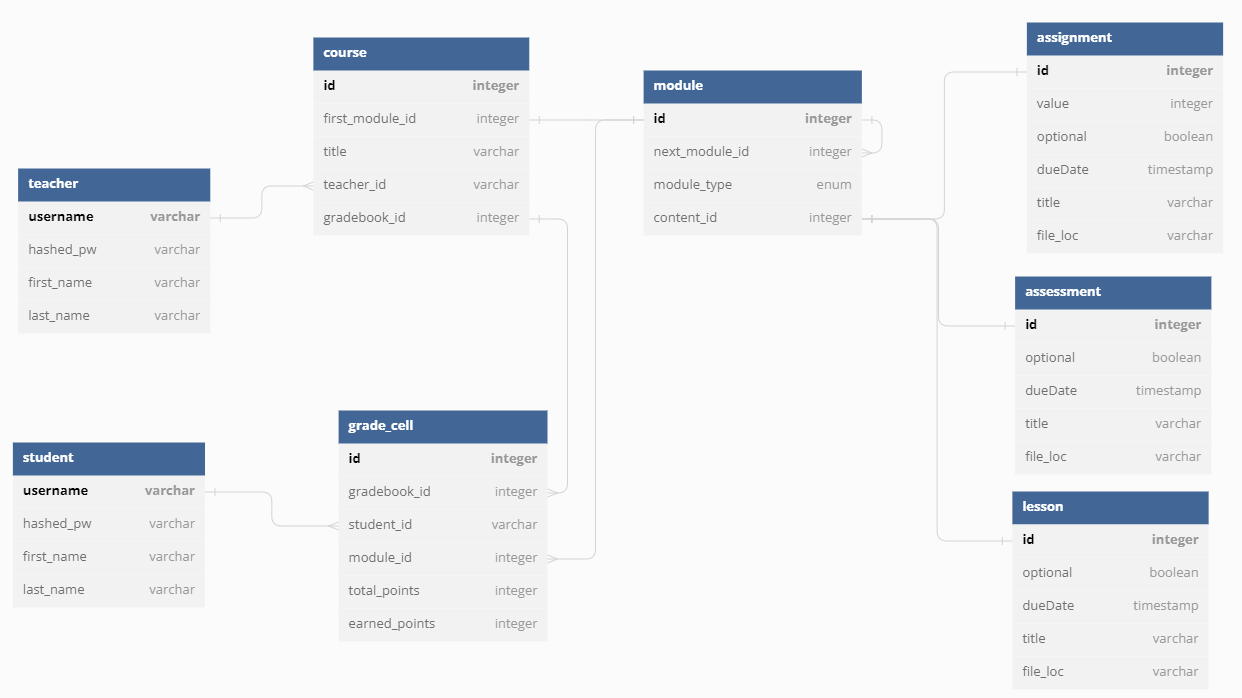
This would return all the grade cells associated with the specified student ID.

There are several pre-built queries that can be used to interact with the system in specific ways, such as retrieving all grades for all students and modules in a specific class.

**High-Level UML diagrams:**

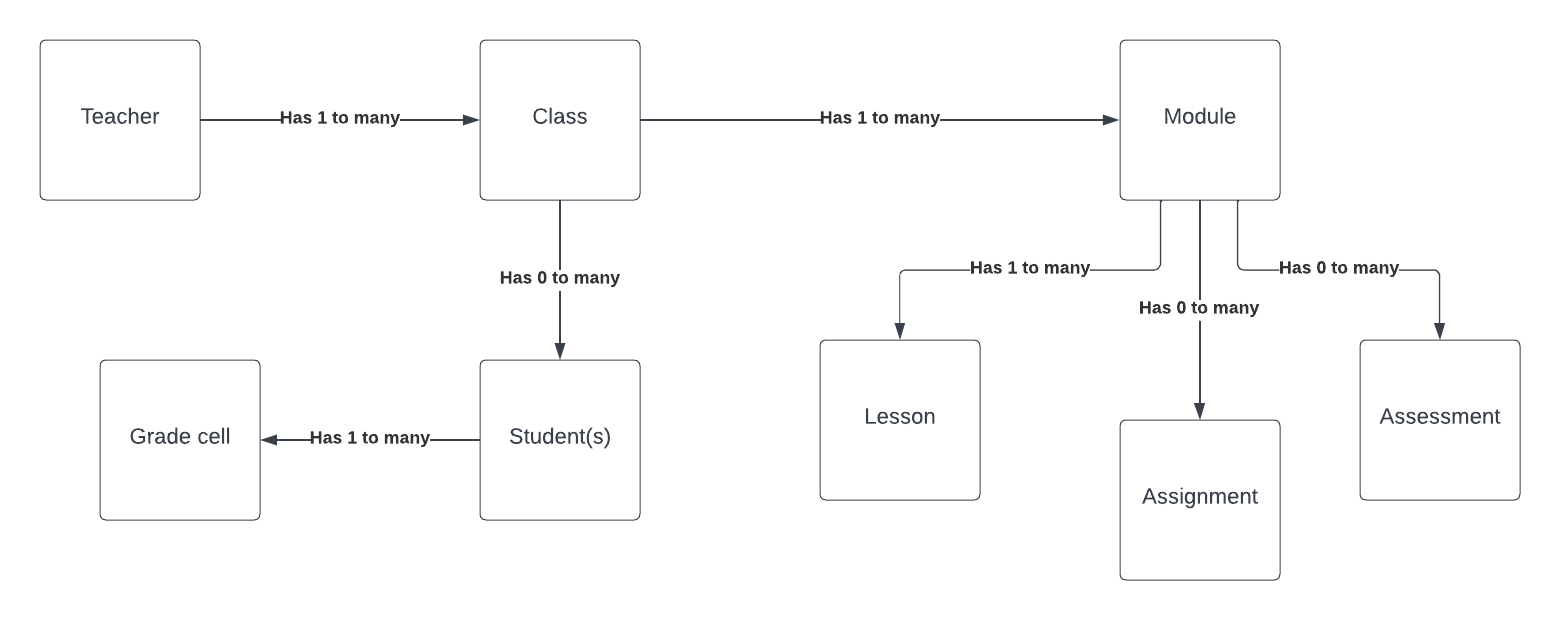
1) High-level UML class diagrams for implementation classes of core functionality:

**Database Design**

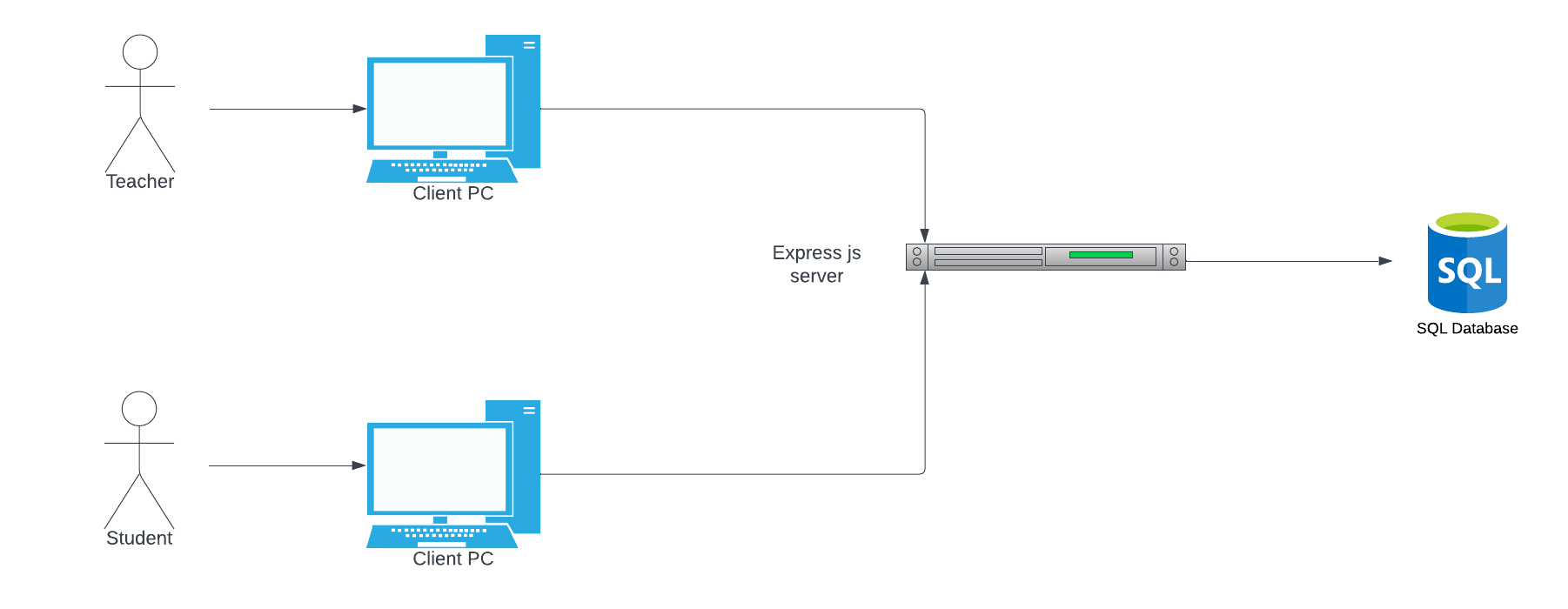
****

**UML Component & deployment diagrams:**

UML Component diagram:

****

Deployment diagram:

****

**Identified Key Risk**

Skills Risks:

Not everyone in the team has sufficient knowledge or experience in the development tools and technologies required for the project. This could lead to delays in development, mistakes or errors in coding, and lower quality outputs.

Resolution Plan:

To address this risk, the team has assigned more experienced team members to mentor or support those who need assistance. Additionally, we are using more familiar tools and technologies to reduce the learning curve and improve efficiency. We have decided to use React to develop so assist in this.

**PROTOTYPE PRESENTATION**: https://youtu.be/rdE4C4CspT0

**Team**

Ethan Curtis - Product Owner, Back-End

Kevin Horta - Scrum Master, Front-End

Bryan Cooke - Front-End

Marco Parucho - Back-End

Oliver Pennanen - Github Master, Back-End