

CEED Phone Application

Final Report

CS4624: Multimedia, Hypertext, and Information Access

Instructor: Edward A. Fox

Virginia Tech, Blacksburg, VA 24061

May 2, 2018

Group Team Members:

Madhur Mahajan

Zachary Hensley

Randy Liang

Sean Greynolds

Client:

Center for Enhancement of Engineering Diversity (CEED)

Client Team Members:

Dr. Bevlee Watford

Kristin Morrill

Karis Boyd-Sinkler

Table of Contents

Table of Figures	3
Table of Tables	5
Executive Summary	6
1.0 Introduction	7
1.1 Our Objective.....	7
1.2 Software Life Cycle.....	7
1.3 Project Introduction.....	8
1.4 Problem Context.....	8
2.0 Requirements	9
2.1 Functional Requirements	9
2.2 Non-Functional Requirements	9
3.0 Design	11
3.1 Frontend Design Prototype.....	11
3.2 Implemented Features	14
4.0 Implementation	15
4.1 Appery.io	15
4.2 Explored Options.....	16
4.2.1 React Native/AWS.....	16
4.2.2 AppMakr.....	17
5.0 Testing/Evaluation/Assessment.....	20
5.1 Usability Testing.....	20
5.2 Objectives of Usability Testing	20
5.3 Task Scenarios	20
5.4 Usability Test Documentation	20
5.5 Weinre Debugger	22
6.0 Users' Manual.....	24
6.1 Discussion of the Use Environment	24
6.2 Use Cases	24
7.0 Developer's Manual.....	31
7.1 Development Environment.....	31

7.2 Development Tutorial	31
7.2.1 Logging In	31
7.2.2 Editing the App	33
7.2.3 Database Management.....	39
7.2.4 Server Code	41
7.2.5 Push Notifications.....	43
8.0 Lessons Learned	45
8.1 Timeline/Schedule	45
8.2 Problems and Solutions	45
8.2.1 Complexity.....	45
8.2.2 Communication.....	46
8.2.3 Learning Curve.....	46
9.0 Acknowledgements	48
10.0 References.....	49

Table of Figures

Figure 1.1 - Software Development Lifecycle.....	7
Figure 3.1 - Account Creation Prototype.....	11
Figure 3.2 - Preference Page Prototype.....	11
Figure 3.3 - Hamburger Menu Prototype page.....	12
Figure 3.4 - Embedded Calendar Prototype.....	12
Figure 3.5 - Announcements Prototype.....	13
Figure 3.6 - Forum Prototype.....	13
Figure 4.1 - Appery.io User Interface.....	15
Figure 4.2 - Appery.io Tech Stack.....	16
Figure 4.3 - AppMakr User Interface.....	17
Figure 4.4 - About CEED AppMakr.....	18
Figure 4.5 - Programs Page AppMakr.....	18
Figure 4.6 - AppMakr Home Page.....	19
Figure 4.7 - AppMakr Forum.....	19
Figure 5.1 - DOM Element View.....	23
Figure 5.2 - Custom JS Console.....	23
Figure 6.1 - Login Page.....	25
Figure 6.2 - Create Account Page.....	25
Figure 6.3 - Home Page.....	26
Figure 6.4 - Hamburger Menu.....	26
Figure 6.5 - About CEED.....	27
Figure 6.6 - Calendar.....	27
Figure 6.7 - Programs Page.....	28
Figure 6.8 - Program Details Page.....	28
Figure 6.9 - Forum Page.....	29
Figure 6.10 - Forum Post Details Page.....	29
Figure 6.11 - Announcements Main Page.....	30
Figure 6.12 - Update Tags.....	30
Figure 7.1 - Appery.io Home Page.....	31
Figure 7.2 - Appery.io Login Page.....	31
Figure 7.3 - Dashboard.....	31
Figure 7.4 - Editor Mode.....	33
Figure 7.5 - HTML Element.....	34
Figure 7.6 - HTML Editor.....	34
Figure 7.7 - CSS Editing.....	35
Figure 7.8 - Index Page Editor.....	35
Figure 7.9 - List Forum Post Code.....	36
Figure 7.10 - Get Forum Comments Code Database.....	36
Figure 7.11 - Adding A New Functionality.....	37
Figure 7.12 - Adding a New Plugin.....	38
Figure 7.13 - Adding a Global Function/Variable.....	38
Figure 7.14 - Databases.....	39
Figure 7.15 - Users Database.....	40
Figure 7.16 - Forum Database.....	40
Figure 7.17 - Scripts.....	41
Figure 7.18 - CreateNewPost Script Editor.....	42

Figure 7.19 - RegisterNewUser Script Parameters.....	42
Figure 7.20 - RegisterNewUser Testing.....	43
Figure 7.21 - SendPush Script Parameters.....	44
Figure 7.22 - SendPush Script Run.....	44
Figure 8.1 - Project Timeline.....	45

Table of Tables

Table 5.1 - SUS Calculation Example.....21
Table 5.2 - Our SUS Calculations..... 21

Executive Summary

This report addresses a problem that is often faced by many current and prospective Center for Enhancement of Engineering Diversity (CEED) members and staff. Members may range from pre-college students (e.g., high school) to parents of students. CEED needs an easier way to communicate information about their programs to current and prospective members and their parents as well. Our solution to this problem is creating a cross platform mobile application for an end user. In our application, a user can learn more about CEED and become familiar with all the services offered. Thereafter they can make an informed decision about their program choices, and also can reach out to CEED employees and fellow students with any questions that they might have. This report details our process of implementing this mobile application and the various functionalities it offers. We also discuss the design and requirements for our application.

Below is a list of all the functionalities implemented in our application:

1. User Login Functionality
 - a. Users have ability to create and delete accounts.
 - b. Users can register for an account with minimal information
2. Announcements
 - a. The client will have the ability to make important announcements.
 - b. Users will be notified through an in-app inbox message or push notifications.
3. Calendar
 - a. Users will have the ability to view the CEED event calendar.
4. Database functionality
 - a. NoSQL database to store forum posts and other organization information
5. Programs
 - a. Users will have the ability to view various programs offered by CEED.
 - b. Programs will be categorized by undergraduate and graduate categories.
6. Forum
 - a. Users will have the ability to make forum posts about any topic.
 - b. The client will have the ability to delete any inappropriate forum posts.
 - c. Users will be able to post comments on existing forum posts.

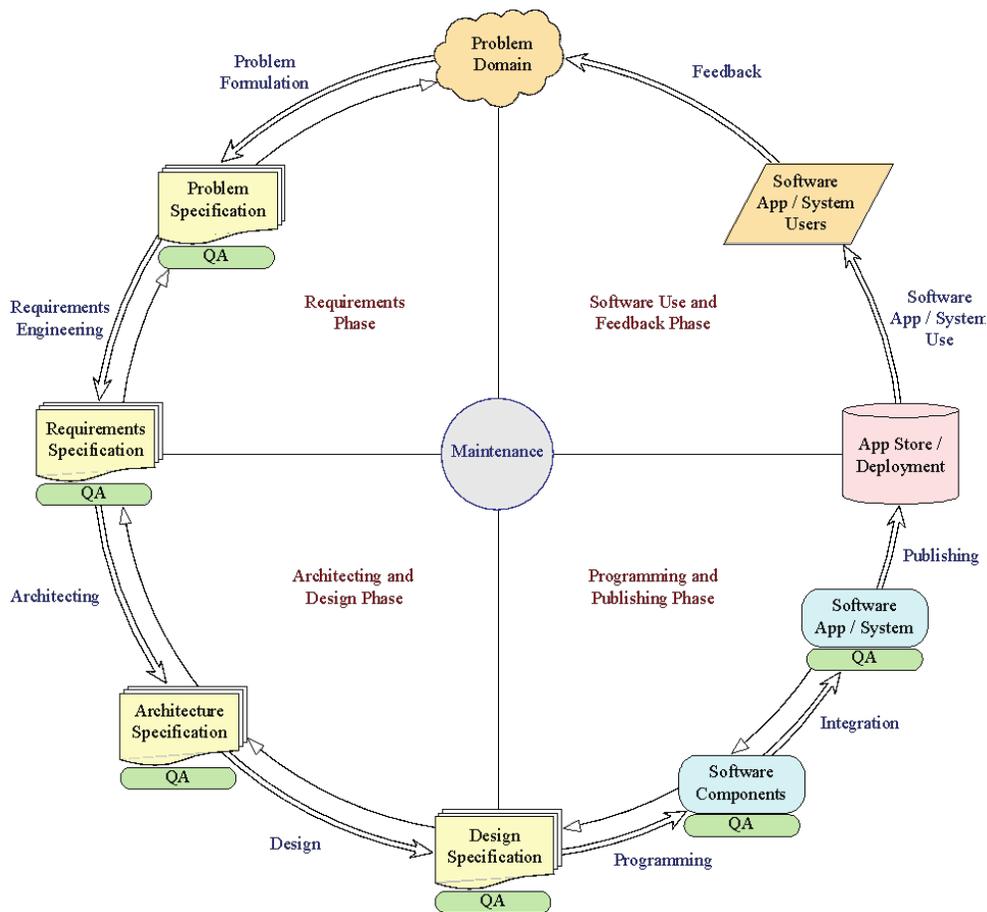
1.0 Introduction

1.1 Our Objective

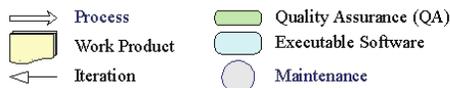
The objective of our team project is to solve our client's problem in a way that satisfies the client and can be completed in a relatively short time frame. Our mobile app solution should cover all the requirements given by the client while putting a secondary focus on showcasing our skills as developers.

1.2 Software Life Cycle

A good software engineer develops software by following the software life cycle shown in Figure 1.1. This is the systems development life cycle (SDLC) that we intend to follow for our mobile application project.



Legend:



Copyright © Osman Balci

Figure 1.1 - Software Development Lifecycle

1.3 Project Introduction

Communication with organization members is stressful enough with the variety of logistics involved from creating a listserv, to constantly reminding members about upcoming deadlines, etc. Our solution to alleviate this stressful process is to develop a cross platform mobile application which can be used to remedy the aforementioned problems.

1.4 Problem Context

Users often have to access CEED's website to find out more information about the organization. Additionally, they do not have a way to ask questions and coordinate events with other prospective members. The current process used to reach out to members is through emails which are often misplaced or not read in a timely manner, creating a communication gap. The app would automatically link to CEED program information and the event calendar. The app will also have a forum functionality which will help further fill the communication gap between CEED and its members. Users can ask questions about specific programs and post their thoughts on the forum page prior to becoming a member of the program.

2.0 Requirements

This section specifies the Functional and Non-Functional Requirements under which our mobile application will be developed.

2.1 Functional Requirements

1. The user shall have the ability to download and open the mobile application regardless of the platform they are using.
2. The user shall have the ability to create an account, depending on client requirements.
3. The user shall have the ability to learn more about CEED as an organization and the various services offered.
4. CEED employees shall have the ability to send announcements to all users registered on the app. The user will receive this announcement as either an in-app message or a push notification.
5. The user shall have the ability to view all of the programs offered by CEED.
6. The user shall have the ability to receive in-app messages from CEED employees.
7. CEED employees shall have the ability to send users push notifications to relay urgent information.
8. The user shall have the ability to view the CEED event calendar and find out information about upcoming events and deadlines.
9. The user shall have the ability to post on the forum regarding a particular topic.
10. The user shall have the ability to view and like (upvote) posts from other users on the website.
11. CEED employees shall have the ability to delete any inappropriate posts made on the website.

2.2 Non-Functional Requirements

1. The solution provided to the client must be a cost-effective solution for the project requirements.
2. The source code for the mobile application shall be handed over to the client at the end of the semester project.
3. The client shall be provided with all the login credentials and other information for all the technologies used in this project.
4. In addition to being cost effective, the solution provided must be relatively easy to maintain by CEED employees who have little prior coding experience.

5. The solution shall not use any outdated technologies which may be harder to maintain and support in the future.
6. Reliability -- prevent crashes/errors
 - a. We want the app to continue running gracefully in case of invalid user input.
 - b. Safeguards will be put in place to prevent the app from crashing.

3.0 Design

3.1 Frontend Design Prototype

For a prototype of our mobile application, we decided to design several wireframes to graphically represent the design. Figure 3.1 shows the account creation page on first opening the application; the user will be given an option either to create an account or remain anonymous depending on the client requirements. Thereafter they will be directed to a page where they can fill out their preferences and information about themselves as shown in Figure 3.2. Depending on the project scope, these preferences will later be used to show filtered forum posts and announcements to the user.

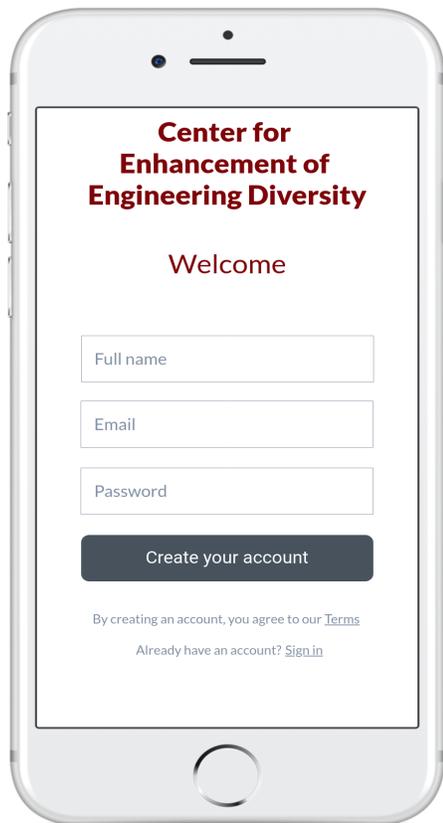


Figure 3.1 - Account Creation Prototype

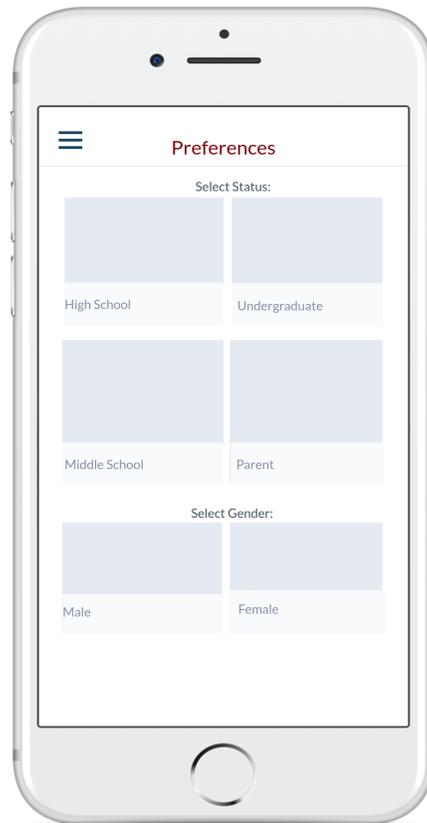


Figure 3.2 - Preference Page Prototype

Figure 3.3 shows the hamburger menu which will be implemented in our application. The menu button will always be visible on the top left corner of the app and can be used to switch between the different pages of the application. The menu shall also have the logout option if user account functionality is implemented. Figure 3.4 shows the CEED event calendar which can be accessed by clicking the "Calendar" option in the menu. Users will not have the ability to add any new events

to this calendar but rather will only be able to view information about upcoming deadlines and events. Depending on the project scope and budget, this calendar functionality will either be embedded within the app or a link to an external page on the phone's web browser.

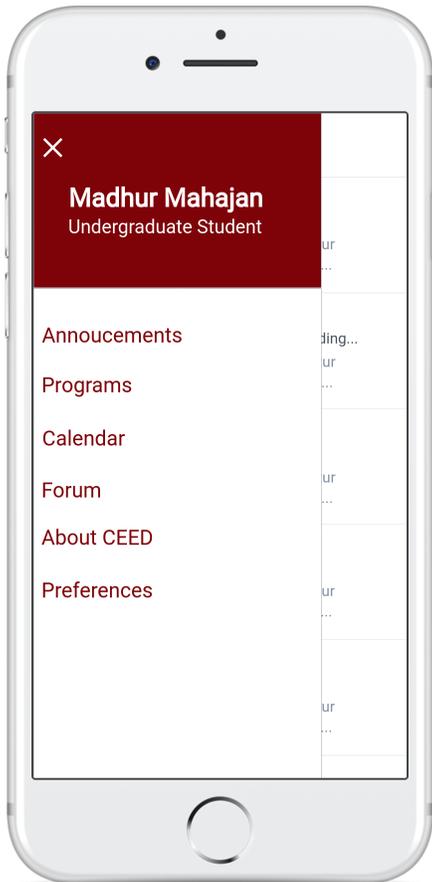


Figure 3.3 - Hamburger Menu Prototype

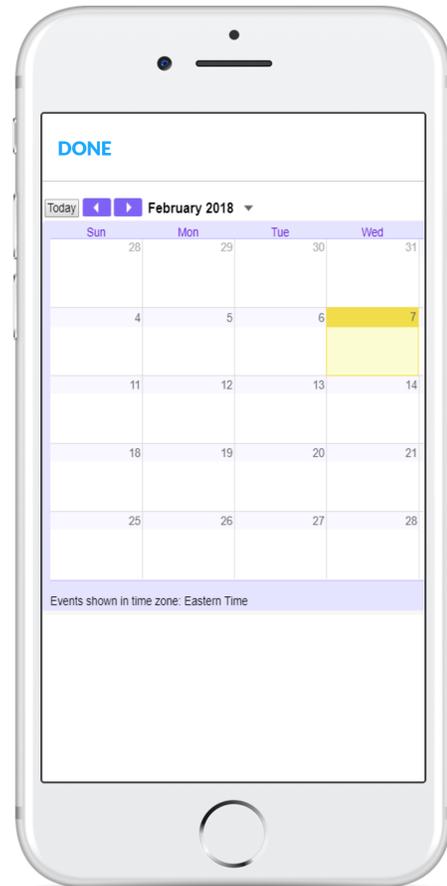


Figure 3.4 - Embedded Calendar Prototype

Since the app will primarily be used to convey time-sensitive information, we have implemented the announcements functionality where they can convey such information as shown in Figure 3.5. Users will receive these announcements as either an in-app message or a push notification depending on the urgency of the information.

Figure 3.6 shows the forum page which is the most complex functionality in this application. This page is where users will be able to post questions and comments. The users can remain anonymous and post a thread about a certain topic (CEED program). The scope of the project determines whether these forum posts will be tagged under a certain tag predetermined by the client. The client will have an administrative account on the forum page to reply to the questions asked.

Additionally, the administrative account will also have the ability to delete certain forum posts which are deemed inappropriate.

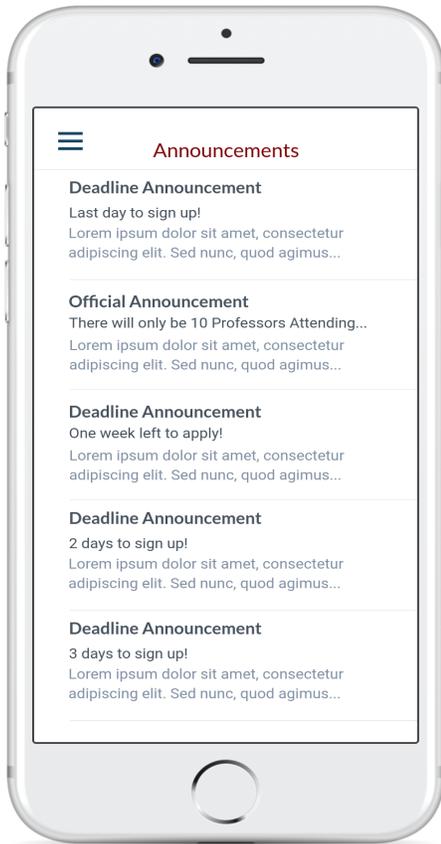


Figure 3.5 - Announcements Prototype

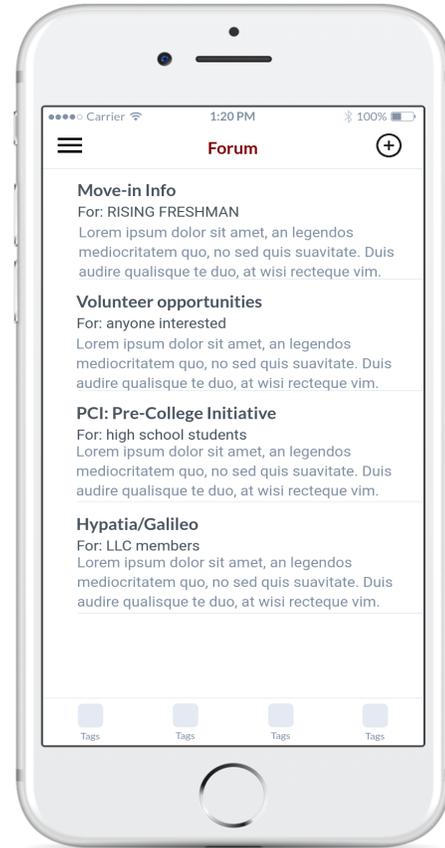


Figure 3.6 - Forum Prototype

3.2 Implemented Features

Certain features will be necessary to implement in order to achieve optimal user experience and functionality:

- Hamburger Menu - useful for navigating between various in-app pages & features.
- Calendar - to view upcoming deadlines and events
- Embedded Video - informational CEED video embedded in the application
- Push Notifications - notify users about important information when the app is not open
- In-app Community - forum where users can share their thoughts and interact with one another

4.0 Implementation

4.1 Appery.io

After careful consideration, we selected Appery.io, which is a cloud platform that makes developing cross-platform mobile applications simple. Appery.io provides a backend containing server-side Javascript and a NoSQL database, which the developer can write their own scripts for querying the database or pushing notifications to the client. The ability to write your own server-side scripts is one way Appery.io offers extended functionality. Appery.io also has a superior, extensive GUI for creating and editing components in the application as shown in Figure 4.1. Similar to that of xCode, Appery.io's GUI offers numerous features that immediately gets converted into source code which may be exported at any time.

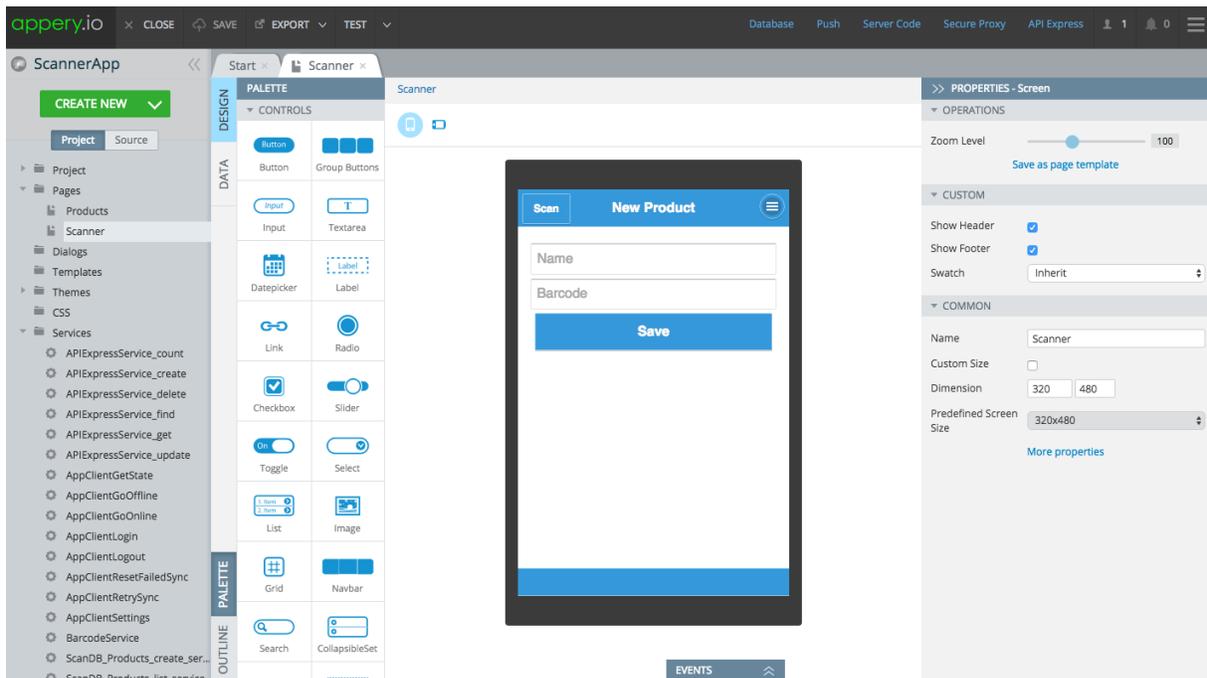


Figure 4.1 - Appery.io User Interface

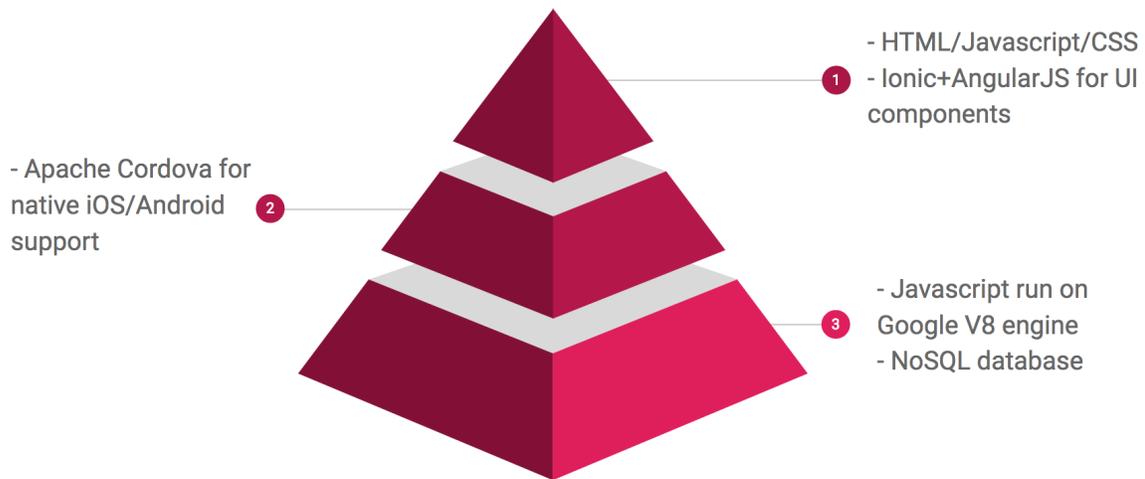


Figure 4.2 - Appery.io Technology Stack

Figure 4.2 shows the technology stack that Appery.io uses. On the top level we have the front-end elements which are heavily customizable through the languages and frameworks listed. Next, we have Apache Cordova which allows for third party plugins. Our database is in NoSQL.

4.2 Explored Options

4.2.1 React Native/AWS

One of the possible solutions our group came up with when evaluating the needs of our client was a React Native front-end with a Node.js backend hosted on AWS. React Native allows for the easy development of cross-platform native mobile applications for both iOS and Android using Javascript, React, and native Java/Swift code when needed. We chose Node.js as our server-side language as it allows for easy scalability and event-driven architecture which makes client-server communication more intuitive.

We chose AWS as our hosting service as they have many services available which makes developing a mobile application much easier, and CEED could not provide us with a Linux server to host our application. After the initial 12-month free trial of AWS, we estimated that it would cost roughly \$4-8 a month for a EC2 t2.nano or t2.micro instance (Amazon EC2 Pricing)^[3]. The AWS Mobile Hub provides access to managing and adding the various AWS Services to our React Native project such as: AWS Cognito for user account management, AWS DynamoDB for a NoSQL database, and AWS Lambda for an event-driven, serverless computing platform.

The client did not choose this particular route, mainly because adding features, modifying the project, and maintaining the application would mean they would have to actually edit the React or Node.js code, make changes to the AWS configurations, and re-publish the application to the app stores. CEED stressed that they needed ease of use whenever they wanted to update the application as they would not have a dedicated development team that would be able to assist them in the future.

4.2.2 AppMakr

When coming up with our implementation of the mobile application for CEED, our group evaluated all of the client's requirements such as price, future usability, and functionality. While discussing the various options we evaluated, the client decided that the platform AppMakr can possibly satisfy their needs. AppMakr^[4] is a do it yourself mobile application creation platform that allows you to create cross-platform iOS and Android applications with minimal to no coding required. Figure 4.3 shows the application editor where you can add various features such as menus, photo galleries, forums, calendars, and even your own custom HTML5 pages.



Figure 4.3 - AppMakr User Interface

Although AppMakr doesn't provide as much functionality and features as some of the other options described below, it can be enough to create a minimum viable product for CEED, and the paid version of the product will only be \$9/month which

includes additional features over the free version, including hosting and publication to the app stores. Furthermore, AppMakr also has an option to supply the source code for the premium package. This is far and beyond the most economical choice which appeals to CEED; however, there are many other options available that offer greater functionality. We have yet to reap the benefits of the premium version, though the ease of use of the UI is promising for future updates to the app on CEED's part.

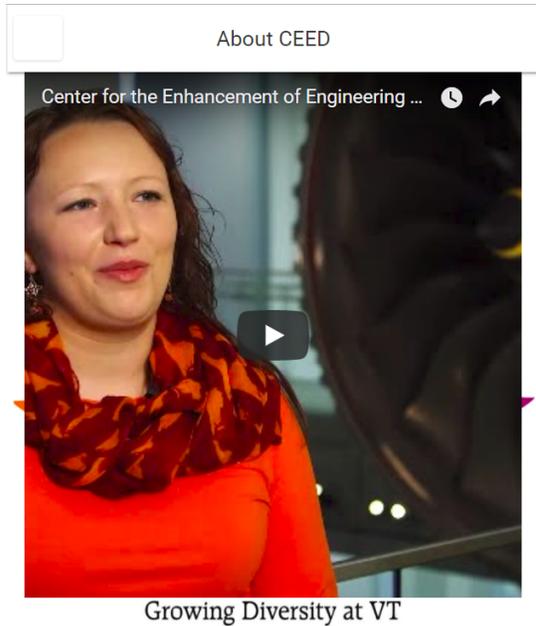


Figure 4.4 - About CEED AppMakr

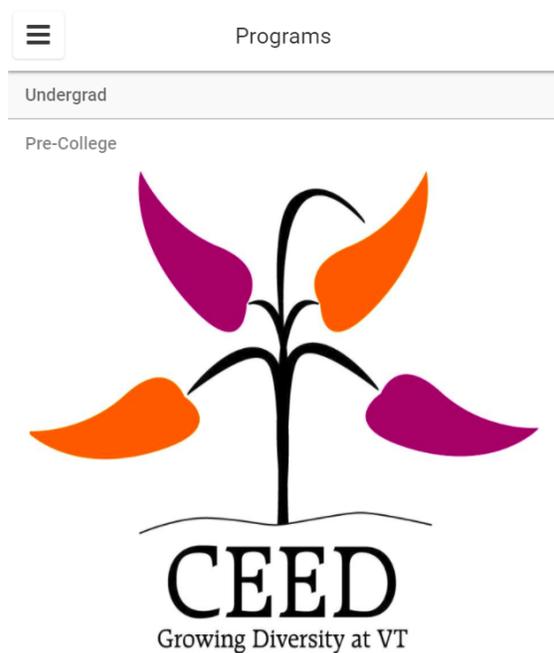


Figure 4.5 - Programs Page AppMakr

Figure 4.4 shows our “About CEED” which contains an embedded video describing the organization. Figure 4.5 contains links that link directly to the available programs that CEED has to offer.

Virginia Tech



Figure 4.6 - AppMakr Home Page

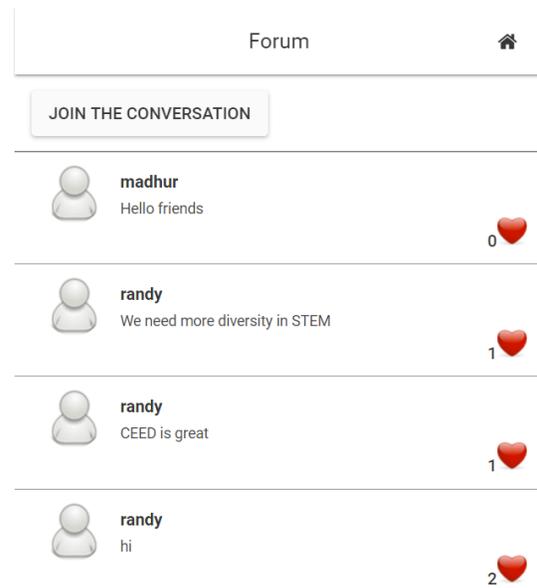


Figure 4.7 - AppMakr Forum

Figure 4.6 shows the home page with a hamburger menu on the bottom left corner. Figure 4.7 shows our current progress for our forum page which only allows for anonymous posting without threading due to the aforementioned limitations of the free version.

5.0 Testing/Evaluation/Assessment

5.1 Usability Testing

The app will need to be user-friendly which is often lost to mobile app developers that test the app by giving it to other more technically proficient users. Ideally, a variety of users should be sampled to gauge what is missing, what could be enhanced, and any use-cases that are overlooked.

5.2 Objectives of Usability Testing

The app will undergo UX and UI testing throughout the development phase and also after completion of the minimum viable product. The intended user demographic ranges from pre-college, undergraduate, and parents who would like to know more about Virginia Tech's engineering diversity among other inquiries they may have. To reiterate our problem context, we would like to provide users who are interested in CEED a centralized means of keeping up with CEED's events and programs by sending push notification alerts and offering a forum section for any lingering questions that can be addressed by other members or CEED administrators.

5.3 Task Scenarios

Sample users will be given the app to determine if it indeed accomplishes what we intended on top of the aforementioned ease of use. The most crucial features that need to be met include: sorted forum posts by topic/program, a portal to the calendar of events, access to information regarding the available programs, and announcements sent through push notifications as previously stated.

5.4 Usability Test Documentation

5.4.1 System Usability Scale (SUS) Test

Our test users will be provided the above information and asked to provide feedback using the System Usability Scale (SUS). Upon consideration between it and Nielsen's heuristics, we found this to be the superior usability test as it is easier to follow and provides us more useful feedback for the purposes and intents of this project.

The following statements would be evaluated on a scale of 1-5^[8]:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Our calculations will be made by subtracting 1 from the odd-numbered questions and subtracting from 5 for the even-numbered questions, multiplying the sum of these results by 2.5 as in Tables 5.1 and 5.2:

	A	B	C	D	E	F	G	H	I	J	K	L
1	SUS Calculation											
2												
3	Participant	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	SUS Score
4	p1	1	5	1	5	1	5	1	5	1	5	0.0
5	p2	2	4	2	4	2	4	2	4	2	4	25.0
6	p3	3	3	3	3	3	3	3	3	3	3	50.0
7	p4	4	2	4	2	4	2	4	2	4	2	75.0
8	p5	5	1	5	1	5	1	5	1	5	1	100.0

Table 5.1 - SUS Calculation Example

Participant	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	SUS Score
p1	1	2	4	2	3	2	3	2	3	2	60
p2	1	1	4	1	2	1	4	1	4	1	75
p3	1	1	5	1	3	1	3	2	3	1	72.5

Table 5.2 - Our SUS Calculation

We had three participants: a college student without a strong technical background (p1), a high schooler with a technical background (p2), and a high school student who is familiar with mobile APP UI but does not have a technical background per se (p3). This feedback was gathered prior to our Appery.io funding. Constructive criticism was welcomed, even if it did not pertain to the 10 questions above. All the users felt comfortable navigating the application while they collectively found it to be not too complex. Even P1 did not indicate help was needed. We felt that sampling a larger variety of people may have been beneficial, though we prioritized targeting

younger individuals since that is our primary demographic (that would be using this app). While the app was relatively easy to use, we received comments to make the app more visual appealing which was evident as it was still in its early stages. Some of the major things we implemented after receiving feedback were: a main page that mirrors the pages that exist on the hamburger menu, and web pages that were directly integrated in the app with cleaner margins (rather than opening it in a separate window namely the Google Calendar of CEED events).

5.5 Weinre Debugger

Since the platform automatically generates its own tests (provided in the source code) there was little room for us to do any manual testing. Because of this, Appery provides a debugger tool known as Weinre. This tool allows us to debug the app while it is being tested in another session, which proved to be convenient with concurrent users on the platform as seen in Figure 5.1. We are able to view the DOM (DIV, HTML, BODY) elements and modify attributes, styles, and properties similar to in xCode or Android Studio^[12]. The local storage and session storage variables were found in the resources tab while the network inspector allowed us to view the network requests. Similarly, in the Console view on the top menu a user can execute custom JavaScript code in the app shown in Figure 5.2.

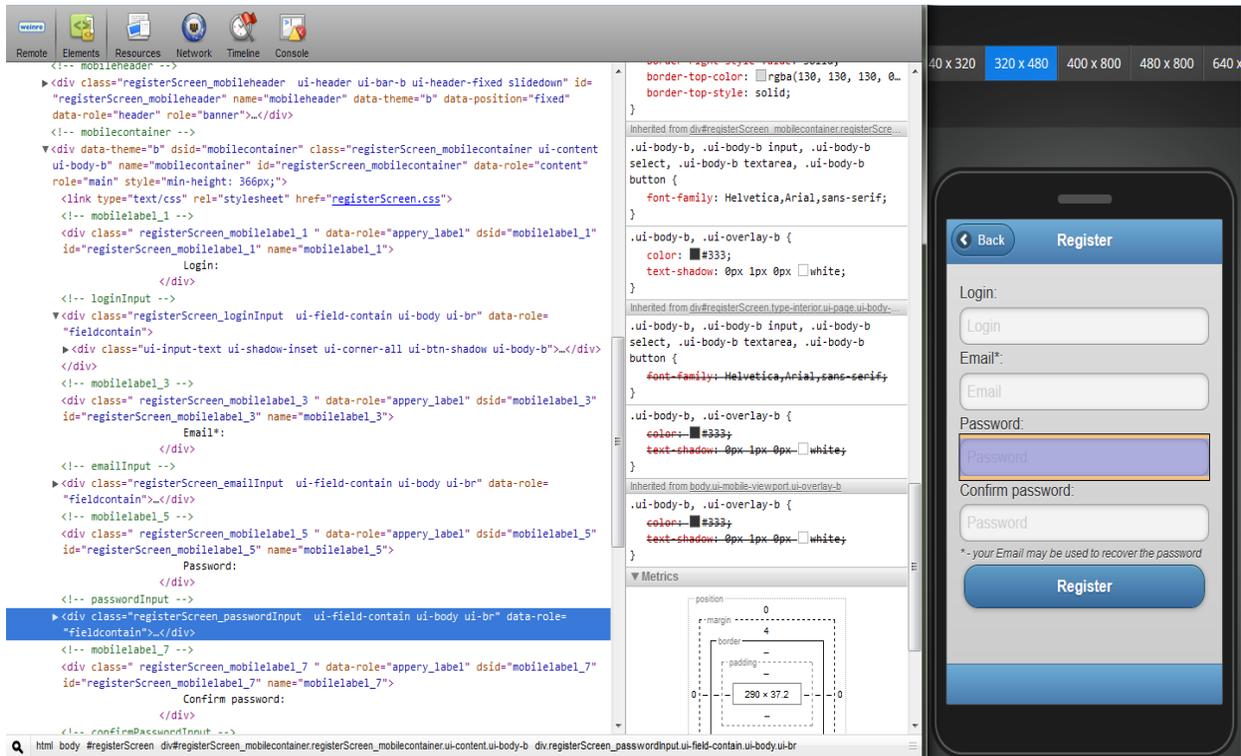


Figure 5.1 - DOM Element View

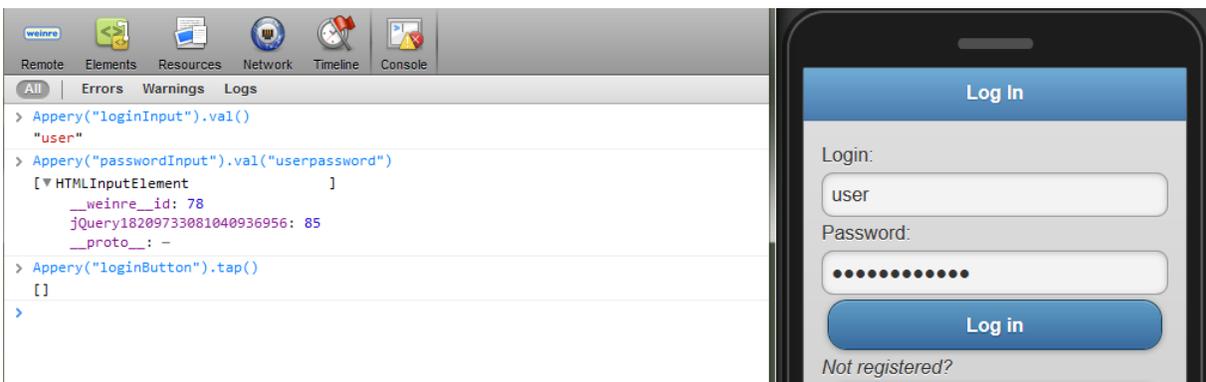


Figure 5.2 - Custom JS Console

6.0 Users' Manual

6.1 Discussion of the Use Environment

The use environment for our app will be an iOS and Android cross-platform application. This will allow updates to be made to the app in one place and have it rolled out to both platforms in the same way without duplicating work. Using the app will be as simple as downloading the app from the app store on either platform and installing it on a mobile device. From there, the wide array of uses of the app will be available right at the fingertips of the user.

6.2 Use Cases

Use Case UC1: Check the CEED calendar

Primary Actor: User

Success guarantee: User is shown CEED calendar

Main success scenario:

1. User opens the app
2. User selects the Calendar from menu
3. User is routed to the official CEED Google calendar

Frequency of Occurrence: Daily/Weekly

Use Case UC2: Learn about CEED

Primary Actor: User

Success guarantee: Video is played for user

Main success scenario:

1. User opens the app
2. User selects About CEED from the menu
3. User is shown a video about CEED

Frequency of Occurrence: Once or twice ever

Use Case UC3: Make a forum post

Primary Actor: User

Success guarantee: A post is created by user

Main success scenario:

1. User opens the app
2. User selects Forum from the menu
3. Forum is loaded
4. User enters text he wants to post
5. User selects Post

6. Post is created by the app and added to the forum

Frequency of Occurrence: Weekly/Monthly

Use Case UC4: Receive urgent information and upcoming deadlines through push notifications

Primary Actor: User

Success guarantee: User obtains and reads the information sent by CEED

Main success scenario:

1. User does not have the app open
2. The user is notified by the CEED phone app
3. The user is aware of the information sent by CEED

Frequency of Occurrence: Weekly

6.3 User Tutorial

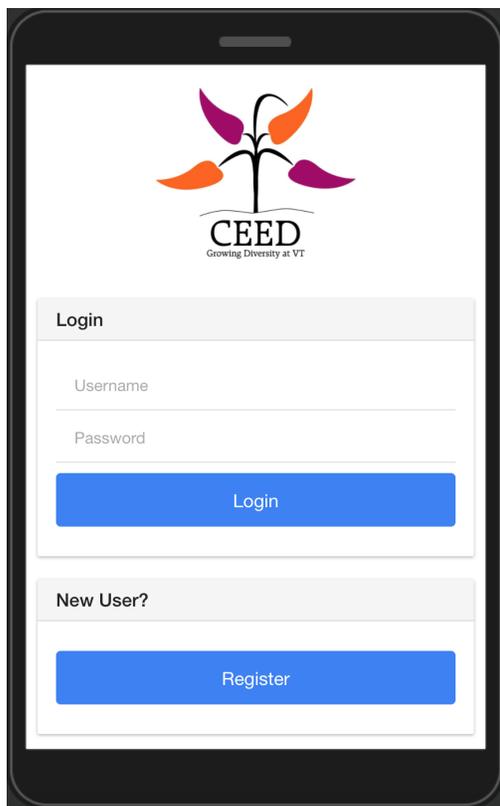


Figure 6.1 - Login Page

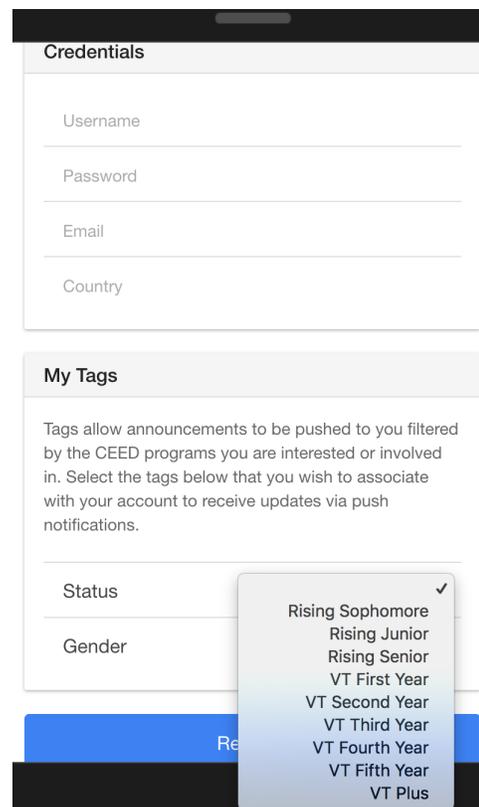


Figure 6.2 - Create Account Page

Figure 6.1 shows the login page which the user will be directed to when they open the app. Here the user will have the option to create an account or login if they created an account in the past. As shown in Figure 6.2, the user has the ability to

create an account with numerous options for status which are simply curated tags that the user specifies upon creation of the account.

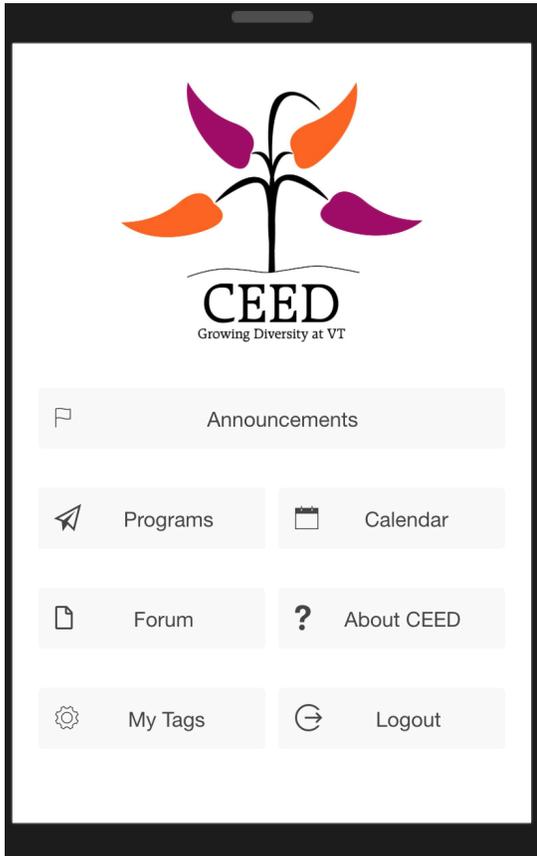


Figure 6.3 - Home Page

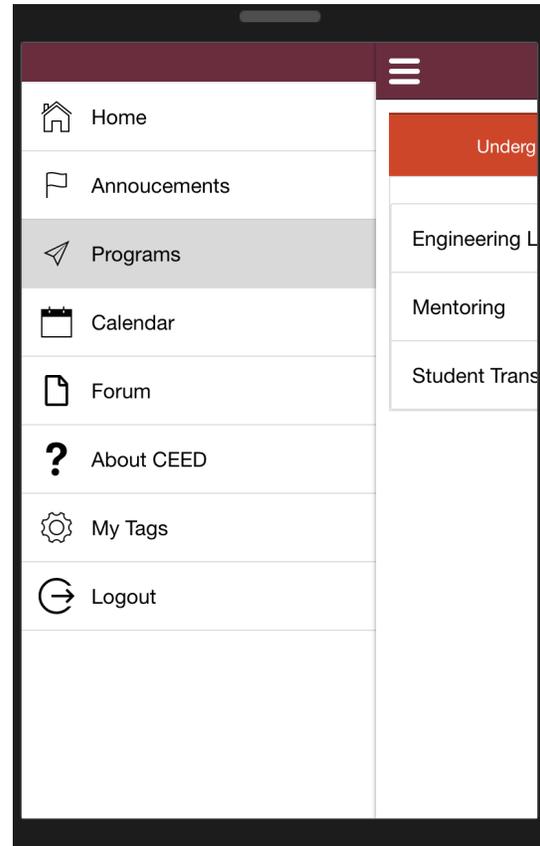


Figure 6.4 - Hamburger Menu

Once logged in, the user is directed to the home page as shown in Figure 6.3. The user can find links to various other app pages on the home page. Similarly, we have also added hamburger menu (Figure 6.4) functionality to allow the user to navigate to other app pages with one click, enhancing the user experience. The user will also have the ability to log out of their account by clicking the logout button on the menu and also the home page.

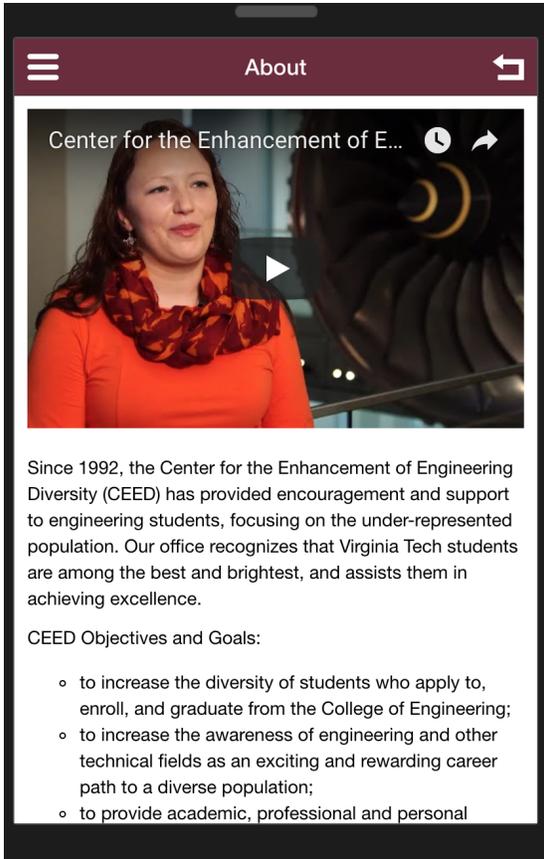


Figure 6.5 - About CEED

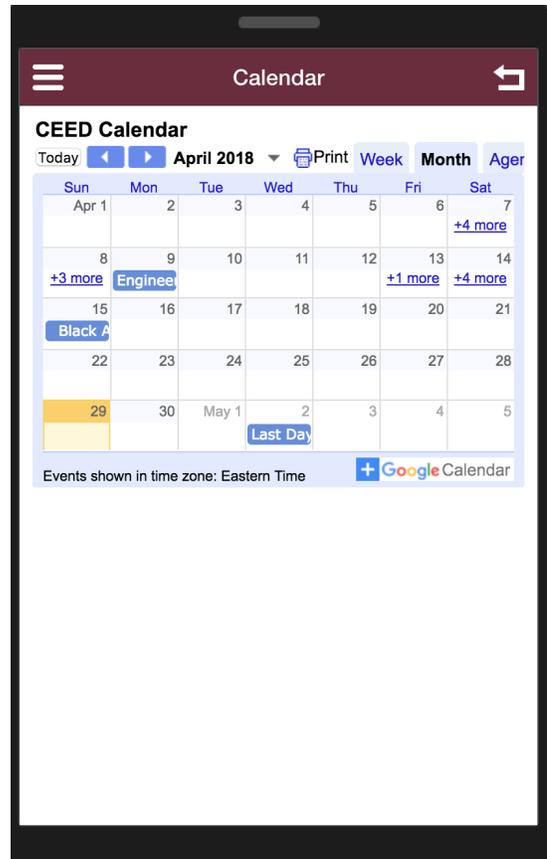


Figure 6.6 - Calendar

Figure 6.5 is the About CEED page where users can learn about CEED and its mission^[5]. This page includes an embedded video which the users can view without exiting the app. Likewise, the user can also navigate to the calendar page which has the embedded CEED calendar (Figure 6.6). The calendar lists CEED events and other important deadlines. The user can click on any event to find out more information about it. The information will be displayed in a pop-up dialog box.

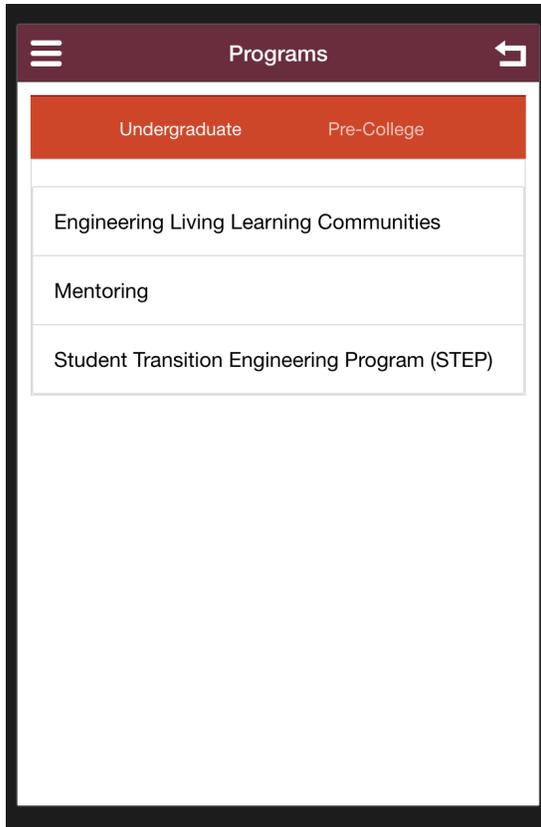


Figure 6.7 - Programs Page

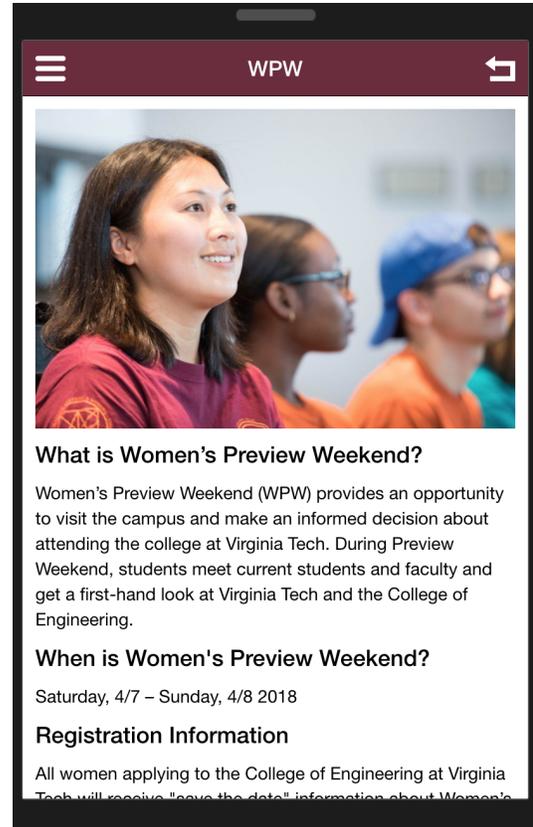


Figure 6.8 - Program Details Page

The users can navigate to the programs page to view the list of all the programs offered by CEED. Figure 6.7 shows the list of undergraduate college programs offered by CEED. Clicking on the “Pre-College” tab on the top right will show a list of all the pre-college programs offered. Thereafter, the user can click on any of these programs to find out more information about them as shown in Figure 6.8. Program information will either be displayed on a different app page or in an in app browser for external programs.

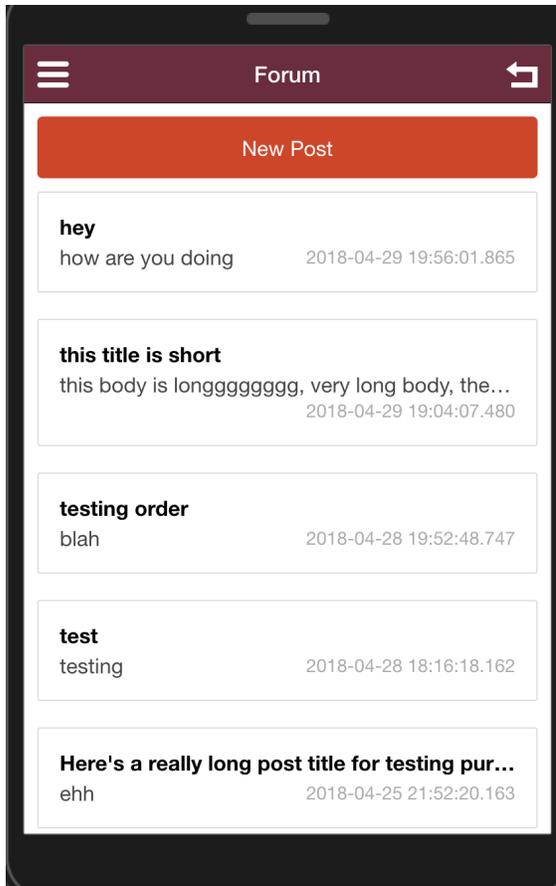


Figure 6.9 - Forum Page

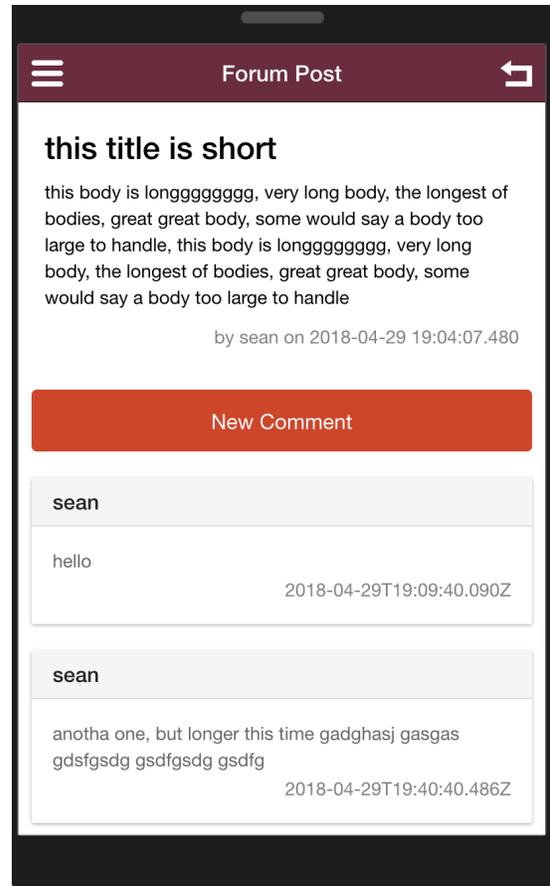


Figure 6.10 - Forum Post Detail Page

One of the main goals of our app is to allow users to communicate with one another and discuss various programs and events. The forum page shown in Figure 6.9 is where the users can view all the forum posts. The username of the user will also be displayed under the title of the post. Clicking the “New Post” button on the top will direct the user to a different page to create a new post.

Clicking on any of the posts will direct the user to the forum post details page (Figure 6.10) where they can comment on their thoughts about the post. The comments are displayed in a manner where the latest comment is shown at the bottom of the page.

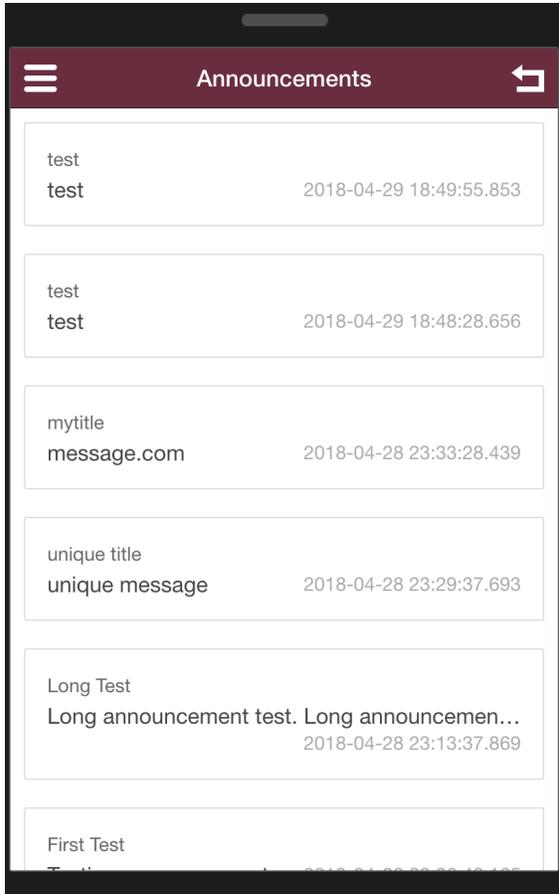


Figure 6.11 - Announcements Main Page

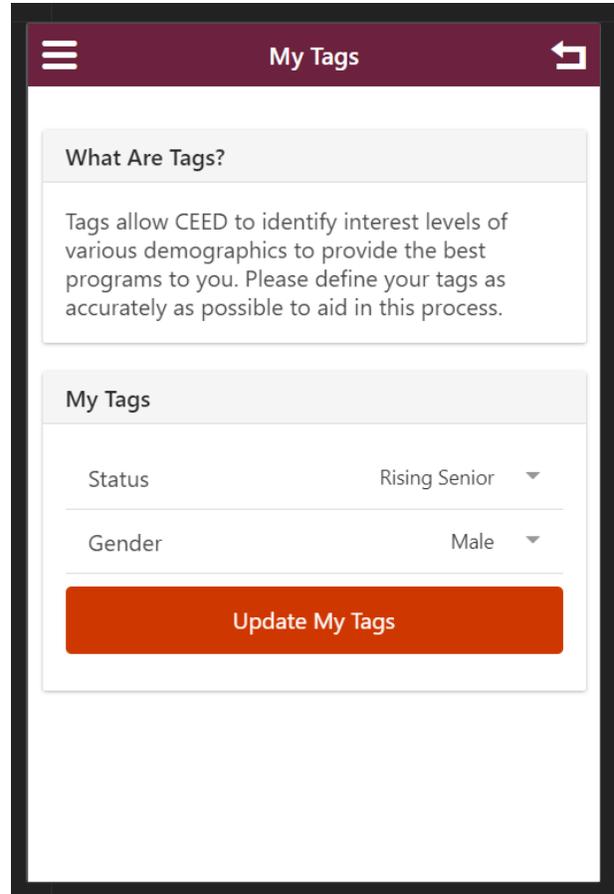


Figure 6.12 - Update Tags

Figure 6.11 shows the announcements page. Announcements include important CEED information and upcoming deadlines. CEED administrators will have the ability to send push notifications to the user to alert them. These push notifications will be stored in the database and also be listed in the announcements page.

The “My Tags” page from the hamburger menu in which the user can change their status and/or gender (Figure 6.12) allows CEED to gather target demographic data to create more effective programs. It is up to the user to provide information that is as accurate as possible.

7.0 Developer's Manual

7.1 Development Environment

The development environment of our app is Appery.io, a cross-platform mobile app development environment. This platform offers end to end services ranging to a database to push notifications and forum capabilities. Although this platform has a learning curve for the developer, it is quite easy to perform trivial updates to the user interface of the application. Appery.io is a do it yourself platform which provides the user with an interface to develop the application. The developer can leverage drag and drop capabilities to make simple edits. However, it is important to note that deeper knowledge of the platform would be necessary to make complex updates to the application. The benefit of this platform is that a developer does not have to worry about most of the source code of the application. Additionally, the developer does not have to write unique code for IOS and Android applications as that is handled by the platform.

7.2 Development Tutorial

7.2.1 Logging In

In order to make changes to the app one first must log in at <https://appery.io/>. This is done by selecting “go to platform” in the top right corner of the home page (Figure 7.1) and entering the required credentials on the login page (Figure 7.2).

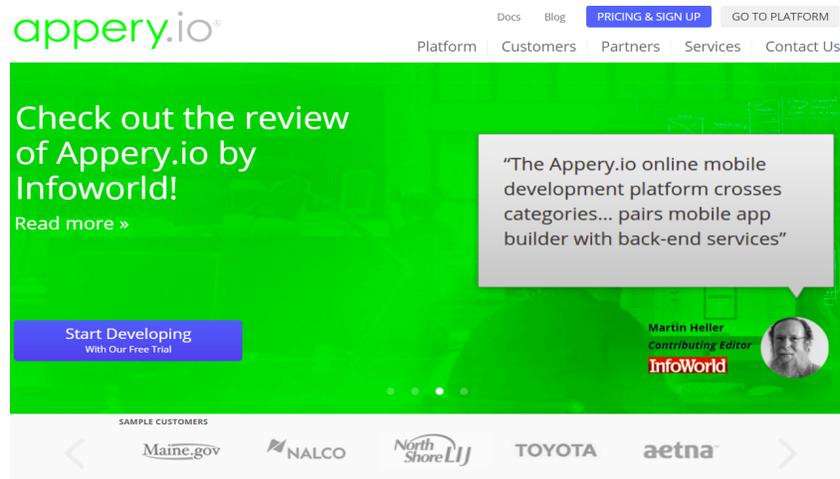


Figure 7.1 - Appery.io Home Page

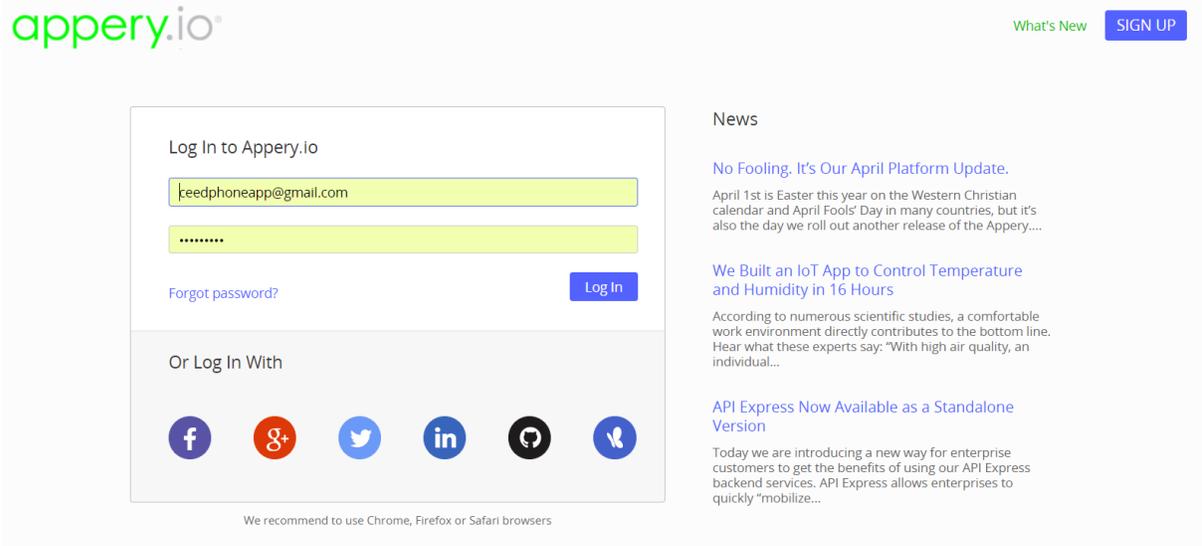


Figure 7.2 - Appery.io Login Page

Once logged in, access to the app dashboard (Figure 7.3) is available to edit both front and back end parts of the app. To edit the front end of the app, one must click open on the respective (CEED) app to open the in-browser editing software. To update the back end of the app, the develop can navigate to the databases and server code tabs.

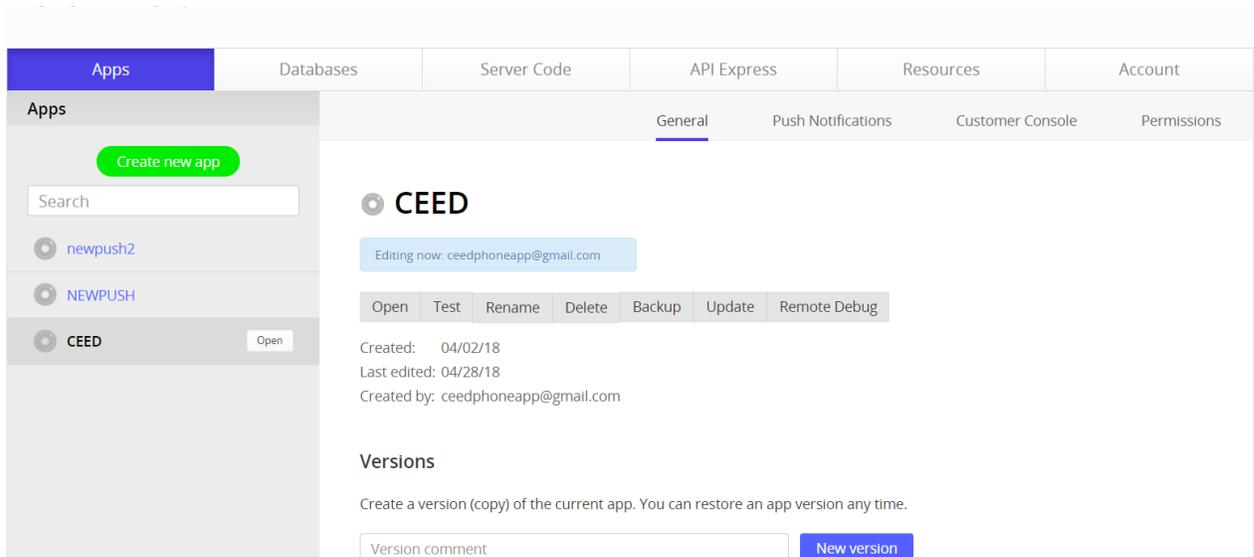


Figure 7.3 - Dashboard

7.2.2 Editing the App

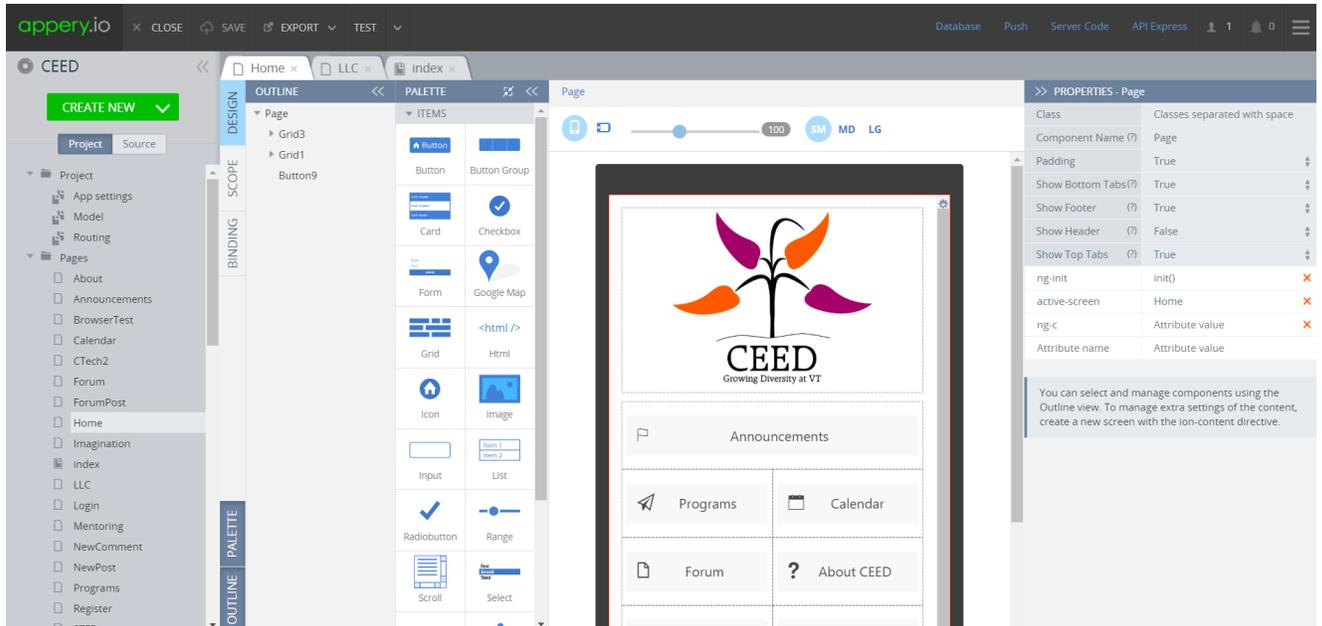


Figure 7.4 - Editor Mode

The editor mode (Figure 7.4) allows customization of the front end of the app. It features drag-and-drop elements that have easily changeable options that are formulated into code by Appery.io. Buttons, images, grids, and textboxes are just a few of the available elements. In order to place an element on a page, just click and drag a new element from the items section onto the canvas. Some pages, such as the About page, are purely HTML and contain just an HTML element. This element can be edited by selecting the HTML element in the Page then selecting the “edit” button to the right of the HTML section in the rightmost Properties section (Figure 7.5). This will bring up an HTML editor (Figure 7.6) that can easily allow updates to the page.

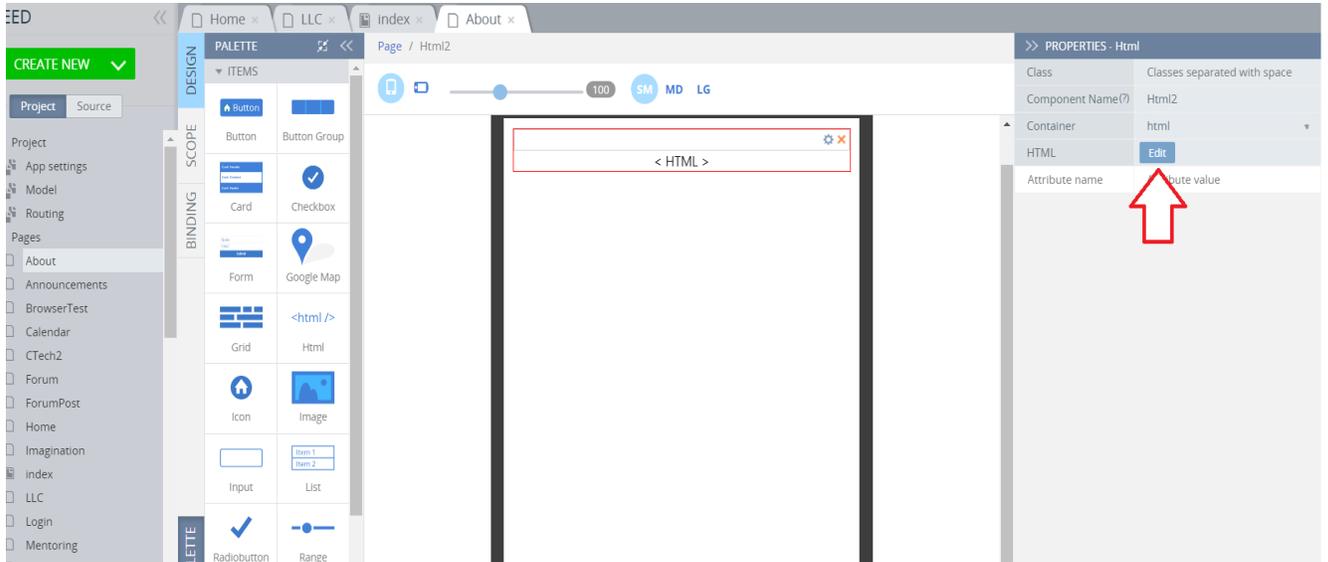


Figure 7.5 - HTML Element

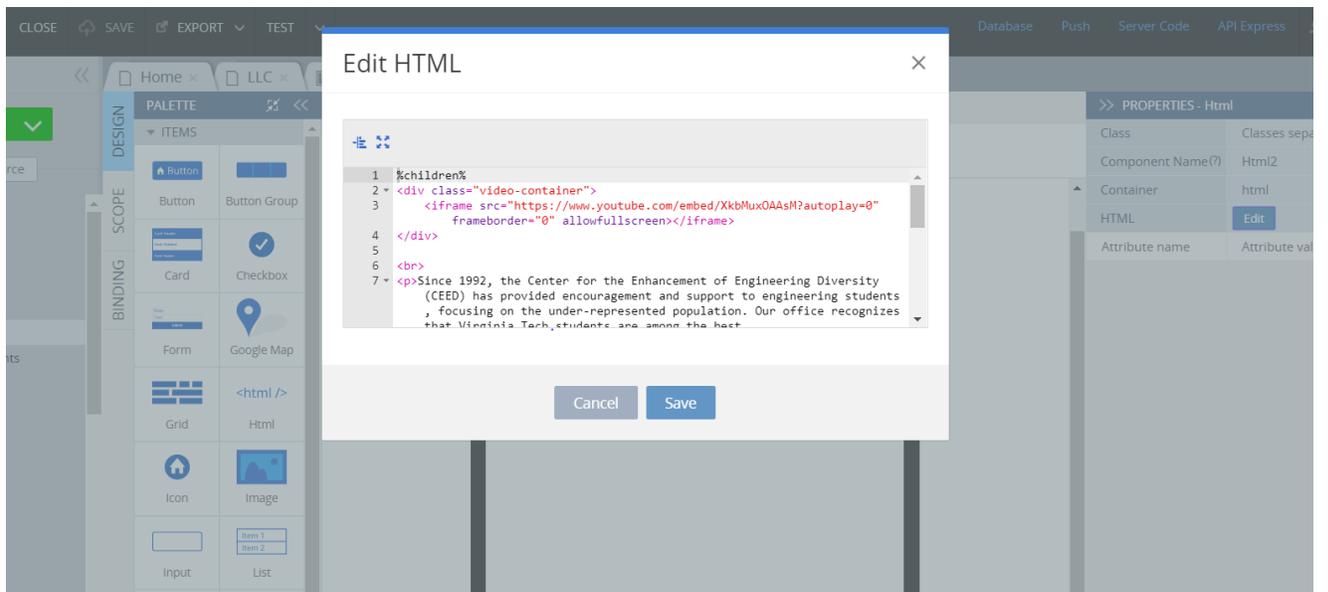


Figure 7.6 - HTML Editor

The leftmost section in the editor allows navigation between pages and scripts in the app. The CSS and JavaScript dropdowns allow editing of the code that controls layout (Figure 7.7) and client-side scripting within the app.

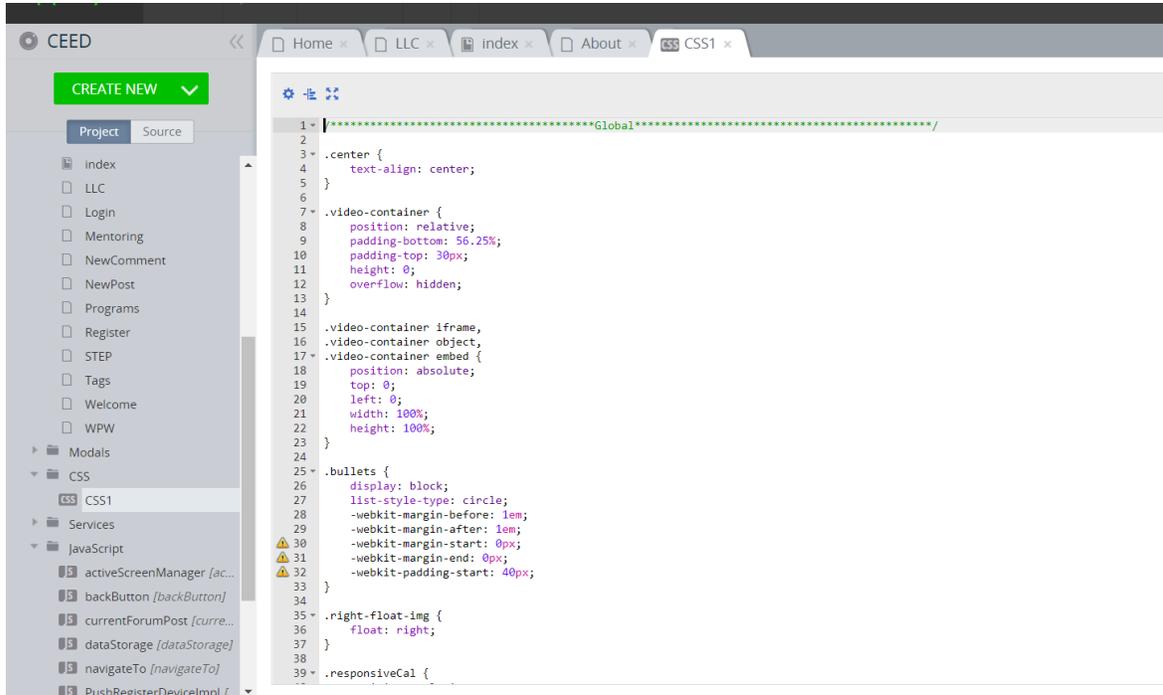


Figure 7.7 - CSS Editing

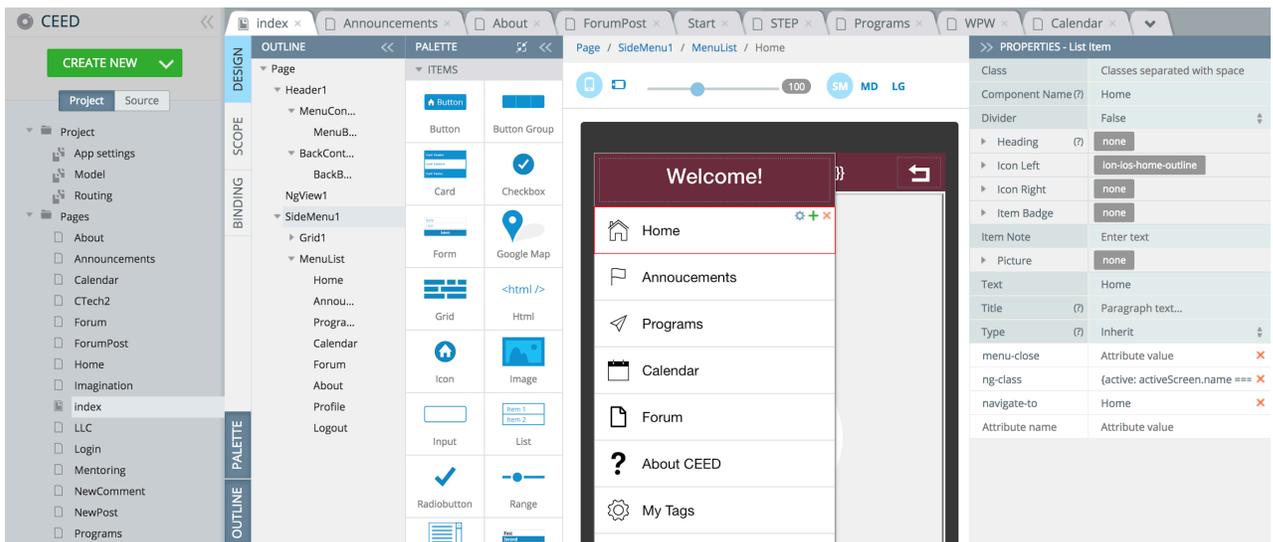


Figure 7.8 - Index Page Editor

Figure 7.8 shows the index page editor where the user can update the sidebar and the hamburger menu. The developer can simply use the visual editor on the right-hand side to add more elements to the menu list by clicking the “+” button the top right.

In Figure 7.9, the admin is able to access the code that allows users to view the forum posts, while Figure 7.10 is the code that allows users to view the comments within the forum posts. On this page, Snippets are also available, that provide a basic template; however, developers would have to manually populate the required fields.

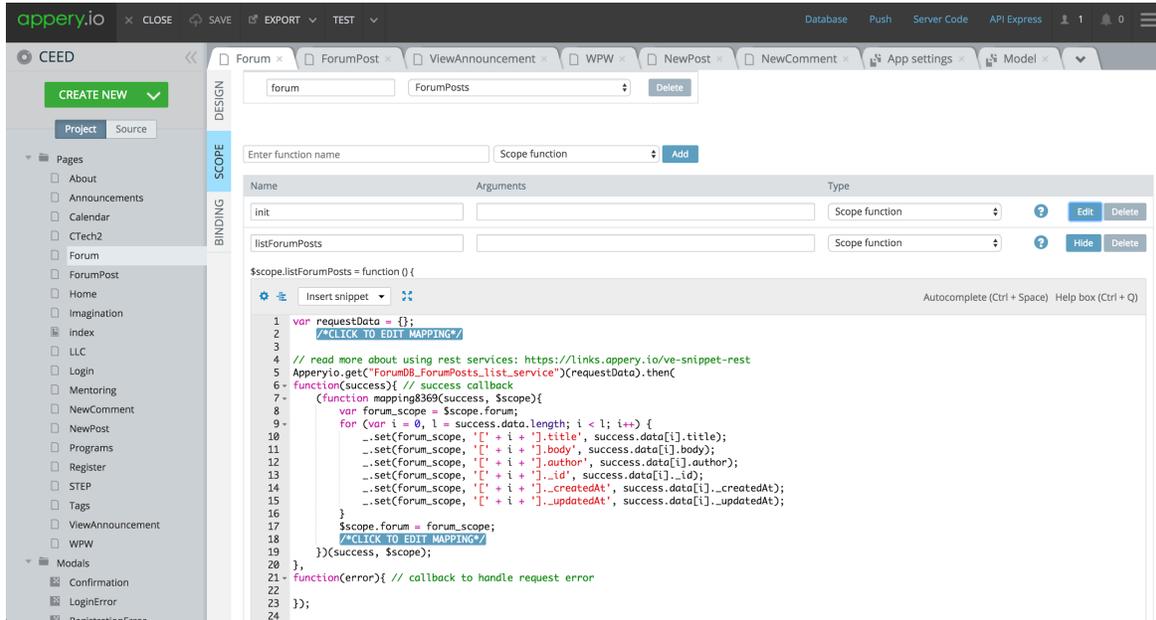


Figure 7.9 - List Forum Post Code

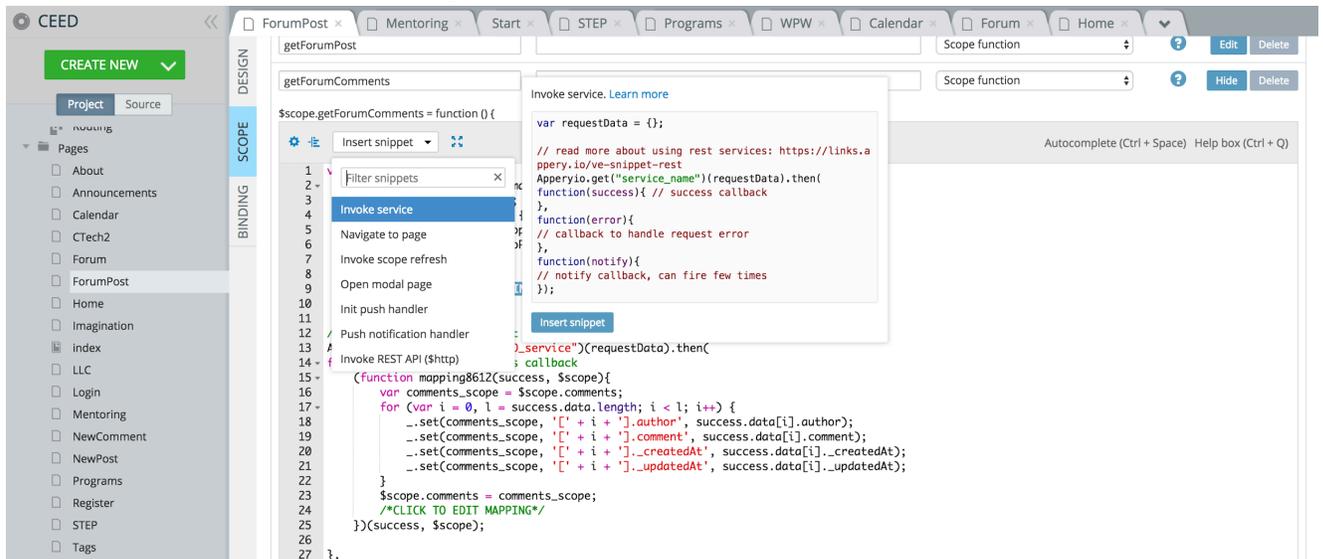


Figure 7.10 - Get Forum Comments Code

In order to make any major future development changes to the application, developers have the ability to write custom code and leverage these services in their application. They can create new database, server, or API services as shown in Figure 7.11. Clicking on any of the offered services could direct the developer to a

dialog box listing the service name and the past services already utilized in the application.

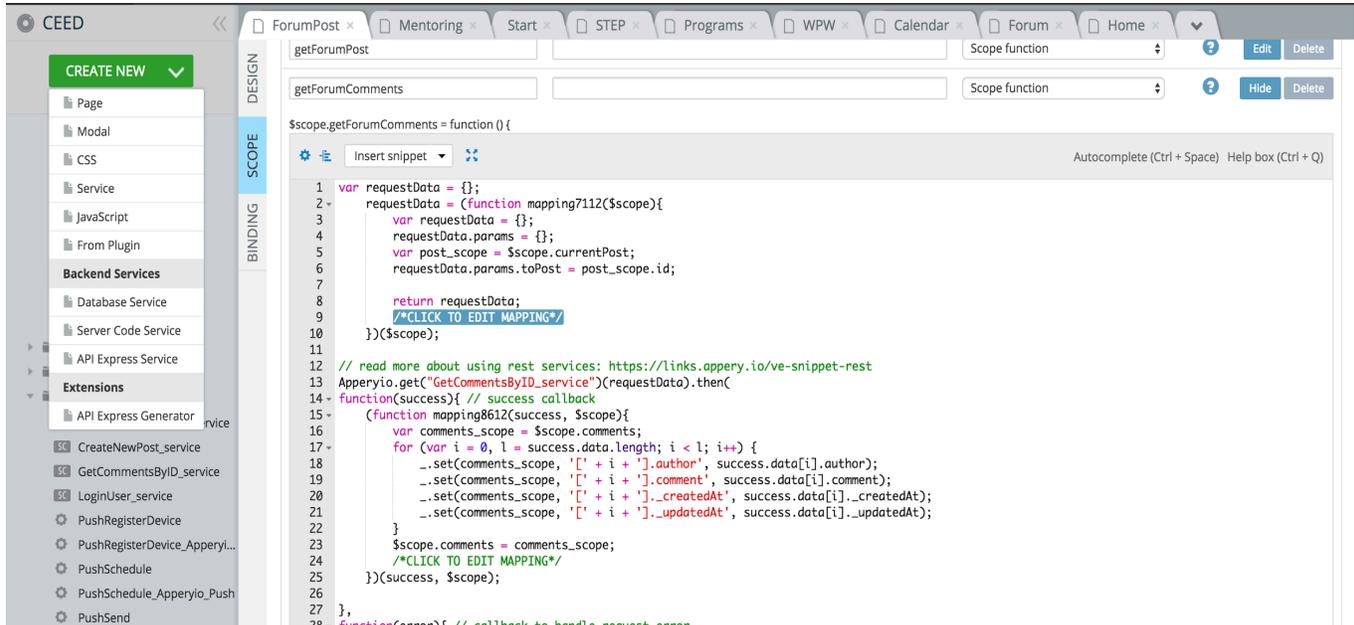


Figure 7.11 - Adding A New Functionality

The developer also can take advantage of the different plugins offered by Appery.io. Plugins are essentially functionalities that are already implemented for the developer. To add a plugin to the application, the developer would simply need to click on the “Create New” button and select the plugin option. As shown in Figure 7.12, a dialog box will appear with a list of the existing plugins available.

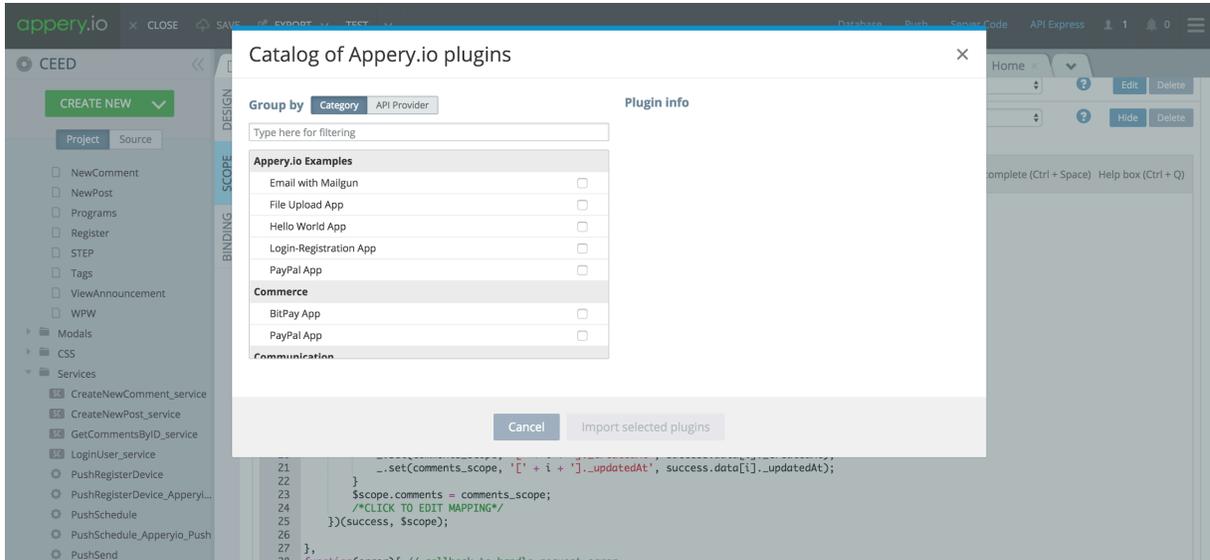


Figure 7.12 - Adding A New Plugin

The developer can also add custom variables and functions to the application. In order to add global functions and variables, the developer must navigate to the index page and select the “SCOPE” tab (Figure 7.13). Thereafter, they can fill out the required fields and add their custom code. Likewise, in order to add location variables, the developer can navigate to any of the other app pages and select the “SCOPE” tab. Global variables and functions are accessible from any page within the app whereas local ones are only accessible within the selected page.

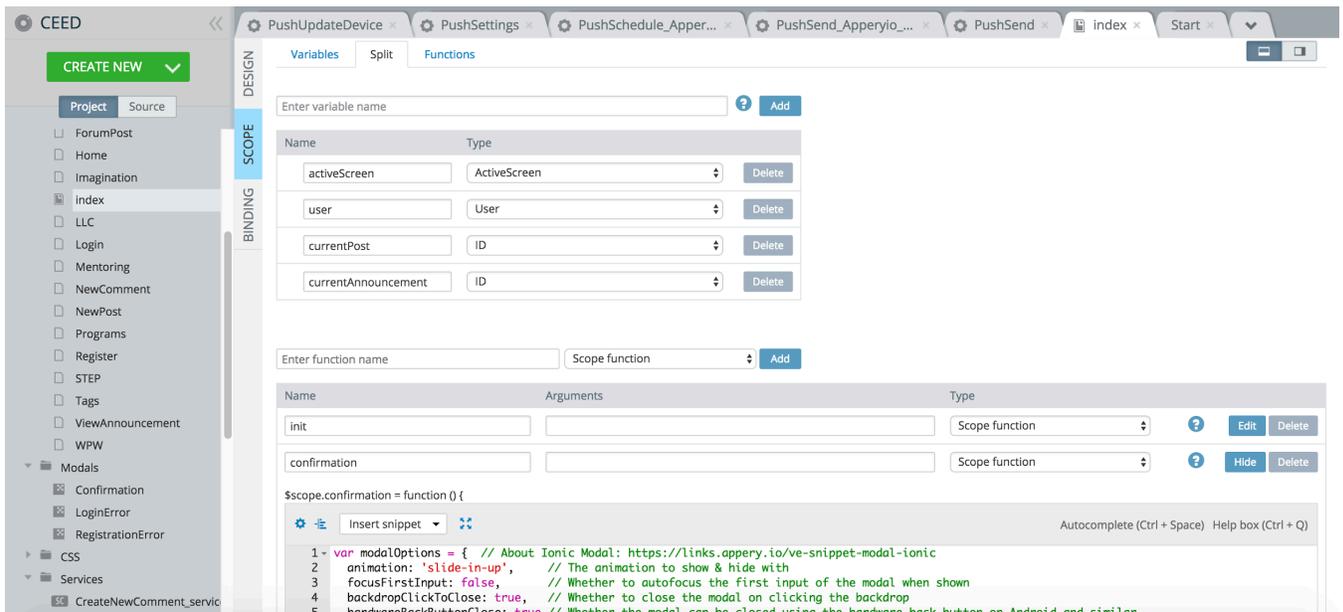


Figure 7.13 - Adding a Global Function/Variable

7.2.3 Database Management

Our application uses a NoSQL database to store information. User account information, announcements, and forums are all stored in the database. Figure 7.14 shows the database dashboard. All of the application databases can be managed from there. Additionally, this page also offers the ability to view the size of the database used and also the calls to the database.

The developer can create a new database simply by clicking the “Create new database” button on the top left. Doing so pops up a dialog to specify the database name. The developer would then have to specify the various tables that they want to add to the database.

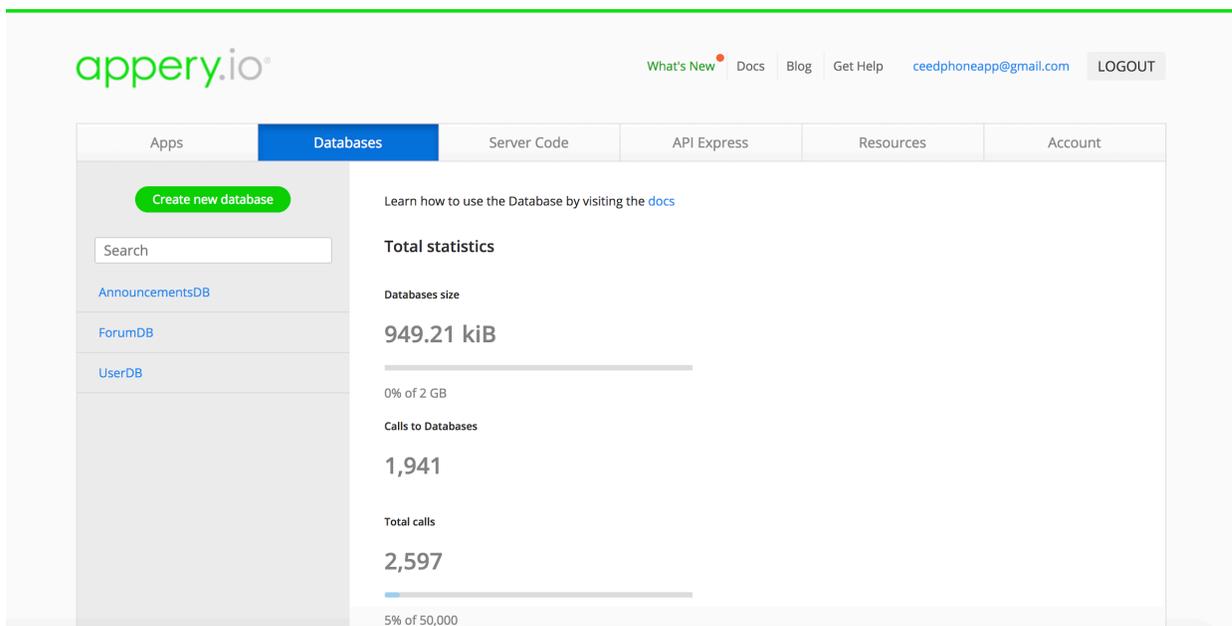


Figure 7.14 - Databases

The developer can click on the database names on the left in order to make edits to the already existing databases. Figure 7.15 shows the users database which contains the information about the users once they create a new account. As shown below, user account passwords are hidden from the developer and are secured safely using an encryption scheme on the Appery.io back end. The developer can also delete the database by navigating to the settings tab on the top right corner of Figure 7.15.

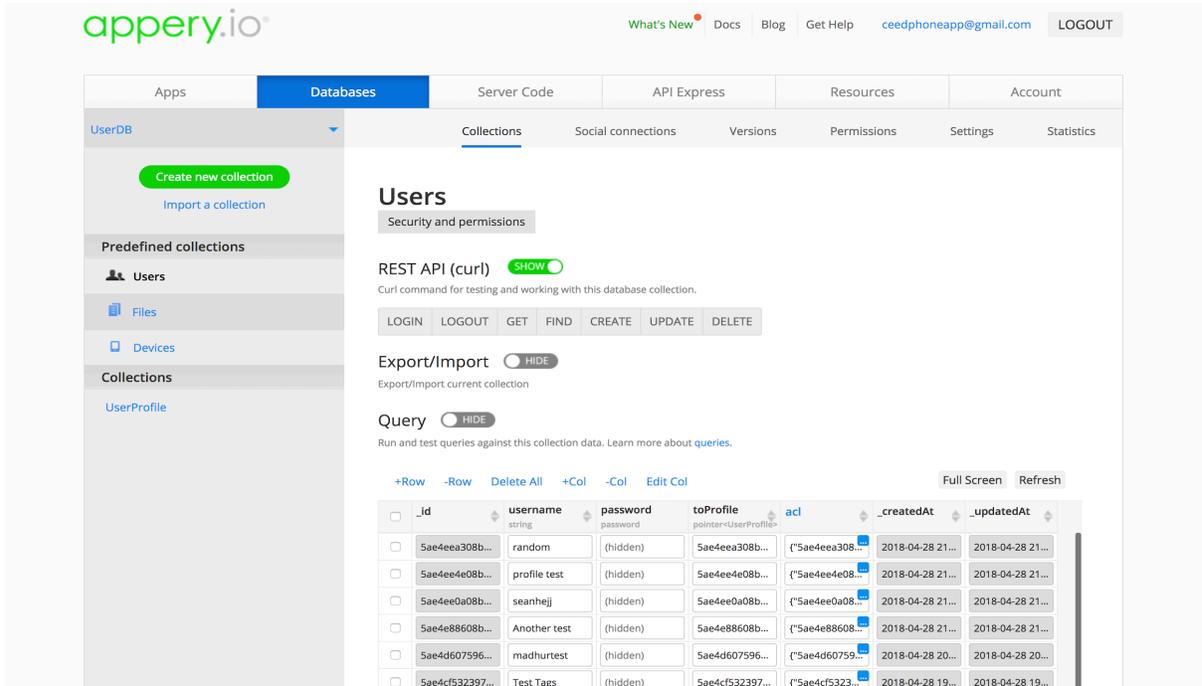


Figure 7.15 - Users Database

Figure 7.16 shows the forum database which stores all the users' forum posts. The developer has the ability to create, update, and delete any forum post of their choice simply by selecting the post and clicking on the “-Row” button. Additionally, the developer can delete all the forum posts by clicking the “Delete All” button. Depending on future client needs, more columns can be added to already existing databases.

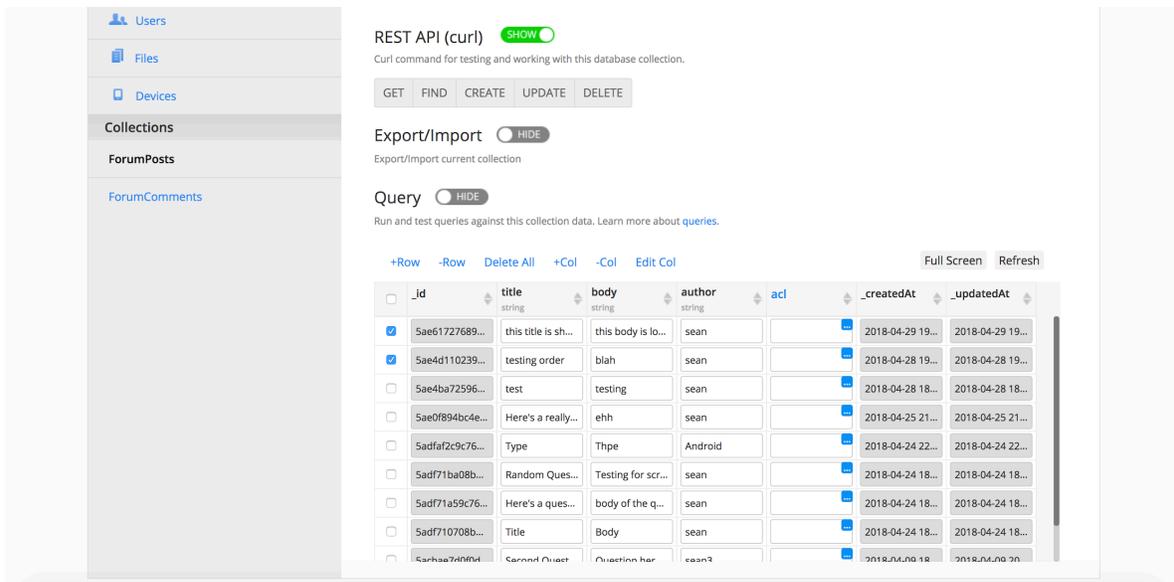


Figure 7.16 - Forum Database

7.2.4 Server Code

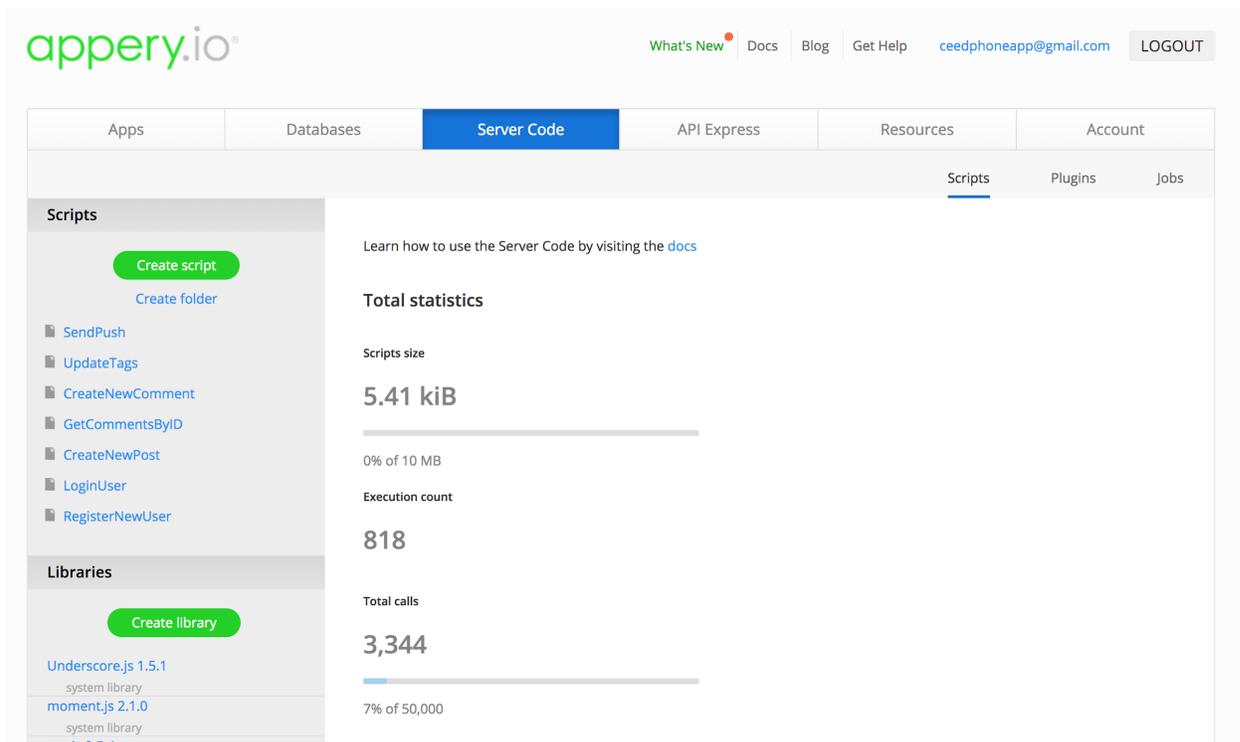


Figure 7.17 - Scripts

The scripts perform different functions within the app as described by their name in a list (Figure 7.17). Server Code allows you to write custom app code using JavaScript, that will run on the Appery.io server. The developer can create a new script by clicking on the “new script” button, and will be redirected to a JavaScript editor to make code changes. As shown in Figure 7.18 below, the platform allows you to add snippets of code previously written by Appery.io. A list of all the available snippets is in the right side of the code editor. Clicking the “insert” button would add code to your script file. Through this script functionality, we can send push notification messages, connect to the Appery.io database to retrieve or update data, and invoke third party REST API's^[11].

The screenshot shows the 'CreateNewPost' script editor. The script is as follows:

```

1 var title = request.get("title");
2 var body = request.get("body");
3 var author = request.get("author");
4
5 try {
6   if(title===="" || body===="" || author===="") {
7     throw "Bad request";
8   }
9
10  var result = Collection.createObject("Sacba2d00f0d311902589c58",
11    "ForumPosts", {"author":author, "title":title, "body":body});
12  response.success(result, "application/json");
13
14 } catch (error) {
15
16  response.error({
17    "message": "Could not create post."
18  }, 400);
19 }

```

The right sidebar contains a list of snippets:

- callRestApiGet: This snippet provides code to do a GET to any REST API.
- callRestApiPost: This snippet provides code to do a POST to any REST API.
- consoleLog: This snippet provide code to print debug info to trace.
- createObject: This snippet creates a new object in a specified database collection
- debugConsole: This snippet provides code to add debug information to Trace tab.
- deleteObject: This snippet provides code to delete an object by its id.
- multiDeleteObject

Figure 7.18 - CreateNewPost Script Editor

The script above allows users to create a new post and simply throws an error when bad requests are made, but otherwise should be successful. If unsuccessful, the user will be prompted for reasons, such as having a blank field.

The screenshot shows the 'RegisterNewUser' script editor. The script is as follows:

```

1 // This script requires creating a custom UserProfile collection. Then creating
2 // a 'user_profile' column in Users collection with Pointer type
3 // referencing UserProfile collection.
4 // See more in tutorial:
5 // https://links.apery.io/link-minifier/l/samples-login-registration-app
6
7 var databaseId = "Sac516830f0d310772dd1970";
8 var username = request.get("username");
9 var password = request.get("password");
10 var email = request.get("email");
11 var country = request.get("country");
12 var _status = request.get("status");
13 var gender = request.get("gender");
14
15
16 try {
17
18 // Register new user
19 var userLoginInfo = DatabaseUser.signUp(databaseId, {
20   "username": username,
21   "password": password
22 });
23
24
25 // Save additional registration information into 'UserProfile' custom
26 // collection
27 var userProfileInfo = Collection.createObject(databaseId, "UserProfile", {
28   "email": email,
29   "country": country,
30   "status": _status,
31   "gender": gender
32 }, userLoginInfo.sessionToken);
33
34
35 // Link Users and UserProfile collection
36 DatabaseUser.update(databaseId, userLoginInfo._id, {

```

The right sidebar shows the 'Request parameters' section for a POST method:

- username: seanhejj
- password: testvnmndc
- email: sean003@vt.edu
- country: USA
- status: VT Fourth Year

The 'Request body' section shows the MIME type set to 'Text'.

Figure 7.19 - RegisterNewUser Script Parameters

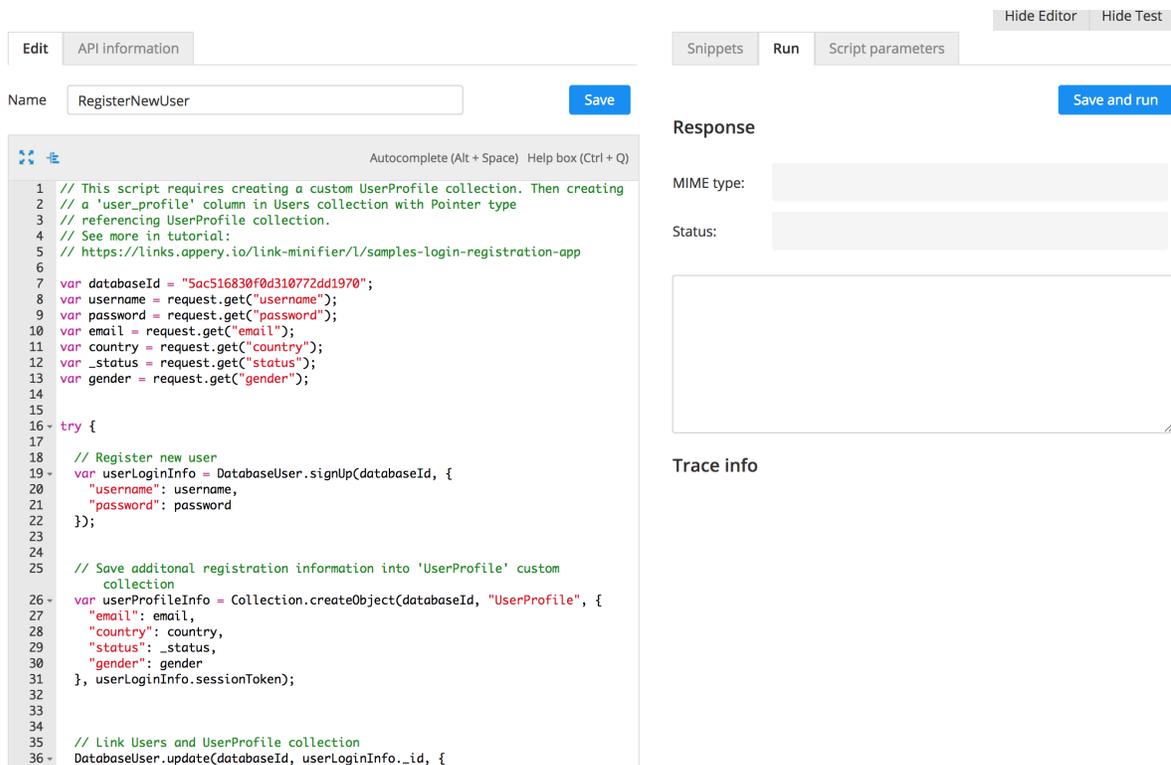


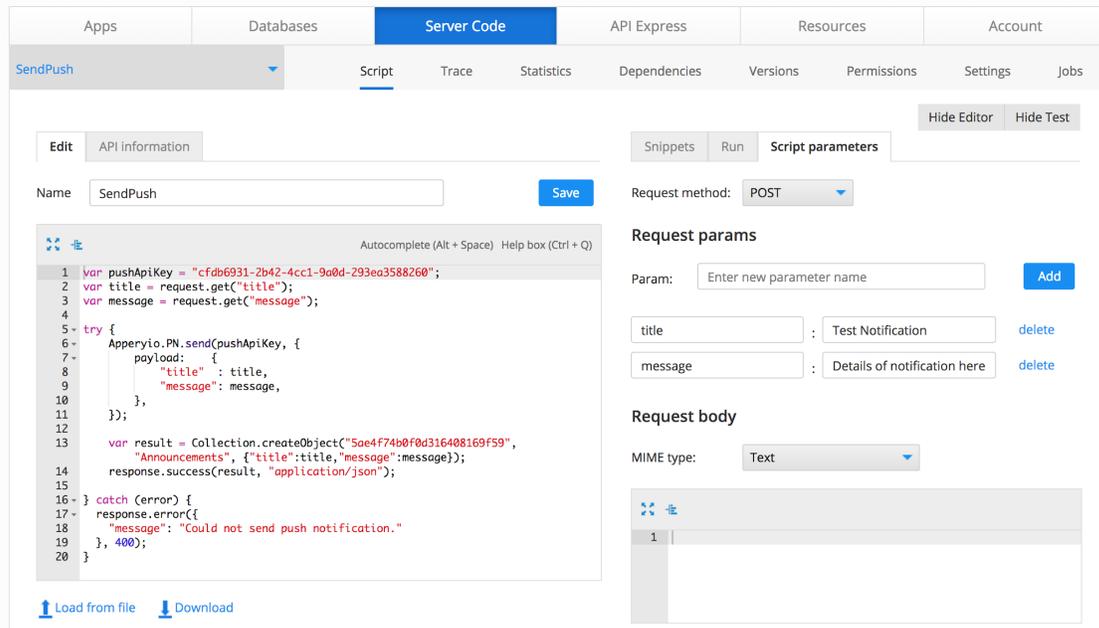
Figure 7.20 - RegisterNewUser Testing

In Figure 7.19, the script requests (user entered) information for several fields in the “Script parameters” tab to generate verifiable credentials, with a username and password, which are used for future logins. The ancillary information such as status and gender help the user receive information that is specifically catered to the demographic they belong in. In this screenshot, we have created a test account which we can execute as seen in Figure 7.20 beneath the “Run” tab where we then click on the “Save and run” button to determine if this request is valid.

7.2.5 Push Notifications

Push notifications are alerts on a mobile device commonly used to announce important information. In order to send notifications to the users, the developer would need to navigate to the “Server Code” tab on the main Appery.io dashboard. Thereafter, they would need to select the “SendPush” script from the left menu as shown in Figure 7.17. Thereafter, the developer will be directed to the screen shown in Figure 7.21. The code on the left side of the screen is the code that sends the push notifications. The developer does not need to change any code. The developer has to navigate to the “Script Parameters” tab on the right side of the screen and fill out the push notification information. The title of the notification will be filled, in the box next to the “title” box. The notification details will be filled, in the box next to “message”. To

send the push notification, navigate to the “Run” tab A and click the “Save and run” button. A JSON file of a timestamp and ID are generated if successfully sent (Figure 7.22)



Figure

7.21 - SendPush Script Parameters

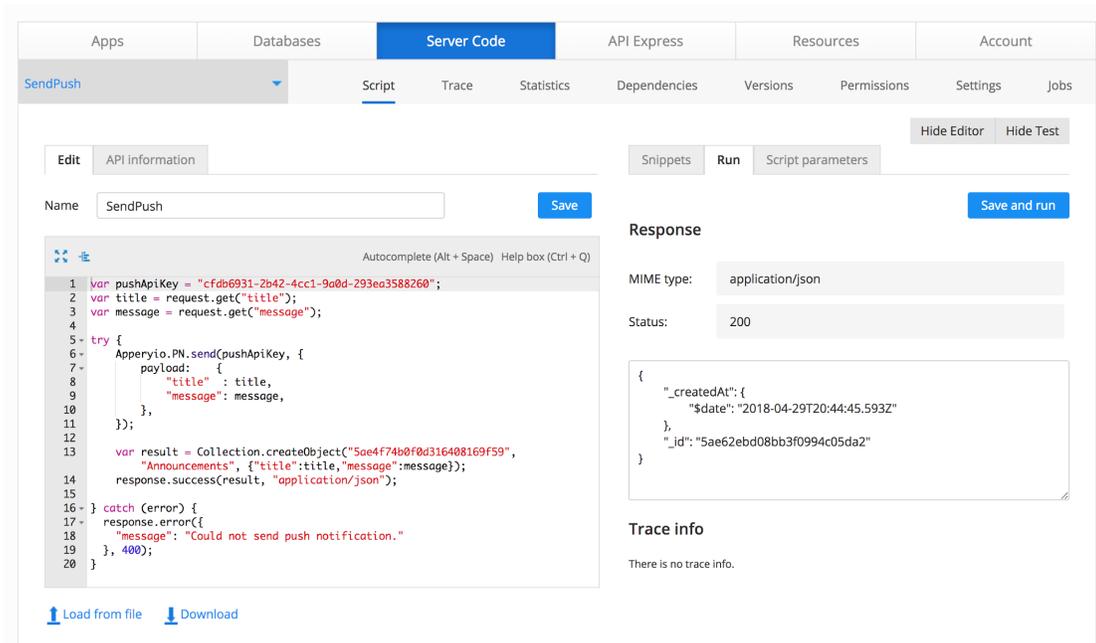


Figure 7.22 - SendPush Script Run

8.0 Lessons Learned

8.1 Timeline/Schedule

The team members drafted a comprehensive timeline at the beginning of the project and presented it to the client. The team was initially able to follow the timeline and make progress on the project. However, we later encountered various roadblocks which hindered the project's progress. Our goals and dates are listed on the timeline to allow progress to be assessed at each given date and ramp up or slow down production accordingly. Figure 8.1 below shows our initial timeline.

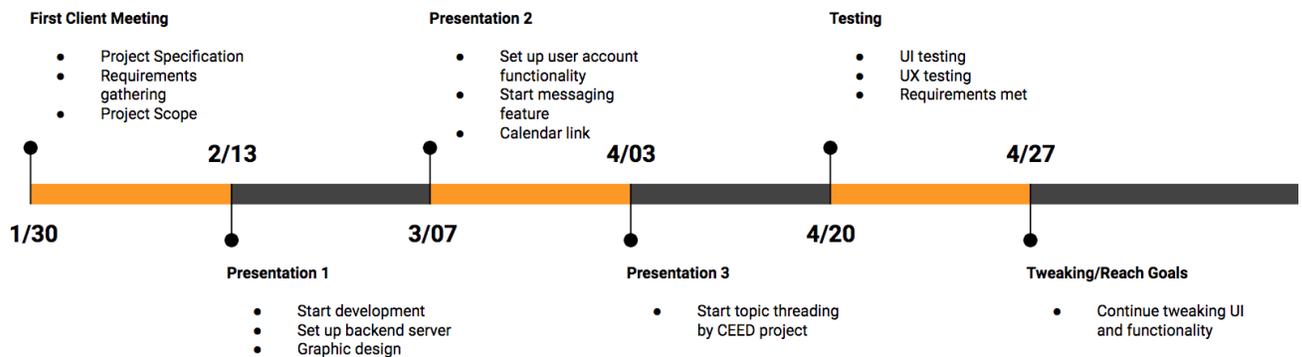


Figure 8.1 - Project Timeline

Due to the roadblocks encountered, it was difficult to follow the above timeline, and some schedule changes were inevitable. The team was approved to use Appery.io as our development platform at the beginning of the month of April. The team had to adapt to the updated schedule and complete all the software development phases within a month.

8.2 Problems and Solutions

8.2.1 Complexity

This application is relatively complex as our team needs to implement a forum functionality where users will be able to interact with one another. The most complex aspect of this application is the fact that it needs to be supported cross platform. Due to this, we are somewhat limited to the technologies that we can use in our solution. Moreover, this may also cause more work for the team as we will need to create two separate project solution folders. Lastly, push notifications are also difficult to

implement as they are handled differently based on the platform being used^[6]. The team lacked mobile app development experience which had a steeper learning curve than we had initially anticipated.

In order to combat the complexity of our problem, we consulted various online, third party technologies and platforms that allowed streamlined cross-platform development, hence alleviating many of our concerns. These platforms handle many of the issues that would arise during cross-platform development, albeit at a monthly cost. This was also beneficial for the client as it makes future maintenance much easier to carry out and is clear to others within the client's department.

8.2.2 Communication

Communicating with our client was a big hurdle and point of contention for this project. Our group was advised by the instructor to do weekly meetings to discuss the progress on the project for that week and ask questions. Unfortunately, many emails were not responded to in a timely manner, which made these meetings not possible. We ended up only meeting twice in the first nine weeks. This lack of communication led to setbacks in development due to not having client decisions made available when the group was ready.

Had we known earlier what the client's budget and specific requirements for the project were, we may have had a more filled-out finished project. Despite these challenges, the group still managed to get important meetings set up later in the project on short notice and worked with the client to iron out the details while there was still time left to develop.

8.2.3 Learning Curve

Since no member of the team had prior experience with the Appery.io platform, there was a significant learning curve that had to be overcome to effectively use the software. Appery.io offered many tutorials and walkthroughs for the majority of their features. This allowed the team to follow along while practicing how to use the product which sped up the learning process significantly. In some cases, the documentation was out of date or missing features. This led to many time sinks as the team spent time trying to discover how the software was intended to work.

A few features, specifically push notifications, did not work at all for the team, even after many hours and ardently following the tutorials. As a result, we were forced to email and converse with the Appery.io development team to try and get a better understanding of their system and how to implement it within our vision. This back-and-forth process ended up taking more time than we originally anticipated.

8.3 Future Work

This project has a lot of room for growth. The user base is very large, ranging from high school students to parents. The mobile application developed in this project is very limited and can be expanded significantly to serve other needs of the organization. Given the short timeframe of the project, the team's primary goal was to create a minimum viable product in order to serve basic organization needs. Thereafter, the client can further evaluate these needs and expand the application appropriately.

The client has expressed the need for creating different pools of users which will help them reach out to a certain group of their user base and send them information which is aligned with their chosen preferences. The client has also expressed the need of categorizing the forum page and filtering it in a manner that users are only shown forum posts based on the preferences they choose while logging into the app for the first time. The process of searching for and deleting inappropriate posts from the forum can be automated in the future. The preference tag functionality that we have implemented in our application is limited and can be further improved. Currently, the user has to manually change their preference tags every year as they progress. This process can be automated such that updating the tags is handled by the development platform. Additionally, the application can be improved by updating the CSS of the front-end templates. Some future work can include aligning the app templates to Virginia Tech's color scheme.

Future work on the client's end includes learning and familiarizing themselves with the Appery.io platform. The platform has a learning curve discussed in the previous section and requires some research and practice. Subsequently, they will be able to easily update/maintain it in the future using cross-platform development tools. They can also export the project source code and switch to a different cost-effective environment. There are other platforms that offer services similar to Appery.io at a relatively cheaper price. The client can re-evaluate their needs and choose a platform that offers services satisfying their updates needs.

Lastly, the client will need to publish these apps on the Apple and Android app stores. Publishing an app on the Apple app store requires having an Apple developer membership which costs \$100 annually. Publishing the app on the Android app store is cheaper and easier. Virginia Tech probably has memberships for both these platforms. Please refer to references 9 and 10 for more information on how to post this application on the app store

9.0 Acknowledgements

We would like to extend our sincere gratitude and appreciation for our client, Dr. Watford, as well as all involved parties, particularly Kristin Morrill for regularly communicating and meeting with us. We would also like to acknowledge Dr. Edward Fox for helping us along the way by pointing us in the right direction when we encountered difficulties as well as giving us useful feedback on our project progressed at various stages in the semester. Lastly, we would like to thank CEED for providing us the opportunity to develop this application for them.

Sponsor: CS4624: Multimedia, Hypertext, and Information Access

- Dr. Edward A. Fox
 - Email: fox@vt.edu

Client: Center for Enhancement of Engineering Diversity (CEED)

- Dr. Bevelee Watford
 - Email: deuce@vt.edu
- Karis Boyd-Sinkler
 - Email: karisb@vt.edu
- Kristin Morrill
 - Email: gracou@vt.edu
- Chandler Raynes - IT Specialist
 - Email: craynes@vt.edu

10.0 References

- [1] "Pricing." *Appery.io*. Appery, LLC. Web. <https://appery.io/pricing/>. Accessed 29 Apr. 2018.
- [2] "Documentation." *Appery.io*. Appery, LLC. Web. <https://docs.appery.io/docs/>. Accessed 28 April 2018.
- [3] "Amazon EC2 Pricing." *Amazon Web Services*. Amazon. Web. <https://aws.amazon.com/ec2/pricing/on-demand/>. Accessed 28 April 2018.
- [4] "Pricing." *AppMakr*. AppMakr. Web. <https://www.appmakr.com/pricing/>. Accessed 28 April 2018.
- [5] "Center for Enhancement of Engineering Diversity-CEED." *Virginia Tech College of Engineering*. Virginia Tech. Web. 28 April. 2018. <https://eng.vt.edu/ceed.html>. Accessed 28 April 2018.
- [6] "Getting Started With Push Notifications." *Appery.io*. Appery, LLC. Web. <https://devcenter2.appery.io/tutorials/jqm/getting-started-with-push-notifications/>. Accessed 28 April 2018.
- [7] Balci, Osman. "Dr. Balci's Software Life Cycle." Web. 28 April. 2018. CS3714 *iOS Mobile Software*. Virginia Tech. Virginia Tech, <https://manta.cs.vt.edu/cs3714/StudentsOnly/Handouts/SELifeCycle.html>. Accessed 28 April 2018.
- [8] Thomas, Nathan. "How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website." Usability Geek, 4 Aug. 2015, usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/. Accessed 28 April 2018.
- [9] Apple Inc. "Certificates." Certificates - Support - Apple Developer, developer.apple.com/support/certificates/.
- [10] "Publish Your App | Android Developers." Android Developers, developer.android.com/studio/publish/.
- [11] "Server Code Overview." *Appery.io*. Appery, LLC. Web. <https://docs.appery.io/docs/servercode-overview>. Accessed 29 April 2018.
- [12] "Weinre Debugger." *Appery.io*. Appery, LLC. Web. <https://docs.appery.io/docs/weinre-debugger>. Accessed 29 April 2018.