Attendio: Attendance Tracking Made Simple

Benjamin L. Greenberg
University of Tennessee, Knoxville, bgreenb3@vols.utk.edu

Spencer L. Howell
University of Tennessee, Knoxville, showel17@vols.utk.edu

Tucker R. Miles
University of Tennessee, Knoxville, tmiles7@vols.utk.edu

Vicki Tang
University of Tennessee, Knoxville, wph612@vols.utk.edu

Daniel N. Troutman
University of Tennessee, Knoxville, dtroutm1@vols.utk.edu

Attendio: Attendance Tracking Made Simple

Benjamin Greenberg

[illegible]

Spencer Howell

[illegible]

Tucker Miles

[illegible]

Vicki Tang

[illegible]

Daniel Troutman

[illegible]

Detailed Design Report
ECE402/COSC402 Senior Design Practicum

Tickle College of Engineering
The University of Tennessee
Knoxville, Tennessee
May 3, 2020

Executive Summary

In this report, we discuss the need for an attendance tracking solution and how we built one to fill this niche. Many student organizations on campus use a plethora of different websites, software, or even written paper to keep track of attendees. People almost always have their smartphones, so we implemented this with a cross-platform app. From the attendee's perspective, our app is as simple as scanning an event QR code and receiving a checked-in notification. From the event manager's perspective, they can generate new events and see who is checked in. Our app can enable both parties to maintain a reliable communication channel for events. Our app revolves around our five engineering characteristics to become a great solution for growing student organizations: security, usability, maintainability, adaptability, and aesthetics.

Table of Contents

Table Number	Description	Page Number
0	List of Tables	3
1	List of Figures	3
2	Customer Requirements	5
3	Engineering Characteristics	5
4	Platform-Benefits Matrix	6
5	Framework-Benefits Matrix	6
6	Backend-Benefits Matrix	7
7	Deep Linking Technologies	8

Table 0: List of Tables

Figure Letter	Description	Page Number
A	Business Model Canvas	11
B	Gantt Chart	11
C	Screenshots	12

Table 1: List of Figures

Attendio: Attendance Tracking Made Simple

Benjamin Greenberg, *University of Tennessee*, Knoxville, Tennessee, bgreenb3@vols.utk.edu
Spencer Howell, *University of Tennessee*, Knoxville, Tennessee, showel17@vols.utk.edu
Tucker Miles, *University of Tennessee*, Knoxville, Tennessee, tmiles7@vols.utk.edu
Vicki Tang, *University of Tennessee*, Knoxville, Tennessee, wph612@vols.utk.edu
Daniel Troutman, *University of Tennessee*, Knoxville, Tennessee, dtroutm1@vols.utk.edu

I. Problem Definition and Background

A. What is the problem? Why is the current situation unsatisfactory?

Our technology is attempting to solve a problem that exists in numerous places, but we are primarily focusing on student organizations in the beginning stages. The problem arises from the fact that there exists a great amount of inconsistency in the way that student organizations track events and meeting attendance. As an example, many organizations will have attendees scan a QR code that links to a Google form. Then, attendees will have to spend much more time than necessary just to prove that they have attended an event. This leads to valuable time being lost in what are already typically short meetings and workshops, and could also lead to lowering attendance in the future.

B. Who is having this problem? Who are the would-be customers for a solution?

As mentioned previously, student organizations are having this issue. If thought about much more broadly, these same issues exist in numerous places including workplaces, classrooms, and other similar events. The customers for our solution would be the leaders of these organizations. Our solution would allow them to easily track attendance.

C. What basic functions must the design perform?

The solution must allow organization officers to easily set up events, track attendance at these events, and view overall attendance statistics for each member. This process needs to be seamless, so organizations can quickly take attendance and move on with their meeting. The individual members need to be able to launch the app on their phone, scan the QR code, and receive a success message as they sign in. Other convenience functions, such as removing members from the roster, and marking members as "excused" from meetings, should be supported as well.

D. How will the design be used by the customer(s)?

The app will be used in meetings that occur on campus. Most of these meeting rooms will have projectors that display the slides for the meeting, but they might not be available in all rooms. The students who are present at the meeting will likely have either a phone or a laptop to use. In the case that a projector is not available, a QR code could be displayed on a laptop screen, or a link could be shared to an organization group chat. Since meeting time is short and valuable, our solution should integrate seamlessly with the current practices of the organization and take less time than existing solutions do to take attendance.

E. What is the underlying theory or background that needs to be understood in order to address this problem?

To address this problem, a background in app development and/or UI/UX would be helpful for quickly and effectively building an app that is both secure and usable.

F. What prior work has been done on this problem?

There are several companies that have made attendance tracking software. Many of these only have paid options and offer features that can make the program seem complicated.

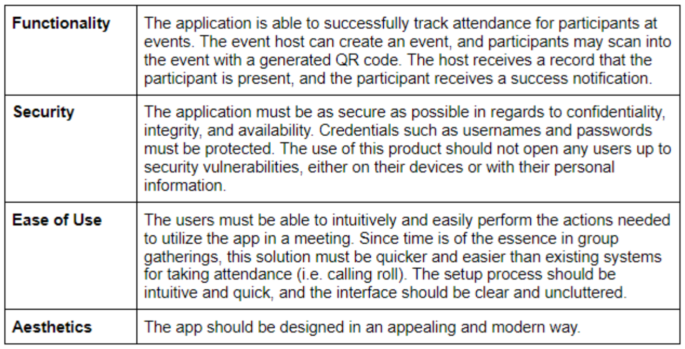
G. What products, currently available, were not designed or intended for this particular application but could be used to perform a similar function?

Other products that could be used to perform a similar functionality of our technology would be employee timesheet software, survey softwares, Google Forms, any type of spreadsheet software, VOLink, and clickers. However, the majority of the attendance tracker softwares made by other companies are more focused on employees and employers rather than student organizations and instructors.

II. Requirements Specification

The final product of our team's development is an attendance tracking app that works for both web and mobile devices. The primary user base is student organizations, so we targeted our initial efforts for them directly. However, our product should also be capable of supporting other use cases, such as professors and students in classrooms and other events where attendance is tracked.

Keeping in mind these varied user segments, our customer requirements are as follows, ranked from most important to least important:

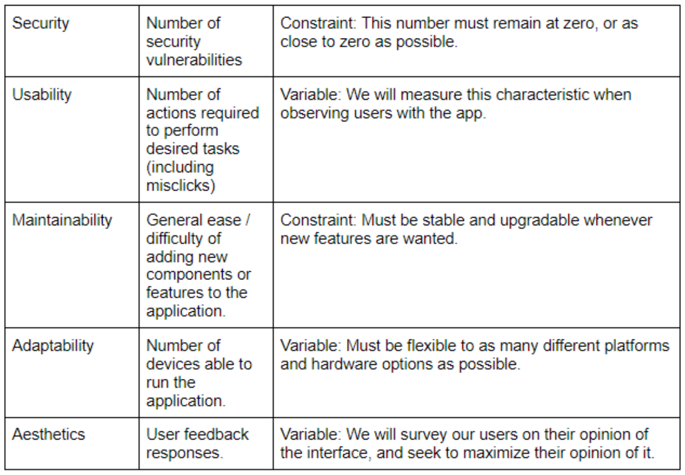


Functionality	The application is able to successfully track attendance for participants at events. The event host can create an event, and participants may scan into the event with a generated QR code. The host receives a record that the participant is present, and the participant receives a success notification.
Security	The application must be as secure as possible in regards to confidentiality, integrity, and availability. Credentials such as usernames and passwords must be protected. The use of this product should not open any users up to security vulnerabilities, either on their devices or with their personal information.
Ease of Use	The users must be able to intuitively and easily perform the actions needed to utilize the app in a meeting. Since time is of the essence in group gatherings, this solution must be quicker and easier than existing systems for taking attendance (i.e. calling roll). The setup process should be intuitive and quick, and the interface should be clear and uncluttered.
Aesthetics	The app should be designed in an appealing and modern way.

TABLE 2
CUSTOMER REQUIREMENTS FOR THE ATTENDANCE TRACKING APP, LISTED FROM MOST IMPORTANT TO LEAST IMPORTANT.

These customer requirements come from our own experiences as members and leaders in student organizations, as well as general knowledge of app usability and best practices.

In addition to the customer requirements, we have generated a list of Engineering Characteristics, based on these requirements, that will allow us to measure our progress and ensure that it aligns with the needs of the customer. Each of these characteristics is either a constraint, that we cannot change, or a variable that we can experiment with. Below in Table 2 is a list of our characteristics.



Security	Number of security vulnerabilities	Constraint: This number must remain at zero, or as close to zero as possible.
Usability	Number of actions required to perform desired tasks (including misclicks)	Variable: We will measure this characteristic when observing users with the app.
Maintainability	General ease / difficulty of adding new components or features to the application.	Constraint: Must be stable and upgradable whenever new features are wanted.
Adaptability	Number of devices able to run the application.	Variable: Must be flexible to as many different platforms and hardware options as possible.
Aesthetics	User feedback responses.	Variable: We will survey our users on their opinion of the interface, and seek to maximize their opinion of it.

TABLE 3
ENGINEERING CHARACTERISTICS

III. TECHNICAL APPROACH

In order to create an application that conforms to all of our stated criteria, the team will need to have a specified technical approach. This approach consists of what tools we use, what practices we employ, and what processes we follow during development.

The first choice that our team must make for our technical approach is what development platform we build our solution on top of. As described in the Design Concepts, Evaluation & Selection section below, our team has decided to use the Flutter framework as it best suits our business needs, as well as aligning with our team's previous experience.

While building a Flutter application, our team will follow best practices as defined by Google in their Effective Dart guide [2]. This will ensure that our code is free from language-based vulnerabilities and is easily refactorable and maintainable, helping us achieve our desired Engineering Characteristic measurements. In addition, we will follow Flutter's defined best practices for performance [3] to ensure that our app is not bloated and will perform at acceptable levels on a wide range of devices.

Developing an app with a team of our size requires coordination and a defined strategy. Therefore, we will follow proven development processes in order to organize our efforts and respond to feedback. Our team is adopting an Agile development philosophy, as defined by the famous Manifesto for Agile Software Development [4]. Our team plans to utilize GitHub's Agile Board features to create tickets for each feature or implementation to be developed, which will then be divided up into defined "sprints" for development tracking. By breaking up our development into pieces, we have the opportunity to test out our ideas with customers and change our priorities if required. This will help us to serve the needs of the customer and remain flexible in our work.

As a part of this Agile process, our team will participate in a code review process, where all submitted code is vetted and learned from by other members of the team. This will help ensure our app is secure, as potential vulnerabilities are caught and corrected before even being merged into the codebase. Further, it will allow our team to learn from each other and develop a shared code style for the project through discussion. This process will help ensure the security and maintainability of our application.

By following this technical approach, our team can ensure that development efforts are well-directed and as productive as possible. We can write maintainable and secure code, iterate quickly, and change direction if needed. All of these benefits will directly serve our customers as outlined in the Customer Requirements and Engineering Characteristics.

IV. DESIGN CONCEPTS, EVALUATION AND SELECTION

There are many ways we can create an attendance tracking application. While we hope to expand our application and port it to as many platforms as possible, we have to start with one for our project. Once our first platform was decided, we needed to pick a language to develop in. Since our application must communicate and store shared information, we also needed to find a backend that integrates with all of our tools and software. For each decision, we have a description of each option as well as their strengths and weaknesses. We also have a decision matrix with ratings for several categories. The higher the rating, the better that option is to the user or us.

A. Platform

Since the customers will be using this application on their device, it is important for us to decide which type to target first. The first thought that comes to mind may be a website where people can sign into their accounts and check into events. This has the benefit of being extremely accessible as it doesn't matter what kind of device you have, as long as it can access the internet. There is no app to install so new people will be able to jump on board very quickly. As far as experience goes, not many of us are experienced in developing web applications, but we were able to include a website as an option for users. We will also address this platform a bit in the frameworks decision section.

Our next possible platform is a desktop application. Several of us have experience with desktop app development in Electron.JS from our COSC 340 class. A severe downside of this platform is the lack of portability and accessibility. Most of the time, people will not be bringing their laptops with them and if they do, it might be in a powered-off state. The installation of an app like this would also be the most intensive of the three on a desktop platform. Users would have to visit a website, download the installer, then install the application.

Our third option would be a mobile application. Like the desktop platform, we have members of our group with mobile app development experience gained from COSC 340. Since virtually everyone has a smartphone on them, a mobile app will have great accessibility. While there is something to install on the device, the process is not as burdensome as the desktop platform.

	Ease of installation	Availability	Easy access	Developer experience
Web	3	3	2	1
Desktop	1	1	2	2
Mobile	2	3	3	2

TABLE 4
PLATFORM BENEFITS MATRIX

Our final decision was to go with a mobile and web application approach. We feel it has the best features to start off with and we have the necessary experience to implement it. While the other platforms can fill other niches, we wanted to pick the ones that have the best availability. When the app was sufficiently fleshed out, we ported it to the web along with some changes that best fit the platform.

B. Framework

For our group's current skill set, most of us have experience with Javascript and Dart from previous projects. Because the skill set of our group is specialized towards these two languages, we ended up narrowing down our options to React Native and Flutter as the frameworks for our app. Although React Native can be brought to the desktop thanks to libraries like Proton Native or support from Microsoft, if we wanted to bring it to the web in the future it would require building a separate ReactJS project as React Native was built for mobile. While this is not necessarily too difficult to accomplish, it is an extra step that could be better accomplished in an alternative framework like Flutter that will have official native support for the web.

Aside from everyone in this group already being familiar with Dart and developing with the Flutter SDK, we can also use this framework to bring our app to the web and desktop in the future if we choose to. Another positive of Flutter over React Native is that Flutter has native support for Material UI design guidelines. This will allow us to more efficiently design a clean and efficient UI/UX for our app instead of having to install another library to a React Native project. For the reasons of familiarity, native Material UI support, cross-platform compatibility, and future platform development, we have decided to go with the Flutter SDK for the framework for our app.

	Database	Pricing	Time and Labor	Official Flutter Support
AWS	3	2	2	1
Firebase	1	3	3	3
Custom	3	1	1	1

TABLE 5
FRAMEWORK BENEFITS MATRIX

C. Backend

Along with the platform and framework for our project's app, we also needed to choose a suitable backend for authentication, hosting, storage, etc. The options we considered for this project were AWS, Google Firebase, and possibly a custom backend written in Django, Flask, etc. As we are developing our app for this project, a custom backend seemed unnecessary as we would have to handle hosting the backend as well as handle any scalability issues if the user base of the app grew in the future. Therefore our options for a backend were down to AWS and Firebase. The deciding factors between AWS and Firebase came down to the following options: database requirements, complexity, free services vs. paid services, and support for the Flutter SDK.

After comparing both AWS and Firebase and determining which backend is more suitable for our app, we decided to go with Firebase for our backend. Even though Firebase only offers NoSQL databases while AWS allows for a choice of which database to use, Firebase seemed more appropriate for our app's needs as we don't expect our database requirements to grow too complex. As we need a backend that's relatively easy to set up and communicate with, we found that with Firebase it was easier to both set up and interact with callable functions than in AWS. At the moment, Firebase provides more services for free such as user authentication than AWS does. Firebase also looks to be easier to set up than AWS as even though it may offer fewer services out of the box, it is enough for our app at this stage. Flutter even has official support for the Flutter SDK whereas AWS does not. Due to these reasons, we will be using Firebase as the backend for our Flutter-based mobile app.

Now that we have all the major decisions out of the way, we can get started with learning all about these services and software. Since we decided on a mobile platform first, we

	Range of Platforms (Web, Mobile, Desktop, etc.)	Language Experience	Ease of Setup	Material UI Support
Flutter	2	3	3	3
React-Native	2	2	2	2
Ionic	3	1	1	2
Xamarin	2	1	2	2

TABLE 6
BACKEND BENEFITS MATRIX

created our wireframes and mock-ups to fit the cell phone format. We also were able to get a general idea of our web application and created a wireframe for it. We have decided on the overall layout, but were not able to fully implement the design we created in our wireframes as we were more focused on building a MVP under time constraint. Flutter is our chosen platform due to its simple integration with the web, Android, iPhone, and our backend. Our development experience with the language is also a big plus and we started with brushing up on various concepts to begin the application development. Finally, we chose our backend with Firebase due to its ease of setting up and overall experience. By already deciding these major decisions together, we had a smooth transition to implementation and these choices gave us the best chance of success with planning and the logistics of the app.

V. EMBODIMENT DESIGN

A. Product Architecture

1) Modules:

a) Material UI: Material UI is a design language that was released by Google. It uses grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. This more or less defines the "look and feel" of our application, and will provide users with a familiar interface when using our application, as Material UI is a standard used by many.

b) Firebase Authentication: To make the adoption of our application easier for users, we decided to use Firebase Authentication as our authentication solution. Firebase Authentication allows for a relatively simple implementation and integration process to add authentication from sources such as Google with ease. Firebase also allows for easy integration with other authentication providers in the future such as Facebook. As Firebase and Flutter are both products from Google, they are more likely to work together reliably than other authentication solutions for Flutter. This works well for our use case as it saves us the time and effort of trying to develop our own authentication solution while making sure that it is secure and works with multiple authentication providers.

c) Cloud Firestore: Cloud Firestore is one of the options that Firebase provides to use as a backend database for our application. We chose Cloud Firestore because of its high flexibility, scalability, and ease of use. It acts as a place for us to store data, keeping it in sync across all of our applications (iOS, Android, and web) while providing real-time listeners which allow us to interact with it. Additionally, Firebase and Flutter are both products from Google, and staying in this environment gives developers some peace of mind when it comes to implementing this, as it's a very plug-and-play solution that integrates seamlessly into our application.

d) Dynamic Links: Dynamic Links is an implementation of the deep link concept. A deep link is like a hyperlink, but it will send you to a specific part of an app instead of a website. Other behavior like opening the app store or redirecting to a web version is also possible if you do not have the app installed. Dynamic Links is a service provided by the Firebase platform that is already in our app. We use the service to create links that redirect users to the check-in page for a particular event in our app. Each dynamic link is unique because they encode event information that our app will use to look up the event.

e) Riverpod: Riverpod is a state management solution based on the Provider package by the same developer. Riverpod offers improvements over the Provider package such as catching programming errors at compile-time instead of runtime, it removes the need for nesting when listening or combining providers, and helps simplify testing state management. Riverpod benefits our app directly by simplifying the creation of state providers for the entire app and allowing us to create them outside of the widget tree. Riverpod also simplifies combining providers and abstracting data from one provider into another. Another benefit of Riverpod is that any widget within the provider scope tree can access the state of any provider just by importing the provider. This removes the need for us to nest the provider through every widget, cleaning up and making our code more maintainable.

2) Inputs & Outputs:

a) Material UI: Material UI exists in every single component of the front-end of our application. The Material UI of Attendio is codified through defining trees of widgets, showing how they interact, and styling them accordingly. Flutter looks at this code, analyzing the sizes, layouts, and logic that we've defined, and draws the user interface onto the screen.

b) Firebase Authentication: The Firebase Authentication components of our app are seen in several places. First off, it acts as the gateway to Attendio. When you launch the app, you are greeted with a login screen. You give our Firebase Authentication module your Google Credentials, and Google sends back a token that uniquely identifies you and grants you access to several resources. Additionally, this is used in the profile section of Attendio. We are able to retrieve a user's name, profile photo, and unique ID from this communication

c) Cloud Firestore: Cloud Firestore is primarily used in the "Events" section of Attendio. We send our backend a user's Firebase ID token which was retrieved from the Firebase Authentication SDK, and this allows us to make requests on behalf of the user. This token is passed to the backend along with what we are requesting, and the backend sends back whatever data we've requested.

d) Dynamic Links: Our dynamic link module is in several parts of our app. When an event is created in Firestore, the event id is passed to the dynamic link module and it includes it in a link. A QR code is generated that encodes the dynamic link for easy scanning on mobile devices. When someone scans the QR code or goes to the dynamic link in their browser, Attendio is opened and the check-in page is loaded. When the check-in page is loaded, it will extract the event id from the dynamic link that launched the page. The event id will then be used to find and display the event information.

e) Riverpod: Our Riverpod code is in two parts of our app. The first part is in our providers folder where we initialize the providers and develop any logic related to purely state management. The other part is within the UI widgets. By using providers, we can have our app update whenever the state changes. In our application, we used providers to share the authentication state and information of the current user to all widgets in the app that required either the state or information. We were also able to use it to keep track of the current tab the user was on so we could build the UI based on the currently selected tab in the bottom navigation. Our app also has a provider for our Dynamic Links. We use this provider to share the same instance of our Dynamic Links services class throughout the app, allowing us to access this functionality without the need for nesting.

B. Configuration Design

1) Component Selection:

a) Material UI: Material UI was chosen for our application for its simplicity to implement as well as it's popularity among users. Material UI is much more minimalistic than other design options, making it much simpler to implement. This allowed us to focus more on adding functionality to our application rather than getting stuck on defining the look and feel. Additionally, Material UI is common among many applications, and following these design recommendations will allow us to greet our users with an application that seems familiar.

b) Authentication (Google Sign-in): We used Google Authentication because of how easy it was to integrate into our Application. To get this up and running in its most basic way, the work consisted of nothing more than adding the necessary packages to our application and using them. Secondly, this is in the Google environment, and since Flutter is in this same environment, integration is seamless. Lastly, using Google Sign-in keeps us from having to store a bunch of user data, such as ids, photos, and any other metadata that is attached to a user. All of this combined allowed us as developers to spend time on other critical components of the application and not get caught up in the hurdles that can exist with authentication.

c) Cloud Firestore: The biggest contributor to why we chose Cloud Firestore as our backend was its flexibility as well as the simpleness of its implementation in Flutter. Flutter and Cloud Firestore are both Google products, and this allows for a very smooth integration process. To use it, the overhead consisted of nothing more than adding the package to our application. To speak on Cloud Firestore's flexibility, it is a NoSQL database which means it is not modeled through tabular relations as seen in relational databases. This allows us to place our data how we want it, where we want it, and in the exact structure we want it, all without having to worry about relationships.

d) Dynamic Links: There are several reasons we went with Dynamic Links over the other deep linking protocols. Android and iOS use different ways that they interpret links and offer multiple libraries. iOS can use custom URL schemes or Universal Links while Android can use App Links or Deep Links. There are many differences between all of these with the main one being specific hosts and files [5].

	Requires Specified Host	No Specific Host
iOS	Universal Links	Custom URL schemes
Android	App Links	Deep Links

TABLE 7
DEEP LINKING TECHNOLOGIES

We want to support as many platforms as possible but having specific platform channels that handle it differently would make the app very complicated and hard to maintain. With Dynamic Links, all of the OS-dependent operations are abstracted away so you get a single link that works with all platforms. Since we already had Firebase in our app, the decision to use this service was even simpler.

e) Riverpod: Although Riverpod is still relatively new and not quite ready for production-grade applications, the benefits it offers over the Provider package outweigh this risk for the scope of this project. Being based in Dart instead of Flutter, any Riverpod provider can be used outside of the widget tree. Riverpod simplifies combining providers by providing a reference parameter when creating the provider that can read the state of other providers. This package also provides an observer that can be very helpful for debugging any issues with Riverpod providers and state management. Riverpod also offers a variant package called hooks_riverpod that further simplifies using providers by providing a custom "hook" function that can be used with the Flutter Hooks package.

VI. Test Plan

A. Security Test

The first engineering characteristic we listed is security. Security is very important to us because we want users to trust us with their information. If any user information got stolen, it would be very embarrassing and hurt the app's reputation. Adversaries could also use user data to gain knowledge about people that they may not want to be public. To minimize vulnerabilities, we should try to patch as many bugs as we can. A simple way to find bugs is just by providing a bug reporting system within the app. Once the app is more stable and there are fewer vulnerabilities found, we could add some sort of bug bounty program. This program would reward people for finding vulnerabilities in our service. There are several ways to go about this but we could start with a crowdsourcing platform like Bugcrowd. Researchers on this website will try to find parts of apps to break and they get a reward for exposing them. If crowdsourcing was not appealing, we could have a security engineer probe the app to see if they can find any revealing information or exploits that could compromise our app. One member of our team currently works in this type of position, so locating the right person would not be a challenge. No software is perfect so if there is ever a time where a vulnerability is found, we hope to fix it and update as soon as possible.

B. Usability Test

If our app is not very user-friendly, then people will not want to use it for their events. One way we can test usability is with a small group of volunteers. These participants will have a few tasks to do and can be observed doing them. We can note what parts of the app are confusing and what mistakes are made. This feedback is great to see how usable Attendio is to the average user. The app should be simple enough to not need a tutorial, but it may be necessary as more features are added.

C. Maintainability Test

To make sure the application is robust and is reliable with every update, we can use automated testing. Since it is hard to test every function in the app for potential breaking, we can write unit tests to take advantage of our app's modularity. These tests can tell us if functions no longer behave as they should. Since it is all automated, we can run tests after every feature that gets added. While automated testing is not perfect, it can be a great way to ensure parts of our app do not get overlooked. As Attendio gets more complicated, we can write more complex integration tests to make sure the entire modules are working accordingly.

D. Adaptability Test

This engineering characteristic is a bit more abstract. We want to be very adaptable and support as many devices as possible. We used a cross-platform language that makes this easier for us to support many devices. While the language gives compatibility with the underlying systems, we still need to make sure devices have an easy way to update. This can be accomplished with CI/CD or continuous integration and continuous deployment. Whenever we have updates to the app, we can set it up so all platforms will receive the latest versions that are ready to be installed. Easy updates are important to make sure users apply security patches and have access to the latest features.

E. Aesthetics

The last engineering characteristic is more subjective. While everyone has their own opinions on how an app should look, there are best practices to make sure content is presented clearly and pleasing. We can get feedback from our app by sending out a survey on MTurk. MTurk would let us crowdsource feedback for a relatively inexpensive price. The survey can include several screenshots of Attendio pages and ask various questions about the looks. If there is significant concern about colors, buttons, layout, or other UI features, we may need to reconsider how the app looks. We can even have an area for feedback to see if any user has ideas that would improve the design.

VII. Project Deliverables

As a starting point our project has two main deliverables, a phone application and a web application. These products can be seen as our turn-in deliverables, but our true deliverable is to create a tool which provides users with our service through whatever means necessary. Fortunately, we have picked tools which allowed us to reach this objective, and our methods are described in detail within this proposal as well. These deliverables are applications which possess the functionality that has been previously mentioned in the background sections of this proposal. Our first deliverable was in the form of an Android mobile application. Developing a web application for our project is built into the timeline; however, in the event of setbacks, this portion of the project would have been the first item to be withdrawn from the project requirements, as our main focus was initially on completing our minimum viable product as a phone application.

However, we were able to successfully build not only a phone application, but also an adaptive web application that allows our users to access our service using any device with a web browser. Lastly, our chosen framework allows us to create our mobile application in both an Android and an iOS environment with minimal code changes. While our team was not able to complete this deliverable due to time and hardware constraints, we plan on continuing development on the project and including an iOS build in future releases.

VIII. Project Management

During the planning phase of the project, our team created a blueprint for the application in terms of user design and user experience by using Figma, a vector graphics editor and prototyping tool. We discussed various design problems, created solutions, and developed a unique look and feel for our application.

We used this initial product design to list out milestones of the project that are significant for the minimum viable project. After organizing our work and dividing it up, we began active development in early January 2021. Our development schedule was organized into a series of sprints with bi-weekly and weekly goals in mind.

We assigned a variety of roles to each team member during the development process. As we only have five members on the team, we each had to take on a variety of roles with different skill sets required. All team members actively participated in the software development lifecycle, but certain team members specialized in areas of the process.

Benjamin focused on setting up a Continuous Integration system to test our builds as we made changes, as well as implemented the Riverpod state management solution to ensure the quality and maintainability of our software. Daniel created the dynamic link system that allows a user to scan our QR codes from any camera app and be automatically redirected to our application. Vicki acted as lead mobile designer for the application, including implementing many of the user interfaces on the mobile application. Spencer was responsible for the web design mockups, as well as implementing a variety of screens on the mobile and web apps. Finally, Tucker served as the project manager for the duration of the project, organizing team meetings and managing tasks on the agile board, as well as development tasks including the user authentication system.

Due to the fact that we are developing a software application, we did not require any external funding for our application. We used free software to build our project and the cost will be based on maintaining our database and information within the application. We have not yet reached a scale where we need to pay for our database usage, and therefore did not need to acquire external funding. In the future, we might need to begin using a paid data plan if our storage requirements increase. In addition, in order to release on the Apple App Store, our team would need to acquire an Apple Developer License, which would cost $100.

Security protection is one aspect that posed the greatest risk in our application. For authentication, we ended up using Google since it is widely used and well vetted. Our databases are stored in the Firebase platform so they are protected by our development credentials and not stored in Attendio.

Overall, the project went smoothly due to a detailed plan and a great development team. We completed all the team and class milestones to achieve project viability. We might continue work on this project to finish up some additional features and polish it around the edges. Attendio was a great learning experience where we practiced agile app development and team collaboration.

IX. Budget

A. Google Firebase - Free

Although we don't expect for our app to have enough traffic to warrant fees from Firebase initially, we have a budget of $100 for any possible fees we may encounter should our app's traffic grow large enough. With our small amount of testing and basic use, we did not get anywhere near the thresholds that the Firebase free tier constrained us to.

B. Github - Free

To host our code and enable collaboration between group members, we used Github. By using Github, we could work on different parts of the same project at the same time while keeping the repository private if we need to.

C. Flutter - Free

We based our codebase on the Flutter SDK by Google as this allowed us to develop our app both at a low cost in terms of money and time. The Flutter SDK also has native support for Material design guidelines as they're both by Google, allowing us to easily create a desirable UI/UX. With Flutter, we will also be able to easily bring the app to the desktop if we decide to go in that direction.

D. Android Studio/Visual Studio Code - Free

Our group used both Android Studio and Visual Studio Code to develop our app. This means each developer could choose which IDE they use as long as they could work on the app without affecting the codebase itself.

X. References

[1] B. Kopf, "The Power of Figma as a Design Tool," toptal.com.[Online]. Available: https://www.toptal.com/designers/ui/figma-design-tool. [Accessed November 20, 2020].

[2] "Effective Dart," 2020. [Online]. Available: https://dart.dev/guides/language/effective-dart. [Accessed: 20-Nov-2020].

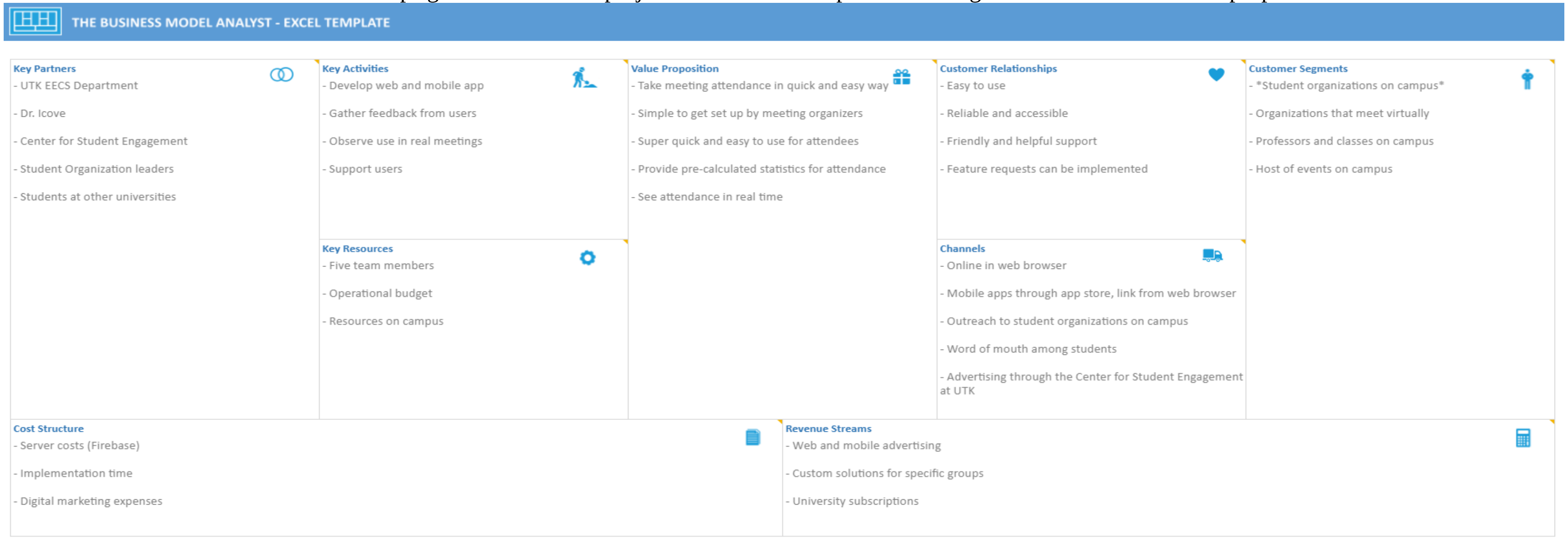
[3] "Performance Best Practices," Flutter, 2020. [Online]. Available: https://flutter.dev/docs/perf/rendering/best-practices. [Accessed: 20-Nov-2020].

[4] K. Beck, Manifesto for Agile Software Development, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed: 20-Nov-2020].

[5] A. Denisov, "Deep Links and Flutter applications. How to handle them properly.," Medium, 03-Jul-2019. [Online]. Available: https://medium.com/flutter-community/deep-links-and-flutter-applications-how-to-handle-them-properly-8c9865af9283. [Accessed: 01-May-2021].

Appendix A: Business Model Canvas

This model helps give our team and project a direction in respect to our target audience and define its purpose.



THE BUSINESS MODEL ANALYST - EXCEL TEMPLATE

Key Partners
- UTK EECS Department
- Dr. Icove
- Center for Student Engagement
- Student Organization leaders
- Students at other universities

Key Activities
- Develop web and mobile app
- Gather feedback from users
- Observe use in real meetings
- Support users

Key Resources
- Five team members
- Operational budget
- Resources on campus

Value Proposition
- Take meeting attendance in quick and easy way
- Simple to get set up by meeting organizers
- Super quick and easy to use for attendees
- Provide pre-calculated statistics for attendance
- See attendance in real time

Customer Relationships
- Easy to use
- Reliable and accessible
- Friendly and helpful support
- Feature requests can be implemented

Channels
- Online in web browser
- Mobile apps through app store, link from web browser
- Outreach to student organizations on campus
- Word of mouth among students
- Advertising through the Center for Student Engagement at UTK

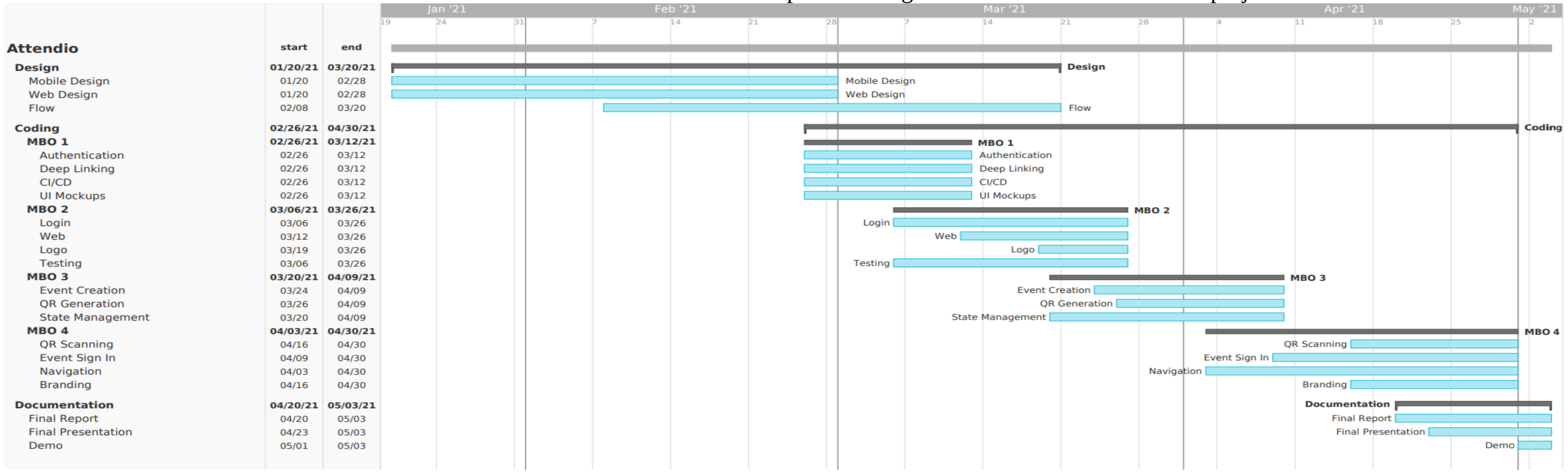
Customer Segments
- *Student organizations on campus*
- Organizations that meet virtually
- Professors and classes on campus
- Host of events on campus

Cost Structure
- Server costs (Firebase)
- Implementation time
- Digital marketing expenses

Revenue Streams
- Web and mobile advertising
- Custom solutions for specific groups
- University subscriptions

Appendix B: Gantt Chart

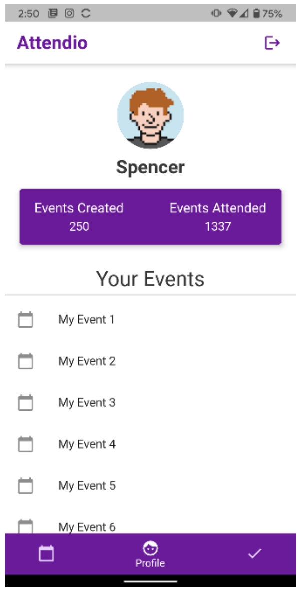
This visual shows the timeline of tasks we have completed throughout this semester as build our project.

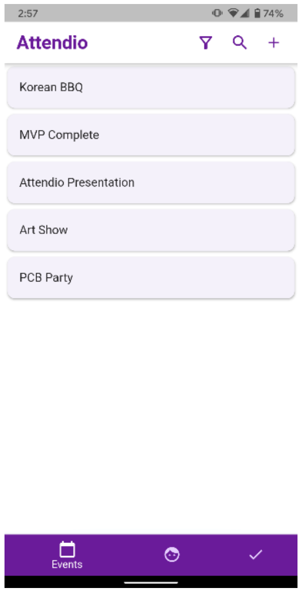


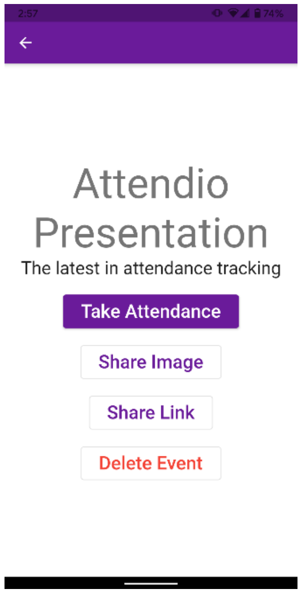
Attendio	start	end
Design	**01/20/21**	**03/20/21**
Mobile Design	01/20	02/28
Web Design	01/20	02/28
Flow	02/08	03/20
Coding	**02/26/21**	**04/30/21**
MBO 1	**02/26/21**	**03/12/21**
Authentication	02/26	03/12
Deep Linking	02/26	03/12
CI/CD	02/26	03/12
UI Mockups	02/26	03/12
MBO 2	**03/06/21**	**03/26/21**
Login	03/06	03/26
Web	03/12	03/26
Logo	03/19	03/26
Testing	03/06	03/26
MBO 3	**03/20/21**	**04/09/21**
Event Creation	03/24	04/09
QR Generation	03/26	04/09
State Management	03/20	04/09
MBO 4	**04/03/21**	**04/30/21**
QR Scanning	04/16	04/30
Event Sign In	04/09	04/30
Navigation	04/03	04/30
Branding	04/16	04/30
Documentation	**04/20/21**	**05/03/21**
Final Report	04/20	05/03
Final Presentation	04/23	05/03
Demo	05/01	05/03

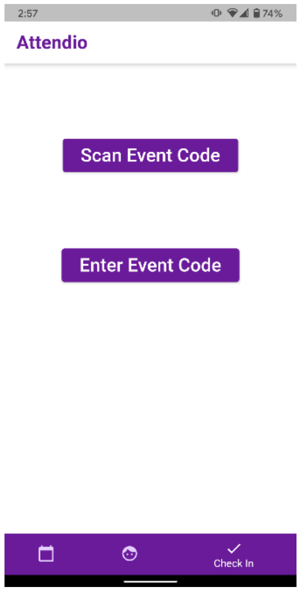
Appendix C: Screenshots

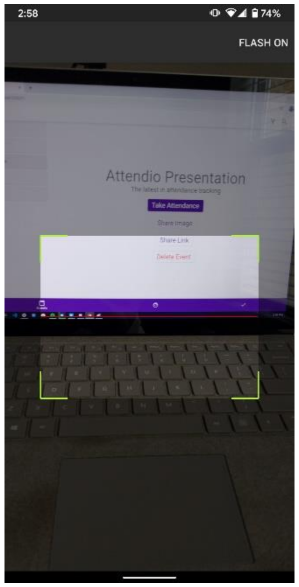
These screenshots are from various pages within the mobile and web app.

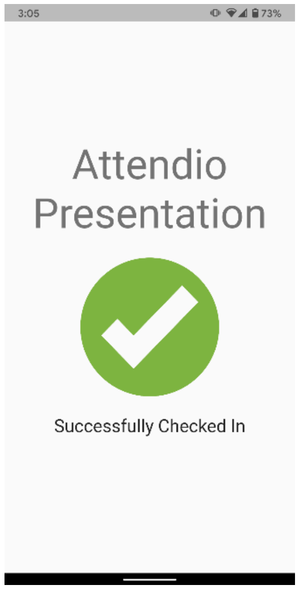


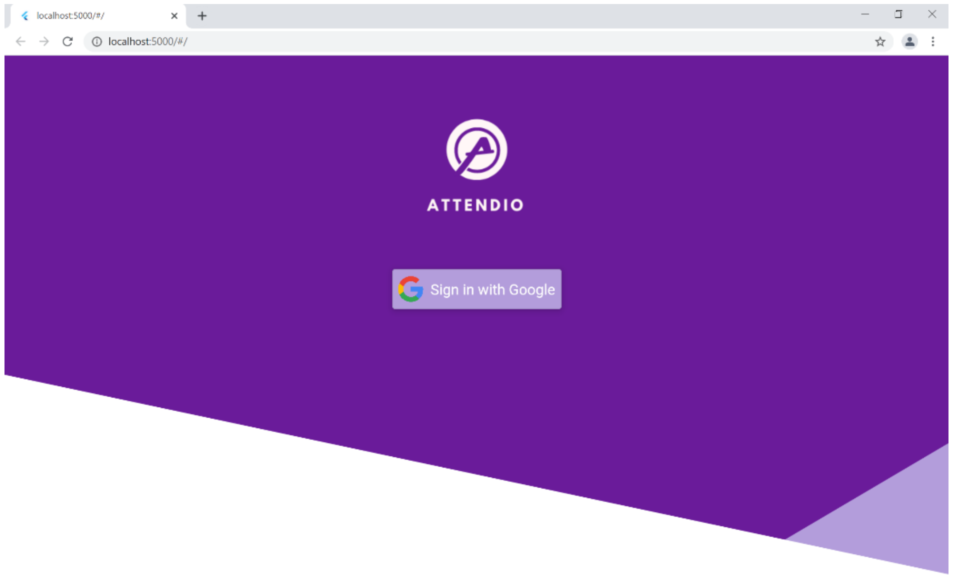


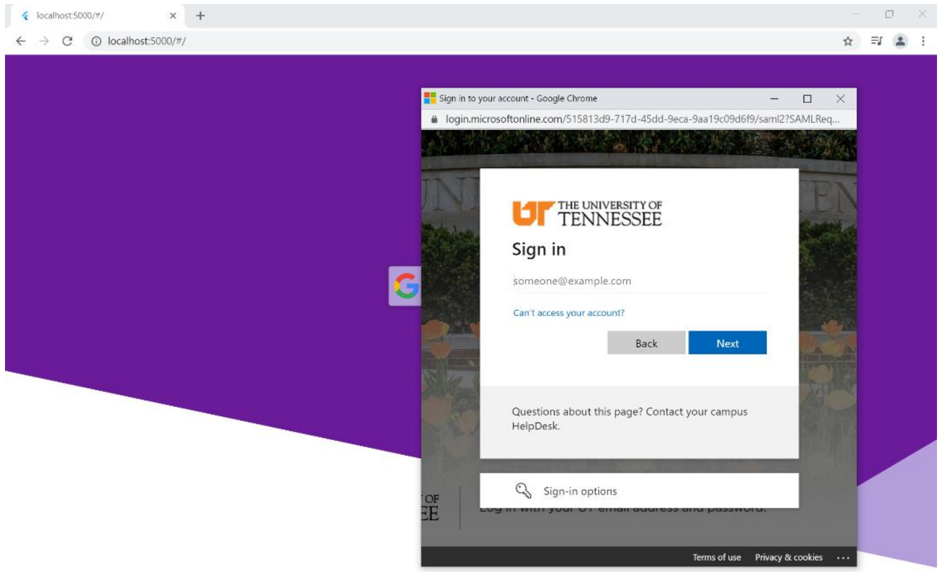


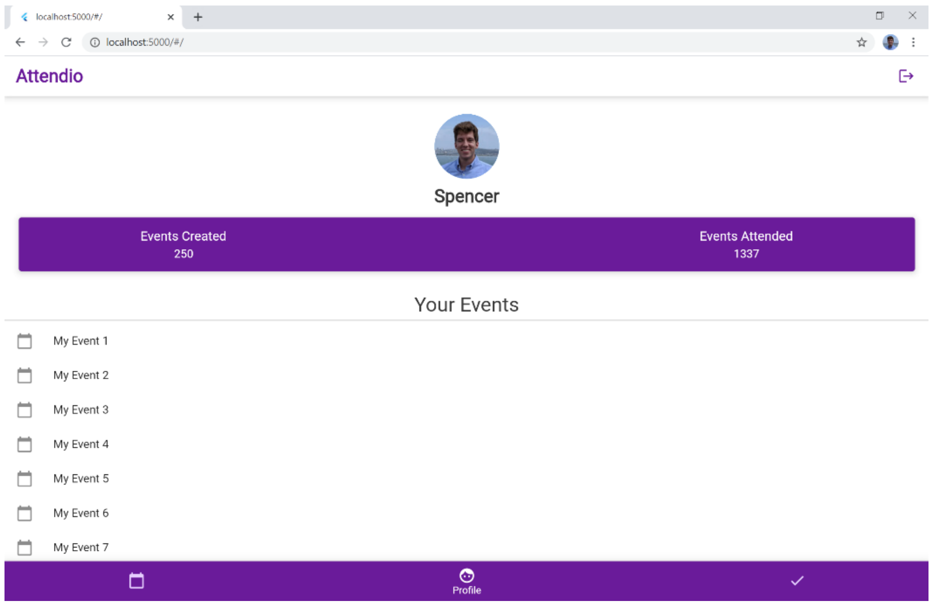









localhost:5000/#/
Sign in to your account - Google Chrome
login.microsoftonline.com/515813d9-717d-45dd-9eca-9aa19c09d6f9/saml2?SAMLReq...
THE UNIVERSITY OF TENNESSEE
Sign in
someone@example.com
Can't access your account?
Back
Next
Questions about this page? Contact your campus HelpDesk.
Sign-in options
Terms of use
Privacy & cookies



localhost:5000/#/

Attendio

Spencer

Events Created 250

Events Attended 1337

Your Events

My Event 1

My Event 2

My Event 3

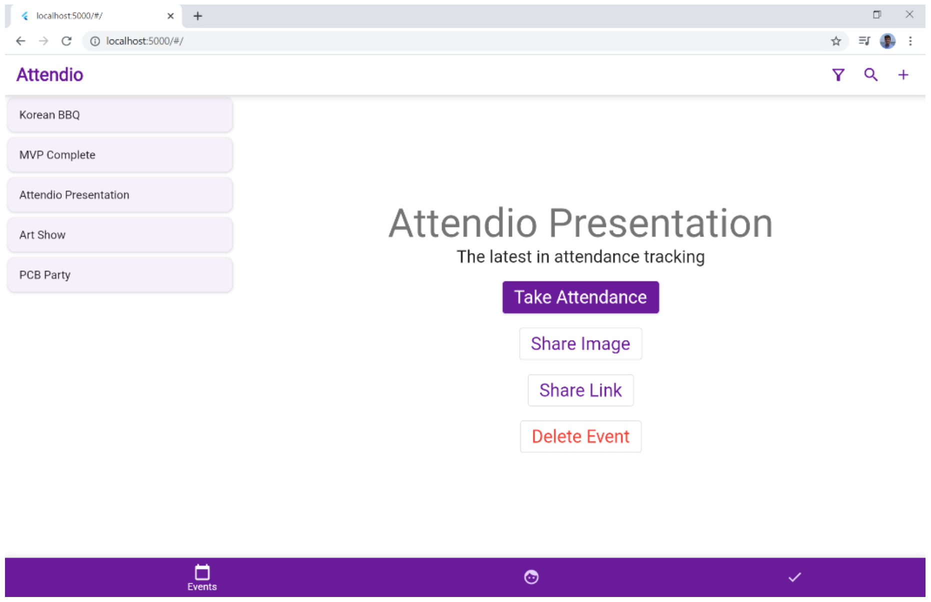
My Event 4

My Event 5

My Event 6

My Event 7

Profile



localhost:5000/#/

Attendio

Korean BBQ

MVP Complete

Attendio Presentation

Art Show

PCB Party

Attendio Presentation

The latest in attendance tracking

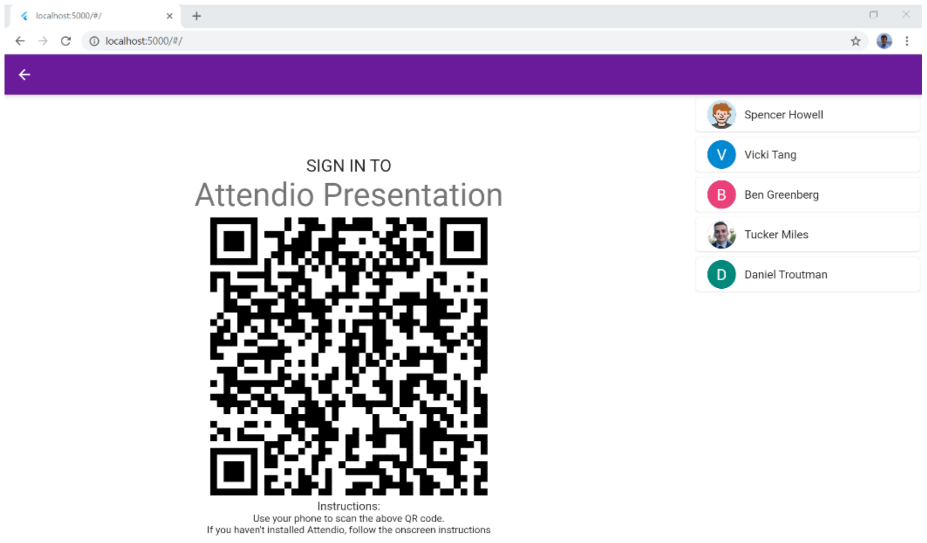
Take Attendance

Share Image

Share Link

Delete Event

Events



localhost:5000/#/

Spencer Howell

Vicki Tang

Ben Greenberg

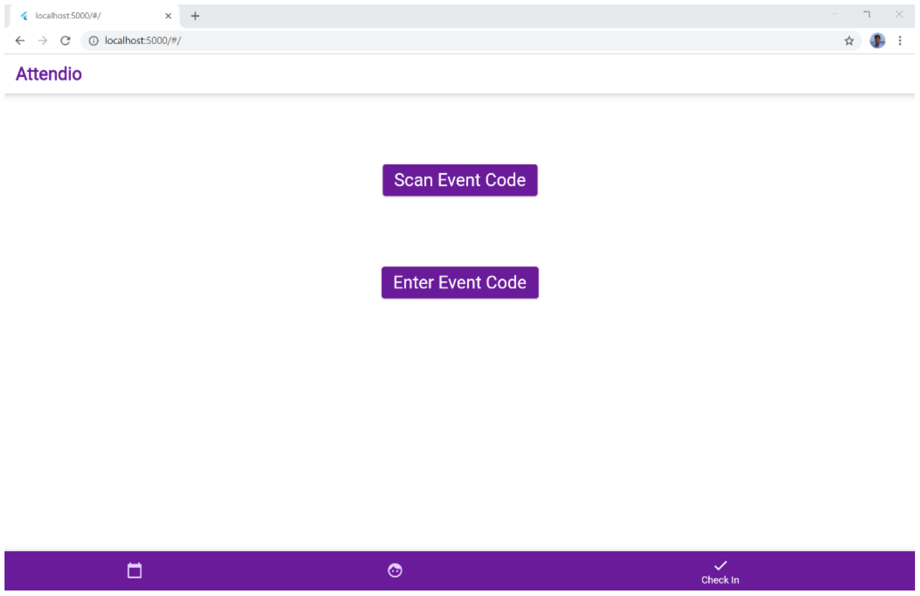
Tucker Miles

Daniel Troutman

SIGN IN TO

Attendio Presentation

Instructions:
Use your phone to scan the above QR code.
If you haven't installed Attendio, follow the onscreen instructions



localhost:5000/#/

Attendio

Scan Event Code

Enter Event Code

Check In