Documentation

2023 FBLA State Leadership Conference
Mobile Application Development

Updated February 2023

Wallace McCarthy
Dheeraj Vislawath

Documentation

Table of Contents

1. Software Documentation

1.1 Overview

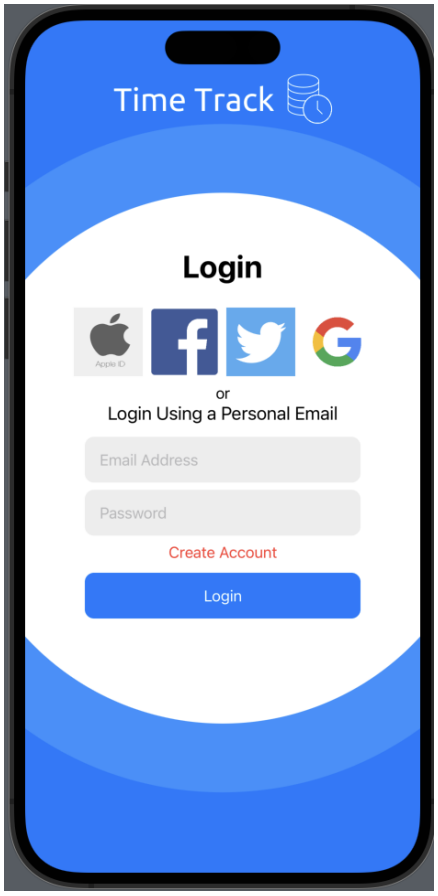
The Time Track application is a community information app that provides information about upcoming events, activities, and news to help keep parents and the community up to date.

1.2 Splash Screen

The Time Track Application incorporates an animated, entry screen when first loading the application. This screen prominently showcases the name of our application “Time Track” as well as our self created Time Track Logo. This screen is displayed for a brief period before transitioning to the subsequent screen, where the user will be able to login.

1.3 Login Screen

The login screen allows for the user to either enter an email and password or to login via a browser application which include: Apple ID, Twitter, Facebook, and Google. For logging in with an email and password, if one or both of these fields are incorrect, the text box surrounding them will be highlighted red, allowing the user to identify if either their password is incorrect, their email is incorrect or both. Under the password textbox there is a button called “Create Account” which can be used for a new user to create a new account. This button will direct the user to a new screen full of information about creating an account.



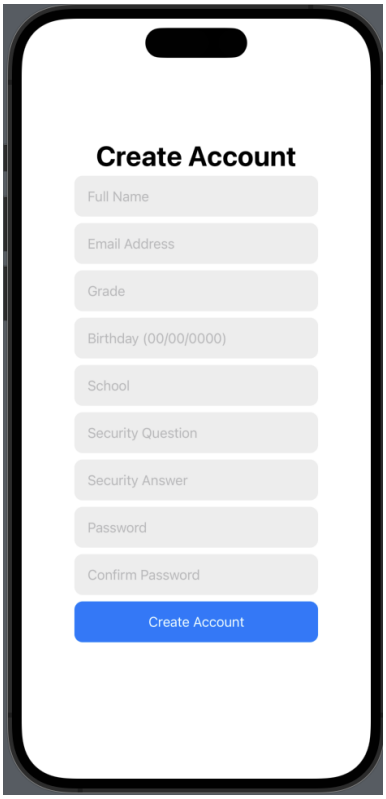
1.3.1 Create Account Screen

The create account screen prompts the user with 9 different text boxes all of which ask the user a different question. These include:

1. Full Name: The user is prompted for their Full Name. The user's full name is used for identification later in the application.
2. Email Address: The user is prompted for their personal or work email address. The user's email address is used for logging into the application as well as for newsletters and urgent information from the application.
3. Grade: The user is prompted for their grade school grade. As this is an application to aid mainly high school and middle school students, the user's grade is important for identification purposes later in the application.
4. Birthday: The user is prompted for their birthday. The user's birthday is used mainly for identification purposes including identifying if the user is age restricted to certain posts, competitions, drawings, etc.
5. School: The user is prompted for their school. The user's school is used to ensure that this user is connected with the correct school or a school of their choice.

6. Security Question: The user is prompted to provide a security question. The user's security question that will be used if the user changes any information regarding their account as well as resetting their password.
7. Security Answer: The user is prompted to provide an answer to their created security question. The user's security question answer is always used in conjunction with the security question, mainly where account information is being changed.
8. Password: The user is prompted to enter a password for their account. The user's password is used every time that user logs into this application as well as with changing any sensitive information throughout the application such as account information.
9. Confirm Password: The user is prompted to confirm their password. The user's confirmed password is compared with the previous field "Password" to ensure that the user both did not make a keystroke mistake and making sure that the user will remember this password as they must enter it twice.

Following answering all prompt fields correctly, the user will have made a new account and will be directed back to the login page to login.



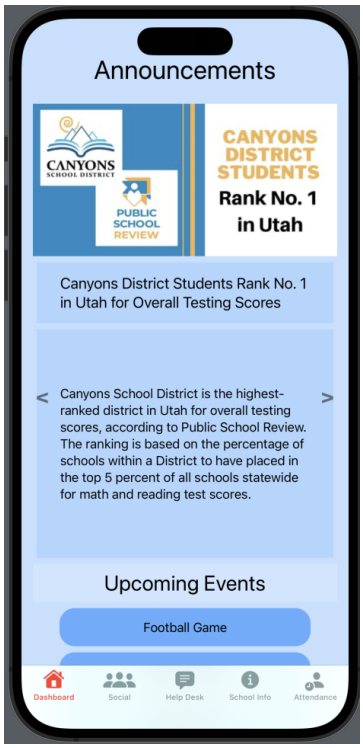
1.4 Dashboard Screen

The dashboard screen appears after the user has logged in. Here the user can find any important information that the school would like to share. This is in the form of a slide show at the top of the screen. Under this slideshow, the user is presented with all upcoming events that

pertain to the school. Furthermore throughout the application there is a tab bar at the bottom of each page which has icons that will light up when a user is on that specific screen.

1.4.1 Slideshow

The slideshow entails a title that briefly describes the announcement, a picture if necessary, and a description section that provides the user with detailed information about the announcement at hand.



1.4.2 Upcoming Events

The upcoming events section includes academic meetings and competitions, athletic competitions and competitions or performances from the performing arts department. When the user clicks on one of these events, the user is presented with a new screen called “Event Screen” that provides information pertaining to the event clicked.

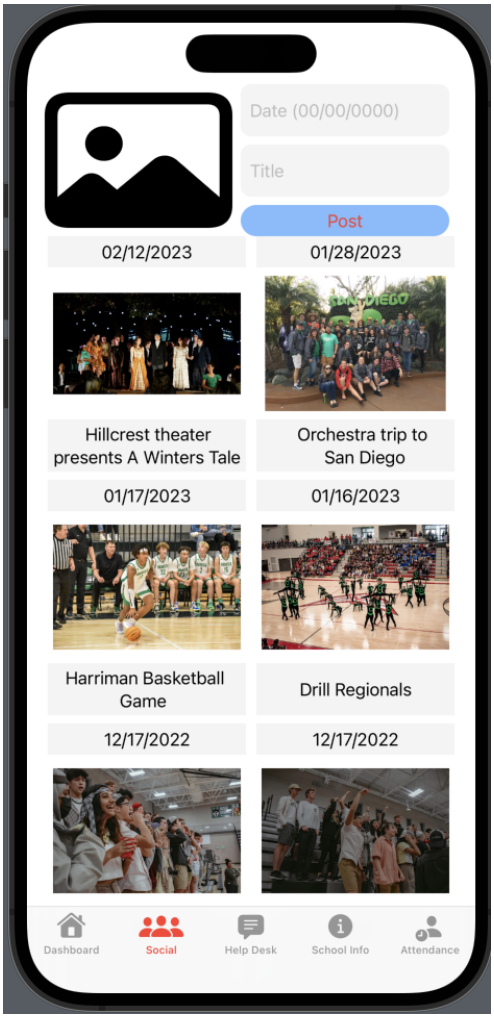
1.4.3 Event Screen

The event screen is enabled if and only if the user has clicked on an upcoming event. This screen provides the user with substantial information about the event including: the event title, the event date, a short description of the event, and the location of the event compared to the user’s highschool. This page emphasized on the location of the event in the form of a map. Lastly at the top of this screen, the user can be redirected back to the user dashboard.

The map entails two pins, one at the user's school location and one at the location of the event. The map is also positioned such that both pins can be seen in comparison to each other.

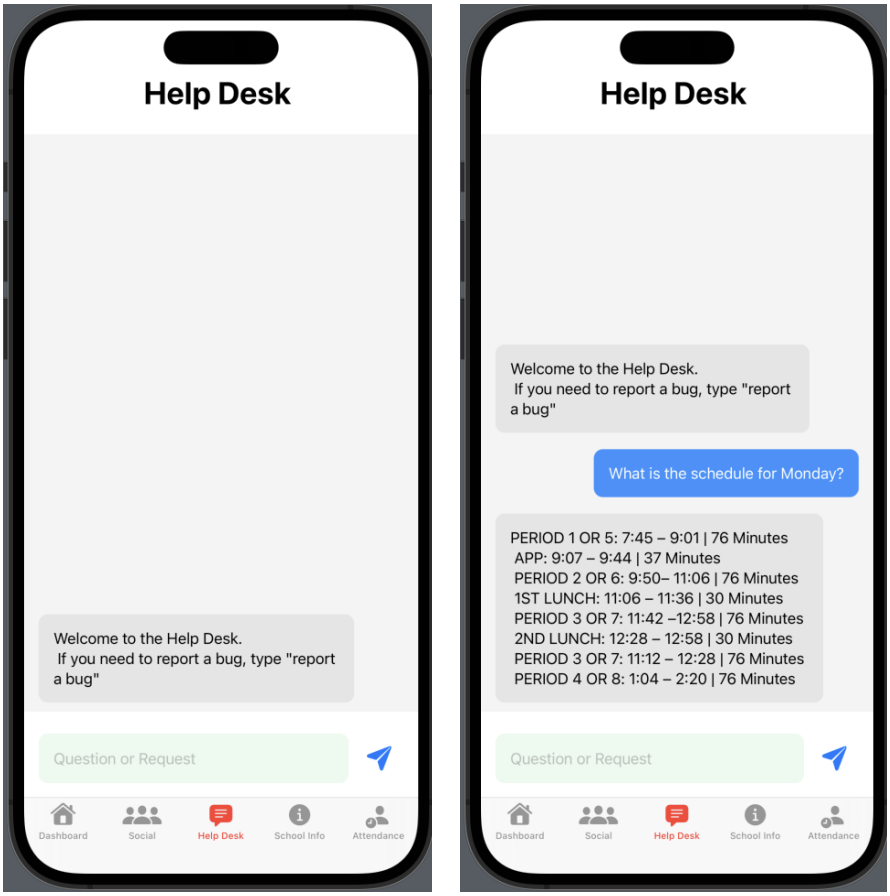
1.5 Social Screen

The social screen is a screen that allows the user to share pictures and text with all other users including parents, students and teachers. This screen has the main functionality of allowing the user to import an image from their phone and display it with a date and description for their image. This screen is similar to a social media platform as it has the ability to connect all of the app's users to one location where they can interact as a community.



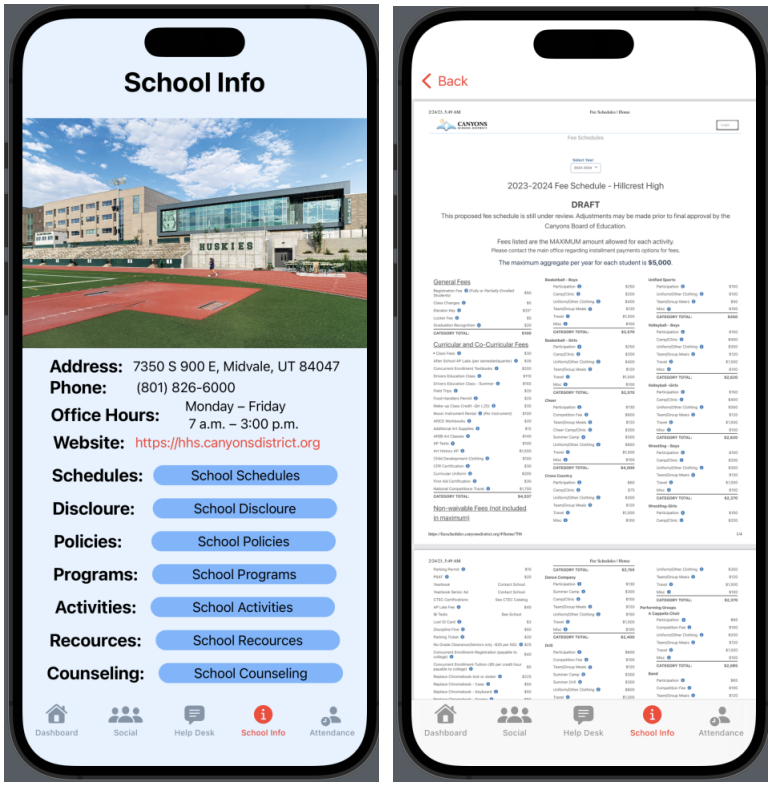
1.6 Help Desk

The help desk is a chatting system for the user to ask questions to either an AI automation or with a school representative. The user can request any information and a representative will get back to that specific user as soon as possible. The help desk also functions as a Bug Reporting System, where the user can report any bugs that they find in the application with the application developers.



1.7 School Info Screen

The school info screen allows the user to access and see any and all information pertaining to the school. This includes: schedules, disclosures, policies, programs, activities, resources, and counseling. Each of these sections is enclosed in a button, and when clicked it will direct the user to a new screen with the respective PDF or photo.

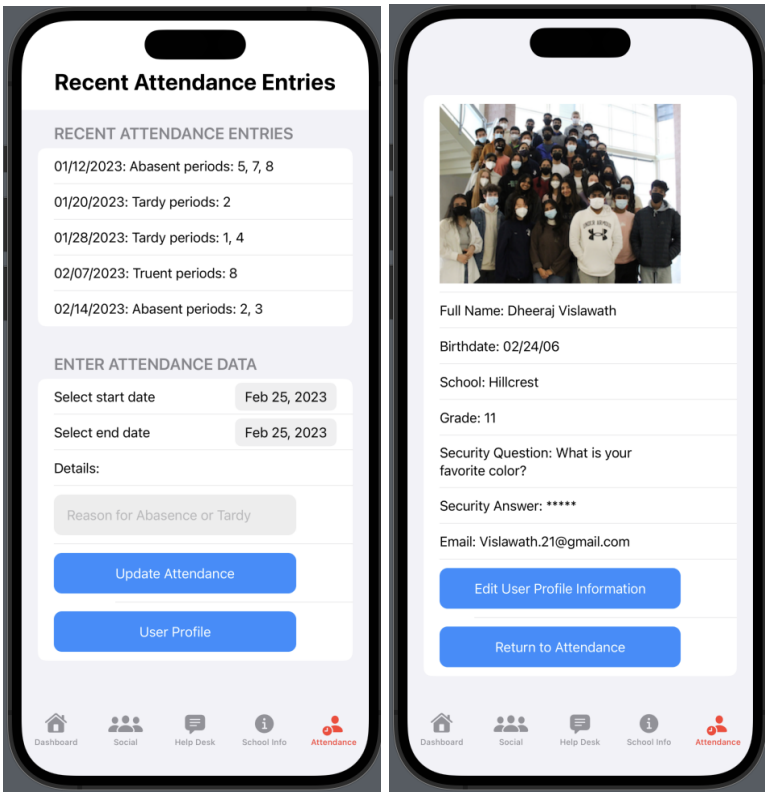


1.8 Attendance Screen

The Attendance screen allows the user to see their recent attendance data entries as well as a reporting system. This reporting system allows the user to enter attendance data such as a parent excused absence. Following this, the user's profile is also at the bottom of this page enclosed in a button.

1.8.1 User Profile Screen

The user profile screen displays all information pertaining to the user's profile including: First Name, Email, Security Question, Security Question Answer, Grade, School, Birthday, and User Profile Picture.



1.9 Usage Tips

In addition to the information regarding various features listed above, there are a handful of useful tips to know when using our application that will vastly improve your experience.

- After filtering for tourist attractions, use the Sort By option in the top right of the main window to sort by a desired attribute. This way, you can further improve your search results by sorting the displayed attractions by highest rating, lowest price, etc.
- Use the my location feature to find attractions close to you.
- Want to keep looking at the search results, but want to mark a specific attraction? Bookmark it, and it will be stored in the bookmarks tab.
- Looking for more in depth information regarding an attraction? Visit its website, found below the map preview on the right-hand side.

2. Source Documentation

2.1 Sources

Time Track uses a variety of different libraries, modules, and resources. The external data and images we use are accredited properly. All images were given credit and copyright laws were followed.

2.2 Image Documentation

Our application makes use of attraction images, application logos and icons.

- Slideshow (Announcements) Images
    - https://www.canyonsdistrict.org/
    - Available to the public for free use
    - All rights belong to original author
- Image Sharing (Social) Images
    - https://hillcresthuskies.com/
    - https://www.hillcrestorchestra.com/
    - https://www.hillcresttheatre.com/
    - Available to the public for free use
    - All rights belong to original author
- School Info PDFs and Images
    - https://hhs.canyonsdistrict.org/
    - Available to the public for free use
    - All rights belong to original author
- Login Icons
    - https://www.flaticon.com/
    - Available to the public for free use
    - All rights belong to original author

2.3 Data Documentation

Information regarding upcoming events and announcements were found as such:

- Announcement Information
    - https://www.canyonsdistrict.org/
    - Available to the public for free use
    - All rights belong to original author

2.4 Code Documentation

Code documentation with comments is a way to provide explanations and details about the code within the code itself. This makes it easier for other developers to understand what the code does, how it works, and how it can be used.

Code Documentation with comments has been done throughout the entirety of the project.

This is an example of code documentation:

```
/*
 Creates a splash screen that is really an annimated screen with text and a logo. Once the time delay finoishes it will bring us to the login screen.
 */

struct SplashScreenView: View
{
    @State private var isActive = false
    @State private var size = 0.8
    @State private var ocacity = 0.5

    var body: some View
    {
        // if isActive is true then it will direct thte user to the login screen.
        if isActive
        {
            LoginView()
        }else{
            ZStack
            {
                Color.blue
                    .ignoresSafeArea(.all)
                VStack
                {
                    VStack
                    {
//                        Image(systemName: "hare.fill")
//                            .font(Font.custom("hare", size: 80))
//                            .foregroundColor(.white)
                        /*
                         Calls the logo and creates the text area with specific pararmeters.
                         */
                        Image("icon")
                        Text("Time Track")
                            .font(Font.custom("Baskerville-Bold", size: 26))
                            .foregroundColor(.black.opacity(0.80))
                    }
                    .scaleEffect(size)
                    .opacity(ocacity)
                    .onAppear
                    {
                        // this is hte annimation part. it will increace by a size of 170 %
                        withAnimation(.easeIn(duration: 1.2))
```

2.5 Licenses

Within our application we used a multitude of different tools, libraries, and resources. Below are the licenses of the tools and products we used.

- **XCode:** Xcode is distributed under Apple's software license agreement, which is a proprietary license. This means that Xcode and its accompanying tools are proprietary software and can only be used under the terms and conditions set forth in the license agreement.

  The license agreement for Xcode includes several restrictions on how the software can be used, including limitations on reverse engineering, decompiling, and disassembling the software. The agreement also prohibits the use of Xcode to develop software for certain purposes, such as nuclear facilities, aircraft navigation or communication systems, or life support machines.

- **OpenAI:** OpenAI provides a range of APIs and software tools for artificial intelligence and machine learning, including natural language processing, computer vision, and reinforcement learning. These APIs and tools are subject to the OpenAI API Terms of Service, which govern the use of OpenAI's services.

  The OpenAI API Terms of Service outline the terms and conditions for using the OpenAI API, including restrictions on how the API can be used, limitations on liability, and provisions for termination of the agreement. The terms of service also include provisions related to intellectual property rights, confidentiality, and data privacy.

  In general, the OpenAI API is available to developers and researchers who agree to the terms of service and meet certain eligibility requirements. Developers are also required to comply with any applicable laws and regulations related to their use of the API.

  For our application we only use language processing to reply to users when problems are raised.

- **Swift:** The license under which Swift 5.7 is distributed is the Apache License, Version 2.0. This is a permissive open-source license that allows users to use, modify, and distribute the software for both commercial and non-commercial purposes.

  It also includes a patent license that grants users the right to use any patents owned by the licensor that are necessary to use, modify, or distribute the software. The full text of the Apache License, Version 2.0 can be found at the following link:

  https://www.apache.org/licenses/LICENSE-2.0

- **SwiftUI:** SwiftUI is distributed as part of Apple's Xcode development environment and is subject to the terms and conditions of the Apple Developer Agreement and the Apple Public Source License Version 2.0. This license allows developers to use SwiftUI and any accompanying source code for the purpose of developing and distributing applications for Apple's operating systems, including iOS, macOS, watchOS, and tvOS.

  The license also allows for modification and distribution of the source code as long as certain conditions are met, such as maintaining attribution and copyright notices. The full text of the Apple Public Source License Version 2.0 can be found at the following link:

  https://opensource.apple.com/license/apsl/

3. System Requirements and Setup

The Xcode requirements are:

1. A Mac running macOS Big Sur 11.0 or later.
2. At least 8GB of RAM, but 16GB or more is recommended.
3. At least 15GB of available disk space, but 50GB or more is recommended.
4. Xcode 13 or later, which can be downloaded for free from the Mac App Store or from the Apple Developer website.

3.1 iOS

To run an Xcode project on an iOS device, you will need:

1. A Mac computer running macOS operating system.
2. Xcode installed on the Mac computer. Xcode is an integrated development environment (IDE) that allows developers to create, debug, and test their iOS applications.
3. An iOS device that supports the minimum deployment target specified in the Xcode project. The device should also be connected to the Mac computer through a USB cable.
4. A valid Apple Developer account that allows you to sign the app with a distribution certificate and provisioning profile. You will need to create a certificate and a profile for your app in the Apple Developer portal.
5. The correct target selected in the Xcode project. You can select the target from the dropdown menu next to the scheme in the Xcode toolbar.
6. A valid code signing identity for the selected target. You can select the code signing identity from the project settings.
7. A valid provisioning profile for the selected target. You can select the provisioning profile from the project settings.
8. Any additional dependencies or frameworks required by the Xcode project.

Once you have all the necessary requirements, you can build and run the Xcode project on your iOS device by selecting the "Run" command from the Xcode toolbar. The app will be compiled, signed, and installed on your device for testing.

3.2 iPadOS

To run an Xcode project on an iPad running iPadOS, you will need:

1. A Mac computer running macOS operating system.
2. Xcode installed on the Mac computer. Xcode is an integrated development environment (IDE) that allows developers to create, debug, and test their iOS and iPadOS applications.
3. An iPad running iPadOS 13 or later. The iPad should be connected to the Mac computer through a USB cable or through the network using Xcode's wireless debugging feature.

4. A valid Apple Developer account that allows you to sign the app with a distribution certificate and provisioning profile. You will need to create a certificate and a profile for your app in the Apple Developer portal.
5. The correct target selected in the Xcode project. You can select the target from the dropdown menu next to the scheme in the Xcode toolbar.
6. A valid code signing identity for the selected target. You can select the code signing identity from the project settings.
7. A valid provisioning profile for the selected target. You can select the provisioning profile from the project settings.
8. Any additional dependencies or frameworks required by the Xcode project.

Once you have all the necessary requirements, you can build and run the Xcode project on your iPad by selecting the "Run" command from the Xcode toolbar. The app will be compiled, signed, and installed on your iPad for testing.

Note that the iPad must be in developer mode, which can be enabled by going to the Settings app, selecting "Developer", and toggling on "Enable UI Automation". Additionally, your iPad may need to be added to your developer account as a test device before you can run your Xcode project on it. This can be done in the Apple Developer portal under the "Devices" section.

3.3 tvOS

To run an Xcode project on an Apple TV running tvOS, you will need:

1. A Mac computer running macOS operating system.
2. Xcode installed on the Mac computer. Xcode is an integrated development environment (IDE) that allows developers to create, debug, and test their tvOS applications.
3. An Apple TV running tvOS 13 or later. The Apple TV should be connected to the Mac computer through a USB cable or through the network using Xcode's wireless debugging feature.
4. A valid Apple Developer account that allows you to sign the app with a distribution certificate and provisioning profile. You will need to create a certificate and a profile for your app in the Apple Developer portal.
5. The correct target selected in the Xcode project. You can select the target from the dropdown menu next to the scheme in the Xcode toolbar.
6. A valid code signing identity for the selected target. You can select the code signing identity from the project settings.
7. A valid provisioning profile for the selected target. You can select the provisioning profile from the project settings.
8. Any additional dependencies or frameworks required by the Xcode project.

Once you have all the necessary requirements, you can build and run the Xcode project on your Apple TV by selecting the "Run" command from the Xcode toolbar. The app will be compiled, signed, and installed on your Apple TV for testing.

Note that your Apple TV may need to be added to your developer account as a test device before you can run your Xcode project on it. This can be done in the Apple Developer portal under the "Devices" section. Additionally, if you are using wireless debugging, both your Mac computer and Apple TV must be on the same Wi-Fi network.

3.4 watchOS

To run an Xcode project on an Apple Watch running watchOS, you will need:

1. A Mac computer running macOS operating system.
2. Xcode installed on the Mac computer. Xcode is an integrated development environment (IDE) that allows developers to create, debug, and test their watchOS applications.
3. An Apple Watch running watchOS 6 or later. The Apple Watch should be paired with the Mac computer via Bluetooth and have developer mode enabled.
4. A valid Apple Developer account that allows you to sign the app with a distribution certificate and provisioning profile. You will need to create a certificate and a profile for your app in the Apple Developer portal.
5. The correct target selected in the Xcode project. You can select the target from the dropdown menu next to the scheme in the Xcode toolbar.
6. A valid code signing identity for the selected target. You can select the code signing identity from the project settings.
7. A valid provisioning profile for the selected target. You can select the provisioning profile from the project settings.
8. Any additional dependencies or frameworks required by the Xcode project.

Once you have all the necessary requirements, you can build and run the Xcode project on your Apple Watch by selecting the "Run" command from the Xcode toolbar. The app will be compiled, signed, and installed on your Apple Watch for testing.

Note that to enable developer mode on your Apple Watch, you will need to go to the Watch app on your paired iPhone, select "General", then "About", and tap on the "Build Number" seven times. After doing this, you should see a new "Developer" menu option in the Watch app. In the "Developer" menu, you can enable options such as "Show on Apple Watch" and "Remote Logging". Additionally, your Apple Watch may need to be added to your developer account as a test device before you can run your Xcode project on it. This can be done in the Apple Developer portal under the "Devices" section.

4. Requirements and Accessibility

4.1 Application Required Programs

XCode:

Xcode is an integrated development environment (IDE) for developing iOS, macOS, watchOS, and tvOS apps. Here are the steps to download and use Xcode:

1. Go to the Mac App Store
2. Search for "Xcode" in the search bar
3. Click on "Get" to download Xcode
4. Enter your Apple ID credentials if prompted
5. Wait for the download to complete

Once Xcode is downloaded and installed on your Mac, you can start using it to develop iOS, macOS, watchOS, and tvOS apps. Here are the steps to create a new project in Xcode:

1. Open Xcode from your Applications folder
2. Click on "Create a new Xcode project"
3. Choose the appropriate template for your project (e.g. "iOS App", "macOS App", etc.)
4. Enter a name for your project and other necessary information
5. Click on "Create"

Xcode will create a new project with some default settings and files. From there, you can start editing the code, adding new files, and testing your app in the simulator.

To run your app on a physical device, you will need to have a valid Apple Developer account and connect your device to your Mac using a cable. You can then select your device from the list of available devices in Xcode and run your app on it.

Xcode also includes various debugging and profiling tools to help you identify and fix issues in your code. You can use these tools to inspect variables, view log output, and analyze performance metrics.

Overall, Xcode is a powerful IDE for developing iOS, macOS, watchOS, and tvOS apps, and it provides everything you need to create, test, and debug your code.

Simulators

To download simulators in Xcode, follow these steps:

1. Open Xcode on your Mac.
2. Click on the "Xcode" menu in the top left corner of the screen.

3. Select "Preferences" from the drop-down menu.
4. In the Preferences window, click on the "Components" tab.
5. Find "Simulators" in the list of components and click the "Install" button next to it.
6. The simulators will begin downloading and installing automatically.
7. Once the download and installation are complete, you can access the simulators from within Xcode by selecting "Window" from the top menu and then selecting "Devices and Simulators".
8. In the Devices and Simulators window, you will see a list of available simulators. Select the one you want to use and click the "Run" button to launch it.

Note that you can also download additional simulators from the Components tab in Xcode Preferences, such as older versions of iOS or watchOS simulators. Simply select the desired simulator and click the "Install" button next to it.

4.2 Packages and Modules

OpenAI API

To access the OpenAI API, you need to follow these steps:

1. Create an OpenAI account: Go to the OpenAI website (https://openai.com/) and create an account.

2. Generate API key: Once you have created an account, go to the API keys section of the OpenAI dashboard (https://beta.openai.com/dashboard/api-keys). Click on the "Generate New Key" button and give a name to the key.

3. Choose the API endpoint: OpenAI offers several APIs for different purposes. Choose the API endpoint that best suits your needs. You can find a list of OpenAI APIs on the OpenAI website.

4. Install an API client: To use the OpenAI API, you need an API client. OpenAI provides API clients for several programming languages, including Python, Java, and JavaScript. Install the API client for your preferred language.

Use the API: Once you have installed the API client, you can use it to interact with the OpenAI API. The API client provides functions to send requests to the OpenAI API endpoints and receive responses.

It's important to note that some OpenAI API endpoints require a specific subscription plan, and there may be limits on the number of requests you can make per month. Make sure to review the OpenAI API documentation and pricing information before integrating the API into your project.

SwiftUI
You can download the latest version of Swift from the official Swift website: https://swift.org/download/

On the website, you'll see different download options depending on your platform (macOS, Ubuntu, etc.). Select the appropriate option for your system and follow the installation instructions.

For macOS, you can download the Swift toolchain package, which includes the Swift compiler, libraries, and REPL. Once you've downloaded the package, double-click it to extract its contents, then drag the Swift folder to your Applications directory.

You can also download Xcode, which includes the Swift compiler and other development tools. Xcode is available on the Mac App Store. Once you've installed Xcode, you can start using Swift by creating a new Swift project or playground.

UIKit
UIKit is a built-in framework in iOS and is included with Xcode. To start using UIKit in your iOS app, follow these steps:

1. Open Xcode on your Mac.
2.
3. Create a new project or open an existing project.
4.
5. In the left-hand panel, click on the project file to open the project settings.
6.
7. Click on the "General" tab and scroll down to the "Frameworks, Libraries, and Embedded Content" section.
8.
9. Click on the "+" button and select "UIKit.framework" from the list of available frameworks.
10.
11. Choose whether you want to link the framework "Embed & Sign" or "Do Not Embed".
12.
13. Click "Finish" to add the framework to your project.

Once you have added the UIKit framework to your project, you can start using its classes and methods in your code. Make sure to import the UIKit module in any Swift file where you want to use it:

MapKit
MapKit is a framework provided by Apple and is included in Xcode. Therefore, to use MapKit, you don't need to download or install it separately. You only need to have Xcode installed on your Mac.

Here are the steps to get started with MapKit in Xcode:

1. Open Xcode on your Mac.
2. Click on "Create a new Xcode project".
3. Select "App" under the "iOS" tab on the left-hand side of the screen.
4. Choose the template you want to use, such as "Single View App" or "Tabbed App".
5. Click "Next" and give your project a name and a location to save it.
6. Choose the language as "Swift" from the options given.
7. Click "Next" and then "Create".
8. Once your project is created, you can start using MapKit by adding it to your project. To add MapKit, click on your project in the Project navigator and select your app target. Under the "General" tab, scroll down to the "Frameworks, Libraries, and Embedded Content" section, click the "+" button, and search for "MapKit". Select it and click "Add".

You can now start using MapKit in your project by importing it in your code and using its classes and functions.
Note: MapKit requires that your app has a valid Apple Developer Program account and a Maps API key. You can get an API key from the Apple Developer website and add it to your app's Info.plist file.

4.4 Accessing Source Code

The Time Track App source code can be found at our github:

https://github.com/WallacexMcCarthy/TimeTrackApp

To get source code from Github, follow these steps:

1. Go to the Github website and search for the repository that contains the source code you want to download.
2. Once you have found the repository, click on it to view its contents.
3. Navigate to the directory or file that you want to download.
4. Click on the "Code" button located on the right-hand side of the screen.
5. Click on the "Download ZIP" option to download the repository as a ZIP file.
6. Save the ZIP file to your computer.
7. Extract the contents of the ZIP file to a directory on your computer.

You now have the source code on your local machine and can use it as needed.
Alternatively, if you have Git installed on your computer, you can clone the repository using the Git command line interface. To do this, follow these steps:

1. Open a terminal or command prompt on your computer.
2. Navigate to the directory where you want to clone the repository.
3. Run the following command: git clone <repository URL>
4. Replace <repository URL> with the URL of the repository you want to clone.
5. Press Enter and wait for Git to download the repository.

Once Git has finished cloning the repository, you will have the source code on your local machine and can use it as needed.

5. Program Architecture

5.1 Program Structure

Models
- ➢ User
- ➢ Event
- ➢ Post
- ➢ Attendance

Views
- ➢ Login View
- ➢ Registration View
- ➢ Dashboard View
- ➢ Event Details View
- ➢ Social View
- ➢ Help Desk View
- ➢ School Info View
- ➢ Attendance View
- ➢ Profile View

View Models
- ➢ Login View Model
- ➢ Registration View Model
- ➢ Dashboard View Model
- ➢ Event Details View Model
- ➢ Social View Model
- ➢ Help Desk View Model
- ➢ School Info View Model
- ➢ Attendance View Model
- ➢ Profile View Model

Controllers
- ➢ Login Controller
- ➢ Registration Controller
- ➢ Dashboard Controller
- ➢ Event Details Controller
- ➢ Social Controller
- ➢ Help Desk Controller
- ➢ School Info Controller
- ➢ Attendance Controller
- ➢ Profile Controller

Services
- ➢ Authentication Service

- ➢ Event Service
- ➢ Post Service
- ➢ Attendance Service
- ➢ OpenAI Service

Utilities

- ➢ Networking Utils
- ➢ Date Utils

The app has the following features:

1. Login and Registration: The user can log in or create a new account to access the app.
2. Dashboard: The user can view a list of upcoming events on their dashboard.
3. Event Details: The user can view details about an event when they click on it from the dashboard.
4. Social Page: The user can share text and photos with others.
5. Help Desk: The user can get help from an OpenAI chat bot for any questions or issues they have.
6. School Info: The user can access links to relevant school information, such as PDFs and URLs.
7. Attendance: The user can view and change their attendance data.
8. User Profile: The user can view and edit their profile information.

The app will use several services, including an authentication service for login and registration, an event service to retrieve and display event information, a post service to allow users to share text and photos, an attendance service to allow users to view and update their attendance data, and an OpenAI service to provide a chat bot help desk.

Each feature will have its own view, view model, and controller to handle user interaction and data flow. Additionally, there will be various utility classes to handle networking and date-related operations

5.2 Swift and SwiftUI

Swift is a programming language developed by Apple for building software applications for its platforms, including iOS, iPadOS, macOS, watchOS, and tvOS. Swift was first introduced in 2014 and has since become a popular choice for developers due to its modern syntax, performance, and safety features.

Swift was designed to be easy to learn and use, while also providing advanced features for building complex software applications. Some of the key features of Swift include:

1. Safety: Swift is designed to prevent common programming errors, such as null pointer exceptions and array out-of-bounds errors, through features like optionals and safe memory management.

2. Speed: Swift is designed to be fast, with performance on par with C++ and Objective-C. Swift uses advanced optimization techniques to generate efficient machine code.

3. Interoperability: Swift is designed to work seamlessly with Objective-C, which means that developers can use Swift and Objective-C code in the same project.

4. Modern syntax: Swift has a modern, concise syntax that is easy to read and write. It features a range of advanced language constructs, such as closures and type inference.

5. Open source: Swift is an open-source programming language, which means that developers can contribute to its development and use it on non-Apple platforms.

Overall, Swift is a powerful and flexible programming language that can be used to build a wide range of software applications, from simple mobile apps to complex server-side applications.

SwiftUI is a user interface (UI) framework developed by Apple for building applications across all of its platforms, including iOS, iPadOS, macOS, watchOS, and tvOS. It was introduced in 2019 as part of the release of Xcode 11 and Swift 5.

SwiftUI is built using the Swift programming language and provides a declarative approach to building user interfaces. This means that instead of writing imperative code to create and update UI elements, you describe what the UI should look like and let the framework take care of the rest. SwiftUI uses a variety of concepts and features, such as property wrappers and modifiers, to make it easy to create complex, responsive, and visually appealing interfaces with minimal code.

Some features of SwiftUI include:

1. Rapid prototyping: SwiftUI provides a live preview feature that allows you to see the changes you make to your UI in real-time.
2. Minimal code: SwiftUI provides a concise syntax and a range of built-in components that can help you write less code.
3. Cross-platform support: SwiftUI allows you to build user interfaces that can run on multiple Apple platforms without significant changes.
4. Accessibility: SwiftUI makes it easier to build interfaces that are accessible to all users, including those with disabilities.

5.3 Module Purposes

5.3.1 OpenAI API

The OpenAI API is a powerful artificial intelligence platform that provides developers with access to cutting-edge natural language processing (NLP) models. The API allows developers to integrate state-of-the-art language models into their applications, enabling them to perform tasks such as language translation, sentiment analysis, and text generation.

The purpose of this module is to provide an overview of the OpenAI API and its capabilities. The module will cover the following topics:

1. Introduction to the OpenAI API: This section will provide an overview of the OpenAI API and its key features. It will explain how the API works, what types of models are available, and how developers can use the API to build powerful NLP applications.

2. Getting Started with the OpenAI API: This section will walk developers through the process of getting started with the OpenAI API. It will cover topics such as setting up an account, creating an API key, and making API requests.

3. Using the OpenAI API: This section will provide examples of how developers can use the OpenAI API to perform common NLP tasks. It will cover topics such as language translation, sentiment analysis, and text generation, and will provide code snippets and examples of how to use the API in practice.

4. Advanced Topics: This section will cover more advanced topics related to the OpenAI API, such as custom training of models, fine-tuning pre-trained models, and optimizing API performance.

By the end of this module, developers should have a good understanding of the OpenAI API and how it can be used to build powerful NLP applications. They should be able to create an account, set up an API key, and make API requests, as well as perform common NLP tasks such as language translation and sentiment analysis. They should also be familiar with more advanced topics such as custom training and fine-tuning of models, and be able to optimize API performance for their specific use case.

5.3.2 SwiftUI

SwiftUI is a modern user interface framework for building applications for Apple's platforms, including iOS, iPadOS, macOS, watchOS, and tvOS. The purpose of this module is to provide an introduction to SwiftUI and its capabilities, and to help developers get started with building user interfaces using SwiftUI.

The module will cover the following topics:

1. Introduction to SwiftUI: This section will provide an overview of SwiftUI and its key features. It will explain how SwiftUI differs from traditional user interface frameworks, and what benefits it offers to developers.

2. Getting Started with SwiftUI: This section will walk developers through the process of creating a basic SwiftUI application. It will cover topics such as creating a new project, creating a user interface using SwiftUI, and adding functionality to the user interface.

3. User Interface Components: This section will cover the various user interface components available in SwiftUI, such as buttons, labels, text fields, and lists. It will provide examples of how to use these components to create a variety of user interfaces.

4. Layout and Navigation: This section will cover how to layout user interface components in SwiftUI, and how to navigate between different screens in an application.

5. Advanced Topics: This section will cover more advanced topics related to SwiftUI, such as animations, custom components, and integrating with other frameworks and libraries.

By the end of this module, developers should have a good understanding of SwiftUI and its capabilities, and should be able to create basic user interfaces using SwiftUI. They should also be familiar with the various user interface components available in SwiftUI, and be able to layout and navigate between different screens in an application. Additionally, they should be familiar with more advanced topics such as animations and custom components, and be able to integrate SwiftUI with other frameworks and libraries as needed.

5.3.3 UIKit

UIKit is a powerful user interface framework for building applications for Apple's platforms, including iOS, iPadOS, and macOS. The purpose of this module is to provide

an introduction to UIKit and its capabilities, and to help developers get started with building user interfaces using UIKit.

The module will cover the following topics:

1. Introduction to UIKit: This section will provide an overview of UIKit and its key features. It will explain how UIKit is used to create user interfaces, and what benefits it offers to developers.

2. Getting Started with UIKit: This section will walk developers through the process of creating a basic UIKit application. It will cover topics such as creating a new project, creating a user interface using Interface Builder, and adding functionality to the user interface.

3. User Interface Components: This section will cover the various user interface components available in UIKit, such as buttons, labels, text fields, and table views. It will provide examples of how to use these components to create a variety of user interfaces.

4. Layout and Navigation: This section will cover how to layout user interface components in UIKit, and how to navigate between different screens in an application.

5. Advanced Topics: This section will cover more advanced topics related to UIKit, such as animations, custom components, and integrating with other frameworks and libraries.

By the end of this module, developers should have a good understanding of UIKit and its capabilities, and should be able to create basic user interfaces using UIKit. They should also be familiar with the various user interface components available in UIKit, and be able to layout and navigate between different screens in an application. Additionally, they should be familiar with more advanced topics such as animations and custom components, and be able to integrate UIKit with other frameworks and libraries as needed.

5.3.4 MapKit

MapKit is a framework provided by Apple that allows developers to integrate maps into their iOS and macOS applications. The purpose of this module is to provide an introduction to MapKit and its capabilities, and to help developers get started with building map-based applications using MapKit.

The module will cover the following topics:

1. Introduction to MapKit: This section will provide an overview of MapKit and its key features. It will explain how MapKit is used to display maps in an application, and what benefits it offers to developers.

2. Getting Started with MapKit: This section will walk developers through the process of creating a basic MapKit application. It will cover topics such as creating a new project, adding a map to the user interface, and customizing the map display.

3. Annotations and Overlays: This section will cover how to add annotations to a map using MapKit, such as pins and callouts. It will also cover how to add overlays, such as polygons and circles, to a map.

4. User Interaction: This section will cover how to allow users to interact with the map in various ways, such as panning, zooming, and selecting annotations.

5. Advanced Topics: This section will cover more advanced topics related to MapKit, such as geocoding and reverse geocoding, routing and directions, and integrating with other location-based services.

By the end of this module, developers should have a good understanding of MapKit and its capabilities, and should be able to create basic map-based applications using MapKit. They should also be familiar with adding annotations and overlays to a map, allowing user interaction with the map, and using more advanced features such as geocoding and routing. Additionally, they should be able to integrate MapKit with other location-based services as needed.

6. Version History

6.1 Planning Process

Prompt

Create a mobile application for your school to help keep parents and the community up to date. The app needs to include upcoming events, important information such as school calendar and activities schedule, a way for teachers and students to share photos, and a way for parents to notify school of student absences. The app must also include one additional item that is recommended by your administration.

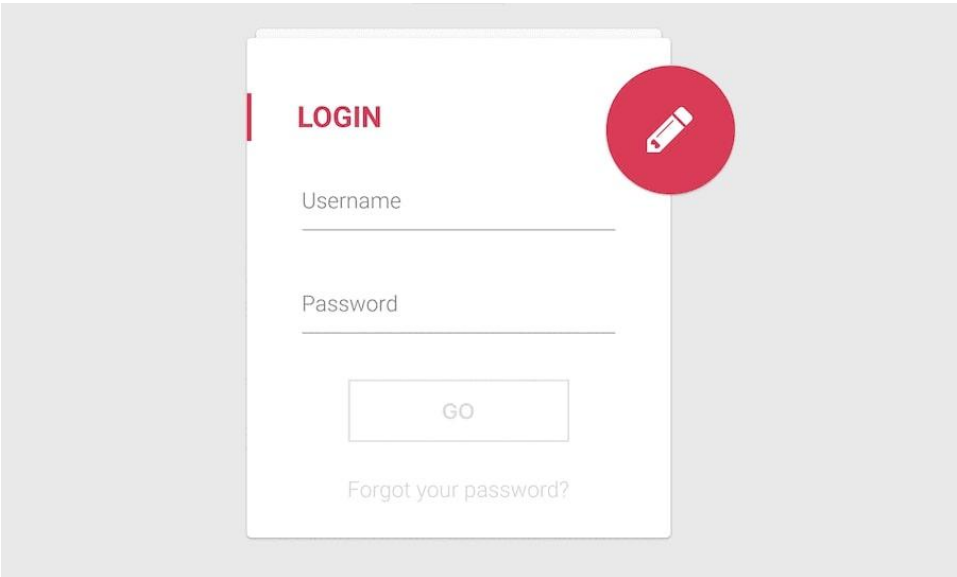
- The app must be designed for a phone/tablet.
- The operating system must be mobile based such as Android or iOS.
- The app should state its licensing and terms of use.

In order to accomplish this prompt, we just add the following Screens and Functionalities: Login Screen, Dashboard Screen, In App Messages, School Information Screen, and an Attendance Screen

Login

In order to develop a professional, industry leading, app, one must create a login screen to allow users to have separate accounts. In this App, there will be a login screen that requests an email and a password to login. If either of these fields are incorrect, the app will highlight them and display an error message. Lastly there will also be a “Create Account” Button that will allow the user to create a new account.

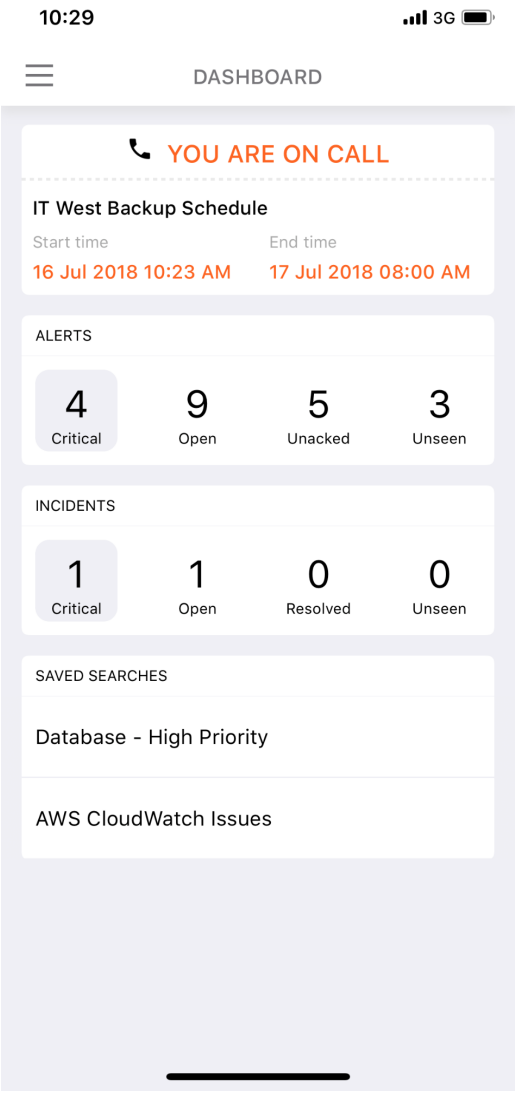
The Login Screen will look something like the following:



Dashboard

To fulfill the task of creating a mobile application for your school to help keep parents and the community up to date, we must have a dashboard that can display important information. This dashboard will display any upcoming events, important dates, as well as any announcements that the school would need to share.

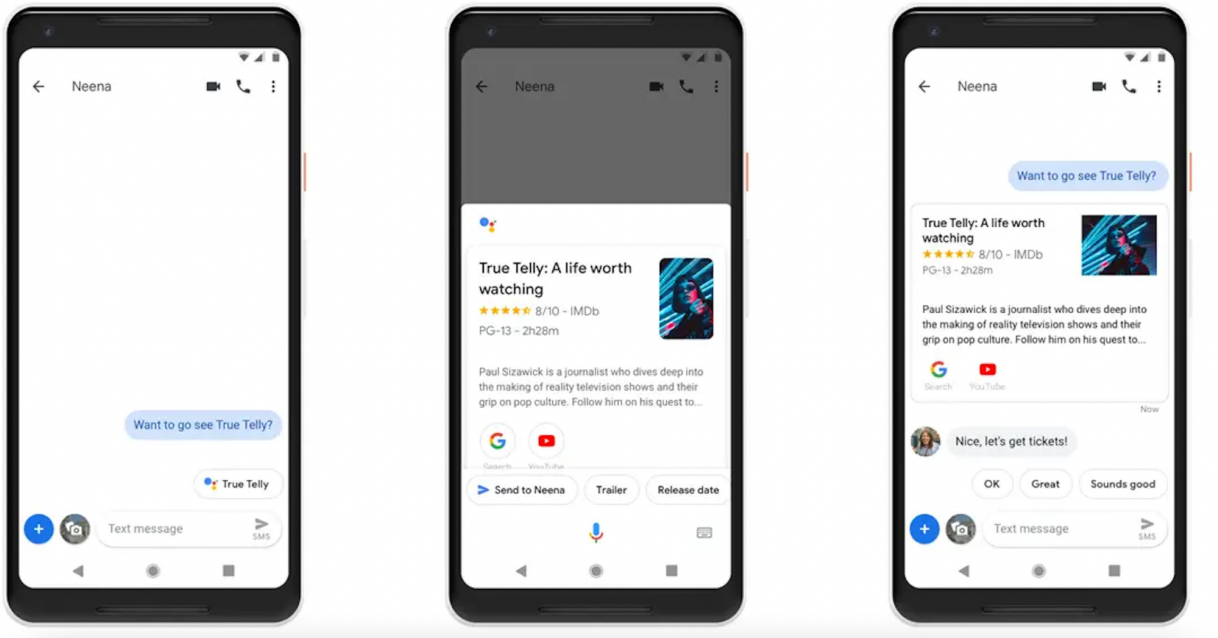
The Dashboard Screen will look something like the following:



Here the User will be able to see any relevant information about the current day and upcoming days. Following that, the user will be able to see information like alerts and announcements. Lastly, the user will be able to see any “High Priority” events or dates / deadlines.

In App Messages

This App will fulfill the prompt’s image sharing of students, teachers and parents. Here there will be an in-app messaging platform that will allow the parents, students and teachers to message each other as well as share a public chat, images and any accompanying text. Here each individual will be able to send messages to any specific person or group of people (teachers, students and parents included). These messages can include images, links, or plain text. The in app messaging page will look something like the following:

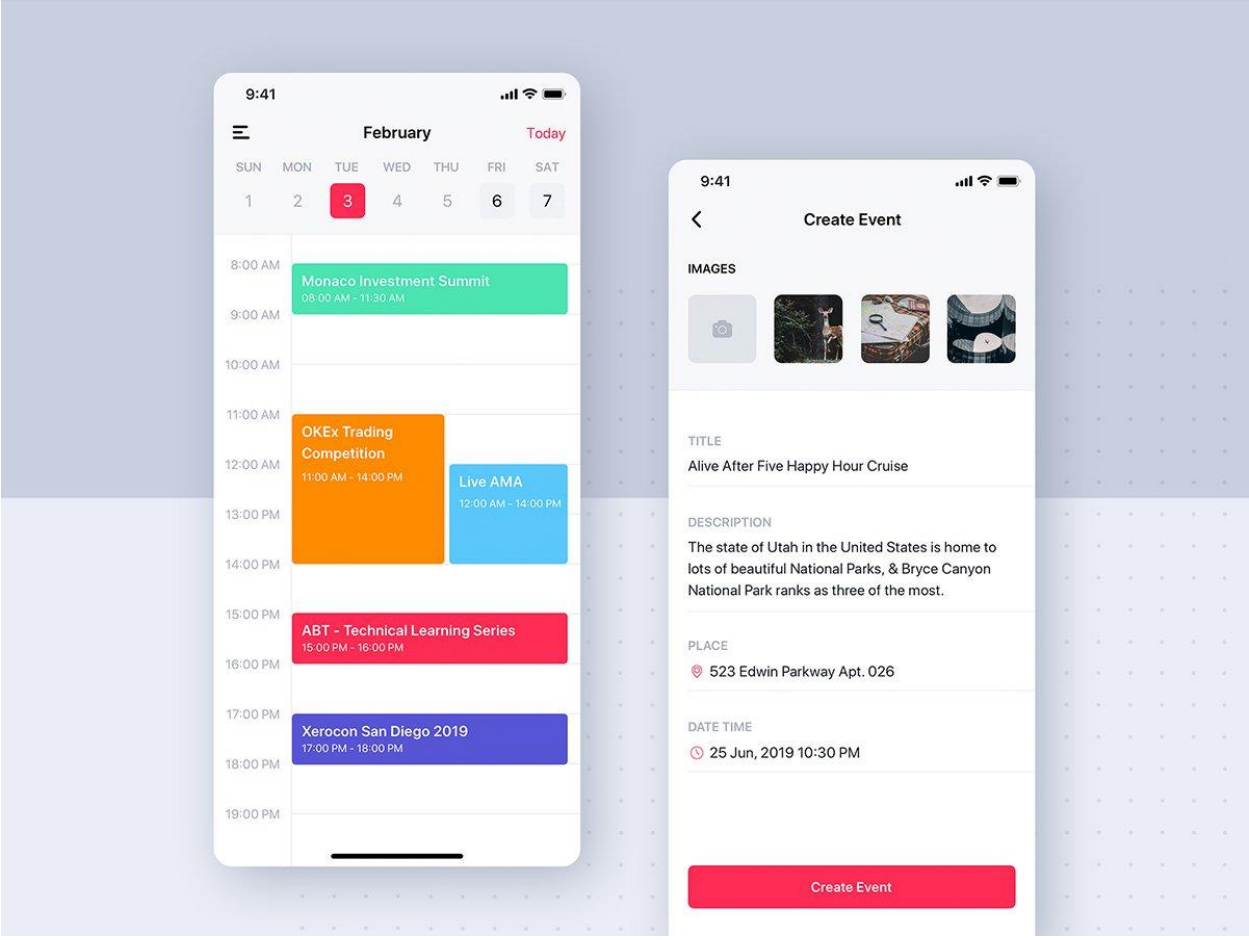


This will essentially be in the form of IMessages or WhatsApp messages, allowing a user to send video, images, and text at will.

School Info

This page will have all important info relating to the school and relative dates. Here there will be a school calendar which will show all upcoming events for a specific month as well as any important dates. Here there will also be important links and schedules and pdfs that would be important to a user. Furthermore, in this page there will be activity schedules and anything pertaining to school activities such as sports.

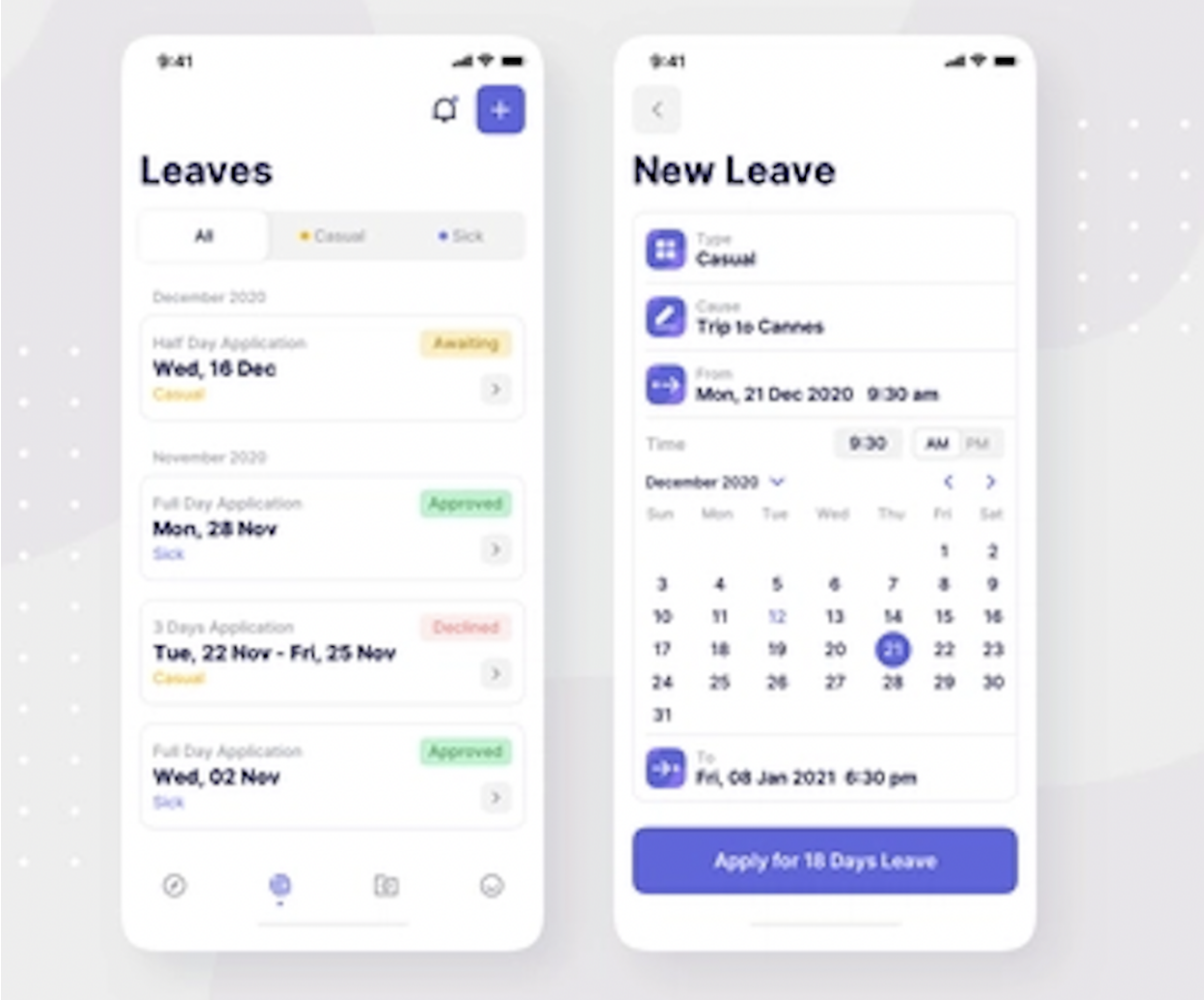
This page will look something like the following:



Attendance Page

The Last page of this app will include anything pertaining to absences. Here there will be date pickers for adding attendance data for students. There will also be a log of attendance data that the student will have, showing all attendance data from the past 5 - 10 occurrences. Following this, the user profile will also be held within this page.

This page will look something like the following:



6.2 Previous Versions

The mobile application offers a variety of features that cater to the user's needs. With the login and registration feature, users can easily access the app by logging in with their existing account or creating a new one.

The dashboard feature provides a list of upcoming events that are important to the user. This way, they can easily keep track of their schedule and plan their activities accordingly. The event details feature allows users to view additional information about a specific event by clicking on it from the dashboard.

The school info feature provides access to essential school information, such as PDFs and URLs, so that parents and the community can stay informed about the school's latest news and events.

The attendance feature allows users to keep track of their attendance data and make changes if necessary, ensuring that they stay up to date with their academic progress.

Finally, the user profile feature enables users to view and edit their personal information, ensuring that they have complete control over their data.

6.3 Current Version

The app provides a comprehensive set of features to enhance the user experience. The user can either log in with their existing account or create a new one to access the app.

- The dashboard allows the user to view a list of upcoming events, enabling them to plan and schedule their activities accordingly. By clicking on an event, the user can view additional details about it, such as date, time, location, and description.
- The social page feature provides an interactive platform for users to share text and photos with others, fostering a sense of community among app users.
- To ensure a seamless user experience, the app includes an OpenAI-powered chatbot help desk. The chatbot provides users with instant assistance and addresses any queries they may have.
- The school info feature provides easy access to links to relevant school information, such as PDFs and URLs, ensuring users are always up to date with the latest school-related news and events.
- The attendance feature enables users to view and update their attendance data, allowing them to keep track of their academic progress and ensuring they are aware of their attendance status.

Finally, the user profile feature enables users to view and edit their profile information, ensuring that they have complete control over their personal data.

To ensure seamless functioning of each feature, the app will use a range of services, including an authentication service for login and registration, an event service to retrieve and display event information, a post service to allow users to share text and photos, an attendance service to

allow users to view and update their attendance data, and an OpenAI service to provide a chatbot help desk.

Each feature will have its own view, view model, and controller to handle user interaction and data flow, ensuring that the app operates smoothly and provides a seamless user experience. Additionally, various utility classes will handle networking and date-related operations, ensuring that the app is up to date and performs optimally.

6.4 Future Versions

Services:

- ➢ Authentication Service
- ➢ Event Service
- ➢ Activity Service
- ➢ Photo Service
- ➢ Absence Service

Utilities:

- ➢ Networking Utilities
- ➢ Date Utilities

The future version of the app will have the following features:

1. Login: The user can log in to the app to access its features.
2. Dashboard: The user can view a list of upcoming events and important information about the school, such as the school calendar and activities schedule.
3. Event Details: The user can view details about an event when they click on it from the dashboard.
4. Activity Schedule: The user can view the schedule of school activities and events.
5. Photo Gallery: The user can view and share photos of school activities and events.
6. Absence Reporting: Parents can notify the school of student absences through the app.

The app will use several services, including an authentication service for login, an event service to retrieve and display event information, an activity service to display the school's activities schedule, a photo service to allow teachers and students to share photos, and an absence service to allow parents to report student absences.

Each feature will have its own view, view model, and controller to handle user interaction and data flow. Additionally, there will be various utility classes to handle networking and date-related operations.

Future versions of the app could include additional features, such as:

- ➢ Push Notifications: Users can receive push notifications about important updates and announcements from the school.
- ➢ Parent-Teacher Communication: Parents can communicate with their child's teacher through the app.
- ➢ Grades and Assignments: Parents and students can view grades and assignments on the app.