University of Arkansas – CSCE Department

Capstone II – Final Report – Spring 2022

Fayetteville Public Library: Volunteer Time-Tracking and Management System

Alexis Carter, Kagen Crouch, Bradley Lithalangsy, Nathan Secrest, Steven Trinh

Abstract

The Fayetteville Public Library's vison is to be powerfully relevant and completely accessible to its attendants and community. Their mission is to strengthen the community and empower citizens through free and public access to knowledge. To accomplish this vision and mission, the public library offers volunteer opportunities for people looking to participate in their community. Currently, the system set in place for managing and quantifying volunteers is outdated and must be operated on in-house machines. This makes the system slow and not easily accessible to users. Due to this system inhibiting the performance of the library along with its ability to impact the community, the library has tasked our capstone team with the creation of a new application for management of its volunteers.

The new application will be supported on both web and mobile devices, allowing for users to easily access this system inside and out of public library facilities. The system allows individuals to register to become volunteers, once approved, the users can sign into their account. From the home page, users can view future volunteer opportunities, enroll to participate in volunteer opportunities and view their volunteer history. This system will maintain key features of its predecessor, while offering a more streamlined approach to volunteer management. Public library staff and/or admins of the system will have access to view and modify volunteer information including name, address, email, and other information requested at volunteer registration. The system also allows for event information to be managed and viewed by admins of the system. To allow for empowerment of citizens through free and public access to knowledge, the system has been developed in an easily modifiable manner to be used by other volunteer organizations in the Northwest Arkansas region who are in need of a volunteer management system.

1.0 Problem

Library volunteers provide extremely valuable services to the community that the library would like to track and quantify. Currently the public library does not have an easily accessible way for volunteers to log their hours since the current volunteer tracking system can only be

accessed on-site at library computers. As a result, it is important for the library to provide an application that allows users to track their volunteer hours, achievements, and milestones from anywhere while providing a more user-friendly interface. Library personnel would also like to see a solution where the application we build could generalized for other organizations to rebrand as their own, so that it is able to be distributed to other non-profit, volunteer-oriented communities with similar needs for time-tracking and data management. This problem, if not solved, would mean the library would have little to no ability to quantify the overwhelming impact of volunteers on library programs and productivity in a modern and capable manner.

2.0 Objective

The objective of this project is to create a new, user-friendly application for multiple operating systems that will modernize the outdated system currently in use at the Fayetteville Public Library. This new system would improve the everyday experience of volunteers and administrative coordinators alike. This is due to its ability to allow for users and admins to access and interact with the system efficiently and outside for library facilities. To provide relevant tools for volunteers, the application must have the ability to allow users to register to become a volunteer, register for future events, log their volunteer time of an event, and view their past volunteer history. In addition, the database will need to be able to manage a large-scale user base with irregular, heavy traffic.

The administrative side of the application should provide admins with the proper permissions the ability to override the volunteers manually logged time, change volunteer status, manage, and delete volunteer accounts. Administrators should also have the capability of creating, modifying, and removing volunteer events. Both the administrative and volunteer views of the application should be clean, intuitive, and easy to understand for non-technical users.

3.0 Background

3.1 Key Concepts

PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system. It employs and extends SQL to enable the system to safely store and scale complicated data. It was developed in POSTGRS project at the University of California at Berkeley in 1986 and has over 30 years of development. It runs on all major operating systems. It allows administrators to protect data while building fault tolerant environments. It is highly extendible, allowing for indexes and defined API's.[1]

Python Django

Python Django is a high-level Python web framework which works to encourage rapid development with pragmatic designs. It is free and open source and written by experienced developers to enable users to focus on their own projects instead of the intricate details of web development.[2]

Psycopg2

Pyscopg2 is a popular PostgreSQL database adaptor for Python. It is Unicode and Python3 friendly. Primarily, it features the complete implementation of Python DB API 2.0. It

was designed for multi-threaded applications that create and delete cursors frequently, such as using “INSERTS” or “UPDATE” commands. It was implemented in C and employs server-side and client-side cursors.[3]

Flutter

Flutter SDK is an opensource, cross-platform tool created by Google that allows users to develop applications for desktop, mobile, and embedded devices using a single codebase, allowing user to hasten development. Google’s User Interface toolkit employs Dart, a fully object-oriented programming language.[4]

Material Design Google

Material Design is a design language created by Google in 2014. This system is used to flesh out the designs of a mobile app. Material Design uses grid-based layouts, animations, and transitions to create a consistent and fluent user interface that is also especially pleasing. Material Design uses designs that reflect the physical world, specifically the medium of paper and ink. It also maintains a hierarchy within its elements to create a focus for the viewer’s experience. In addition, Material Design has a built-in state system that has component libraries available for Android, iOS, and Flutter. Those components have the potential to cover display, navigation, actions, input, and commination. If necessary, Material Design also has a theming option that allows for a reliable consistency throughout the system.[5]

To use Material Design within Flutter, we must import a library that gives us access to the widgets (GUI elements) encapsulated by Material Design. From there on, we have access to customizable properties that allow us to stylize our app as we wish.

PgAdmin

PgAdmin is deployed along with PostgreSQL as a management tool. It can be run either as web-based or desktop application. It’s multiplatform and is compatible with Linux, Microsoft Windows, and macOS. It has multiple deployable models, including a desktop and server mode. In addition, there’s routine maintenance, such as auto-vacuum management and scheduling agents which will be useful while interacting with such a large number of volunteers.[6]

3.2 Related Work

GivePulse[7] is a volunteerism tracking system used by many organizations at the University of Arkansas. However, we cannot use it because existing data from the library’s current system cannot be transferred to GivePulse. Also, GivePulse is proprietary software so we would not be able to give the software to other libraries and organizations for their own adoption of the system if we went this route. Almost all the existing solutions require a paid subscription or have a “freemium” monetization scheme, which we want to avoid. Ours will be different because it will cater to the needs of the library and be open source to adapt to the needs of other libraries/organizations.

4.0 Design

4.1 Design Goals

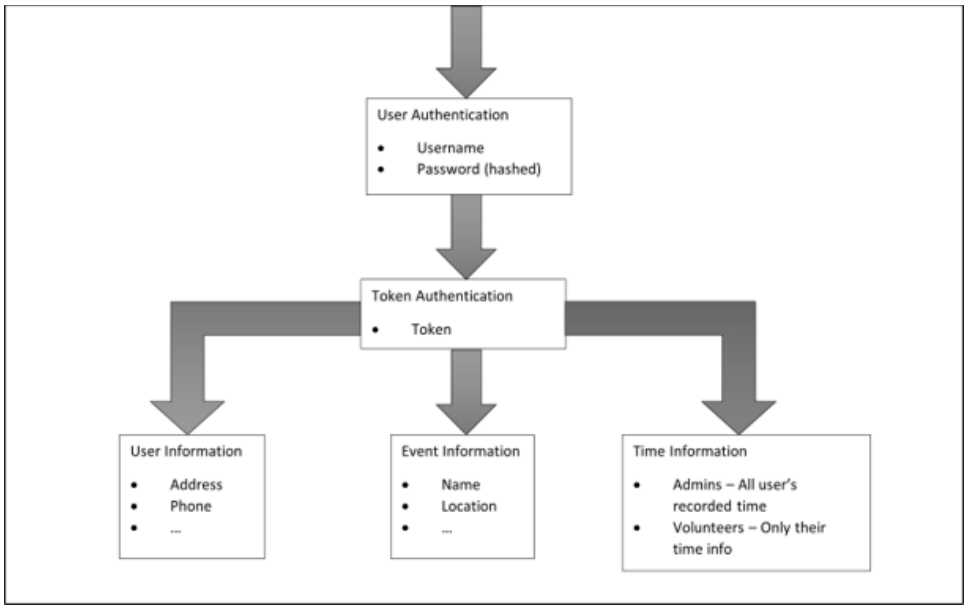
The design goal for this project is to have a system that can allow volunteers to log in, select from a list of events, and be able to clock in and out of these events to record their

volunteer time. The system should also be capable of acting as a management system for volunteers and the events they participate in. To accomplish this goal, users must be able to become administrators of the system in order to create and manage events, as well as other users. The system is also required to be accessible from any device, web or mobile, to promote ease of use.

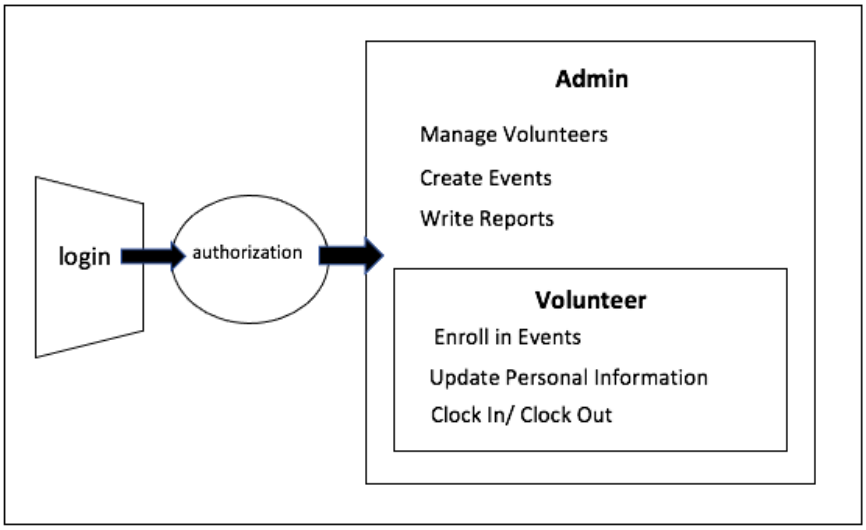
4.2 Detailed Architecture

To accomplish the goal of this project, the architecture of the application had to perform similar tasks as the system currently in place at the Fayetteville Public Library. This new system has additional features and could function from within and outside of library facilities. Flutter allows our system to have a front-end application that functions across web, Android, and iOS devices. We are using Material Design by Google, which supports Flutter in creating simple user-friendly interfaces. Our volunteer management system uses PostgreSQL to create a relational database that, once paired with psycopg2, allows for our front end and back end to work together.

Database Design



Front End Design



Our PostgreSQL database is hosted on Heroku and uses Django's Rest Framework to authenticate the users accessing the data. The specific authentication it uses is called OAuth and works by creating a user table that holds a username, hashed password, user id, and other user information. This table is populated whenever a new user account is created and acts as a foreign key to the authentication token table, which is also automatically generated. When a valid user signs in, a token is generated in the authentication token table and associated with the user id of the user that logged in. To access the rest of the tables, the program will have to send a request to the API with this token in the header. When received, the token is verified as a valid token. Then it will perform the requested actions and return the appropriate information from the database. If the token is not provided or is invalid, the data will not be fetched. In this case, an error will be thrown attempting to access the database.

Upon opening the application, the user of the system will be prompted to login to their account or register to become a volunteer. Admins and volunteers alike will use the same login screen to sign in and register for an account. This results in a simple front end and the ability to transition current volunteers into administrators of the system. Accounts for users allows volunteers to uniquely sign up for events, check into events, and view their unique volunteer history.

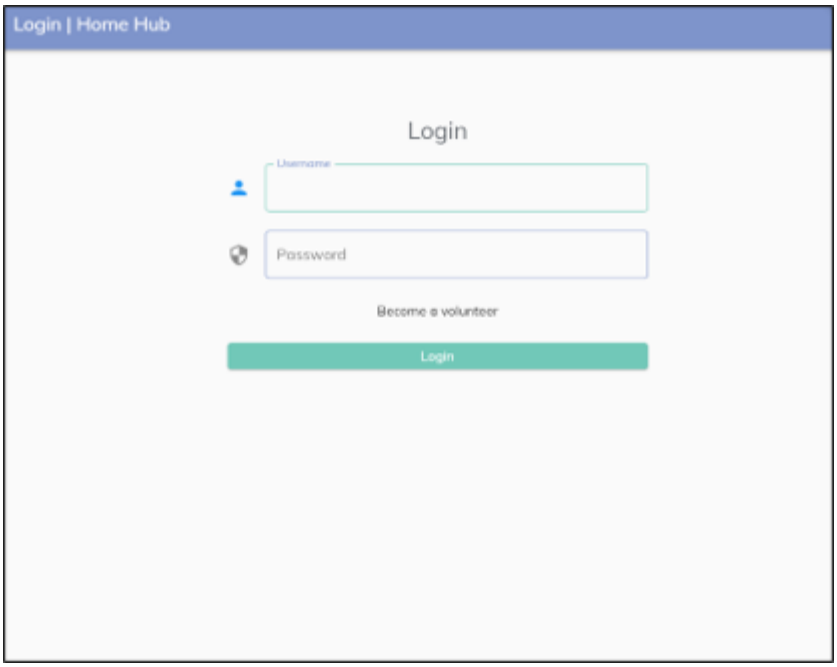


Figure 1: Login page

When clicking the login button, the password is encrypted and sent to the API along with the username to verify that the login credentials are correct. If they are not correct, an appropriate error message will be displayed. Otherwise, a new user object with user id, token, and username will be returned in the response. When valid login credentials are provided, the login screen will route users to their relevant homepage dependent on their status as a user. If there are no stored users with the given login information, the system will not allow the user to enter the system and prompt the user to register to become a volunteer. Once the user has interacted with the "Become a volunteer" button, they will be routed to the sign-up page. Here the user can input relevant volunteer application information that our system will then use to create a user account. They will be added to the user authentication table and will be able to obtain a token after logging in. After the user account has been approved and given authorization to become a volunteer by an admin, they will be able to login through the login page.

Figure 2: Sign up page

After logging into the system, the program will check if the username is categorized as an admin or volunteer level user. Depending on the user's status (volunteer or admin), the next page they will be sent to will differ. Volunteer status users will be routed to the user home page where

various actions can be performed. Volunteers can navigate to currently available events, events they are currently enrolled in, and their personnel volunteer history. Users are also able to directly clock into events via the clock-in navigation option. In addition, all users, admins and volunteers alike, can view their user account page and user settings page from their relative home menu.

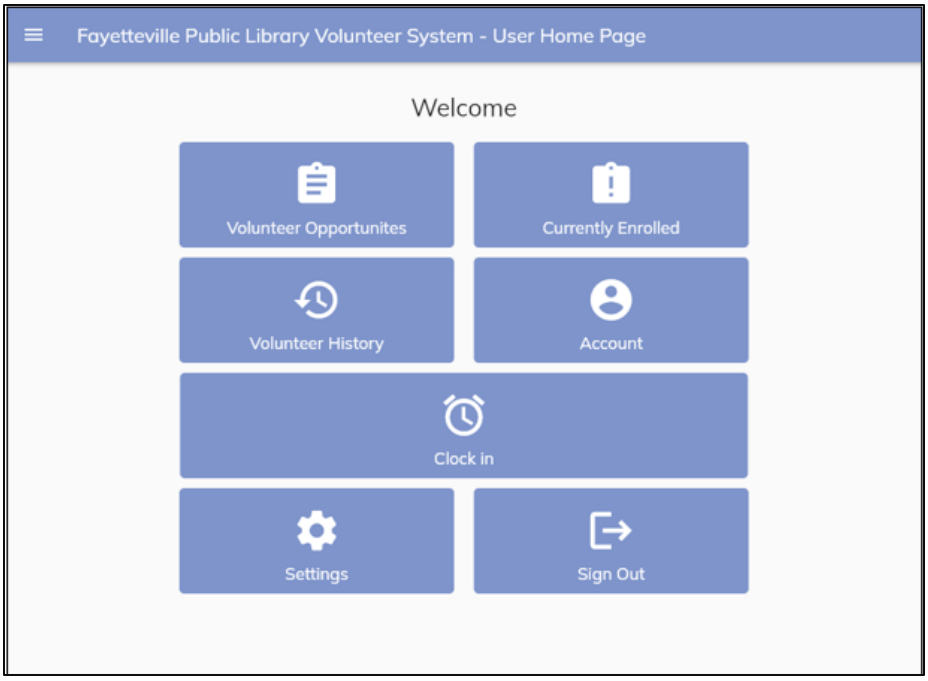


Figure 3: User home page

If a volunteer navigates to volunteer opportunities the user will be presented with a list of currently available events posted by administrators of the program. This list of events contains the title of the event, the date and time it occurs and ends, the organizer of this event, the location and a description of the event. Each event contains a unique sign-up button that will send event information to relevant database table to store that the user has registered to participate in the event. This event removes itself from the user's view to prevent confusion when signing up for events in the future.

Figure 4: User volunteer events page

Volunteers who are routed to the currently enrolled volunteer opportunities will be able to view events their account is currently registered to participate in. This view of events is formatted identically to the users view of currently available volunteer events. While the contents of this page are similar to the view of currently available events, this view offers the volunteer an option to clock into each event. When the volunteer selects this option, they are routed to the clock-in page with relevant event information being provided to the clock in page.

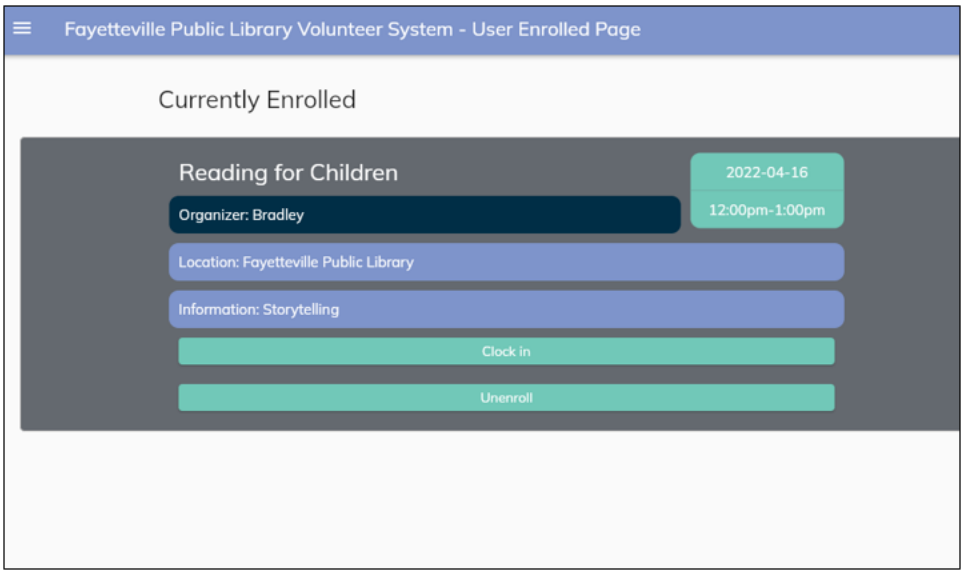


Figure 5: User currently enrolled events

Due to the unique user authentication, users can access a view of their past volunteer history. This page presents event information in the same manner as current volunteer events and enrolled volunteer events, but also includes information on the time that was recorded from each event the user has participated in. The purpose of this page is to give volunteers a user-friendly record of their volunteer history using the system.

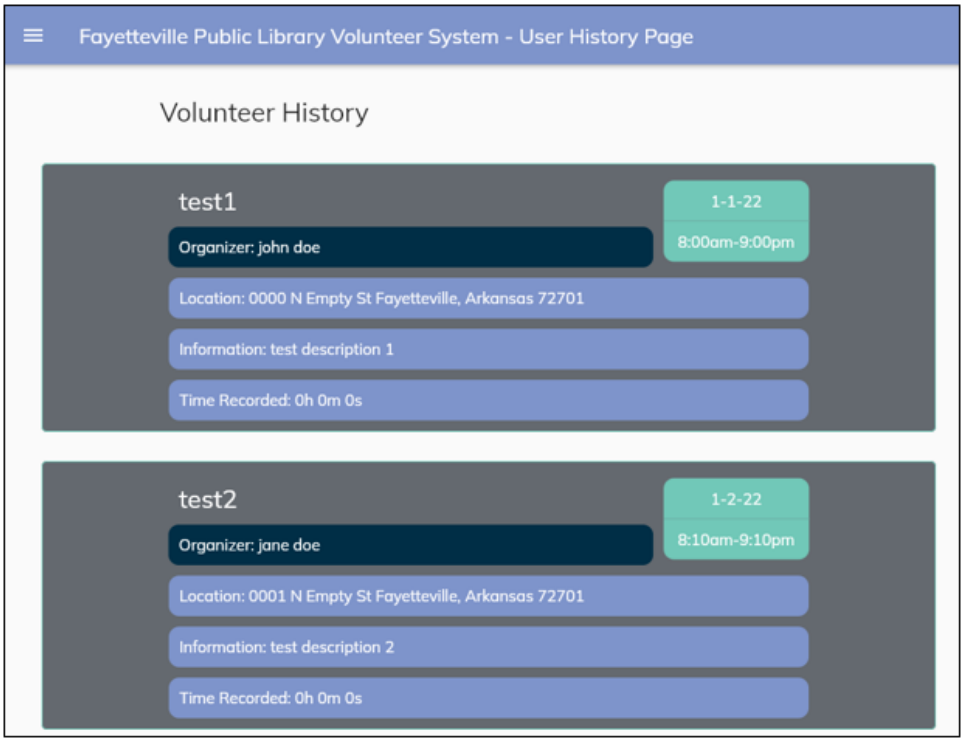


Figure 6: User volunteer history page

Once volunteers have registered for an event, they can move to the clock-in page. This page allows volunteers to select from their currently enrolled events and record time for the user for a particular event. This page also allows for a stopwatch like function, housed in the clock-in button, that allows for recording volunteer time automatically rather than manual submission. After the user has recorded their volunteer time, this information can then be sent to the database from the submit button.

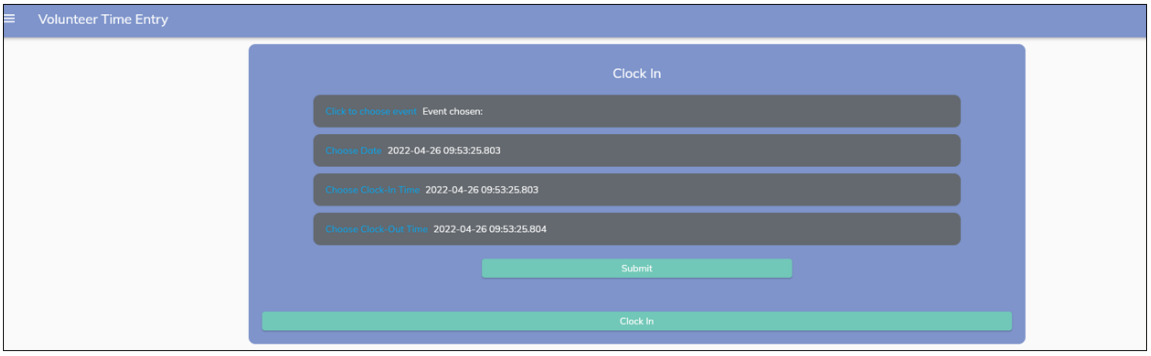


Figure 7: User clock in clock out page

There are four fields for the user to select values from: event name, date, clock-in time, and clock-out time. When the user clicks on each of the buttons associated with each field, they have the option to specify details about their volunteer service.

Figure 8: Dropdown menu and search bar allows the user to search for the event they volunteered for.

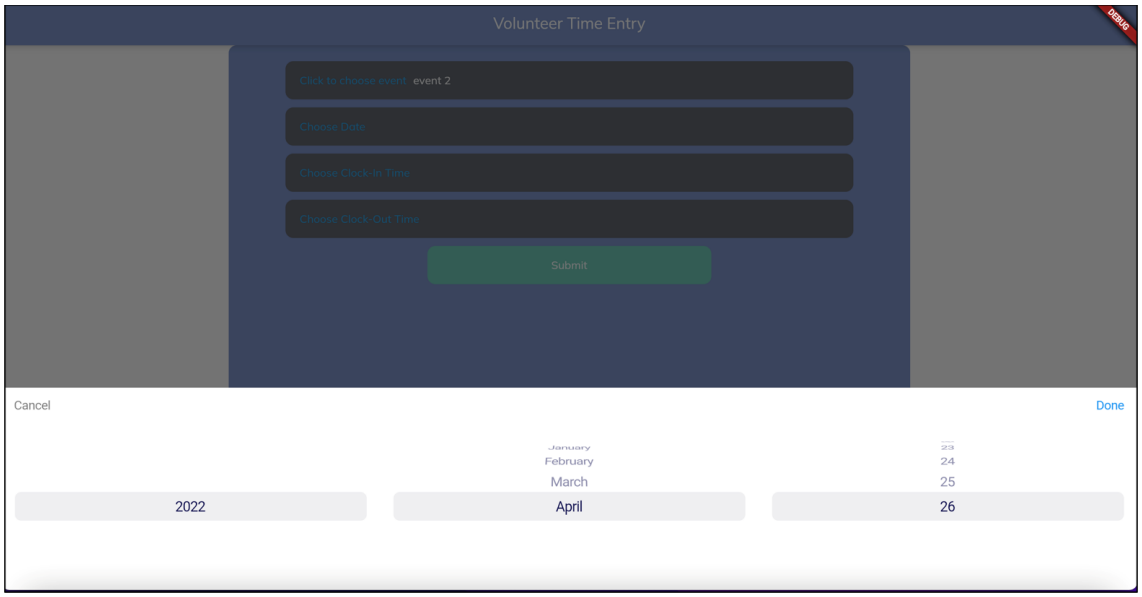


Figure 9: Scrollable date picker allows the user to pick the year, month, and day they volunteered.

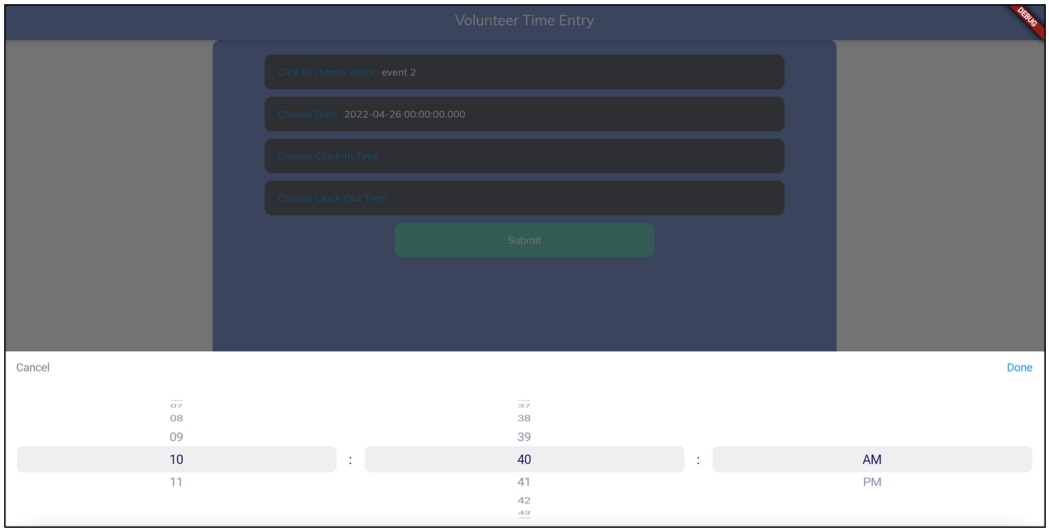


Figure 10: Scrollable time picker allows the user to pick the time they clocked in. The same design is shown when the user picks a clock-out time.

Volunteers have access to view personal account information regarding their account via the account profile page. This page displays all currently available information the database

stores regarding the user. From this page, the user can access a user settings page through the "Edit account details" button to manage and modify their relevant volunteer information as well.

Figure 11: User account page

Account settings for volunteers allow users to change personal information either in mass or individually. Each user attribute is given their own section to for a new attribute value and a relative save button to pass this information back to the database. The user is also able to change multiple attribute fields and send the various new attributes back to the database using the save all button appearing at the button of the page.

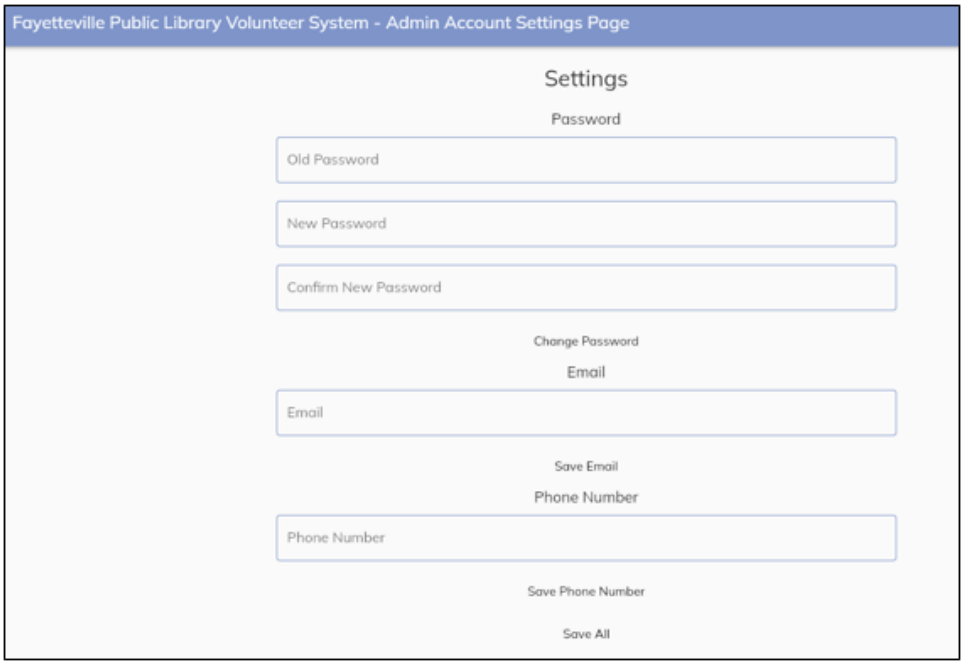


Figure 12: User account settings page

Users of the system who have administrative status are routed to the admin homepage instead of the user homepage from the login page. This admin homepage functions similarly to the user homepage but differs in its contents. The content of this page allows administrators to access lists of events, current volunteers, and current admins.

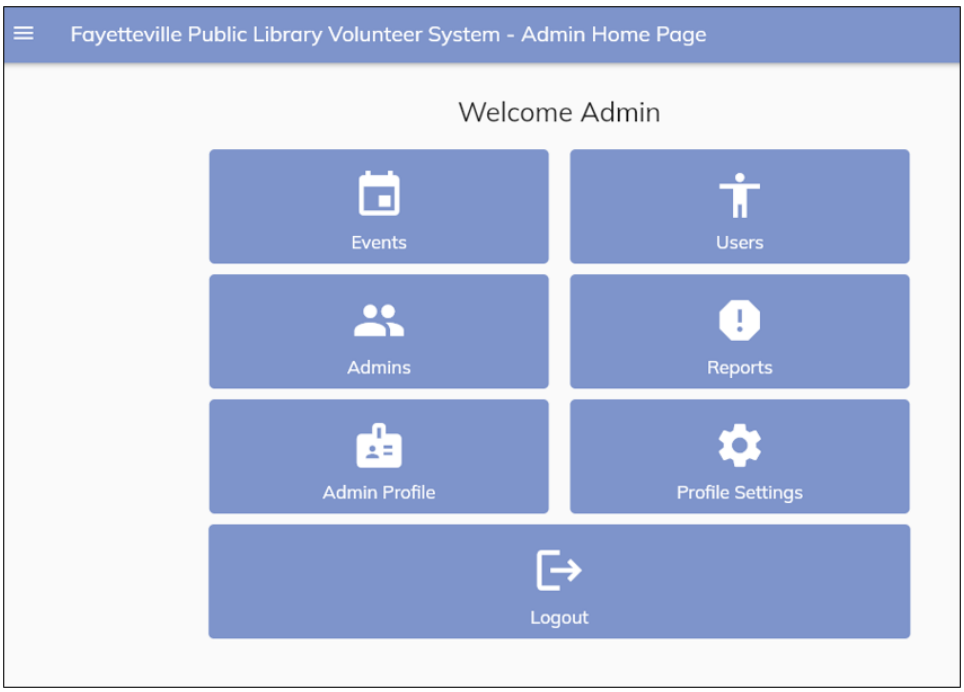


Figure 13: Admin home page

The administrative view of events presents events in a user-friendly way like that of event displays for user pages. Events can also be added to the list of currently available events by interacting with the "Create event" button. When this button or any display of a specific event is pressed, the admin is then sent to the edit event page.

The eventInfo model is used to convert the information that the admin sees into Json data that the backend event model can use. These contain information fields: id, organizer, event name, description, start date, start time, location, and completion status. Using these, the admin can create new events or manage current events.

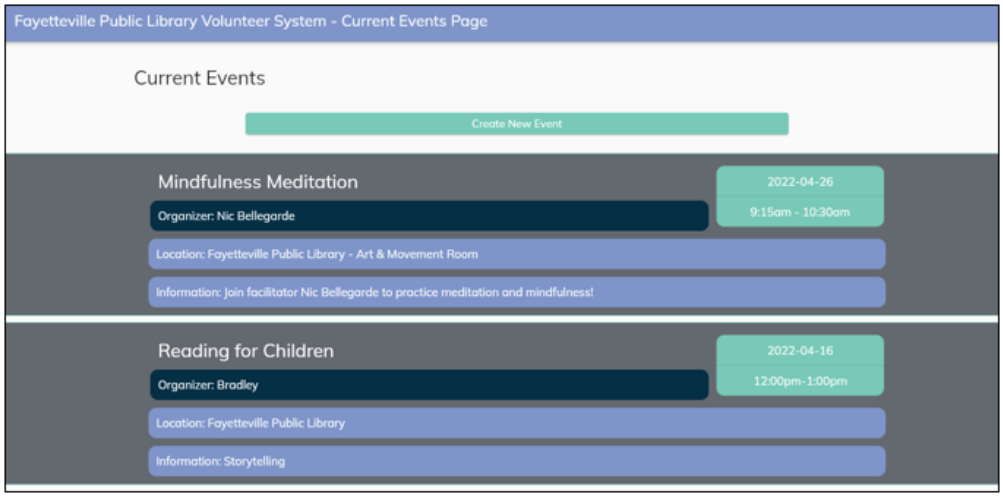


Figure 14: Admin events page

After the administrator of the system has interacted with either the create event button or a specific event, they are brought to the edit event page to modify the attributes of the event. This

page presents each event attribute title, previous value, and a text form for new event values. If this page is accessed by the “create event button”, the previous value of event attributes is given a null value. If this page is reached by interacting with a particular event, that event information will be passed to the page for modification. Once the user has finished filling out data for the event, they can submit the event to the list of current volunteer opportunities. There is also a “Cancel” button present on the page that will return the user back to the admin view of the events.

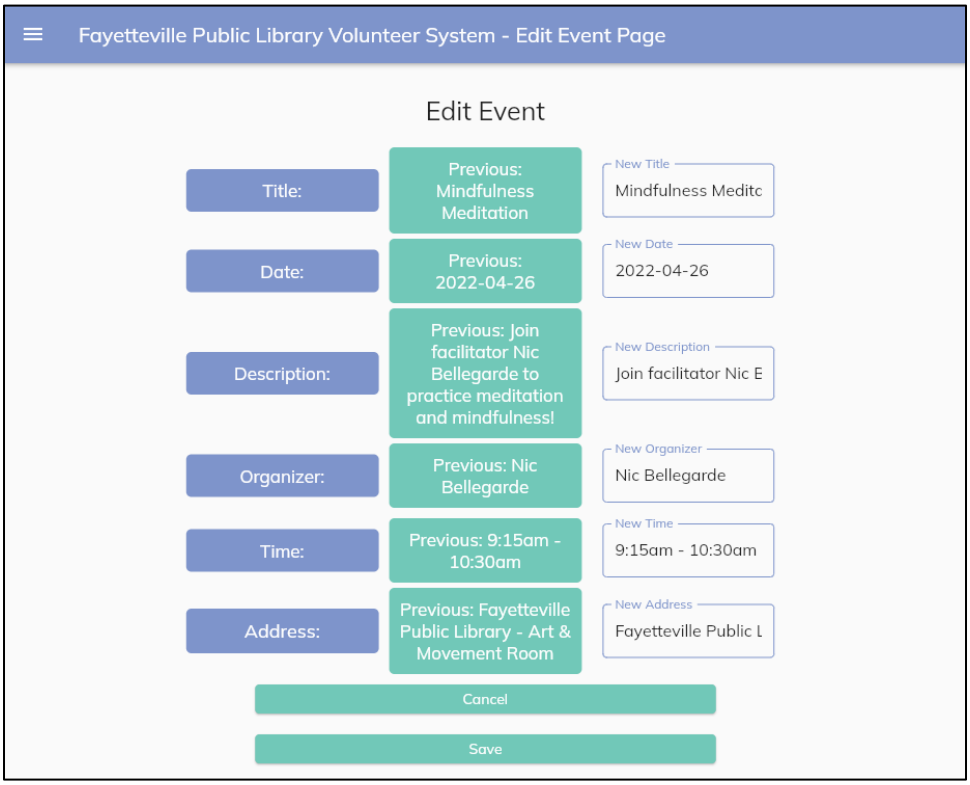


Figure 15: Admin edit event page

If the admin wishes to view a list of current users and their relevant data, they will find this information on the admin view users' page. This page displays all information relevant to each user that currently has an account on the system in a data table format. This information can be sorted using any column by simply pressing on the column title. Values for each attribute except for is active can be manually edited as well by selecting the edit icon on the bottom left of the page.

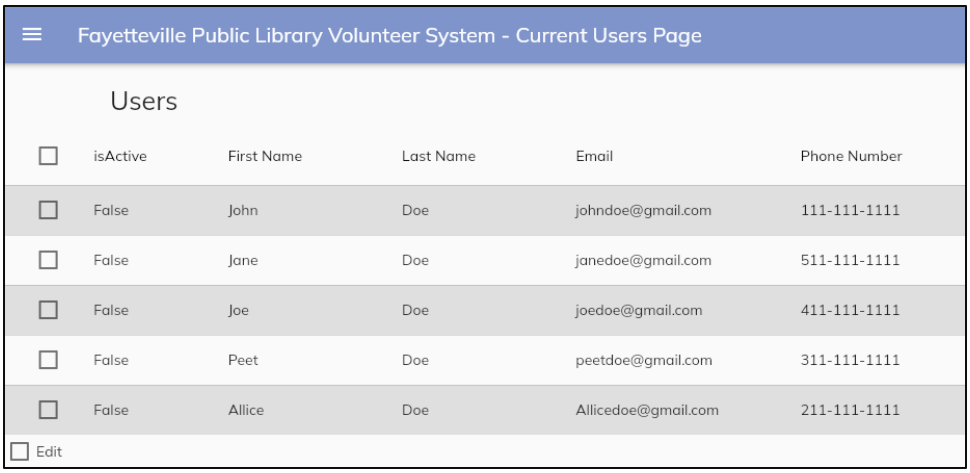


Figure 16: Admin view users page

Viewing admin accounts functions identity as the admin view user page, this functionality lies within the admin view admins page. The difference in this page and view users page is the accounts listed are accounts that only contain admin status on the system.

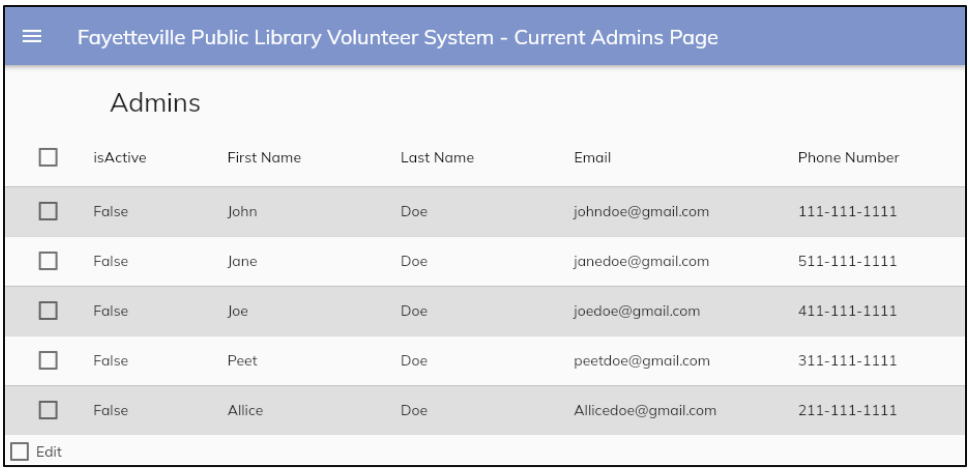


Figure 17: Admin view admins page

Administrators looking to gain information regarding events can access the reports page to gain insight on user interaction with events. Users can select from events and request information pertaining to user participation.

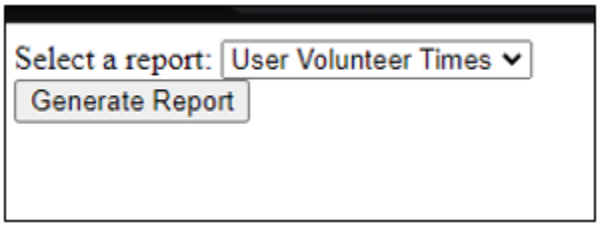


Figure 18: Admin reports page

The admin account page functions identically to the volunteer account page. This page displays all currently available information the database stores regarding the user. From this

page, the user can access a user settings page through the “Edit account details” button to manage and modify their relevant volunteer information as well.

Figure 19: Admin account page

Once an admin navigates to their account settings page, they will see several forms available to them to update or change the personnel information associated with their account. The first of these forms starts the process of changing the user’s password. To change their password, an admin must enter their old password and then their desired new password. To ensure complete confidence in the new password, the admin must re-enter the new password. Beneath that, the admin can update or change their email address. Finally, the admin can change their phone number if needed. In each instance, the user must click the respective button beneath the updated information to successfully update their settings.

The admin account model is equivalent to the volunteer model except for the volunteer/admin status. For the admin to change their settings, the backend goes through a similar process. Using the user model to convert between Json forms, a remote service function sends the data to the backend server.

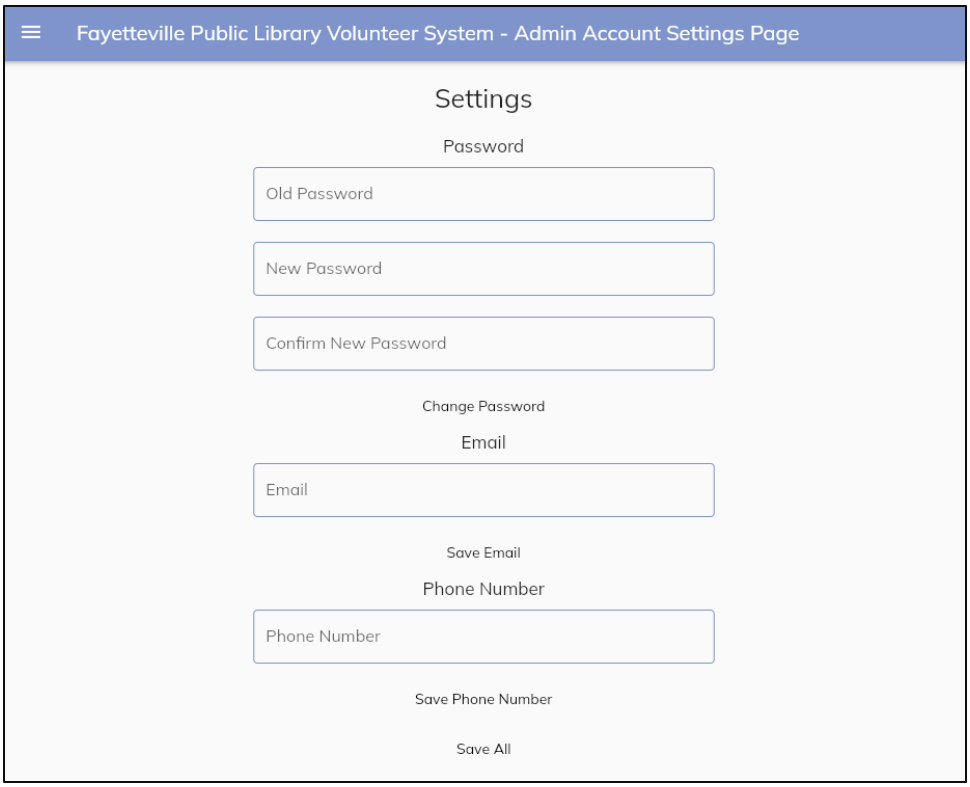


Figure 20: Admin account settings page

The user navigation sidebar includes shortcuts to the homepage, Volunteer Opportunities, Currently Enrolled, and Volunteer History pages. It also has links to view the user's account, account settings, and an option to logout from their session.

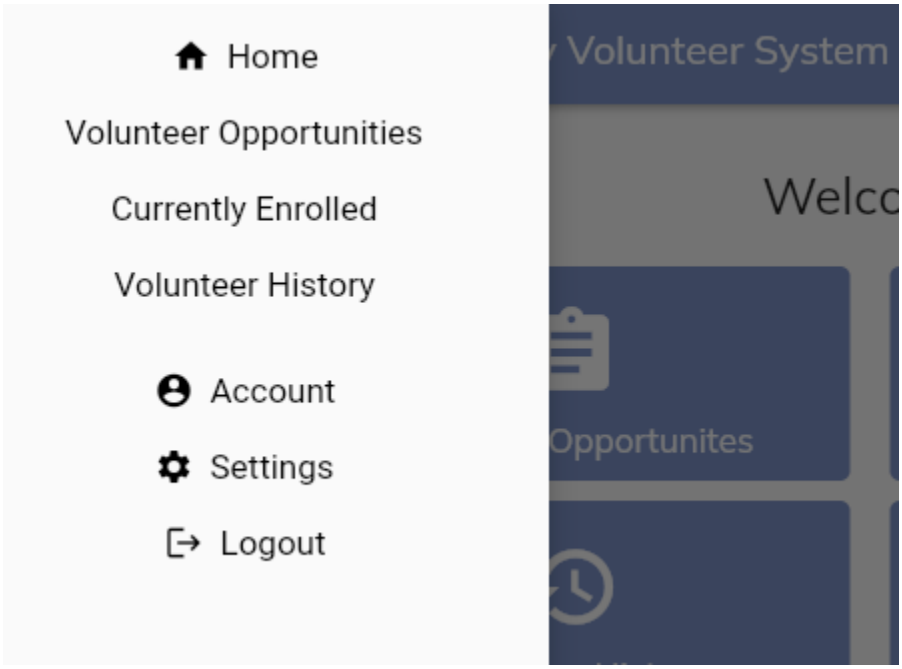


Figure 21: User navigation

The admin navigation sidebar includes shortcuts to the homepage, events listing, user database, admin database, and Reports pages. It also has links to view the admin's account, account settings, and an option to logout from their session.

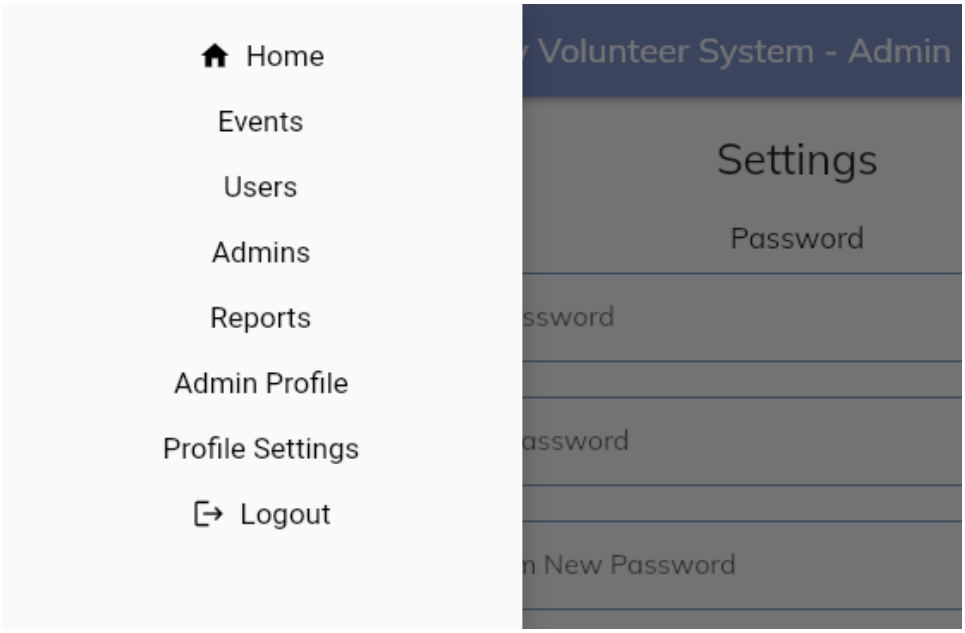


Figure 22: Admin navigation

During the completion of this project our team has learned many valuable skills in application development as well as facing some challenges along the way. Throughout the creation of this project our team has gained newfound experiences and knowledge in various programming languages and libraries never used in our programming careers. Due to our inexperience in many of the languages and libraires used in the creation of this application, there were many points of development that were inefficient. Early in the project we struggled to get a working codebase that contained basic forms of all individual components working together. This problem could have been avoided by focusing on the connection of these components rather than their solo development. Ultimately though, a working protype was made containing most of the functionality found within our team's proposal. Future works for this project would be the inclusion of functionality missing from the proposed design such as geofencing. We hope that this project can positively impact the public library along with their volunteer needs and can then be used in similar communities looking for a volunteer management system.

4.3 Risks

Risk	Risk Reduction
Breach of Sensitive Information	- Hash user passwords and use secure programming practices
Unreliable Scalability as New User Accounts are Added	- Implement with Django, using built-in scalability features and verify all new accounts that aren't previously in the database.

Security Issues Revolving Around Volunteers with Unintended Access and Permissions	- Create administrator and volunteer roles to establish separate permissions for each - Ensure administrator role has control over volunteer data
Bad Database Design for Storing User Info	- Provide documentation for database - Normalize data - Choose proper primary keys - Use good naming conventions for attributes
Volunteers Incorrectly Log Time, Purposefully or Accidentally	- Allow administrators to edit volunteer hours that users have manually entered to prevent time logging from being used improperly.
Former Admin with Valid Access and Administrator Privileges	- Allow users' roles to be promoted or demoted from active status after account has been created - Change the user's permissions accordingly.
Problems with Maintaining Code Across Multiple Workspaces	- Provide documentation so that team members and future developers can understand the code. - Use GitHub for collaboration and version control. - Variables and functions should be named well, and there should be comments to explain pieces of code - Use interfaces and abstract classes to secure data

4.4 Tasks

1) Visit Fayetteville Public Library on-site to obtain a comprehensive understanding of how volunteers interact with the different aspects of the library and what is needed to provide a better experience for those volunteers.
2) Gain an understanding of how the library organizes the system currently in use.
3) Design and discuss the high-level view and expected features of the application with the sponsors from the library.
4) Research and learn the capabilities of Flutter by beginning to build applications with the language.
5) Create Heroku project and team.
6) Connect Heroku project to GitHub repository.
7) Connect the Heroku database using pgAdmin 4.
8) Begin the implementation of the front end:
    - Create login page.
    - Create new user creation page.
    - Create user account page.
    - Create user settings page.
    - Create volunteer opportunity registration page.

- Create volunteer opportunity page that displays events the user is currently scheduled to attend.
- Create volunteer opportunity page that displays events the user has already attended.
- Create clock in/out pages for user.
- Create admin area where admins can create, edit and remove volunteer opportunities.

9) Begin the implementation of the back end:
   - Create login activity table.
   - Create account termination table.
   - Create user time info table.
   - Create user info table.
   - Create user login table.
   - Create event table.
   - Create models for each aforementioned table.
   - Create the ability to get time info for a user.
   - Create the ability to add time info to time info table.
   - Create functionality for creating a user.
   - Create functionality to modify user information
   - Create functionality for users to enroll/unenroll in desired events.
10) Align the existing volunteer database with the new back-end implementation.
11) Finalize implementation of the front-end:
    a) Finalize administrative areas and functionalities (create, modify, and delete volunteer information and opportunities)
    b) Finalize user areas and functionalities (create new profiles, edit user information, register for events, and delete user account).
12) Test the functionality of the back-end:
    a) Ensure insertions, deletions, and updates to database can occur.
    b) Review anomalies for above functions do not occur
13) Link the front-end and back-end together:
    a) Connect events within the application to its respective back-end function:
        i) Handle login event for user and admin respectively.
        ii) Handle user creation.
        iii) Handle account details retrieval.
        iv) Handle user account deletion.
        v) Handle user registration for volunteer opportunities.
        vi) Handle retrieval of user enrolled opportunities.
        vii) Handle retrieval of user completed opportunities.
        viii) Handle admin creation of volunteer opportunities.
        ix) Handle admin modifications of volunteer events.
        x) Handle admin deletion of events.
14) Test the system in its entirety:
    a) Test cases such as:

i) User tries to create an account with a username that already exists. Does the application prevent it from happening or does it overwrite the first account created?
ii) User tries to create an account and then tries to log in to another account. Does the application check that both fields are valid before creating account?
iii) User logs out of one account and then logs into another account. Does everything work as expected with no transfer of data?
iv) User exits the app without closing it then re-enters. Does the app resume normally?
v) User is signed in, but then they close their app completely. Will they still be signed in when they reopen the app?
vi) Does the UI scale correctly on phones of different sizes and on different operating systems (such as iOS/Android)?
vii) What happens if a user clocks in to an event and never clocks out? Will the timer run indefinitely or stop once a time limit has been reached?

15) Align aesthetic design of front the library's theme:
    a) Update colors.
    b) Update fonts.
    c) Update information displays (such as date displays).
16) Final round of testing.
17) Polish product before delivery.
18) Documentation.

4.5 Schedule

Tasks	Dates
1. Visit the library to understand the process needed to be currently taken by volunteers.	11/01-11/05
2. Understand how the library currently organizes its systems	11/08-11/12
3. Discuss and design the high-level view and expected features of the application with sponsors	11/15-11/19
4. Learn Flutter and Material Design and how to build applications with it. • Create a simple flutter web page program • Create GitHub repository of basic flutter application • Incorporate Material Design by google	1/18-1/24
5. Front end implementation • Create login page • Create volunteer and admin areas • Begin to build user functionality such as inputting personal information and availability, as well as volunteer time tracking forms	1/24-3/31

• Created separate views for users and admins – admins have an interface to edit event information, view users and their info, separate home screen, and buttons that allow them to create a new event • Create admin functions such as adding and removing volunteers and hours for individuals. • Create the ability for admins to add and remove events • Begin to implement volunteer registration for events	
6. Back-end implementation • Create functions that pull and store information in the database • Build connectors between the front end and back end • Build functions that are ready to receive information from the front end	1/24-3/31
7. Link front end, back end, and database • Create tables for login activity, account termination, users, user personal info, events, and volunteer history information. • Create functions within Flutter to add, retrieve, and modify information from our PostgreSQL database, convert it into JSON, and use that information to populate our pages.	3/1-3/31
8. Align existing library volunteer database with back-end implementation • Build authentication for users • Reformat webpages to include user authentication • Set up post and get interactions with front and back end	4/1-4/7
9. Extensively test all aspects to ensure proper functions • Validate admins can create and post events that the user can view • Validate the clock in clock out functionality gets stored to backend	4/7-4/10
10. Port to mobile • Flutter code can run on Android and iOS with the same codebase used for the web application • Address all UI bugs and account for varying display sizes	4/10-4/17
11. Extensively test all aspects of mobile port • Ran the app on Android emulator to test its functionality and appearance on a smaller screen • Error handling (null inputs)	4/17-4/24
12. Final polishing and testing • Comment pages extensively • Handle any bugs that have gone unaddressed	4/24-4/28

13. Documentation	4/28-5/04

4.6 Deliverables – Give a thorough listing and description of each item which will be submitted with your final, working project. Each major component should be described. The below is just an example list which should be replaced with your own.

- Design Document: Contains a listing of each major software component as well as documentation on how to operate, edit and implement each component.
- Database scheme: The Ubuntu PSQL code for our database implementation
- Web Application: The front-end volunteer management system that will allow for user and admins to interact with the system.
- Application Code: This code base will contain Python, Django, Flutter, and Material Design code used for the application itself. This code will also include appropriate and expected commenting and documentation.
- Final Report

5.0 Key Personnel

Alexis Carter – Carter is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses such as Database Management, Information Retrieval, Information Security and Software Engineering. She has used Python, Java, JavaScript, JFlex, and SQL. She has built a database from the backend and frontend and used SQL to manipulate databases. She has also designed several simple multi-page websites with corresponding the interfaces.

Kagen Crouch – Crouch is a senior Bachelor of Science of Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as software engineering and programming paradigms. Crouch has gained extensive knowledge in Java, JavaScript, and Python programming languages through classes at the university. Crouch also contains knowledge of Node.JS, HTML, and CSS. This knowledge was obtained through a software engineering class taken at the university. Crouch will be responsible for the creation and implementation of the front end of our web application and the various mobile applications that will be created.

Bradley Lithalangsy – Lithalangsy is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Database Management and Software Engineering, and he is currently enrolled in Mobile Programming. Lithalangsy is knowledgeable in VB.NET, Java, JavaScript, Python, and SQL. He has experience creating a pricing website through his internship using VB.NET and JavaScript. Lithalangsy will be responsible for the cross platform mobile application development in Flutter and helping with the front-end development of the web application.

Nathan Secrest – Secrest is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses such as Database Management Systems, Software Engineering, Programming

Paradigms, and more. He is currently taking Information Retrieval. Secrest is experienced in C++, Java, MySQL, and JavaScript. He has also had an internship at EngwIT, a software development company located in Gravette, AR. Secrest will be responsible for backend development.

Steven Trinh – Trinh is a senior Computer Science major at the University of Arkansas. He has completed relevant courses including Database Management Systems, Software Engineering, and Programming Paradigms. He is currently taking Mobile Programming which involves Android development programming projects. He is knowledgeable in C++, Java, JavaScript, Python, and SQL. Trinh will be responsible for front-end development that includes storing/retrieving user information from the database when the user interacts with the application and getting the UI to display information whenever a user performs an action correctly and responsively.

Carlye Dennis, Industry Champion – Dennis is the current Community Engagement Manager at the Fayetteville public library. Dennis has received her Bachelor of English at the University of Arkansas in 2004, while also receiving her master's in Library Science and Administration at the University of North Texas in 2008. Carlye has performed several tasks for the public library such as, developmental assistant and manager of volunteer and outreach.

Chris Moody, Industry Champion – Moody is the current Directory of IT/AV at the Fayetteville Public Library. Moody received his Bachelor of Education at the University of Arkansas. Moody also worked under the university for 13 years performing several tasks one of which being IT administrator for 4 years.

Kent Watson, Industry Champion – Watson has worked under several roofs over the years. Some of these companies include Tyson Foods, University of Arkansas, RockFish and Metova Inc. While at these companies Watson has served as several positions from PC support all the way to executive technology strategy. Watson has a Bachelor of Computer Science and a Master of Computer Science from the University of Arkansas.

Matthew Patitz, Professor – Patitz received his Bachelor of Science, Master of Science and Doctorate's Degree in Computer Science from Iowa State University. Patitz worked at the start cup software company SupportSoft as a software engineer and team lead from 2000 to 2005. Patitz has been in the Department of Computer Science and Computer Engineering at the University of Arkansas. He is currently an associate professor. His research interests are DNA computing, algorithmic self-assembly, and theoretical computer science in general.

6.0 Facilities and Equipment

No facilities will be needed for the completion of the project. The only equipment that will be required is various mobile devices, but these can be emulated from individual group members' personnel workstations.

7.0 References

[1] Group, PostgreSQL Global Development. *PostgreSQL*, 25 Apr. 2022, https://www.postgresql.org/.

[2] Django, https://www.djangoproject.com/foundation/

[3] Pyscopg2, https://pypi.org/project/psycopg2/

[4] Flutter, https://flutter.dev/docs

[5] Material Design, https://material.io/

[6] pgAdmin, https://www.pgadmin.org/

[7] GivePulse, https://www.givepulse.com/