Functional Specification Document

Joanna Macedo, Trinity Klein, Maria Abejide

CIS 4339 Group9

Table of Contents

1. Introduction

Welcome to the Functional Specification Document for the CIS4339 web application development project. In this document, we outline the scope, objectives, and technical details that the development team was tasked with for extending the functionality of the existing application built by the previous students.

This online application was first created by a group of students and is used as a data platform by a Houston-area non-profit. Helping customers with basic needs like food assistance and adult education is the organization's main goal. Community Health Workers (CHWs), who are the application's end users, use it to organize events, collect client data using the "Intake Form," and assist with client sign-ups. The application's goal is to help corporate resource planning while giving CHWs valuable insights to help them serve clients more effectively.

The goal of our team is to improve the web application by adding new features and building on the foundation set by the original developers. It is important to understand that, as opposed to beginning from scratch, our method entails assessing and expanding upon the current codebase. The application will be constructed using a conventional full-stack architecture on the MEVN stack, which consists of MongoDB, Express, VueJS, and Node.

2. Project Scope

The project scope includes a series of enhancements and feature implementations tailored to the application's requirements:

a. Transition to Composition API:

- Update the codebase from the Options API to the Composition API to improve code organization and maintainability.

b. User Authentication and Role-based Access Control:

- Develop a user login system with hashed password storage for security.
- Implement role-based access control with two user groups: viewers and editors.
- Design the frontend to display a login page and restrict menu options related to client and event management based on user authentication status and role.

c. Integration of Services at the Data Layer:

- Introduce CRUD functionality for managing event services, replacing the current hard-coded system.
- Enable creation, modification, and soft deletion of services tailored to individual organizations.
- Extend front-end navigation to accommodate service management pages, allowing users to create and edit services.
- Integrate an API endpoint to dynamically retrieve and display the list of currently active services when creating events.

d. Dashboard Enhancement with Visualization:

- Extend the dashboard page to incorporate a Pie or Doughnut chart depicting client distribution by zip code.
- Initially utilize hard-coded data for the chart during frontend development in Sprint 1, transitioning to dynamic data retrieval via an API endpoint in Sprint 3.

Through these improvements, we will make sure that frontend and backend components integrate seamlessly, that best practices for security and code quality are followed, and that new features and functions are thoroughly documented. We will also make sure to put accessibility and user experience first, making it easier for editors and viewers to navigate and utilize effectively.

3. Deliverables

User Authentication and role-based access control:

Functional Deliverables:

1. Create a login page in the front-end allowing users to authenticate with their credentials.
2. Implement frontend logic to hide menu options related to client and event management when a user is not logged in.
3. Ensure the dashboard is the only visible page for unauthenticated users, restricting access to other functionalities.
4. Develop backend functionality to handle user authentication and authorization based on roles (viewers or editors).
5. Update the front-end navigation to reflect the user's authentication status and role, displaying appropriate menu options accordingly.
6. Implement password hashing for secure storage and transmission of user passwords.
7. Configure backend to store user information and roles in the database, allowing for setup by a database administrator.
8. Ensure user access is restricted to one organization/instance, preventing access across organizations.

UI Deliverables:

- Design login page layout with input fields for username and password.
- Create UI elements for the login button, username, and password input fields.
- Design error message prompts for incorrect login attempts or unregistered users.
- Customize UI to display menu options dynamically based on user authentication and role.
- Implement UI components for hiding/showing menu options and displaying the dashboard.

Integration of Services at the Data Layer:

Functional Deliverables:

1. Develop CRUD functionality for services at the data layer, allowing the creation, modification, and soft deletion (active/not active) of services.

2. Design and implement front-end pages to facilitate the creation and editing of services, ensuring seamless integration with backend functionality.
3. Extend frontend navigation to include links to the new service management pages, reflecting the updated functionality.
4. Integrate backend logic to fetch the list of services from an API endpoint when creating events, ensuring that only currently active services are displayed.
5. Configure frontend to allow users with the "Viewer" role to view the list of services, while restricting their ability to create or edit them.
6. Grant users with the "Editor" role access to all pages and functionalities related to services, allowing them to create, modify, and delete services as needed.

UI Deliverables:

- Design UI layout for service management pages, including forms for creating, editing, and deleting services.
- Create UI components for navigation links to service management pages.
- Design buttons and input fields for interacting with service data (e.g., create, edit, delete buttons).
- Customize UI to display different functionalities based on user roles (e.g., hide editing options for the "Viewer" role).
- Implement UI elements for displaying service lists and details.

Dashboard Enhancement with Visualization:

Functional Deliverables:

1. Design and implement a Pie or Doughnut chart component within the Dashboard page to visualize client distribution by zip code.
2. Develop backend functionality to retrieve client data grouped by zip code and provide it to the front end for chart rendering.
3. Integrate the chart component with the backend data to dynamically display client distribution.
4. Ensure the chart updates dynamically as new client data is added or modified.
5. During Sprint 1, use hard-coded data to populate the chart for initial front-end development.
6. During Sprint 3, create an API endpoint to dynamically fetch client data for the chart, allowing for real-time updates.

UI Deliverables:

- Design layout for the dashboard page to accommodate the new chart component.
- Create UI components for the Pie or Doughnut chart to visualize client distribution by zip code.
- Design chart styling and colors to enhance visualization.
- Customize UI to ensure seamless integration of the chart with backend data retrieval.
- Implement UI elements for displaying client distribution information and updating chart visuals.

Use Cases

Transition to Composition API:

Use Case Name: Transition to Composition API
ID: COMPOSITION_API_01
Area: Software Development - Codebase Refactoring
Actor(s): Software Developers, Codebase Maintainers
Description: Refactor the existing codebase to use the Composition API instead of the Options API to enhance code organization and maintainability.
Trigger Event: Decision to improve the software architecture and maintainability of the code.
Steps Performed:

The developer reviews existing components using the Options API.

The developer rewrites a component using the Composition API.

The developer verifies that the new component maintains existing functionality.

The developer commits changes to a version control system.

The codebase is reviewed, and changes are merged into the main branch.
Information for Steps:

Identify components, their dependencies, and side effects.

Reorganize the component's logic using the **setup** function and reactive references.

Test the component in a development environment.

Document changes and updates the version control.

Peer review and automated testing before merging.
Preconditions: Developers are trained in the Composition API.
Postconditions: The codebase solely uses the Composition API for component logic.
Assumptions: The Composition API provides all the necessary features to replace the Options API.
Requirements Met: Codebase utilizes Composition API for improved organization and maintainability.

User Authentication and Role-based Access Control:

Use Case Name: User Authentication and Role-based Access Control

ID: AUTH_RBAC_01

Area: Security - Access Management

Actor(s): System Users, System Administrators

Description: Authenticate users and provide access based on predefined roles to ensure secure access to system resources.

Trigger Event: User attempts to access the system.

Steps Performed:

1. The user provides credentials (username/password) at the login screen.

2. The system validates credentials against the user database.

3. Upon successful authentication, the system retrieves the user role.

4. The system grants access based on the role-specific permissions.

5. The user accesses the permitted system resources.

Information for Steps:

1. Entry of secure username and password.

2. Credential verification with encryption methods.

3. Role information retrieval from the database.

4. Access control enforcement based on role.

5. User interaction with the system within role constraints.

Preconditions: The user has an active account with the assigned role(s).

Postconditions: User gains access to resources according to their role.

Assumptions: Role definitions and permissions are preconfigured.

Requirements Met: Secure, role-based access to the system is provided.

Integration of Services at the Data Layer:

Use Case Name: Interacting with Service Data

ID: SERVICE_INTERACTION_01

Area: Backend Development - Data Management

Actor(s): Application Users (Backend Developers, Database Administrators)

Description: Users interact with service data through CRUD operations to manage services effectively within the application.

Trigger Event: User initiates an action to add, delete, or update service data.

Steps Performed:

1. User navigates to the service management section of the application.

2. User selects the desired service to add, delete, or update.

3. User enters or modifies the necessary information for the service.

4. User confirms the action to add, delete, or update the service.

5. The application processes the user's request and updates the service data in the backend.

Information for Steps:

1. Accessing the service management section through the application interface.

2. Identifying and selecting the specific service to be managed.

3. Inputting or modifying relevant information related to the service.

4. Confirming the action to add, delete, or update the service.

5. Backend processing of the user's request, including validation and database updates.

Preconditions: User is logged into the application and has appropriate permissions to manage services.

Postconditions: Service data is successfully added, deleted, or updated in the backend database.

Assumptions: The application provides an intuitive interface for users to interact with service data.

Requirements Met: Users can efficiently manage services within the application, ensuring data consistency and integrity.

Dashboard Enhancement with Visualization

Use Case Name: Dashboard Enhancement with Visualization

ID: DASH_VIS_01

Area: User Interface Development - Dashboard Improvement

Actor(s): UI/UX Designers, Front-end Developers, Data Analysts

Description: Enhance the dashboard interface by incorporating advanced data visualization tools to improve user experience and provide better data insights.

Trigger Event: The need for improved data representation and decision-making support is identified.

Steps Performed:

1. UI/UX Designer proposes new visualization features for the dashboard.

2. Front-end Developer integrates visualization libraries and tools into the dashboard.

3. The Data Analyst configures the data sources for accurate visualization.

4. The new visualization features are tested for usability and accuracy.

5. Feedback is collected from users and further adjustments are made if necessary.

Information for Steps:

1. Visualization requirements are gathered, and mockups are created.

2. Visualization components are developed and embedded into the dashboard.

3. Data integrity checks are performed to ensure accurate visual representation.

4. User testing is conducted to evaluate the effectiveness of the visualizations.

5. Iterative improvements are made based on user feedback.

Preconditions: The existing dashboard is functional with access to relevant data sources.

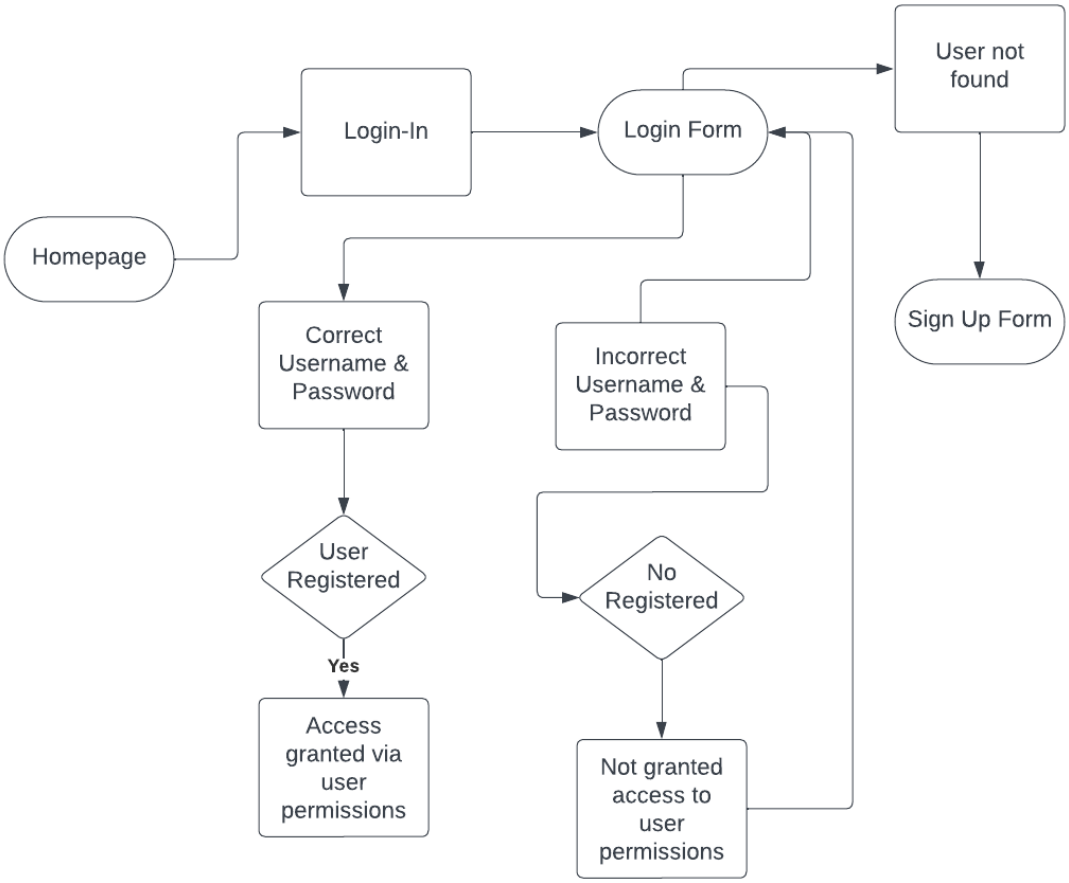
Postconditions: The dashboard has enhanced visualization features for data representation.

Assumptions: Users require graphical representations for better data interpretation.

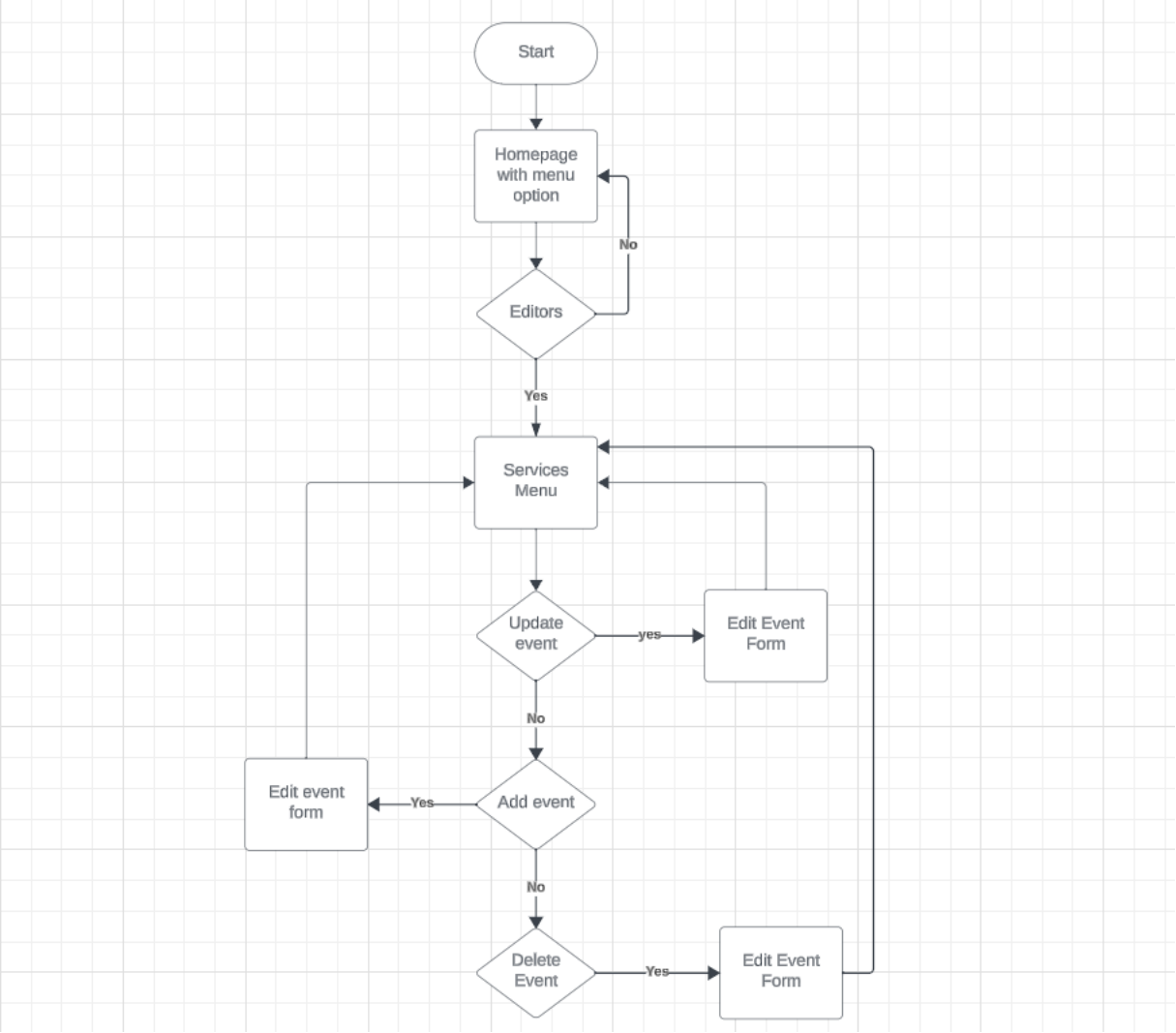
Requirements Met: The dashboard effectively displays data through enhanced visual tools.

User Flow Diagrams

User Authentication and Role-base Access Control



Event Services Flow Diagram



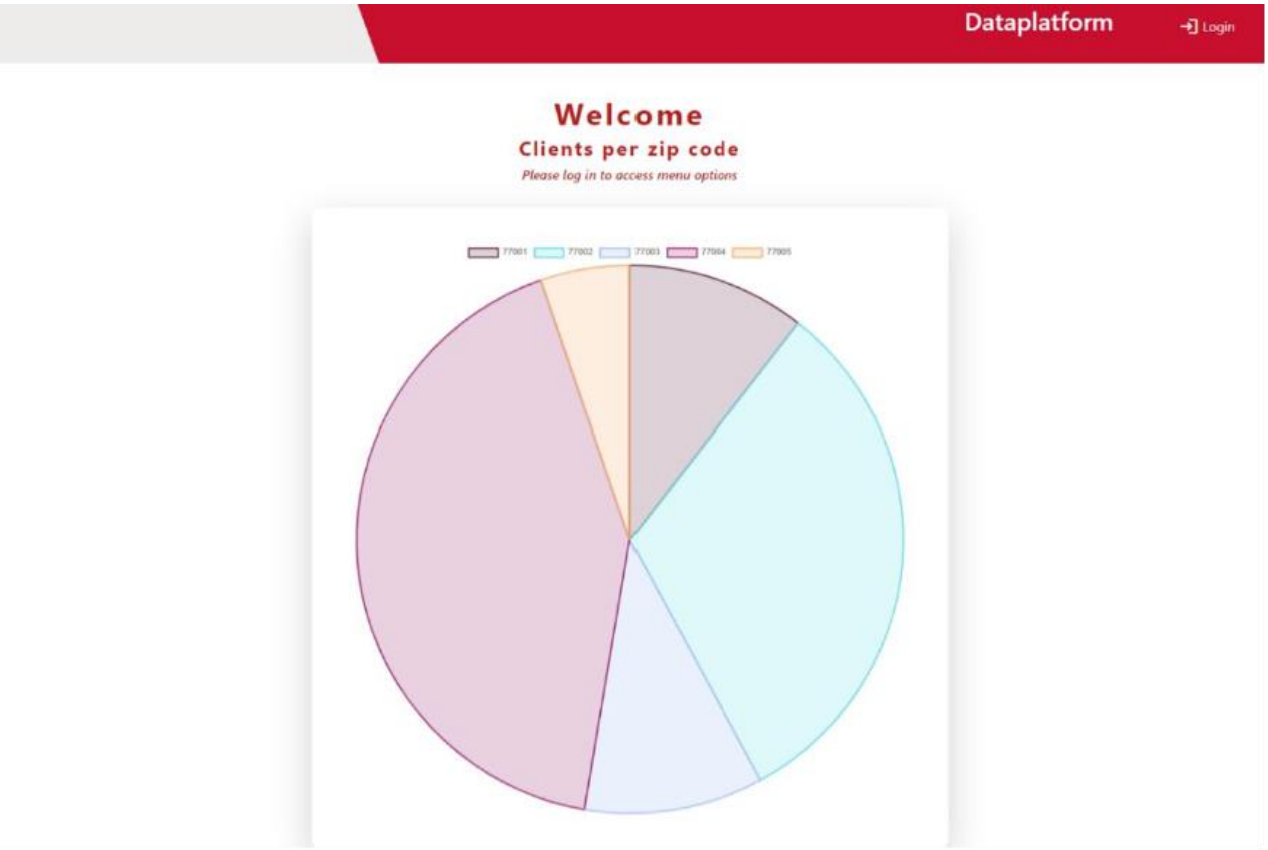
Timeline and Project Schedule

ID	Task Name	Duration	Start	Finish	Assigned Roles
0	**Project Timeline**	**70 Days**	**January 25th**	**May 5th**	
1	**CIS4339 Spring 2023 - Project**	**70 Days**	**January 25th**	**May 5th**	
2	**Sprint 1 - Compose Functional Specification Document**	**11 Days**	**January 25th**	**February 15th**	
3	Review Initial Application Functions and Code	13 Days	January 25th	February 6th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
4	Determine End-User Needs	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
5	**Determine Changes Required to Current UI to Satisfy Requirements**	**2 Days**	**February 6th**	**February 8th**	
6	Login and User Authentication Function	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
7	Dashboard Chart Function	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
8	Event Services Function	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
9	**Develop Use Case Scenarios for New Features**	**2 Days**	**February 6th**	**February 8th**	
10	Use Case for Login Functionality	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
11	Use Case for Event Services - CRUD	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
12	**Develop Flow Diagrams for New Features**	**2 Days**	**February 6th**	**February 8th**	
13	User Flow Diagrams for Login and Authentication Function	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
14	User Flow Diagrams for Event Services Function - CRUD	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
15	Develop Outline & Assign Roles	2 Days	February 6th	February 8th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
16	Review and Update Functional Specification Document	8 Days	February 8th	February 15th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
17	Submit Final Functional Specification Document	1 Day	February 15th	February 15th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
18	Sprint 1 Completion	1 Day	February 15th	February 15th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
19	**Sprint 2 - Implemenation of New Features in Front-End UI**	**21 Days**	**Febuary 16th**	**March 7th**	
20	**Dashboard**	**3 Days**	**Febuary 16th**	**February 18th**	
21	Update Dashboard Chart to have a Pie Chart Showing all Current Clients per Zipcode	3 Days	Febuary 16th	February 18th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
22	**Login and User Authentication Implementation**	**7 Days**	**February 19th**	**February 25th**	
23	Add Login Link on Home Page	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
24	Hide Menu Options for Unauthenticated Users	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
25	Create a Login-In Form to Authenticate All Users	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
26	Create a Prompt If User and/or Password is Incorrect	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
27	Create a Prompt When a User is Not Registered	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
28	Create a Function to Unhide Menu when Users are Authenticated	7 Days	February 19th	February 25th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
29	**Event Services Implementation**	**7 Days**	**Febuary 26th**	**March 3rd**	
30	Update UI Menu Options to Add	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
31	Create a view to list and search all current events	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
32	Implement UI to Add an Event Service	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
33	Implement UI to Update an Event Information	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
34	Implement UI to Delete an Event Information	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
35	Implement UI to Open an Event Information	7 Days	Febuary 26th	March 3rd	Klein, Trinity; Abejide, Maria; Macedo, Joanna
36	Review UI and Functionality Changes	3 Days	March 4th	March 6th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
37	Approve Implemented UI and Functionality Changes	1 Day	March 7th	March 7th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
38	Sprint 2 Completion	1 Day	March 7th	March 7th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
39	**Sprint 3 - Expansion of Backend API and Documentation and Presentation**	**36 Days**	**March 8th**	**April 30th**	
40	**Implemention of API and Documents**	**23 Days**	**March 8th**	**March 29th**	
41	Implement API Service to Get Clients Ordered By Zipcode on Pie Chart in Dashboard	8 Days	March 8th	March 15th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
42	Implement API Service to Authenticate Users Through a Username, Password, and User Groups	23 Days	March 8th	March 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
43	Implement API Service to Update Event Service	23 Days	March 8th	March 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
44	Implement API Service to Delete Event Service	23 Days	March 8th	March 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
45	Implement API Service to Create Event Service	23 Days	March 8th	March 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
46	Implement API Service to Read Event Service	23 Days	March 8th	March 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
47	Test API Functions	8 Days	March 30th	April 6th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
48	**Update API and Documents**	**7 Days**	**April 6th**	**April 12th**	
49	Update Dashboard API Documents	7 Days	April 6th	April 12th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
50	Update Login API Documents	7 Days	April 6th	April 12th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
51	Update Event Service API Documents	7 Days	April 6th	April 12th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
52	Review and Update API Implementations and Documents	5 Days	April 13th	April 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
53	Record and Edit Presentation Video of Project Features	5 Days	April 19th	April 29th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
54	Sprint 3 Completed	1 Day	April 30th	April 30th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
55	**Sprint 4 - Individual Peer Evaluations**	**5 Days**	**May 1st**	**May 5th**	
56	Evaluate Two Peers Project	5 Days	May 1st	May 5th	Klein, Trinity; Abejide, Maria; Macedo, Joanna
57	Spring 4 Completed	1 Day	May 5th	May 5th	Klein, Trinity; Abejide, Maria; Macedo, Joanna

Appendix A: Proposed UI Changes to Meet Requirements

Users that are not authenticated

- Menu options



Resources

Just a Moment..." Just a Moment.., www.indeed.com/career-advice/career development/functional-specification.

Kendall, Kenneth E., and Julie E. Kendall. Systems Analysis and Design. Prentice Hall, 2010.

Otto, Mark, et al. "Introduction." Bootstrap · The Most Popular HTML, CSS, and JS Library in the World, getbootstrap.com/docs/5.0/getting-started/introduction/.

Vue.js - The Progressive JavaScript Framework | Vue.js, vuejs.org/.