CIS4339 Fall 2024 - Project Description

Synopsis

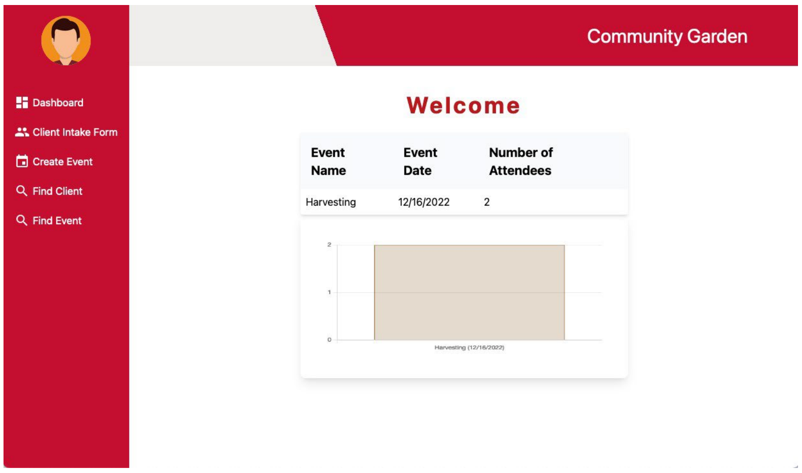
Your team has to take over a web application development project. The overall goal is to expand the original project, add new features and expand documentation. You will have to assess existing code and NOT start an application from scratch.

The application follows a standard full-stack architecture using the MEVN stack (MongoDB, Express, VueJS, Node). The project will be executed as a group project. You will be assigned to a team by the instructor.

This is a prototype implementation, e.g. we are not following all best practices for securing a web application.

Background

A team of students has developed a basic dataplatform for a non-profit organization in the Houston area. In general, an organization wants to help clients with basic needs (e.g. receiving food aid or help with adult education). The end users for the application are Community Health Workers (CHWs) who work at the non-profit with those clients. CHWs want to understand how to help their clients but also want to make sure that the organization can plan resources. They enter information about a new client in the "Intake Form". Events can be created. Existing clients and events are listed in a separate view. Furthermore, CHWs can "sign-up" clients for events.



Event Name	Event Date	Number of Attendees
Harvesting	12/16/2022	2

Figure 1. Screenshot

A demo of the current project (with read access only) can be found at https://witty-ocean-0485b6410.2.azurestaticapps.net.

The application can be used by multiple organizations. The key is consolidation at the data level and data for all organizations are stored in one single database. The idea is that there are multiple instances of the application - one instance per organization. An instance is configurable via an environment variable that sets up the instance. When the front-end of the instance connects to that backend it will be used only for that organization (see architecture drawing and configuration example).

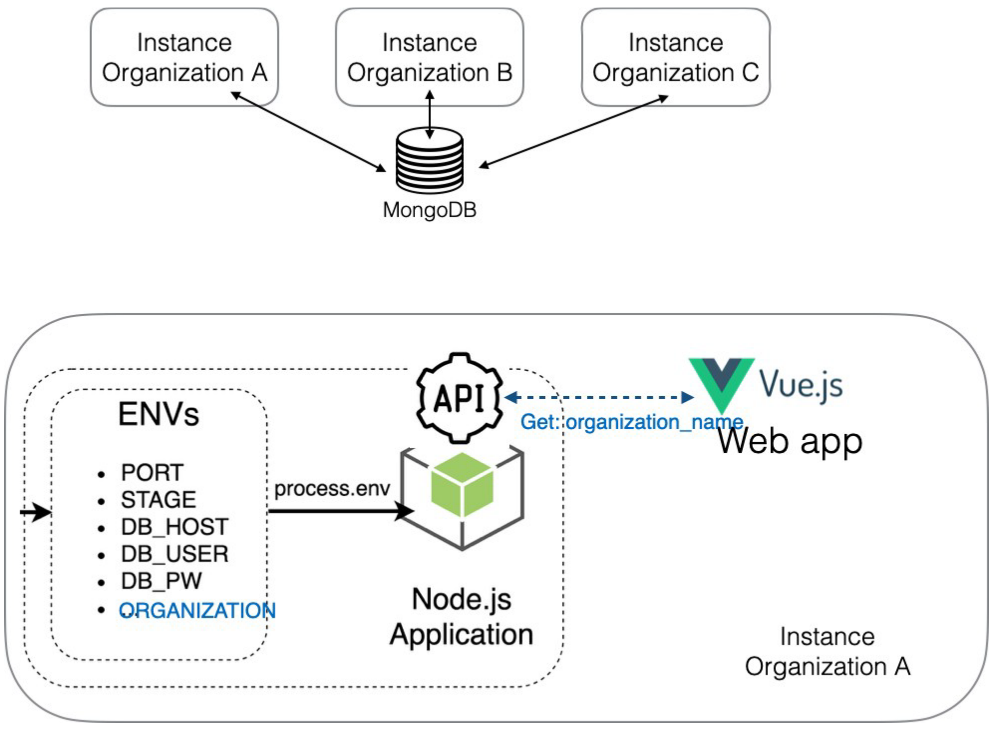
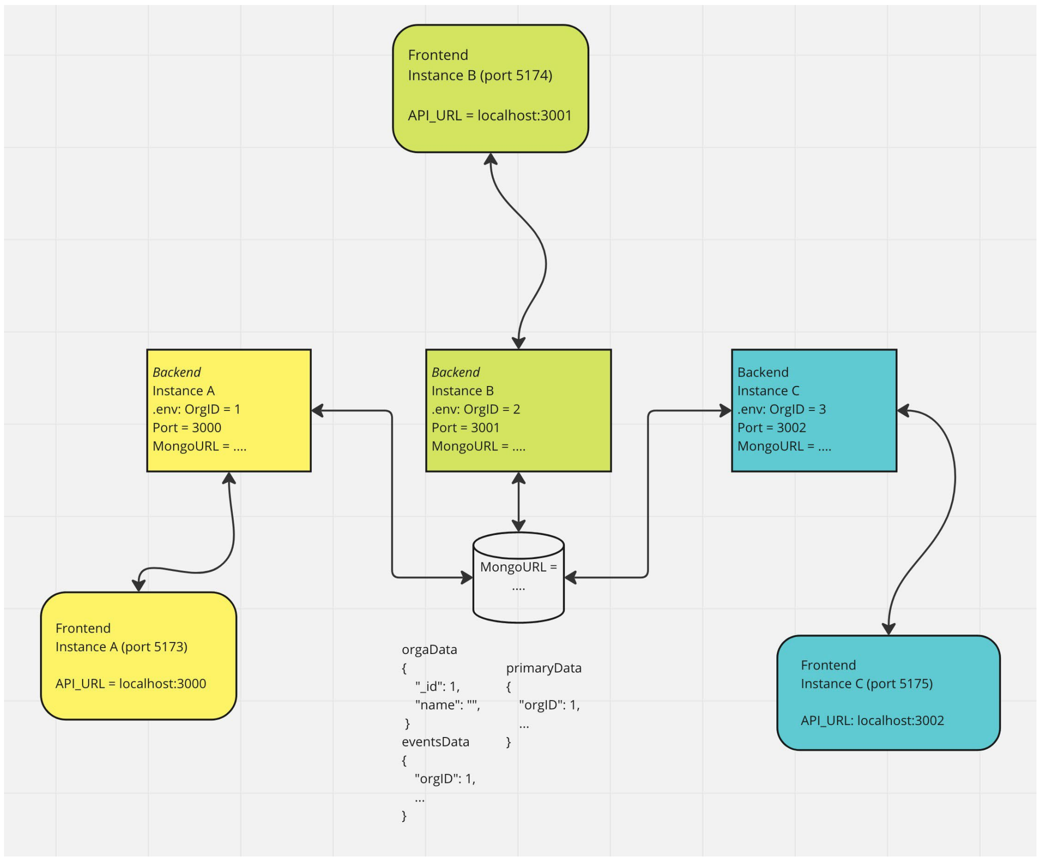


Figure 2. Architectural drawing



```
orgaData
{
   "_id": 1,
   "name": "",
}
eventsData
{
   "orgID": 1,
   ...
}
```

```
primaryData
{
   "orgID": 1,
   ...
}
```

Figure 3. Configuration example

Getting started

The existing project code will be provided to you in a GitHub repository. This repository will be created when your team creates their group in GitHub classroom. Use the instructions provided in Canvas for more details.

Requirements

a. Update the code from **Options API** to **Composition API**.

b. Add **user login** capabilities to the application. The front-end should have a login page that allows a user to login. When a user is not logged in, they should not see and be able to access the menu option pertaining to client and event management. They should only see the dashboard. Users can have 2 roles: viewers or editors. Viewers, for instance, should be allowed to access the "Find Client" and "Find Event" pages but not the "Client Intake Form", the "Create Event Form" or the functionalities related to editing client or event information. Editors can access all pages/functionalities.

   You need to create a login page, update navigation and also update the backend. You will not need to create user registration, password reset etc. functionalities. Passwords should be sent and stored as "hashed strings" (see https://stackoverflow.com/questions/43092071/how-should-i- store-salts-and-passwords-in-mongodb for a simple example). Users and their roles will be set up in the database directly by a DB admin. Users will not need access across organizations (they will essentially belong to one organization/instance only).

c. Add **Services** at the data layer and implement CRUD functionality.

   Events offer a list of services. The list of services is currently hard-coded. You need to create functionality that allows the creation, modification and soft deletion (active/not active) of services for an instance (meaning the list of services can be different for different organizations). The navigation in the front-end has to be extended to reflect the new functionality and page(s) have to be implemented that allow users to create and edit the services. When an event is created it should pull the list of services from an API endpoint and only show currently active services.

   Users with the Role "Viewer" need to be able to see the List of Services but won't be able to create or edit them. Editors will be able to access all pages/functionalities related to Services.

d. The **Dashboard** page should be extended with a Pie or Doughnut chart that shows clients by zip code.

   When you work on the front-end during Sprint 1, you can use "hard-coded" data for the graph. During Sprint 3, you need to create an API endpoint to provide the data dynamically.

You will work on those requirements in **4 Sprints** (see below).

Each deliverable for the project is due at midnight of the listed due date. No late submission will be accepted.

Sprint 1: Group - Create a Functional Specification Document - Deadline February 15, 2024

Using the Requirements listed above, create a Functional Specification document (pdf) for the **new features**.

Before starting, you should conduct prototype testing using the existing version of the fullstack application. Conducting some early testing may help you gain insight into how users might interact with the product. Testing at this stage may help you confirm or reject your initial assumptions about the product, such as how users navigate between interfaces and how they apply the product’s features.

This stage is important for determining what product essentials to include in your specification document. Prototype testing allows you to gather data on how users are most likely to use a product, which can inform the development process and help you adapt the product to best serve your users.

Your document should cover the following aspects:

1. Develop an outline

   Once you have gathered all the background information and tested the existing prototype, draft an outline for the planning document. You may choose to use a template with pre-defined sections or create your own depending on the needs of your project. Think about what information you and your team need to fulfill the project objectives. Creating distinct section headings for each part of the document may help you organize your document and ensure you include all the necessary and relevant content.

2. Create use cases and scenarios

   Use cases are an important part of a functional specification document because they describe how actual users are most likely to use the product. They may help you and your team understand the context in which the user may apply each of your product’s features. Understanding the function of each feature may help you determine the best design to help your users navigate the product to meet their goals. A use case can be as simple as describing a scenario in which a user encounters a problem and uses the product to resolve it.

3. Include user flows

   A user flow specifies the sequence in which the user interacts with the product to achieve an objective. This section typically includes a diagram that maps out how users might navigate the different windows of your application to complete a task. Mapping user flows may help you predict the different methods that users might use to accomplish the same task. For example, you may create a system that allows two different users to navigate to the same page by following a different series of links within the application. Identifying different user flows can help you design the application for ease of use.

4. Define your timeline and assign tasks

   Finally, you need to include a timeline for development. The timeline should include deadlines for each stage of the development process, in addition to deadlines for each feature implementation and testing. Incorporating a timeline might help you and your team identify how long each step in the development process may take, develop strategies for time management and efficiency and track progress toward objectives. Each task in the timeline should list the group member(s) responsible for the implementation.

You will be allowed to make changes to your specification document but the original grade received won’t change.

NOTE	The Sprint 1 assignment has been created using: https://www.indeed.com/career-advice/career-development/functional-specification

Sprint 2 Group - Implementation of New features in Frontend - Deadline March 7, 2024.

The second sprint focuses on the UI development in JavaScript using VueJS.

For Sprint 2 you will have to implement the **front-end** extensions as described in the requirements. You are required to update the code base and use composition API for the front-end. You must integrate your extensions into the existing front-end implementation. You will be testing your extended code without the need to have the API endpoints implemented that make up the backend for the app.

Your team will only have to submit the link to your GitHub repo in Canvas for Sprint 2. Do not forget to comment and document your references in code.

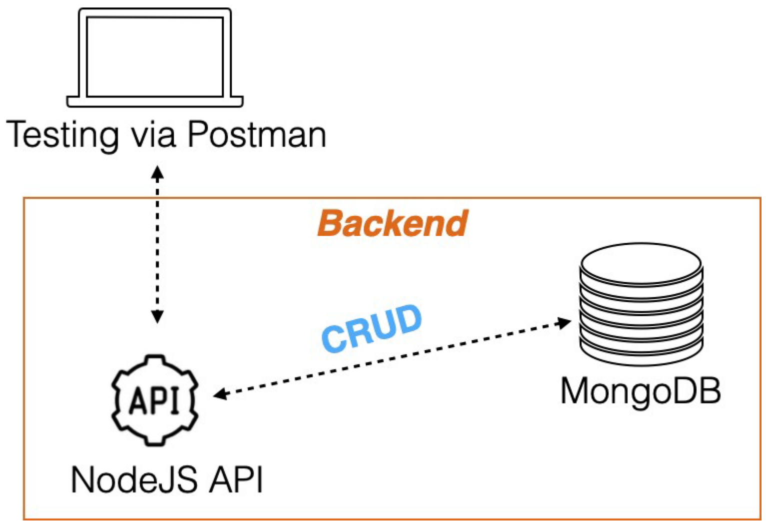
Sprint 3 Group - Expansion of Backend API, documentation and presentation - Deadline April 30, 2024

The third sprint focuses on the implementation of the extended backend, setup of the MongoDB and the additional functionality of the API services according to the above listed requirements. These services need to be implemented using MongoDB, Express and Node and have to be **integrated with the existing** backend code.

In addition to the *code* you are required to submit *API documentation* for the extensions. You will find an additional document in Canvas that shows how you can use Postman to generate your documentation.

Code submissions will happen in the same repository on GitHub Classroom.

You MUST setup a cloud based MongoDB instance (e.g. https://www.mongodb.com/atlas) for your group as it will make the development process much easier. Please submit a Mongo URL string in Canvas. (don't just put it in GitHub).

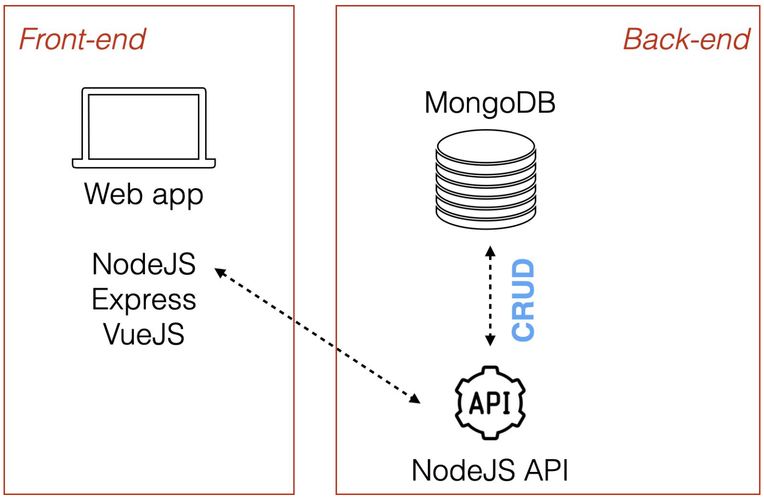


Part of the submission for Sprint 3 will be a 10-15 minutes long group **presentation video**, highlighting the new features, and explaining the solution in detail.

NOTE

Changes to backend due to front-end development

You are allowed to make changes to your code base for the front-end after the Sprint 2 deadline has passed. We will however grade the version that was submitted originally for Sprint 2.



Details for the Github setup (to be used for submission in Sprint 2 and 3)

We are using Github Classroom to setup the remote repo for this assignment. The link to accept the assignment and create the private GitHub repo will be posted in Canvas. Accept the link and a new GitHub repository will be created for your team containing the existing code.

Since this is a group assignment, you will be asked to join/form a team in GitHub classroom. There are three common cases, and you will have to follow one of them:

a. There are no groups yet. You need to enter the name of a new group (following the group formation table provided in canvas) to create it.

b. There are one or more groups already formed. You need to click on the existing group that you should join (see group formation table).

c. Your group is not listed yet. You need to enter the name of the new group to create it.

Sprint 4 Individual - Peer Evaluation - Due May 5, 2024

As part of this assignment, you will need to evaluate 2 of your classmate's projects. The github repositories of all teams will be made available to you after Sprint 3 has finished. There will be a detailed rubric provided for how to evaluate the project code and presentation. This asignment will open up with all details right after the deadline for Sprint 3.

Other Requirements

- Make sure your project compiles and runs locally and the github README provides instructions for local setup. If the project doesn't run, you will forfeit points.

What to Turn in

Commit your source code (properly commented) to your private repository in the Github classroom. Do not zip the files together. The github repository must show multiple **meaningful commits for**

each of the team members that will be used to judge individual contributions of each team member.

There will be 4 assignments listed in Cavas - one for each Sprint.

a. For Sprint 1, your team must submit a document (pdf) via Canvas.
b. For Sprint 2 submit the link to your github repository containing your project via Canvas.
c. For Sprint 3 submit the link to your video presentation. **Video MUST be uploaded into MS Stream and accessible** for everyone in our class team.
d. The peer-evaluation will be done with MS forms (details posted in Canvas after the end of Sprint 3).

IMPORTANT

Please make sure your team agrees on procedures and setup of branches etc. to support a productive development process. We have discussed one workflow example in class. You are not required to follow it. However, for grading we will only check the main/master branch in the repository setup via Girhub Classroom.

The deadlines for this project are not flexible since we need to allow for enough time for the peer-evaluation and grading.

Extra Credit

You can earn extra credit if you are able to deploy the complete or parts of the application (backend code and front-end code) into the cloud. Do not attempt deployment of the backend/frontend until you have finished the other parts of the project. We will grade you on the local deployment first! (Make sure the username and password for the database you setup are credentials you're willing to share. Do not use personal passwords for this homework, which you might be using anywhere else.)

Please submit a short description of the deployment strategy/setup and credentials to connect to your cloud DB instance via Canvas to receive extra credit.

Basic Grading rubric for the project

Remember the whole project accounts for 35% of the overall grade in the class. All team members will receive the same points based on the assumption that a team has equally distributed commits for all team members and every team member **contributed equally** to code development. Comments are not code. We are using git tools to check on individual contributions. The Peer- Evaluation points are given on an individual basis.

Item	Points in Sprint 1 and 2	Points in Sprint 3 and 4
Functional Specification Document	10	
Options API to Composition API	5	
Login page	5	
Updated Navigation reflecting login/roles	10	
Pages and navigation for Services	10	
Dynamic list of services for events	5	
Additional Graph in dashboard	5	
Cloud setup for DB		5
Code for new schemas for users and services with comments		5
Code for extended Backend APIs for login, services and new graph		15
Extended API documentation		5
Video presentation		10
Quality of Peer Evaluation		10
Total	50	50
Deduction 50% if code doesn't compile	(up to -50)	
Deduction for not submitting link to Github	-5 points	
Deduction for not submitting link to MS Streams		-5 points
Deduction for not having multiple meaningful commits done by each team member showing progress	-20 points	-20 points
Extra credit for Cloud deployment	up to 10 points (backend 5 points /front-end 5 points)	