James Schreiner, Collin Youngwirth-Hupka, David An
Consumer Energy Education Foundation
Project: Energy Day Quest
6/18/19

Table of Contents

Introduction

The Consumer Energy Education Foundation (CEEF) looks to create programs and activities designed to motivate students to follow a STEM path and to improve understanding of career opportunities in STEM and energy industries.

CEEF has been hosting "Energy Day" for the last 4 years in both Houston and Denver. Energy Day has used phone applications in the past, but the software was not easily editable and kids did not find it enjoyable.

Therefore, CEEF asked our team to develop a mobile app from scratch that allows users to login and store the basic user information, create a points system using attendee reviews, create links to CEEF social media, and a map for users.

Requirements

Functional Requirements:

- User Registration
  - Asks for name, age, and contact info (phone/email)
  - Information is output into a JSON file (can be converted to .csv)
  - Information is stored in a server that individual user accounts access throughout use of the app
- Point System
  - Users rate different stands, which earns them points
  - Back end leaderboard for the company to distribute prizes (Not accessible to users)
- Map
  - Map image can be uploaded by administrators, allowing the map to be updated
- Social Media Links
  - CEEF Facebook and Twitter links located within the app so that students and parents can leave public reviews of the event

Non-Functional Requirements:

- The program uses the Xcode IDE, which is the developing environment for macOS to create applications for Apple Operating Systems.
- The program is an iOS mobile application made using the Swift 5 language
- Includes documentation of code for the client for potential future changes and maintenance of the application
- Interface
  - Interface is simple so that younger children can understand and interact with it
  - Interface is visually appealing

System Architecture

Firebase:

- Integration with the app
- Information output to file
- Database maintenance

Firebase is an online Google service that functions as a database for our application. The service can connect to an application that gives it commands to read and write different information to the database. The implementation of this service was simple, as it had the libraries required for it to be integrated into the app. These libraries were also specifically made for the coding language and IDE that were chosen for the project. To push from the Swift language to Firebase, a unique ID is assigned to the app, which allows the developer to reference the database directly, including any of its children roots. The app inputs information into the database by finding the proper pointer node and assigning its value directly. This allows the database to be queried so, for example, the app can check if the entered username and password are correct within the database. This can also be applied to the point system and rating systems.

During development, it was decided that Firebase is the best option for handling data from the application. Firebase is able to reliably output the information from the database into a readable file (.json), which can then be converted into other file types. This allows CEEF to use the data they collect in almost any form that they need, making it easier to read feedback and improve their event.

There is a section in our documentation regarding the use of the Firebase database. It is simple in that it explains how to download the data off of the database and convert it to a different file type.

Password and account management:

- Differentiate users
- User privacy
- Password recovery

Finding an easy way to differentiate users became a significant logistical issue. This was solved by restricting the username to be unique. This also made it easier to deal with large numbers of users within the database.

We did not want to require that users put in any information they didn't feel comfortable sharing. To work around this problem, there is a location in the app that asks to make sure users are comfortable with giving this information before continuing. If they are, contact information is stored into the database, and they become eligible for prizes. If not, the user still gets an account, but they are not eligible for prizes.

If a user forgets their password, password recovery is an option if they provide their email. If a user did not provide email during account registration, they are unable to recover their password, however, it is a viable option to create a new account.

Data sent to server from users:

- The method of communication between client and server
- Maximum amount of data that can be sent total/at a time

When it comes to communication between Firebase and the application, information is sent regarding the users' username, password, role, personal information, and points. The username and password can be used multiple times to login. After registration, the role and personal information is stored within the database for CEEF to use, with no additional purpose within the app. Finally, the user's point count is the only information in the database that may change as they are earned throughout the event.

Information on the different sponsors at the event is also sent, including their name and ratings. Similarly with the users, the name will not change, but the ratings each sponsor can receive will change during the event.

Being able to send this information to Firebase through the app is simple as all it requires is connecting the app to Firebase and downloading all the necessary libraries. These libraries include query methods to fetch information from the database while other methods allow the app to write to the database. Firebase has a max number of writes that can be performed on the database before payment is required. Due to the way the application needs to work and the number of users that attend the event, the app does not attempt to minimize the number of database writes because it risks losing the user data. The amount of data stored on the database is not an issue because storing strings and integers required minimal storage.

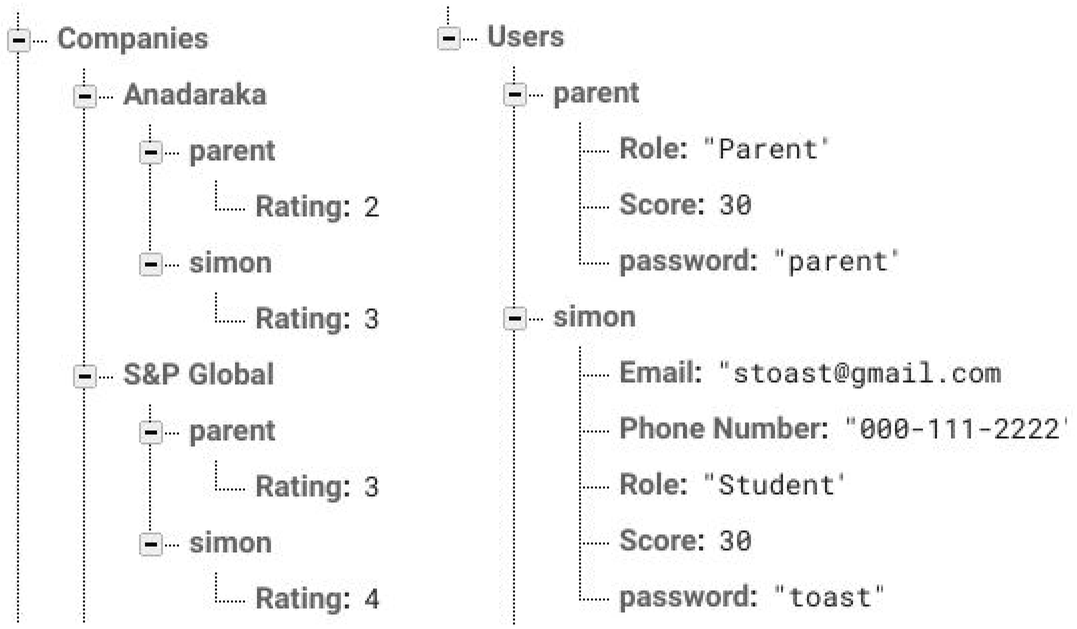
Social media uploads:

- Monitoring uploads
- Linking to websites

There are potential issues with allowing users to upload pictures and comments about the event to social media. These have all been discussed with the client to provide solutions to implementing this service. CEEF has employees dedicated to filtering out online posts throughout the event to prevent any inappropriate uploads from appearing on their social media. With this figured out, the users themselves can access those social media websites with an in-app link. This system is also used to let users follow a link to an online survey where they can provide feedback regarding the event.

Technical Design

The bulk of this project can be separated into two parts: the Firebase database, and the application itself. The Firebase database is the main source of data storage for the application. It stores user information and sponsor information, which can be accessed by both the application itself and by members of CEEF (Figure 1).



```
Companies
    Anadaraka
        parent
            Rating: 2
        simon
            Rating: 3
    S&P Global
        parent
            Rating: 3
        simon
            Rating: 4

Users
    parent
        Role: "Parent'
        Score: 30
        password: "parent'
    simon
        Email: "stoast@gmail.com
        Phone Number: "000-111-2222'
        Role: "Student'
        Score: 30
        password: "toast"
```

Figure 1: How user and sponsor information is stored within database

Storyboards are where the screens of the application are built, and allow software engineers to build screens either visually using the Interface Builder or using code similar to how Java GUI is created. Information is then passed between different storyboards within the app to make sure that each one has the information it needs to perform its function. For example, the username of the person currently logged into the app needs to make sure they stay logged in, to make sure they can log out, to increase their score for the event, and to make sure the user is unable to get points for rating the same sponsor booth more than once.

Information is obtained from the database using queries. Many times, these queries require other information about the current user or data in the database, so queries are constantly made to update and provide that information. They also return null if the child node does not exist, which allows an if-else statement to check if a username is already in use.

When first opening the application, the user views a title page. By tapping the enter button, the user is then taken to the login page (Figure 2). It is here that a user registers for a new account, recovers a lost password, or signs in with an existing account.

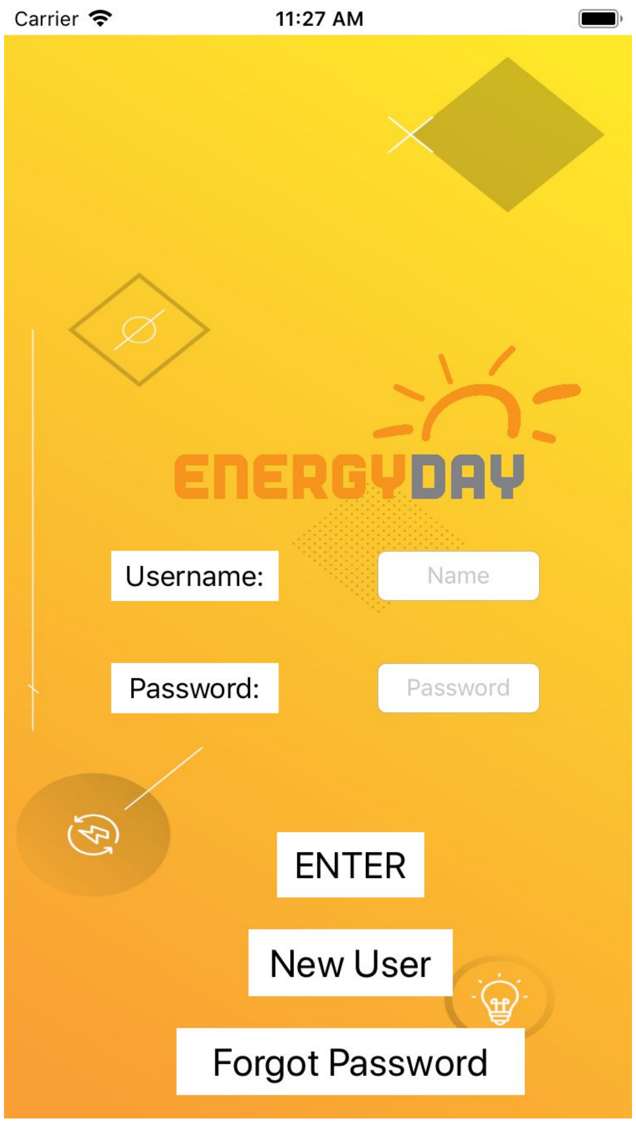
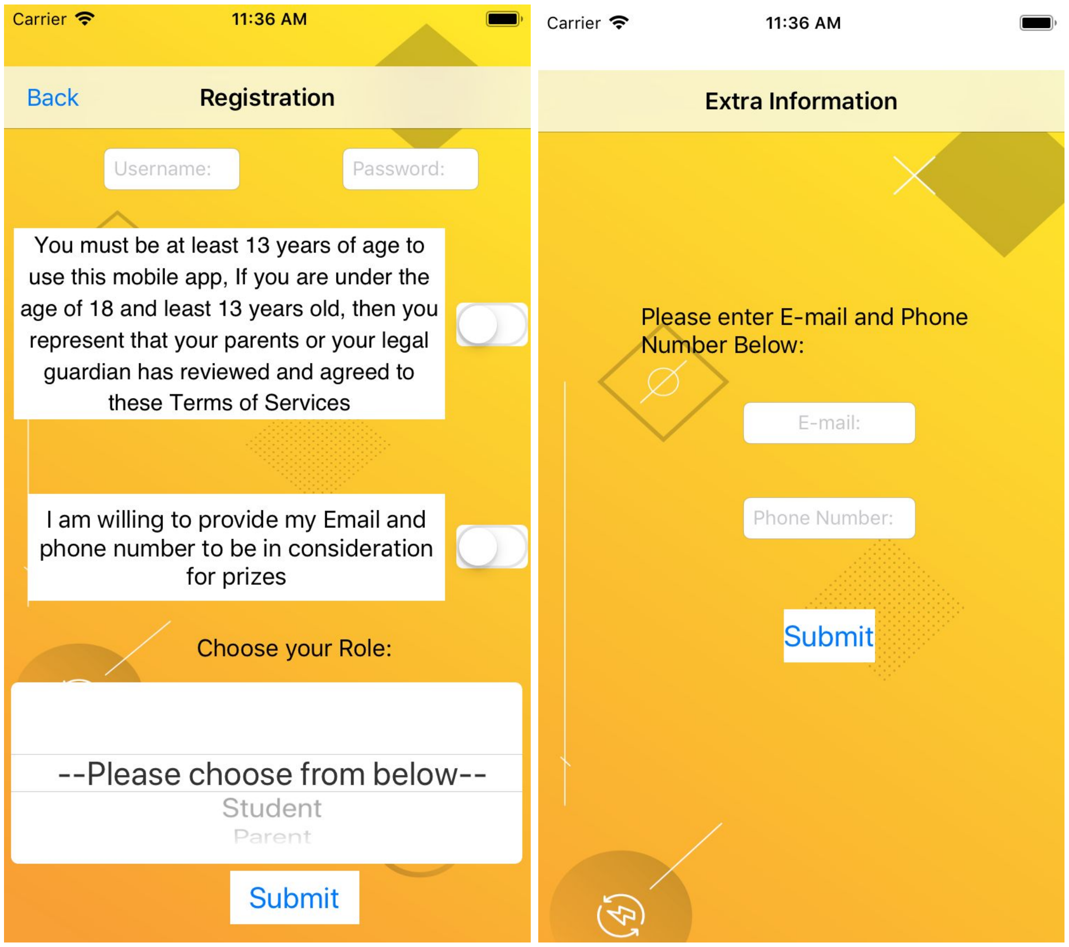


Figure 2: The login page for the app

The user registration button takes you to a page that asks for a new username and password, however, it also asks you if you are willing to provide contact information to be eligible for a prize. Figure 3 below shows how a user must be over the age of 13 with approval from an adult. The reasoning for this was that the Children's Online Privacy Protection Rule (COPPA) legally restricts the information that apps use about minors, and risking legal trouble was not a risk that our client felt was worth handling.

The role wheel above the submit button allows the app to store why people were attending the event. Our client uses this information to improve the app in following years by following the demographics of the users.



You must be at least 13 years of age to use this mobile app, If you are under the age of 18 and least 13 years old, then you represent that your parents or your legal guardian has reviewed and agreed to these Terms of Services

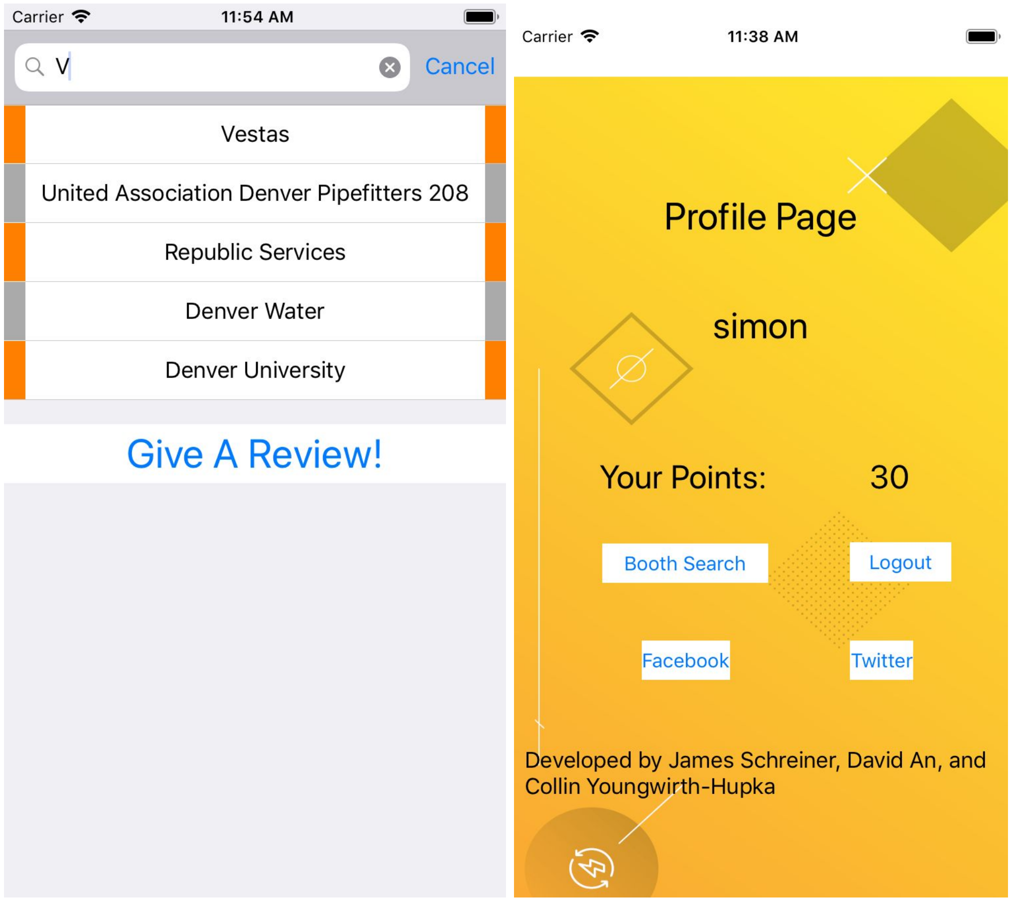
I am willing to provide my Email and phone number to be in consideration for prizes

Please enter E-mail and Phone Number Below:

Figure 3: The new account registration and personal information pages

By agreeing to this, the users are taken to another page which prompts for their email and phone number in Figure 3 above. Instead of assuming that all users are willing to give email and phone number, the app gives the option to the user to give this information, but functions with or without the personal information. All four of these entered values are sent to the database to be stored and the user is taken back to the login page.

If a user forgets their password, then they can use the "Forgot Password" button to recover it. They must enter their username and email to be able to recover the password. If a user did not opt to give their email, then they are unable to recover their password. Despite this, it is fine for them to create a new account, as the main reason their contact information is used is to notify if they have won a prize or not. After getting their password back, a user can return to the main login page to enter the application.



Vestas
United Association Denver Pipefitters 208
Republic Services
Denver Water
Denver University

Developed by James Schreiner, David An, and Collin Youngwirth-Hupka

Figure 4: The booth search and booth rating pages

Once a user successfully logs in, they are presented with a list of the participating sponsors with booths at the event, as shown in figure 4 on the left. They can use the provided search function to find any specific ones they want, or they scroll through the list and choose one. The booth search screen was created using a table view controller, which allows dynamic prototypes of the cells, which was useful since at the time of development, the number of companies and their names is unknown. The search bar is connected to the data of the table, and reloads the table as soon as a user begins entering a string. Once a sponsor has been chosen, users are taken to a page that allows them to review that sponsor's booth out of five stars, shown in figure 4 on the right. This rating can be submitted, the user gains points, and the sponsor has the user rating assigned, all of which is saved on the database. The profile pages updates to show the unique username with their points total displayed by reading from the Firebase Database. The social media buttons below are links to the websites, which will open in the phone's browser. After submitting a review, the user is taken back to the list of sponsors to repeat this process (Appendix: Product Storyboards).

Other functionalities in the application include a map of the event, a survey link, and a logout feature. The latter three can be found on the user's profile page. This page also provides information the user with their username and total score from event participation. The map can be accessed from a tab. When accessed, the button takes the user to an image of the event map to help them find booths during Energy Day. Another button returns the user back to the previous screen with the booths.

One interesting challenge for this project was how to design an easily updatable application. Information about what companies are attending and where they are located at the event is unknown, because Energy Day occurs later in the year, which means it needs to be added into the application manually once the event is organized. Everything within the application has to be able to take this information, read it, and use it correctly since it changes from year to year. For example, each year the layout of the event will be different, therefore, the map of the event must also change to reflect this. To update the map, simply upload a new image file in the correct place. This process is similar when it comes to links and sponsors participating in Energy Day. The Documentation written for updating the app is located in the Appendix. This simplified the process and makes it easy to change images and text without extensive coding knowledge.

The final non-functional requirement was making our interface user-friendly. This meant our screens were simple and relied on buttons for user interaction. The white color of the buttons contrasts with the orange background color, making them easy to see. Overall, the flow of the app is simple and we made it difficult to become lost within the app, which fulfills the requirements of a user interface that children can comprehend.

Quality Assurance Plan

Xcode UI:

- Verified button functionality and text field input by recording UI interactions into source code. Buttons correctly transition between different storyboards fluidly and go to appropriate locations.
- Risks: Application does not properly send users to correct destinations based on selection of options. Also, users find the UI to be difficult to use and are unable to find relevant information, such as username, password recovery, cumulative points, etc. This makes the application diminish the effect of Energy Day for participants.
- Testing: Using a simulation provided by Xcode, this provided a way to go into the application and pretend to be a user with an iPhone on the computer screen that fully functions as a normal phone.
    - Registration Test: First entering the application, select the create new user button. From here, the tester successfully created the account with a unique username and password. The tester received a confirmation back from the server by advancing to the next screen. This tested to ensure users can register for the application in order to be eligible for points.
    - First Time Login Test: Tester entered an existing user and password created on the server. This allowed the user to enter the app and use it. The tester had 0 points initially and 0 visited booths. This test ensured that newly registered users start with the correct number of points and can only earn points for their account.
    - Forget Password Test: Tester (who has an account already) selected the “forget password” option. This option requested username and email, and displayed the correct password when the email and username matched an account in the

database. This test ensured users can still access our application, even if they forget their password.

Social Media:

- In order to further promote Energy Day, the client has asked that the application implement a way for users to interact with the CEEF's Facebook and Instagram pages. To fulfill this requirement, the application provided links to these social media outlets. Users can upload images and comments about Energy Day.
- Risks: The accessibility of the links means potential users could have uploaded inappropriate comments and images to the CEEF social media pages. This risked reflecting a skewed image of CEEF and defeated the purpose of encouraging more participants at Energy Day. However, a designated CEEF employee monitors the associated social media outlets to act as a filter for any inappropriate images and comments.
- Testing: We tested the application by clicking the links through the simulation, and found that users are sent to the correct pages in the phone's browser.

Firebase:

- The large amount of information we needed to collect and store required us to use a database, so we decided to use Google Firebase. This service allowed us to hold information about the users and allowed our client to analyze general trends in age and which booths had higher rating than others.
- Risks: We needed to find a database to hold our data. No one on the team has ever used Firebase in the past, and this meant we all needed to quickly learn it. After we worked on Firebase in the last couple of weeks, we have become more comfortable and confident with including user authentication and handling the amount of users.
- Testing: Testing was performed to check the limits of Firebase and make sure it is secure enough to store the information it holds.
    - Storage Capacity Test: Increments of large samples of data were pushed onto the Firebase. The analytics included in the Firebase website showed that our app requires minimal storage, since we only stored strings and integers. The test determined that the free plan for Firebase offered enough storage for the expected number of users.
    - Security Test: The tester attempted to view data stored into the Firebase using the Database Rules that limited data accessibility. The rules in place prevented users from accessing data outside of what they should be reading and writing to. Data is secure behind Firebase Authentication and the information edited by users is expected.

Results

Performance testing results:

- The application runs smoothly and performs at a good speed
    - During development, we made sure to keep a close eye on the performance. The application did not crash the app and transitions are smooth, meaning the app did not load too much data at a time.
- Firebase connectivity works daily - no loss of information
    - The other performance issue we fixed was losing the connection to Firebase through the app. Making sure the application can communicate updates to Firebase about info that changes during the event is a basic functionality, and we found how to connect the online database to the code. This allowed for the app to connect to Firebase whenever the application was built and run.

Summary of testing:

- A few cases occurred that caused us to realize parts of the authentication system that needed to be fixed:
    - While testing our registration, we tried to enter in duplicate usernames. This caused any information from the previous username to be overwritten, so we needed to add a check to make sure that this was not possible.
    - Based on certain user inputs, the segues between storyboards in the app needed to change to accommodate whatever information the user may have given.
- Data can be accessed and stored in Firebase
    - The application is fully able to both upload and retrieve data that is in the Firebase database. Additionally, any users with direct access to the Firebase database (admins) are able to download all of the data in a JSON file.
- In-app links take users to the correct social media websites
    - The links in the application successfully take the user to the correct website. These links are also easy to change, so CEEF can change them each year, or update them to other sites.

Features that could not be implemented:

- In-App Notification System: The client wanted the application to have a way to contact people through it as a stretch goal. This would be used to notify users of scholarships being given out or offers at nearby food trucks.
- Sponsorship Level: It was requested that the different sponsors at the event are given different sponsor levels (platinum, gold, silver, bronze) based on their donations. This would give them certain benefits within the application.

Future work and potential updates:

- **Google Maps GPS implementation with Energy Day map**
    - The map function of the Energy Day map would utilize GPS tracking to help users find locations throughout the event. These include food trucks, activity stations, and bathrooms.
- **Additional locations for Energy Day**
    - In case CEEF decides to expand their Energy Day event into cities other than Houston and Denver, there needs to be an update to support these new locations for the event. Currently, Firebase does not differentiate between the different Energy Day data, and has to be manually pulled from the server after each separate Energy Day.
- **App development for android systems**
    - The client wants the application for both Android and iOS eventually, so that more people can utilize the app. This can be done by finding a framework that adapts general code to fit iOS and Android without writing separate code, or a team could try and mirror the iOS app in an Android language.

Lessons learned

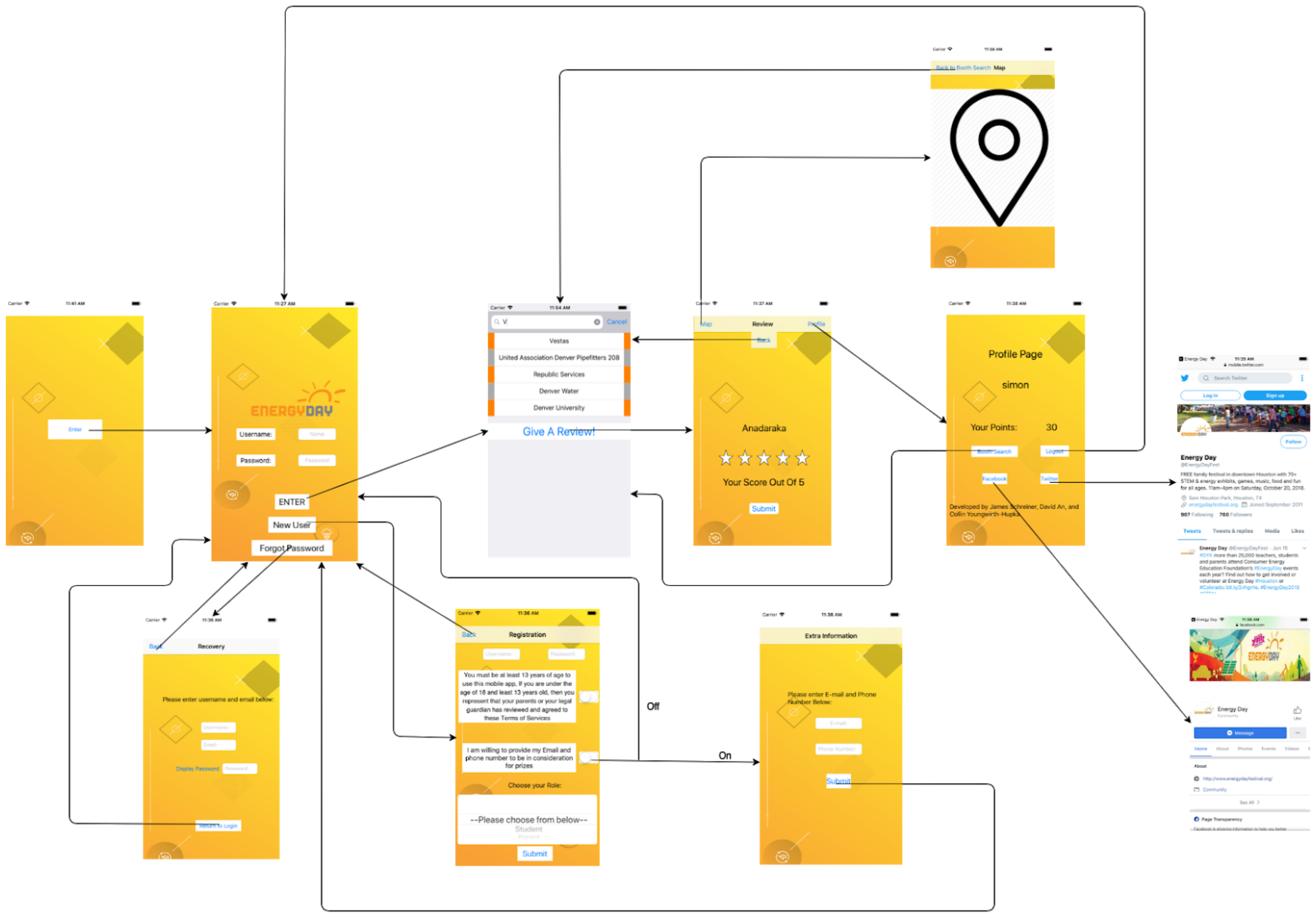
Underestimating an implemented feature:
One of the main components of our application was the table that displayed all the companies at Energy Day. This table needed to use a specific the table view in Swift took multiple days to understand. We could not figure out the size we needed for the table initially. We quickly figured out how to make a static table, but struggled to make a dynamic table that adjusted based off the number of items.

Prioritizing key features:
Another example was the Firebase connection. At first, it seemed simple, but it ended up being difficult to connect our app to Firebase. Despite having the Google Documentation that explained how to set up this vital connection, we ran into issues with keeping our app connected. We learned it would have been better to have started Firebase at the beginning of the project, instead of leaving it to the end.

Limiting the scope of potential implementations:
All of the client's requests could not be satisfied within our time constraint. We needed to explain to our client the aspects of the project that could be met by the deadline. Our client had great enthusiasm and always tried to test the limits of the software; this resulted in us understanding our application more in-depth. However, this meant at times we slowed down the process to ensure that we learned and implemented skills our client requested. This taught us how to gauge our speed of development and professionally communicate what was possible without jeopardizing our relationship with the client. Being able to estimate our time to complete tasks allowed the app to be a compliment of the event, helping CEEF improve the event each year and making the event more interactive for students.

Appendix

Product Storyboards

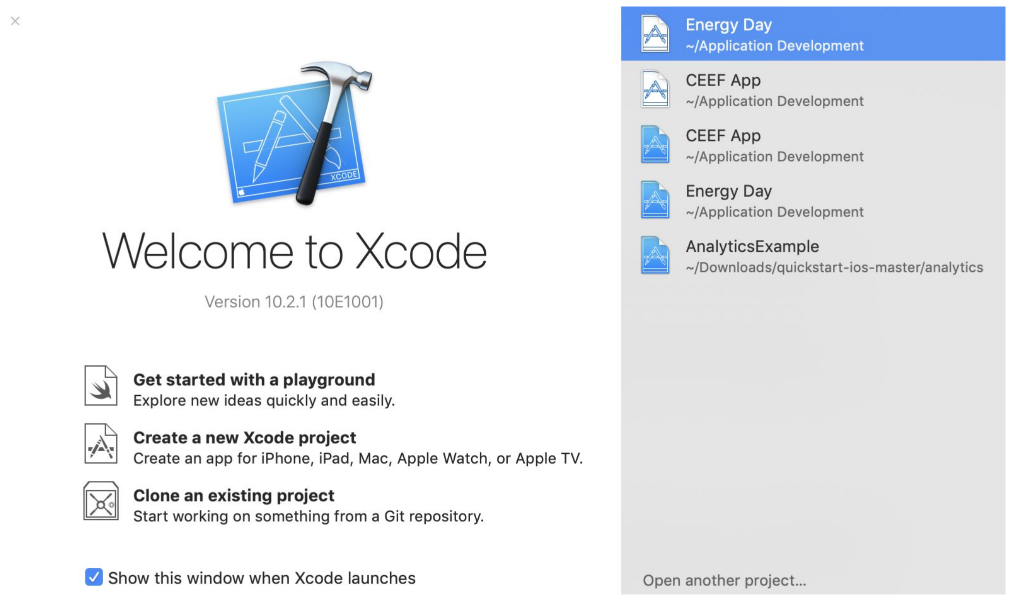


Enter
ENERGY DAY
Username:
Password:
ENTER
New User
Forgot Password
Recovery
Back
Please enter username and email below:
Display Password
Return to Login
Cancel
Vestas
United Association Denver Pipefitters 208
Republic Services
Denver Water
Denver University
Give A Review!
Registration
You must be at least 13 years of age to use this mobile app. If you are under the age of 18 and least 13 years old, then you represent that your parents or your legal guardian has reviewed and agreed to these Terms of Services
I am willing to provide my Email and phone number to be in consideration for prizes
Choose your Role:
--Please choose from below--
Student
Submit
Off
On
Extra Information
Please enter E-mail and Phone Number Below:
Submit
Review
Map
Profile
Back
Anadaraka
Your Score Out Of 5
Submit
Back to Booth Search
Map
Profile Page
simon
Your Points:
30
Booth Search
Logout
Facebook
Twitter
Developed by James Schreiner, David An, and Collin Youngerth-Hupka
Energy Day
@EnergyDayFest
FREE family festival in downtown Houston with 70+ STEM & energy exhibits, games, music, food and fun for all ages. 11am–4pm on Saturday, October 20, 2018.
Sam Houston Park, Houston, TX
energydayfestival.org
Joined September 2011
907 Following
760 Followers
Tweets
Tweets & replies
Media
Likes
Energy Day
Community
Message
Home
About
Photos
Events
Videos
http://www.energydayfestival.org/
Page Transparency

Product Documentation/User Guide

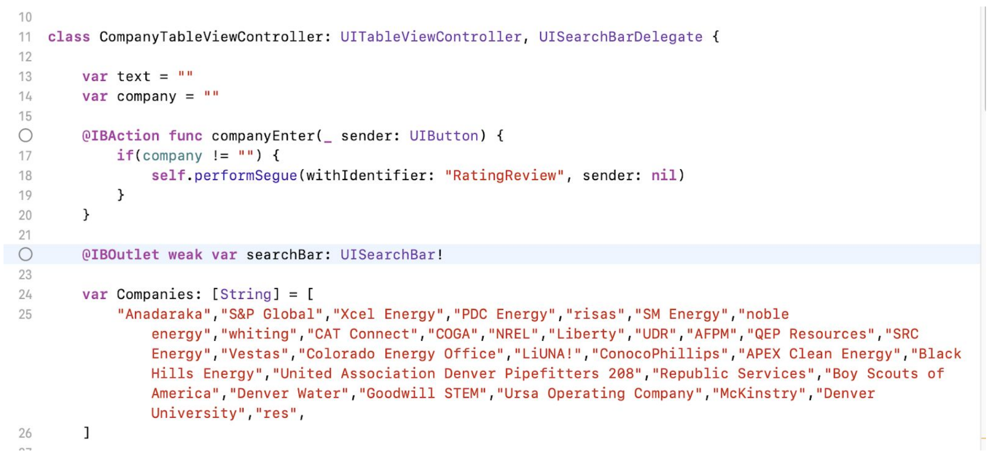
Setting Up Xcode

1. To get Xcode, find it on the Apple App store, download, and install it.
2. When you open Xcode, you can open up the Energy Day application by clicking Open Another Project and finding the "Energy Day" App with the white background in the icon.


Welcome to Xcode
Version 10.2.1 (10E1001)
Get started with a playground
Explore new ideas quickly and easily.
Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.
Clone an existing project
Start working on something from a Git repository.
Show this window when Xcode launches
Energy Day
~/Application Development
CEEF App
~/Application Development
CEEF App
~/Application Development
Energy Day
~/Application Development
AnalyticsExample
~/Downloads/quickstart-ios-master/analytics
Open another project...

Updating Sponsors List

1. While in Xcode with the Energy Day app open, navigate to the CompanyTableViewController.swift file on the left bar.
2. There, find the Companies list. Replace this list with the correct companies by putting quotation marks around each company individually, with a comma following each entry. Leave the braces at the beginning and end and remember to leave the extra comma at the end.



```swift

class CompanyTableViewController: UITableViewController, UISearchBarDelegate {

    var text = ""
    var company = ""

    @IBAction func companyEnter(_ sender: UIButton) {
        if(company != "") {
            self.performSegue(withIdentifier: "RatingReview", sender: nil)
        }
    }

    @IBOutlet weak var searchBar: UISearchBar!

    var Companies: [String] = [
        "Anadaraka","S&P Global","Xcel Energy","PDC Energy","risas","SM Energy","noble
            energy","whiting","CAT Connect","COGA","NREL","Liberty","UDR","AFPM","QEP Resources","SRC
            Energy","Vestas","Colorado Energy Office","LiUNA!","ConocoPhillips","APEX Clean Energy","Black
            Hills Energy","United Association Denver Pipefitters 208","Republic Services","Boy Scouts of
            America","Denver Water","Goodwill STEM","Ursa Operating Company","McKinstry","Denver
            University","res",
    ]
```

Updating Event Map

1. While in Xcode with the Energy Day app open, navigate to the Map.storyboard (screen) that contains the map image.
2. To replace the image with whatever map image you want to use, first make sure you put the map image into the folder "Energy Day" with the other storyboards. Click on the image and change the images.png to the desired image.

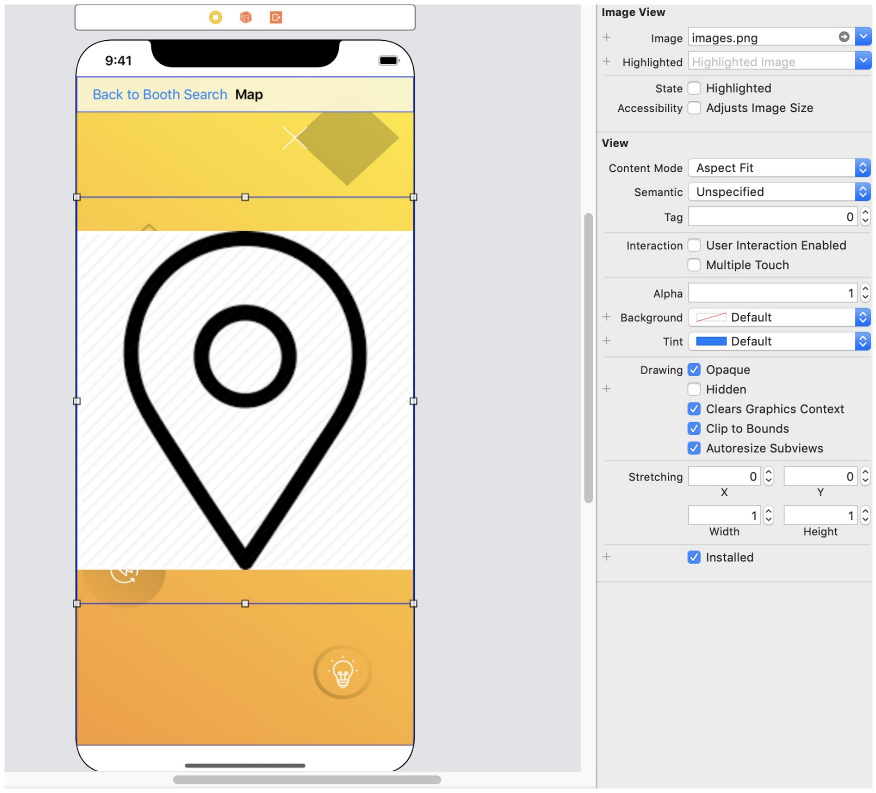
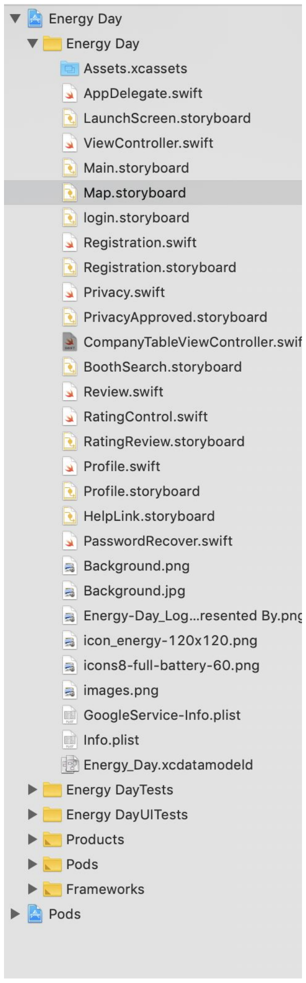


Image View
Image: images.png
Highlighted: Highlighted Image
State: Highlighted
Accessibility: Adjusts Image Size

View
Content Mode: Aspect Fit
Semantic: Unspecified
Tag: 0
Interaction: User Interaction Enabled
Multiple Touch
Alpha: 1
Background: Default
Tint: Default
Drawing: ✓ Opaque
Hidden
✓ Clears Graphics Context
✓ Clip to Bounds
✓ Autoresize Subviews
Stretching: X 0, Y 0
Width 1, Height 1
✓ Installed



- Energy Day
  - Energy Day
    - Assets.xcassets
    - AppDelegate.swift
    - LaunchScreen.storyboard
    - ViewController.swift
    - Main.storyboard
    - Map.storyboard
    - login.storyboard
    - Registration.swift
    - Registration.storyboard
    - Privacy.swift
    - PrivacyApproved.storyboard
    - CompanyTableViewController.swif
    - BoothSearch.storyboard
    - Review.swift
    - RatingControl.swift
    - RatingReview.storyboard
    - Profile.swift
    - Profile.storyboard
    - HelpLink.storyboard
    - PasswordRecover.swift
    - Background.png
    - Background.jpg
    - Energy-Day_Log...resented By.png
    - icon_energy-120x120.png
    - icons8-full-battery-60.png
    - images.png
    - GoogleService-Info.plist
    - Info.plist
    - Energy_Day.xcdatamodeld
  - Energy DayTests
  - Energy DayUITests
  - Products
  - Pods
  - Frameworks
- Pods

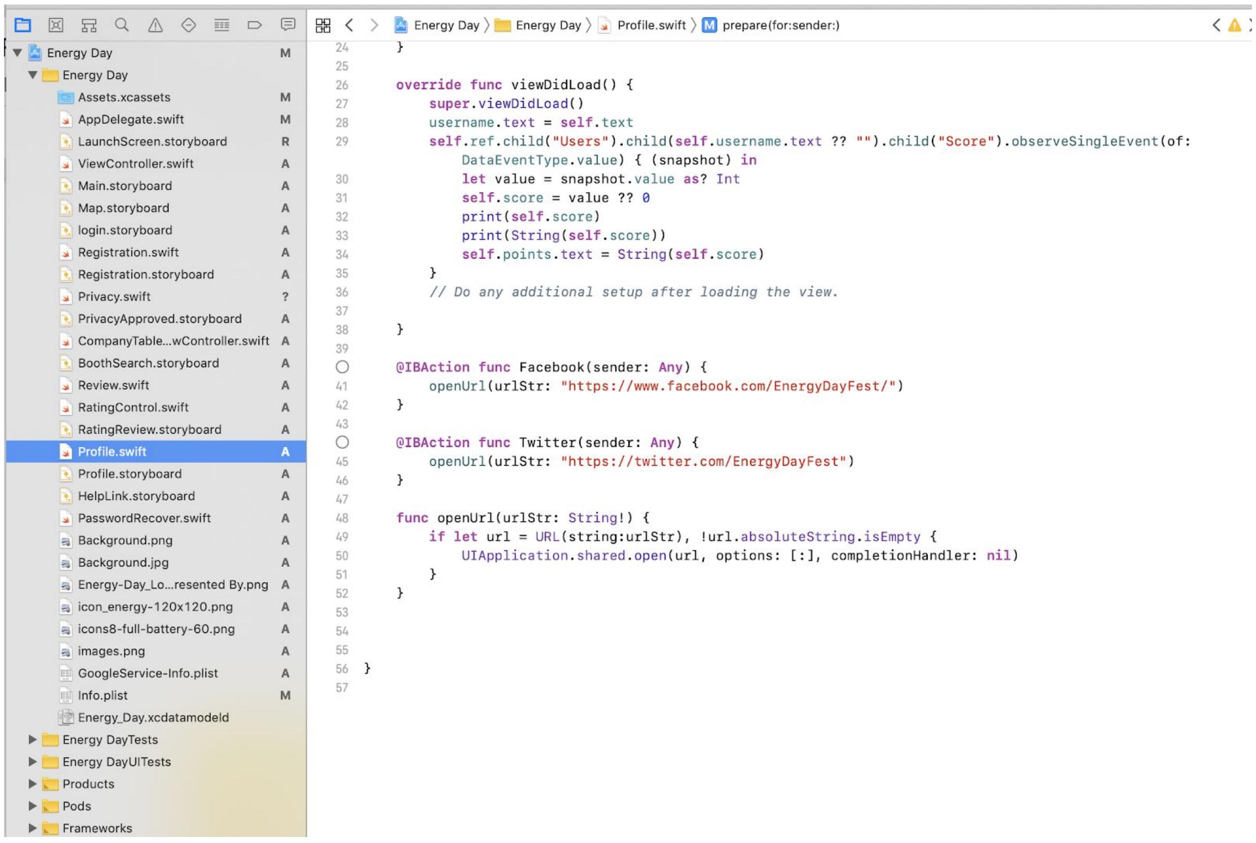
Extracting Data From Firebase

1. Login to Google Firebase using the CEEF account.
2. Click on the CEEF Energy Day Quest project.
3. On the left side of the screen, under the Development tab, find the section labeled Database and click on it.
4. At the top right corner of the panel in the middle, click on the three dots. There should be an option to Export JSON. Select this and the data in the database will be downloaded to your computer as a JSON file.
5. (Optional) If you want to reformat your JSON file into another file format such as a CSV or Excel file, there are a few options:
   a. Online converters allow you to upload the JSON file you previously downloaded and convert it to your desired format. Of course, there is some personal information in this file, so you might want a safer option than uploading this info online (although it likely won't be an issue)
   b. Another option includes downloading and installing some software that can be found online. This should be easy to Google and won't require you to upload the

file onto a website. This keeps any sensitive information safe, but still allows you to convert between file types.

Updating Links

1. While in Xcode with the Energy Day app open, navigate to Profile.swift.
2. On the storyboard, find the link that you want to change and update the link within the parentheses.



```
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        username.text = self.text
        self.ref.child("Users").child(self.username.text ?? "").child("Score").observeSingleEvent(of:
            DataEventType.value) { (snapshot) in
            let value = snapshot.value as? Int
            self.score = value ?? 0
            print(self.score)
            print(String(self.score))
            self.points.text = String(self.score)
        }
        // Do any additional setup after loading the view.

    }

    @IBAction func Facebook(sender: Any) {
        openUrl(urlStr: "https://www.facebook.com/EnergyDayFest/")
    }

    @IBAction func Twitter(sender: Any) {
        openUrl(urlStr: "https://twitter.com/EnergyDayFest")
    }

    func openUrl(urlStr: String!) {
        if let url = URL(string:urlStr), !url.absoluteString.isEmpty {
            UIApplication.shared.open(url, options: [:], completionHandler: nil)
        }
    }



}
```