Faculty of Engineering and Applied Science

SOFE 4640U Mobile Application Development

Date: November 4, 2022

Project Report

[Calsync Latest Version (github.com)]

Group 12

Name: Preet Patel
Student ID: 100708239

Name: Tiwaloluwa Ojo
Student ID: 100700622

Name: Aaditya Rajput
Student ID: 100622434

Problem Statement

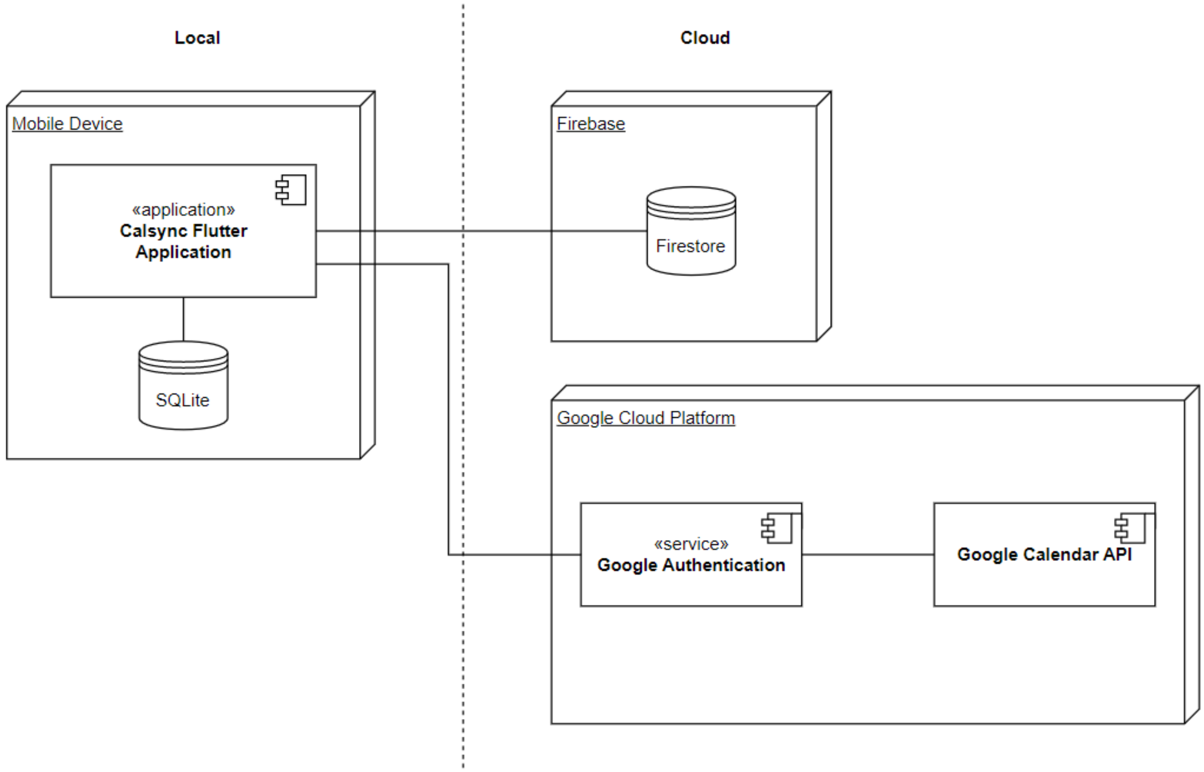
Maintaining a schedule is a difficult task. Gone are the days of composing and managing multiple events on a singular paper copy of a calendar. They do not notify you of your event nor are they easy to view. Poor user interface often results in missed events/deadlines. Our application will be focusing on designing a mobile calendar application for managing and maintaining personal schedules. It will provide a user-friendly interface that can be tailored to personal preferences.

Goals, Requirements, and Analysis

ID	Description	Analysis (Yes/No)
1	View Events in different calendar views - Monthly, Weekly, Daily.	Yes
2	View Events in a list view.	Yes
3	Create, Delete, and Edit Events - Type, Name, Description, Date, Time	Yes
4	Send reminders to user prior to event	No - Time constraint presented by issues prevented the group from implementing this project
5	Sync google calendar events with the application	Yes
6	Search events by name or date	No - time constraint presented by issues prevented the group from implementing this project

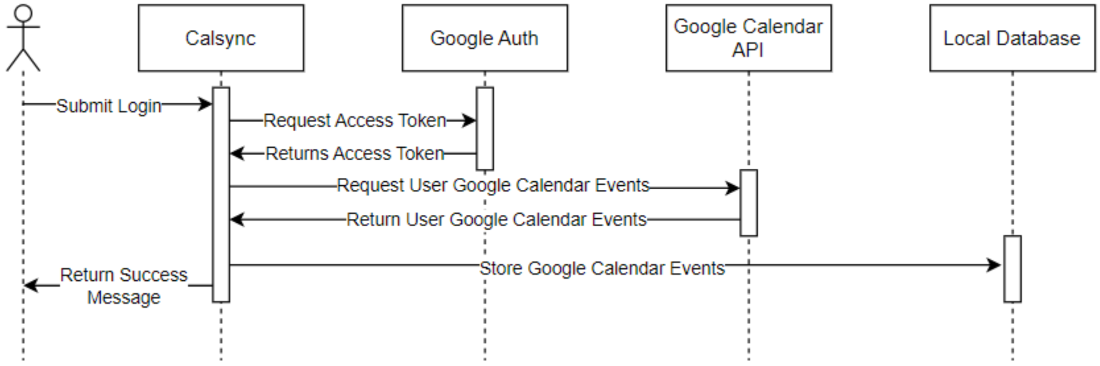
Architecture

The architecture of the Calsync application can be described with the various different diagrams shown in this section. The following deployment diagram highlights how the various components which were used as part of the application are deployed:

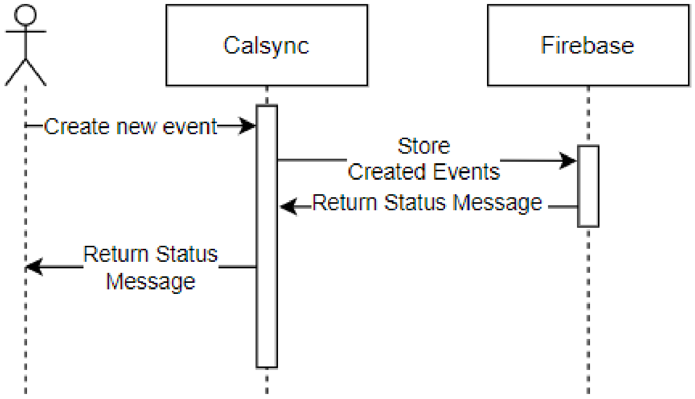


The deployment diagram separates the components into two different environments which are local and cloud. In the local environment, which in this case would be any mobile device, there is the Calsync application that was developed using Flutter and there is also the SQLite database. The SQLite database stores all the events that are fetched from the user's google calendar account. In the cloud environment, the first container we have is Firebase which is used for hosting the cloud database called firestore. In firestore, any events created in the calsync application are stored so that they can be transferred between devices. Secondly, there is the Google Cloud Platform container in which we have the Google Authentication service and the Google Calendar API. The Google Authentication service is used for getting the user permission and access token which allows the Calsync application to access and fetch the user's google calendar events. The Google Calendar API is used by the application with the obtained access token to actually fetch all the Google calendar events of the user and then these events are stored in the local SQLite database. The user can login to multiple accounts and get events from multiple different accounts.

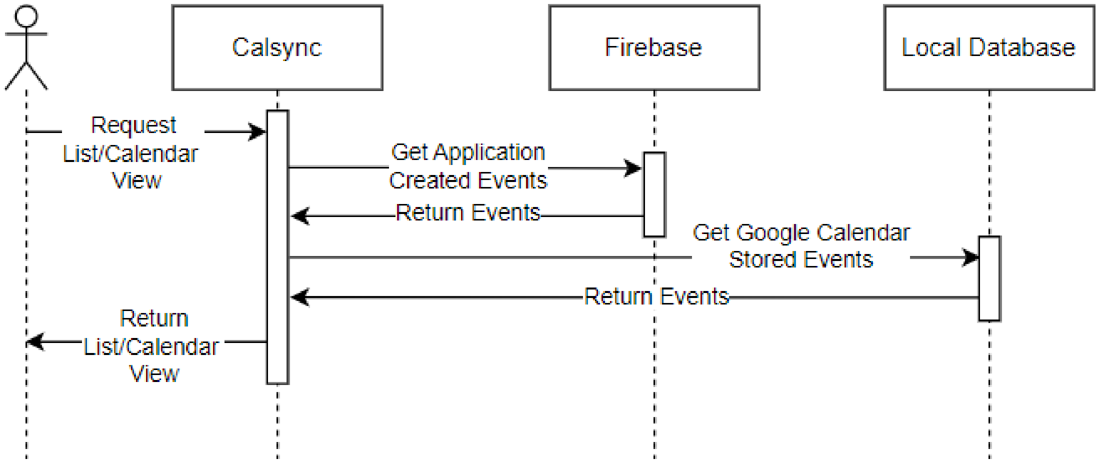
The following sequence diagrams explain the various processes in the application and how they are performed:



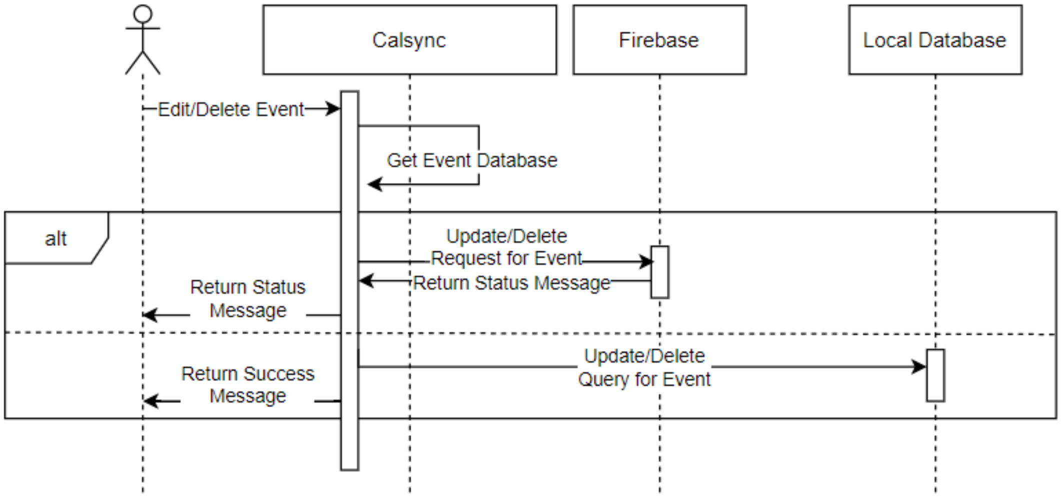
The above sequence diagram explains how the Calsync application fetches all the user's Google Calendar events to be used by the application. First when the user opens the application, the application prompts the user to login to their google account of choice and agree to the request permissions required by the application which in this case is the permission to access their Google calendar. Once the user enters the login details, the application communicates with the Google Authentication service to get the access token which then is used to make a request to the Google Calendar API to get a list of the user's events. Once the Google Calendar API returns the user's calendar events, the Calsync application stores all the events in the local SQLite database. The application then sends a success message to the user on the user interface to show that the synchronization process was successful. Next, the second sequence diagram explains the creation of events within the Calsync application.



The above sequence diagram explains the creation of calendar events in the Calsync application. First, the user will open, fill out and submit the add event form from the user interface. Then, the application will create a request to the firebase firestore cloud database to store the created event. The firebase will then return a status message to the application which checks if the request was successful and if it was then a success message is shown to the user. The third sequence diagram below shows how the different views such as the list view and calendar view (includes daily, weekly, and monthly views) are created and updated.


Calsync
Firebase
Local Database
Request List/Calendar View
Get Application Created Events
Return Events
Get Google Calendar Stored Events
Return Events
Return List/Calendar View

The user first opens the list view or the calendar view from the navigation bar in the user interface. The application then creates a get request to the firestore datastore which will then return all the events which are on the user account (query using the user email). Then the application will also query the local SQLite database to get all the events which were fetched from the user's Google calendar. Once the application has the events from both the local and cloud database, it will then render the updated list or calendar view. The last sequence diagram explains how the remaining CRUD operations are performed.



The above diagram explains how the Calsync application performs the edit event and delete event operations. To make either request, the user has to open the event from the calendar, which opens a pop up with the edit or delete options. Edit option will ask the user to fill out a form similar to the add event form whereas delete option will simply create the delete request to the application. To perform either operation, the application will first check where the event is stored which it can get from an attribute of the event object which contains the information whether the event is on firebase or SQLite. Depending on the storage different paths are followed. If the event is on the firebase firestore database then the application will make a request to update/delete the event which will then return a status message for the request which the application displays on the user interface. On the other hand, if the event is stored locally, the application will perform an update or delete query and return a success message. The above discussed diagram covers the entire architecture and processes of the Calsync application.

Development Tools and Technologies

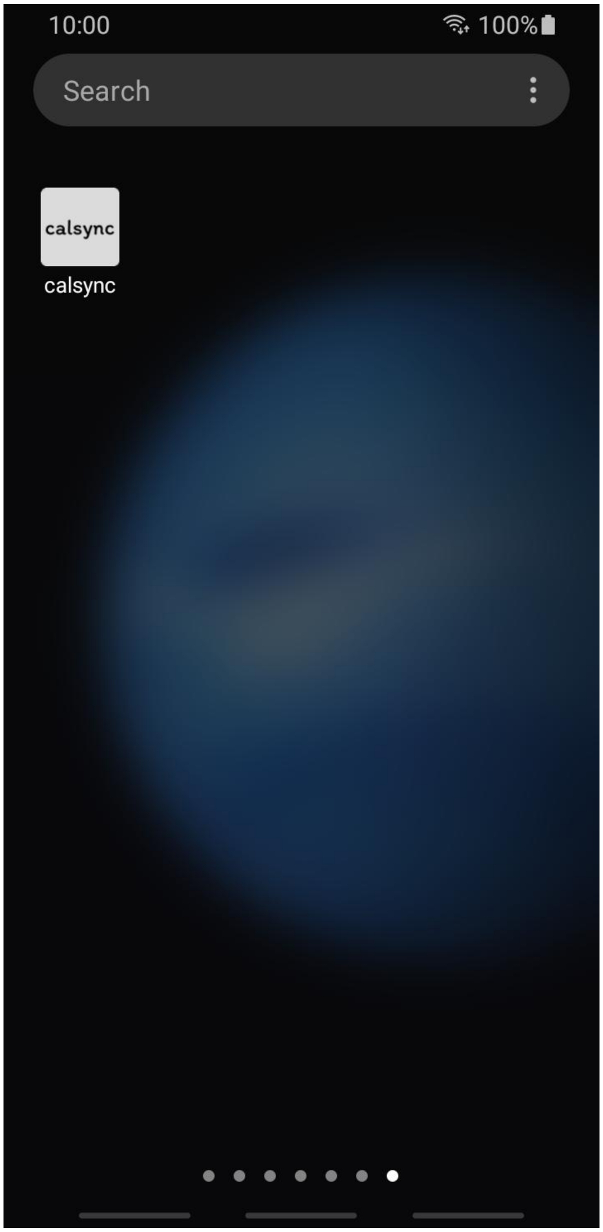
We used various tools in our project to help address both its functional and non-functional requirements. These tools will allow us to notify our users of our purpose with their data as well as the scope. We will be able to also compile for multiple targets, of which Android and IOS devices are our primary targets. Below are a description of the tools we will be using in our project

- Flutter- Cross-platform (Android and IOS) application development
    - We can cross compile for multiple targets and have a single code base.
- Google Calendar API - Events synchronizations with the Google Calendar API for reading and writing
- Firestore (Firebase) - a cloud based No-SQL database for managing large collections of data
- SQLite- Events store for locally created events on our mobile calendar app
- Various other Dart Packages such as Shared Preferences, Syncfusion and Providers. These will be used for tasks such as saving user themes and app personalization settings, and manipulating state at different times during the app's lifecycle.

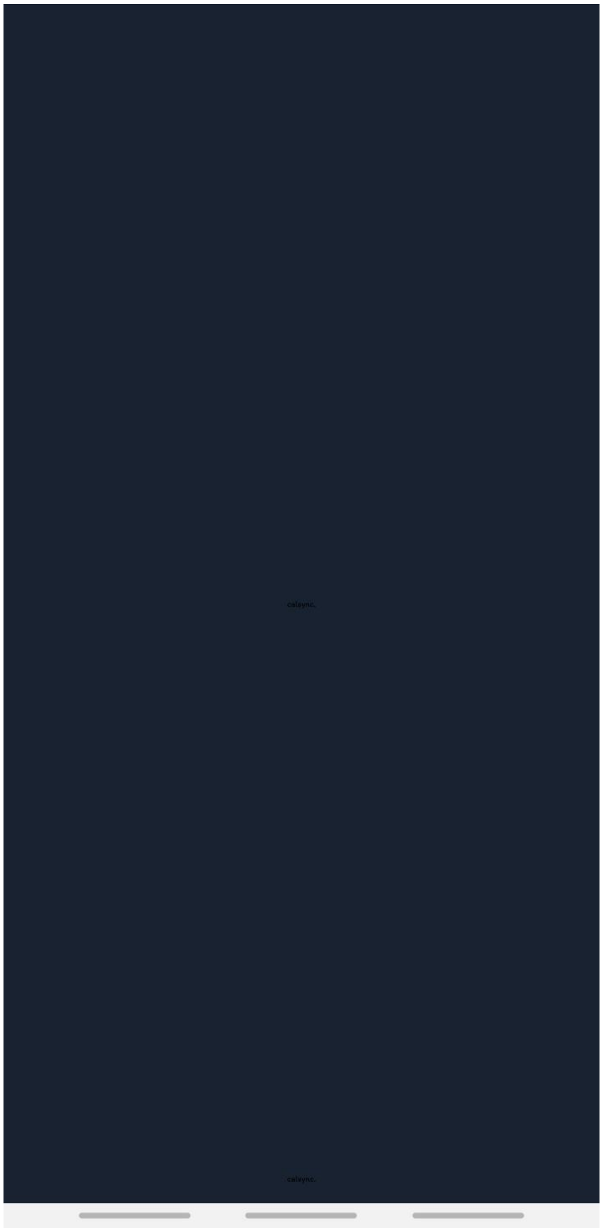
Moreover, we utilized both VSCode as well as Android Studio IDE as our environment for development as well as used physical devices for debugging purposes. Github was our source for version control. Github allowed the team to simultaneously deliver code and updates with little to no interruption to other members of the team.

User interface

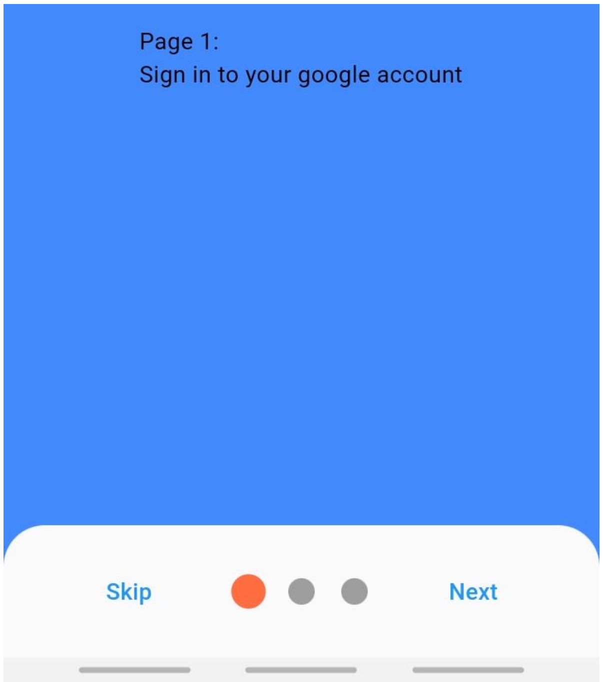
The following series of images below will feature the applications functionality as well as the two themes implemented. Moreover, they were captured on an Android 10 Samsung S9 mobile device.



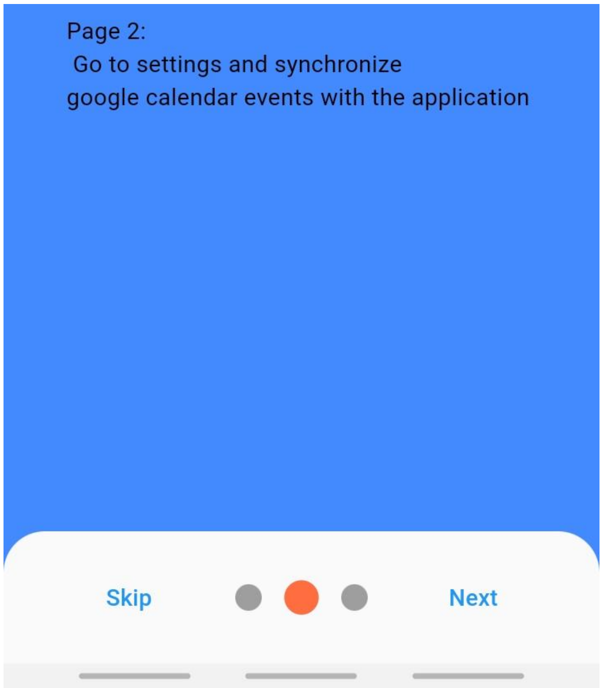
App in the app drawer with app logo and app name



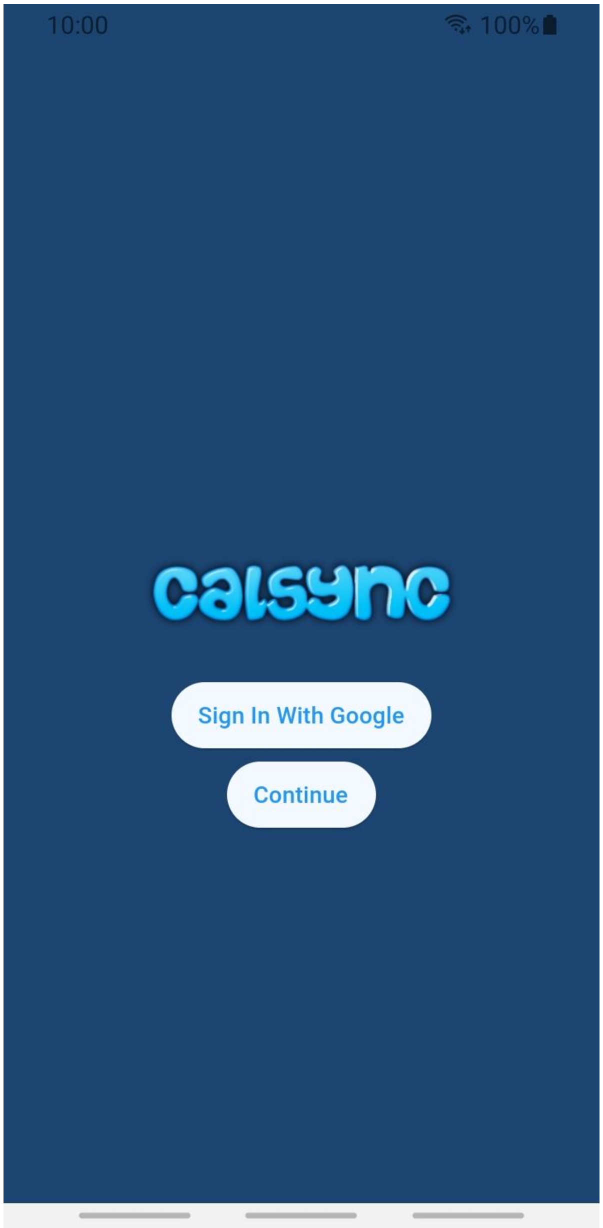
Splash screen with logo in center and branding at bottom. This bug could not be replicated in the Android Studio Emulators



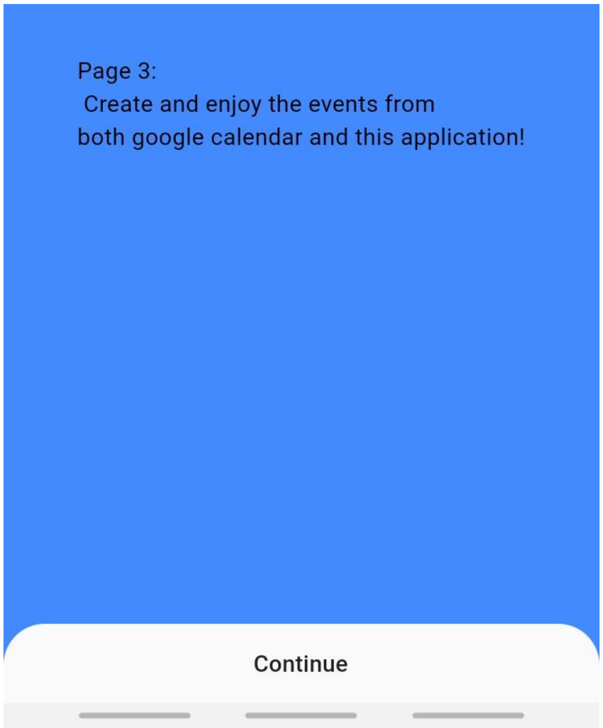
Page 1:
Sign in to your google account



Page 2:
Go to settings and synchronize
google calendar events with the application

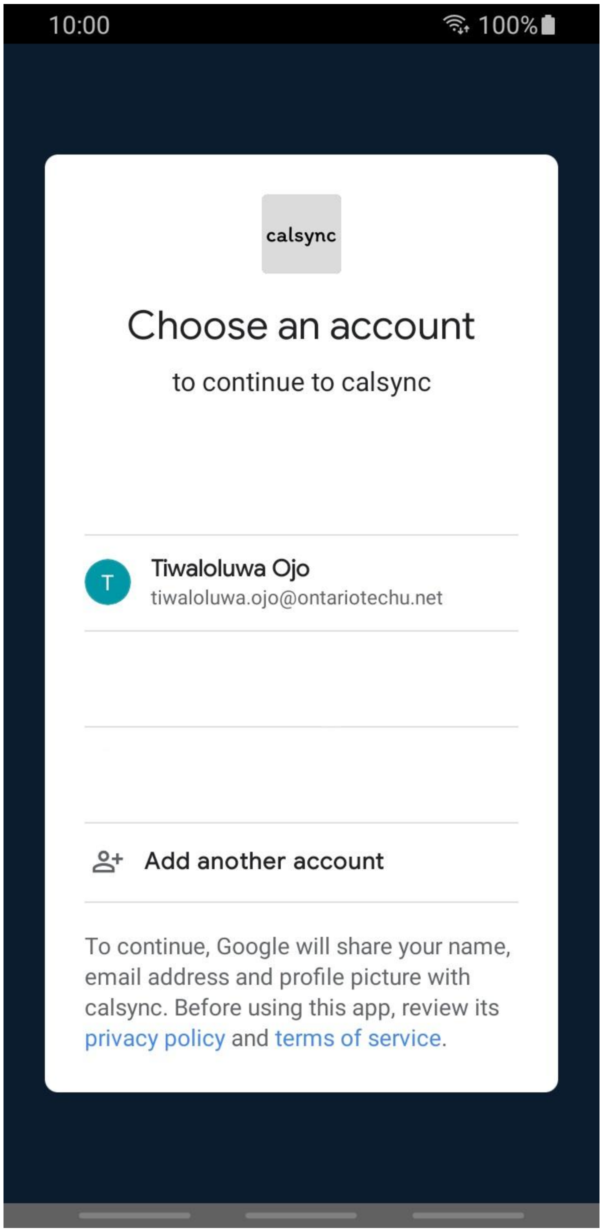


Login Page(Light Mode)



Page 3:
Create and enjoy the events from
both google calendar and this application!

Onboarding pages



Choose an account

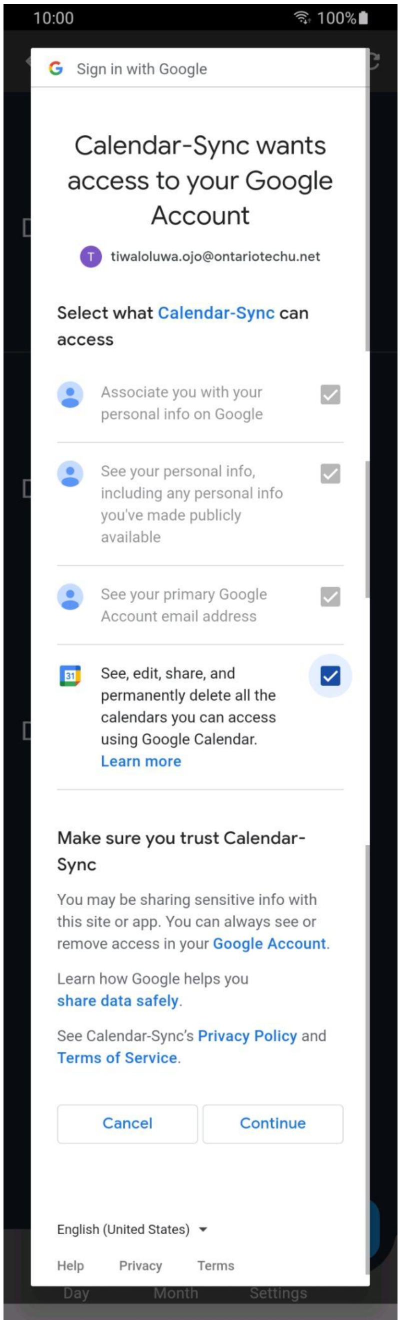
to continue to calsync

Tiwaloluwa Ojo
tiwaloluwa.ojo@ontariotechu.net

Add another account

To continue, Google will share your name, email address and profile picture with calsync. Before using this app, review its privacy policy and terms of service.

OAuth Login



Calendar-Sync wants access to your Google Account

tiwaloluwa.ojo@ontariotechu.net

Select what Calendar-Sync can access

☑ Associate you with your personal info on Google

☑ See your personal info, including any personal info you've made publicly available

☑ See your primary Google Account email address

☑ See, edit, share, and permanently delete all the calendars you can access using Google Calendar. Learn more

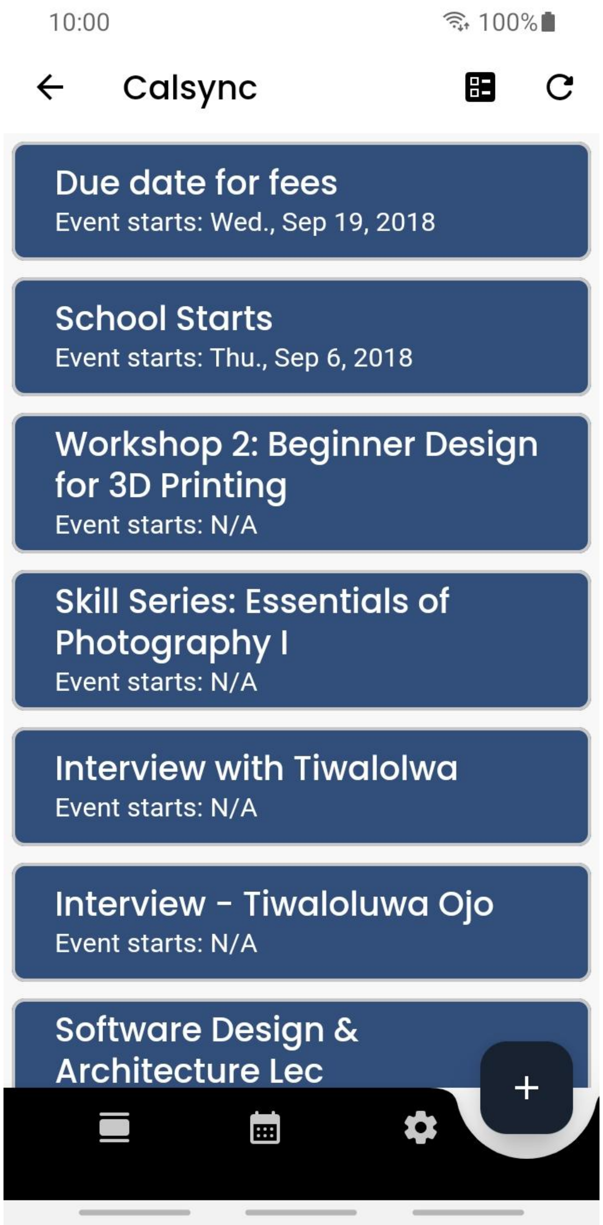
Make sure you trust Calendar-Sync

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See Calendar-Sync's Privacy Policy and Terms of Service.

OAuth with Scope to CRUD events on Google Calendars



10:00 100%

Calsync

Due date for fees
Event starts: Wed., Sep 19, 2018

School Starts
Event starts: Thu., Sep 6, 2018

Workshop 2: Beginner Design for 3D Printing
Event starts: N/A

Skill Series: Essentials of Photography I
Event starts: N/A

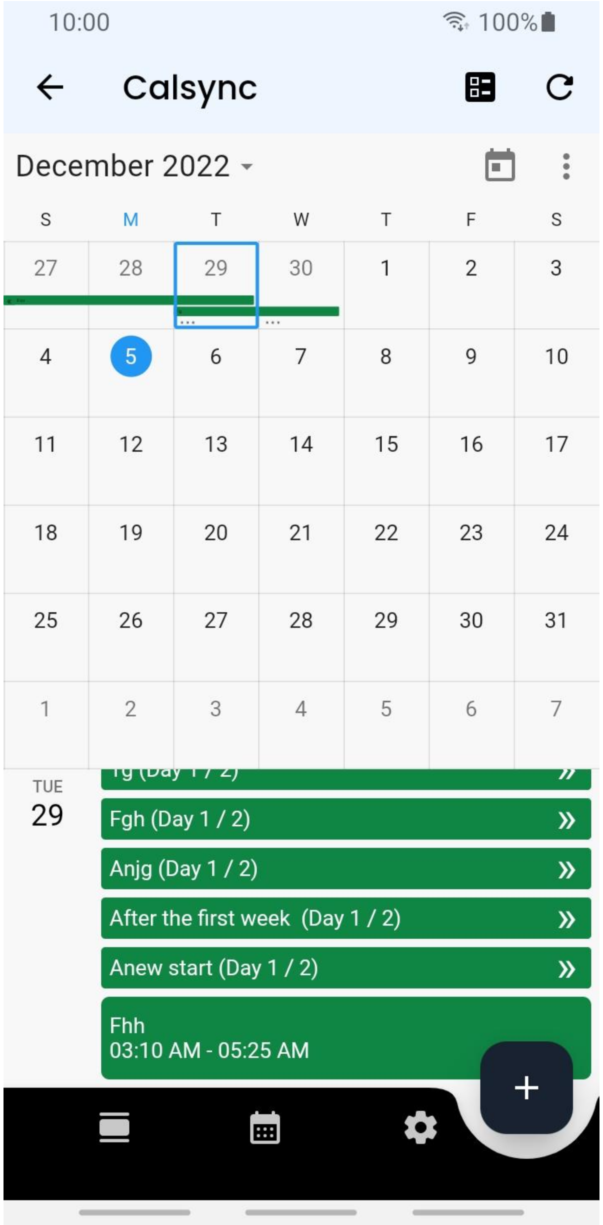
Interview with Tiwalolwa
Event starts: N/A

Interview - Tiwaloluwa Ojo
Event starts: N/A

Software Design & Architecture Lec

+

Day Page with all Google Calendar Events



10:00 100%

Calsync

December 2022

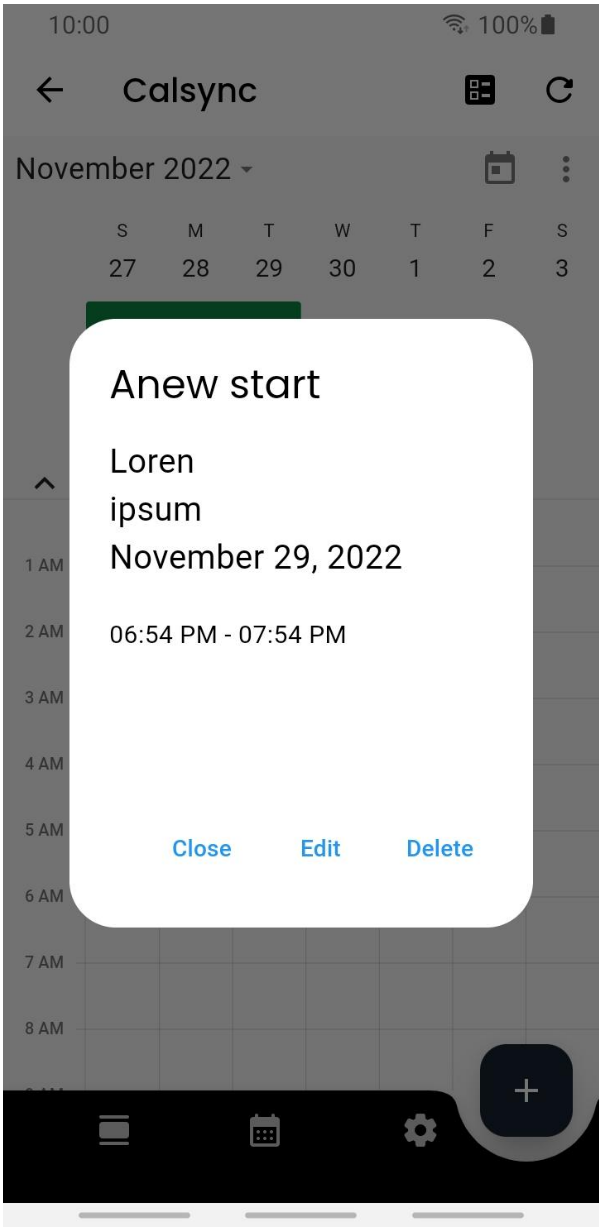
S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

TUE
29

Tg (Day 1 / 2) »
Fgh (Day 1 / 2) »
Anjg (Day 1 / 2) »
After the first week (Day 1 / 2) »
Anew start (Day 1 / 2) »
Fhh
03:10 AM - 05:25 AM

+

Month View With day events below

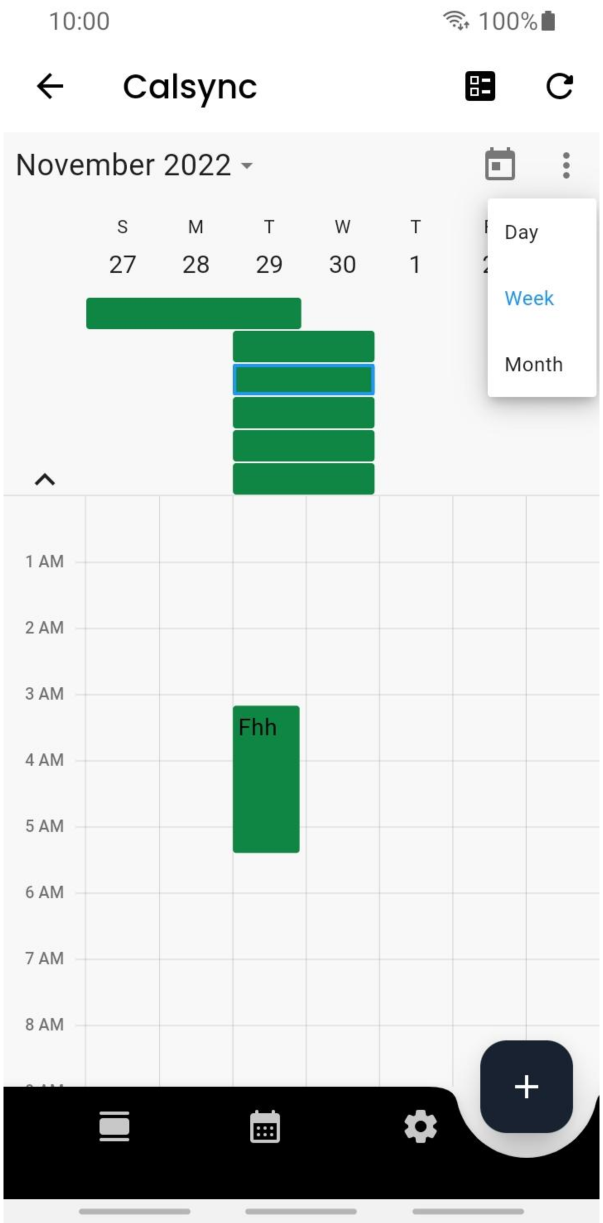


Anew start

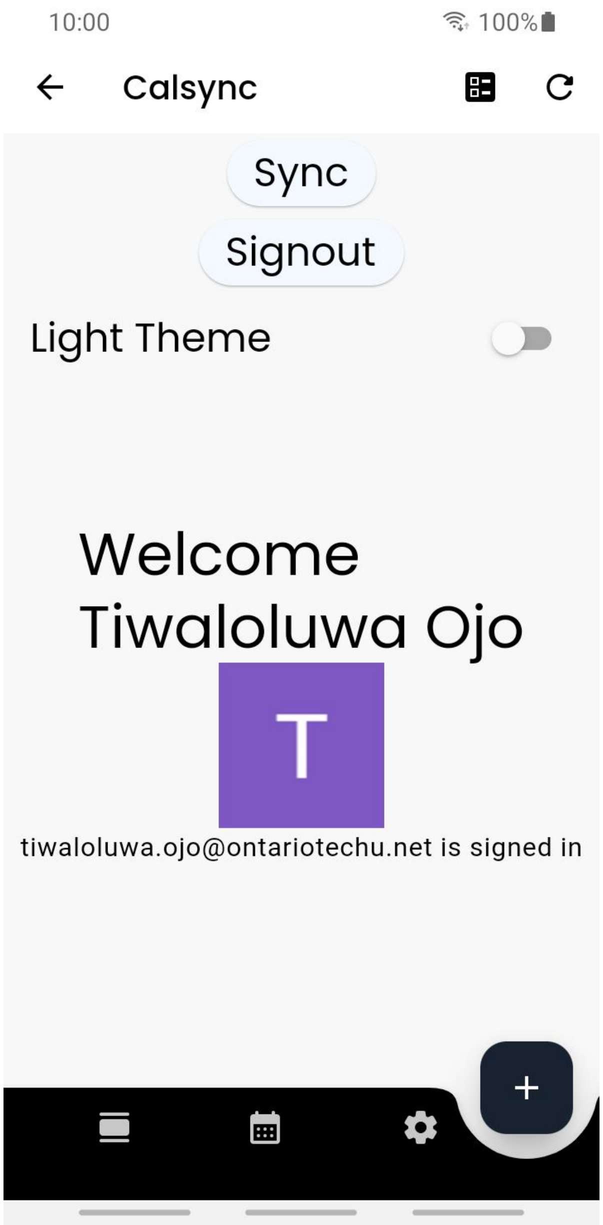
Loren
ipsum
November 29, 2022

06:54 PM - 07:54 PM

Event selected on the calendar day view. Shows Title, Description, start date, and time of event



Calendar Day view in month page

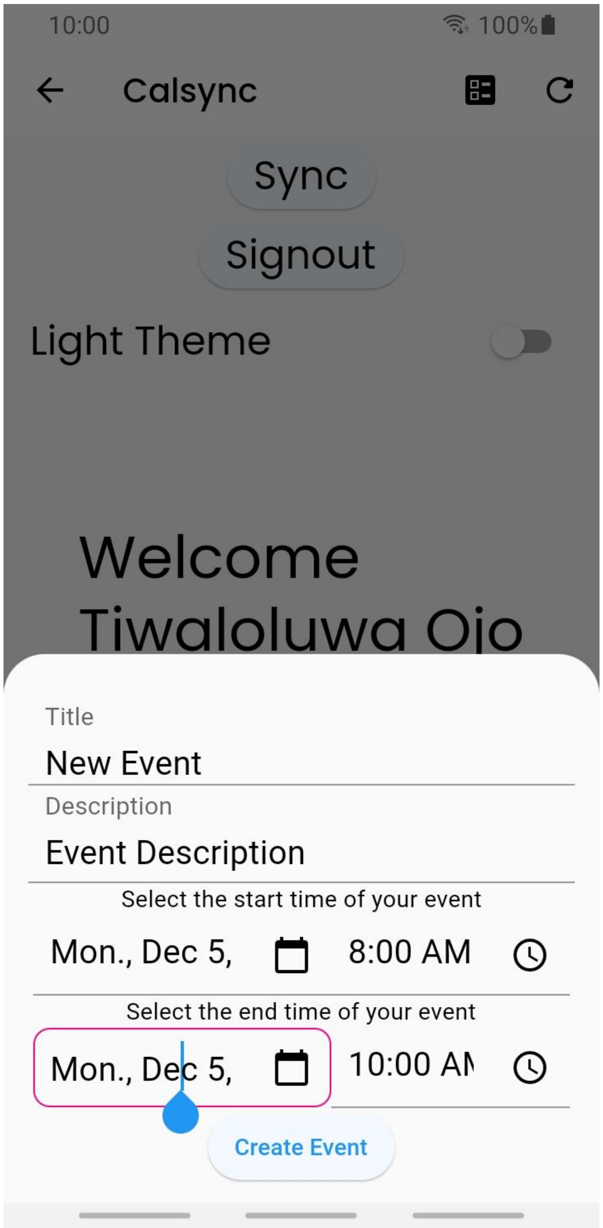


Light Theme

Welcome Tiwaloluwa Ojo

tiwaloluwa.ojo@ontariotechu.net is signed in

Settings page



Light Theme

Welcome Tiwaloluwa Ojo

Title: New Event

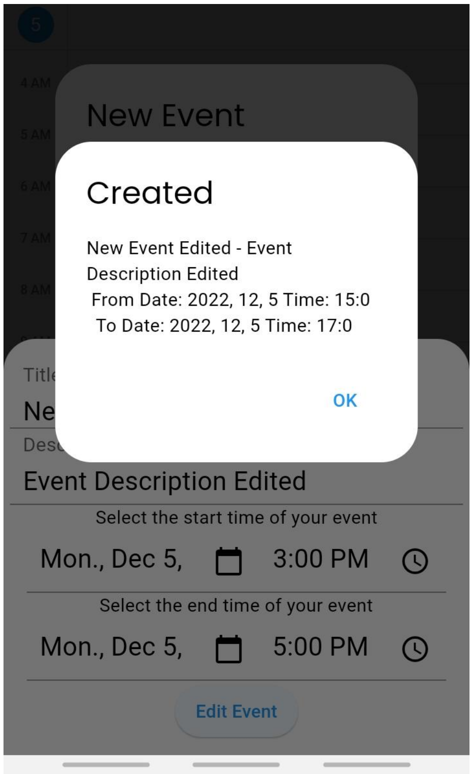
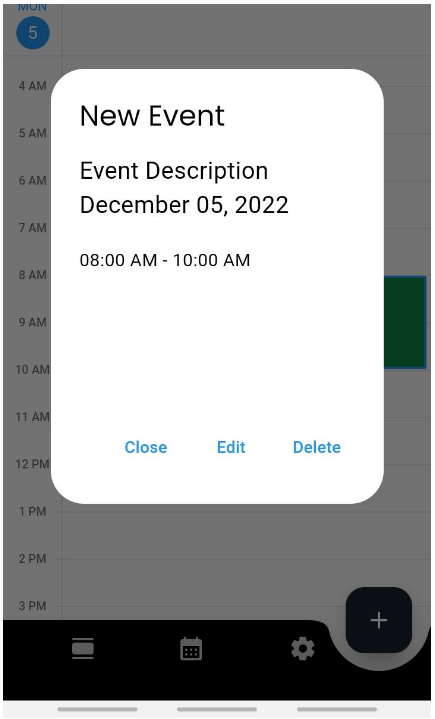
Description: Event Description

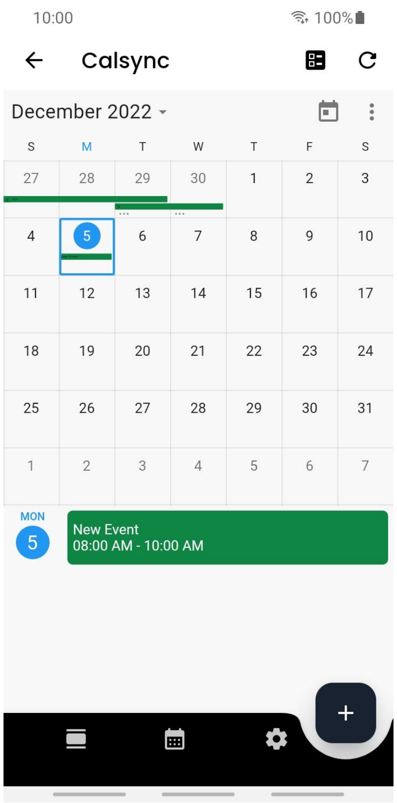
Select the start time of your event

Mon., Dec 5, 8:00 AM

Select the end time of your event

Mon., Dec 5, 10:00 AM

	 New Event Created New Event Edited - Event Description Edited From Date: 2022, 12, 5 Time: 15:0 To Date: 2022, 12, 5 Time: 17:0 Title Ne Desc Event Description Edited Select the start time of your event Mon., Dec 5, 3:00 PM Select the end time of your event Mon., Dec 5, 5:00 PM  New Event Event Description December 05, 2022 08:00 AM - 10:00 AM MON 5 4 AM 5 AM 6 AM 7 AM 8 AM 9 AM 10 AM 11 AM 12 PM 1 PM 2 PM 3 PM Close Edit Delete +



10:00 100%

Calsync

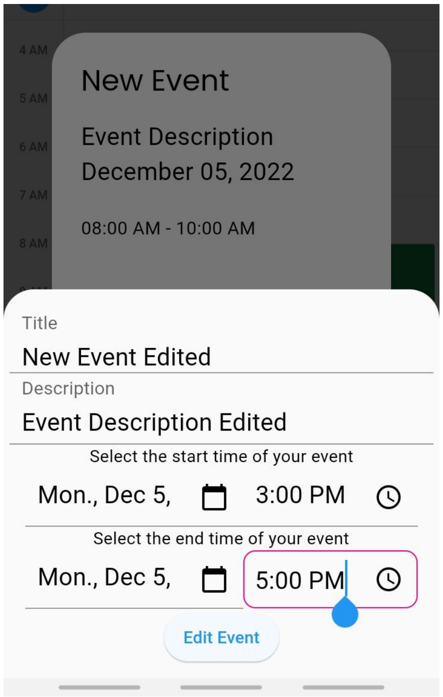
December 2022

S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

MON 5

New Event
08:00 AM - 10:00 AM

Adding New events and displaying them in day view



New Event

Event Description
December 05, 2022

08:00 AM - 10:00 AM

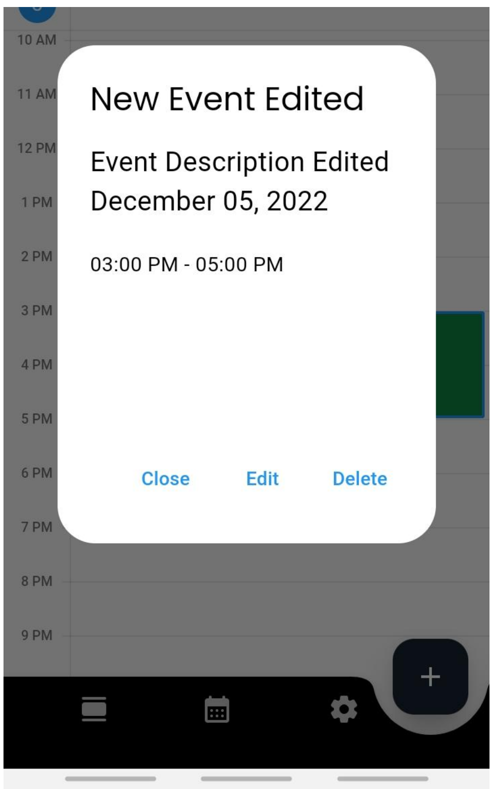
Title
New Event Edited

Description
Event Description Edited

Select the start time of your event
Mon., Dec 5, 3:00 PM

Select the end time of your event
Mon., Dec 5, 5:00 PM

Edit Event



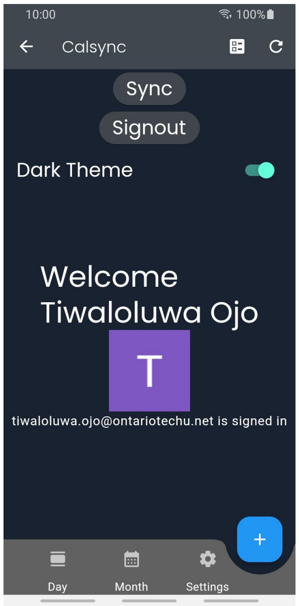
New Event Edited

Event Description Edited
December 05, 2022

03:00 PM - 05:00 PM

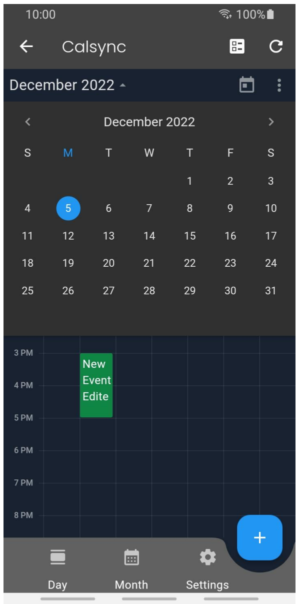
Editing an event

The following images below illustrate the application in dark mode as well as other aspects that do not impact the functionality of the application.

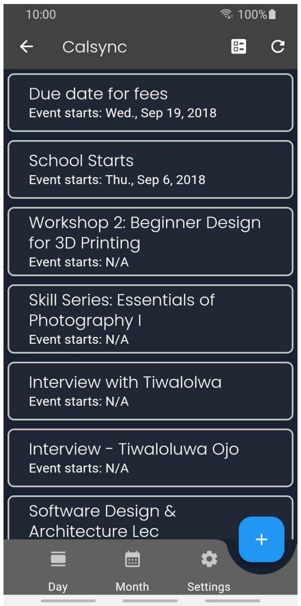


Welcome Tiwaloluwa Ojo

tiwaloluwa.ojo@ontariotechu.net is signed in



S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31



Due date for fees
Event starts: Wed., Sep 19, 2018

School Starts
Event starts: Thu., Sep 6, 2018

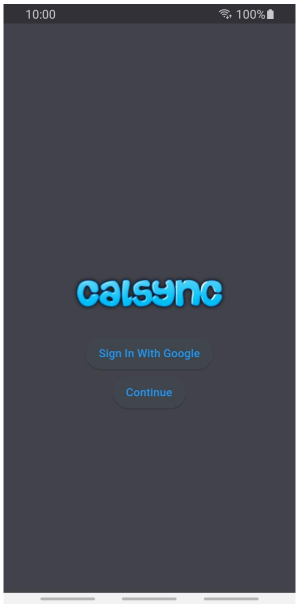
Workshop 2: Beginner Design for 3D Printing
Event starts: N/A

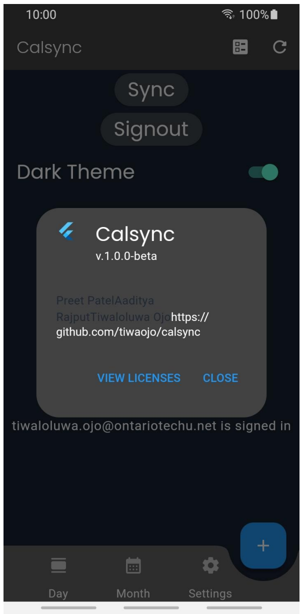
Skill Series: Essentials of Photography I
Event starts: N/A

Interview with Tiwalolwa
Event starts: N/A

Interview - Tiwaloluwa Ojo
Event starts: N/A

Software Design & Architecture Lec

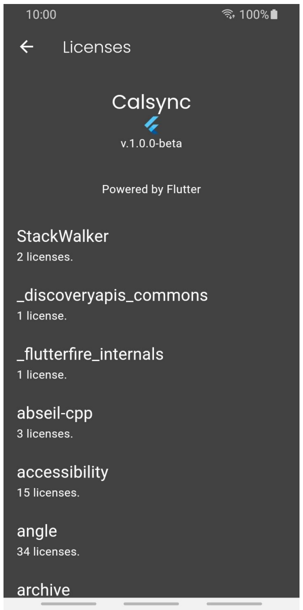




Calsync
v.1.0.0-beta

Preet PatelAaditya RajputTiwaloluwa Ojo https://github.com/tiwaojo/calsync

tiwaloluwa.ojo@ontariotechu.net is signed in



Calsync
v.1.0.0-beta

Powered by Flutter

StackWalker
2 licenses.

_discoveryapis_commons
1 license.

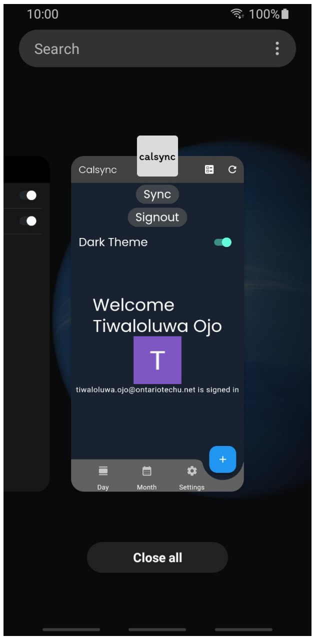
_flutterfire_internals
1 license.

abseil-cpp
3 licenses.

accessibility
15 licenses.

angle
34 licenses.

archive



Welcome Tiwaloluwa Ojo

T

tiwaloluwa.ojo@ontariotechu.net is signed in

Issues and Future Goals

During the progression of our CalSync App, we encountered multiple issues that impacted the completion and some functionalities of the Application. The project was developed on the Flutter framework which required us to read over numerous Flutter documentations in order to get a good grasp and understanding of the technology. This was not an issue or an error per say but rather it was a hurdle of knowledge we had to get over before we could even lay out the foundation of our project.

Once that was resolved we had encountered our first real roadblock in connecting our application to Firebase Firestore and Google Cloud Platform. The problem came in authentication of users who signed into their google accounts on our application. The following is an example of the error we faced regarding this issue: "PlatformException(sign_in_failed,com.google.android.gms.common.api.ApiException: 10: , null, null)" . To resolve this issue we tried numerous tactics such as attempting to fix it using online forums (StackOverFlow) , Dart developers website as well as even recreating the application again ground up. After much investigation we figured out that the problem was between Firebase Project and Google Cloud Platform, as well as our project's fingerprint. We had to link the FireBase Project and the GCP project together so that the “SHA” fingerprint we used wouldn’t conflict and say there is another project using that same SHA ID. This is due to GCP and Firebase won’t allow the same fingerprint on multiple projects.

Lastly we had another issue in the form of using Flutter for IOS. Initially we had assumed that developing in Flutter would allow us to run our application on both Android and IOS however we quickly realized that Apple had requirements that were beyond our reach, such as paying to join their developer program and using their API. Moreover, we had issues configuring the application on XCode as we required it for debugging in their emulators. Unfortunately we could not implement any feature on IOS by the time of our presentation, but we do plan to implement this feature in the event we release our app on the app stores.

The future goals of this project are to include this application to work on IOS as well as implement features like push notifications and incorporate an email invite system for events. Furthermore, we hope to build for other targets such as web and windows desktop, as well as synchronize events between other calendar clients.