Three’s Crowd

Events+ - An events organization and registration app

Project Overview

Our application was developed to enhance event organization and registration. Annual conventions like CppCon, music festivals, and one off events like concerts or local street parties. Currently these events use a variety of different interfaces, with many attempting to use custom ones to track event registration. These vary in quality, with the custom ones often being harder to use and crashing when the event initially opens for registration. This app provides one interface for participants to view all the events they are committed to attending, as well as providing an easy way for local events to organize and get a good estimation of the number of people attending. To run, ensure flutter gen-l10n is run first. This will generate the localisation components required for the application. After this the app can be compiled and run normally with flutter run.

Group Contributions

Group Member	**Contributions**
Ben Armstrong	UML diagramming, time picker implementation, text field formatting, date/time formatting, welcome notification, error handling, bugfixes, assorted report content/formatting.
Caleb Depatie	Date picker, event form, event list, event tiles, initial event view, initial login page, local storage, cloud storage, internationalization, analytics page, snackbars, page navigation, various sections of the report and report formatting.
Yanguang Yang	Maps, Geolocation, Geocoding, Dialogs, Login page UI, navigation to map page, snackbar in map part.

Code Design

We focused on organization and code maintainability within our codebase. Each class is contained within a separate file, the localization files are kept within a separate folder, and we split repeated or larger segments of logic into their own methods. Additionally, there are a few

things we did to improve the performance of the application. A simpler optimization is to ensure as many widgets are compile time constant as possible, so we can move runtime cost to compile-time. We also used the builder static function of the ListView component to dynamically create each event tile rather than generating them all at once. With a larger list, this has increasing benefits as the ListView builder will lazily create the widgets rather than render all of them, including ones that would not be visible to the user.

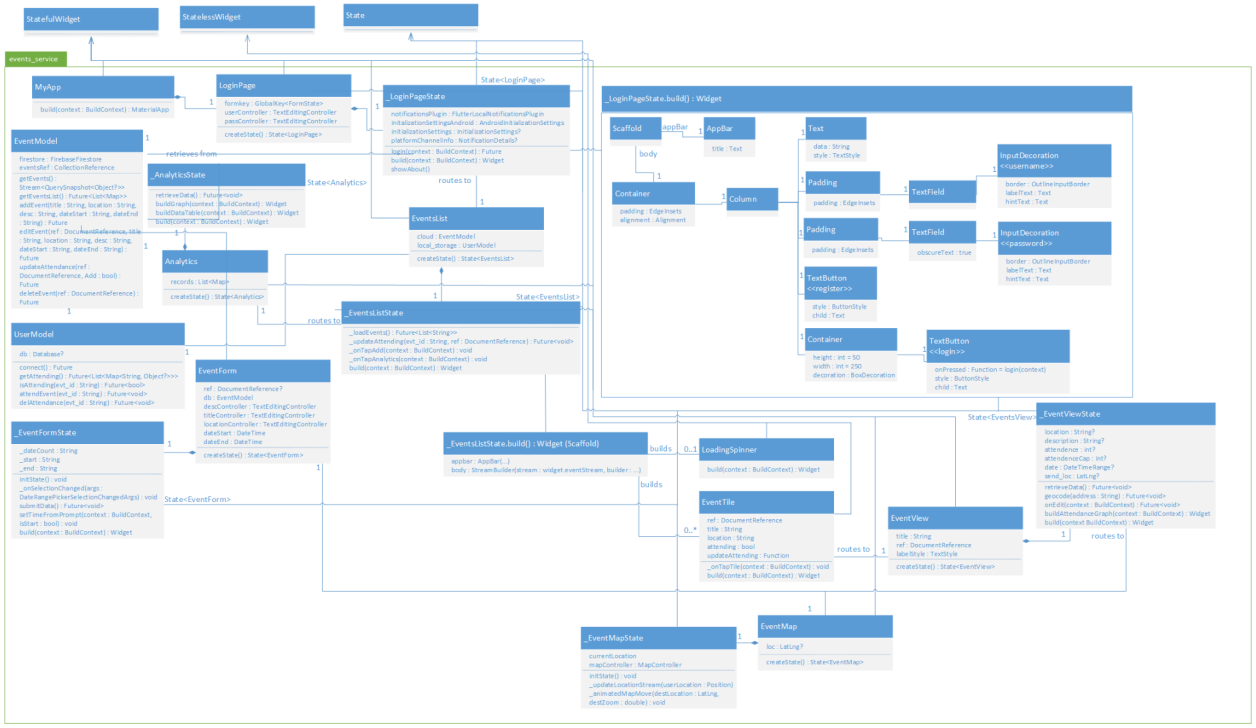

Figure 1. UML diagram representing every class written for the app, in addition to some instances to illustrate implementation and structure.

User's Guide

Upon launching the app, you will be met with a login screen. Currently, the login/account system is unimplemented to make demonstration of the app easier, but in the future, you would ideally be able to click "Register" to create an account, or "Login" with your credentials. For now, you can click "Login" to proceed to the event list.

From the event list, you can use the radio buttons to confirm which events you plan to attend. On the app bar, at the top of the screen, you can press the + button to add a new event, or the bar chart button to view analytics for all of your events. The list of events shows the name of each event, as well as the location - tapping on a tile will bring up the event view.

From the event view, you can easily view the attendance rates, scheduled date and time, location, and description. If the location is a valid address, tapping on the map icon will bring up a map, with a line from your current location to that of the address. Tapping on the pencil icon in the header will bring up a variant of the event creation screen, allowing you to adjust the details of the event.

In the event creation/edit screen, tapping on a given text field will allow you to alter the title, description, and location of the event. The location field will take any text input: you can use this to be more specific, or specify an informal location, but this may render the address location services nonfunctional. To select a single day for an event, simply tap once on the selected day. To set a date range, you can either tap twice - once to select a single date, and then again to pick the other end of the range - or tap and drag to create a range between the beginning and end of the gesture. To pick a time, tap on the time you wish to change, and drag the hand of the clock around its edge until the specified hour and minute appear in the dialogue box. Alternatively, you can tap the keyboard button to enter the times manually. Note that the time picker currently only uses 12-hour time; we intend to localize the time picker to use 24-hour notation where appropriate in future updates. You can press the save button when you are satisfied with the changes. If the app refuses to save, make sure that your start time is before your end time - you wouldn't want your event to be over before it's had time to begin!

Functionality

Navigation

Navigation is used to logically separate the organization of data. The app opens to a login page, which leads users to the primary list based view of the events. A user can then either navigate to a form to add a new event, or by clicking on an event they can see a more detailed view. In the detailed event view, you can go to the same form for adding an event but in this case to edit the current one, or you can open a map that shows the event location. The navigation forms a UI structure that enables users to easily remember where specific options or information for events are.

Dialogs and Pickers

We use a date and time picker within the EventForm, to make selecting the date range and set time of an event much simpler and responsive to touch. We use a dialog on the Login page to present some quick and simple information about the app, including the current version. This version information would be particularly useful for when a user submits a bug report, allowing us to ensure we're able to determine when a bug was introduced.

Snackbars and Notifications

Snackbars are used to provide quick feedback to users in a number of locations. It's primarily used to indicate if an action related to data updating has completed or failed, such as in the EventForm and when updating user attendance from the EventsList page. We have a test notification displayed when users load the EventsList page, but in production we have notifications set to remind users about upcoming events. We used the test one to display the notification functionality during testing as we would have to keep changing event dates to ensure it continued working.

Local Storage

The SQLflite access model uses the local storage for some information related to the user. This is primarily used to store which events a user has set as attending, and is queried from the EventsList page.

Cloud Storage

The cloud storage leverages firebase to store all the information related to individual events, including current attendance numbers. This data is used from most pages including the EventsList, the more detailed EventView, and in the EventForm when editing or creating an event. This cloud storage is integral for synchronizing the event data across multiple users, and ensures all information relating to the event can be edited and updated for users.

Data Tables and Charts

One chart is viewed from the EventView page, and displays the number of current users attending an event versus the set attendance cap. This provides a quick way for an event administrator to make better decisions using the current estimated attendance. From the EventsList page, an Analytics page can be viewed by clicking the chart button in the top right corner. This will display an overview of all the events, showing a bar graph with every event's attendance in the top half of the page, while the bottom half displays a data table showing the exact values of the attendance for each event. This page can help a user who administrates multiple events to quickly review the attendance details for each of them.

Maps

A map is viewed from the EventMap page, which is created by mapbox by FlutterMap function. There are two markers, event's location and users' current location. The current location of users live updates, so users can know their location in real time. And there is a polyline between them which is created by polylineLayerOptions that users can clearly see what direction the event is in. Map center will move to the current location when clicking the floating button in the bottom right corner.

Geolocating

In the EventView page, require the permission of geolocator and check it first, then the current location position is set by Geolocator.getCurrentPosition then sent to the EventMap page. In the EventMap page, the current location and also the marker live update by getPositionStream after transforming the position to a LatLng.

Geocoding

In the EventView page, the locationFromAddress translates a full address of an event into a LatLng. The address has to be workable for locationFromAddress to read, so that it can return a position, otherwise a snackbar will display to warn users to set an available address. If a position is returned, the navigator will push the app to the EventMap page and a map will be viewed, the position will be also sent to the EventMap page to mark this event's location in the map.

Internationalization

Internationalization is applied across the app, and allows for users to choose from English, French, and Spanish as their preferred language. The localisation is generated from language files, using flutter gen-l10n. Using the generated file saved significant time in creating the related classes and determining the users locale. To enable this, unfortunately, many components that were able to be compile time constant required determining language and text during the run time, sacrificing some performance as text descriptors are used on every page of the app.