# Project Plan

## 2.1
## Project Management/Tracking Procedures

We are adopting a Project Management procedure in alignment with the values of Agile Development. We will be designing core components/views and receiving feedback / adjusting in quarters repeated on a timeline of every week or 2 once we meet with the client. We want to be able to provide a product that Maruf would be happy to use, meaning keeping him posted at different deadlines and receiving changes to be immediately implemented to result in a successful end project.

We will be using GitLab issues to track stories, and collaborate via Discord when we finish, or update a PR to be merged so that way we can keep versioning working at all times. Git will be our method of version control for our code.

## 2.2
## Task Decomposition

1. **Conceptual Analysis**
   *Goal: Establish a fundamental understanding of the problem our project is solving and the users involved.*
   a. Requirements analysis
   b. User needs analysis
   c. View wireframing

2. **High-Level Design**
   *Goal: Create the general structure of our solution and decide on how it should roughly look once complete.*
   a. Page breakdown
   b. Interface designs
   c. Backend API definitions
   d. Database schemas

3. **Low-Level Design**
   *Goal: Decide on the necessary low-level details of our solution to prepare us for the implementation of the minimum viable product.*
   a. Frontend library selection
   b. Interface component designs
   c. Component hierarchy design
   d. Backend class diagrams

4. **Minimum Implementation**
   *Goal: Implement our solution at the minimum level.*
   a. Minimum implementation of the frontend (using mock data)

    b. Minimum implementation of the backend (on local machine)
    c. Basic frontend user testing
    d. Basic backend API testing

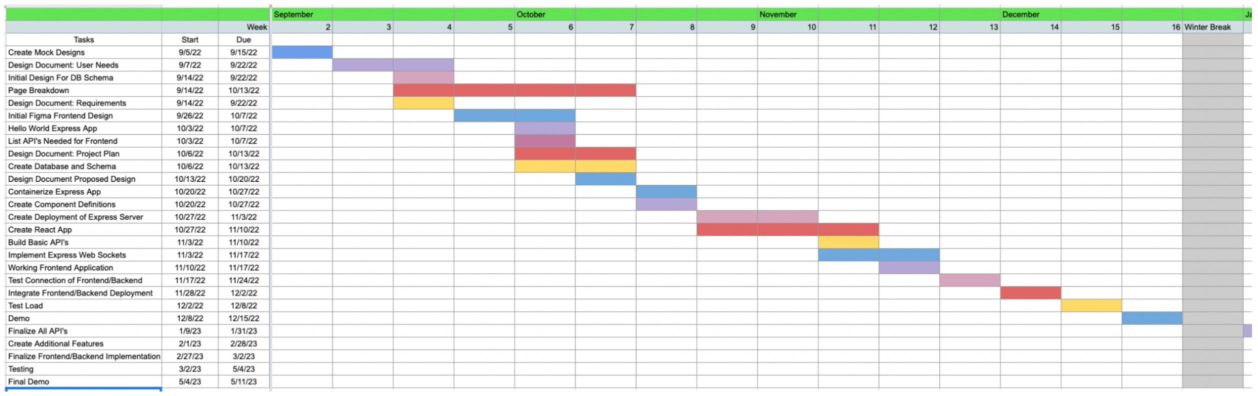
5. **Integration / Initial Prototype**
   *Goal: Connect the frontend and backend and iron out the resultant bugs to create the initial prototype of our solution.*
    a. Connection of frontend to backend APIs
    b. Deployment of backend to server
    c. Integration testing

6. **Product Iteration (Continuous)**
   *Goal: Continuously iterate on the solution to achieve all requirements and satisfy the client.*
    a. Iterative achievement of all requirements
    b. Continuous iteration on new/old functionality
    c. Expansion of automatic testing

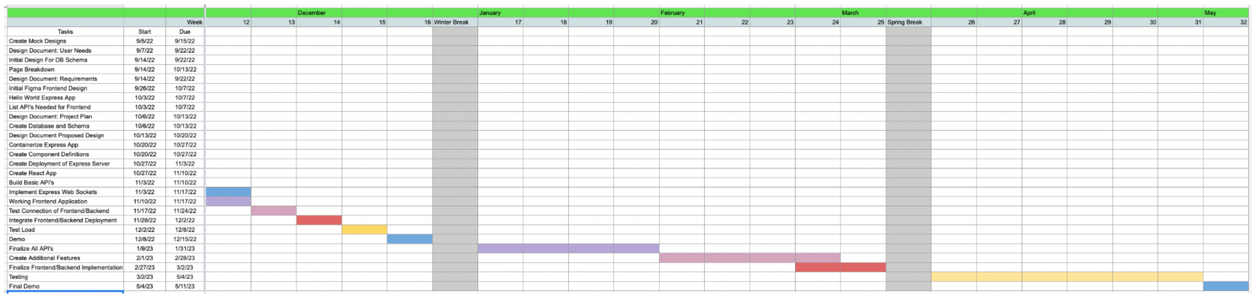
## 2.3
## Project Proposed Milestones, Metrics, and Evaluation Criteria

- Simple back-end server is able to handle requests locally
  - Simple health check returns status 200
- Back-End server is hosted on our Iowa State Server
  - Server is running in a container
  - CI/CD pipeline handles deployment
- Web Socket connection working
  - User is able to maintain a live connection to the server for messaging
- Core APIs working
  - All APIs required by the project are functioning
- Front-End has working pages with Mock data
  - Designs are created in React and are working with mock data
- Front-End incorporates Back-end APIs and the app is fully working locally
  - Front-End no longer uses mock data
  - Core features working locally
- Front-End and Backend Integrated deployment
  - Both front-end and backend are deployed on our server
- Application can handle a classroom sized load
  - Application is able to work with 200+ people connected

## 2.4
## Project Timeline/Schedule



| Tasks | Start | Due |
|---|---|---|
| Create Mock Designs | 9/5/22 | 9/15/22 |
| Design Document: User Needs | 9/7/22 | 9/22/22 |
| Initial Design For DB Schema | 9/14/22 | 9/22/22 |
| Page Breakdown | 9/14/22 | 10/13/22 |
| Design Document: Requirements | 9/14/22 | 9/22/22 |
| Initial Figma Frontend Design | 9/26/22 | 10/7/22 |
| Hello World Express App | 10/3/22 | 10/7/22 |
| List API's Needed for Frontend | 10/3/22 | 10/7/22 |
| Design Document: Project Plan | 10/6/22 | 10/13/22 |
| Create Database and Schema | 10/6/22 | 10/13/22 |
| Design Document Proposed Design | 10/13/22 | 10/20/22 |
| Containerize Express App | 10/20/22 | 10/27/22 |
| Create Component Definitions | 10/20/22 | 10/27/22 |
| Create Deployment of Express Server | 10/27/22 | 11/3/22 |
| Create React App | 10/27/22 | 11/10/22 |
| Build Basic API's | 11/3/22 | 11/10/22 |
| Implement Express Web Sockets | 11/3/22 | 11/17/22 |
| Working Frontend Application | 11/10/22 | 11/17/22 |
| Test Connection of Frontend/Backend | 11/17/22 | 11/24/22 |
| Integrate Frontend/Backend Deployment | 11/28/22 | 12/2/22 |
| Test Load | 12/2/22 | 12/8/22 |
| Demo | 12/8/22 | 12/15/22 |
| Finalize All API's | 1/9/23 | 1/31/23 |
| Create Additional Features | 2/1/23 | 2/28/23 |
| Finalize Frontend/Backend Implementation | 2/27/23 | 3/2/23 |
| Testing | 3/2/23 | 5/4/23 |
| Final Demo | 5/4/23 | 5/11/23 |



## 2.5
## Risks And Risk Management/Mitigation

| Risk | Probability | Mitigation |
|---|---|---|
| Performance target may not met when our users exceed 100 | 75% | We can acquire another server (or multiple servers) to help balance out the load when users exceed 100. |
| Malicious users get into the course that they don't belong in | 60% | We could store a list of eligible users (listed by their student IDs) for each course. Then when a malicious user tries to enter/add a course it will not allow them to add it unless they are on the eligible list of users. |
| Course still says it is "LIVE" after the class has adjourned for the day.<br>In this case the professor constantly forgets to | 30% | |

| | | |
|---|---|---|
| adjourn/end the class | | |
| Professor continuously reveals the students name (after posting anonymously) to the class while sharing their screen. | 20% | |
| Technical risk: There could be many more feature requests once we have our project in use. We will not be able to implement all of these requests in a short amount of time. | 20% | |

## 2.6
## Personnel Effort Requirements

| **Task** | **Hours** |
|---|---|
| Database initialization | 40 |
| Initial Docker container deployment | 100 |
| Web socket routing | 50 |
| Express.js initial setup | 5 |
| Dockerfile creation | 25 |
| REST API creation | 250 |
| Frontend page creation | 350 |
| Frontend/backend integration | 50 |
| Deployment of frontend | 100 |
| Load Testing | 50 |

While we believe that our project will take significant amounts of time in many areas, by far the tasks that will be most time consuming are the creation of the frontend screens and APIs in the backend. This is because they will require the most logic and actual code/testing, while much of the other work will be some form of devops or a particular feature, and therefore less time consuming.

## 2.7
## Other Resource Requirements

The server necessary to run the application on, and some sort of hosting/storage capability for the app's data once it begins to be hosted via Iowa State and used live in classrooms on campus.