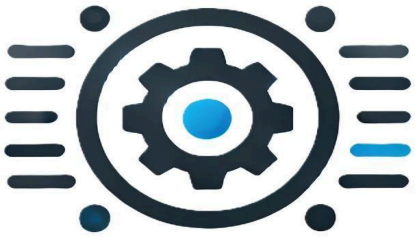
GCISLFullStackApp

Project Prototype Report

Sponsor: WSU Granger Cobb Institute for Senior Living (GCISL)



Justin Keanini,

Teni Olugboyega

Naomi Dion-Gokan

[10.19.2024]

Table of Contents

Introduction

I. Project Introduction

The Granger Cobb Institute for Senior Living (GCISL) at Washington State University is dedicated to advancing the field of senior living through research, education, and community engagement. As part of this mission, GCISL uses gciConnect, a web-based senior living resident volunteer report tracking system. That site was originally implemented by a student group in 2023 and improved by another group after that. GciConnect is a crucial tool to help faculty, staff and students track and manage volunteer reports for senior living residents, which contribute to ongoing research progress that has the objective to improve the quality of life for older adults.

However, GCISL seeks to enhance their existing system because it faces challenges in usability, reliability, and administrative capabilities, which interfere with its efficacy to meet the needs of the GCISL's stakeholders. By making feature enhancements such as the improvement of the interface, reliability, and administrative portal as adding other features, the team aims to create a more user-friendly and reliable tool that will support future research projects and make it easier for administrators to manage the platform and smoothly perform day-to-day operations.

II. Background and Related Work

The domain of this project falls within healthcare technology, specifically focused on senior living communities. The Granger Cobb Institute for Senior Living (GCISL) at Washington State University uses gciConnect, a web-based system, to manage and track volunteer reports for senior living residents. The broader context involves the intersection of gerontology, technology, and data management to improve the quality of life for older adults through research, education, and community engagement.

Senior living and healthcare technology have increasingly embraced digital systems to streamline operations, enhance resident care, and support research. Web-based systems like gciConnect play a crucial role in data collection, volunteer management, and research tracking. However, many systems in this domain face challenges related to usability, reliability, and administrative management, similar to the ones faced by gciConnect

To successfully enhance gciConnect, the following technical knowledge and skills will be necessary:

Web Development and Frameworks: Familiarity with front-end frameworks (e.g., React or Vue.js) to enhance the user interface and improve usability.

Database Management: Understanding of database systems (e.g., MySQL or MongoDB) to ensure the backend reliably manages volunteer report data and facilitates smooth administration.

Security and Compliance: Knowledge of healthcare data security, especially related to HIPAA compliance, to ensure that data collected through gciConnect is managed securely.

User Experience (UX) Design: Learning best practices in UX/UI design to improve the overall user experience, ensuring that administrators and users find the system intuitive and accessible.

API Integration: Skill in working with RESTful APIs to potentially integrate external services or tools that could enhance the functionality of gciConnect.

III. Project Overview

This volunteer report tracking system was designed as a capstone project for the Granger Cobb Institute for Senior Living (GCISL). To fully meet the requests of the volunteers, staff, and faculty at GCISL, the following areas need to be improved in the system:

- *Usability*: The current user interface lacks inclusive accessibility features, which makes it difficult for all users (particularly senior living residents and volunteers) to interact with the system. Improving the system's user interface and overall user experience is critical for ensuring better accessibility/ease of use.
- *Reliability*: The system should include the implementation of automated tests to ensure stable functionality, error tracking to help administrators quickly address issues, and Application Performance Monitoring to ensure availability and reduce downtime.
- *Admin Portal Improvements*: The administrative portal needs to be enhanced to provide better system management capabilities. This includes giving administrators tools to track performance, manage users, and troubleshoot issues.
- *Feature Enhancements*: Several features requested by the GCISL team were either partially implemented or missed in the original version. These gaps must be addressed by identifying, developing, and integrating new features.

The goal of the project is to create a highly usable, reliable, and feature-complete system that satisfies the needs of all stakeholders. Specifically, the outcomes are:

- The system should provide an easy-to-use, accessible interface for volunteers, staff, and residents, with particular attention to responsiveness and mobile compatibility.
- By implementing automated testing and logging, the system should be stable to use in daily operations. Administrators should be able to monitor system performance and availability using APM tools.

- The enhanced administrative portal should allow GCISL staff to manage users, assist users who face issues, and monitor system performance in real time.
- The project will include the development of comprehensive documentation and training materials so that GCISL staff and faculty can effectively manage and maintain the system after deployment.

IV. Client and Stakeholder Identification and Preferences

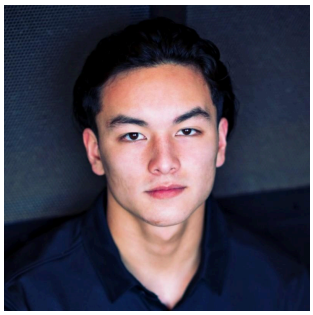
Our primary client is Washington State University's Granger Cobb Institute for Senior Living (GCISL). Our primary liaison contact will be Mrs Darcie Bagott, GCISL's Program Specialist. The final project will be predominantly used by GCISL faculty and staff members, who will oversee the volunteer report tracking system to better serve the institute's goals and ongoing research initiatives. Also, the GCISL residents and their families will be using the system as well.

The needs of our client and stakeholders are an improved system with a more intuitive user interface, enhanced administrative control, and improved system reliability to support their research on senior living. Some preferences include hands-on experience in managing and accessing senior living data as well as a robust admin portal with advanced features like rolling logs and user assistance tools to address any system issues effectively.

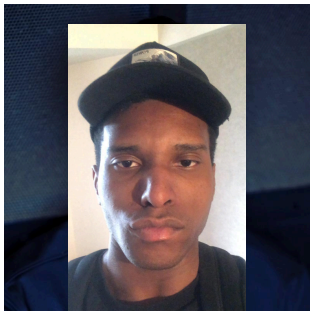
There are multiple stakeholders involved in this project. Those stakeholders are the GCISL faculty and staff, who rely on the system for data tracking and research purposes; both students studying Senior Living Management and those involved in research, the administrators who manage the system. Not to forget the research participants, whose data will be used.

Potential clients can be other senior living institutions in Eastern Washington or Washington as a whole.

Team Member Introduction



Justin Keanini is a senior at Washington State University pursuing a degree in Computer Science. With a strong focus on frontend development, he has experience building responsive and user-friendly interfaces using HTML, CSS, and JavaScript. He brings a solid foundation in programming, gained through coursework in C/C++/C#, and certifications in Python and R for data analysis and visualization. For this project, he expanded his skills by working with React for dynamic UI development and MongoDB for backend data integration. Justin contributed to designing and implementing intuitive user interfaces while collaborating with the team to integrate and refine system features, ensuring the application meets both functional and usability goals.

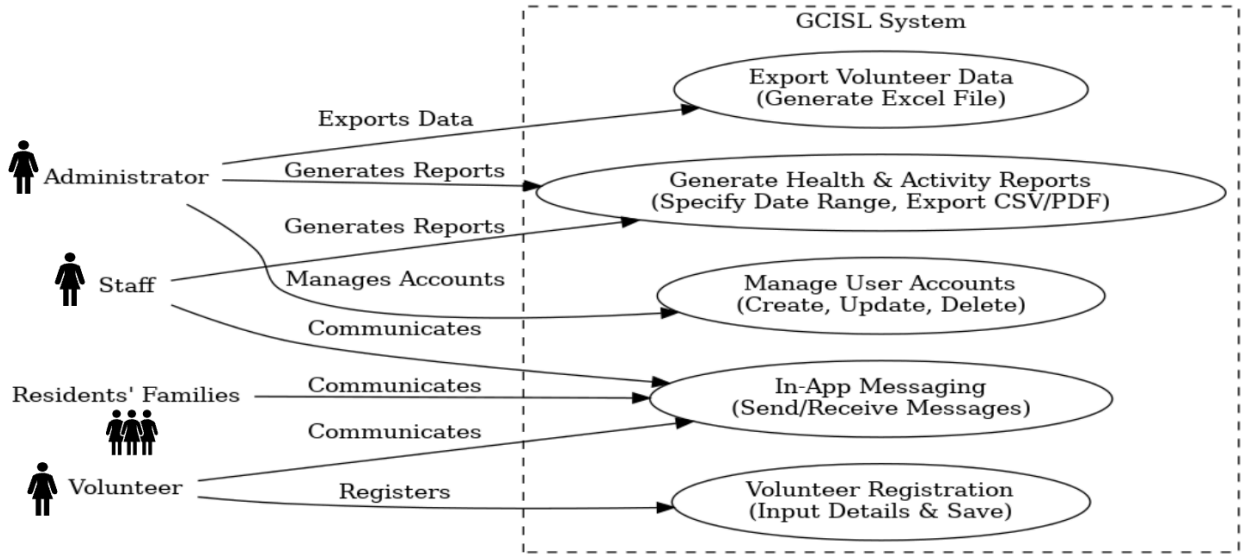


Teni Olugboyega is a senior at Washington State University majoring in computer science graduating in Spring 2025. He has an interest in full-stack development, Artificial intelligence and Machine Learning. He has strong background in java, React and MERN full stack development which is essential for the completion of the project. For this project Teni played a role in constructing visually appealing pages that follows the clients specified theme, connecting frontend pages to the backend and database to make use of data stored ,and retrieve to add more definition to some pages .

Naomi Dion-Gokan is a senior at Washington State University, majoring in Computer Science and set to graduate in Spring 2025. She is passionate about web development, especially front-end engineering, and is excited to expand her skills into back-end work with the goal of becoming a full-stack software engineer. Naomi has developed expertise in C#, React, JavaScript, and SQL, particularly in building scalable full-stack solutions—skills she earned over two years during her internship. For this project, Naomi's responsibilities include setting up the database, managing deployment, and facilitating team communication. Her technical toolkit also includes HTML, CSS, C++, and REST APIs, giving her a strong foundation for both project organization and technical implementation.

Project Requirements

V. Use Cases



- **Use Case 1: Managing User Accounts**

Actors: Administrator

Description: An administrator manages user accounts by creating, updating, and deleting profiles. This ensures that only authorized individuals can access the system, and their information is always up to date.

Preconditions: The user is logged in as an administrator and has access to the User Management Module.

Main Flow:

The administrator selects the “Manage Users” option from the admin dashboard.

The administrator selects “Add New User” and inputs required details such as name, contact information, and role.

The administrator can also select an existing user, update their information, or delete the user from the system.

The system saves the changes and updates the user database.

Postconditions: The system updates the user list, reflecting the newly added, updated, or deleted profiles.

Exceptions: If the user does not have the required admin access, the system denies access to user management functionality.

Related Requirements:

User Management Module, Requirement 1: User profile management (creation, updating, deletion).
User Management Module, Requirement 2: Role-based access control.

- **Use Case 2: Generating Resident Health and Activity Reports**

Actors: Administrator, Staff

Description: An administrator or staff member generates detailed reports of resident health and activity based on data collected over time. This helps in tracking trends and making informed decisions about care.

Preconditions: The administrator or staff has access to the Data Analysis and Reporting Module. Main Flow:
The user selects the “Generate Report” option from the reporting dashboard.

The user selects a specific resident or group of residents, and specifies the date range and type of data (health or activity).

The system compiles the data and generates the report.

The user can view the report on-screen or choose to export it as a CSV or PDF file.
Postconditions: The report is displayed and optionally exported in the selected format.
Exceptions: If no data is available for the selected parameters, the system displays an error message.

Related Requirements:

Data Analysis and Reporting Module, Requirement 1: Generate resident health and activity reports.

Data Analysis and Reporting Module, Requirement 2: Export reports in CSV or PDF.

- **Use Case 3: In-App Messaging for Communication**

Actors: Staff, Volunteers, Residents' Families

Description: Staff members communicate with external stakeholders such as volunteers or residents' families through the in-app messaging system, improving coordination without relying on external platforms.

Preconditions: The user has access to the Communication Module and appropriate permissions to send and receive messages.

Main Flow:

The user opens the messaging feature within the system.

The user selects a recipient or group (staff, volunteers, or residents' families).
The user types a message and sends it through the system.

The recipient receives and responds to the message within the system.

Postconditions: Messages are sent and received, enabling real-time communication between stakeholders.

Exceptions: If the recipient is not available, the system notifies the user that the message was not delivered.

Related Requirements:

Communication Module, Requirement 1: In-app messaging.

- **Use Case 4: Volunteer Registration**

Actors: Volunteers, Administrator

Description: Volunteers input their personal information through a form on the website, enabling GCISL to maintain an updated volunteer database.

Preconditions: The volunteer has access to the Volunteer Management Module. Main Flow:
The volunteer accesses the registration form from the system's website.

The volunteer enters their name, contact details, availability, and other requested information. The system validates the input and saves the volunteer's information to the database. Administrators can access the information through the volunteer management dashboard.
Postconditions: The volunteer's information is stored in the system and accessible to administrators.

Exceptions: If the form is incomplete or contains errors, the system prompts the user to correct the input.

Related Requirements:

Volunteer Management Module, Requirement 1: Input volunteer

information.

- **Use Case 5: Exporting Volunteer Data**

Actors: Administrator

Description: The administrator exports the volunteer database into an Excel file to analyze the data or share it with external parties.

Preconditions: The administrator has access to the Volunteer Management Module.

Main Flow:

The administrator selects “Export Volunteer Data” from the admin dashboard.
The system generates an Excel file with the details of all registered volunteers. The administrator downloads the file for further use.
Postconditions: The Excel file is downloaded, containing the latest volunteer data.

Exceptions: If there is a system error during export, the system displays an error message and logs the issue.

Related Requirements:

Volunteer Management Module, Requirement 2: Export volunteer information in Excel format.

VI. Functional Requirements

1. User Management Module

Requirement 1: The system must allow for the creation, updating, and deletion of user profiles.

Description	The system will provide an interface for administrators to manage user accounts. This includes adding new users, editing their information (such as name, contact details, and role), and deleting users when they are no longer part of the system. This functionality is essential to ensure that only authorized users have access to the system and that user information remains accurate.
Source	GCISL
Priority	Priority Level 0: Essential and required functionality

Requirement 2: The system must support different user roles with corresponding access levels.

Description	The system will differentiate between user roles, such as administrators, staff, and volunteers. Each role will have specific permissions regarding what they can view and modify in the system. For example, administrators will have full access, while volunteers may only be able to update their own information. This ensures secure access control based on user responsibility.
Source	GCISL
Priority	Priority Level 0: Essential and required functionality

2. Data Analysis and Reporting Module

Requirement 1: The system must generate reports based on resident date, including health and activity information.

Description	The system will collect and organize resident data (e.g., health reports, activity logs) and allow administrators to generate detailed reports. These reports will help staff monitor resident health trends and daily activities, enabling better decision-making and care planning.
Source	GCISL
Priority	Priority Level 1: Desirable functionality

Requirement 2: The system must allow exporting of reports in CSV and PDF formats.

Description	The system will include a feature for exporting generated reports into widely used file formats, such as CSV and PDF. This will allow administrators to easily share data with external parties, such as healthcare providers, or for offline record-keeping.
Source	GCISL
Priority	Priority Level 0: Essential and required functionality

3. Communication Module

Requirement 1: The system must support in-app messaging between staff and stakeholders.

Description	The system will provide a messaging feature that enables real-time communication between staff members and external stakeholders (such as volunteers or residents' families). This will streamline internal communication and reduce the need for external messaging platforms, improving the efficiency of communication workflows.
Source	Internal requirements elicitation among members of the team
Priority	Priority Level 2: Extra features or stretch goals

4. Volunteer Management Module

Requirement 1: The system must provide a form on the website for volunteers to input their information, including name, email, and contact details.

Description A dedicated web form will be available for volunteers to register their details, including personal contact information and their availability for volunteering. This will allow GCISL to maintain an organized database of volunteers, ensuring that they can be contacted and scheduled for tasks easily.

Source	GCISL
Priority	Priority Level 0: Essential and required functionality

Requirement 2: The system must allow administrators to export the collected volunteer information into an Excel file.

Description	Administrators will have the ability to export the volunteer database into an Excel file, providing flexibility for data analysis, reporting, and manual editing. This feature ensures that volunteer data can be easily shared with other departments or used for event planning and coordination.
Source	GCISL
Priority	Priority Level 0: Essential and required functionality

6. Other Modules

System Reliability Improvements

Description	Implement tests to check for errors and ensure continuous functionality during updates.
Source	Developer best practices
Priority	Priority Level 0: Essential a

Error Logging:

Description	Add error tracking and logging for rapid debugging by administrators.
Source	GCISL administrators
Priority	Priority Level 0: Essential

VII. Non-Functional Requirements

Reliability

The system shall be reliable and operational most of the time. It must handle user requests without crashing or becoming unresponsive. This should be GCISL's requirement for a stable system to support ongoing research and volunteer tracking.

Response Time

The system shall process and respond to user actions such as login, task assignments within a few seconds. For larger operations, like generating reports, the system shall complete the task within less than 10 seconds. Usability needs for staff and volunteers.

Scalability

The system shall support scalability, accommodating a large number of concurrent users without degradation in performance. This ensures that future expansions to other institutions or larger datasets will be manageable. It would be GCISL's potential expansion to other senior living institutions.

Usability

The system shall be intuitive and easy to use, requiring no more than a few hours of training for an administrator or volunteer to perform basic tasks such as assigning tasks, submitting volunteer reports. The need for accessibility and ease of use for staff, volunteers, and senior residents.

Maintainability

The system shall be modular and well-documented to ensure ease of updates, troubleshooting, and adding new features. New developers should be able to understand the system within a few days of reading the documentation. Requirement for future enhancements and system evolution.

Data Storage and Retention

The system shall store data efficiently, allowing for a minimum of few years of resident and volunteer reports while maintaining optimal performance. Compliance with data retention

VIII. System Evolution

The project is based on many assumptions:

- **Flexible and extensible design**

The system will be designed with flexibility and adaptability, allowing for easy future updates, refinements, and extensions. Given that GCISL's needs may evolve in the future, the architecture must support adding new features (example: additional reporting tools) without requiring a complete rebuild of the system. Future research initiatives may introduce new data requirements, necessitating extensions to the data analysis or reporting modules. However, without proper modular design, future changes could become costly and difficult to implement, leading to technical debt.

- **Several and consistent user testing and feedback integration as well evolving user needs**

The system will go through regular user testing, especially with staff, volunteers, and senior residents. Based on feedback, the interface, features, and overall usability will be refined to meet evolving user expectations. User feedback may highlight usability or feature gaps that were previously overlooked, requiring new iterations to improve the system. However, failing to incorporate feedback quickly enough could result in poor user adoption or require costly adjustments post-deployment.

- **Adaptability for broader use**

The system is primarily created for use within GCISL, but with future iterations, we might need to adapt it to other senior living institutions or expand functionality for different research

projects. This means the system should be scalable in terms of both user management and data processing capabilities. The system might be adopted by other institutions, necessitating changes in the database architecture to handle a higher volume of users and reports. However, if the system is not adaptable, it may require significant redevelopment efforts to accommodate a larger user base, delaying deployment in new environments.

Solution Approach

IX. Project Design Introduction

Our team aims to explore the potential of digital tools for senior living management by rebuilding the gciConnect system from scratch. Originally developed by a capstone team in 2023, the platform was intended to support the Granger Cobb Institute for Senior Living's (GCISL) research efforts by tracking volunteer activities. However, the original system faced usability, reliability, and administrative challenges, limiting its effectiveness for staff, students, and volunteers.

As part of this project, we have successfully redeployed the platform with a fully functional Contact Us page. This feature allows users to submit inquiries directly through the platform, triggering automated notifications to administrators for timely follow-up and efficient communication. This section of the document serves as a comprehensive overview of the inter-relational design of the gciConnect platform, focusing on three main areas: Architecture, Data, and User Interface.

- Architecture: Each subsystem will be explored, with a focus on concepts, algorithms, and interface properties that ensure the system's smooth operation.
- Data: Key data types and relationships will be mapped to ensure consistent data flow throughout the platform.
- User Interface: Mocked versions of key pages will outline their components and functionality, ensuring accessibility and ease of use.

This document serves as a structured guide for developers, ensuring alignment with project goals. Developers will conduct verification tests based on the following:
i. Subsystem descriptions
ii. Subsystem relationships
iii. Data type objects
iv. Data and database relationships
v. Database schema
vi. User interface mocked pages

Additionally, the document provides stakeholders with a reference to ensure that the design aligns with both operational and research needs. Our goal is to deliver a stable, user-friendly, and reliable platform that empowers GCISL faculty, staff, and volunteers to manage activities seamlessly, enabling them to focus on research efforts and improving the quality of life for residents.

X. System Overview

The gciConnect platform is designed to streamline the management of volunteer activities and support the research efforts of the Granger Cobb Institute for Senior Living (GCISL). It enables

staff, volunteers, and administrators to coordinate tasks, track engagement, and maintain communication, ensuring effective operations and meaningful research outcomes.

The design of the platform revolves around three core areas:

- Architecture: The system is divided into subsystems, such as User Management, Volunteer Management, Communication, and Reporting. The interaction between these subsystems ensures that tasks, data, and notifications flow smoothly throughout the platform.

- Data: The system uses MongoDB to store key data objects, including users, tasks, and activity logs. Relationships between data entities are structured to facilitate easy retrieval and reporting, ensuring consistency and reliability. The Contact Us page data is collected through Google Forms and stored in Google Sheets for easy tracking and access by administrators.

- User Interface: The interface is built to be accessible and intuitive, supporting multiple roles such as staff, volunteers, and administrators. Key pages, such as the Contact Us page, allow seamless communication, with automatic notifications sent to administrators upon message submission. Additionally, a new information feed will be implemented to keep users updated with relevant announcements and events.

The following sections will provide detailed insights into the platform's design. Architecture Design will cover the subsystems and their interactions, Data Design will focus on the database schema and relationships, and User Interface Design will describe key page layouts and functionality. Together, these elements ensure that the enhanced gciConnect platform will be reliable, user-friendly, and scalable, effectively supporting GCISL’s operations and research initiatives.

XI. Architecture Design

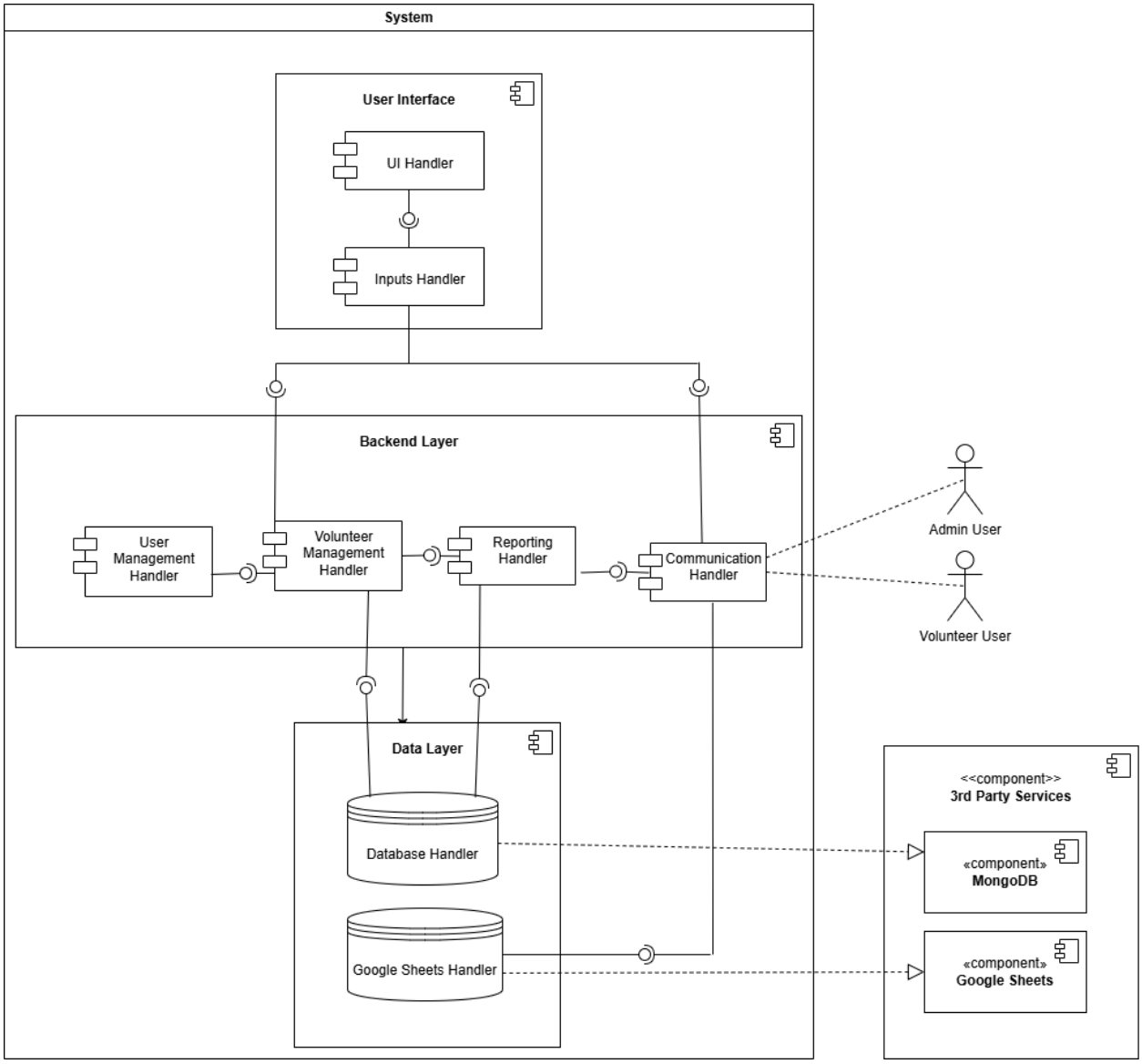
1. Overview

Our team has adopted a **component-based architectural pattern** for the gciConnect platform to support the volunteer management and research efforts of the Granger Cobb Institute for Senior Living (GCISL). This pattern ensures modularity, allowing each subsystem to function independently while seamlessly interacting with others. The platform’s architecture consists of three layers: User Interface Layer, Backend Layer, and Data Layer. Each layer plays a specific role in ensuring the platform remains scalable, maintainable, and reliable.The User Interface Layer handles interactions between users and the system. The **UI Handler** manages layouts, page transitions, and user navigation, ensuring an accessible experience for all users. Working alongside the UI Handler, the **Inputs Handler** validates input data, such as form submissions, and routes it to the appropriate backend subsystems.

The Backend Layer manages system logic. The User Management Handler oversees accounts and permissions, ensuring proper access to features. The Volunteer Management Handler assigns tasks to volunteers and tracks their activities, sending status updates to the Reporting Handler for logging and to the Communication Handler for notifications. The Communication Handler sends notifications and logs inquiries received from the Contact Us page.The Data Layer ensures efficient data storage and retrieval. The Database Handler stores key data in

MongoDB Atlas and logs inquiries in Google Sheets. This combination allows for both structured data storage and real-time tracking of communication.

Below is a diagram showing the system's structure. This design will allow future development without disrupting other components. In the next sections, detailed descriptions of the individual subsystems, algorithms, and services will be provided.



2. Subsystem Decomposition

2.1. [User Management Handler]

a) Description

The User Management Handler manages user accounts, roles, and permissions. It ensures that only authorized users can access specific features, maintaining security and data integrity.

b) Concepts and Algorithms Generated

This subsystem applies Role-Based Access Control (RBAC) to manage different user roles (e.g., volunteers, staff, and administrators) and uses session management algorithms to validate user access during login.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
User Account Management	Volunteer Management Handler	Provides user account data (such as roles and permissions) to facilitate the assignment of tasks to volunteers.
Role Verification	Communication Handler	Verifies roles and permissions to ensure notifications and messages are routed to the appropriate administrators and users.

Services Required: None (self-contained).

2.2. [Volunteer Management Handler]

a) Description

The Volunteer Management Handler manages volunteer tasks and engagement, ensuring that volunteers are assigned tasks and their activities are tracked.

b) Concepts and Algorithms Generated

This handler implements a task scheduling algorithm to assign tasks based on volunteer availability and a tracking system to log hours and monitor task progress.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Task Management and Tracking	Reporting Handler	Sends task completion data and volunteer activity logs to the Reporting

		Handler for record-keeping and report generation.
Status Notification	Communication Handler	Sends updates on task status to the Communication Handler to notify administrators and relevant users about task progress or issues.

Services Required:

Service Name	Service Provided From
Account and Role Management Description: Uses user data to assign tasks to volunteers.	User Management Handler

2.3 [Communication Handler]

a) Description

The Communication Handler manages notifications and internal communication, ensuring administrators are informed of important updates or new submissions through the Contact Us page.

b) Concepts and Algorithms Generated

This handler applies notification algorithms to send automated emails to administrators and stores message logs in Google Sheets through integrated forms.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Notification Management	Admin Users via Email Notifications	Sends notifications upon task updates or message submissions and stores logs in Google Sheets.

Services Required:

Service Name	Service Provided From
Account and Role Management	User Management Handler

Description: Ensures notifications are routed to the correct administrators.	

2.4. [Reporting Handler]

a) Description

The Reporting Handler aggregates data from other subsystems and generates reports for administrators. It allows exporting of volunteer data to Excel for analysis and sharing.

b) Concepts and Algorithms Generated

This handler uses data aggregation algorithms to compile logs and export functionality to generate downloadable Excel files for offline use.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Data Reporting and Export	Admin Users	Aggregates volunteer activity logs and allows exporting to Excel for reporting purposes.

Services Required:

Service Name	Service Provided From
Task Management and Tracking Description: Uses volunteer data to generate reports.	Volunteer Management Handler

2.5. [Database Handler]

a) Description

The Database Handler manages persistent storage for the platform using MongoDB Atlas and Google Sheets integration. It ensures seamless storage and retrieval of data across all subsystems.

b) Concepts and Algorithms Generated

MongoDB is used to store users, tasks, and volunteer logs, while Google Sheets stores Contact Us submissions. The handler ensures consistent data flow between the frontend and backend.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Data Storage	Volunteer Management Handler	Stores volunteer activity data and task assignments to ensure seamless management and updates across the system.
Data Retrieval	Reporting Handler	Provides access to stored task and activity logs to generate reports for administrators. Also, export data into Excel files.

Services Required: None (directly integrated with other handlers).

2.6. [UI Handler]

a) Description

The UI Handler manages all user-facing elements, including page layouts, icons, and navigation. It ensures that the user interface is responsive and intuitive for all users.

b) Concepts and Algorithms Generated

The UI Handler applies dynamic layout algorithms to adjust the interface for different devices and users. It also handles navigation flow across various sections of the platform.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Layout Management	Inputs Handler	Provides layout updates to align input elements and navigation flow.

Services Required: None (interacts directly with the frontend interface).

2.7. [Inputs Handler]

a) Description

The Inputs Handler processes data input from users, such as form submissions and interactive elements. It ensures that data is validated and routed to the appropriate handlers.

b) Concepts and Algorithms Generated

This subsystem applies input validation algorithms to ensure correct data entry and routes the inputs to backend services such as task assignment or notifications.

c) Interface Description

Services Provided:

Service Name	Service Provided To	Description
Input Processing	Communication Handler	Task Input Routing
Task Input Routing	Volunteer Management Handler	Processes input related to task assignments and routes it to the Volunteer Management Handler for scheduling and tracking.

Services Required:

Service Name	Service Provided From
Layout Management Description: Uses layout updates to ensure inputs are aligned with the user interface.	UI Handler

XII. Data design

For this application, two primary data structures are utilized, both interacting with external subsystems: MongoDB and Google Sheets. Each plays a crucial role in data management and retrieval for different parts of the system.

1. **MongoDB Data Structures**

MongoDB serves as the primary database for managing system users, volunteer profiles, activities, and reports. The database is structured around collections that store documents in a flexible, JSON-like format. Below is an overview of the key collections and their internal data structures:

userId: Unique identifier for each user.

name: Full name of the user.

email: Email address for communication.

role: User role determining access level.

contact: Phone number or other contact details.

createdAt / updatedAt: Timestamps for tracking the user's record creation and updates.

User Collection

This collection stores user accounts, including administrators, staff, and volunteers, with role-based access control

Google Sheets Data Structure

Google Sheets serves as a complementary tool for Contact Us form submissions and volunteer activity tracking. It allows administrators to access certain data quickly and interact with it via familiar spreadsheet interfaces.

Contact Us Submissions Sheet

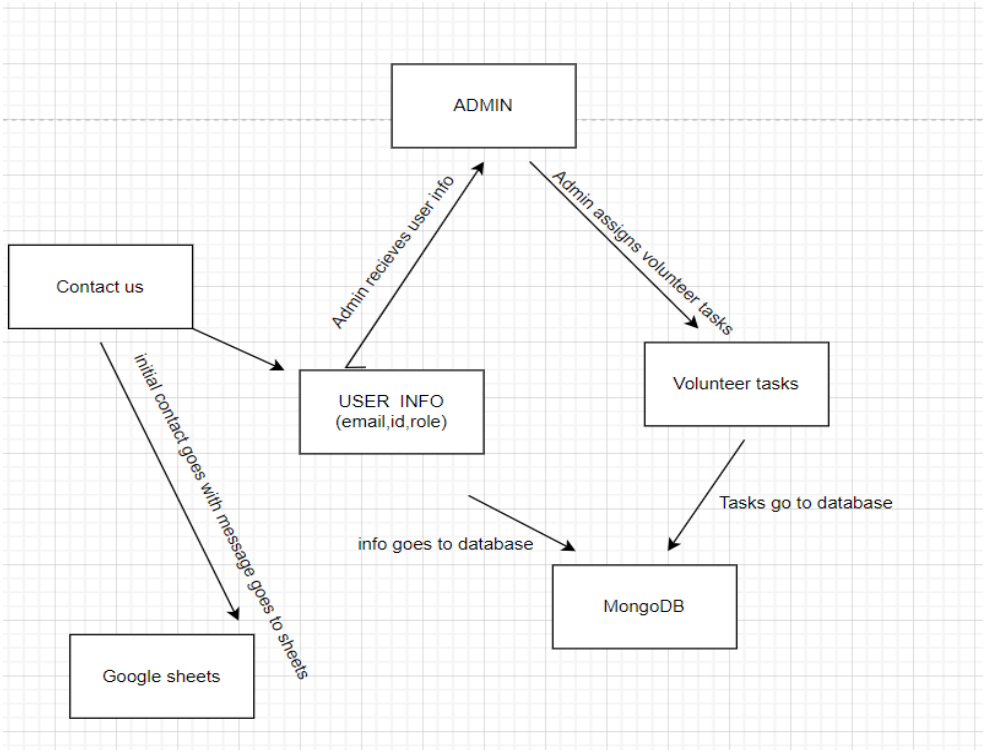
Stores information submitted by volunteers via the contact form.

- **Columns in Google Sheets:**
  - **Name**: Full name of the person submitting the form.
  - **Email**: Contact email.
  - **Message**: The content of the message or inquiry.
  - **Date**: Date of submission.
- **Usage**:
  - Admins access this sheet to review submissions and respond directly via email if required.
  - This data is not stored long-term in MongoDB, as it mainly serves as a communication channel.

Volunteer Tracking Sheet

Stores high-level summaries of volunteer participation for offline analysis and event planning.

- **Columns in Google Sheets:**
  - **Volunteer Name**: Full name of the volunteer.
  - **Activity Type**: Type of the activity the volunteer participated in.
  - **Date**: Date of the activity.
  - **Hours**: Number of hours contributed by the volunteer.
  - **Notes**: Additional notes on participation.
- **Usage**:
  - Admins export volunteer data from MongoDB into this Google Sheet for easy sharing with external teams or event organizers.
  - The sheet acts as a temporary data store to facilitate collaboration without requiring direct database access.



XIII. User Interface Design

The GCISL Full Stack Application features a user interface designed to be intuitive, accessible, and user-friendly for administrators, staff, volunteers, and other stakeholders. The UI focuses on simplicity and efficiency, ensuring that users can navigate the system and perform essential tasks with minimal effort. Each page has been designed with accessibility and clarity in mind, ensuring that users of all technical backgrounds, particularly those in the senior living community, can easily interact with the system.

Home Page[Image 1]:
The Home Page serves as the entry point for all users, providing a brief overview of the system's features and quick access to important sections. It includes:

- Navigation Bar: Links to the main sections of the website, such as Get Involved, Contact Us, About, Login, and Register.
- Welcome Message: A brief introduction to GCISL's mission and purpose.
- Featured Opportunities/News Feed: A section highlighting current volunteering or research involvement opportunities, with quick links to more detailed information.

Get Involved Page[Image 2]:
The Get Involved Page serves as the primary hub for volunteers and community members looking to contribute to GCISL activities. It includes three main sections:

- Research Involvement: Provides opportunities for volunteers to participate in ongoing research projects.
- Education and Mentorship: Lists opportunities for users to contribute through educational programs and mentorship roles.
- Outreach and Charitable Contributions: Allows users to engage in charitable activities and community outreach programs.

This page is designed to be engaging and easy to navigate, with large, interactive buttons and concise descriptions to guide users through available opportunities. Users can quickly explore the roles they are interested in and sign up with minimal effort.

Contact Us Page[Image 3]:
The Contact Us Page is designed to streamline communication between users and the administration. It includes:

- Use Case: Contact Us
  Description: Users submit inquiries or feedback to GCISL administration.
  Interface Interaction:
  Users fill out the contact form on the Contact Us Page with their name, email, and message.
  After submitting the form, the message is sent to administrators, and users receive an automated confirmation.
  This page is simple enough for users of all technical skill levels to navigate, with a clean and straightforward layout. It ensures that users can easily get in touch with GCISL staff for any inquiries or issues they may have.

About Page[Image 4]:
The About Page provides an overview of the GCISL organization, its mission, and its past activities. The page includes:

- FAQ (Frequently Asked Questions): Answers to common questions regarding the organization and its services.
- Picture Gallery: A visual gallery of images from past events and activities at GCISL.

This page is visually appealing and informative, designed to give visitors a comprehensive understanding of the organization's purpose and initiatives. The combination of text and images ensures a user-friendly and engaging experience.

Login Page[Image 5]:
The Login Page provides a straightforward mechanism for users to access the system. It includes:

- Use Case: User Login
  Description: Users log into their accounts to access personalized features and services.

Interface Interaction:
Users navigate to the Login Page, where they enter their username and password.
The system authenticates the credentials and redirects users to the Home Page.
This page has been designed with accessibility in mind, featuring large text fields and a clear error message system that provides immediate feedback in case of invalid login attempts. The design uses bold, easily readable fonts to ensure that the login process is seamless for all users.

Register Page[Image 6]:
The Register Page allows new users to create an account and join the GCISL community. It includes:

- Use Case: User Registration
  Description: New users can create an account to join the GCISL community.
  Interface Interaction:
  Users access the Register Page to fill out personal information fields (name, contact details, password).
  Upon submission, the system creates a user profile and sends a confirmation email.
  The registration process is designed to be quick and easy, with large input fields and clear instructions. The page is simple, ensuring that new users can sign up without encountering difficulties.

Admin View Page[Image 7]:
The Admin View Page is the central hub for administrators to manage volunteers, tasks, and reports within the GCISL system. After logging in, administrators are presented with a dashboard overview of key activities and controls for managing the platform. The page is designed to provide a clear and concise overview of all admin functions, ensuring efficient workflow and management.

- Use Case: Admin Task Management
  Description: Admins can view, assign, and manage tasks for volunteers, ensuring efficient allocation of resources.
  Interface Interaction:
  Admins access a list of all current tasks, with options to assign or reassign tasks to specific volunteers.
  They can add new tasks with relevant descriptions and deadlines, as well as mark tasks as complete or update their statuses.
  The page provides a summary of active, completed, and pending tasks, making it easy for admins to track volunteer contributions and project progress.
- Use Case: Volunteer Reporting
  Description: Admins can generate detailed reports on volunteer activity and task completion to assess engagement and productivity.

Interface Interaction:
The Admin View Page includes options for filtering reports by time period, volunteer, or task type.
Reports can be exported in various formats (e.g., CSV, PDF) to allow for offline analysis and sharing with other stakeholders.

- Use Case: System Notifications
  Description: Admins receive real-time notifications about critical system events, such as new volunteer registrations or task completion updates.
  Interface Interaction:
  Notifications appear as alerts on the Admin View dashboard, allowing admins to quickly address any issues or take action on pending items.
  This feature ensures that admins are kept up-to-date with important activities within the system, improving response times and overall efficiency.

Accessibility and Usability:
The GCISL Full Stack Application is designed with accessibility in mind to meet the specific needs of the senior living community. Key features include:

- High Contrast and Large Fonts: These design elements ensure that the interface is easy to read, even for users with visual impairments.
- Clear Navigation: Simple, intuitive navigation ensures that users can move between different sections of the site without confusion.
- Feedback and Error Handling: The system provides real-time feedback through error messages and confirmation prompts to guide users through tasks and prevent mistakes.

Testing and Acceptance Plans

XIV. Testing and Acceptance Plans Introduction

1. Project Overview

The GCISL Full Stack App is a web-based system designed to streamline volunteer management and reporting for the Granger Cobb Institute for Senior Living. Its main functions include collecting volunteer information, generating participant reports, and providing an intuitive platform for administrators to view and manage data. The system emphasizes usability and security to ensure data privacy and ease of navigation, meeting the needs of administrators who rely on timely access to volunteer data. To ensure functionality and reliability, this document details a comprehensive testing plan, including both automated and manual testing approaches.

2. Test Objectives and Schedule

The primary objective of this testing plan is to ensure that the GCISL Full Stack App meets all functional and nonfunctional requirements outlined by GCISL, verifying that each feature operates as expected, performs consistently across environments, and upholds security standards.
This objective will be met by combining automated and manual testing to thoroughly evaluate functionality, performance, and user experience. Automated testing, including both unit and integration tests, will cover core features such as user registration/login, volunteer reporting, and data retrieval, accounting for both typical and edge cases to catch unexpected issues. Manual testing will focus on usability and functional aspects to confirm that the user interface is intuitive and meets stakeholder expectations. Acceptance testing will be conducted with GCISL stakeholders, confirming that the final product fulfills all requirements and meets both user experience and performance standards.

To support this approach, the project will follow a structured CI/CD pipeline using GitHub Actions, integrated with our Vercel hosting to automate testing, merging, and deployment processes, providing continuous validation of each build. If any issues arise with GitHub Actions or our CI/CD setup, the development team will implement documented manual deployment and testing steps to ensure consistent quality until automated processes are restored.

Testing will proceed in three phases aligned with our development milestones. In Phase 1, beginning in October 2024, unit tests will focus on isolated features, verifying basic functionality for individual components like registration and login to ensure each module performs as expected before integration with other parts of the system. In Phase 2, scheduled for November 2024, integration and system testing will confirm that components interact seamlessly between the front-end and back-end, with special attention to data transfer and security to ensure data flows smoothly and is stored accurately in MongoDB. This phase will also address any performance bottlenecks, particularly in data-intensive features such as the Volunteer Reporting tab. Phase 3 will conclude with acceptance testing in late November 2024, where GCISL stakeholders will assess the app’s usability, data accuracy, and overall experience, and stakeholder feedback will guide final revisions to ensure alignment with project goals.

This testing process will result in several deliverables, including a comprehensive suite of automated unit and integration tests covering core features and major data flows, a detailed documentation package for manual tests outlining test procedures, expected outcomes, and observed results, and a CI/CD pipeline configured via GitHub Actions and integrated with Vercel to automate testing and deployments. Should any CI/CD limitations emerge, documented manual deployment procedures will ensure continuity in deployment and testing without disrupting the project timeline.

3. Scope

This document outlines the approach and detailed procedures for testing the GCISL Full Stack App. Testing will ensure that all features perform as expected, are user-friendly, and meet stakeholder requirements. Unit, integration, and system tests will validate both functional and non-functional aspects.

XV. Testing Strategy

Our testing strategy for the GCISL project will include fully automated tests for core functions, making sure they run smoothly through a Continuous Integration (CI) pipeline. Additionally, we'll use Continuous Delivery (CD) for easier and automated deployment processes. The key areas for testing include user authentication, CRUD operations on tasks and user data, and role-based access control. Non-core functions may also be tested, although they might not be prioritized due to time limits. Our testing process is divided into two main parts: the developer testing process and the playtesting process.

Developer Testing Process

1. **Developer Writes Code:** The process begins with adding the feature or fixing a bug. The implementation is considered complete when the feature achieves the required functionality.
2. **Determine Test Cases:** For each core function, at least two test cases will be created: one for expected behavior and one for unexpected or invalid behavior. Non-essential functions might include only test cases for expected behavior.
3. **Run Tests:** The developer runs all tests, including newly added ones, ensuring the new code doesn't break existing functions.
4. **Fix Issues**: If any tests fail, the developer finds and fixes the issues, rerunning tests until all pass.
5. **Developer Pushes Code to Remote:** The developer pushes their code to the remote repository. The code will be run through a CI pipeline, showing the results. Only branches that pass all tests in the CI are allowed to merge.
6. **Developer Makes a Pull Request Against Main:** The developer creates a pull request against the main branch and adds at least one reviewer. The branch cannot be merged until approved by the reviewer(s).
7. **Merge Branch:** Once approved, the branch is merged into the main branch. If a Continuous Delivery (CD) pipeline is set up, it will deploy a new release if needed.

Playtesting Process

1. **Create Playtest Build:** A developer creates a playtest build, either fresh or with modified content to focus on specific test areas.
2. **Deliver Build & Allow Time for Testing:** The build is delivered, allowing several weeks for thorough testing.
3. **Deploy Feedback Forms:** Testers are asked to fill out feedback forms about their experience, including enjoyment, learning, and retention.
4. **Collect Feedback Data:** Feedback is gathered and analyzed for insights on improvements.
5. **Make Any Necessary Changes:** Based on feedback, necessary changes to content, UI, or user experience are made for the next version.

Not using CI/CD would mean relying on manual testing and deployment, which is time-consuming, prone to errors, and less efficient. For a project aiming for continuous improvement and quick updates, CI/CD is essential. By implementing this testing strategy, we can ensure strong, reliable, and efficient testing and delivery of all project components for the GCISL project.

XVI. Test Plans

1. Unit Testing

The team will use unit testing to validate individual components of the gciConnect system, isolating the smallest units of functionality from other parts of the codebase to ensure that each component works as intended. The following approach will guide unit testing:

- Testing Framework: We will use the Node js unit test framework for backend functions, focusing on volunteer reporting, data retrieval, and administrator operations. For frontend components, we will use Jest and React Testing Library.
- Core Functionalities: Each core functionality will have unit tests, including functions for adding, updating, deleting volunteer reports, and verifying user authentication.
- Test Coverage: The team will aim for high code coverage for all core functions, especially focusing on database transactions and API endpoints.
- Testing Sequence:

- Developers will write and run tests for newly developed functions before merging code.
- Tests will be automated in CI/CD pipelines to ensure no core functionality breaks with future updates.

- Expected Outcome: All core functions should pass unit tests without errors or unexpected behavior. If any test fails, the responsible developer will refactor the code, rerun the tests, and ensure success before merging

2. Integration Testing

Integration testing will focus on groups of components, ensuring they work together correctly within gciConnect's architecture. The following steps will guide integration testing:

- Testing Strategy: Integration tests will be written using Node js unit test for backend testing and Jest for the frontend. Testing will begin with pairing backend components (such as API routes with database operations) and frontend components (UI elements interacting with backend APIs).
- Key Integration Points:

- **Data Sync**: Testing the integration between the frontend forms and the backend APIs that handle volunteer reports.
- **User Authentication and Session Management**: Ensuring that the login sessions and user access levels (volunteers vs. administrators) are managed properly across the system.

- Test Coverage: The integration tests will cover end-to-end scenarios, such as:

- Adding a new volunteer report from the frontend and checking its presence in the backend.
- Ensuring user role permissions restrict or allow specific functionalities as per requirements.

- Testing Sequence:

- Integration tests will be automated to run after unit tests, and successful integration will be a criterion for code merging.
- For each feature update, integration testing will validate the combined functionality of associated components.

- Expected Outcome: Integration tests should confirm smooth interactions between components, with no issues in data transfer, user sessions, or permissions. Any failures will lead to a review and refactoring of the affected code

3. System Testing

System testing will focus on validating the complete gciConnect system as a whole, treating it as a black box to confirm all requirements are met. This includes functional, performance, and user acceptance testing

i. Functional testing:

Functional testing will involve validating the core requirements of gciConnect, focusing on user-facing functionalities:

- **Scope**: Functional requirements include volunteer report tracking, user authentication, report generation, and administrative dashboard functionalities.
- **Test Approach**: Each requirement will have one or more corresponding test cases. For instance:
    - Submit a volunteer report and confirm its presence in the administrator dashboard.
    - Generate a summary report of volunteer activities and verify data accuracy.
- **Testing Sequence**:

- Each functional requirement will be manually tested by a developer to ensure the system behaves as expected.
- QA testers will also execute these tests independently to confirm results.

Expected Outcome: The system should meet all functional requirements without errors. In case of issues, the developer will make necessary modifications and re-run tests.

ii. Performance testing:

Performance tests check whether the nonfunctional requirements and additional design goals To ensure that the GCISL application meets its nonfunctional requirements and design goals, our team will employ a combination of backend and frontend performance testing tools. We will use **Postman** for load testing the API endpoints to monitor response times and throughput under various loads. Additionally, **Google Chrome DevTools** will assist in assessing frontend performance metrics, such as page load speed and memory usage.

Since a key non-functional requirement is system reliability during high traffic, **stress testing** will be performed. This involves pushing the API beyond typical usage limits to observe its failure points and determine when performance degradation begins. The primary areas of focus will include latency, CPU, and memory usage, all of which will be monitored throughout these tests.

For any bottlenecks discovered during stress tests, the team will refine the code and database queries to enhance system efficiency and stability. These improvements will be documented and assessed against the initial design goals.

iii. User Acceptance Testing:

To ensure the GCISL application aligns with the needs of end-users and satisfies the project's requirements, our team will conduct User Acceptance Testing with selected users who represent the target audience.

A. Resources

- **Working Build**: A stable build of the GCISL application will be provided for UAT participants.
- **Feedback Form**: Users will receive a form to submit their feedback regarding usability, functionality, and overall satisfaction.
- **Developer Access**: Developers will be available to assist users and address any usability issues encountered.

B. Instructions

- **User Selection**: A small, representative group of end-users will be selected to participate in UAT.
- **Testing Process**:
  - Each participant will receive access to the GCISL application along with a brief orientation.

    - Users will interact with core features, such as the user registration, login, and dashboard functionalities, under typical usage scenarios.
- **Feedback Collection**: Users will be prompted to complete a feedback form, detailing their experience, issues encountered, and suggestions for improvement.

C. Revision Process

- **Feedback Review**: The development team will review all feedback and identify common themes or critical issues.
- **Modification and Documentation**: Based on feedback, adjustments will be made to enhance usability and functionality. Any bugs identified will be documented and resolved in the next iteration.
- **Iteration**: If necessary, further testing iterations will be conducted to refine the application until it consistently meets user expectations.

The final iteration of User Acceptance Testing will focus on validating that the GCISL application is user-friendly, meets all functional requirements, and is ready for operational deployment.

XVII. Environment Requirements

Specify both the necessary and desired properties of the test environment. The specification should contain the physical characteristics of the facilities, including the hardware, communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Identify special test tools needed.

Answer: Our testing environment is predominantly self-contained, utilizing modern web development tools compatible with both backend and frontend testing. Testing will be conducted through local environments as well as through the GitHub CI/CD pipeline for consistency and efficiency.

For testing purposes, unit and integration tests will utilize Jest and React Testing Library for JavaScript modules and React components, respectively. Additionally, Supertest (Node.js library used for testing HTTP endpoints – for API testing) will be used for testing backend API endpoints to ensure expected interactions with MongoDB.

If available, the team will incorporate GitHub Actions for continuous integration (CI) to automate tests on each commit and merge, guaranteeing that tests are consistently run and passed before changes are integrated into the main branch.

There are no specific hardware requirements.

XVIII. Alpha Prototype Description

Our alpha prototype focuses on the core functionalities of the enhanced gciConnect system. The prototype includes the following subsystems and their implemented features:

1. **User Management Subsystem**
    - **Functions and Interfaces Implemented**:
      The User Management subsystem facilitates account creation, role assignment, and authentication. Role-Based Access Control (RBAC) is fully functional, ensuring secure access by user role (e.g., administrator, volunteer). Session management ensures secure logins and user data integrity.
    - **Preliminary Tests**:
      Manual tests were conducted to validate account creation and authentication. Testing verified that role-based permissions worked correctly, with unauthorized users unable to access restricted areas.
2. **Volunteer Management Subsystem**
    - **Functions and Interfaces Implemented**:
      The Volunteer Management subsystem supports volunteer data entry, task assignment, and activity logging. A task scheduling algorithm prioritizes task assignments based on availability.
    - **Preliminary Tests**:
      Tests verified that tasks could be assigned and retrieved accurately. Logged activities displayed correctly in reports.
3. **Communication Subsystem**
    - **Functions and Interfaces Implemented**:
      The Communication subsystem handles notifications and message management, including Contact Us submissions. Integrated Google Sheets for temporary data storage.
    - **Preliminary Tests**:
      Messages were logged and routed to administrators successfully. Notifications for task updates were tested via automated emails.
4. **Reporting Subsystem**
    - **Functions and Interfaces Implemented**:
      Generates activity reports and exports them in CSV and PDF formats. Data aggregation and export algorithms were implemented.
    - **Preliminary Tests**:
      Export functionality and data accuracy in reports were verified.
5. **Database Subsystem**
    - **Functions and Interfaces Implemented**:
      MongoDB was used to store volunteer and user data. Integration with other subsystems ensures seamless data management.

- **Preliminary Tests**:
  Data retrieval from the database was consistent and accurate across all integrated features.

XIX. Alpha Prototype Demonstration

Summary of Demonstration:

During the alpha demonstration, the team showcased:

- User registration and login functionality with role-based access.
- Volunteer data entry and task assignment workflows.
- Automated notifications sent through the Communication subsystem.
- Report generation and export to CSV format.

Mentor Comments/Suggestions:

- Introduce an "Events" feature where tasks can be grouped under larger event categories, streamlining organization and management.
- Add visual indicators for task progress to improve usability and provide users with a clear understanding of their task status at a glance.

XX. Future Work

The following tasks will be prioritized to enhance the system in the next development phase:

1. **Admin Dashboard Enhancements**
   - Add a **Logs Page** for tracking system events, user activities, and error logs, enabling administrators to monitor and troubleshoot effectively.
   - Introduce a **Research Page** to facilitate the management of research-related data and findings, providing a centralized location for administrators to oversee research progress.
2. **In-App Messaging**
   - Develop and implement an in-app messaging feature to enable real-time communication between administrators, staff, volunteers, and other stakeholders. This will streamline coordination and reduce dependency on external platforms.
3. **Volunteer View Completion**
   - Finalize and implement the Volunteer View, ensuring that volunteers have an intuitive interface to view assigned tasks, track their contributions, and view update task statuses.
4. **Automated Testing Implementation**
   - Develop and integrate unit, integration, and system tests into the CI/CD pipeline to ensure reliability during future updates.
5. **Advanced Reporting Features**

    - Allow users to generate customized reports based on filters such as date range, user role, or task type.
6. **Scalability Improvements**
    - Optimize database queries and API endpoints to support higher traffic and large datasets.

These updates aim to enhance usability, ensure scalability, and provide critical functionality for both administrators and volunteers.

XXI. Glossary

gciConnect: A web-based system used by GCISL for tracking and managing volunteer reports.

Automated Testing: A process where software tools are used to automatically run tests on the system to ensure its functionality, detect bugs, and improve reliability.

Application Performance Monitoring (APM): Tools and processes used to monitor and manage the performance and availability of software applications in real-time.

***Admin Portal*:** A web-based interface used by system administrators to manage users, track performance, and troubleshoot issues within the system.

User Interface (UI): The part of a software application that users interact with, which should be designed to be intuitive and accessible.

*V**olunteer Report Tracking System**:* A system that helps track and manage the reports submitted by volunteers working in senior living environments, primarily for research purposes.

RESTful APIs: An architectural style for designing networked applications. It stands for Representational State Transfer, and it's used to allow communication between systems over the web.

Database: An organized collection of structured information or data, typically stored electronically in a computer system, that can be easily accessed, managed, and updated.

Scalability: The capability of a system to handle an increasing number of users or transactions without performance degradation.

System Reliability: The ability of a system to consistently perform its intended functions without failure over time.

Component-based architecture(CBA): an architectural style where systems are built by assembling pre-built and reusable software components

Role-Based Access Control (RBAC): A system of managing user permissions based on their role within an organization (e.g., volunteer, staff, administrator).

User Interface (UI) Layer: The part of the application responsible for managing interactions between users and the system, such as page layouts and navigation.

Backend Layer: Manages the system's core functionality, including user and volunteer management, task assignments, and notifications.

MongoDB Atlas: A cloud-based NoSQL database used for storing user, task, and volunteer information in a flexible document format.

Task Scheduling Algorithm: A method used by the Volunteer Management Handler to assign tasks to volunteers based on availability and need.

Session Management Algorithm: Ensures user sessions are validated and managed securely throughout the platform.

UAT (User Acceptance Testing) Participants: They are real end-users or people who represent the target audience of the application. They help confirm that the system meets the intended requirements and is easy to use.

Postman: it is a popular API testing tool that lets developers send HTTP requests to their backend and check responses, making it easier to validate and debug APIs.

CI/CD (Continuous Integration/Continuous Deployment): A set of practices that automate code integration and deployment processes. CI/CD pipelines run tests automatically upon each code push, ensuring continuous validation of code changes and quick deployment of updates.

Jest: A JavaScript testing framework primarily used for testing React applications. It enables developers to run unit and integration tests on components and functions to verify correctness.

React Testing Library: A tool that enables testing of React components by interacting with the application the way a user would, focusing on component behavior rather than implementation details.

Supertest: A Node.js library for testing HTTP endpoints. It allows for testing API interactions, verifying that backend services function correctly and respond with the expected results.

Unit Testing: A software testing method where individual components or functions of an application are tested in isolation to ensure they work as expected.

Integration Testing: Testing that verifies multiple components of the system work together. In this context, it ensures that front-end elements interact correctly with backend APIs and databases.

System Testing: Testing the entire application as a whole to validate that it meets functional and non-functional requirements, ensuring overall system integrity.

Functional Testing: A type of testing that validates the features and operations of an application, ensuring that they align with specified requirements and that the system behaves as expected in various scenarios.

Performance Testing: Tests that evaluate the application's speed, scalability, and stability under load, focusing on metrics such as response time, memory usage, and throughput.

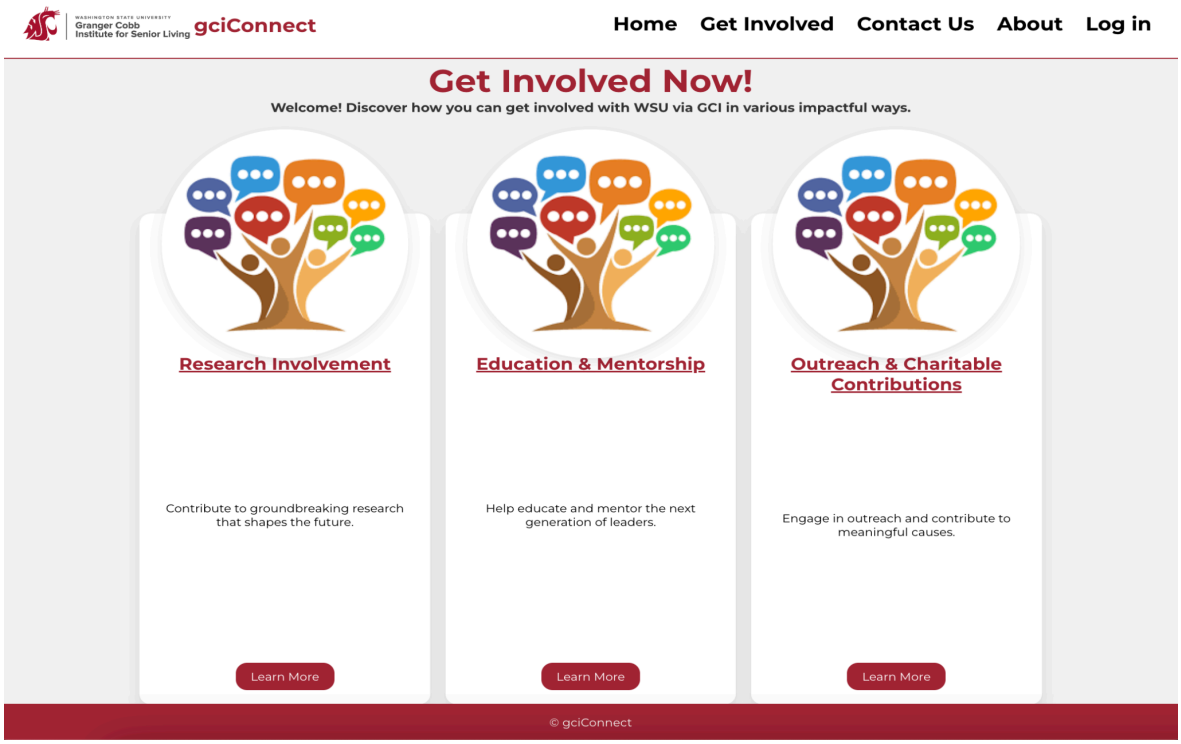
Stress Testing: A type of performance test where the system is subjected to extreme workloads to identify its breaking point and measure how it handles high traffic or data volumes.

MongoDB: A NoSQL database used for storing and retrieving data in the GCISL application. Known for its scalability and flexibility, MongoDB stores data in JSON-like documents.

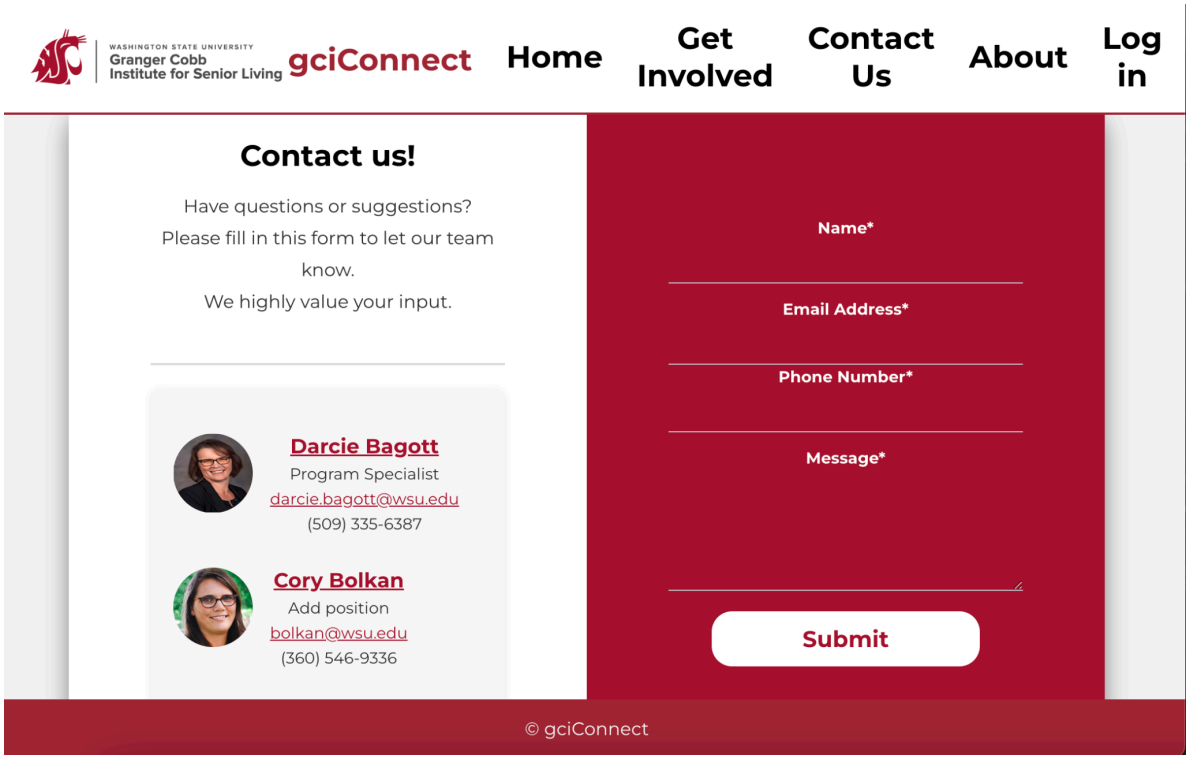
XXII. Appendix

[Image 1]:

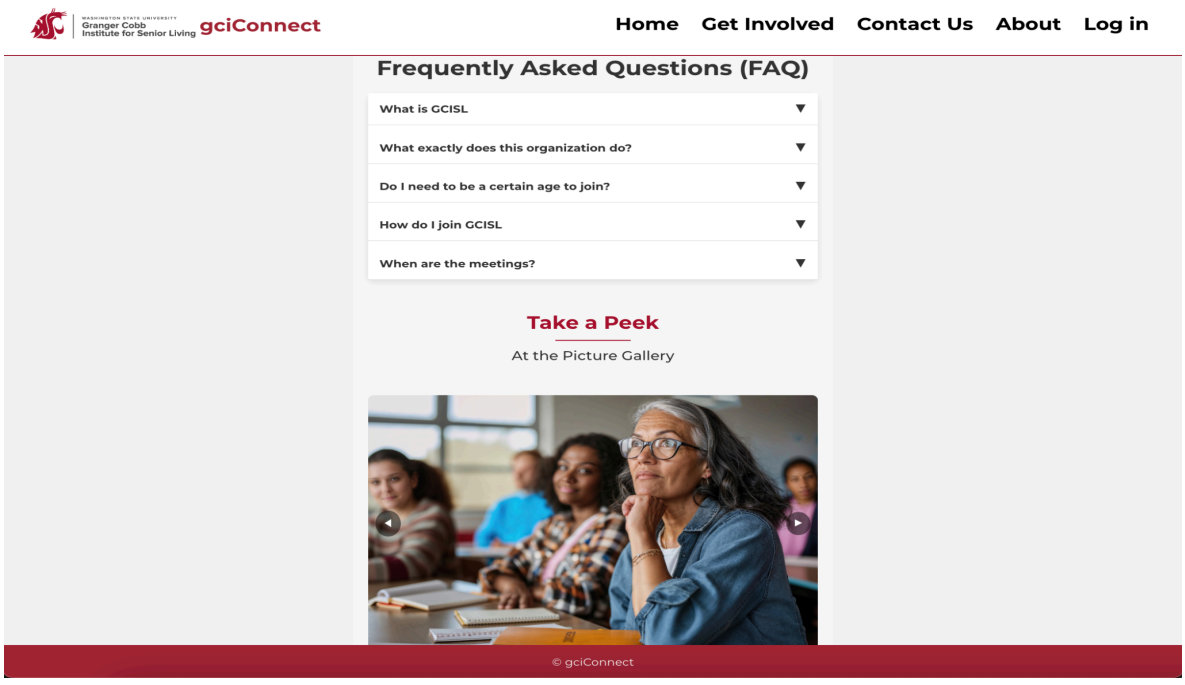
[Image 2]:



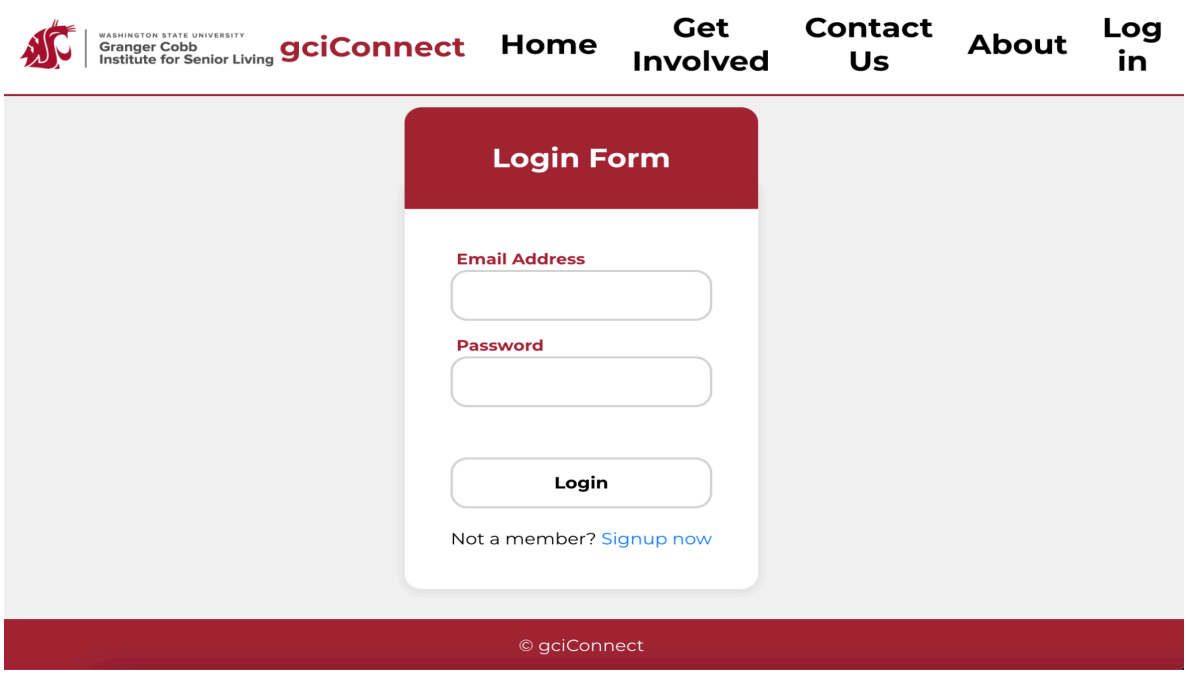
[Image 3]:



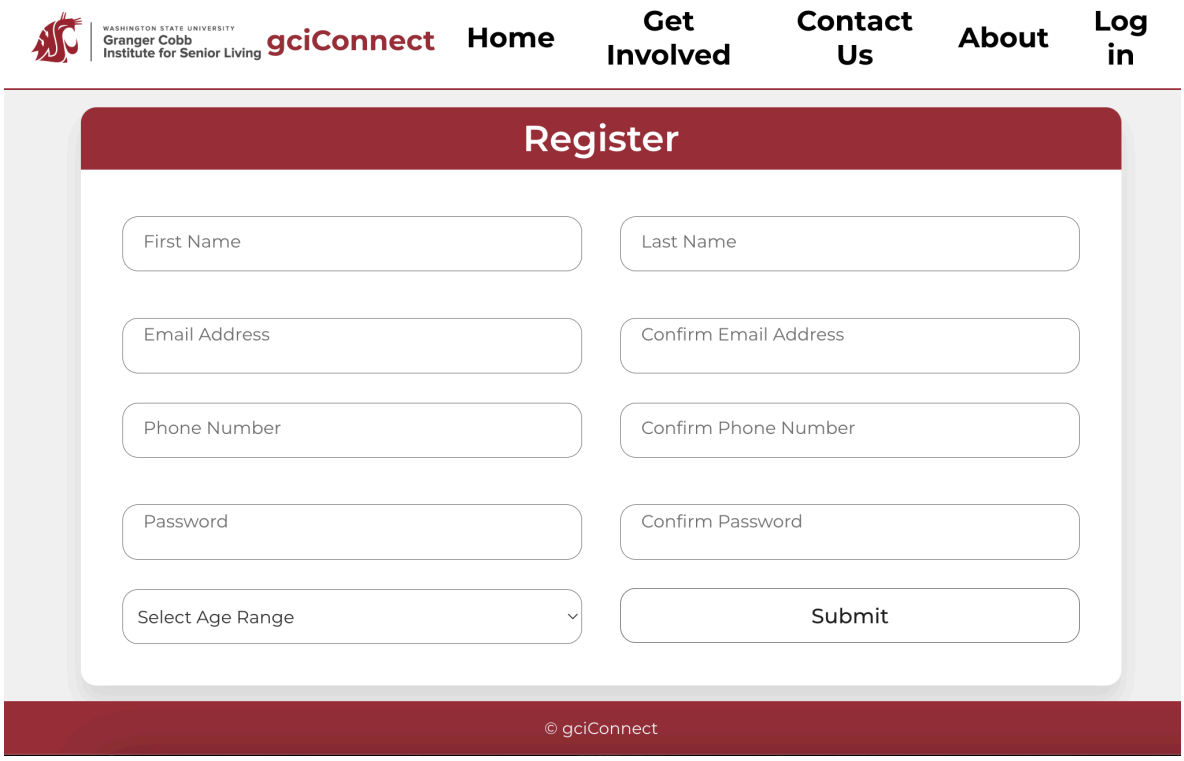
[Image 4]:



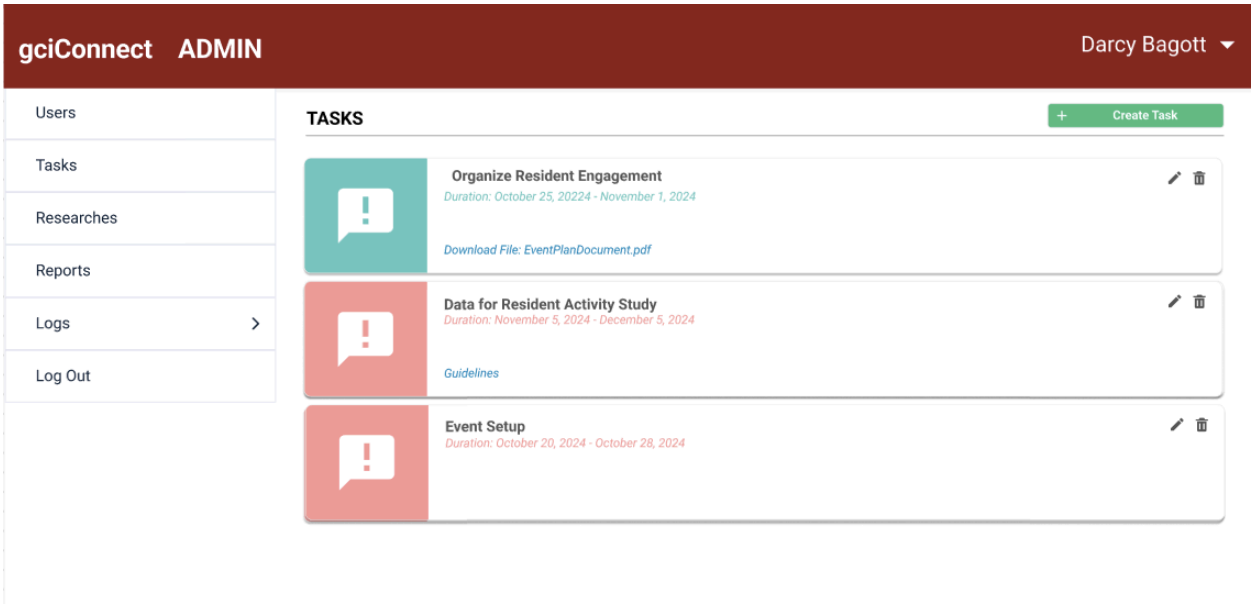
[Image 5]:



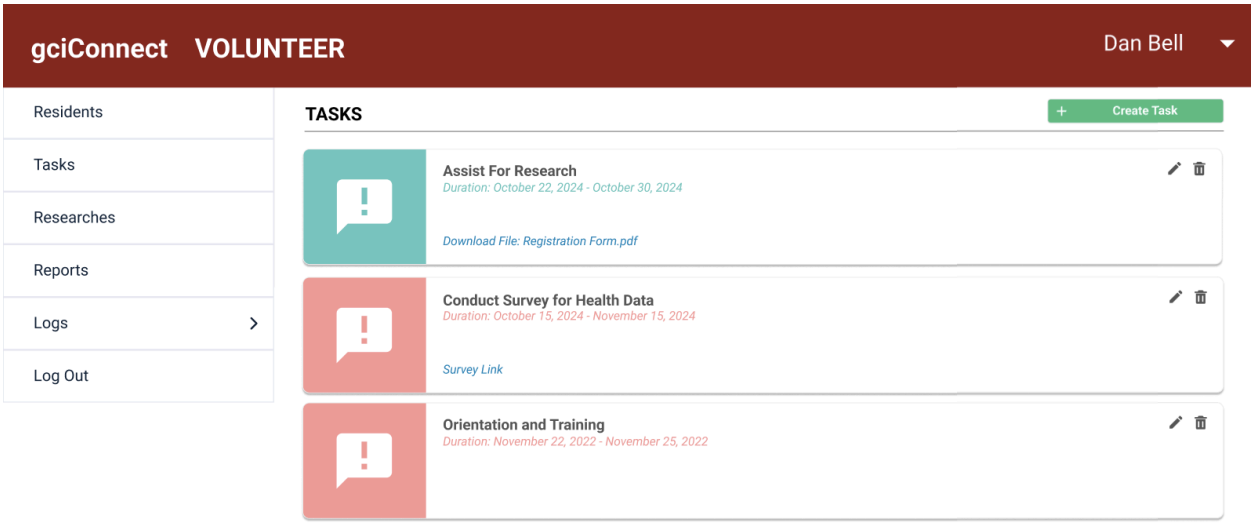
[Image 6]:



[Image 7]:


gciConnect ADMIN
Darcy Bagott
Users
Tasks
Researches
Reports
Logs
Log Out
TASKS
Create Task
Organize Resident Engagement
Duration: October 25, 20224 - November 1, 2024
Download File: EventPlanDocument.pdf
Data for Resident Activity Study
Duration: November 5, 2024 - December 5, 2024
Guidelines
Event Setup
Duration: October 20, 2024 - October 28, 2024

[Image 8]:


gciConnect VOLUNTEER
Dan Bell
Residents
Tasks
Researches
Reports
Logs
Log Out
TASKS
Create Task
Assist For Research
Duration: October 22, 2024 - October 30, 2024
Download File: Registration Form.pdf
Conduct Survey for Health Data
Duration: October 15, 2024 - November 15, 2024
Survey Link
Orientation and Training
Duration: November 22, 2022 - November 25, 2022

XXIII. References

WSU Granger Cobb Institute for Senior Living, "Volunteer Report Tracking System Enhancement," internal document, ACME8-GCISL FullStackApp, 2023

"MongoDB Documentation," MongoDB Inc., 2023. [Online]. Available: https://docs.mongodb.com

"Google Sheets API Documentation," Google LLC, 2023. [Online]. Available: https://developers.google.com/sheets/api

"React Documentation," Meta Platforms, Inc., 2024. [Online]. Available: https://reactjs.org/docs/getting-started.html

Gillis, Alexander S. “What Is User Acceptance Testing (UAT)?: Definition from TechTarget.” *Software Quality*, TechTarget, 14 Mar. 2022. [Online]. Available: www.techtarget.com/searchsoftwarequality/definition/user-acceptance-testing-UAT. [Accessed: 02-Nov-2024]

Postman. [Online]. Available: https://www.postman.com/postman/test-examples-in-postman/overview. [Accessed: 02-Nov-2024]