VolunteerOne Design Portfolio 3

Requirements Document: VolunteerOne Specifications Document
Instructions: Final Report
Design Portfolio 3: Group 4: Design Portfolio 3 Final

GitHub Repository Link

Document:	Design Portfolio
Version:	3.0
Date:	Apr 30, 2023
Author:	Group 4
Status:	Delivered

Contributors:

Name	**Email**	**GitHub**
Carmen Lee	leec61@unlv.nevada.edu	carmen-lee
Michael Lazeroff	lazerm1@unlv.nevada.edu	lazeroffmichael
Edward Sung	sunge1@unlv.nevada.edu	edwardsung4217
Mathis Tessier	tessim1@unlv.nevada.edu	Mtessier809
David Zeleniek	Zeleniak@unlv.nevada.edu	dzeleniak
Matthew Yapjoco	yapjoco@unlv.nevada.edu	mattacy
Cicelia Siu	siuc1@unlv.nevada.edu	yaaacii
Cera Samson	samsoc2@unlv.nevada.edu	cerasamson
Dillon Harder	harded1@unlv.nevada.edu	Rahgid
Genaro Lopez Baez	lopezbae@unlv.nevada.edu	GoSec-Hub

1. Introduction:

A. Abstract

VolunteerOne is a peer-based mobile app that aims to connect volunteers to their communities through posted opportunities and events. Its purpose is to increase community engagement and emphasize input from volunteers on their skills, interests, and experience to match suitable opportunities and create more impactful contributions. Features include clear communication channels between organizations and volunteers, hour tracking for groups looking to volunteer together, and various social features, including sharing experiences with friends, family, and organizations.

B. Executive Summary

VolunteerOne is a modern volunteering approach that aims to simplify and streamline the volunteering experience for both organizations and volunteers. Our one-stop solution includes a management system that facilitates communication, searching for volunteer opportunities, and consistency. Unlike other products with a similar goal, such as VolunteerMatch [1], we have taken into account the dissatisfaction of users with confusing layouts and layouts with overwhelming details. Our mobile app is designed to be simple, user-friendly, and interactive, addressing the main issue at heart: how to increase community engagement through accessibility.

We have addressed major problems that occur in current volunteer management services, problems such as handling communication and coordination, using volunteer skills effectively, and managing volunteer burnout, as noted by volunteering logistics company Volgistics [2]. In order to address these problems, our app features multiple communication channels, including an announcement board for organizations to keep information up-to-date for their new and existing volunteer opportunities. It also displays key details for those opportunities and events in a succinct and visually appealing way, allowing volunteers to make well-informed decisions about how they can make an impact in their community. Additionally, we have included a social aspect to our app, allowing users to interact with friends and family, allowing them to share their experiences, photos, and the other activities they participate in.

VolunteerOne is meant to be the first place people look for anything related to volunteering. It is a full-service app designed to benefit its community. Volunteers and organizations looking for ways to interact with their community will benefit greatly from VolunteerOne.

2. Motivation:

A. Journal Articles

1. *Understanding the volunteer market: The what, where, who and why of Volunteering.*

   In this survey paper published in 2002 by Bussell and Forbes, they claim that recruiting and retaining volunteers requires understanding the target market by understanding what volunteering is, who volunteers, where it occurs, and why people volunteer. The authors present the relevant literature on these four problems, attempting to define each area as well as probe the current problems. To understand the success of an organization's ability to recruit and retain volunteers, the article analyzes prior research and current metrics with regard to the donation of time. This concludes to the understanding of volunteer motivations which prove to not simply be due to altruism. [3]

   With the design of VolunteerOne, we tackle the volunteer problem by considering specifically the who and why of volunteering. Brussel and Forbes explain how other studies found younger people are less likely to volunteer than those over 50 [3]. VolunteerOne aims to help organizations rectify this age variability by implementing a system that lowers the friction needed to start volunteering and incentivizes it through social connections and statistics. Furthermore, Brussel and Forbes identified multiple "whys" for volunteers - from seeking to help others, to the ability to gain skills, value alignment, and egoistic motives [3]. VolunteerOne recognizes the different motivating values and seeks to appeal to those with features such as interest matching and organization pages so prospective volunteers can choose organizations and events serving missions vital to them. This article helped shed light on some of the fundamental problems with understanding and acquiring volunteers, allowing VolunteerOne to deliver a technical solution.

2. *Volunteer Motivation and Satisfaction*

   Volunteer Motivation and Satisfaction, an article by Hyejin Bang and Stephen D. Ross examines the motives behind volunteers. Historically, theories on the motivation of volunteering have been attributed to altruism, social contact, personal interests, and emotional needs [4]. Studies have shown that many of the reasons for volunteering are to satisfy individual needs or interests and the primary reason for volunteering was not altruistic.The purpose of the study is to conduct research on the scale development of volunteer motivation [4]. One result of the study finds that it is imperative for volunteering event managers to understand the motivations of volunteers and how each volunteer can be assisted in achieving a sense of personal satisfaction [4].

   This article notes that the way younger individuals view volunteering has shifted from how older generations viewed it. As stated, one of the primary reasons for volunteering for new generations is to satisfy individual needs and interests. With VolunteerOne, we are catering to the younger individuals by creating a social network platform centered around volunteering, and we enable users to select events that align with their personal

interests. By modernizing how volunteering recruitment is done, we approach the process with a more fun and social aspect that will continue to attract newer generations to volunteering.

3. *Meanings of Organizational Volunteering: Diverse Volunteer Pathways*

Being a volunteer means many things and takes many shapes. As expressed in the article *Meanings of Organizational Volunteering: Diverse Volunteer Pathways* by Kirstie McAllum, there exists what one may call volunteering pathways. The article explores the motivations and reasons for volunteering. As well as, the benefit that comes from volunteering both for the individual and the organization [5]. More importantly, the study is unable to define volunteering as "participants struggled to pinpoint which type of activities could be considered volunteering" [5]. However, it does shine a light on some potential different types of pathways named "freedom–reciprocity" and "giving–obligation" [5]. The article goes more in-depth on what each of these pathways means and how individuals benefit from volunteering, how they interact with organizations, and how they behave after volunteering is no longer an option.

This article showcases that participants may volunteer to achieve a prefer-self under "freedom-reciprocity" while those under the "giving-obligation" pathway aim to achieve a sense of ideal-self [5]. Regardless of what pathway an individual is seeking to take, VolunteerOne is the perfect idea to help everyone achieve their individual goals. With a centralized hub of opportunities, users looking for casual opportunities can find them. The user looking to volunteer more often and on a more personal level can also find opportunities. Or for those who have stopped volunteering and are seeking to come back [5] for one more shift. Regardless of the needs and motivations behind the user, VolunteerOne is the tool that will help them find what they need.

3. Requirements Document:

A. High Level Description of Functional Requirements

The functional requirements tables for the frontend and backend tasks highlight the necessary tasks to make our mobile application functional. Each task acts as a general objective and serves as a guideline for how the task should be completed and what the expected behavior of the completed task and component should be.

The tables and descriptions have been updated after thorough planning and analysis of the necessary functionality we plan for our app to have, as well as timeline. They have been revised to reflect the naming conventions used in the app as well as more concrete descriptions to describe the actual events of the defined functions. Due dates and task status have also been updated to accurately reflect task deadlines.

B. Functional requirements table and description

Front End				
ID	**Title**	**Requirement Description**	**Date**	**Status**
FS 0.0	Onboarding screen	Redirects user to Login screen	3/9/23	**X**
FS 0.1	Login screen	Displays username/password text fields with clickable routes to the Registration and Forgot Password screen.	3/9/23	**X**
FS 0.2	Account Creation Screen	Create screens to allow users to sign up for a VolunteerOne account as a volunteer or an organization	3/9/23	**X**
FS 0.3	Registration screen	Prompts users to enter their full name, email, and password. Submitting the form will result in account creation.	3/28/23	**X**
FS 0.4	Forgot Password screen	The Forgot Password screen will prompt the user to enter their email. A link is sent to the email to reset the account password.	3/28/23	**X**
FS 0.5	Create Password screen	Users will be prompted to create a new password. The user will input a new password and confirm it.	4/4/23	**X**
FS 1.0	Announcements Tab	Toggles between read-only announcements from organizations that the user follows and ‘all’ organizations.	3/21/23	**X**
FS 1.1	Post Announcements Modal	Modal that allows the user to make a text and/or photo post that will be shared to follower’s feeds. These announcements include changes to times, dates, etc.	4/4/23	**X**
FS 2.0	Explore tab	Interactive screen with swipeable cards that displays nearby volunteering opportunities the user may register to attend.	4/13/23	**X**
FS 2.1	Events Screen	Displays event information such as description, date, location, requirements, and organization information.	4/4/23	**X**
FS 3.0	Feed Tab	Displays pictures and text of what the user and their friends have posted.	3/28/23	**X**

FS 3.1	New Post modal	Modal that allows the user to make a text and/or photo post that will be shared to friend's feeds.	4/4/23	**X**
FS 4.0	Profile tab	Display user's information, recent activities, friends, and following organizations. Access to account settings screen.	3/9/23	**X**
FS 4.1	View Friends screen	Displays a list of the user's friends.	3/21/23	**X**
FS 4.2	View Following screen	Displays a list of organizations the user follows.	4/4/23	**X**
FS 4.3	Account Settings screen	Allows the user to modify their profile description, display name, and profile picture. Also provides the option to change password/email.	4/6/23	**X**
FS 4.4	Notifications screen	Displays notifications from friends and followed organizations about new posts, reactions, and requests.	3/9/23	**X**
FS 5.0	Navigation	Route tables that connects all screens	3/28/23	**X**
FS 5.1	Explore Tab	Allow the user to search for other users or organizations	3/18/23	**X**

Figure 1.1: This shows the functional requirements table for the frontend team with an ID, requirement title, description, due date, and status.

Back End				
ID	**Title**	**Requirement Description**	**Date**	**Done**
UC1	Account Creation	UC1 will include all necessary functionality for creating accounts, organizations, and configuring those entities.	03/09/23	**X**
UC1.1	Logging into Existing Account	Users will be able to log into an already existing account. JWT Authentication will be handled.	04/06/23	**X**
UC1.1.1	Forgotten Password	Users will be brought to the reset password page from the login page.	03/02/23	**X**
UC1.2	User Account Creation	Users will go to the signup page and: 1) Fill out a form to create an account 2) Confirm the account with their email 3) Redirect to login page	03/02/23	**X**
UC1.3	Profile Creation	Users will be able to create their individual profiles.	03/07/23	**X**
UC1.3.1	User Configuration - Basic Info/Interests	Users will be able to configure basic info about themselves as well as their interests in what they like to volunteer with	03/07/23	**X**
UC1.3.2	User Configuration - Change Password	Users will be allowed to change their password, given they are logged in and know their password.	03/07/23	**X**

UC1.3.3	User Configuration - Change Email	Users will be able to change the email on their account given they can verify correctly their username/password.	03/07/23	**X**
UC1.3.4	User Configuration - Change Profile Picture	User will be able to change to a different profile picture, or use a default profile picture.	03/07/23	**X**
UC2	Organizations	UC2 will handle all logic for organizations	03/23/23	**X**
UC2.1	Organization Creation	Users will be brought to the Organization Creation page where they will be greeted with how VolunteerOne can assist their organization. Users will also be prompted to: 1) Create a new organization. 2) Manage existing organizations.	03/21/23	**X**
UC2.1.1	Org Permissions - Roles	Managers to Organization accounts will be able to create roles and manage their permissions.	03/21/23	**X**
UC2.1.2	Adding Members to Orgs	Volunteers will be able to officially join various organizations as members to view announcements and new events directly on their feed, as well as be featured on the Organization's profile page. Organizations will also be able to manage their members.	03/21/23	**X**
UC2.1.3	Org Configuration - Info / Interests	Organizations will be able to edit their information and interests. Interests are defined by categories that pertain to the values of the organization and what their events may entail.	03/23/23	**X**
UC3	Volunteer Events	UC3 will cover all volunteer event related logic.	04/04/23	**X**
UC3.1	Submitting and configuring a new event	Organizations will be able to submit a new event to User dashboards.	03/30/23	**X**
UC3.2	Volunteering for an event	Users will be able to sign up for an event that they are interested in. If the organization is interested in allowing the user to volunteer based on the user's profile (skills, interests, previous work), then they will accept their request to sign up.	03/30/23	**X**
UC4	Social	UC4 will cover the social related logic.	04/13/23	**X**
UC4.1	Org Communication Channel	The "Org Communication Channel" will be a page on the volunteer's view where they can view the latest announcements from organizations they follow. It will also be accessible from the organization's view, where they can create and edit their own announcements.	04/06/23	**X**

UC4.2	Events Channel	To provide a detailed page about volunteer events. It will include details such as: description, date/time, location, associated organization, requirements, skills, suitable types of volunteers, cause areas.	04/06/23	**X**
UC4.3	Recent User Profile Activity	For others to see what that user has recently posted, participated in, and achieved.	04/11/23	**X**
UC4.4	Recent Org Profile Activity	For users to view a specific organization's past events, announcements, and other recent news.	04/11/23	**X**
UC4.5	Friends	UC 4.5 will cover friend logic	04/13/23	**X**
UC4.5.1	Adding Friends	Users can add new friends.	04/13/23	**X**
UC4.5.2	Removing Friends	Users will be able to remove friends	04/13/23	**X**
UC4.6	Following Orgs	UC4.6 will cover following orgs	04/13/23	**X**
UC4.6.1	Following an Org	Users can follow an organization	04/13/23	**X**
UC4.6.2	Unfollowing an Org	Users can unfollow an organization	04/13/23	**X**

Figure 1.2: This shows functional requirements table for the backend team with an ID, requirement title, description, due date, and status.

C. Non-functional requirements table

ID	**Title**	**Requirement Description**
NF1	Supported operating systems	Cross-platform (Android, iOS)
NF2	Programming Languages	JavaScript, Go
NF3	Project Dependencies	React Native, Expo, Gin, GORM, Mockery
NF4	Containerization	Docker
NF5	Database	SQL Server 2019 Standard
NF6	Hosting	Digital Ocean
NF7	Licensing	GNU GENERAL PUBLIC LICENSE

Figure 1.3: This shows non-functional requirements table with an ID, title, description.

D. Non-functional requirements description

- NF1 - Supported operating systems
  - We will support Android and iOS due to how they are the two most popular mobile operating systems and used by consumers worldwide
- NF2 - Programming Languages
  - We will use Go since it is a popular choice for backend development due to its efficient and fast performance which makes it ideal for handling complex tasks and large-scale projects. Go also features a garbage collector which helps manage system memory and improve performance
  - We will also be using Javascript because it complements the React Native library.
- NF3 - Project Dependencies
  - Our project dependencies will include Gin, which is an industry-standard web framework that is known for its speed, efficiency, and ease of use. We will also use SQL drivers, which will allow us to interact with the SQL database we have chosen. The ORM we will use for the SQL interactions is GORM.
  - We will also use React Native and Expo. With Expo, we can build and deploy React Native apps for both iOS and Android without having to know any native mobile coding. We have also included the MUI library since it has many user-friendly components available to use that will complement our app and design.
  - Mockery is used to auto generate mock interfaces so we can unit test our backend code.
- NF4 - Docker
  - Docker is used for containerizing our backend code into app, and database containers. Allows for easy configuration and deployment to the server.
- NF5 - Database
  - We will be using SQL Server 2019 Standard. SQL Server is a stable and secure database management system that is widely used in enterprise environments
- NF6 - Hosting
  - We will be using Digital Ocean, a cloud computing platform that offers a simple and intuitive user interface, as well as flexible pricing options. Since our team is small, we believe that Digital Ocean will provide us with the resources and support we need, without the added complexity and cost of larger platforms.
- NF7 - Licensing
  - GNU GENERAL PUBLIC LICENSE: Guaranteed freedom to share and change all versions of a program and to make sure it remains free software for all its users

4. Design Document:

A. Backend Architecture

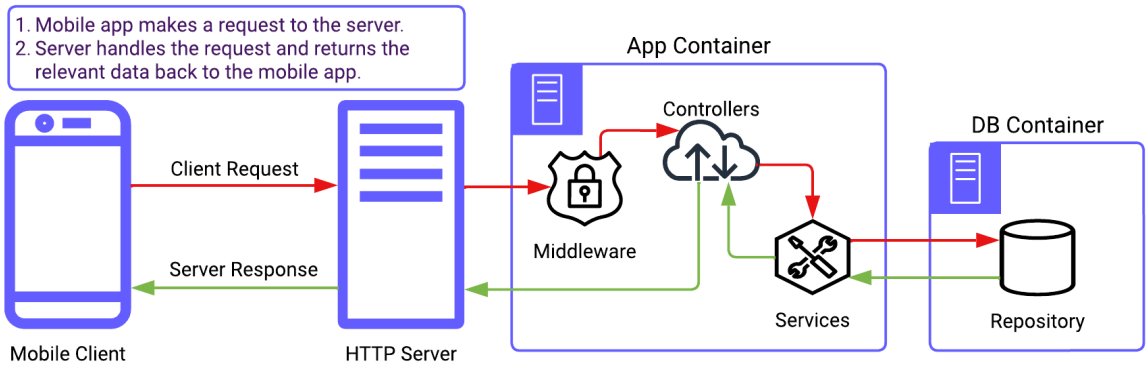


Figure 2.1: This shows the backend architecture.

From a high level perspective, the backend architecture, shown in Figure 2.1, is broken down into two separate docker containers: the app and database containers. This design choice was made for easy and consistent deployment for not only local testing, but actual production hosting on the server. With Docker, we can have one script which provisions all necessary containers and dependencies, in order for our app to function properly no matter where it's being hosted.

B. Controller, Service, Repository Methodology

The backend architecture follows the *Controller, Service, Repository* design model. This design model was chosen because it allows abstraction between different layers of the application. The benefit of this abstraction will be detailed heavily in this section.

The *Controller* layer contains the public facing handles for routes of the API. HTTP requests from the client are made to these routes, and the controller layer contains the method interfaces necessary to consume these requests. For example, the /signup route for the API is handled by a Signup() method implemented as a LoginController. Specific implementation details will be in section C where architecture for individual use cases are described.

The *Service* layer contains business logic that the *Controllers* use. For example, when a user signs up, their password must be hashed before it’s placed in the database. Instead of implementing this functionality directly in the controller, this logic is implemented as a service method, which is called by the controller instead.

The *Repository* layer contains all logic which interacts directly with the database. Only at this layer will actual calls to the database be allowed.

The flow between the different layers is illustrated by this example: suppose a controller needs to perform some complex logic, such as hashing a password, and also needs to insert a new record into the database. Without the Controller, Service, Repository design, the controller method might do something like this:

```
ExampleController(c *gin.Context) {
    // Digest request information

    // Implement hashing logic here

    // Add to DB directly

    // Respond to request with status code 200
}
```

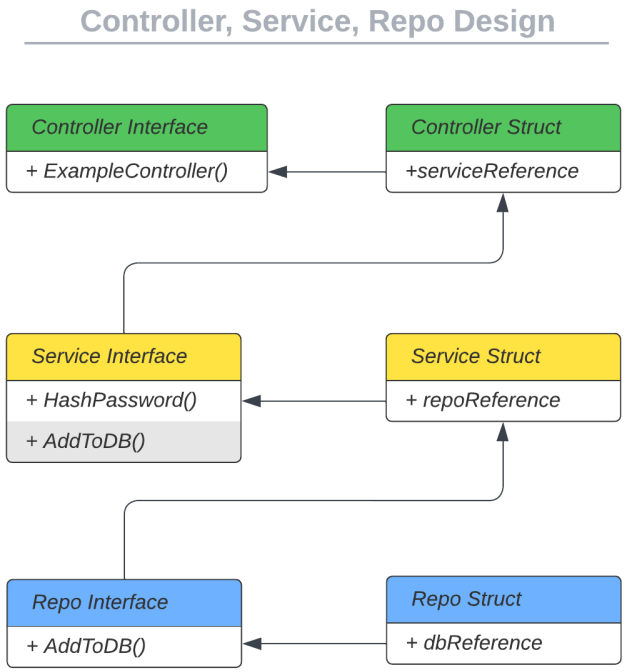


Figure 2.2: This shows the controller, service and repository design.

There are two main issues with implementing everything in the controller - not only is all of the logic contained in one massive function, but it becomes incredibly difficult to test. This design can be good for quick prototyping, but for maintaining and testing, it becomes very difficult to manage.

The Controller, Service, Repository design, shown in Figure 2.2, is implemented in Go using interfaces and structs at each layer (since Go does not have traditional object oriented design with classes), with an example shown here. Note how each layer has the methods needed in the interface, and has a reference to the lower level in a struct.

This abstracted design allows for not only a logical separation of duties - but also allows proper testing. To mock in Go, it is required to have interfaces, as we can't mock methods directly. These interfaces can be redefined to have mocked methods, and is the approach used with testing. For example, suppose we are testing ExampleController() which uses the HashPassword() and AddToDB() methods from the services layer. We can use a mocked implementation of the service interface, supposedly called ServiceMockInterface, which will re-implement HashPassword() and AddToDB() to return whatever we want from them. This mocked implementation allows us to mock the lower level methods and not have to actually call them - ultimately breaking the chain between the layers. We can then test individual layers, and as long as the individual layers are functioning correctly, we can be sure that the whole chain will work when the actual interfaces are used. Figure 2.3 shows how the chaining is broken when we use a mocked service interface (pink) for testing a controller method.

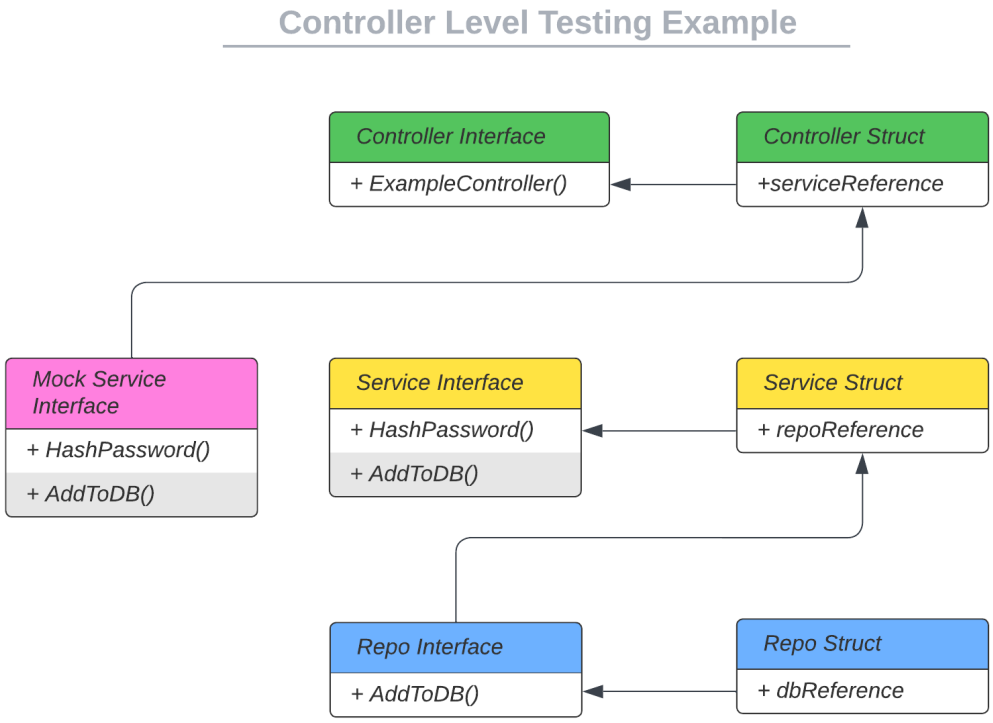


Figure 2.3: This shows an example of the controller level testing.

C. Use Case Diagrams

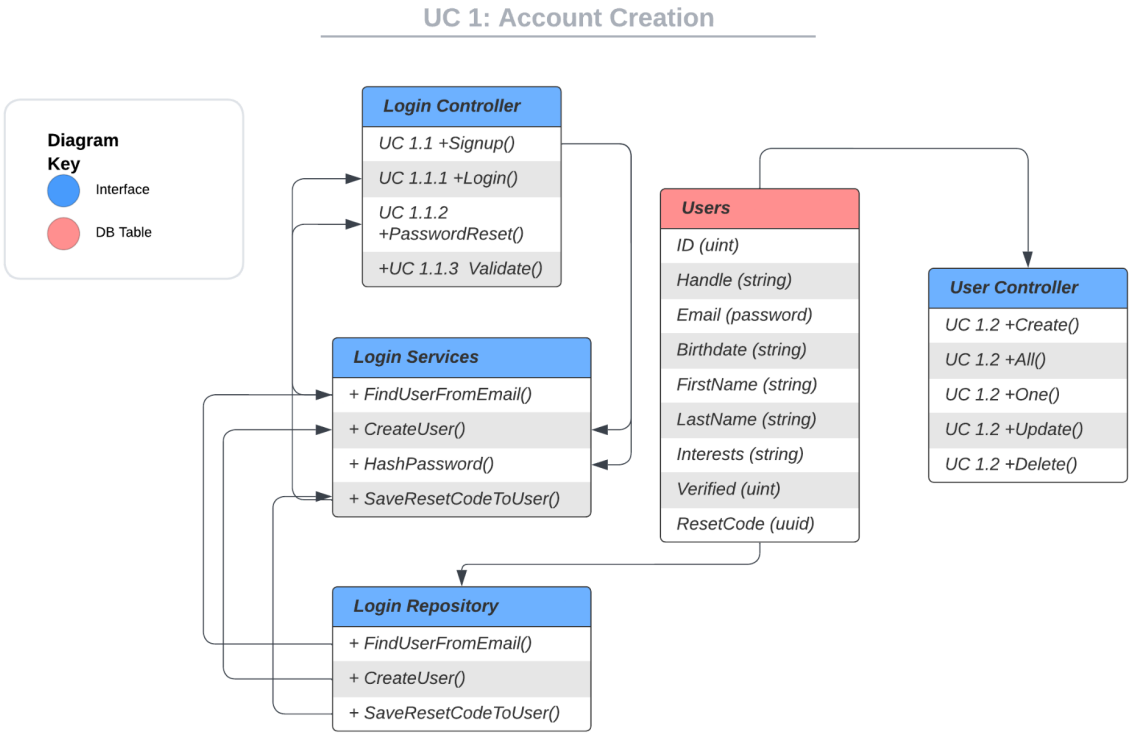


Figure 3.1: This shows UML diagram of Account Creation.

Figure 3.1 allows the application to create a user through using 4 key interfaces and then the schema table named “Users”. The application can create or login the user by utilizing the login controller

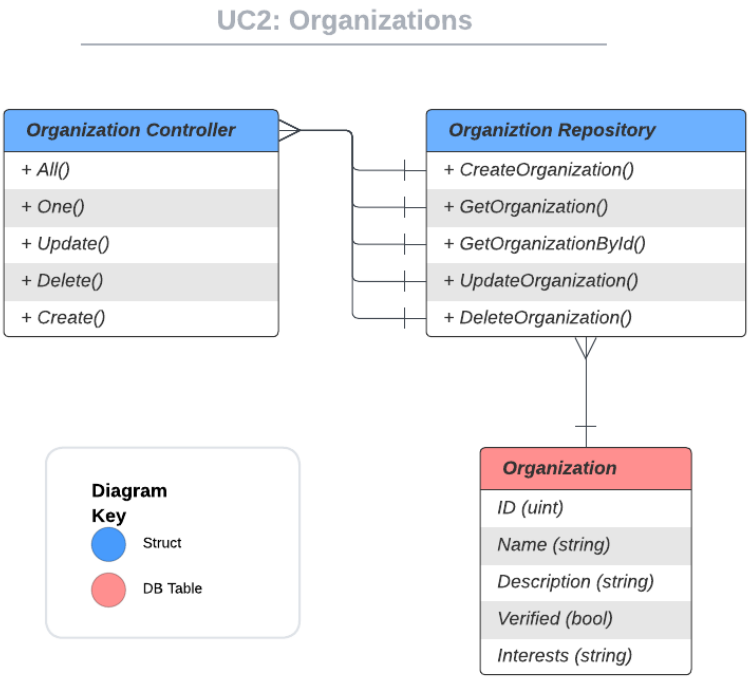


Figure 3.2: This shows UML diagram of Organizations.

Figure 3.2 allows the application to call CRUD operations on the Organization controller interface. The repository can then pull or store information about the organization from the Organization database schema

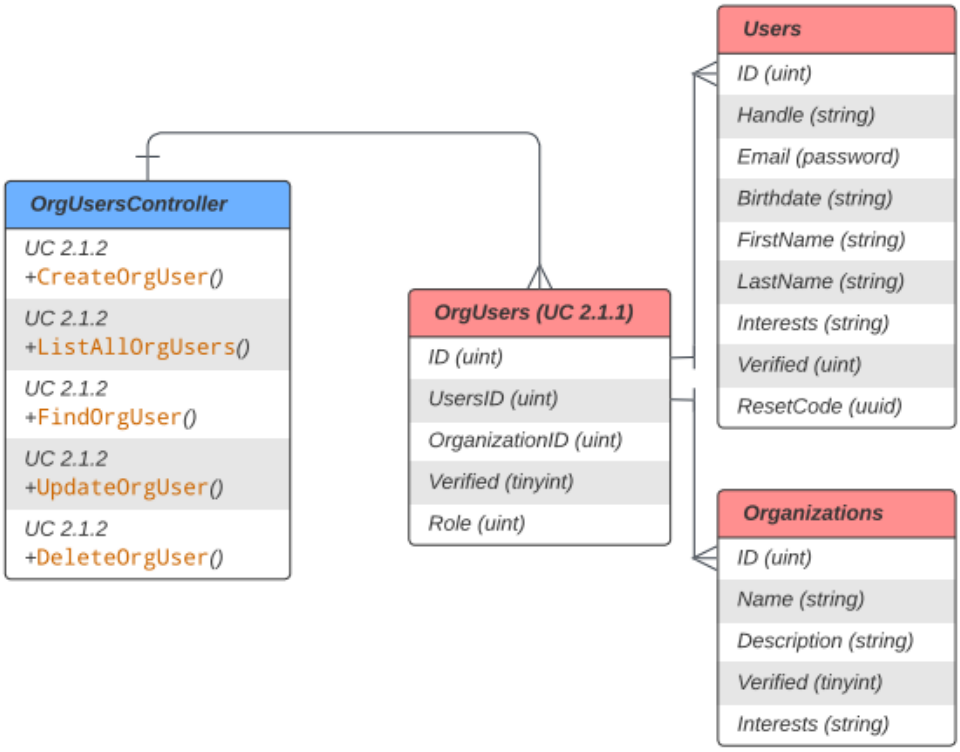


Figure 3.3: This shows UML diagram of Organization User Roles.

Figure 3.3 allows the application to do CRUD operations on the OrgUser database. This is done through the interface named “OrgUsersController” which utilizes three database schemas inorder to process these requests

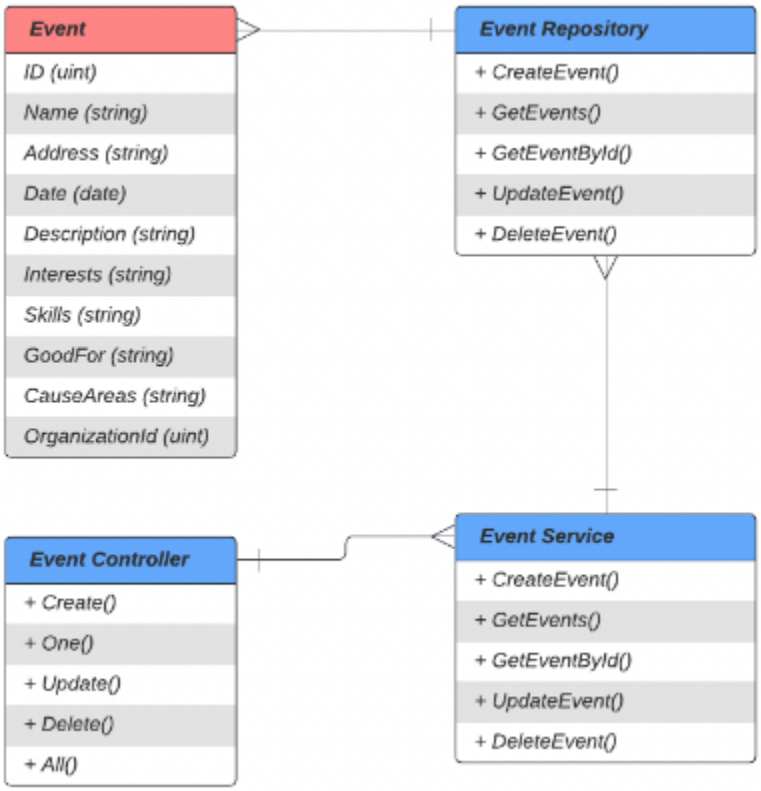


Figure 3.4: This shows UML diagram of events.

Figure 3.4 allows the application to do CRUD operations on the “Event” database schema. This is done through the Event Controller -> Event Services -> Event Repository which then utilizes the Event database schema to store or get information

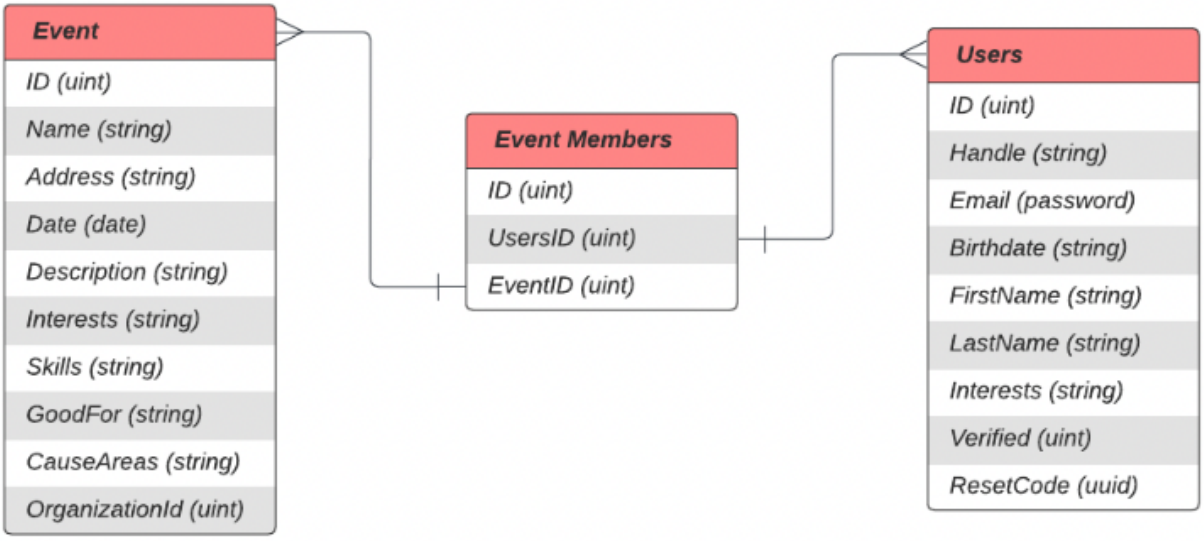


Figure 3.5: This shows UML diagram of event members.

Figure 3.5 shows the database schema for "Event Members" which has two foreign keys which are referencing the "Event" and "Users" database schema

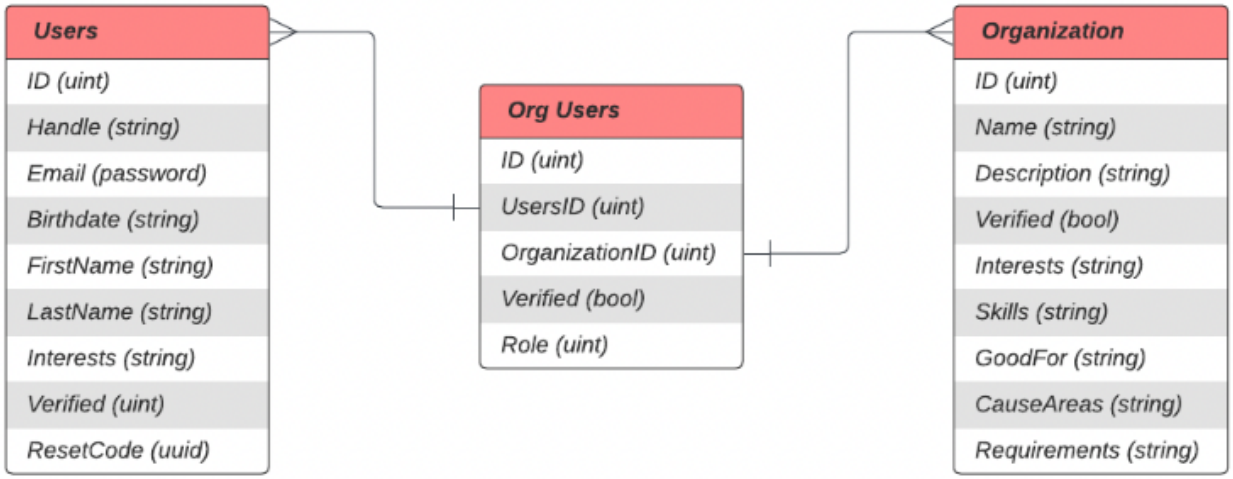


Figure 3.6: This shows UML diagram of organization members

Figure 3.6 shows the database schema for "Org Users" which has two foreign keys which are referencing the "Organization" and "Users" database schema

5. Prototype:

A. Demo Video and Front-End Screenshots

https://drive.google.com/file/d/1XIxWBsbeUadtWj45t0HURjW3-yqiUmWK/view?usp=share_link

Account Creation Screens

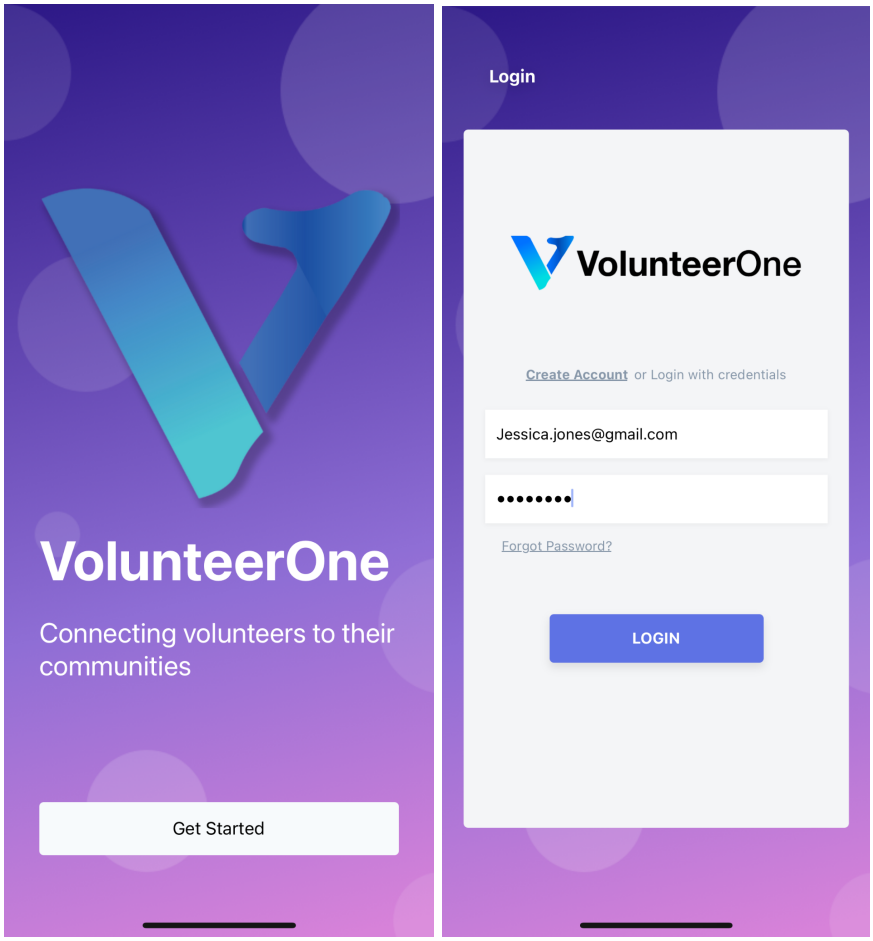


Figure 4.1 and 4.2: This shows Account Creation Screens.

Explore Screens

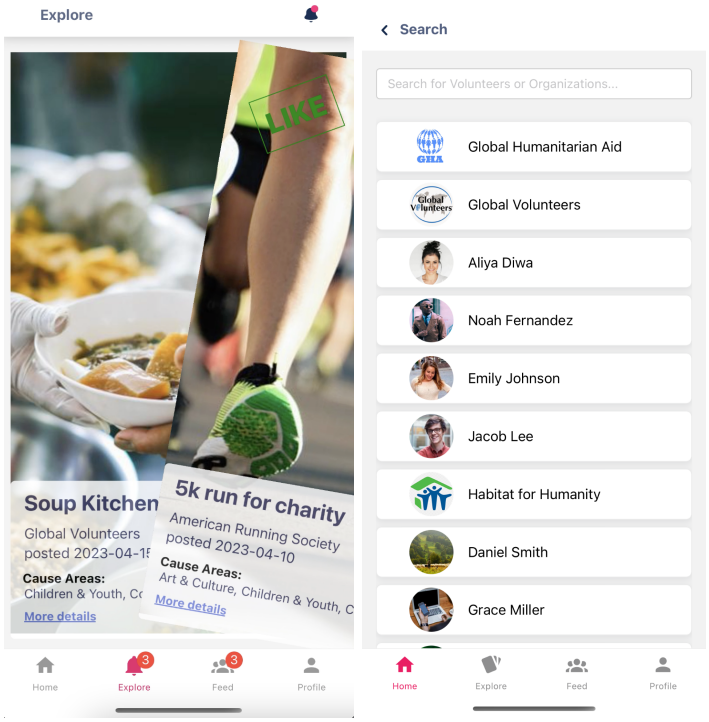


Figure 4.2 and 4.3: This shows Explore Screens.

Profile Creation Screens

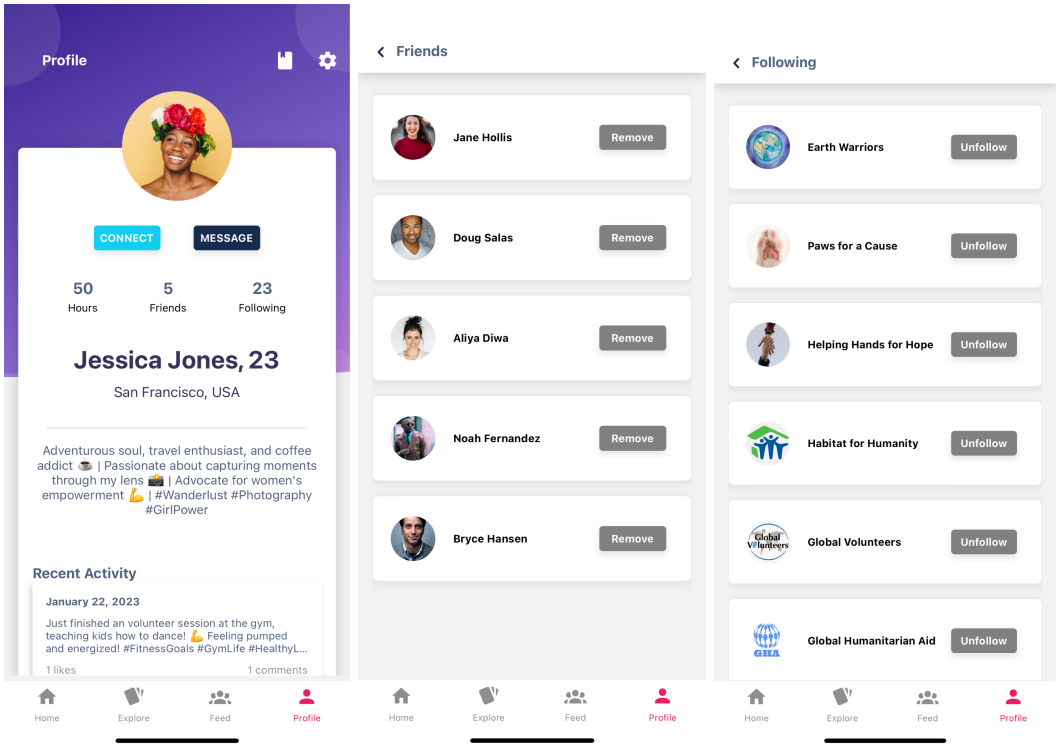


Figure 4.4, 4.5, and 4.6: This shows Profile Creation Screens.

Post and Announcement Screens

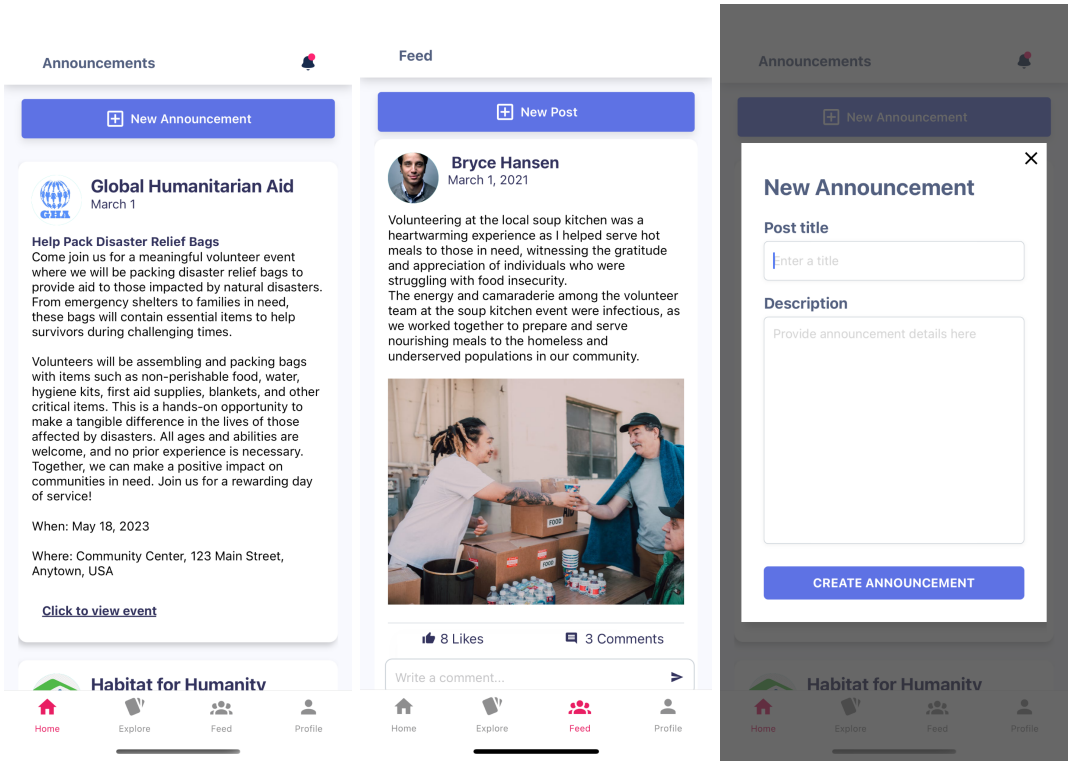


Figure 4.7 and 4.8: This shows Account Creation Screens.

B. Test coverage screenshots

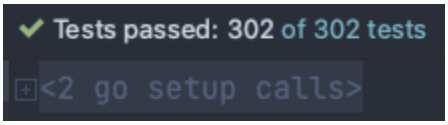


Figure 5.1: Total Tests Amount

Our test suite finished with a total of 302 tests. For our statement coverage we were able to achieve 100% statement coverage for all of our controllers, repositories, and services. Statement coverage is obtained by running `go test ./… -cover` in the `backend` folder of the repo.

```
ok      github.com/VolunteerOne/volunteer-one-app/backend/controllers   (cached)        coverage: 100.0% of statements
ok      github.com/VolunteerOne/volunteer-one-app/backend/repository    (cached)        coverage: 100.0% of statements
ok      github.com/VolunteerOne/volunteer-one-app/backend/service       (cached)        coverage: 100.0% of statements
```

Figure 5.2: Statement coverage from running go test

For our branch coverage, we were also able to achieve 100% coverage for all of our controllers, repositories, and services. `Gobco` is used to report branch coverage, and is terminal based. Unfortunately, go does not have a sophisticated branch coverage library such as Java's JaCoCo. However, we are still able to get the statistics from the command line.

```
→  backend git:(main) ✗ gobco controllers
ok      github.com/VolunteerOne/volunteer-one-app/backend/controllers   0.195s

Branch coverage: 190/190
```

Figure 5.3: Branch coverage for controllers

```
→  backend git:(main) ✗ gobco service
ok      github.com/VolunteerOne/volunteer-one-app/backend/service       1.376s

Branch coverage: 0/0
```

Figure 5.4: Branch coverage for service

```
→  backend git:(main) ✗ gobco repository
ok      github.com/VolunteerOne/volunteer-one-app/backend/repository    0.200s

Branch coverage: 60/60
```

Figure 5.5: Branch coverage for repository

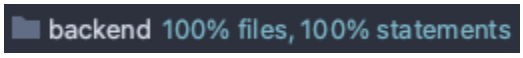


Figure 5.6: Goland coverage results of the entire Backend Folder

Overall, one of the most important changes we made was the refactoring of the architecture beginning from Design Doc 2. Without the change to the new architecture model, we would not have been able to get anywhere close to these coverage amounts.

C. CI Pipeline Prototype

Unfortunately, we were not able to get a fully working CI pipeline implemented using GitHub Actions. The high level design is the following:

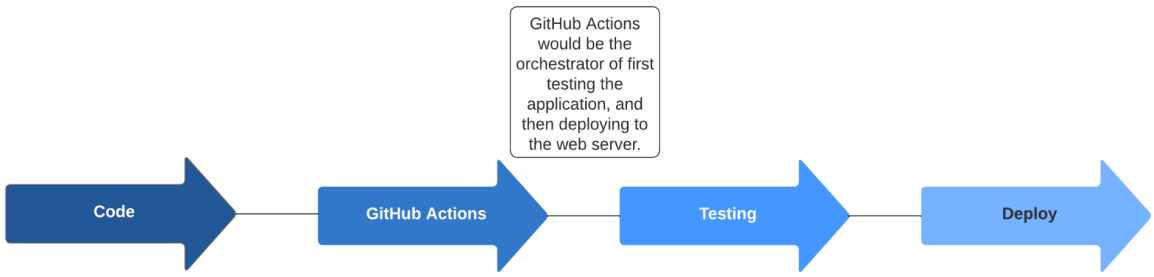


Figure 6.1: CI Pipeline

The main section of the pipeline we were focused on was for Testing, and having checks in place on pull requests and merges into the main branch to ensure that all tests were still passing when it was merged. However, the Go GitHub actions toolsets for creating go environments are not amazing, and it was extremely difficult to get the pathing and dependencies correct when running the GitHub actions bot for testing.

With more development time, we would have fixed our testing pipeline issues, as well as implemented the proper deployment onto our web server with cron jobs.

6. User Study:

A. Interview Questions

For the background information, we asked their name, if they had an abundance, some, or minimal experience volunteering, if they have ever looked for/recruited volunteer members, and if they had to schedule volunteer members for shifts. These background questions made it easier to understand how close in demographics they were in relation to volunteering and organizing volunteer events.

The interviewers also questioned, “What challenges had they faced when volunteering and/or organizing these volunteers?” By asking this question, if they had any problems that could be fixed by a feature or by a design, we could implement the solution into the design or even expand our app to include such a feature.

The following design questions were asked to the interviewees with photos of our prototype(s) to see if the design helped to encourage others to volunteer without challenges: “When you see other people or friends volunteering, as shown in the prototypes as the social tab, do you feel encouraged to volunteer? Do you think posting about your volunteer experiences will encourage others to volunteer?”, “ Does the ‘Events Sign Up’ screen lack any critical information regarding specific events you would need to know to decide if you would like to volunteer (Events Screen)?”, “Would an easy swiping feature make it easier to find potential events you are interested in volunteering in? Is there any more quick glance information you would like to see on the swipe cards?”, and “From an organization's perspective, would personal profiles make it easier for you to find potential volunteers? What other information would you like to see on personal profiles?”. If there were any issues with what the interviewee saw in the prototypes, they were highly encouraged to speak up and give detailed feedback.

Two of the questions focused on two different features. The questions, “Would auto-scheduling of shifts for potential volunteers be a welcomed feature? If you managed an organization, how would you want to see this implemented?” were used to see if an auto-scheduling feature would help organizers to schedule volunteers without hassle. It also helped volunteers know what their schedule consisted of without needing to wait for the organizer to complete a schedule. To understand a little more about how a volunteer would choose their event, we asked this to all volunteers, “If you volunteer, how do you choose what events to participate in? (ie, looks interesting, something unique and new, familiar skill, passion, close by, …)” Any other comments, or concerns were welcome at the end.

B. Demographics

The interviewees were chosen such that they had some experience volunteering or were in charge or organizing a volunteer in some way. The interviewee’s names were Leora, Kelly, Matt, Alyson, Omari, Najmah, and Jeremy. When asked whether they had minimal, some, or a lot of volunteering experience, all seven interviewees said that they had some or a lot of experience. There were three interviewees who were strictly volunteers. The other four have had to organize schedules for volunteers and/or recruit volunteers. Among those that have had to organize events are on the board of organizations such as the Filipino American Student Association and Engineers Without Borders. They were able to give feedback as the customer for the “organization/organizers” side of our product. All interviewees were able to give valuable feedback for suggestions based on their previous volunteer experiences.

C. Results

The “Social Tab” was a successful design choice as many of the volunteers said they and others would feel more inclined to volunteer if their friends were volunteering. Some have said it was due to the fear of missing out, and others said they would feel inspired to volunteer with others if they saw others volunteering for a good cause. One of the interviewees said that this did not matter to them since they did not care for social media and only volunteer for the cause. Social media is not for everyone; however, this interview proves that social media has a strong influence on the majority of the current generation. The results indicate that having a “Social Tab” will have a primarily positive effect on increasing participation of more volunteers.

The quick-swipe design/feature was a popular idea among all interviewees. Many interviewees agreed that with the time, date, and mission statement on the event card, they could quickly look at the event and swipe whether they were interested or not interested. People value their time; thus, looking at the most important information quickly is a feature that many volunteers would appreciate. In addition to making sure their time is valued, when asking what would be needed or preferred to know about an event other than what was given in the prototype, more than half of the interviewees said the end time or the duration of the event, so volunteers know the general hours of when their shift would be around. One of the interviewees praised the amount of information given in the event sign up.

Five out of the seven interviewees replied that they choose events based on events that they are passionate about and how it will help their community, for example, events with their hobbies, for a medical cause, or for well-known organizations that have had some success in reaching their goal(s) were specifically mentioned. Having personal profiles was also a well-liked idea from the interviewees as they could include interest areas to find events they were passionate about. They said it would make it easier to find volunteers given the time availability on their profiles. If a volunteer's profile promotes that the individual is passionate about a cause or volunteers for specific causes more than once, these profiles would make it easier for organizations to reach out to such individuals for more opportunities.

When it came to organizing events, the choice for largest difficulty was unanimous among all organizers: finding committed volunteers. Some volunteers may not stay their whole shift or do not show up at all. There are volunteers that cannot find rides, and because many volunteers are students, they are not able to rideshare due to lack of funds or not being old enough to use a ridesharing app without an adult. A solution to this problem was to create an auto-scheduling feature.

One of the most liked features during the interview was auto-scheduling as it would save time for organizers from manually placing volunteers in shifts, especially if the organization has multiple volunteer events and for volunteers so they can choose their own shift times. For this feature to work, the interviewees suggested a time slot design where they could choose the days/times they could volunteer for. Choosing their own shift times also increases the chances that the volunteer will commit to the whole shift.

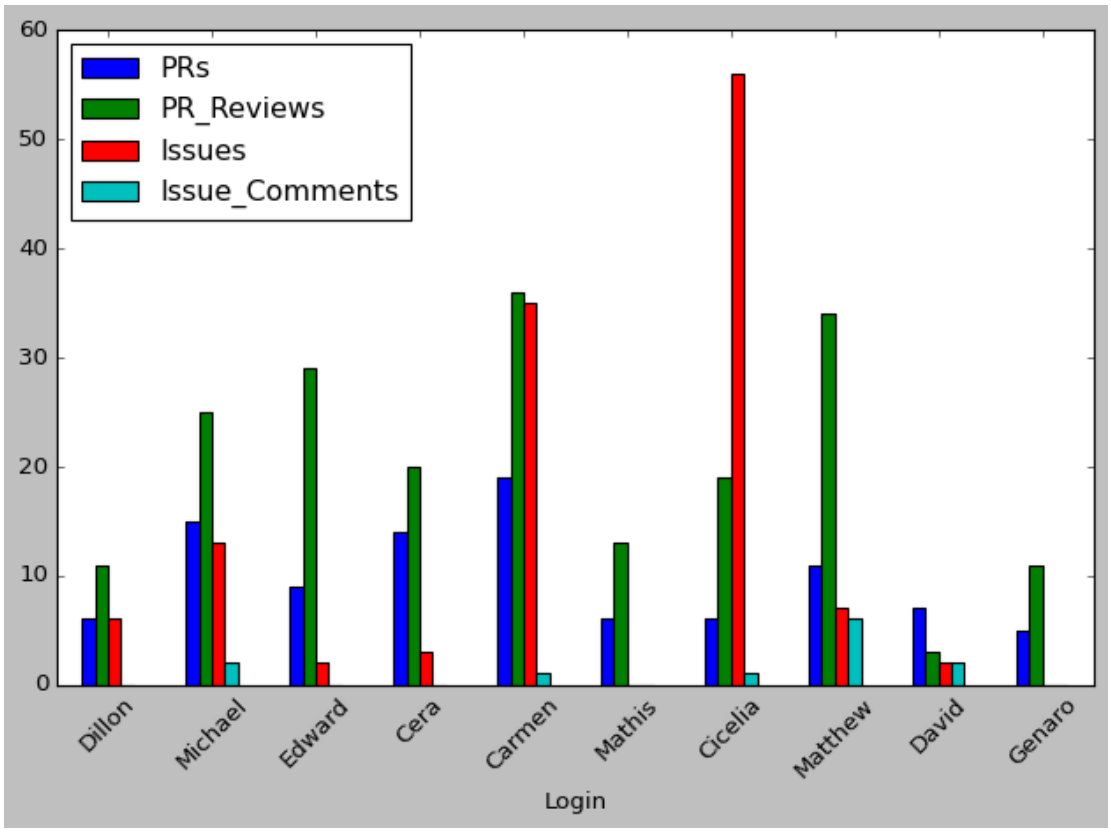
At the end of the interviews, the floor was opened up for the interviewees to make suggestions. One of the interviewees suggested including a self-cancellation feature. This would help relieve the stress of the volunteer and will notify the organization if they need to find more volunteers for a shift. Another suggestion was to implement a way to automatically include the event in their iOS and Google Calendars. The last suggestion was to add a tab for events based on locations near the individual, as this was one of the reasons why it was a challenge to find volunteers that could not find rides. With this feature, volunteers are more likely to bike or rideshare to get to the event.

7. Team Reporting:

A. Team Tasks Chart

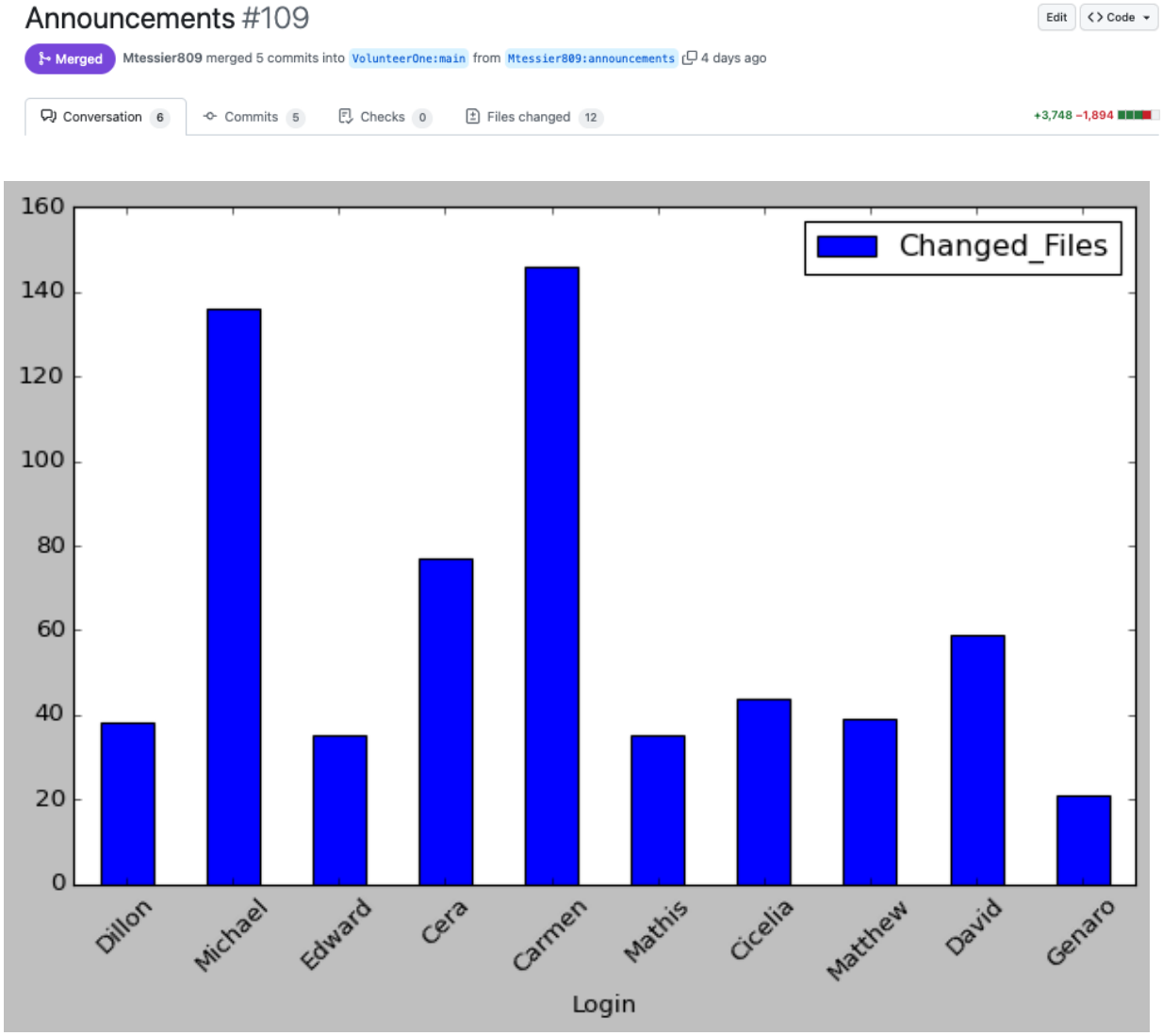
Figure 7.1: This shows our Team Contributions as recorded by our assigned tasks during our meetings. Everyone assigned a task has completed it, hence we are all at 100%. :)

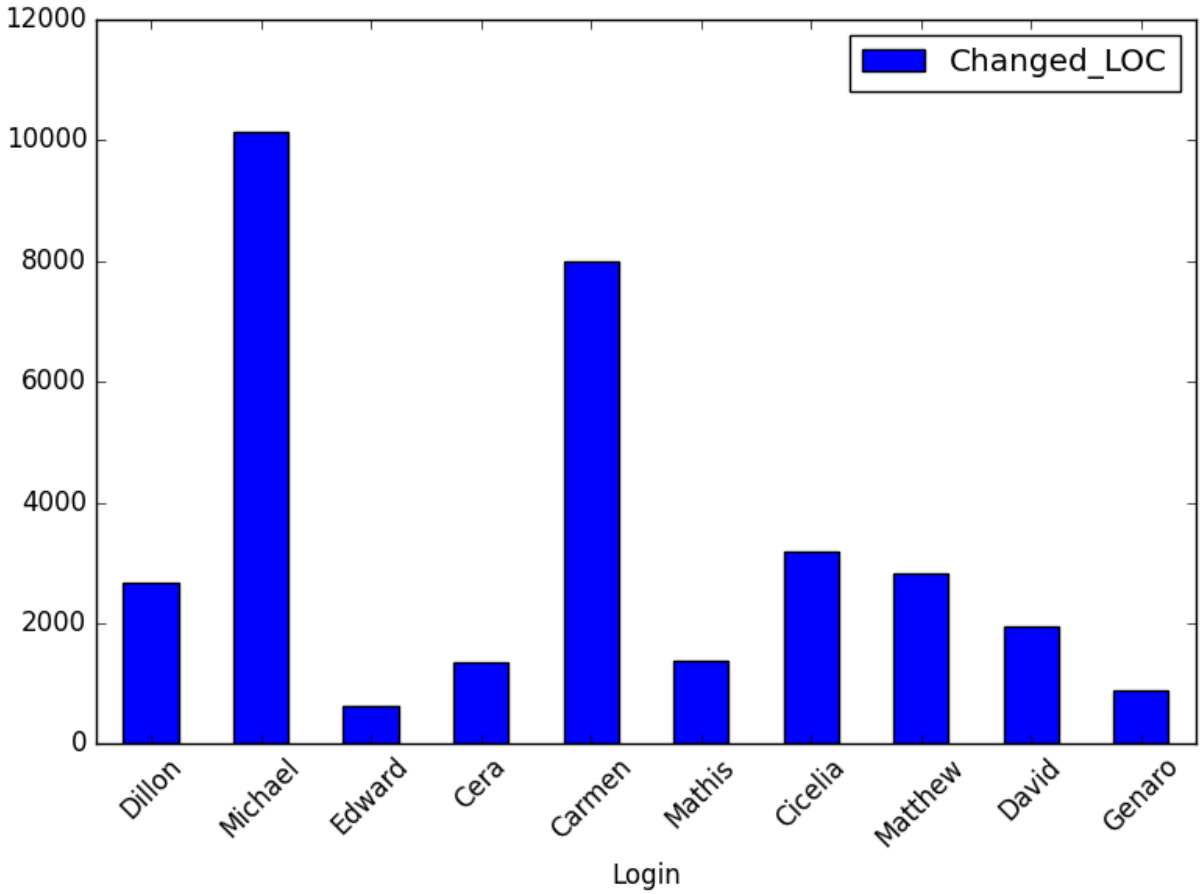
B. Team Contribution Chart



Note: In these final two graphs, the information does not appear to be 100% accurate. Mathis' information was not showing up in the final two graphs for Changed_Files and Changed_LOC in the original script. For reference, PR #109 has almost 4000 lines of code change and 12 files changed. Removing the exclusions for contributions for JavaScript (since the entire frontend is written in React Native/JavaScript) added some of the files/LOC changed for Mathis. We also had to include the ".lock" file extension to the ignored_file_extensions list, or else if someone installed a library, it would add all the lines from the installation to the individual's total LOC, which we do not want to include.

The changes for several other members also appears low, please take this into consideration and verify some more accurate numbers while reviewing the PRs in the list in the section below.





C. Individual Contributions

Carmen Lee

1. https://github.com/VolunteerOne/volunteer-one-app/pull/154
    a. Implemented Explore tab screen. The screen renders a stack of images from a database and users are able to swipe left ('Pass') and right ('Like') on the card to express interest in a volunteer event.
2. https://github.com/VolunteerOne/volunteer-one-app/pull/183
    a. Enhanced Announcement tab screen and Feed tab screen by implementing new announcement and post creation. After filling out details for a new announcement/post in a modal, an announcement or post is published to the feed on the screen.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/179
    a. Enhanced User Settings screen by restyling and adding skill personalizations. Users are able to add new skills to their profile by clicking on a button which will allow them to type in a new skill. The new skill will appear in their profile.

Michael Lazeroff

1. UC 1.1 | JWT Authentication
    a. [UC 1.1] Implemented the required authentication for the backend following the JWT flow. Implements proper delegating of access and refresh tokens, including middleware to validate tokens.
2. [UC 1] Login Controller/Service/Repository Tests
    a. [UC 1.1] Implemented 100% coverage testing for all Login related logic.
3. [UC 4.5] Friends Testing
    a. [UC 4.5] Implemented 100% coverage testing for all Friends related logic.
4. [UC 4] Posts Tests
    a. [UC 4] Implemented 100% coverage testing for all Posts related logic, which includes all posts, comments, and likes interfaces.
5. [UC 2] Organization Tests
    a. [UC 2] Implemented 100% coverage testing for all Organization related operations.
6. [UC 3] Event Tests
    a. [UC 3] Implemented 100% coverage testing for all Events.
7. [UC 1] Users Tests
    a. [UC 1] Implemented 100% coverage testing for all User logic.

Edward Sung

1. https://github.com/VolunteerOne/volunteer-one-app/pull/101
    a. [UC 1] Implemented the User table for the database in raw SQL before we switched to using GORM
2. https://github.com/VolunteerOne/volunteer-one-app/pull/103
    a. [UC1.1.1] Implemented the endpoint for logging in. A request will be sent containing the user's email and password in which the backend will check if it matches whatever is in the database. If it succeeds or not, a JSON message with 2 key values will be returned explaining a success or failure.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/126
    a. [UC1.1.2] Implemented the forgotten password endpoint. When a user forgets their password they are able to receive a code to their email in which they need to input to reset their password. Reset code is sent with a third-party email software, code is saved to the database according to the user that requested it. Requests are made using the /login endpoint
4. https://github.com/VolunteerOne/volunteer-one-app/pull/198
    a. [FS 5.1] Implemented search functionality within the front-end. Users are now able to search through all organizations or other users at once. Added a new view/screen in order to get to this functionality.
5. https://github.com/VolunteerOne/volunteer-one-app/pull/146
    a. [UC1.1.2] Implemented password reset with code verification. After entering the verification code sent to the user's email they are then able to reset and change their password
6. https://github.com/VolunteerOne/volunteer-one-app/pull/160
    a. Refactored code all previous code in order to support password hashing

Mathis Tessier

1. https://github.com/VolunteerOne/volunteer-one-app/pull/97
    a. Created Event Card component used for the posts in the announcements feed. This component is used for posting new events and announcements. Date is also generated dynamically based on how long ago a post was created
    b. Created announcements screen where you can see a list of posts
2. https://github.com/VolunteerOne/volunteer-one-app/pull/109
    a. [FS 1.0] Created tabs “Following” and “All” in the header of the announcements feed. This allows users to toggle between feed from only the people they follow and the feed from everyone in the area.
    b. Cleaned up code for announcements tab
3. https://github.com/VolunteerOne/volunteer-one-app/pull/153
    a. Created Event Details Screen and added navigation from the announcement cards to their corresponding event details. User can press “Click to View Event” to see the event details
4. https://github.com/VolunteerOne/volunteer-one-app/pull/185
    a. Created a bookmark page where a user can track the events they are interested in. Contains a list of events that they either swiped right on or liked from their announcements feed. You can click on one of the events, and it will take you to the corresponding event details page. You can also choose to remove an event from the list if you aren’t interested in it anymore
5. https://github.com/VolunteerOne/volunteer-one-app/pull/189
    a. Add functionality to the sign-up button for the event details page. When a user clicks on sign up, it asks for confirmation. If the user presses yes, then the organization is notified and the user is given confirmation that the sign up was successful. Sign-up button is then disabled
6. https://github.com/VolunteerOne/volunteer-one-app/pull/201
    a. Added a like button for the event details page and simplified styling for event details. When the like button is pressed, the color changes to red, and the user is notified that the event has been added to their bookmarks.
7. https://github.com/VolunteerOne/volunteer-one-app/pull/202
    a. added profile images to bookmarked events. Preparing the app for the demo video

David Zeleniak

1. https://github.com/VolunteerOne/volunteer-one-app/pull/81
    a. This pull request is for the initial backend project creation. For this chunk of work a lot of research was done to decide the projects initial architecture. It sets up the following packages:
        i. Server
            1. API Application entrypoint, along with the applications router.
        ii. Database
            1. Handles creating the connection to the database, and distributing that object through the application.
        iii. Models
            1. Database tables described in code.
            2. Migration of the described database state.
        iv. Controllers

        1. Endpoint functions for the API's router to use.
    b. The PR also containerizes the application using docker, and implements the database with the API using docker compose. This will allow the deployment of the application in production to be simplified
    c. Enables configuration to be customizable using environment variables
2. https://github.com/VolunteerOne/volunteer-one-app/pull/115
    a. [UC1.4] Implemented the model and controller for organizations. This pull request creates the database table for organizations. Also allows for requests to be made to the /organizations endpoint to Create, Read, Update and Delete organizations from the database.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/164
    a. [UC1.4] Refactor of the organizations endpoint to match the applications architecture. Adds a Service and Repository to the pattern and alters the controller to use this pattern.
4. https://github.com/VolunteerOne/volunteer-one-app/pull/203
    a. [UC3] Creates a model, service, repository, controller and endpoint for registering users to events. The database table will track all event registrations using the user id and event id. The endpoint allows for Create, Read, Update and Delete operations to be performed.

Matthew Yapjoco

1. https://github.com/VolunteerOne/volunteer-one-app/pull/145
    a. [FS4.1 & FS4.2] View Friends/Following Screen: Combined Friend and Following component cards and used a boolean to differentiate what type of card to use depending on the user type. Added an alert popup to verify if the user is sure they want to unfollow or remove organizations or friends.
2. https://github.com/VolunteerOne/volunteer-one-app/pull/157
    a. [FS3.2] New Event Modal: Created the New Event Modal, added the button to the feed page and implemented an image picker to the modal.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/159
    a. [FS3.1] Revised the New Post modal using new standards.
4. https://github.com/VolunteerOne/volunteer-one-app/pull/172
   and https://github.com/VolunteerOne/volunteer-one-app/pull/173
    a. [FS4.1 & FS4.2] View Friends Screen & View Followers Screen: Linked correct profiles from the View Friends screen and View Followers screen from any profile. Uses stack screens so users can go back by swiping the screen right.
    b. [FS4.4] Notifications Screen: Linked notifications to relevant profiles so users can quickly view the profile of a notification.
5. https://github.com/VolunteerOne/volunteer-one-app/pull/192
    a. [FS0.1] Validate Login Credentials: Enabled login validation with sample usernames and passwords.
6. https://github.com/VolunteerOne/volunteer-one-app/pull/194
    a. Update Constants Profiles: Create sample profiles for volunteers and organizations with descriptions and recent activities.

Cicelia Siu

1. https://github.com/VolunteerOne/volunteer-one-app/pull/2
    a. [UC 1] Implemented the User table for the database in raw SQL before we switched to using GORM
2. https://github.com/VolunteerOne/volunteer-one-app/pull/105
    a. [UC 1.2] Using the new GORM format, user account creation, reading all account information, reading a specific account information, updating a specific account's information, and deleting an account record were implemented.
    b. Have yet to implement the unit testing reformation.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/141
    a. [UC4.5.1 and UC4.5.2] CRUD for friends with a status update between two users or organizations.
4. Scheduled interviews and interviewed potential customers for our product. In addition, I completed the two page summary of the user study.
5. Planned and set up projects and issues in Github.
6. Set up the DigitalOcean server with Dillion with the public keys.
7. https://github.com/VolunteerOne/volunteer-one-app/pull/150
    a. [UC 1.2] Refactoring user account creation for a new updated table
8. https://github.com/VolunteerOne/volunteer-one-app/pull/163
    a. [UC 1.2 - 1.3] Update account creation for unit testing.

Cera Samson

1. https://drive.google.com/file/d/1XIxWBsbeUadtWj45t0HURjW3-yqiUmWK/view?usp=sharing
    a. Created the demo video for use in the presentation. I recorded the app walkthrough using Expo Go. I edited the clips and created the voiceover for the video.
2. https://github.com/VolunteerOne/volunteer-one-app/pull/170, https://github.com/VolunteerOne/volunteer-one-app/pull/181
    a. Feed Tab / Friends Screen (FS 3.0) - Expanded on existing Reaction component functionality:
        i. Number of likes are predefined for dummy posts on the Feed page, and increments on press.
        ii. The Comment bar has been entirely redesigned to be loaded on default below each post. The user can type a comment in the text field, and after submitting, the number of comments displayed is incremented.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/169
    a. Explore Tab Screen - This PR adds event details as an overlay to images on the explore tab. The component reads data from a .json file, which includes the title, organization, post date, and cause areas. This overlay works with the swipe functionality of the Explore page.

Dillon Harder

1. https://github.com/VolunteerOne/volunteer-one-app/pull/207
    a. [UC 2.1.1 & 2.1.2] This PR was meant to refactor and organize the code so it can be abstracted into several different layers. We set the code up in a

Controller/Service/Repository pipeline so that it would be more secure, easier to handle, and make testing simpler. This specific change was regarding OrgUsers.

2. https://github.com/VolunteerOne/volunteer-one-app/pull/214
    a. Someone else's PR accidentally added some unwanted changes and reverted a file to its previous state. This PR was meant to resolve that by reverting it to its updated/changed state. This is a somewhat small PR, but it was still an important fix to ensure our endpoints were all working correctly.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/230
    a. [UC 2.1] This was my largest change, adding unit testing for all of OrgUsers. It contains 100% statement and 100% branch coverage for each layer of OrgUsers (controllers/service/repository layers). This was a challenge due to the new language, frameworks, and overall lack of testing support/documentation in Go, but it was rewarding to complete.
4. https://github.com/VolunteerOne/volunteer-one-app/pull/231
    a. This PR is not yet submitted as of writing this (though it will be by the time of our presentation and the competition). Its description will also be updated. It is meant to add support for messaging between users.
5. https://github.com/VolunteerOne/volunteer-one-app/pull/232
    a. This PR is also not yet ready to merge, but it will be by the time of our presentation/competition. Its description will also be updated. It is meant to add all unit testing for the Messages endpoint, another very intensive PR due to the difficulties of unit testing in Go.
6. In addition to my PRs, I have also helped contribute towards the slide deck for the presentation, doing the content for several slides as well as designing the entire template and converting every slide into the new format. I also helped with the foam-mounted poster board for the senior design competition. Finally, there were many tasks involved with the server integration on Digital Ocean; this involved installing Docker, setting up the repository on the server, getting SSH keys for anyone who needed access. Unfortunately, the GitHub Actions pipeline (mentioned in another section) had some unforeseen issues, so the CI/CD is not fully implemented to run our unit tests; however, the server is capable of running the backend and serving API endpoints for the app.

Genaro Lopez Baez

1. https://github.com/VolunteerOne/volunteer-one-app/pull/111
    a. Onboarding Screen, Forgot Password [0.4] - In this screen users will be asked to input their email. Once submitted, the screen shall navigate to [FS0.5] where users will be asked to create a new password for their accounts.
2. https://github.com/VolunteerOne/volunteer-one-app/pull/110
    a. Onboarding Screen, Create New Password [0.5] - In this screen users will be asked to input a new password to replace their old password. Password shall be compared and user to be taken back to the Login screen.
3. https://github.com/VolunteerOne/volunteer-one-app/pull/165
    a. Profile Screen, Account Settings Screen [4.1] - From this screen user will be able to edit their user information. This including being able to edit their profile image, name, bio, location, and email.

D. Success Report

Frontend

Initially, our main goal for the mobile application was to create a minimum viable app that contained all the necessary screens correctly routed together. As we have completed that goal with more than 15 different screens and seamless navigation, our next goal was to enhance our app by creating an accessible, interactive, and personable experience for our users. We continued to deliver features that would serve those objectives through our quick swipe, post reactions(like/comment), and user skills feature. Since making those additions to our app, we have met our goals and objectives.

Although we created an intuitive volunteering app, we acknowledge that we could have expanded it further to benefit our target audience even more. We had considered features like hour logging and an event scheduler, but due to time constraints and the complexity of previous and new tasks, we were unable to implement them.

Our approach to creating an effective user interface for our mobile app set us up for success. By using React Native as our framework, we utilized its strength of code reusability. We were able to plan and structure our project with reusable components and it not only made the project more collaborative and efficient, but it also made the code more readable. By starting with a well-thought out and structured project skeleton, it made it possible to develop and build upon one another's contributions with ease. Our approach worked well for our team dynamic and we would strongly recommend planning a project with an approach similar to ours.

Backend

Since the first design document, a large portion of the API is complete. We have implemented all the requirement endpoints needed. The JWT protocol authentication has since been added as a feature so our endpoints are secure through the authentication middleware. Since this is implemented it was very easy to apply to all of the protected routes.

Currently, most of the test coverages are relatively high in comparison from the last document. Refactoring for unit testing has been the hardest factor since adopting the new Controller, Service, and Repository architecture. Because we chose to use GORM and Golang, the testing portion of this project has been the longest and most difficult problem. In addition, the Docker setup was highly faulty on some members' computers. It mainly worked half the time, but when it did. We were able to test the code using GOBCO, Postman, and etc.

Our final backend was successfully completed with less problems that we had initially thought. Thankfully, we were able to get all APIs working. The full backend team agrees that learning GO and GORM for this one project was a risky idea. Only one person fully knew how GO and GORM worked, while the rest of the team had to learn it while

producing results. We were all able to finish our tasks, but had we chosen a language we all were comfortable with, the project would have gone faster and smoother. Furthermore, we should have emphasized deploying our web server and CI pipeline from the very beginning, that way we could have iterated more slowly and methodically.

Full Team

The full team has gone through this semester with many sleepless nights while trying to learn new tasks that they have never tried before. We as a team were able to complete the app, and complete issues at hand as a team. There were many miscommunication issues, however, we all were able to work through it. Our teams were well put together and think it was the best idea to split the team from backend and front end.

8. Conclusion:

Design Portfolio 3's focus was merging the frontend and backend portions of the project in addition to completing any final remaining tasks necessary for a functional demonstration of the application. Upon determining what the contents of the demonstration video should be, we identified several overlooked and absent elements from the application as it was.

By using the outline of our demonstration as a guideline, we focused our efforts on the fundamental and essential features of the application to demonstrate general functionality, communication, and interactivity. This allowed us to hard-code trivial functionalities while integrating the critical frontend and backend components to emphasize the primary objectives of VolunteerOne.

Moving forward, there are still numerous upgrades and fixes that need to be addressed. In its current state, future improvements are necessary to make the application fully functional and further enhance the application. If the application were to be launched, ongoing updates and maintenance would be required.

9. References:

[1] *Remarkable outcomes*. VolunteerMatch. (n.d.). Retrieved March 28, 2023, from https://www.volunteermatch.org/.

[2] Team, V. (2022, September 19). *9 Problems You May Be Facing With Your Current Volunteer Management Process*. Volgistics. Retrieved March 28, 2023, from https://www.volgistics.com/blog/problems-current-volunteer-management-process/.

[3] Bussell, H., & Forbes, D. (2002). Understanding the volunteer market: The what, where, who and why of Volunteering. *International Journal of Nonprofit and Voluntary Sector Marketing*, *7*(3), 244–257. https://doi.org/10.1002/nvsm.183.

[4] Bang, H., & Ross, S. D. (2009). Volunteer motivation and satisfaction. *Journal of venue and Event Management*, *1*(1), 61-77. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=57bcd374bf289b02f99651d06adda4bdde4208f5.

[5] McAllum. (2014). Meanings of Organizational Volunteering. Management Communication Quarterly, 28(1), 84–110. https://doi.org/10.1177/0893318913517237.