Chi Alpha: Christian Fellowship Full-stack Mobile Application

Alpha Prototype Report

Chi Alpha WSU

Web Ready

Vincent Yen, Justin Lee

I. Introduction

The intent of this project is to create a resource in the form of a mobile application for Chi Alpha WSU. The mobile app will allow the organization to make announcements, advertise events, a medium for individuals to make donations, and communicate with the organization. The mobile app is intended for the organization to increase their presence and access to and for current members of the fellowship. This document serves as an overview of the intended work for this project.

II. Team Members – Bios and Related Roles

Name: Vincent Yen

Degree Plan: Computer Science B.S.

Project Role: Team Lead, Software Architect

Skills and Areas of Expertise:

- Computer Languages: C/C++, Java, Python, JavaScript/TypeScript, HTML, CSS
- Database: SQL, MongoDB
- Web Tools: Angular, React, Flask, NodeJS, Django, Karma, Jasmine

Description:

Vincent is a computer science student that has experience working with full-stack web applications which carry into this project since mobile applications follow a similar framework. His responsibilities for this team is to: frame the project so that it meets the needs of the client as well as the abilities of the team, work documentation, back-end, database, and aid in whatever areas of the project that cannot be fulfilled by other team members.

Name: Justin Lee

Degree: Computer Science B.S.

Project Role: UI Developer

Skills and Areas of Expertise:

- Computer Languages: C/C++, Java, Python, HTML, CSS
- Database: SQL
- Web Tools: Flask

Description:

Justin is a computer science student that has some prior experience building web applications using Python Flask. Justin sought to work on this project because of his interest in working on web applications. His role in this project is primarily to work on the user interfacing components of the application.

III. Background and Related Work

Chi Alpha WSU currently does not have a mobile resource application that they can use to communicate with the fellowship. The client has requested that this application provide an easier way for the web administrator to advertise activities that are happening within the organization.

Currently, the native development environment for mobile apps uses XCode to code in the Swift language for iOS applications for Apple mobile devices and Java to code applications for Android mobile devices [1, 2, 6, 9, 10, 15]. React-native combines the flexibility and ease of development by using the react framework in conjunction with web elements such as JavaScript and html to code in an environment that would allow any developer to create mobile apps for both iOS and Android devices without having to code in the native environment, especially considering XCode and Swift is only available for coding on mac devices [1, 2, 3, 6, 7, 9].

IV. Project Overview

III.1 Use Cases

Story: Hailey would like to add an announcement to the resource application. Hailey opens the app and logs into the application using an administrator login. The application now shows the ability to add new templated elements to the app, specifically a form to add a new announcement to the application dashboard. Hailey enters the new announcement information into the form and submits for publication. The new announcement is reflected on the announcement dashboard of the application.

Source: Vincent Yen

Story: A member of the fellowship would like to donate to the organization. The member selects “Giving” from the menu and is taken to the “Giving” page. The member enters their payment information into the third-party payment API and submits [4]. The organization receives the payment that was given by the member.

Source: Vincent Yen

III.2. Functional Requirements

- The application should allow adding and displaying announcements from the organization.
- The application should allow adding and displaying events for signing up and viewing.
- The application should allow a medium for donations from any individual who wishes to give.
- The application should show information about the organization and its staff members.
- The application should allow the user to login.
- The application should allow other users to communicate with each other that are using the application.

- An API should provide a medium between the application and the database which stores user information and media [4].
- The application should differentiate general users from admin users and limit certain access permissions to only the admin.

III.3 Non-Functional Requirements

- The mobile application should adhere to federal standards of accessibility.
- User experience should be intuitive and easy to use.
- The application should be intuitive for both general users and the administrators
- The application should be easily maintained by administrators who have little to no technical experience with mobile applications.

V. Client and Stakeholder Identification and Preferences

Client: Hailey Galletly

Summary: Hailey Galletly is the principal client in this project as she is the current administrator for all media related concerns for Chi Alpha WSU.

Needs/Preferences:

- Seeking a mobile application that allows the organization to advertise and communicate to members of the fellowship
- Flexibility and ease of use for general users and admin
- Avenue for chat communication with general users

Stakeholders: fellowship members, Chi Alpha WSU, media administrators

Summary:

- Fellowship members are anyone that is involved with the organization, but not working in the organization.
- Media administrators are individuals that have been given the responsibility of the public relations and media related to the organization
- Chi Alpha WSU has particular interest in this application because the organization is being represented in the mobile application

Needs/Preferences:

- Application should be intuitive to any individual that uses it
- Any future media administrator should be able to manage the content intuitively without technical experience
- Application should accurately reflect WSU Chi Alpha as an organization and its needs

VI. System Requirements Specification

V.1 Use Cases

Story: Hailey would like to add an announcement to the resource application. Hailey opens the app and logs into the application using an administrator login. The application now shows the ability to add new templated elements to the app, specifically a form to add a new announcement to the application dashboard. Hailey enters the new announcement information into the form and submits for publication. The new announcement is reflected on the announcement dashboard of the application.

Source: Vincent Yen

Story: A member of the fellowship would like to donate to the organization. The member selects “Giving” from the menu and is taken to the “Giving” page. The member enters their payment information into the third-party payment API and submits [4]. The organization receives the payment that was given by the member.

Source: Vincent Yen

Story: A member of the fellowship would like to sign up for an ongoing fall retreat event with the organization. The member selects “Events” to view the event page and is able to view the different ongoing events. The member selects the fall retreat event from the event listings. The page displays the form for signing up for the fall retreat. The member enters and submits the pertinent information for the form. The organization receives the form with the needed information from the member and is aware that the member’s interest in the event.

Source: Vincent Yen

V.2. Functional Requirements

V.2.1 Announcement Display

The application should allow adding and displaying announcements from the organization.
Source: Hailey Galletly. **Priority:** 0

V.2.2 Event Display

The application should allow adding and displaying events for signing up and viewing.
Source: Hailey Galletly. **Priority:** 0

V.2.3 Donation Medium

The application should allow a medium for donations from any individual who wishes to give.
Source: Hailey Galletly. **Priority:** 0

V.2.4 Organizational Information

The application should show information about the organization and its staff members. **Source:** Hailey Galletly**. Priority:** 0

V.2.5 User Login

The application should allow the user to login. **Source:** Hailey Galletly. **Priority:** 1

V.2.6 Peer-to-peer communication

The application should allow other users to communicate with each other that are using the application. **Source:** Hailey Galletly. **Priority:** 1

V.2.7 Database and API middleware

An API should provide a medium between the application and the database which stores user information and media [4]. **Source:** Vincent Yen. **Priority**: 1

V.2.8 User Permissions

The application should differentiate general users from admin users and limit certain access permissions to only the admin. **Source:** Vincent Yen. **Priority:** 1

V.3 Non-Functional Requirements

V.3.1 Standards

The mobile application should adhere to federal standards of accessibility. **Source**: Project Advisor. **Priority:** 1

V.3.2 User Experience

The application should be intuitive for both general users and the administrators. **Source:** Hailey Galletly. **Priority:** 1

V.3.3 Application Flexibility

The application should be easily maintained by administrators who have little to no technical experience with mobile applications. **Source**: Hailey Galletly. **Priority:** 1

V.3.4 Screen Size Scaling

All elements that are meant to be displayed on the screen should be visible on any mobile device. **Source:** Vincent Yen. **Priority: 1**

V.3.5 Processing Performance

Increasing amounts of content in the application should not hinder the performance of the loading and runtimes of the application. **Source:** Vincent Yen. **Priority:** 1

V.3.6 Network Performance

Increasing traffic on the application should not hinder the performance of the application. **Source:** Vincent Yen. **Priority:** 2

VII. System Evolution

The nature of this project is such that once it is built it should not have to be updated since the design requires that it allows for the media administrator(s) to add content without the need for a technical mobile developer to make updates to the code base. Given this nature, the mobile application provided for the media administrator(s) must be intuitive for any general user and should anticipate as many user experience issues that these stakeholders might experience with this product. The major risk of this project is that the platform runs into operational errors or new requirements surface when using the final product that was not revealed in the building process that requires updates from a technical mobile developer.

Furthermore, it is possible that if the APIs integrated with the application undergoes an update that is incompatible with the product it may render the website or simply the APIs unusable [4]. These updates may also occur during the development process which may render the current progress of the development null, requiring an overhaul of the work already done.

VIII. System Overview

The project is intended for the organization to increase their presence and access to and for current members of the fellowship. The mobile app will allow the organization to make announcements, advertise events, serve as a medium for individuals to make donations, and communicate with the organization.

IX. Architecture Design

a. Overview

This project follows most closely with a MVC model- model-view-controller, of system architecture. The application closely resembles most website MVC models in that it serves content to pages for viewing, information about users and media are modeled for the database schema, and a controller service controls routes to the database and authentication service based on various requests from the user.

The diagram given by Appendix A figure 1 serves to show how the system is decomposed. The system proposed for the project is decomposed into 4 major parts: pages, user menu, server API, and database [4].

Each page is composed with an application page that a user can view to obtain various content provided by the content creator. The announcement page allows the content creator to serve announcements that the organization would like the fellowship community to be aware of. The events page allows the content creator to publish events that are happening in the community that users can sign up to attend. The giving page allows members of the community to donate to the organization through a third-party payment API service [4]. The contact us page allows users to contact the organization via email service. The about us page gives information about the organization and its staff members. The media page allows the content creator to publish different forms of media such as videos or pictures for users to browse and consume.

The user menu and its subcomponents are necessary specifically for all components that require user differentiation during their session. Currently the only component that requires a user login is the chatroom so that each user can be distinguished from other others in a simultaneous view session.

The server API serves as middleware between the mobile application and the database to aid in serving different media stored in the database as well as user authentication [4, 10].

The diagram in Appendix B figure 1 serves to show the different dependencies and interactions between each package in the system. The diagram given in Appendix B figure 2 serves as an overview of how the application will interact with the various package components.

The pages require the user interface to serve information for viewing. The user may interact with various pages for donations or to submit forms for things like event sign ups. As such the forms require pages and an API server to serve the form for viewing and to save the information into the database for the content creator to view later [4]. Donations also require pages as a way for users to interact with and submit their payments. The donations also require a payment service which will be provided by a third-part payment API [4].

The chat package requires the user interface in order for other users to interact with and submit their messages. The user may or may not login in order to identify themselves or remain anonymous.

The server serves as the middleware between the forms and pages as well as the user authentication service [10]. Forms being submit from a page will be sent to the server for validation and saving into the database. Users logging into the application will need to be authenticated with the user authentication service which will be validated against the database using the server.

The database package serves to save all information about users or store different forms of media based on the models provided in the server.

b. Subsystem Decomposition

VIII.2.1 User Interface

a) Description

The user interface is as its name implies. It serves as a graphical interface for the user to interact with the various components within the application.

b) Concepts and Algorithms Generated

The user interface is built using the React-Native framework which uses components derived from ReactJS [3, 11]. This framework allows the mobile application to be developed for both of the popular mobile operating systems- iOS and Android, simultaneously without having to work specifically with the native code for those operating systems [1, 6].

The drawback for using React-Native is having to use the specific syntax used in that framework as well as not being able to perform more specific functions that are only available in the native language for those operating systems [3].

VIII.2.2 Pages

a) Description

The pages serve as the view component for users to view different content provided by a content creator.

b) Concepts and Algorithms Generated

Since each page is likely not change in its form each page in the page component will hard coded based on the requirements provided by the client. A basic template will allow serving of various content from the database in the media page, announcement page, or the events page.

A drawback of this would be that it would limit the flexibility of the content creator to form the view of the given pages if in the future it is decided that the view of the page is not to the liking of the content provider.

c) Interface Description

Services Required

1. Service Name: RESTful service [8]
   Service Provided to: Server
   Description: The server will provide all content that is requested from each page that is consumed by the user.

III.2.2 Forms

a) Description

Each form will provide various ways for users to submit information that the content creator is asking for.

b) Concepts and Algorithms Generated

Each form will be created from a template which should allow some flexibility for the content creator to enter their own specific information or questions for user submission.

If the content creator requires a more specific model of how the forms should be represented it may require more tailored forms that would deviate from templating and more towards hardcoding.

c) Interface Description

Services Provided

1. Service Name: Form template
   Service Provided to: Forms
   Description: Provides a template for all forms to be used for the content creator.

VIII.2.3 Chat

a) Description

The chat subsystem allows various users to chat with other users in the app.

b) Concepts and Algorithms Generated

The chat subsystem will be a socket interface using session tokens for each user that logs into the chat system during their view session. A user that spends a certain amount of time idle in the chat will be removed from the system in order to alleviate resource usage. Users may login with prior created login credentials to identify themselves in the chat or enter chat without logging in as an anonymous guest.

A particular draw-back of allowing anonymous guests is the possibility of traffic flooding or a form of DDoS attack. Furthermore, by remaining anonymous various individuals may be able to act in other malicious ways in the chat service without repercussions because the individual cannot be easily traced and stopped.

c) Interface Description

Services Required

1. Service Name: Server
   Service Provided to: Chat
   Description: The chat system requires user information for each session user in the chat. The server also serves as the middleware for all information exchanged in the chat so that a single user may broadcast to other session users or receive information that was broadcast from other users.

VIII.2.4 Server

a) Description

The server serves as a middleware between the database and the resource application. The server will route various requests to various functions in the server such as user authentication, saving information submitted from forms into the database, or serving various media requests from the Media page.

b) Concepts and Algorithms Generated

Given that the server is middleware between the database and the mobile application, this system serves as the controller component in the MVC architecture. The server will contain routes for which various requests will be routed to functions.

The database schema will be built based on models provided in the server. Any forms generated for user submission will be built from models provided in this server as well.

A draw-back of this design is that if changes need to be made in database schema it may be difficult to change without affecting the information already being stored in database. Possible ways to overcome this is to make a new schema model to meet the new needs of the content creator while maintaining the old model for previously store information.

c) Interface Description

Services Provided

1. Service Name: User Authentication
   Service Provided to: Chat
   Description: The user authentication service verifies the credentials of a user that is logging into the system. This allows the user to be identified distinctly in the chat system.

2. Service Name: Database Model
   Service Provided to: Database
   Description: The database model provides the database with the template needed to form the database schemas in which each storage item will be follow.

3. Service Name: Routes
   Service Provided to: Pages/Forms
   Description: Any page that requires particular content will require requests from the server which the server will determine which function it should be routed to based on the type of request. Requests may include: get, put, and post.

Services Required

1. Service Name: Object Storage
   Service Provided to: Database
   Description: The server requires storage for all the information that it receives from the front-end.

VIII.2.5 Database

a) Description

The database serves a single point of storage for all information that is needed for serving various content or saving content. Some of this information includes: media storage, user credentials and identification, organization and staff bios, and information submitted from forms.

b) Concepts and Algorithms Generated

The chosen database structure will be MongoDB which will be paired with the Mongoose JavaScript library. MongoDB uses a non-relational type of storage which provides more flexibility and quicker data retrieval when compared to SQL type databases. Currently there are no forms of data being saved that requires indexing therefore this lends itself well to this form storage.

c) Interface Description

Services Provided

1. Service Name: Data Storage
   Service Provided to: Server
   Description: Data storage allows the server to obtain pertinent information that was previously saved for various purposes such as user validation, form data storage, and media storage.

Services Required

1. Service Name: Models
   Service Provided to: Server
   Description: The database requires models that are specified by the server in order to determine how it should form its schema templates.

X. Data design

The database will be using a non-relation type of database called MongoDB [5]. MongoDB represents data in the form of JSON as opposed to tables as seen in SQL type databases [5, 9, 5]. This form of database does not enforce the use of schema's and gives it more flexibility in how objects can be stored. Furthermore, this flexibility improves the performance read/write as it does not require information to be inserted into a rigid table schema [5].

XI. User Interface Design

The picture provided in Appendix C figure 1 serves as a general image of how the graphical user interface should appear. A menu with clear descriptions serves as a means for the user to switch between the different page views for consumption. Future renditions should include a user menu, not shown in the figure, which provides options to login and perform various functions that require user credentials such as the chat system.

User should be able to obtain the application via the Google Play store for Android devices or the Apple App store for iOS devices [1, 6, 15]. In order to obtain the application from these stores the user must already have an account or otherwise register to create an account with the relative store in order to gain access to the store which allows the user to download the application onto their device.

XII. Testing Overview, Objectives and Scheduling

XI.1. Test Overview

The purpose of building the test requirements for this application is to aid in building a meaningful and comfortable user experience to both the administrator that will be creating content for the application as well as the general user who will be consuming the application resources. Therefore, the testing for this application must be able to go through an end-to-end test of different use scenarios which a user and administrator will likely encounter through a session of use.

XI.2. Objectives

The objectives for this application's testing are outlined as follows:

- Each unit test will be representative of a use case for that unit.
- Each end-to-end test will be representative of a start to finish use case of a session of use.
- Each unit test shall account for reasonable edge cases that are representative of a user interaction with the app component.

Required resources:

- A test framework to test each unit.
- A test framework for integration testing and end-to-end testing
- Virtual environment to test on the supposed platform which the application will be deployed.

Overview of Scheduling:

Each project issue will make up a component of the entire application. Prior to completing each issue, each issue should be at the very least manually tested prior to committing to the main source code.

By the sprint 2 deadline, several page components and the routing component should be completed and manually tested for functionality.

By the sprint 3 deadline, the database API middleware should be completed and the majority of the pages if not all page components should be completed [4, 10]. At this point a basic end-to-end test should be viable and also completed prior to this deadline. Furthermore, automated unit tests should be also completed at this point.

XI.3. Scope

The purpose of this testing documentation is to give a general understanding and overview of the testing that will be committed to this application as well as some of the tools that will be applied for the sake of that testing.

XIII. Testing Strategy

The testing strategy will be following a continuous integration model because it lends itself well to the type of application that is being built as well as the timeframe that is given to work on this project. The strategy will be applied to every unit as follows:

1) Implementation: Writing the code for the specific component.

2) Write the tests: Tests should be written based on the requirements set by the team and the more specifically the client.

3) Test code: The component should be tested based on the written tests or requirements that have been set forth.

4) Document: Regardless of the outcome of the test, there should be documentation of when the code was tested and whether or not the test passed or failed. If it failed, there should be indication of where in the test it failed or note as to why it may have failed.

5) Push to source/code review: If the code is tested satisfactorily the code should be pushed into the source code for integration. The code should be reviewed by another team member to check for issues that may be missed.

6) Integration testing: Once all features for a given sprint have been integrated into the source code, integration tests can be written and should be tested. Based on the results of the integration tests components of the code may be removed for further unit work or tweaked as a whole based on the needs of the application.

XIV. Test Plans

XIII.1. Unit Testing

Unit testing for the application will be conducted using a test framework called Jest. Jest allows for a testing methodology called snapshot testing which allows an image of a UI component to be rendered and a static “snapshot” to be taken of that rendering. The snapshot is used as a reference to compare against another snapshot that will be taken to see if they match. This helps to identify if the unit is operating consistently and correctly against an intended output. This portion of the testing is primarily for testing the UI portion of the application only.

The other form of unit testing that will take place will be through the use of an API platform called Postman. Postman allows interaction with the server component of the application without having to integrate with a UI. This will allow for manually testing different forms of RESTful requests to the server to check for proper responses [8].

XIII.2. Integration Testing

The integration testing involved in this process will continue to make use of the Jest testing framework. By testing including information pulled from the database and how it will be represented or rendered in the UI component this will require more substantial use of the snapshot testing methodology provided by Jest.

XIII.3. System Testing

For end-to-end testing, the most popular platform used for React-native applications is Detox because it is specifically designed for React-native. This will allow for designing an automated test that is representative of the application lifecycle through a user's use session [3].

XIII.4 Functional Testing

The functional testing for this application will also be through the use of the Jest framework. As described previously in unit testing, the intended output of rendering of a UI component can be observed with the snapshot tests and the intended input and output of the middleware API and database can be tested manually using Postman.

XIII.5. Performance Testing

There isn't much involvement for the sake of performance testing for the application especially since the performance is relative to the application and the user's device and does not primarily involve user traffic. However, one component that may hinder performance may be traffic from database requests through the middleware API. This would require virtualizing multiple users making simultaneous requests to the middleware; however, this performance test may be considered excessive considering that the application is meant to service a relatively small audience.

XIII.5. User Acceptance Testing

A portion of the acceptance testing will be demonstrating the application to the client to see that it meets their needs. Furthermore, some individuals may be asked to sample the application in order to determine whether there has been oversight on user interaction or that all requirements have been met to provide proper user experience.

XV. Test Environment Requirements

Unit tests and integration tests will be done through the test framework Jest while end-to-end tests will be done using the Detox framework. Preliminary testing will be done through a virtual Android platform delivered via Android Studio [1]. Testing solely through Android initially is because of hardware limitations as the development team uses a pc platform and testing for a iOS platform would require a mac platform to virtualize a iOS device [6]. Once the majority of the basic features have been developed and tested on the virtualized Android platform the prototype should then also be tested on a virtualized iOS platform by testing remotely using a macOS server to host our testing needs [1, 6].

Finally, in final acceptance testing and demonstration the application shall be run on at least two different mobile devices, one containing the Android OS and the other containing the iOS [1, 6]. This will be tested against the requirements of the client and sampling of users to determine the value of the user experience that the application provides.

XVI. Test Cases

The following is a general test case overview of each of the functional and nonfunctional components. Specific details can be found in the test case document. Test case IDs reference test cases found in the test case document. Non-functional test cases will be referenced from this document for testing.

XVI.1 Functional Components

XVI.1.1 Announcement Page

Requirements ID: V.2.1 of prototype report

Test ID(s): P 1.1, P 1.2, SR 3.1 – SR 3.4, SM 3.1 – SM 3.4

Requirement(s) Achievement:

- Page gets content via http request
- Page shows administrator view if admin is logged to admin view
- Page displays content received from http response
- Page displays error if unable to receive correct content

XVI.1.2 Event Page

Requirements ID: V.2.2 of prototype report

Test ID(s): P 2.1, SR 4.1 – SR 4.4, SR 5.1, SM 4.1 – SM 4.3, SM 5.1

Requirement(s) Achievement:

- Page gets content via http request
- Page shows administrator view if admin is logged to admin view
- Page displays content received from http response
- Page displays error if unable to receive correct content

XVI.1.3 Giving Page

Requirements ID: V.2.3 of prototype report

Test ID(s): N/A

Requirement(s) Achievement:

- Page displays form for donation submission

- Form submission finalizes payment from user to organization
- Confirmation receipt of payment transfer and receipt

XVI.1.4 About Us Page

Requirements ID: V.2.4 of prototype report

Test ID(s): P 3.1

Requirement(s) Achievement:

- Page gets content via http request
- Page shows content from http response
- Page displays error if unable to receive correct content
- Page shows admin view if admin is logged to admin view

XVI.1.5 User Functions

Requirements ID: V.2.5 of prototype report

Test ID(s): SR 2.1 – SR 2.2, SM 2.1 – SM 2.2

Requirement(s) Achievement:

- Menu shows login/registration buttons if user is not logged in
- Menu shows logout button if user is logged in
- Menu shows admin view button if user is admin
- Menu shows user view button if admin is in admin view

XVI.1.6 Chat Function

Requirements ID: V.2.6 of prototype report

Test ID(s): N/A

Requirement(s) Achievement:

- Chat entry button is visible if user is logged in
- Chat messages are transmitted to all users in chat room
- Chat messages are visible to all users in chat room

XVI.1.7 Server API

Requirements ID: V.2.7 of prototype report

Test ID(s): SS 1.1

Requirement(s) Achievement:

- Server starts and establishes connection to database

- Server is listening on correct port address
- Server is able to obtain http request and return http response
- Server is able to query database based on http request
- Server returns correct http response

XVI 1.8 Permission Check

Requirements ID: V.2.8 of prototype report

Test ID(s): N/A

Requirement(s) Achievement:

- Application will run user permissions check when attempting to access administrative functions
- Application will correctly identify user permissions
- Application will grant access to administrative functions if user has correct permissions

XVI.2 Non-Functional Components

XVI.2.1 Standards

Requirements ID: V.3.1 of prototype report

Test ID(s): NFT 1

Requirements Achievement:

- Application adheres to all federal standards for accessibility

XVI.2.2 User Experience

Requirements ID: V.3.2 of prototype report

Test ID(s): NFT 2

Requirements Achievement:

- Application is perceived to be easily accessible by user base
- Application is perceived to be easily accessible by admin base

XVI.2.3 Application Flexibility

Requirements ID: V.3.3 of prototype report

Test ID(s): NFT 3

Requirements Achievement:

- Application can be used by admin without much explanation

XVI.2.4 Screen Size Scaling

Requirements ID: V.3.5 of prototype report

Test ID(s): NFT 4

Requirement(s) Achievement:

- Application view is visible by user base
- Application view is visible by admin base

XVI.2.5 Processing Performance

Requirements ID: V.3.6 of prototype report

Test ID(s): NFT 5

Requirement(s) Achievement:

- Performance of application does not hinder user experience
- Functions are optimized for optimal performance

XVI.2.6 Network Performance

Requirements ID: V.3.7 of prototype report

Test ID(s): NFT 6

Requirement(s) Achievement:

- HTTP request and response does not hinder user experience
- HTTP request and response routing is optimized for network performance

XVII. Alpha Prototype Description

A summary of the project description can be found in the diagrams shown in the appendix. A sample of the current application is described in appendix C.Fig.2 Currently the team has completed two basic page iterations and the page menu, specifically the announcement and events page, and are currently working on the remainder of the pages and the back-end component which includes the server API and database. All completed components have been integrated successfully; however, no unit tests or integration tests have yet been created.

XVI.1 Pages

XVI.1.1 Announcements Page

This page is intended to show various announcements that can be added by an administrator/content creator for other users to consume. It lists various announcements and when an announcement is selected, details about that announcement is displayed. The page is currently configured to show place holder JSON data and does not have the ability to create announcements [9]. Future iterations of this page will include changing the configuration to obtain announcements stored in the database via the server API and an administrative view of the page that allows an administrator to added announcements to the database [4].

XVI.1.1.1 Preliminary Tests

The page has been manually tested for functionality; however, unit tests and integration tests have not been written for this feature.

XVI.1.2 Events Page

This page is intended to show various events that can be added by an administrator/content creator for other users to consume. It lists various events and when an event is selected, details about that event is displayed as well as a form created by the administrator for users to sign up for the event. The page is currently configured to show place holder JSON data and does not have the ability to display or create forms [9]. Future iterations of this page will include changing the configuration to obtain events stored in the database via the server API and an administrative view of the page that allows the administrator to add events to the database [4].

XVI.1.2.1 Preliminary Tests

The page has been manually tested for functionality; however, unit tests and integration tests have not been written for this feature.

XVI.2 Menus

XVI.2.1 Page Menu

The page menu is a tab menu at the bottom of the screen that routes the user to the different pages upon selection. The page menu currently routes to all pages shown in Appendix A.fig.1. No future work is planned for this component at this moment.

XVI.2.1.1 Preliminary Tests

The menu has been manually tested for functionality; however, unit tests and integration tests have not been written for this feature.

XVIII. Alpha Prototype Demonstration

The client has viewed the current version of the application and has been thoroughly pleased with the progress made thus far. The client plans to make a logo that can be used as an asset for the application header in the future. Some logistical issues have been discussed with the client in regards to the fees that will be required to use third-party APIs that are being used in the application as well as hosts for the server and database of the application when the final iteration will be deployed. The client presentation for this alpha prototype can be found at the following URL:

https://github.com/WSUCptSCapstone-Fall2022Spring2023/cacf-fullstackapp/blob/master/Documents/CACF_ClientPresentation.pdf

XIX. Future Work

For the remainder of the project, the plan is to focus on the user interfacing components and the major back-end components of the project such as the database and the database server interface. The plan for minimum value product will be composed of the pages that the users will consume, the pages that admin will use to create content, and the database to store all the information that will be used for consumption. The hope for the project is that the team will be able to move beyond the expected plan for the minimum value product and deliver more features for a better user experience.

XX. Glossary

Android – The world's most popular mobile operating system [1].

API – Stands for application programming interface. A third-party application that may be used in conjunction with another application for its specific services [4].

Apple - "Apple is a multinational information technology company based in California, USA." The company make various electronic devices, computers, laptops, as well as software [15].

iOS – A mobile operating system for Apple mobile devices [6].

Java - A programming language and computing platform developed by Sun Microsystems in 1995 and currently owned by Oracle [10].

JavaScript - A scripting language used in web development to implement complex features and logical algorithms [7].

JSON – Stands for JavaScript Object Notation. It is used as a format for data transfer on the web [9].

Middleware – Software that allows two or more applications or application components to communicate with each other [10].

MongoDB – A non-relational database that stores information in the form of objects similar to JSON structure [5].

Mongoose – An Object Data Modeling JavaScript library used in conjunction with MongoDB and a JavaScript library called NodeJS. Mongoose allows relational database management

and schema validations while still maintaining some flexibility in the non-relational structure of MongoDB [2].

React-Native – React Native is an open-source mobile framework using JavaScript to design applications for IOS and Android. React Native is derived from ReactJS, a JavaScript framework used for web application development [3].

ReactJS – A JavaScript library which is used to build reusable UI components in Web development [11].

SQL – A relational database which stores data using tables and rows and enforces "referential integrity" [5].

Swift - A programming language for iOS and other Apple devices [9].

RESTful – REST stands for Representational State Transfer. An API that uses this architectural design is said to be RESTful. REST is used to across various forms of network protocols to transfer information [8].

Xcode - An integrated development environment created by Apple which supports the following languages: Swift, Objective-C, C++, etc [2].

XXI. References

[1] Brown, C. Scott. *What is Android? Here's everything you need to know.* Feb. 19, 2022. Accessed: Sep. 29, 2022. [Online]. Available: https://www.androidauthority.com/what-is-android-328076/

[2] Karnik, Nick. *Introduction to Mongoose for MongoDB.* Feb. 11, 2018. Accessed: Oct. 4, 2022. [Online]. Available: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/

[3] Shilpa S. *What is React Native?*. Jul. 01, 2021. Accessed: Sep, 29, 2022. [Online]. Available: https://www.tutorialspoint.com/what-is-react-native

[4] *Application Programming Interface (API)*. Accessed: Sep 18, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/api

[5] *MongoDB vs. MySQL Differences.* Accessed: Oct. 4, 2022. [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql

[6] *iOS.* Aug. 28, 2012. Accessed: Sep, 29, 2022. [Online]. Available: https://www.techopedia.com/definition/25206/ios

[7] *What is JavaScript?.* Sep. 14, 2022. Accessed: Oct. 4, 2022. [Online]. Available: *https*://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[8] *What is RESTful API?.* Accessed: Oct. 4, 2022. [Online]. Available: https://www.mulesoft.com/resources/api/restful-api

[9] *Swift: The powerful programming language that is also easy to* learn. Accessed: Sep. 29, 2022. [Online]. Available: https://developer.apple.com/swift/

[10] *What is Java technology and why do I need it?.* Accessed: Sep. 29, 2022. [Online]. Available: https://www.java.com/en/download/help/whatis_java.html

[11] *JSON Defined.* Accessed: Oct. 4, 2022. [Online]. Available: https://www.oracle.com/database/what-is-json/

[12] *Middleware.* Mar. 5, 2021. Accessed: Oct. 4, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/middleware

[13] *ReactJS – Overview.* Accessed: Oct. 4, 2022. [Online]. Available: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm

[14] *Cox,* Daniel. *What is Xcode, and What can You Do With it?*. Apr. 7, 2022. Accessed: Sep. 29, 2022. [Online]. Available: https://www.corecommerce.com/blog/what-is-xcode-and-what-can-you-do-with-it/

[15] Joe, Alexander. *Apple Inc. – Company Information.* Mar. 7, 2014. Accessed: Sep. 29, 2022. [Online]. Available: https://marketbusinessnews.com/apple-inc-company-information/14919/

XXII. Appendices

XIII.1. Appendix A

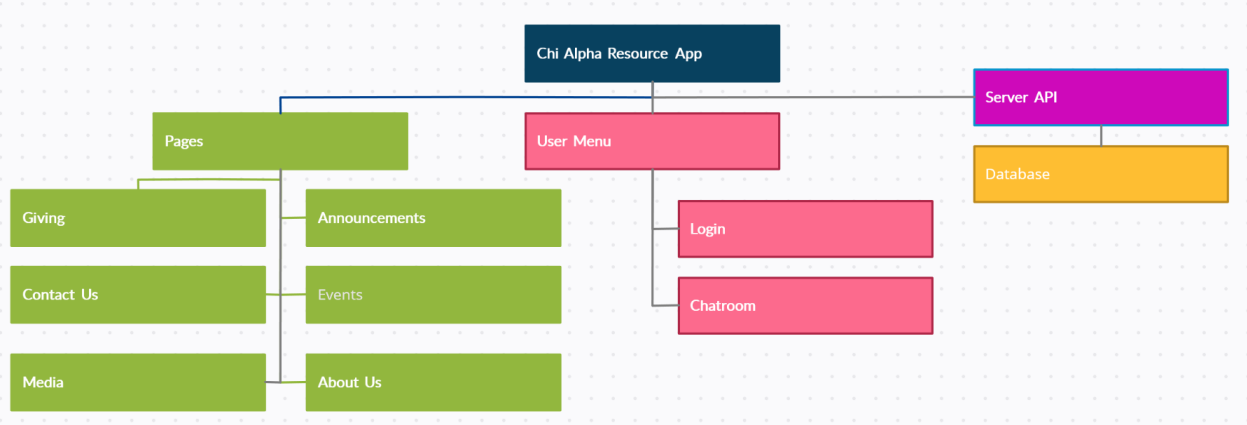

Figure 1

XIII.2. Appendix B

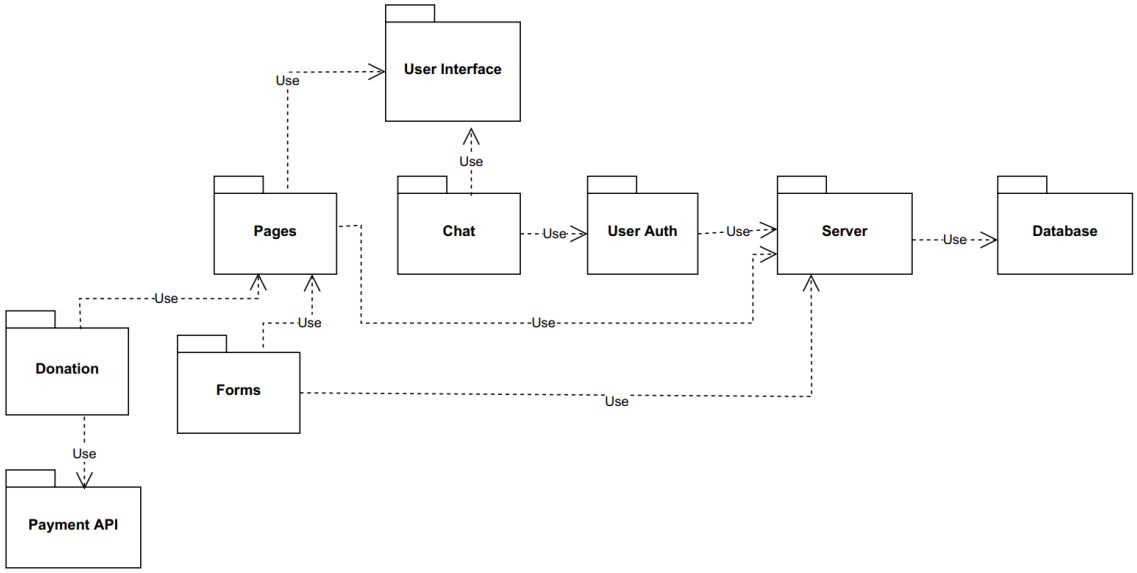

Figure 1

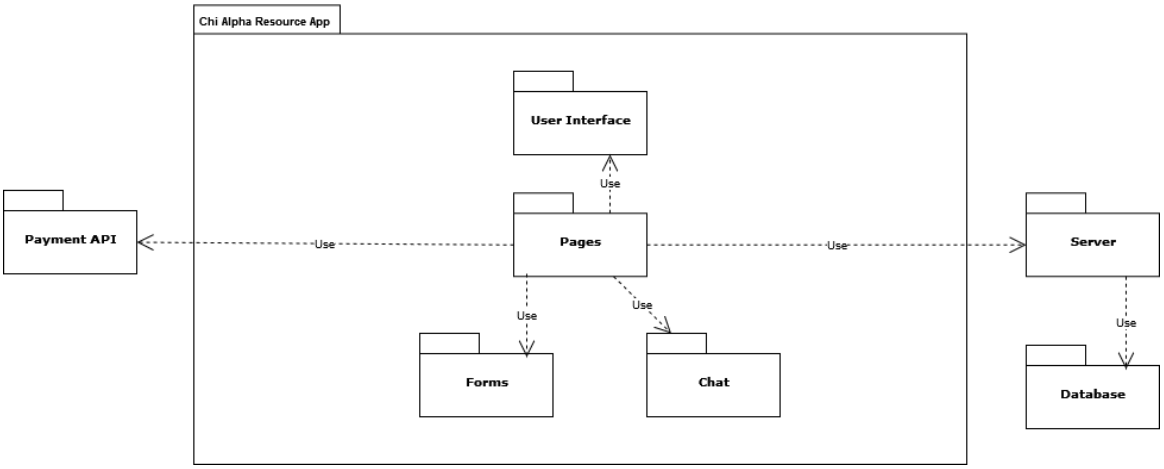

Figure 2

XIII.3. Appendix C

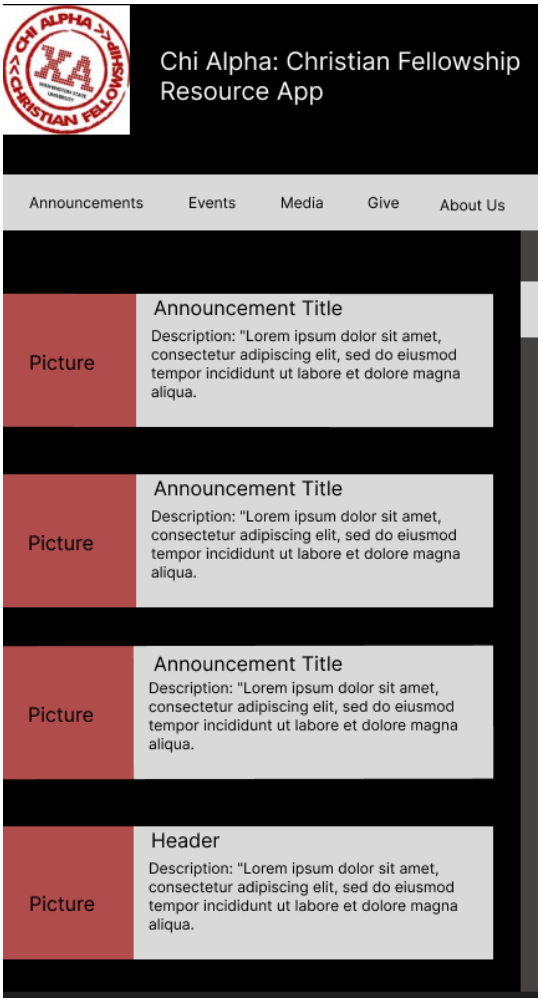
Chi Alpha: Christian Fellowship Resource App
Announcements
Events
Media
Give
About Us
Picture
Announcement Title
Description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Picture
Announcement Title
Description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Picture
Announcement Title
Description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Picture
Header
Description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 1

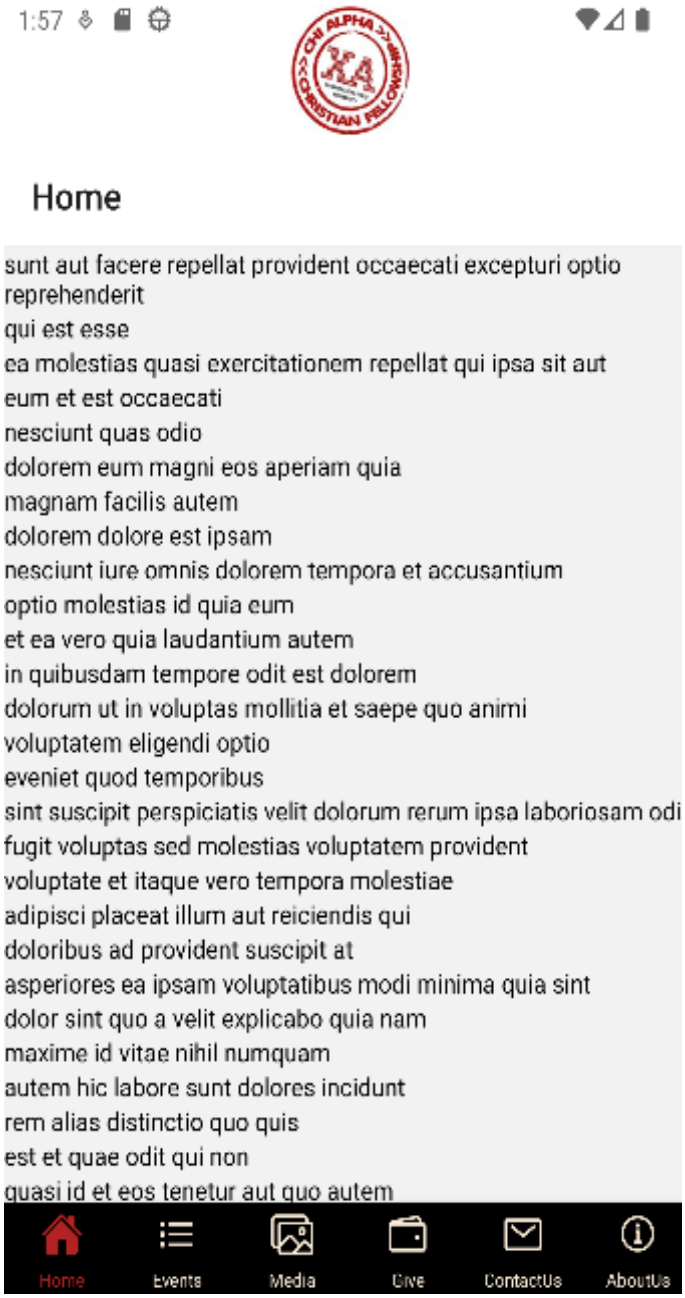
1:57
Home
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
qui est esse
ea molestias quasi exercitationem repellat qui ipsa sit aut
eum et est occaecati
nesciunt quas odio
dolorem eum magni eos aperiam quia
magnam facilis autem
dolorem dolore est ipsam
nesciunt iure omnis dolorem tempora et accusantium
optio molestias id quia eum
et ea vero quia laudantium autem
in quibusdam tempore odit est dolorem
dolorum ut in voluptas mollitia et saepe quo animi
voluptatem eligendi optio
eveniet quod temporibus
sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odi
fugit voluptas sed molestias voluptatem provident
voluptate et itaque vero tempora molestiae
adipisci placeat illum aut reiciendis qui
doloribus ad provident suscipit at
asperiores ea ipsam voluptatibus modi minima quia sint
dolor sint quo a velit explicabo quia nam
maxime id vitae nihil numquam
autem hic labore sunt dolores incidunt
rem alias distinctio quo quis
est et quae odit qui non
quasi id et eos tenetur aut quo autem
Home
Events
Media
Give
ContactUs
AboutUs

Figure 2