# School’s mobile app “iamswau”

Hiram Eliezer Hernandez Garcia

Department of Computer Science,

Southwestern Adventist University

**Spring 2022**

**ABSTRACT: THIS PROJECT WAS DESIGNED TO FACILITATE STUDENTS AND STAFF OF SOUTHWESTERN ADVENTIST UNIVERSITY THE INFORMATION ABOUT DIFFERENT EVENTS AND SERVICES OFFERED TO THEM. THE BASE OF THIS APP IS FLUTTER WHICH HANDLES THE POSTS, USERS, AND FILES SUBMITTED BY THE ADMINS.**

# Index

# 1. Project's Background

## 1.1 Summary and Statement of the problem

When I joined Southwestern Adventist University in 2018, there was a mobile app available but I did not find out about it after two months of classes. The app was not working like it was meant to work and it had a lot of broken things. At that moment I said to myself that I could fix it and make the mobile app even better.

The mobile app at that moment could interact with different pages but each had its issues like the events if there were no events to show, the loading screen would never go away. To check the news around campus but some of the pictures did not fit the news widget. The pages to see the staff directory and the students mugshots looked the same and there was not a search function. There was a timeclock page that anyone could use without the required permission. During that time there was a page from the cafe where you could order food from the cafe's grill but the cafe was not doing its part and eventually it got taken down.

At the time I did not have the necessary knowledge to take my hands on the project. I went to talk to senior developer David Mendoza about what I needed to know. He told me I needed to learn Flutter, golang, and msql. During my second semester, I tried to learn these languages but I was too busy to focus my attention on this project.

## 1.2 Project Overview

For the development of this project, I'll be using several different languages, libraries, and platforms that, when used together, can cover all the necessities for the development of a flutter mobile app.

Flutter (Dart), golang, msql, java, swift, and c++.

All of these will be used to cover all elements and requirements of this project, from management to design, implementation and functionality.

# 2. Requirements and terminology

## 2.1 Frameworks and libraries

**Flutter (Dart)**

Flutter is Google's free and open-source UI framework for creating native mobile applications. Released in 2017, Flutter allows developers to build mobile applications for both iOS and Android with a single codebase and programming language. This capability makes building iOS and Android apps simpler and faster.

**Golang (Go)**

Go is a statically typed, compiled programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. Go is syntactically similar to C, but with memory safety, garbage collection, structural typing, and CSP-style.

**MsQL**

Mini SQL (mSQL) is a lightweight database management system (DBMS) created by Huges Technologies as a memory-efficient, portable and performance-efficient option for enterprises and data houses. Initially developed and introduced in 1994, three versions of mSQL have been released. Its improved engine is suitable for both large- and small-scale databases and applications that require a small-level storage solution.

**Java**

JAVA is a programming language that is used in Android App Development. It is class-based and object-oriented programming whose syntax is influenced by C++. The primary goals of JAVA is to be simple, object-oriented, robust, secure, and high level.

**Swift**

Apple unveiled Swift in 2014 as the modern replacement for Objective-C and it quickly became a staple programming language for building apps for iOS and macOS computers.

**C++**

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes".

**Flutter library - HTTP**

This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform and supports mobile, desktop, and browser.

**Flutter library - INTL**

Provides internationalization and localization facilities, including message translation, plurals and genders, date/number formatting and parsing, and bidirectional text.

**Flutter library - URL_LAUNCHER**

A Flutter plugin for launching a URL. Supports iOS, Android, web, Windows, macOS, and Linux.

**Flutter library - PROVIDER**

A wrapper around InheritedWidget to make them easier to use and more reusable.

**Flutter library - TRANSPARENT_IMAGE**

A simple transparent image. Represented as a Uint8List, which was originally extracted from the Flutter codebase (was private in the test package).

It's a silly, simple library, but I found I needed transparent images in a few projects and found this is the simplest way to represent it :)

**Flutter library - IMAGE**

A Dart library provides the ability to load, save and manipulate images in a variety of different file formats.

The library is written entirely in Dart and has no reliance on dart:io, so it can be used for both server and web applications.

**Flutter library - PATH_PROVIDER**

A Flutter plugin for finding commonly used locations on the filesystem. Supports Android, iOS, Linux, macOS, and Windows. Not all methods are supported on all platforms.

**Flutter library - PATH**

A comprehensive, cross-platform path manipulation library for Dart. The path package provides common operations for manipulating paths: joining, splitting, normalizing, etc. We've tried very hard to make this library do the "right" thing on whatever platform you run it on, including in the browser. When you use the top-level functions, it will assume the current platform's path style and work with that. If you want to explicitly work with paths of a specific style, you can construct a `p.Context` for that style.

**Flutter library - SHARE**

A Flutter plugin to share content from your Flutter app via the platform's share dialog.

Wraps the ACTION_SEND Intent on Android and UIActivityViewController on iOS.

**Flutter library - LOCAL_AUTH**

This Flutter plugin provides means to perform local, on-device authentication of the user. This means referring to biometric authentication on iOS (Touch ID or lock code) and the fingerprint APIs on Android (introduced in Android 6.0).

**Flutter library - DEVICE_INFO**

Flutter plugin provides detailed information about the device (make, model, etc.), and the Android or iOS version the app is running on.

**Flutter library - FLUTTER_SECURE_STORAGE**

Flutter Secure Storage provides API to store data in secure storage. Keychain is used in iOS, and the KeyStore-based solution is used in Android.

**Flutter library - RXDART**

RxDart is an implementation of the popular reactiveX API for asynchronous programming, leveraging the native Dart Streams API.

**Flutter library - FLUTTER_HTML**

A Flutter widget rendering static HTML and CSS as Flutter widgets.

**Flutter library - HTML**

APIs for parsing and manipulating HTML content outside the browser.

**Flutter library - CAMERA**

A Flutter plugin for controlling the camera. Supports previewing the camera feed, capturing images and video, and streaming image buffers to Dart.

**Flutter library - FLUSHBAR**

A flexible widget for user notification. Customize your text, button, duration, animations, and much more. For Android devs, it is made to replace Snack bars and Toasts.

**Flutter library - FLAPPY_SEARCH_BAR_FORK**

A SearchBar widget automatizes most of your asynchronous search cases.

**Flutter library - DATETIME_PICKER_FORMFIELD**

A TextFormField that emits DateTimes and helps show Material, Cupertino, and other style picker dialogs.

**Flutter library - ANIMATED_BACKGROUND**

Animated Backgrounds for Flutter. Easily extended to paint whatever you want on the canvas.

**Flutter library - CAROUSEL_SLIDER**

A carousel slider widget.

**Flutter library - FLUTTER_BARCODE_SCANNER**

A plugin for barcode scanning support on Android and iOS. Supports barcodes, QR codes, etc.

**Flutter library - IMAGE_PICKER**

A Flutter plugin for iOS and Android for picking images from the image library, and taking new pictures with the camera.

**Flutter library - FONT_AWESOME_FLUTTER**

The Font Awesome Icon pack is available as Flutter Icons. Provides 1600 additional icons to use in your apps.

**Flutter library - CACHED_NETWORK_IMAGE**

A flutter library to show images from the internet and keep them in the cache directory.

**Flutter library - PERCENT_INDICATOR**

A library that allows you to display progress widgets based on percentage, can be Circular or Linear, you can also customize it to your needs.

**Flutter library for testing - TEST**

A full-featured library for writing and running Dart tests across platforms.

**Flutter library for testing - MOCKITO**

A mock framework inspired by Mockito with APIs for Fakes, Mocks, behavior verification, and stubbing.

**Flutter library for testing - BUILD_RUNNER**

A build system for Dart code generation and modular compilation.

**Flutter library for testing - FLUTTER_LAUNCHER_ICONS**

A package that simplifies the task of updating your Flutter app's launcher icon.

**Flutter library for testing - NETWORK_IMAGE_MOCK**

Utility for providing mocked Image.network response in Flutter widget tests.

**Golang package- CONTEXT**

Package context defines the Context type, which carries deadlines, cancelation signals, and other request-scoped values across API boundaries and between processes.

**Golang package- DATABASE/SQL**

Package SQL provides a generic interface around SQL (or SQL-like) databases.

**Golang package- ENCODING/JSON**

Package JSON implements encoding and decoding of JSON as defined in RFC 7159.

**Golang package - IO**

Package io provides basic interfaces to I/O primitives.

**Golang package - IOUTIL**

Package ioutil implements some I/O utility functions.

**Golang package - NET**

Package net provides a portable interface for network I/O, including TCP/IP, UDP, domain name resolution, and Unix domain sockets.

**Golang package - NET/HTTP**

Package HTTP provides HTTP client and server implementations.

**Golang package - OS**

Package os provides a platform-independent interface to operating system functionality.

**Golang package - OS/SIGNAL**

Package signal implements access to incoming signals.

**Golang package - RUNTIME**

Package runtime contains operations that interact with Go's runtime system, such as functions to control goroutines.

**Golang package - STRINGS**

Package strings implement simple functions to manipulate UTF-8 encoded strings.

**Golang package - SYSCALL**

Package syscall contains an interface to the low-level operating system primitives.

**Golang package - TIME**

Package time provides functionality for measuring and displaying time.

**Golang package - BUFIO**

Package bufio implements buffered I/O. It wraps an io.Reader or io.Writer object, creating another object (Reader or Writer) that also implements the interface but provides buffering and some help for textual I/O.

**Golang package - BYTES**

Package bytes implement functions for the manipulation of byte slices.

**Golang package - FMT**

Package fmt implements formatted I/O with functions analogous to C's printf and scanf.

**Golang package - STRCONV**

Package strconv implements conversions to and from string representations of basic data types.

**Golang package - UUID**

Package UUID generates and inspects UUIDs.

**Golang package - ICS**

An ICS / ICal parser and serializer for Golang.

**Golang package - LOG**

Package logrus is a structured logger for Go, completely API compatible with the standard library logger.

## 2.2 Software Requirements

**Integrated Development Environment**

It is a computer application that provides comprehensive services to make software development easier for the developer or programmer. Typically, an IDE consists of a source code editor, automatic construction tools, and a debugger. Most IDEs have smart code auto-completion (IntelliSense). Some IDEs contain a compiler, an interpreter, or both, such as NetBeans and Eclipse; others do not, such as SharpDevelop and Lazarus. Recommended IDE: Visual Studio Code.

**MySQL**

MySQL is an open-source relational database management system (RDBMS) with a client-server model. RDBMS is a software or service used to create and manage databases based on a relational model.

**Android Studio**

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.

**Xcode**

Xcode is Apple's integrated development environment for macOS, used to develop software for macOS, iOS, iPadOS, watchOS, and tvOS.

**Local Environment**

A local environment is a place within the local computer where the changes to the project could be done without having to affect the “live” version of the project Recommended Local Environments: iOS simulator or android studio.

**Hardware**

I recommend the following hardware and requirements:

An apple computer capable to run Big Sur (11.6). 3.2GHz 6-core Intel Core i7 processor. 16GB, 32GB, or 64GB 2666MHz DDR4 memory. 1TB or 2TB SSD. 10Gb Ethernet.

An iPhone running the latest iOS version. At this time is 15.4.1.

### 2.3 Installation

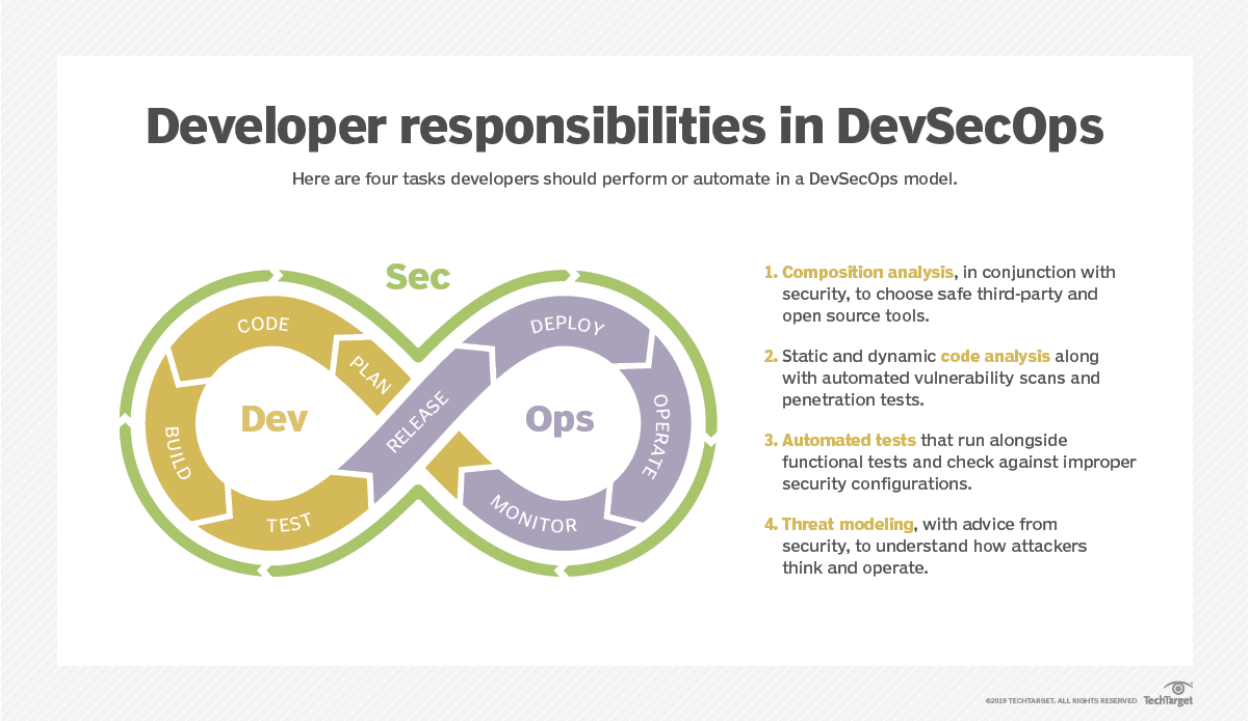
1. Set up a new environment and drag the folders to the location.
2. Download and install Xcode and Android Studio.
3. Download and install Visual Studio Code.
4. Download and install Flutter (see instructions for installation at https://docs.flutter.dev/get-started/install).
5. Download and install Microsoft Azure.
6. See Sr. Developer for database authentication and access.
7. Connect to https://git.swau.edu/
8. Clone the "iamswau" project.
9. You should be ready to modify the project!

# 3. Project Organization

### 3.1 Software Process Model

For this project, I would be using DevOps. DevOps is a methodology meant to improve work throughout the software development lifecycle. You can visualize a DevOps process as an infinite loop, comprising these steps: plan, code, build, test, release, deploy, operate, monitor, and -- through feedback -- plan, which resets the loop.

In its broadest meaning, DevOps is a philosophy that promotes better communication and collaboration between these teams -- and others -- in an organization. In its most narrow interpretation, DevOps describes the adoption of iterative software development, automation, and programmable infrastructure deployment and maintenance. The term also covers culture changes, such as building trust and cohesion between developers and systems administrators and aligning technological projects to business requirements. DevOps can change the software delivery chain, services, job roles, IT tools, and best practices.




The way we use DevOps was Mr. Mendoza would talk to me about a new feature he wanted to see in the app. I would plan how the new feature would work and did drawings, and create a mock-up version of that new feature. Then I would code the backend, create the database, and at the end code the front end of the mobile app. Once that was done, Mr. Mendoza and I would build the new version and put it through the different tests. Once the new version was working I would give green light to Mr. Mendoza to release the new version in the mobile stores. We would wait for a couple of days to see that everything works. In the end, after monitoring the new version we would move on to the test project.

### 3.2 Roles and Responsibilities

Senior developer - David Mendoza Junior developer - Hiram Hernandez

Some of the back-ends were already developed by other developers including Mr. Mendoza. Some databases were designed in the past, and information from the users is coming from OpenCUAS. Projects like transportation, mugshots, staff directory, and visit's back end were

done by Hiram Hernandez. The project for SLAD was done by Hiram Hernandez, from designing and creating a database to the front end.

## User roles:

Each role is declared in the user's token.

Admin: The user with the scope: ITS - Student Programmers. They have access to every option in the app, from administrating the transportation page to SLAD.

SLAD admin: The users with the scopes:  Chaplain - Staff, ITS Staff, ITS - Student Programmers, and Campus Ministries. They have access to administrate the SLAD page.

Staff: The users with the scopes: SWAU Staff and SWAU Faculty. They have access to request a vehicle in the app but they do not have access to the schedule page for students.

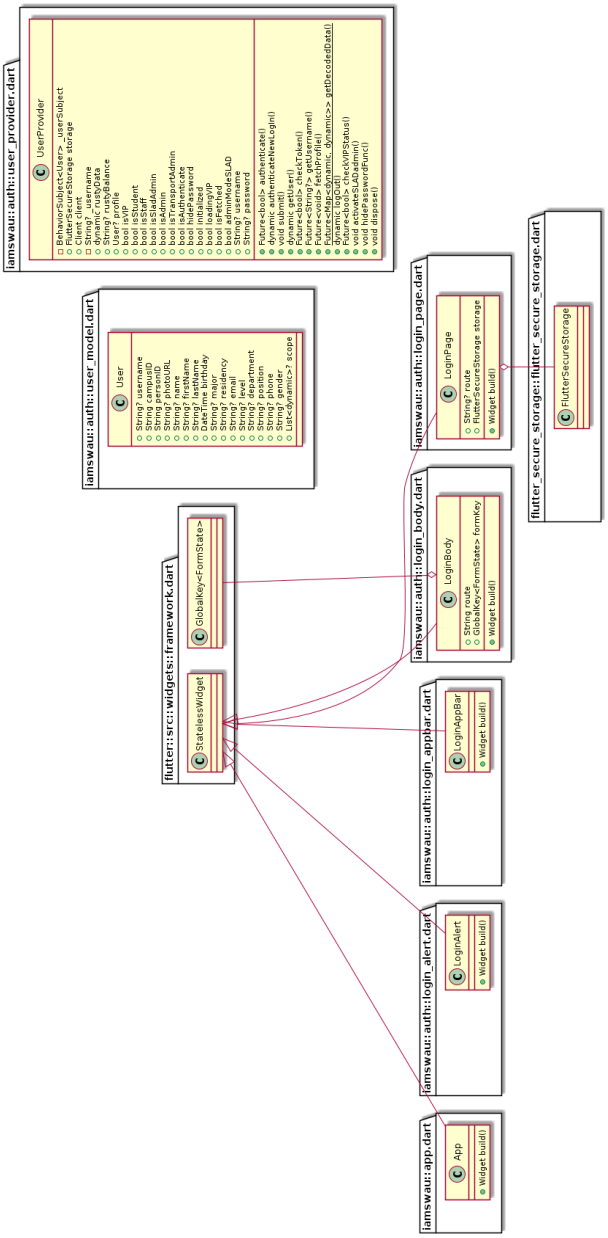
Transportation admin: The user with the scopes:  Campus Services - Staff, ITS Staff, and ITS - Student Programmers. They have access to administrate the transportation page.

Student: The user with the scope: SWAU Students. They have access to all the apps except the admin pages and a transportation page.

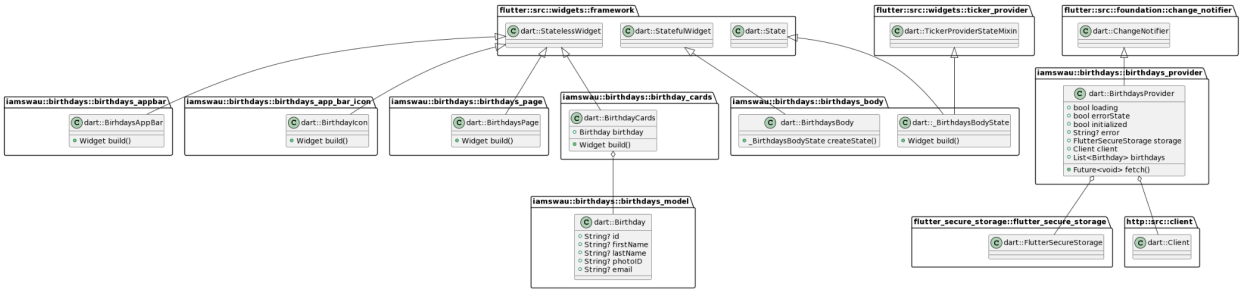
VIPs: The users with special access to use the timeclock page out of the school's network.

See code reference 1

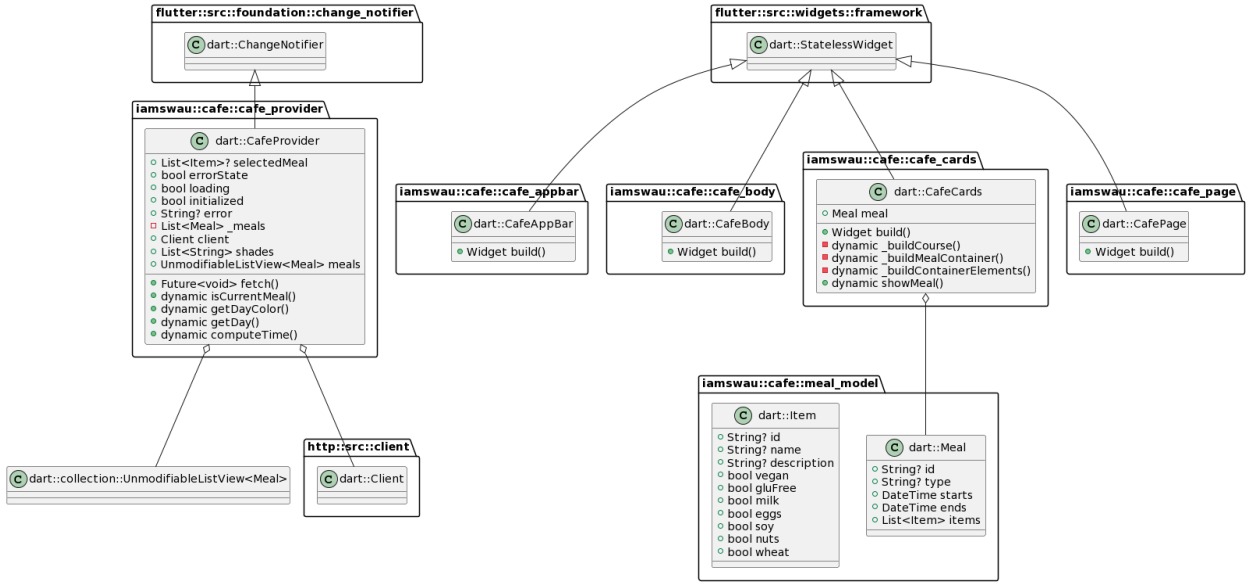
# Graphs


iamswau::auth::user_provider.dart
UserProvider
iamswau::auth::user_model.dart
User
flutter::src::widgets::framework.dart
GlobalKey<FormState>
StatelessWidget
iamswau::auth::login_page.dart
LoginPage
iamswau::auth::login_body.dart
LoginBody
iamswau::auth::login_appbar.dart
LoginAppBar
iamswau::auth::login_alert.dart
LoginAlert
iamswau::app.dart
App
flutter_secure_storage::flutter_secure_storage.dart
FlutterSecureStorage

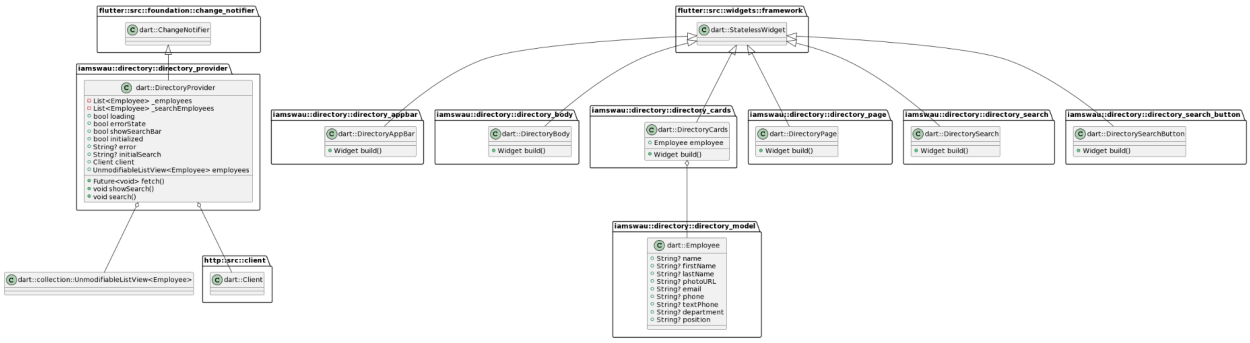
Birthdays:



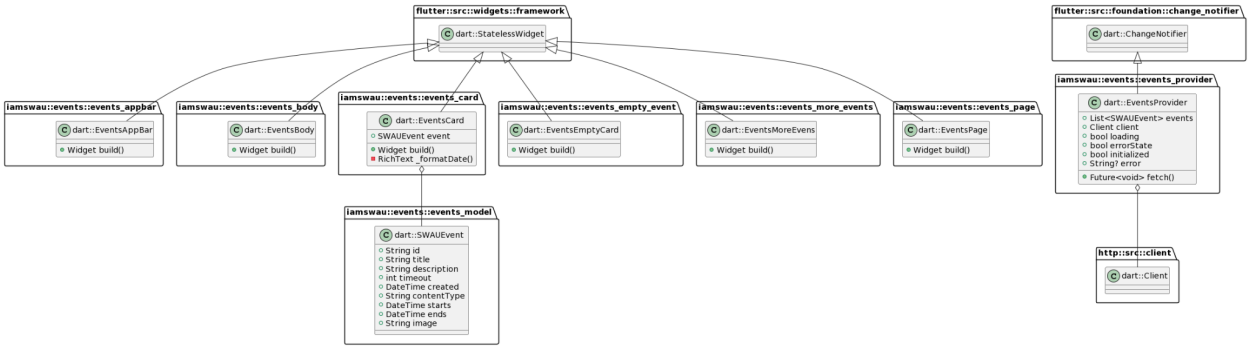

Cafe:



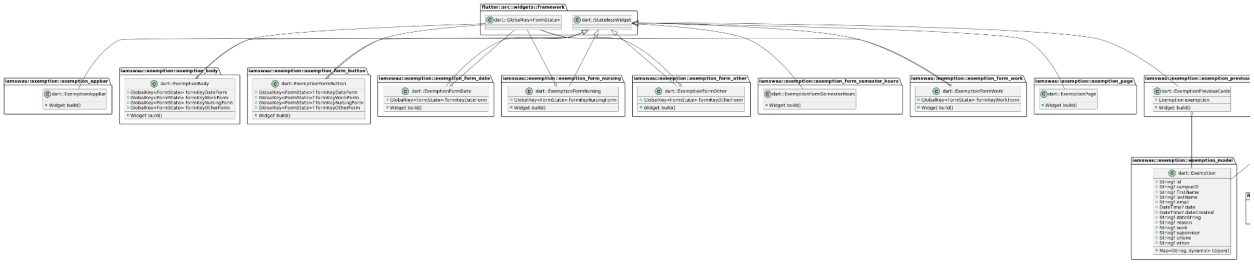

Directory:



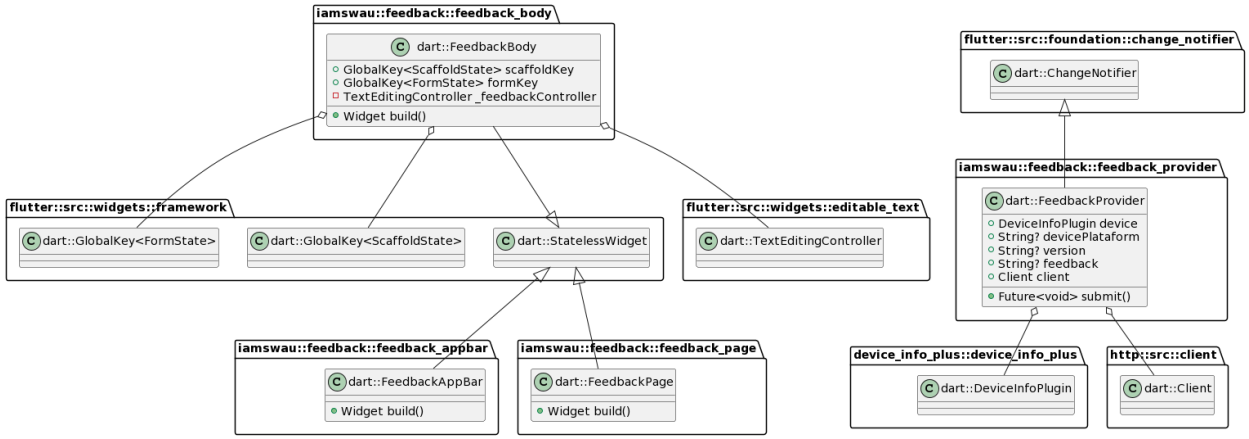
Events:




Assembly Exemption:

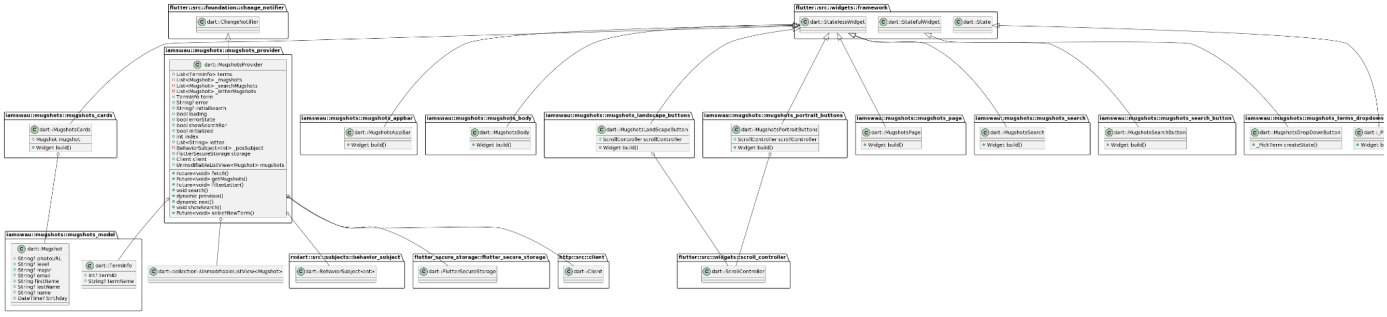


Feedback:

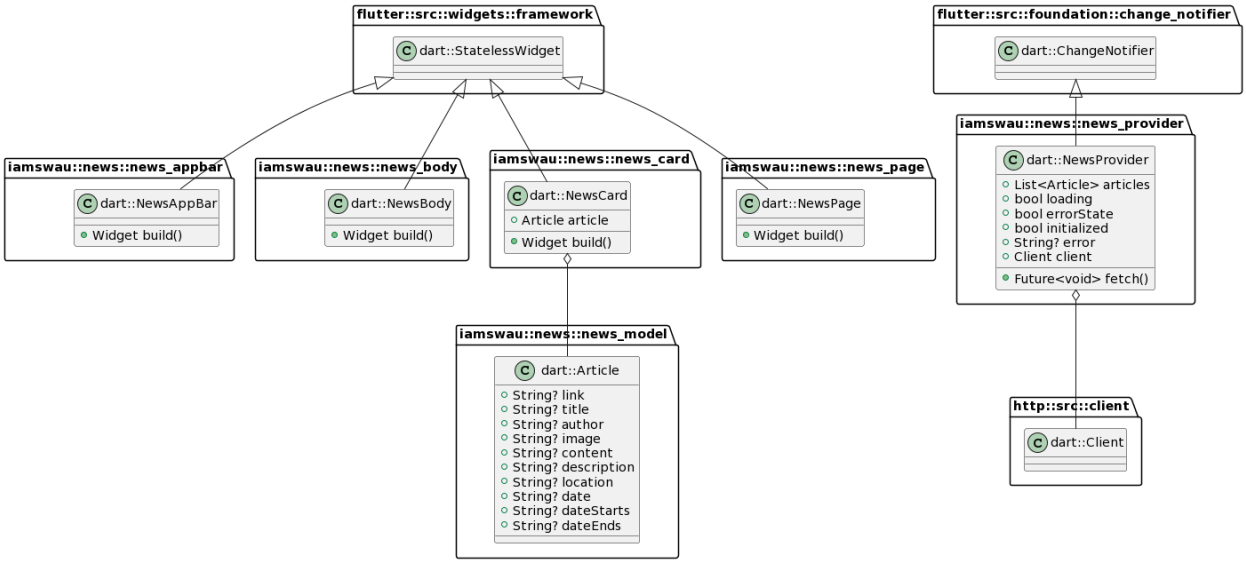


Links:

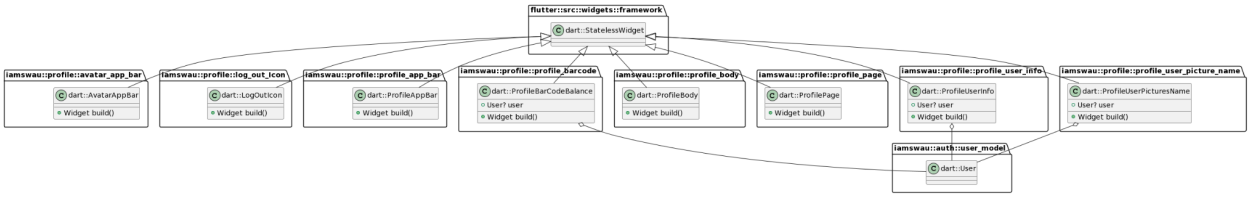

Mugshots:



News:

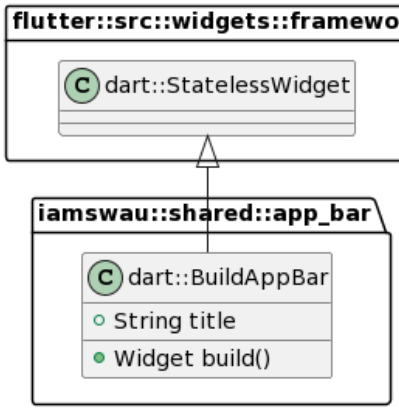



Profile:

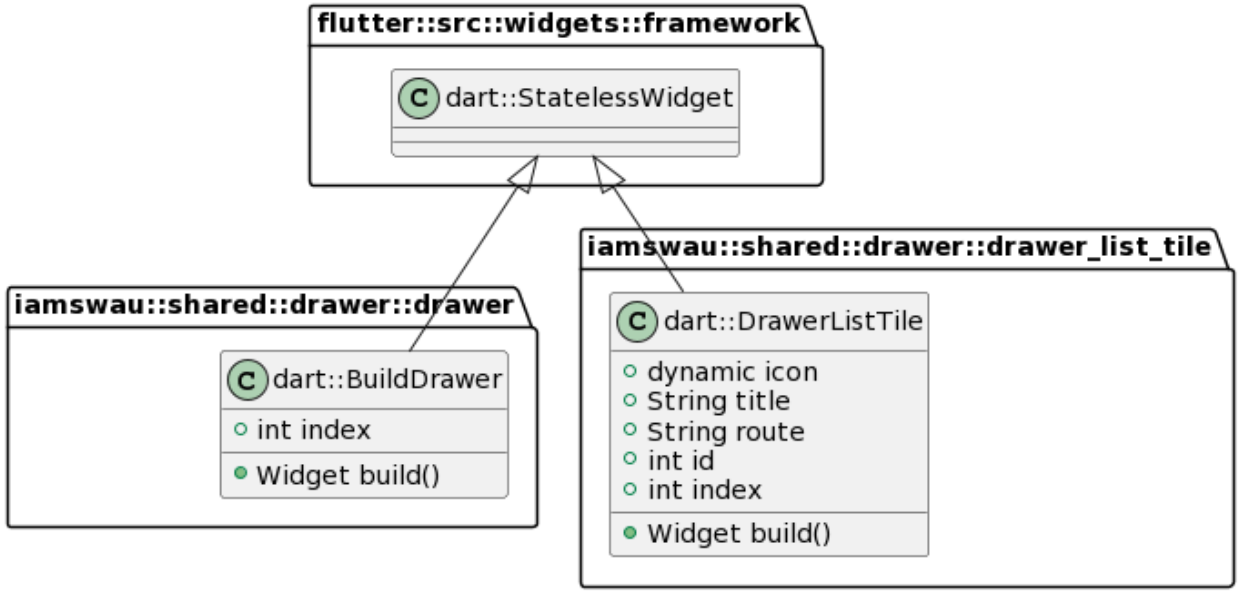


Schedule:

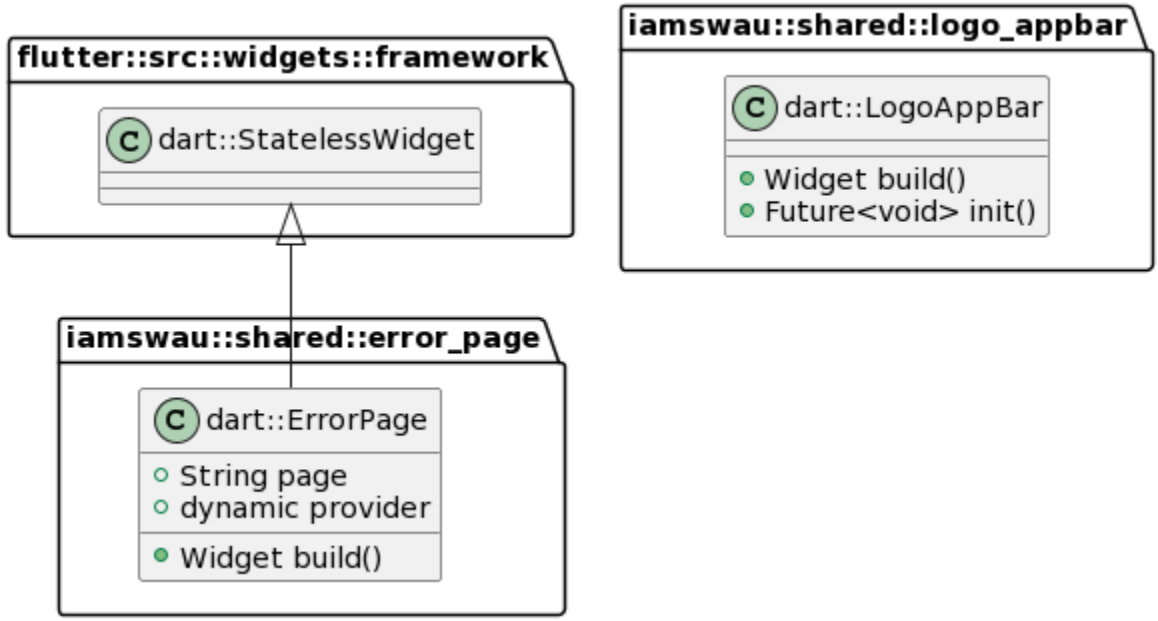
App bar:



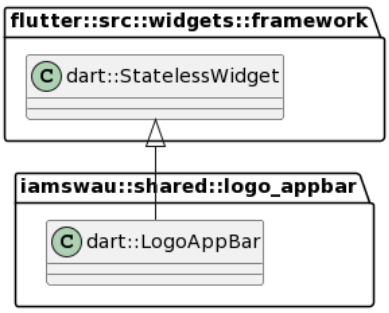
Drawer:




Error Page:

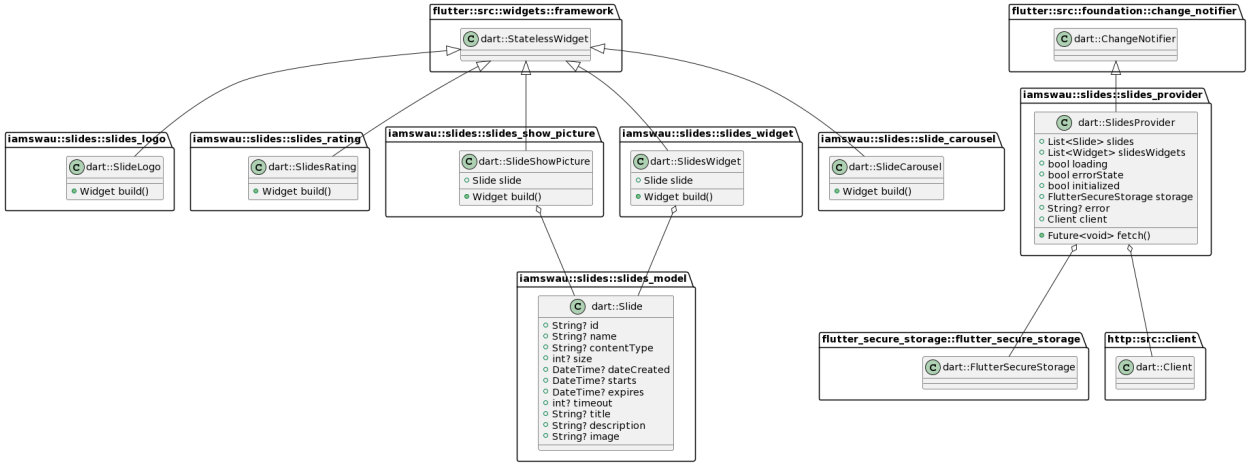



Logo app bar:

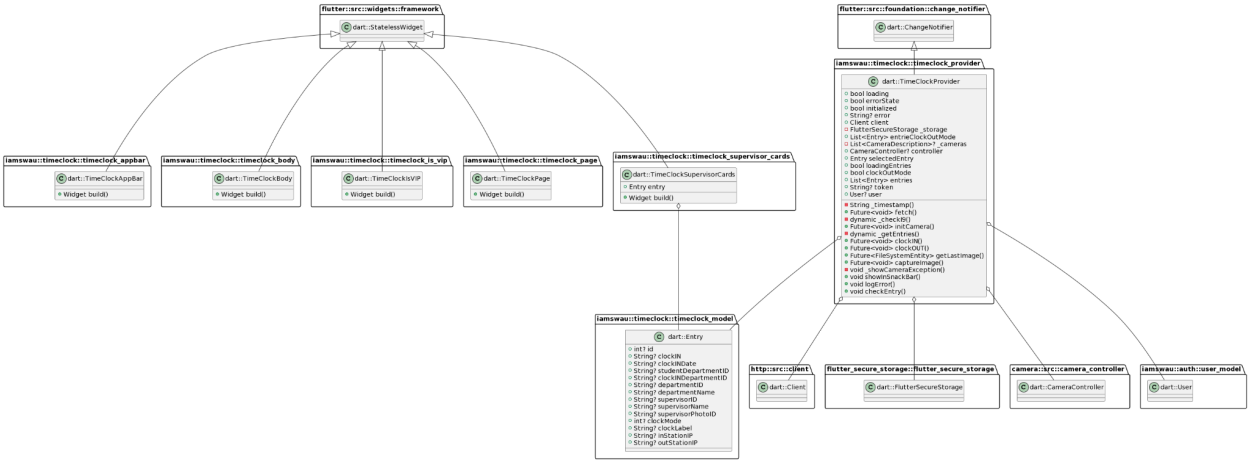


SLAD:
https://www.plantuml.com/plantuml/svg/pLfVS-Cc47_tfz3agPqs_G2QZ7jdtoMJzYwNgQ_jemSBRDEJG0CeRlhdknUnYf1iJ0IdQloIM0iilzykx2v2RwH2GbL5Fijo96LoHL4X3wXAKvaZ37_N21UKwRzuNP1YGuJy0Uj1QVf8io-h0kNvdU1LUPNyFKkIxrOAANBJRvedcI1QRDg--NRsxsnsjSqhfOZGKuWiJGyKxuYIQReLg207BhuqSvWXsoh_tNIuIcx-kRx-AfIp6ARhlX83gk4Bm99DHNCSGGqOZO1c0AyDnyylSslfsrguqB9ncEH4IeUNWf0COR6hESlZqNT6wcQY9VYPeO9cYJQcDTne9CE13YQwpf30RYNBkgLXR4abPfZV2WHJPKVqsx3cgCYnaRuU9Zzdh8BhAPkHtNqweb_20Gzo4CquEzPmhsMJxYS_aW4iShuxiSC7BPkOXGV9K1Phf64nZ5X6rgWiDqYujBrjExqjomKIq_ECnnfZ2OYrFjtJvu1ud34MWAtkLLtl-SO4LdZFjlpz4s4g8VLVVqoLUtwmGpzGgHw9CAF1ZkFZ7HXJ7Nn62AuZSubnQubsnA_vKVUOTc77unniWQOMz4IDZz1Z-l0MZpAKlOaM7gtJimv15yhMB26FlYdNUJp28Qnz1cnKsMpbh_EX-sS-IRqVW3t032Pwdb7LoLWWgQDx0WFzje4-dqf5EPEtN1Yj_Iu6nPApBHK5glkvpWWmRQyN67W4NeCoc-KzVa8BXHpYLQa1P0g4ZKgZ3bmoxKOEmELZw8ZpHZ096Jy4F-wZpUeDBhULgWIvQNR1FA6Cgd7P-86vcBXtXkEjdA1izsD2SVT9yo3Tvv9ccc5FX8dC1AtzNqifKtfKLJGBNrCVYiZKdTBirL4GbJr8jjggHHlESwrOBcj1tXqhMxlKqdT8aS-q85gd_hKWEyhagVmzmxATw4cprXC746zCvVH87zBzExVyuSWcKY5LoIxH3mHXYFeVKVcoFhzFseMg-4yIedCWxCPVNW_jkU9zw3l231mhoL1HLX9wzK6xYO44xFzfkw12xMjnVQgwEJ7el6iisVO37VDZIuOiMbt0E5UicvgLeGnMfykRGtoFAhNdudGfMkW1fxum1dvaS106aMS1C5J4OpMH0W27m_7i6-WCmcF3jekc3sNUZInrkoiq-NaUrSMJLUgUwVru1N-Yc8Zke4O4kCt-JNAEiBP2-qo4uCBWQaLrTgGeftyHizgMEILC9Pdvrx4Z33Nf5TYlIARDkb9SQB29XFzsLzjjxzprDcxDcqBNssRLjRriffh3ktGGd5iYZXNYw6Pq7FSrnLKxm0GR2qq_oaOXs5E-L9qk2_iRLdVE9Wj6p-KuIC3Fqu1A8I5QstLttSeKaGI_tcEbPxUucdnWCqHEa2UBt1TeH-fxC2AIqlnpx4tzsAdRdZZ5ysHBLBPtrtCxei3NqDmRXZ0sROucWgawrvQP4kHSQ-ysoj7kk8jnJUxfO4ZLxdMtz9U0kmyi2d-Z3u7jzgiamcq5Q7VPRu_OZa3pofG7PytqYWxO25Q5nnQBPvV3x4osfvkDjntp0wkZvtBrcwDLPeWjaCWu9WzSQAkwiJx2-huLl7ZSSqRwntCW35mDRKWlmV4UJk5xfSec8q7mRhf2GB-ALTlakhKquNxTnFaqTSiRFQuK4OiWe0J-twQm-oP-YZeGhY44fgbzQiTq8kN9Qv8jhnWshsxI55RDcd55jxGTlJJIXqPeIuqWgFwImsG6k9M5Tm-d9tbh9xEUUs-sRLidnxWBh96dd_vAUqJqWIwnbSlh4zXBIzEpLykVbkozudwomdXSXh76X5jF_qJsxZFGaDTtET-W_6VoV7FBHM6CCq_MM_rRonx8mMgFtoxnR63_37MjYyia6ogUN_oYiASs_pHmchpytHNFBjI8VWSPjqHyC6MhVp3OM5Wk9eM48m_pY_qCmAsozocRUztQO_c0hp5wPp8hFZcENVz2i_rdygU2i8ZKE_k1wfZ41hdPCXpVpV7O0nnjxv8yMkq5_SK4gN3mSKOn1dzLUMrsZtN2xtl7lyoDPtXXLxmZ40bT2_xi1nbzt7DDmDGNav32aGzrWoVgnNzG_GUdpF_9T3YGo_E57t10L2Gcq_cEMP0Bnp_dHm6uC4-4umzqZAd_F5gd_buu4c4WQnDaF3h73ys1ar_GCWp77kOCJwWnEcsa7dWm22pqmn5TMg4VZdqsUqCOhehyFm00

Slides:




Timeclock:




Transportation:

https://www.plantuml.com/plantuml/svg/nLfFS-Ew3R_dKsXoQgTD3-tDaybk_ZGxswOxcTZTdPuyj0ZB75EaIb9ntVPzzqTAY4pPZE9LZBnB7ECF2Fm8WWJfpzOnu-fIdkMIMPkT2rRQ3Qid4sUOifKsZZcXrVxtEUEby7-hYfa_SMRSP3Bh2Lm7_dLLtJ1pdltlBCl-y4Fm0bosg8Nal_lzsIzdP-TBMJi7PZAn9fzCDes0dKoMXfMmqMQDbgUEEP1WRMlYFBly_yK5ZQjdut1OQBvztVQDvzA0GE3dI0GuSs3tJEyitmKk4GG4Zhu3WtdE3BT9ZBi1ReC8eZ5rHgWYSs3A13YUyY2ikvokQkSa1EMhpC2_Qx0kV0dCqs984EXxq3HwyyOOJtg3M2QWafv2cNC4eRmNhSIy90UAap32B0k60ksBr0ic_mxRotjjogbhy5pwVptfhrou89pEMBvLh1Hv1YY5W7zCFFfdPgVK72IQZZ0u_qXdZqt4r7oSdo2DgSCSdz56Lrnlr3pKnjSBjfUwyr8pBuJPF1-IGkYot83_QFuVKH2MjOo1f0_jb20UWyzfzkeO7NgVns8KYjgyCYA7rpCwhBd7849yY25m-NsusBOi3W-3jHEn6TevhZb_U-DWhH3YIkpqRr-5oICSEVyLAmO6U6G5qM8Wy1CH67Sok3FY6KpllFuf4pp-fbW9yNSecP0neLff1G6PyNx49vIaCu4n4oLymZFSZjyq-RRdeEBnzwx_-0UhBd4P_Z73Vwuof_zcjGfJUue01f7CMLdLzklTOK0Fm9wzo27ZjjdAUrQMmbZtRMzI93kavLefo8EdCPKBSqZS1UOP3q91cXdKljNb0ipX_4S4nha8eam6AjVrGe9tPyLC0JPfEH0NMaiVI5qKh82N-MebJpzTts4bSWbFRRkPcZNqnq2fds4ApDaTaULhfJSIU07yYJdepribydLTzMbXtzVBPP-uXgqzQifviqeJy-jOZjEUdexqWkLGCUDAKAwdLvjAM-WBMYTKWrvQ_xQPKzupmwm5LO2nr7FO701iQkwwdAk228KVbT

7FWaFwqkqHcUYQKCwZOYpufHCV1HHicX1IjAbZFbimuU8rYvHGq-pksugfzLRNM8JwKf
Zg1yBxTA644qoA_mBlQ62CDhWVzudFE-txTZ-SGbuRxxWsh83Cjf-TajICU_JxHfejiYFbth42
C0P41Yy-0j_FrxQ23kaz7jQDgNS_eNKN0NZHyDgDxogRjslBTagOI5VHeaimBKXl4FXTh8r3
UdPKDGBvK5yrLVInMoi9AN-kDF02nmpdEt4F56h64Ei92izecujx9kM0w2oaXYr4s8i6n7u8v
S3S663h0Qbf5Q98IwMzQpM6JDPb-VR0NmpZTLUq8dvONU5pdmnSEE1xd0pJ_gjQwa3kxn
WPvZzoTxKsf7LCm5KLzjG4kSlcmrtrfLCC6PRcpF1SSoi5A9VbpKSW_5ELcekbO0i90QFl0Z
PTmaVvVb-xsi3binRS9nkuVELhOPAEQTvhPN0DT26aT8LqO0AmITLkLVQPBC-XSat6lyh0
sUZperNNP-GhoDSFROKuQCSyfcruCnt0Ifbi-VT6bmSIYQfrNokCwW0GxFuI71mepQmhpb
myHae8hoHJB5ilIk5G_rL-FyEnsJdnXXZtetFRx1Xpt2ecapGTBQPscNC2_yYlM3_UAsgFnX-y
YHnRELTXUzvMHLHgAoNHeAr8hgM4l3KsK8SFBrQMkbOSlMXhobnfvxMxqMyRwZSaud
t2IKCPyKQCksHuWbwmj75CGie7Spf_noCItgE7xDEzdDFuEXw9SAitPFtUyu-uNusaW-WC_
kF2OvrGnQrMpeI5PzhkT_VzTS7ErmTnl49qLZj6HJBhRhjRc2CqegkJSBBSCT9PGZ8J9xaq_d
Z7W0iN2ktSUM3HL19Z-Ylj4R6DNrnsfJTpcw-0rxqxy_rHfbvkYcB9iXET2ky4axffSoEbw2SR
hMBRRZNleA_ydEDJulSSAHjFUaDKzWYm6prJpMrCEX6Y9fzcIw3oTXmQGJrwsaZRftk4e
N5sH6RW9SASSOxtKCanyCnypNdyN9FUGGvkbdQij_RNFmU1XdGvFFvLuXS6v26VflRJE
qh_v3hySmtg04-OvVFXaPhmA7CzVQ2YZuaocUdTfK_fE9FfenkNpyCj98v4sVBI-JiIaQ4s4ktJ
jRmqlev78fIP8Uj7jhn7bR4XWR-CBPvfdUZ9EIsUQCQFRfVRX-fZ1AC7wAEakrViOyJn7Vpd
_9ZfzYbye6C_R7tmAMRNGLsWmhpGrnkslQcTUtcV9if0WbQVth4FQ_L9OXjPL8QQ4BHEr
p7H-3eIYS7cT1OuT43Gk3eUXnLJMrsdpNlU-b-wfWE3qD-ppw1uNSfV0G00

Visit:
https://www.plantuml.com/plantuml/svg/pLdTS-Cc47_tNw79KpljTQRjcoUJ-yXTSctlBfdcezCd
3nROfa6W0UHKr_P_Bo2CjH4oadFnvIdcjxlinoxBehnI6abT5MoICwHKTaHHeUvHDPskgQB
Q_vaXZ2LHwWUCfC5kBVQwWOwoVoPPzjqv4tF4VYFryPcGnPL6cfna2_FJG9xJ_BASVr2y
93gRLvJXRxwT_3UPxA5Jl7YH7IrOfJMHqwcI-NHwx_PLq-b2eeBS2tdd9MEc7Ks6IZj-PfGsT
Xzbn_yQtUDDdYG2N9Pp985WMRv1qeGt7hIdiR_FYeJrSu7hjjGRipwevN4B1bWzmwYUCQ
hqG-4RJjT4caNz5jNg54di7OA4h0BBkCTVIh_ezxyGc32_1nHzeb8VQtmY5xaeIiIfuE3md0Pq
G9i9p8aiZDFyV9jDol3X8P_bIEBuBYvvc-W7D0kogL3lht5WiGUv9uzoskPamFc2oW9f8m6sQ
U7nhccG3rJfumO-oIZNH1Ab8wGSscHC37i_GViTvLewF1hbUaRvGi0z7FwBWGTrWS1jzOul
qRGHpxSX96rFvO3WAdT1Gsos58YrjtctfPqRqe1SF9189s9VIvvRGiRxC3OnCqRMeGTUqlm
kxFV14dnoPbtADP55bak2D76-xD4K5XKRxttAvCT5eCVHBpE3ZdSpPTTeNvMkSDqAtHab3
5yihYpSoZN0lrgc4-O-nlUEaryjoqchadvAAUIiH4i2se95BmtedRhIal9bvfW_4gKCvO2lgeIDAH
TNebB6-hRi-mOQS7bupejIRqRwjuPi-P1sF4Z78p7UcCVV7tihfLmphx7vYSr-iyOzBxLr-mckg
OIHmn8y4qsIMYSUPg_TeY356NgOBZNKsFsSxlFj4Crff4N1l_JhwdLPCfgtNoQKwvSPnTlUz
393PIaCN12kRQtsaAvfGJQdsFusH2dfcgWsVqvr3TQYiB8gF7F2a-laPUkf0qKghcKDLTbJTUg
A35YEoL8Iye2LbCPri2Kf46MGYt0CM1PKAlq95GI0aZxaMng0o3RYanSyR73ZtkQH1-GPyc
g0DHQzAV53zzmaqWQ4NX5vkH8Sc5bkWBaGp2nDsToKK87IC2OEqGG2aX3THhH4N9L2
QbS_RSfdwlR_Y4hlxlSPhZagQ7wIQV6hyc_LuNKQoW9ScxPG4qc24UtF4be21aaOT0ec4XP
EDzJxkvxysjVrFvv7MCeZDQAryGlD6TaPRbkFepP-QuyZs1ESqTwohWa2oU8FqfaB3DO8Lq
o3mgWOCpMGjjRi34OXApHWbh5ibxsZsoXTp_If6pKs4KpNZJfVAouiEl1ZJ-1D17QKtcSRu

uJTYN1tndZ_oZZWSo9gGSfgATOK4mdaFEQjjzTMnWJ2b2_3sZrTd7K1efngYXZzJ70CQxg
FYf5kobmKPdVJOLjqTIOfHlLM9QQwRO9U8NxtfwYWJRPCsht7WHOuPPHmdUNkJoihAD
TqJTp2mY_Sb02x_aasgm3GuWkzsZ7v5M6U2ymSsL-LSYFSLju5nOesRRX7f8Uc22Ev6ppzy
DT7FlVtRI_3w-KkyT_zRQnMjDp1vcVX7HpmIurAx-NASPVVcn7ekYvtUM17zQhqKtgxEjQ
8LUJ73l9J1_dP3-VPKegglBLG5_5XX608VXGFarMN0OIzGsr56n8wCO5a48fmTFALe3AiXC
nNH6c90ZANIAbtNA72L7f05VfRWEasCmiXjzikwu8ZfMc-XPJ8Sx9LjHQCaNgE6DlgGkgE
w0tKUY9bQFixycA13vQ-_wRTy49Wkg3soDeDRYcvxnntUDfxnS3fhxQBNKBXoBTE_5cbA
qcEru9YS_SIdQ_CBV4GNnBTxfDnZcRp6B4KIfjODXG_4liJOhi7m4FlFQju7fDeQXTIJCdaT
wUcZcBunQQY8hGtlhQ8hQq8_SfNMOpstjLRX72-gRpOXgu48uJRK8mnCqDPnY9YZduek7
Z4C94xETh7VIUZF_azF98KOjrd2y9uvzRNu4bf6qYCgzr28DrVqMukW1IErdHfq6lX1w3nuu
WV75TQbtviQkxopHlLBOxssNdSIEf7iVqLZVmmzAIAV3ayJar3oGNpz-rqEgJB7aHZwCc7K
ETRU-TziX0LnwWHp5U8BycC2ssaWz9JXtxoO1ZpHzkR6lVTp3ohHTLvB5pxQVhmpvQ1D
gKy4vLKGDW2X_KcOSVa5U6uAjZ_

# 4. Project Management Plan

## 4.1 Tasks

1. Migrate the app to the newer version of flutter.
2. Clean old code.
3. Resign the front end of the app.
4. Make a better UX (User experience).
5. Add new pages to the mobile app.
6. Create unit test.
7. Document the new version.

**Migrate the app from flutter 1.0 to the newer version of flutter 2.0**
**Description:**
Cleaning code from the old version to make it work with the new flutter version.

**Deliverables and Milestones:**
Being able to learn and use null-safe the new flutter version. Make the app work again.

**Resources Needed:**
Flutter 2.0, documentation, and update the libraries.

**Risks and Contingencies:**
Break the app and not be able to make work again.

**Clean old code**
**Description:**
There is a lot of code that will be unnecessary with the new version and some others can be improved. Code References 2

**Deliverables and Milestones:**
Migrate the "_bloc" to providers.

**Resources Needed:**
Flutter providers newer version and understanding how they work.

**Risks and Contingencies:**
Brake the code bad, things might stop working.

**Resign the front end of the app**

**Description:**
The front end can be improved and optimized. There is a lot of duplication in the code. See code reference 3.
**Deliverables and Milestones:**
Separate the widgets and make them small as possible. Eliminate unnecessary code. Improve the connection page - provider.

**Resources Needed:**
Understating classes, stateless widgets, stateful widgets, and material apps.

**Risks and Contingencies:**
Break the code, have a hard time understating how classes and widgets work.

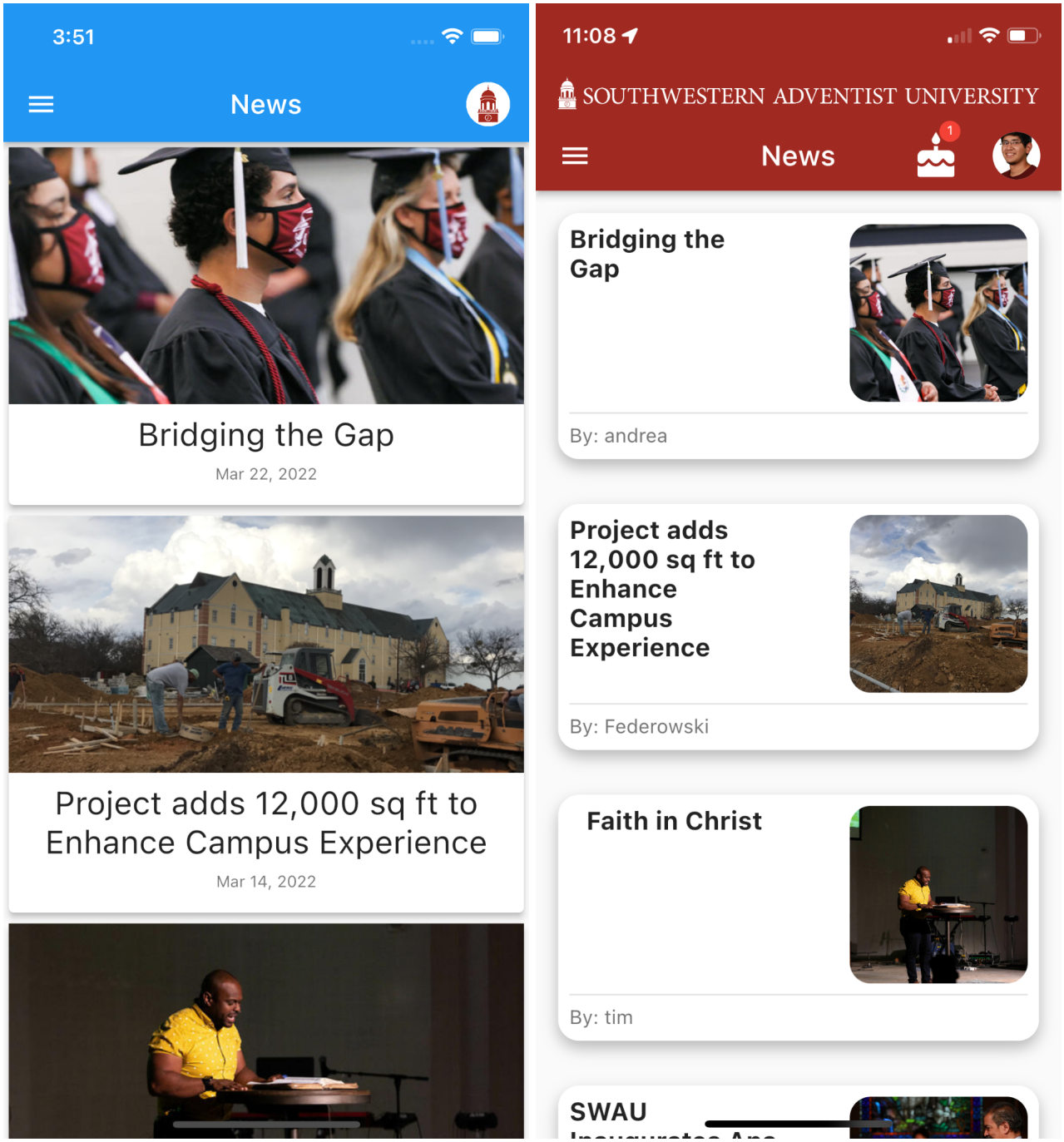
**Make a better UX**
**Description:**
Improve user experience, the UX that was in the app was really simple and easily breakable. Make the user experience more pleasant.

**Deliverables and Milestones:**
We could look for a UX designer but there are free designs online. Based on the new design we could create one that fits our needs. Make the app look as professional as possible.

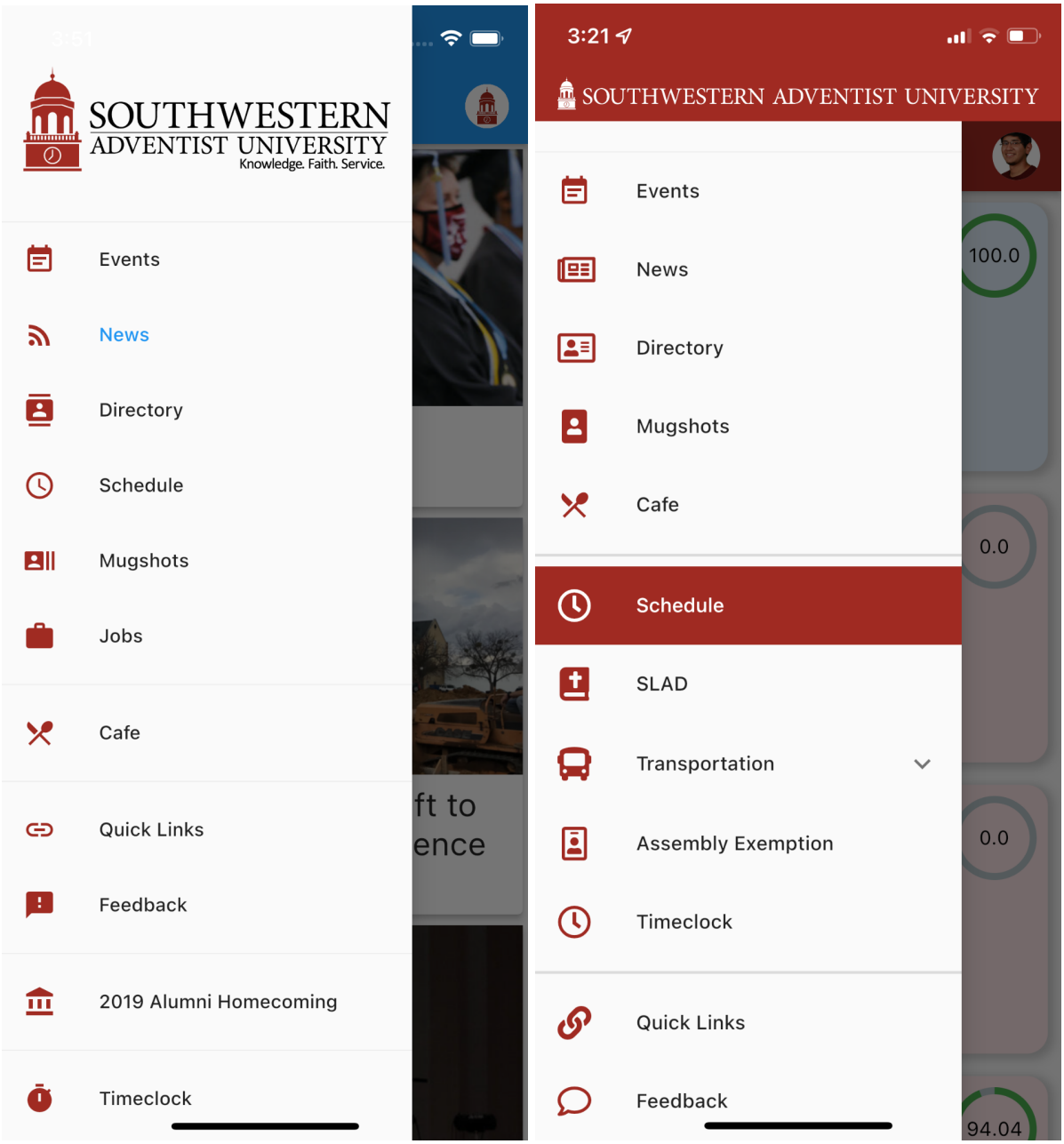
**Resources Needed:**
Designs by a good UX designer.

**Risks and Contingencies:**
Create a worse experience or be hard to use.

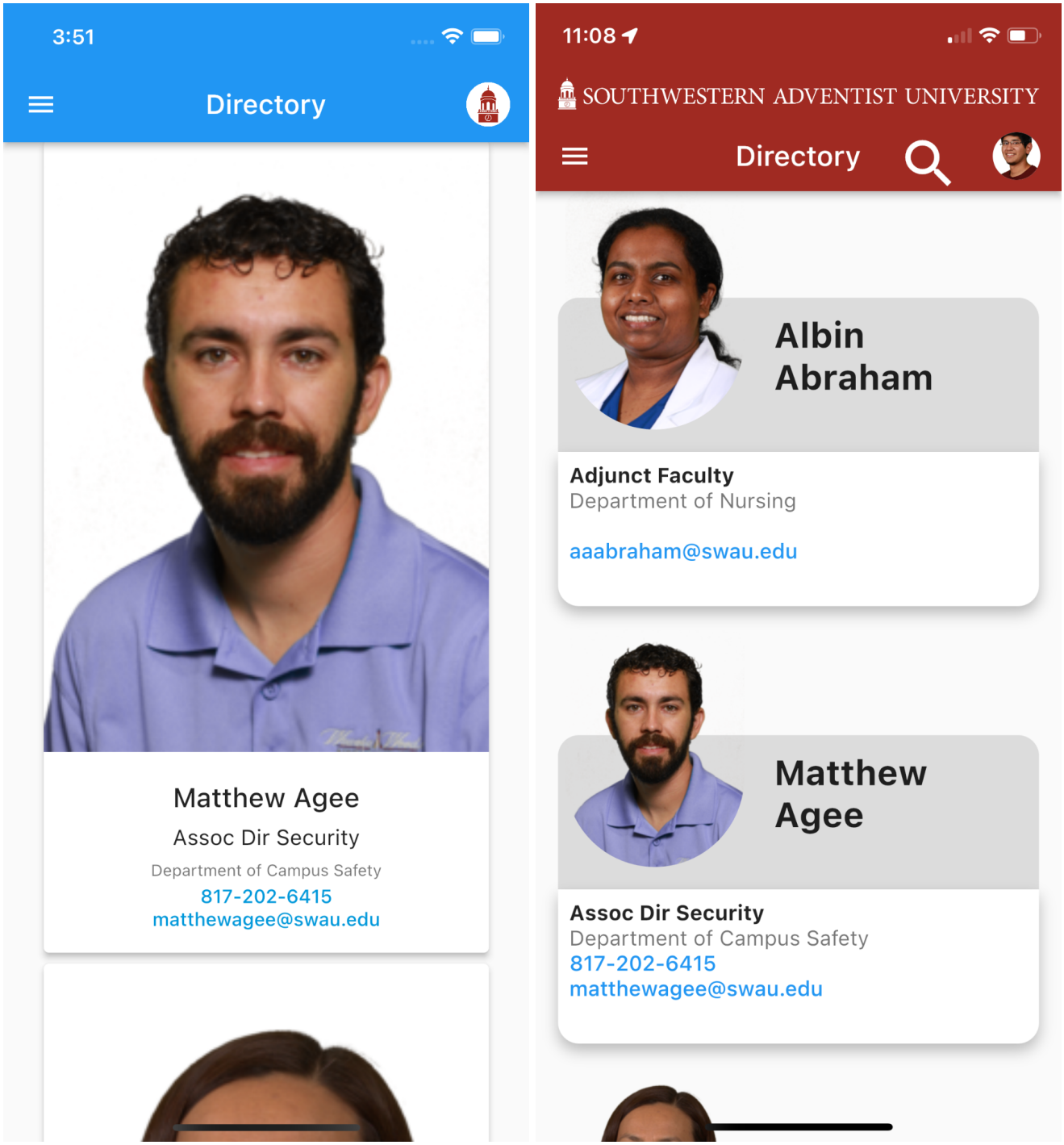
News Page (Old UX left, new to the right): Here I change the design of the cards to look more modern.



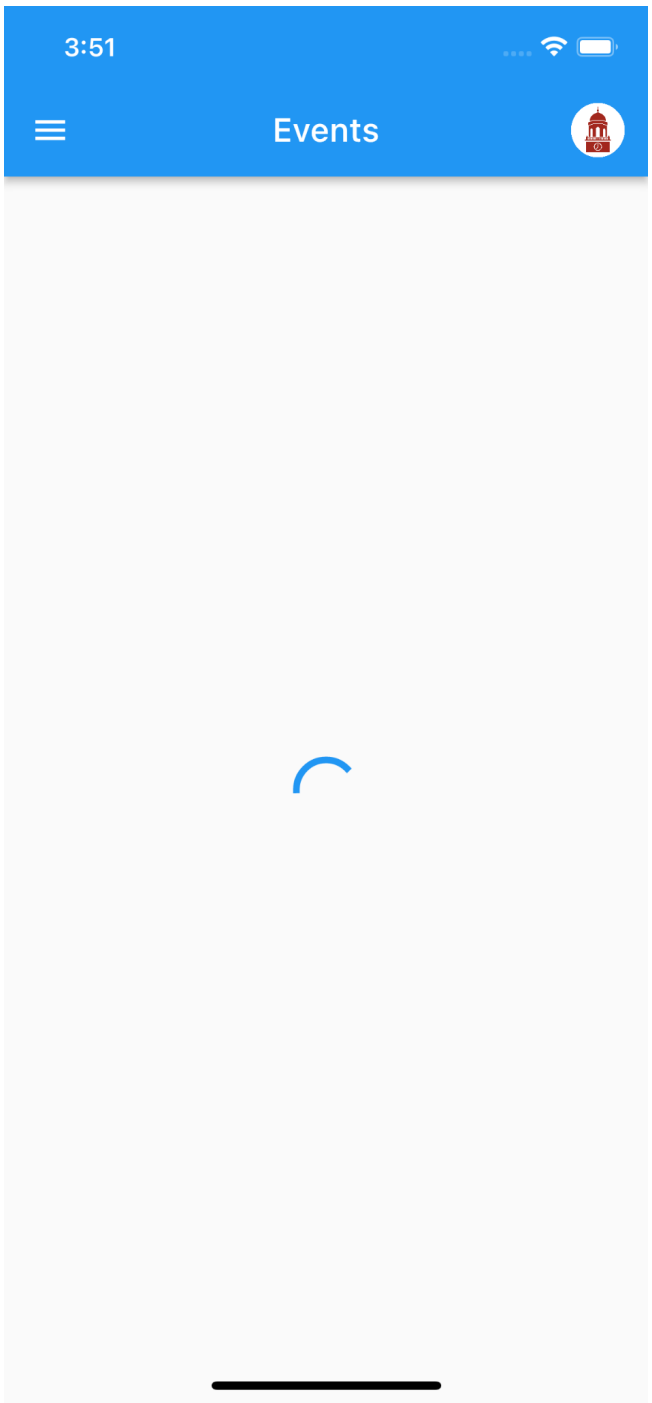
Drawer: Change icons, order of pages, and the design of the selected page.

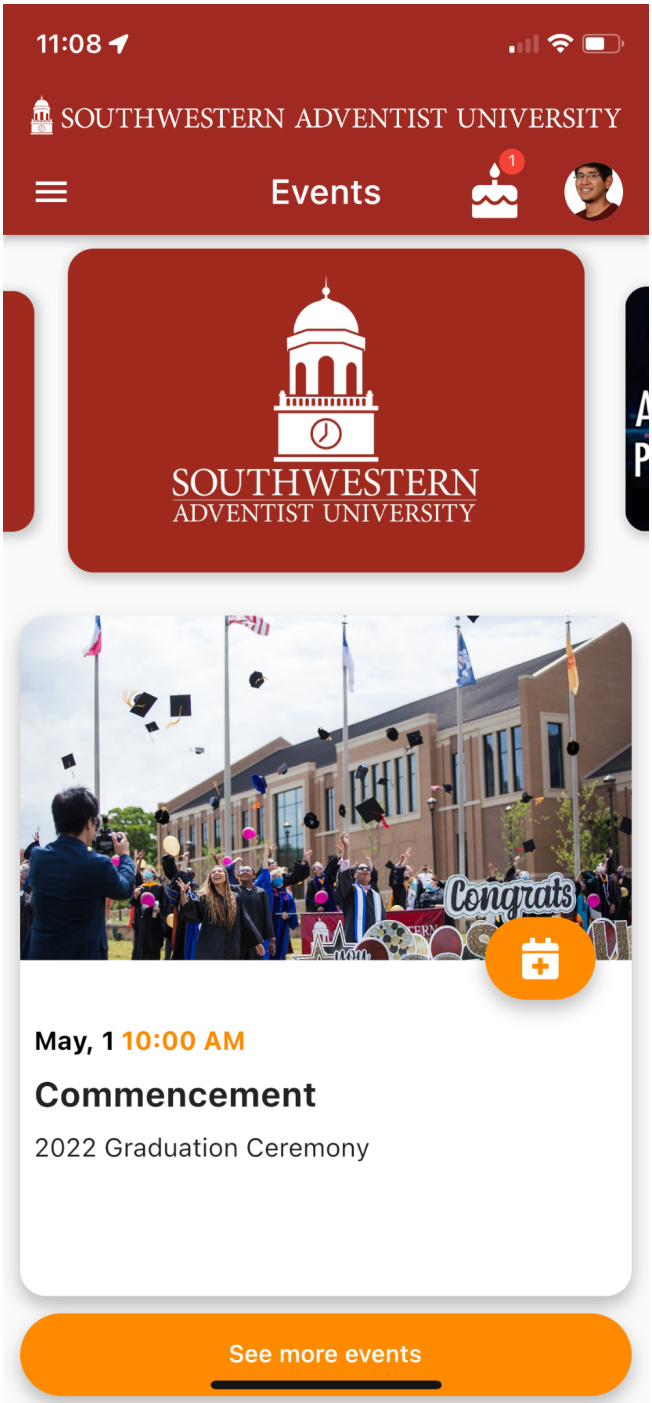


Faculty Directory: The cards look more modern, also I added a search engine.

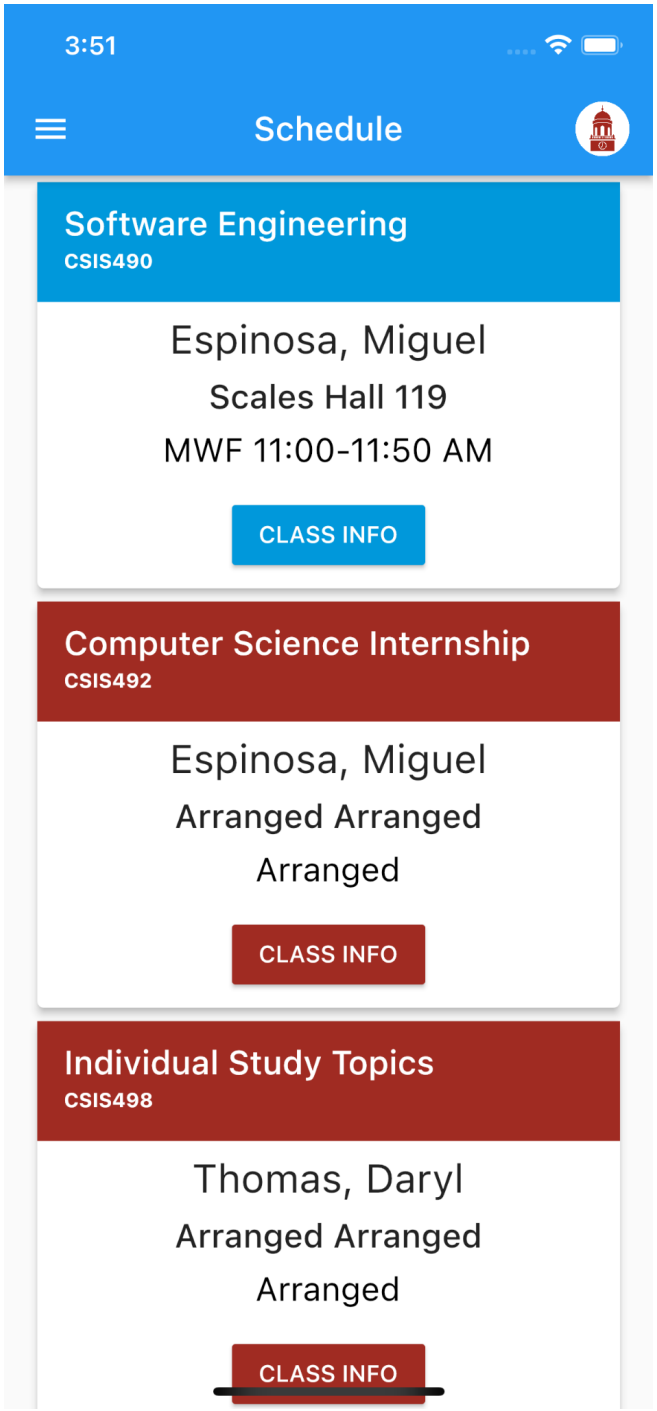


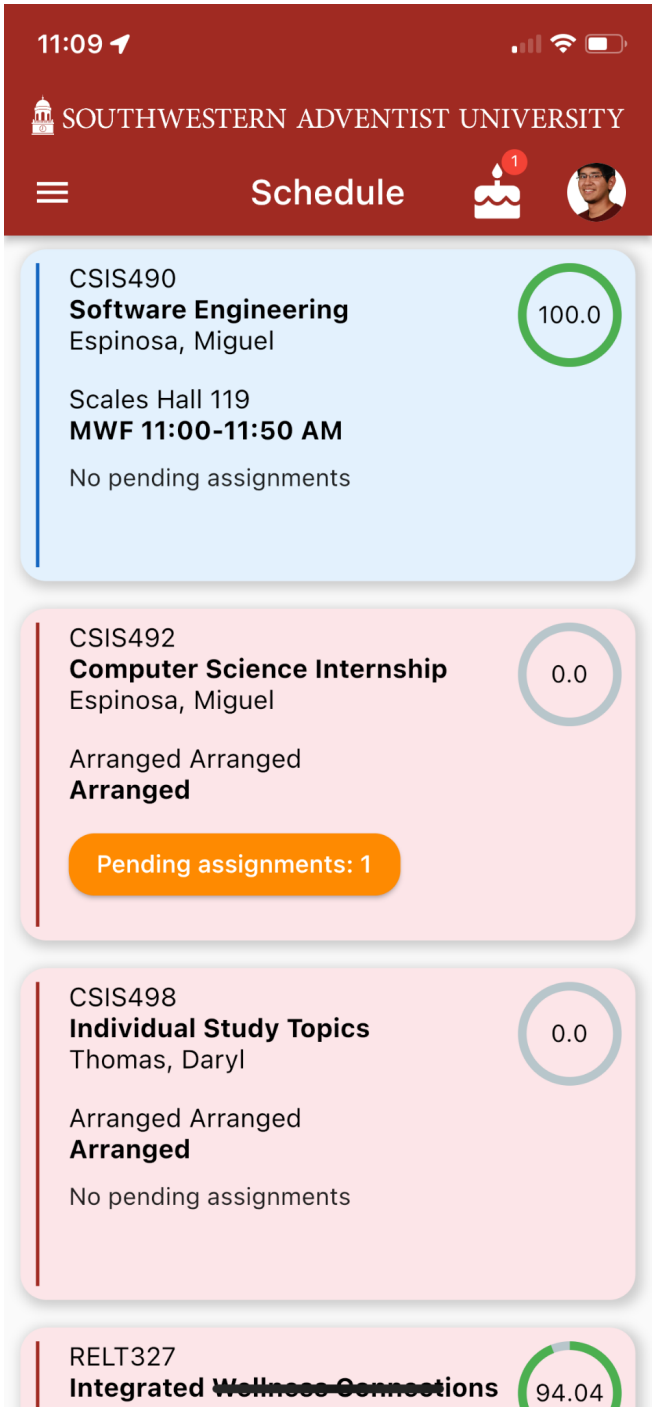
Events: Events page changed a lot since is the home page. On the app bar, we have the option to see the birthdays on that day. The slides from the TVs are now on the events page. Changed the design of the cards, and add a function to add the event to the user calendar. Also, the button at the end opens a link to dudeevents.com. In the old version, the screen was always loading when it didn't have events and now in case of no events, an empty card shows up saying "No events at this moment".



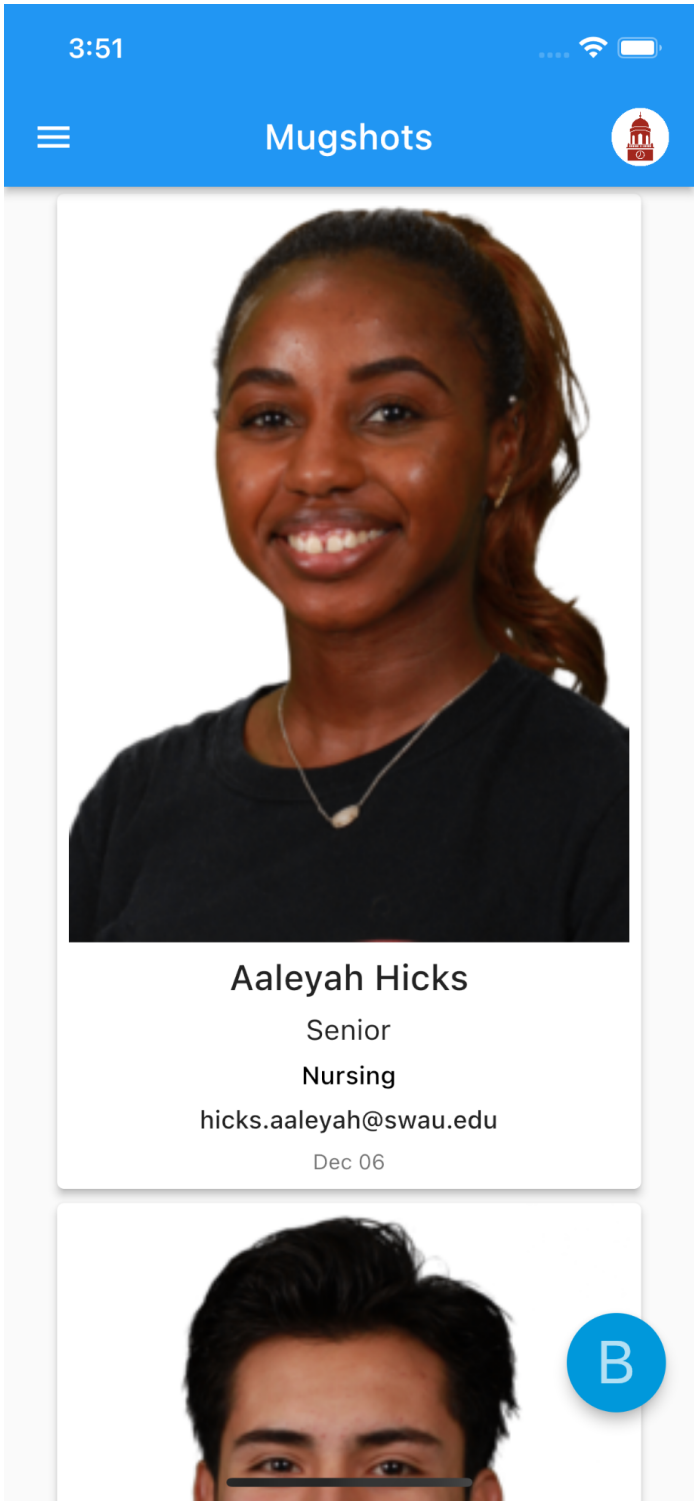



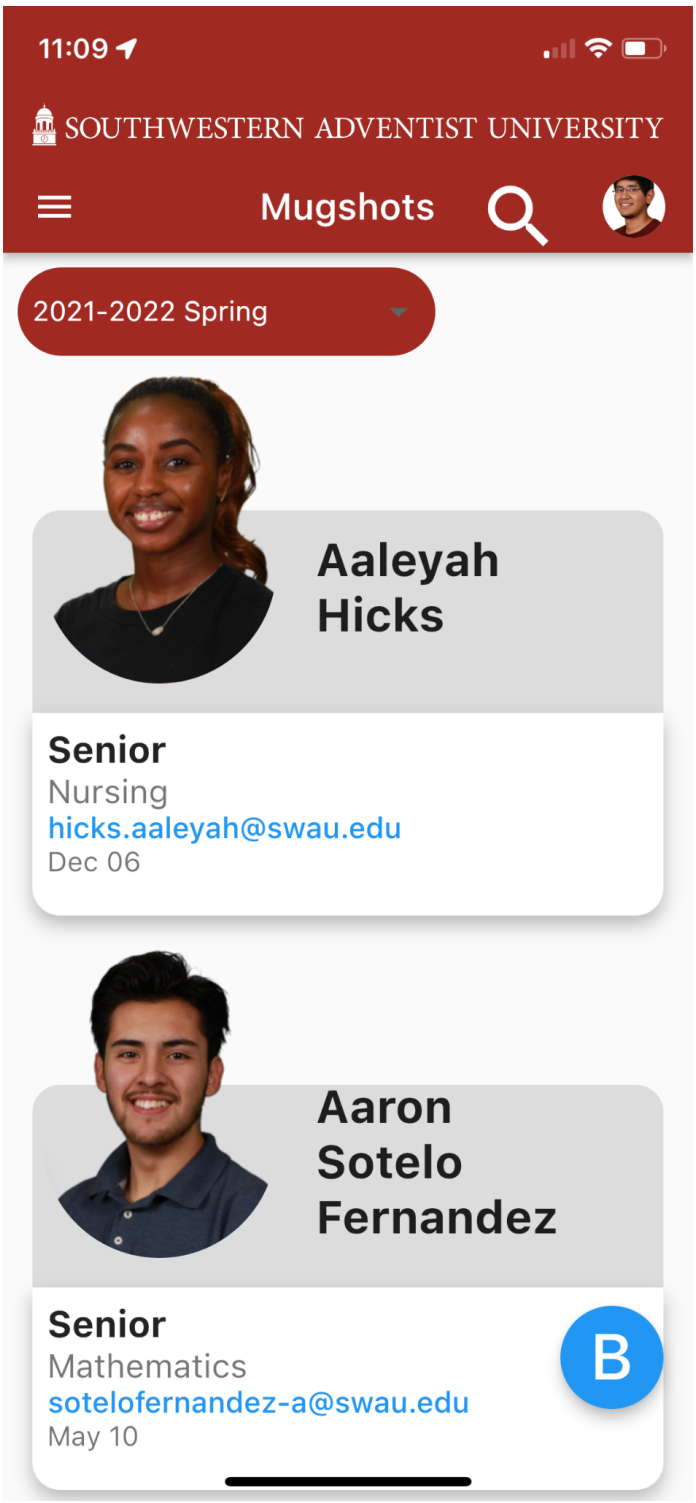
Schedule: I decided to change the style of the cards to look modern and display as much information as possible. Instead of class info, I displayed the score and the number of assignments pending in a button. The blue color represents that that day the student has that class.



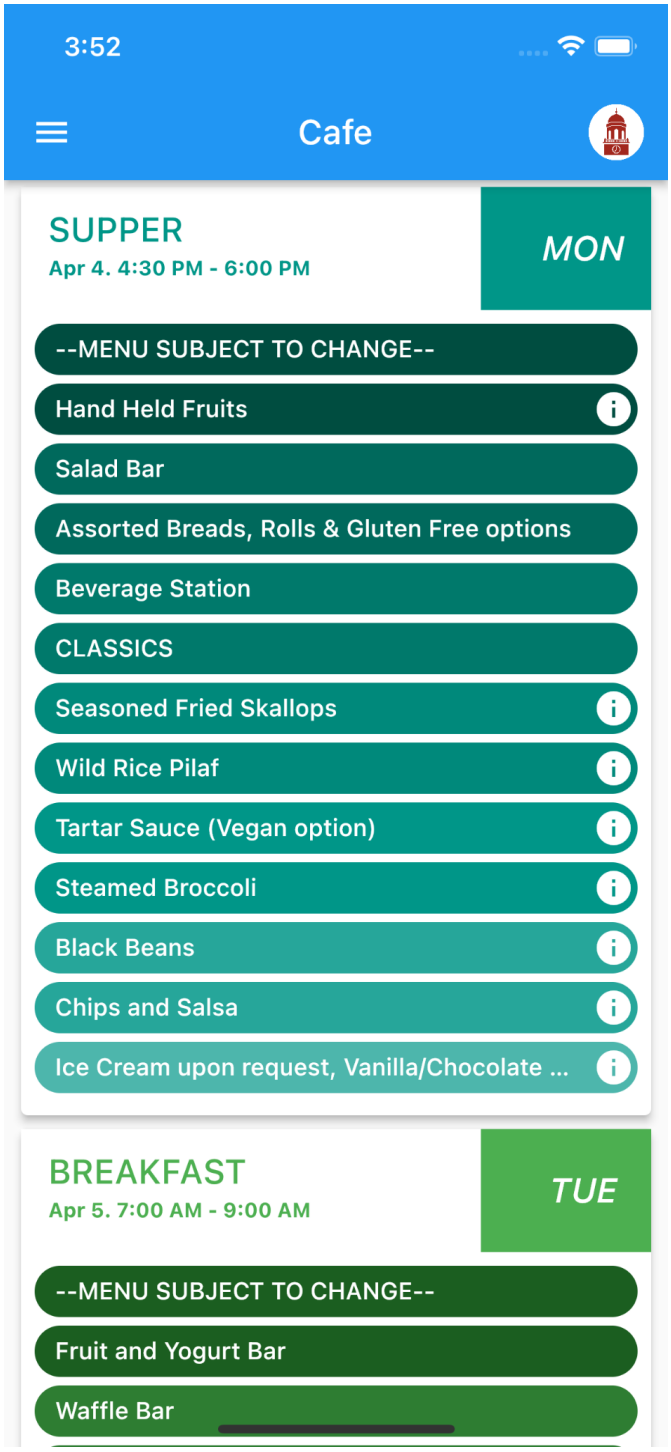



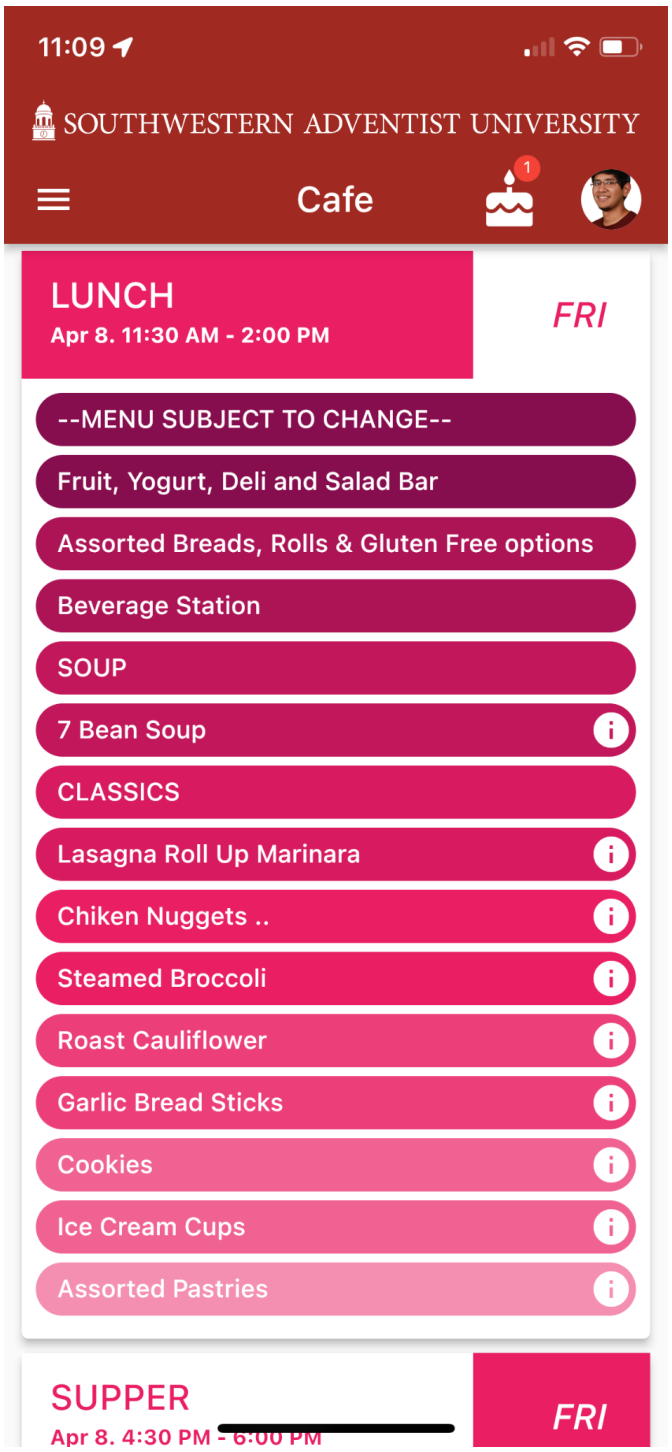
Mugshots: The cards look more modern, also I added a search engine. We can also look for Students of other semesters.



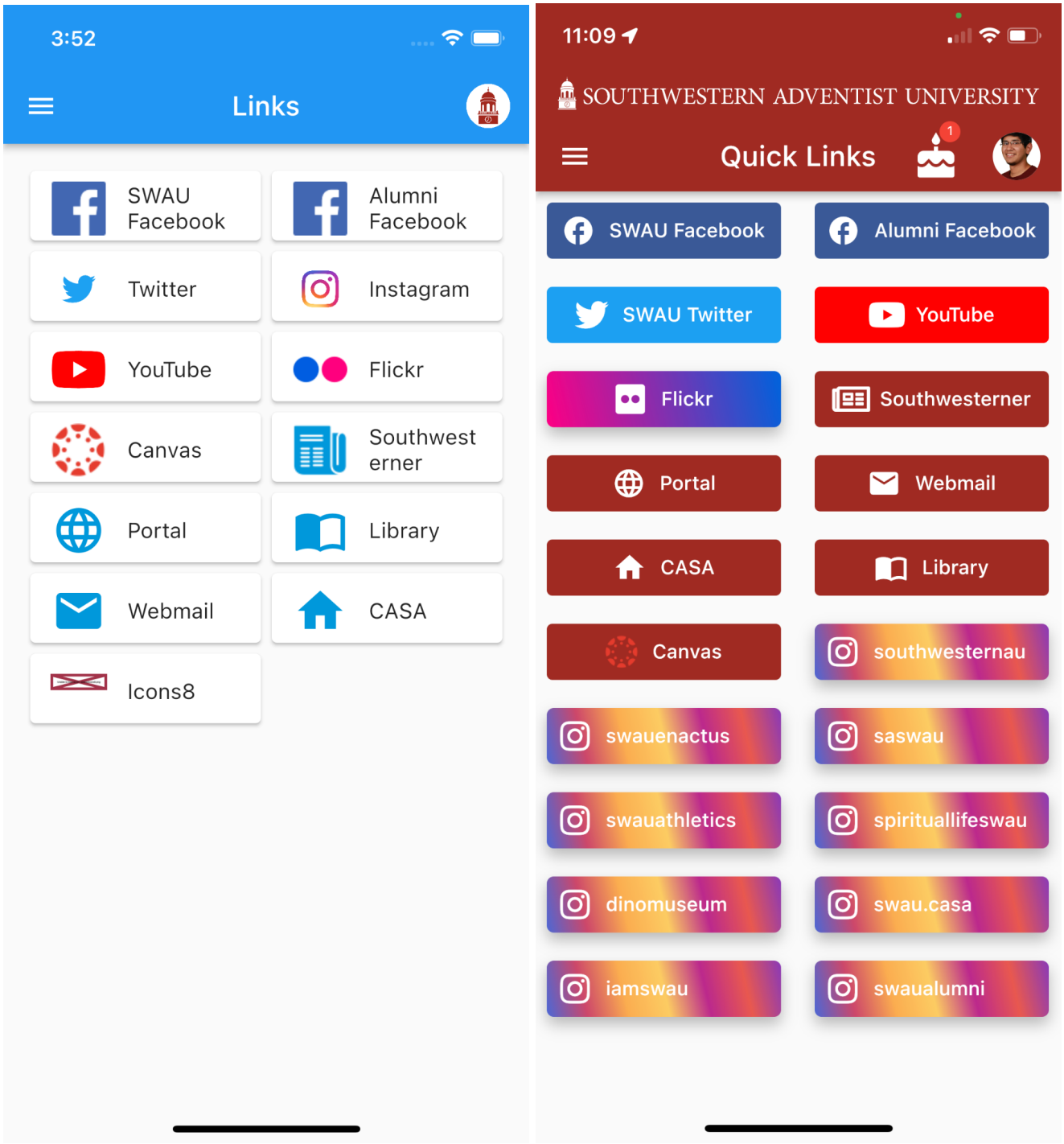



Cafe: Here there were not a lot of changes, just when the user presses the "i" button, there is a dialog popping out instead of going to a whole new page.

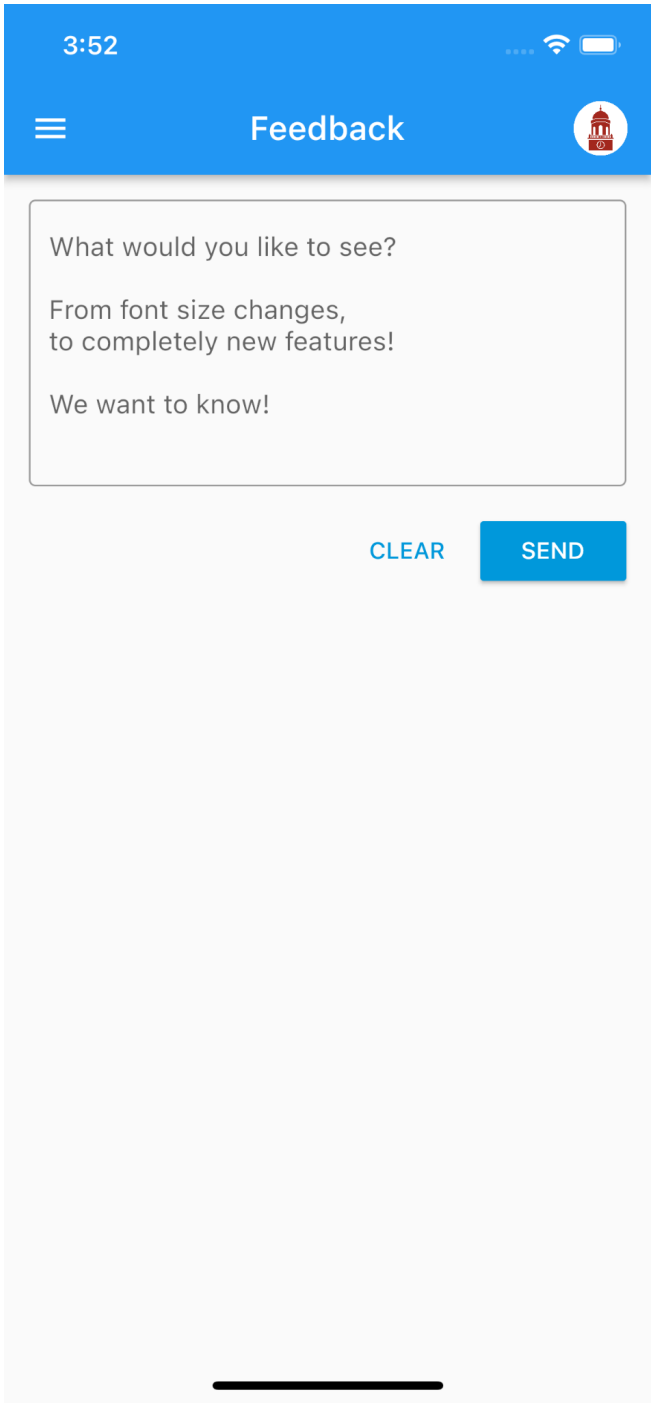


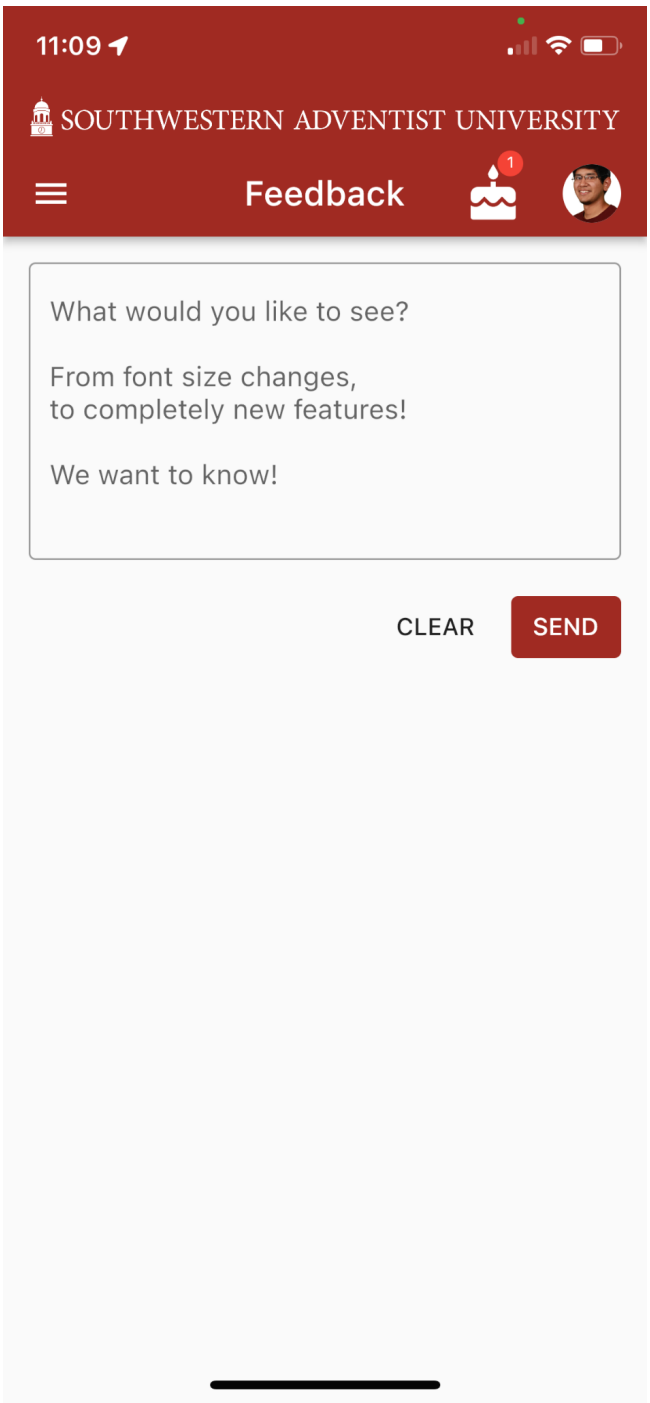



Links: Here I redesign the buttons.

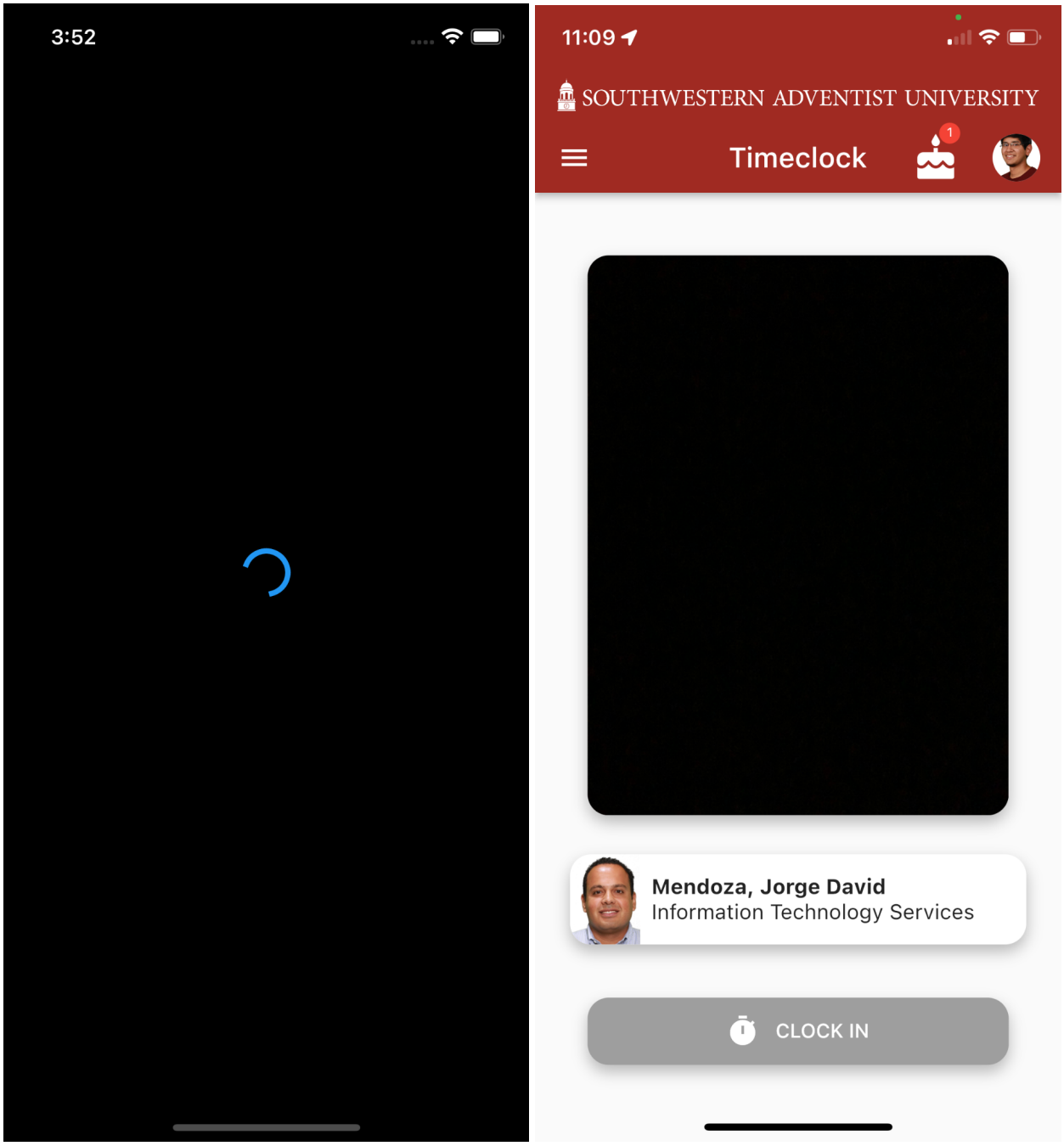


Feedback: No changes here, the code is what changed.

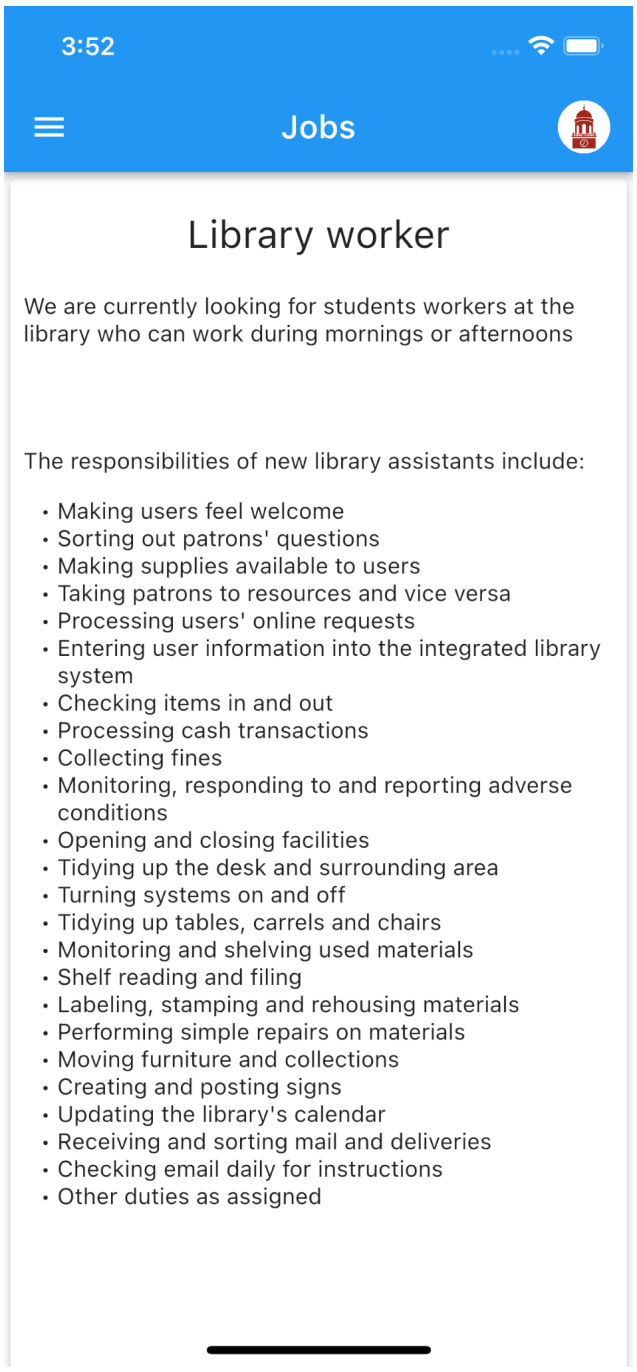


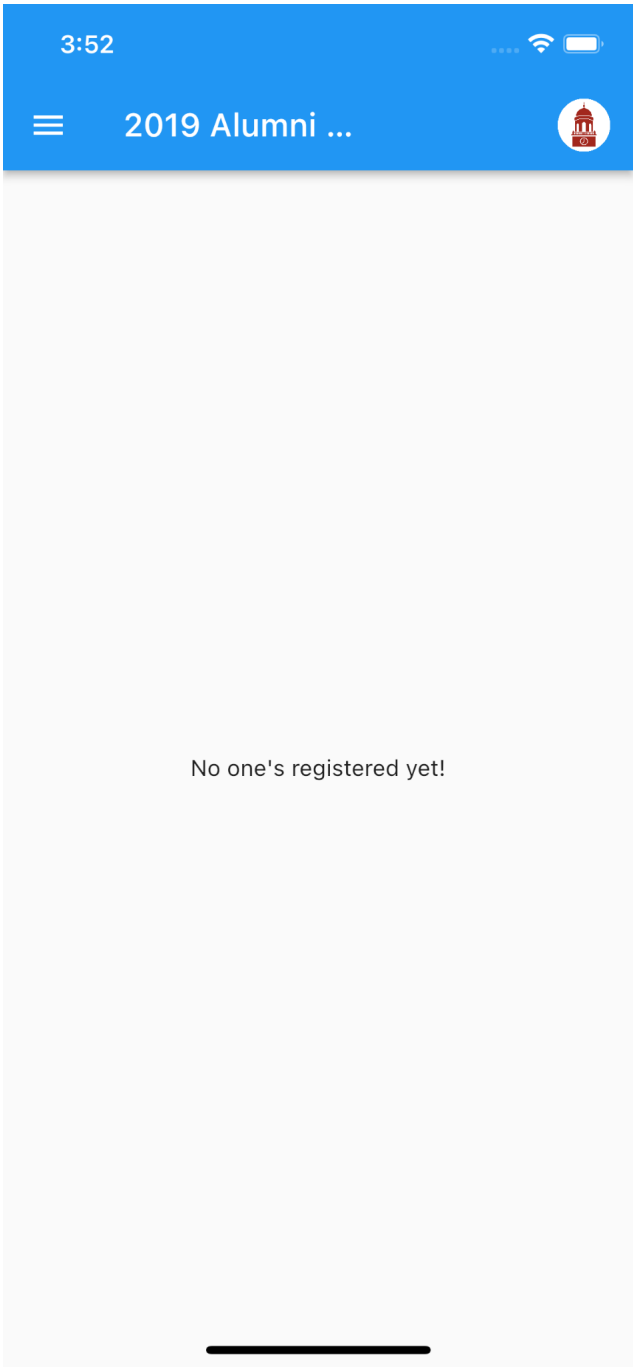



Timeclock: Here I redesigned the camera display and the supervisor's cards. As we can see the old version was not even working.



Deleted pages, jobs, and alumni: The jobs page was deleted after the school decided to use Handshake to apply to the jobs. Alumni was never really used so it got taken out.



**Add new pages to the app**

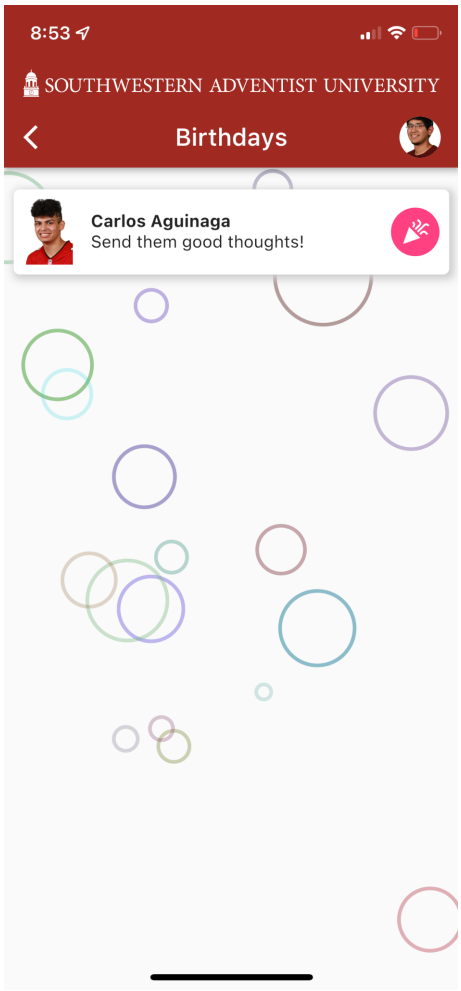
**Description:** Mr. Mendoza wanted to add more functions to the app, like being able to request a shuttle, request a vehicle, display the birthdays, have SLAD small groups, and request an assembly exemption.

**Deliverables and Milestones:** Deliver the best programs possible to make student's life a bit easier.

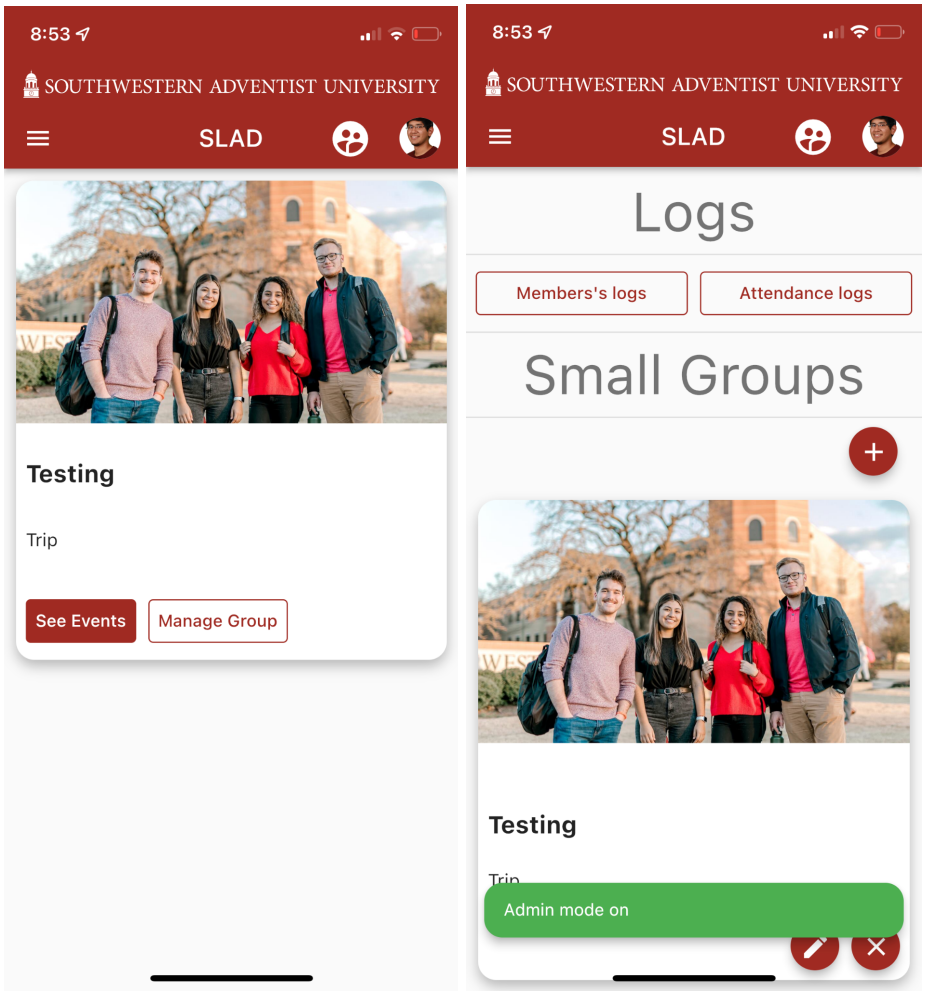
**Resources Needed:** Designs of the pages, back-end for every new page, and create a new database.

**Risks and Contingencies:** Not being able to deliver what was promised and having some possible bugs.

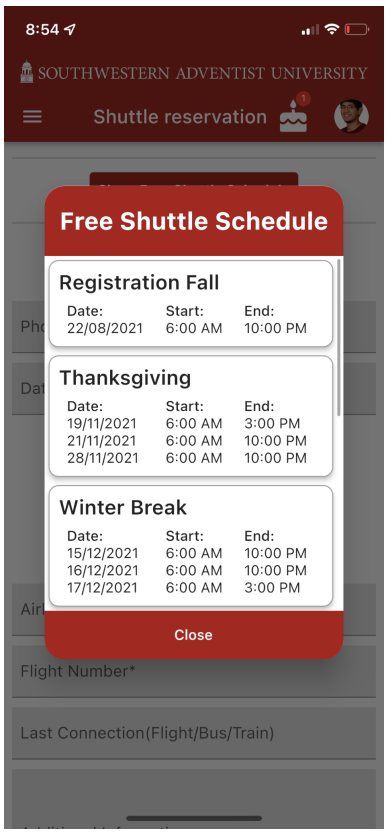
Birthdays page: Here the SWAU students, faculty, and staff that have birthdays that day will show. There is a button with the option to send them an email wishing happy birthday to that person.

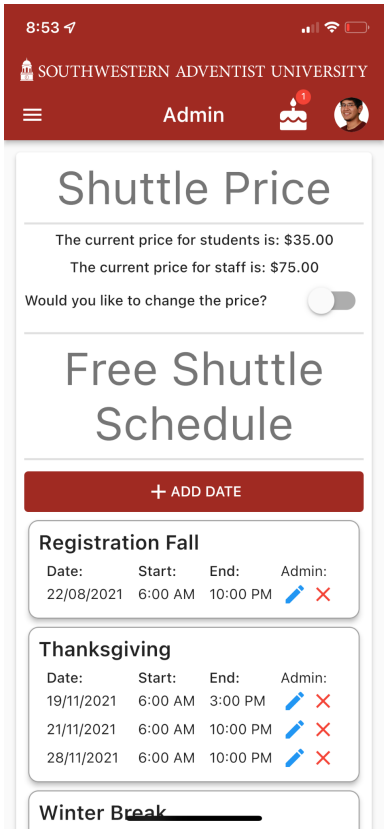


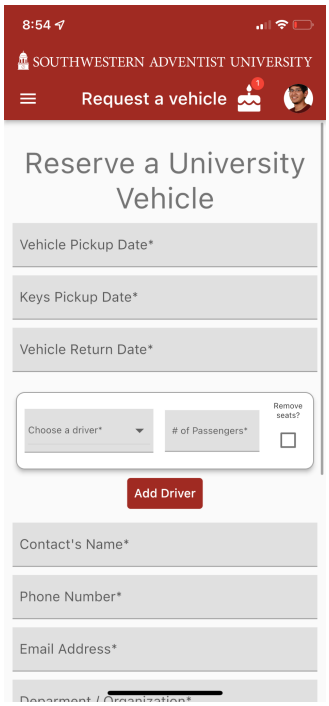
SLAD page: Here the students can join and participate in the small groups around the school. There is an admin side where the admins can delete, edit and add small groups.

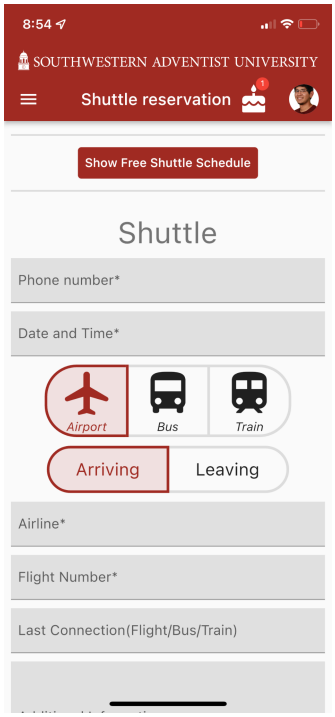


Transportation pages: Here the students can request a shuttle and see the free shuttle schedule. The staff and faculty can request a school vehicle. Admins can modify the free shuttle schedule.

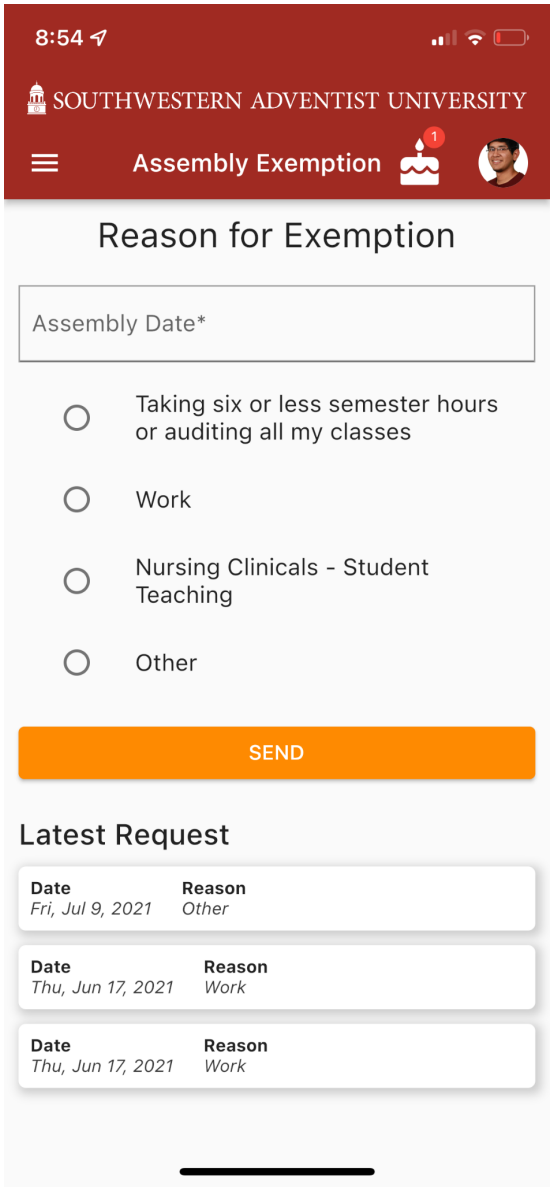


Assembly Exemption page: Here on this page, the students can request an exemption for assembly.



**Create tests**
**Description:** Create and develop a unit test, which is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. See code reference 4.
**Deliverables and Milestones:** Create many successful tests as possible.
**Resources Needed:** Understanding of unit testing with flutter and mockito.
**Risks and Contingencies:** Create bad tests.

**Document the new version**
**Description:** Create the documentation of the mobile app for future developers.
**Deliverables and Milestones:** Create good documentation others can use.
**Resources Needed:** Everything that was used to develop this project.
**Risks and Contingencies:** Create bad documentation and get a bad grade.

## 4.2 Timetable

This project was developed over two years. It went through four different versions. The first version took around six months to develop. Then Flutter changed from Flutter 1 to Flutter 2.0 which introduced null safety. Null Safety in simple words means a variable cannot contain a 'null' value unless you initialized with null to that variable. With null safety, all the runtime null-dereference errors will now be shown in compile time. I refactor the app two more times after that, debugging, cleaning code, and adding new features to the app. Here is roughly the timeline for the last version of the app (5.5.0).

The whole development process took around two months. Working around 12 hours per week.

Week 1: Learning the new method to make small widgets.
Week 2-5: Recoding the old pages and widgets to the new method.
Week 6: Redesigning the drawer and some other widgets.
Week 6-8: Creating unit tests.

# 5. Conclusion

**Future features**

After working for two years on this project there are some functions I would like to recommend some possible features. When I was working on the cafe page, I wanted to add the option for the students to order food from the grill from the app. The caf did not have enough personnel to carry out the project on their side so I did not persuade this feature. For the alumni page, I wanted to create a "social network". The plan was to open the app for alumni and have them provide their information, upload pictures, update their information, and create a way for them to chat inside the app. I did not have time to create this part of the project. Timeclock could display more information but next semester the new system is going online and there would be another timeclock system. Also, the app itself is a bit simple, it can be smart, and analyze data from students to deliver a better experience for them. That is all I have in mind that I could not do.

**Conclusion**

When I first came here to SWAU and found out about the mobile app I promised myself I was going to do it better. Looking back I cannot believe that I did it. It has been a great experience for me. I learned so much, my programming skills improved a lot. I understand and use some concepts that I saw in class. I recommend it to anyone to work on this project. It is fun, and we never know what might happen in the future. I am going to start a business project thanks to this experience. All the work done on the app was so that students, faculty, and staff can enjoy a good mobile app. I am proud of my work, some stuff can be better for sure. Overall it has been a great learning experience. All the feature I added, it was because I was a student who wanted to see them there. It is something am happy to leave for new students to use and enjoy. Thanks to David Mendoza for teaching me so much and allowing me to work on this project.

# 6. Code References

1. iamswau/lib/auth/user_provider.dart

```
 print('Got profile');
      Map claims = await (getDecodedData(token));
      User user =
          User.fromJson(json.decode(response.body), _username,
claims["scope"]);
      if (claims.toString().contains('SWAU Students')) {
        isStudent = true;
        print('FOUND STUDENT');
      }
      if (claims.toString().contains('Campus Services - Staff') ||
          claims.toString().contains('ITS Staff') ||
          claims.toString().contains('ITS - Student Programmers')) {
        isTransportAdmin = true;
      }
      if (claims.toString().contains('SWAU Staff') ||
          claims.toString().contains('SWAU Faculty')) {
        isStaff = true;
        print('FOUND STAFF');
      }
      if (claims.toString().contains('Chaplain - Staff') ||
          claims.toString().contains('ITS Staff') ||
          claims.toString().contains('ITS - Student Programmers') ||
          claims.toString().contains('Campus Ministries')) {
        isSladAdmin = true;
      }
      if (claims.toString().contains('ITS - Student Programmers')) {
        isAdmin = true;
      }
```

2. Comparison between the new and old versions of the providers.

Events:

Old:

```
class EventsBloc with ChangeNotifier {
  //takes url stream and applies to event
  final _eventsSubject = BehaviorSubject<UnmodifiableListView<Event>>();
  Stream<UnmodifiableListView<Event>> get events => _eventsSubject.stream;
```

```
  Client client;

  EventsBloc({client}) : this.client = client != null ? client : Client()
{
    updateEvents();
  }

  //fetch data from url with http.get and putting in _eventsSubject
  Future<Null> updateEvents() async{
    print('Getting Events');
    try {
      final response =
          await
client.get('https://api.swau.edu/v3/advancement/marketing/events').timeout
(const Duration(seconds: 10));
      if(response.statusCode != 200){
        this._eventsSubject.addError('Could not get events:
${response.body}');
        return;
      }

      List<Event> events = [];
      Iterable results = json.decode(response.body);
      for (Map data in results) {
        events.add(Event.fromJson(data));
      }
      print('Putting events on stream');
      _eventsSubject.add(UnmodifiableListView(events));
      // notifyListeners();
    } on TimeoutException catch (_) {
      print('Ran out of time (10 secs)');
      this._eventsSubject.addError('Could not get events. Please try again
later');
    } catch (err) {
      print('Could not get events: ' + err.toString());
      this._eventsSubject.addError('Could not get events:
${err.toString()}');
    }
  }
```

```
  @override
  void dispose() {
    _eventsSubject?.close();
    super.dispose();
  }
}
```

New:

```
class EventsProvider extends ChangeNotifier {
  List<SWAUEvent> events = [];
  Client client;
  bool loading = false, errorState = false, initialized = false;
  String? error;

  EventsProvider({client}) : client = client ?? Client();

  Future<void> fetch() async {
    try {
      initialized = true;
      loading = true;
      final Response response = await client
          .get(Uri.parse('https://apps.swau.edu/tvs/api/v1/events'))
          .timeout(const Duration(seconds: 10));
      if (response.statusCode != 200) {
        errorState = true;
        if (response.statusCode == 404) {
          error = 'Error: 404 page not found, try again later';
        } else if (response.statusCode == 500) {
          error =
              'Error: 500 It is not you, it is us. \nInternal Server
Error\nPlease try later';
        } else {
          error = response.statusCode.toString();
        }
        print('Could not get news: ${response.statusCode}');
        throw Exception('Could not get news: ${response.statusCode}');
      }
      events.clear();
```

```
      Iterable results = json.decode(response.body);
      for (Map<String, dynamic> data in results) {
        events.add(SWAUEvent.fromJSON(data));
      }
    } catch (err) {
      errorState = true;
      error = err.toString();
      if (error!.contains('TimeoutException')) {
        error = 'Internet connection is too slow, try with a faster
connection';
      }
      print('Could not get news: ' + err.toString());
    } finally {
      loading = false;
      notifyListeners();
    }
  }
}
```

3. Old code for the app's pages vs the new code.

Old:

```
class EventsPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    EventsBloc bloc = Provider.of<EventsBloc>(context);
    return Scaffold(
      drawer: buildSwauDrawer(context, 1),
      appBar: buildSwauAppBar(context, "Events"),
      body: RefreshIndicator(
          onRefresh: () => _onRefresh(bloc),
          child: StreamBuilder<UnmodifiableListView<Event>>(
              stream: bloc.events,
              initialData: UnmodifiableListView<Event>([]),
              builder: (context, snapshot) {
                if (snapshot.hasError) {
                  print(snapshot.error);
                  return ListView(children: <Widget>[
```

```
                    Center(child: Padding(
                      padding: const EdgeInsets.all(20.0),
                      child: Text("Could not get events right now.",
style: Theme.of(context).textTheme.title),
                    )),
                    Center(child: Text("Please pull down to try again.",
style: Theme.of(context).textTheme.body1)),
                  ]);
                }
                if (!snapshot.hasData && (snapshot.connectionState ==
ConnectionState.active || snapshot.connectionState ==
ConnectionState.waiting)) {
                  return Center(child: CircularProgressIndicator());
                }
                if (snapshot.hasData && snapshot.data.length == 0) {
                  if (snapshot.connectionState == ConnectionState.active
|| snapshot.connectionState == ConnectionState.waiting) {
                    return Center(child: CircularProgressIndicator());
                  }
                  return ListView(children: <Widget>[
                    Center(child: Padding(
                      padding: const EdgeInsets.all(20.0),
                      child: Text("No events found.", style:
Theme.of(context).textTheme.title),
                    )),
                    Center(child: Text("Please pull down to try again.",
style: Theme.of(context).textTheme.body1)),
                  ]);
                }
                return OrientationBuilder(
                    builder: (BuildContext context, Orientation
orientation) {
                  if (orientation == Orientation.landscape) {
                    return ListView(
                      children: snapshot.data
                          .map((event) => _buildEventLandscape(event,
context))
                          .toList(),
                    );
                  }
```

```
                return ListView(
                  children: snapshot.data
                      .map((event) => _buildEventPortrait(event,
context))
                      .toList(),
                );
              });
            })),
    );
  }

  _onRefresh(EventsBloc bloc) async {
    await bloc.updateEvents();
  }

  _formatDate(Event event) {
    String dateStarts = event.dateStarts;
    String dateEnds = event.dateEnds;
    String date = event.date;
    String startDate;
    String endDate;
    String startTime;
    String endTime;
    String dateRange;

    if (dateStarts != "") {
      startDate =
          DateFormat.yMMMd().format(DateTime.parse(dateStarts).toLocal());
      startTime =
DateFormat.jm().format(DateTime.parse(dateStarts).toLocal());
      endDate =
DateFormat.yMMMd().format(DateTime.parse(dateEnds).toLocal());
      endTime =
DateFormat.jm().format(DateTime.parse(dateEnds).toLocal());
      if (startDate.compareTo(endDate) == 0) {
        dateRange = startDate + " - " + startTime + " to " + endTime;
      } else {
        dateRange =
            startDate + " - " + startTime + " to " + endDate + " - " +
endTime;
```

```
      }
    } else {
      startDate =
DateFormat.yMMMd().format(DateTime.parse(date).toLocal());
      startTime = DateFormat.jm().format(DateTime.parse(date).toLocal());
      dateRange = startDate + " - " + startTime;
    }
    return dateRange;
  }

  _formatDescription(String description) {
    description = description.replaceAll('<br/>', '\n');
    description = description.replaceAll('<p>', '');
    description = description.replaceAll('</p>', '</p>\n');
    var newDesc;

    newDesc = parse(description);
    String parsedDesc = parse(newDesc.body.text).documentElement.text;

    return parsedDesc;
  }

  _formatLocation(String location) {
    if (location.length > 0) {
      var html = parse(location);
      location = parse(html.body.text).documentElement.text;
      return "Location: $location";
    } else {
      return "No Location Provided";
    }
  }

  Widget _buildEventLandscape(Event event, context) {
    ThemeData theme = Theme.of(context);
    return Card(
        elevation: 4.0,
        child: InkWell(
            onTap: () {
              launch(event.link);
            },
```

```
            child: Row(children: <Widget>[
              Container(
                child: FadeInImage.memoryNetwork(
                  placeholder: kTransparentImage,
                  image: event.image,
                ),
              ),
              Expanded(
                child: Container(
                    constraints: BoxConstraints(
                      minHeight: MediaQuery.of(context).size.height /
1.70,
                    ),
                    padding: EdgeInsets.all(8.0),
                    color: Colors.white,
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: <Widget>[
                          Text(
                            event.title,
                            style: theme.textTheme.headline,
                          ),
                          SizedBox(height: 8.0),
                          Text(
                            _formatDate(event),
                            style: theme.textTheme.caption,
                          ),
                          Text(
                            _formatDescription(event.description),
                            style: theme.textTheme.body1,
                          ),
                          Text(
                            _formatLocation(event.location),
                            style: theme.textTheme.body2,
                          ),
                          SizedBox(height: 8.0)
                        ])),
              )
            ])));
```

```
}

Widget _buildEventPortrait(Event event, context) {
  ThemeData theme = Theme.of(context);
  return Card(
      elevation: 4.0,
      child: InkWell(
          onTap: () {
            launch(event.link);
          },
          child: Column(children: <Widget>[
            AspectRatio(
                aspectRatio: 10 / 5,
                child: FadeInImage.memoryNetwork(
                  placeholder: kTransparentImage,
                  image: event.image,
                  fit: BoxFit.fitWidth,
                )),
            Padding(
                padding: EdgeInsets.all(8.0),
                child: Column(children: <Widget>[
                  Text(
                    event.title,
                    textAlign: TextAlign.center,
                    style: theme.textTheme.headline,
                  ),
                  Text(
                    _formatDate(event),
                    textAlign: TextAlign.center,
                    style: theme.textTheme.body2,
                  ),
                  SizedBox(height: 8.0),
                  Padding(
                    padding: EdgeInsets.symmetric(horizontal: 15.0),
                    child: Text(
                      _formatDescription(event.description),
                      textAlign: TextAlign.center,
                      style: theme.textTheme.body1,
                    ),
                  ),
```

```
                    Text(
                      _formatLocation(event.location),
                      textAlign: TextAlign.center,
                      style: theme.textTheme.body2,
                    ),
                    SizedBox(height: 8.0)
                  ]))
            ])));
  }
}
```

New:

```
class EventsPage extends StatelessWidget {
  const EventsPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const Scaffold(
      appBar: PreferredSize(
          preferredSize: Size.fromHeight(40), child: LogoAppBar()),
      body: EventsAppBar(),
    );
  }
}
```

```
class EventsAppBar extends StatelessWidget {
  const EventsAppBar({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const Scaffold(
      appBar: PreferredSize(
          preferredSize: Size.fromHeight(50),
          child: BuildAppBar(
            title: "Events",
          )),
```

```
      drawer: BuildDrawer(index: 1),
      body: EventsBody(),
    );
  }
}
```

```
class EventsBody extends StatelessWidget {
  const EventsBody({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    EventsProvider provider = Provider.of<EventsProvider>(context);
    final _scrollController = ScrollController();

    if (!provider.initialized) {
      provider.fetch();
      return const Center(
        child: CircularProgressIndicator(),
      );
    }
    return provider.errorState
        ? ErrorPage(page: 'events', provider: provider)
        : Scrollbar(
            controller: _scrollController,
            child: SingleChildScrollView(
              controller: _scrollController,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const SlideCarousel(),
                  provider.events.isNotEmpty
                      ? Center(
                          child: Wrap(
                            children: provider.events
                                .map((e) => EventsCard(event: e))
                                .toList(),
                          ),
                        )
                      : Center(
```

```
                          child: Wrap(children: const
[EventsEmptyCard()])),
                  const Center(child: EventsMoreEvens()),
                ],
              ),
            ),
          );
  }
}
```

```
class EventsCard extends StatelessWidget {
  final SWAUEvent event;
  const EventsCard({Key? key, required this.event}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(10),
      child: Container(
        width: 355,
        height: 395,
        decoration: const BoxDecoration(
          borderRadius: BorderRadius.all(Radius.circular(15)),
          color: Colors.white,
          boxShadow: [
            BoxShadow(
              color: Colors.black26,
              blurRadius: 8.0,
              offset: Offset(0.0, 5.0),
            ),
          ],
        ),
        child: Stack(
          children: [
            Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisSize: MainAxisSize.min,
              children: [
```

```
                Center(
                  child: SizedBox(
                      width: 355,
                      height: 200,
                      child: ClipRRect(
                        borderRadius: const BorderRadius.only(
                          topLeft: Radius.circular(15),
                          topRight: Radius.circular(15),
                        ),
                        child: Hero(
                          tag: event.id,
                          child: Image.memory(base64.decode(event.image),
                              fit: BoxFit.cover),
                        ),
                      )),
                ),
                const SizedBox(height: 30),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      _formatDate(event),
                      const SizedBox(height: 10),
                      Text(
                        event.title.toString(),
                        style: const TextStyle(
                          fontSize: 20,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      const SizedBox(height: 10),
                      Text(
                        event.description.toString(),
                        maxLines: 4,
                        overflow: TextOverflow.clip,
                        style: const TextStyle(
                          fontSize: 15,
                        ),
                      ),
```

```
                    ],
                  ),
                ),
              ],
            ),
            Padding(
              padding: const EdgeInsets.only(left: 270, top: 175),
              child: Container(
                decoration: const BoxDecoration(
                  borderRadius: BorderRadius.all(Radius.circular(50)),
                  color: Color(0xFFFF8A01),
                  boxShadow: [
                    BoxShadow(
                      color: Colors.black26,
                      blurRadius: 8.0,
                      offset: Offset(0.0, 5.0),
                    ),
                  ],
                ),
                child: TextButton(
                    onPressed: () {
                      launch(

'https://apps.swau.edu/tvs/api/v1/mobile/calendar/${event.id}');
                    },
                    style: TextButton.styleFrom(
                      backgroundColor: Colors.transparent,
                    ),
                    child: const Icon(
                      FontAwesomeIcons.solidCalendarPlus,
                      color: Colors.white,
                    )),
              ),
            )
          ],
        ),
      ),
    );
  }
```

```
  RichText _formatDate(SWAUEvent event) {
    String dateStarts = event.starts.toString();
    String dateEnds = event.ends.toString();
    String startDate;
    String endDate;
    String startTime;
    String endTime;

    startDate = DateFormat("MMMM, d").format(DateTime.parse(dateStarts));
    startTime = DateFormat.jm().format(DateTime.parse(dateStarts));
    endDate = DateFormat("MMMM, d").format(DateTime.parse(dateEnds));
    endTime = DateFormat.jm().format(DateTime.parse(dateEnds));
    if (startDate.compareTo(endDate) == 0) {
      return RichText(
        text: TextSpan(
          style: const TextStyle(
              fontSize: 15, color: Colors.black, fontWeight:
FontWeight.bold),
          children: <TextSpan>[
            TextSpan(
              text: startDate + " ",
            ),
            TextSpan(
              text: startTime,
              style: const TextStyle(
                color: Color(0xFFFF8A01),
              ),
            ),
          ],
        ),
      );
    } else {
      return RichText(
        text: TextSpan(
          style: const TextStyle(
              fontSize: 15, color: Colors.black, fontWeight:
FontWeight.bold),
          children: <TextSpan>[
            TextSpan(
              text: startDate + " ",
```

```
            ),
            TextSpan(
              text: startTime,
              style: const TextStyle(
                color: Color(0xFFFF8A01),
              ),
            ),
            const TextSpan(
              text: " - ",
              style: TextStyle(fontWeight: FontWeight.normal),
            ),
            TextSpan(
              text: endDate + " ",
            ),
            TextSpan(
              text: endTime,
              style: const TextStyle(
                color: Color(0xFFFF8A01),
              ),
            ),
          ],
        ),
      );
    }
  }
}
```

4. Unit test in flutter.

```
@GenerateMocks([http.Client, FlutterSecureStorage])
void main() {
  testWidgets('Events Page', (WidgetTester tester) async {
    final client = MockClient();

    when(client.get(Uri.parse('https://apps.swau.edu/tvs/api/v1/events')))
        .thenAnswer((_) async => http.Response(

'[{"id":"45a29629-a04d-464d-8880-4b03ebbc5f30.jpg","dateCreated":"0001-01-
```

```
01T00:00:00Z","contentType":"image/jpeg","name":"IMG_3134.jpg","size":1000
321,"starts":"2022-05-01T10:00:00Z","ends":"2022-05-01T12:00:00Z","timeout
":0,"title":"Commencement","description":"2022 Graduation
Ceremony","image":""}]',
            200));

    EventsProvider ep = EventsProvider(client: client);
    UserProvider up = UserProvider(client: client);

    await tester.pumpWidget(MultiProvider(providers: [
      ChangeNotifierProvider<EventsProvider>(create: (_) => ep),
      ChangeNotifierProvider<UserProvider>(create: (_) => up),
    ], child: const MaterialApp(title: 'IAMSWAU', home: EventsPage())));

    expect(find.text('Events'), findsOneWidget);
    expect(find.image(FileImage(File('assets/slide_1.png'))),
findsNothing);
  });

  testWidgets('Find event title', (WidgetTester tester) async {
    final client = MockClient();

    when(client.get(Uri.parse('https://apps.swau.edu/tvs/api/v1/events')))
        .thenAnswer((_) async => http.Response(

'[{"id":"45a29629-a04d-464d-8880-4b03ebbc5f30.jpg","dateCreated":"0001-01-
01T00:00:00Z","contentType":"image/jpeg","name":"IMG_3134.jpg","size":1000
321,"starts":"2022-05-01T10:00:00Z","ends":"2022-05-01T12:00:00Z","timeout
":0,"title":"Commencement","description":"2022 Graduation
Ceremony","image":"iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR
42mP8/5+hHgAHggJ/PchI7wAAAABJRU5ErkJggg=="}]',
            200));

    EventsProvider ep = EventsProvider(client: client);
    UserProvider up = UserProvider(client: client);
    SlidesProvider sp = SlidesProvider(client: client);

    await tester.pumpWidget(MultiProvider(providers: [
      ChangeNotifierProvider<EventsProvider>(create: (_) => ep),
      ChangeNotifierProvider<UserProvider>(create: (_) => up),
```

```
      ChangeNotifierProvider<SlidesProvider>(create: (_) => sp),
    ], child: const MaterialApp(title: 'IAMSWAU', home: EventsPage())));

    await tester.pumpAndSettle();

    expect(find.text('Commencement'), findsOneWidget);
  });

  testWidgets('Show Slides', (WidgetTester tester) async {
    // await mockNetworkImagesFor(() async {
    final FlutterSecureStorage storage = MockFlutterSecureStorage();
    final client = MockClient();

    String token =

"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsib3BlbmlkIiwic3NvIl0sImF1
dGhfdGltZSI6MTY0OTE2NzE5NCwiYXpwIjoic3NvIiwiY2FtcHVzX2lkIjoiMjYxNTExIiwiY2
lkIjoic3NvIiwiY2xpZW50X2lkIjoic3NvIiwiZW1haWwiOiJoaXJhbUBzd2F1LmVkdSIsImV4
cCI6MTY0OTc3MTk5NCwiZmlyc3RfbmFtZSI6IkhpcmFtIiwiZ3JhbnRfdHlwZSI6ImF1dGhvcm
l6YXRpb25fY29kZSIsImlhdCI6MTY0OTE2NzE5NCwiaXNzIjoiaHR0cHM6Ly9hcHAuc3dhdS5l
ZHUvb2F1dGgvdG9rZW4iLCJsYXN0X25hbWUiOiJIZXJuYW5kZXogR2FyY2lhIiwibmFtZSI6Ik
hpcmFtIEhlcm5hbmRleiBHYXJjaWEiLCJwZXJzb25faWQiOiIyNjE1MTEiLCJwaG90b19pZCI6
ODk5Nywic2NvcGUiOlsib3BlbmlkIiwiSGFuZHNoYWtlRXh0ZXJuYWwiLCJJVFMgLSBTdHVkZW
50IFByb2dyYW1tZXJzIiwiQ2xhc3NpZmllZC1MIiwiU3R1ZGVudHNlcnZpY2VzLUwiLCJBbm5v
dW5jZS1MIiwiVlBOIEFsbG93IiwiQ2FmZSAtIFdvcmtlcnMiLCJJVFMgLSBTdHVkZW50IFdvcm
tlcnMiLCJOZXR3b3JrIC0gU3R1ZGVudHMiLCJTV0FVIFN0dWRlbnRzIl0sInVzZXJfaWQiOiIy
NjE1MTEiLCJ1c2VyX25hbWUiOiJoaXJhbSIsInVzZXJuYW1lIjoiaGlyYW0ifQ.ndomC__7Qhi
w5rPkJ0B6X07_fI-bXnilmC29-01yE-mV1reKzHpSiYm-0SQNDj2UtdaiTKjjdbCVtOKzRJZyH
WCJYdSXiM13kS0ZPmkRNS6bn8hoxvnSpdKNHtpp0BuXKsrLbizDs14b6LKxfQj-SRFPJ1thI-S
M4PlIwTIcOtMtyNc4aeUAlmEC26F9VolhWaOQYhsS4IoHSISX_gRFdkamDSd5l_nOlNhHP2HNT
IDZirm2cLO-ezpAeTyk5wUAr_Fmgfd1HbEVwN4wqqWFjFPffkEXkBzqCRAf-gvtdEf_L2uDtY3
npT3RDF8CWKVkN6vmN76Ice21-oOfJ-Y_4Q";

    when(storage.read(key: "swau_token")).thenAnswer((_) async => token);

    when(client.get(Uri.parse('https://api.swau.edu/v3/iamswau/profile'),
        headers: {
          'Authorization': 'Bearer $token'
        })).thenAnswer((_) async => http.Response(
```

```
        '{"personId":261511,"campusId":261511,"name":"Hiram Hernandez
Garcia","webName":"","firstName":"Hiram","middleName":"","lastName":"Herna
ndez
Garcia","preferedFirstName":"","dateofBirth":"2000-06-30T00:00:00Z","gende
r":"M","DegreeMajorName":"Computer Science","ResidencyStatus":"Residence
Halls","syPhotoID":"https://java2.swau.edu/IDCard/getPhoto.jsp?type=2\u002
6id=8997","studentEmailAddress":"hiram@swau.edu","Level":"Senior","status"
:"","type":"","fullTime":"","department":"","position":"","workPhone":"","
degrees":"","classification":""}',
        200));

    when(client.get(Uri.parse('https://apps.swau.edu/tvs/api/v1/events')))
        .thenAnswer((_) async => http.Response(

'[{"id":"45a29629-a04d-464d-8880-4b03ebbc5f30.jpg","dateCreated":"0001-01-
01T00:00:00Z","contentType":"image/jpeg","name":"IMG_3134.jpg","size":1000
321,"starts":"2022-05-01T10:00:00Z","ends":"2022-05-01T12:00:00Z","timeout
":0,"title":"Commencement","description":"2022 Graduation
Ceremony","image":""}]',
            200));

    when(client.get(

Uri.parse('https://apps.swau.edu/tvs/api/v1/birthdays/mobile')))
        .thenAnswer((_) async => http.Response(
            '[{"id":"255861","firstName":"Gloria","lastName":"Miller
Rivera","photoID":"https://java2.swau.edu/IDCard/getPhoto.jsp?type=2\u0026
id=8435","email":"grivera@swau.edu"},{"id":"268696","firstName":"Christian
","lastName":"Tarbox","photoID":"https://java2.swau.edu/IDCard/getPhoto.js
p?type=2\u0026id=9855","email":"ctarbox@swau.edu"}]',
            200));

    UserProvider up = UserProvider(client: client);
    EventsProvider ep = EventsProvider(client: client);
    SlidesProvider sp = SlidesProvider(client: client);
    BirthdaysProvider bp = BirthdaysProvider(client: client);

    up.storage = storage;
    // await up.getUser();
```

```
    await tester.pumpWidget(MultiProvider(providers: [
      ChangeNotifierProvider<EventsProvider>(create: (_) => ep),
      ChangeNotifierProvider<UserProvider>(create: (_) => up),
      ChangeNotifierProvider<SlidesProvider>(create: (_) => sp),
      ChangeNotifierProvider<BirthdaysProvider>(create: (_) => bp),
    ], child: const MaterialApp(title: 'IAMSWAU', home: EventsPage())));
    // });
    expect(find.text('Commencement'), findsOneWidget);
    // expect(find.byWidget(const SlideCarousel()), findsOneWidget);
  });
}
```

```
@GenerateMocks([http.Client])
void main() {
  testWidgets('Got Events Successfully', (WidgetTester tester) async {
    final client = MockClient();

    when(client.get(Uri.parse('https://apps.swau.edu/tvs/api/v1/events')))
        .thenAnswer((_) async => http.Response(

'[{"id":"45a29629-a04d-464d-8880-4b03ebbc5f30.jpg","dateCreated":"0001-01-
01T00:00:00Z","contentType":"image/jpeg","name":"IMG_3134.jpg","size":1000
321,"starts":"2022-05-01T10:00:00Z","ends":"2022-05-01T12:00:00Z","timeout
":0,"title":"Commencement","description":"2022 Graduation
Ceremony","image":""}]',
            200));

    EventsProvider ep = EventsProvider(client: client);

    await ep.fetch();

    expect(ep.events.length, 1);
    expect(ep.events[0].title, 'Commencement');
  });
}
```