Note

My app was developed with a physical android device, but the Pixel 3a emulator should be fine.

Introduction

Keeping grades consolidated is not easy when there is not on solution that ticks all the boxes. This leads to tutors using all sorts of different methods for keeping track of attendance, whether that is the convenience of just writing on paper but the downside of having to be digitised later or using Mylo with grade being entered electronically from the start this has the big benefits of consistent formatting and having all the grades in a central location, but with its clunky UI it is not worth for some. So, how do we take grades in a way that is convenient, but also consistent and consolidated? Well, an app of course!

For this app to be a viable option for tutors they need to be able to enter and remove students, quickly and easily mark the students, be able to change what type of mark they are storing in each week and be able to share this data to other applications. So, in this report I will detail my implementation of just such an app on the iOS platform.

Differences from Prototype

Overall, my final product is very close to my prototype, the layout was kept basically the same with mostly Androids' material design principles for the styling.

The biggest difference between my prototype and the finished app was that I added a modal to the landing screen to let the user set the start of the semester for the current week button and label to use. This was honestly just an oversight in the original prototype. Another difference between my prototype and the final application is that in the StudentDetails and WeekDetails classes allow the grades to be edited straight from the respective list instead of having a different widget or modal for it. I did this mainly for efficiency as it means multiple grades can be changed very quickly instead of having to wait for a different widget to load and then having to double back to change another grade. Another big difference is that the week configuration changer has been made much simpler due to a clarification in the spec. In my prototype weeks could have multiple different types of marking schemes in the one week, whereas in the final implementation it was specified that each week only need one type of mark, so I just made it simpler and made it a popover modal.

A slightly smaller changes made was moving all the buttons that were below the tables in the prototype to the navigation bar or a speeddial. I did this as I felt it was a much better use of space and looked a lot cleaner instead of shrinking the list of grades or students and potentially leading to a bit more scrolling. I also moved the image in the student details page to the side of the student's name and such to save a fair bit of vertical space while still keeping the image a decent size.

Usability Goals and Design Principles

My app was built with efficiency in mind. This is best shown in the StudentDetails and WeekDetails classes where all the grades are immediately present and changeable as opposed to having to go to a separate page for each grade so multiple grades can be entered rapidly both for the one student in StudentDetails class or for multiple students in WeekDetails. This does reduce the forgiveness of the app though as once a grade has been changed there is no way to undo this save apart from just re-entering the old grade, and it could be quite easy to accidently enter a grade. This also highlights one of the biggest flaws of my app, the lack of feedback. When updating grades there is no feedback to tell the user that the grade has been successfully updated. I considered solving this by having a popup every time but that would very quickly destroy the efficiency these widgets and just be plain annoying. This could be fixed by using snackbars with an undo button, but if I did that I think I wold make the snackbars togglable somewhere as it might be more annoying to some users to have a snackbar popup every time they change a grade. To get a bit more positive again I think another great display of efficiency is in the go to current week button on the MyHomePage class. This optimises for what is probably a tutors must common use case which is wanting to enter a grade for a student in the current week, so the app keeps track of how many weeks it has been since the start of semester (it does not factor in holidays or mid semester break unfortunately).

I also tried to make the app more visible and add in some mapping (MapInTime, 2013) with the overall style of the app. All the of buttons in the app are clearly label and can be seen straight away without cluttering up the UI. I also used default icons built into flutter for all navigatiojn bar functions, floating action buttons/apeeddials and to some of the buttons to try to keep the app consistent with other flutter apps using the material library to hopefully line up with most users' mental models of how flutter apps should look and work, though this is probably not the best idea with flutter being cross platform and users not necessarily knowing the app was made in flutter. The StudentsWeeksLists class holding both the StudentsList and WeeksList give the users options that are more efficient for certain used cases. well as make the user aware about the two separate browsing methods. It is defaulted to the WeeksList as I was informed by Lindsay that that is the tab that would be used the most out of the two and I can see why as most tutors won't be putting in marks retroactively and instead will be entering the marks in the app during the tutorial so they can just leave the app on this screen for the tutorial.

I added constraints to the app in the form of the tables in StudentDetails and WeekDetails only display the type of grade that is configured for that week, and even more so for the score weeks as they make sure the user can only enter positive numbers and makes sure the number is less than or equal to the max score, changing a grade that is too big to the maximum score. Similar constraints are also present in showAddStudentDialog and when changing a student's details in the StudentDetails where the student ID has to be a 6-digit number and is also checked to make sure it is unique with clear feedback if one of these criteria are not hit.

I also added some forgiveness to swiping to delete a user on the StudentsList in the form of a popup that makes sure the user is certain they want to delete this student by displaying the name for the student to make sure they are deleting the right student if they even meant to delete a student. I also made the confirmation buttons the opposite colours to expected (green for no and red for yes) to play into a user mental model of green usually being for yes and red for no to trying and even further stop users from accidently deleting students. Similarly, the changeSemesterStart, showWeekConfigDialog and showAddStudentDialog can be dismissed by pressing the back button or simply tapping outside the dialog, in line with how Androids' modals and iOS's alerts work.

As previously mentioned, I think the StudentDetails and WeekDetails highlight one to the biggest flaws in my app being feedback, I think these two views also suffer badly from a lack of forgiveness when changing grades. On top of not giving feedback that a grade has changed, the user has no way to revert a grade change aside from just updating the grade again back to what it was before. This was another one of the trade-offs I made for efficiency that I feel is a little more justified than the lack of feedback, but at the same time could cause problems for users. Another potential problem I can foresee is users potentially not noticing features, the most obvious one in my eyes is not knowing you can swipe to delete students. To fix this I would probably implement a first-time launch tutorial to guide the user through how to use the app.

Code Documentation

WeekConfigs – A model fetch and update students grades from the database.

Student - A structure and model to hold a student's details; first name, last name, student ID, name of their image if they have one and their grades, as well as update all the students' details.

list_tiles – This folder contains all the tiles used in the lists; Attendance, Checkpoints, GradeAToF, GradeHDToNN, Score.

MyApp – This class initalises Firebase and and calls the first class.

MyHomePage – This class displays the current week and takes the user to either the current week or StudentsWeeksLists

changeSemesterStart – This dialog widget allows the user to change what date the semester starts on, which in turn changes which week the MyHomePage's current week button will take you to.

StudentsWeeksLists – This class is a tab controller that display the StudentList and WeekList

StudentList – This class displays a list of students and a speeddial to sort and add students.

WeekList – This class displays a list of the weeks.

showAddStudentDialog – This dialog pops up from a speeddial in the StudentList class and allows the user to add a student with or without an image.

StudentDetails – This class displays and allows the user to change a student's details and grades. This class also allows the user to update grades, load the students images, display alerts and update details.

WeekDetails – This class displays the grade for every student in the selected week.

showWeekConfigDialog – This dialog pops up from the WeekDetails class and lets you change the grade type for the week it was opened from.

utility_functions – This file stores some functions that are used across the app like the share functions and some custom versions of built in widgets.

Nearly all of my code was sourced from the lectures, tutorial content or Flutters own documentation.

Packages used were:

"share_plus" - , fluttercommunity.dev (2021), https://pub.dev/packages/share_plus I used this package and the examples on the linked page to make my three share functions in the utility_functions file.

"FAB – Speed Dial in Flutter" – varshaadhanasekar (2020), https://www.geeksforgeeks.org/fab-speed-dial-in-flutter/ I used this awesome tutorial and package to make the speeddial in the StudentsList class, the code was just a lightly modified version of the example given.

"image_picker" – flutter.dev (2021), https://pub.dev/packages/image_picker I used this package and the examples given to make my image and camera code in showAddStudentDialog to allow the user to user their system camera or image gallery to upload an image.

Some other notable example of third-party code is:

"Flutter: Communication between widgets" - Diego Velasquez (2019), https://medium.com/flutter-community/flutter-communication-between-widgets-f5590230df1e I used this tutorial to pass a callback function to all the list_tile classes so they could update the grade average in their parent views.

"MultiProvider micro lesson" - Eric Windmill (2020), https://flutterbyexample.com/lesson/multi-provider-micro-lesson , used this tutorial to add a multiprovider so I could have separate models for students and the week configurations

Conclusion

To round things out, I think I have taken all the good aspects of my prototype and previous assignments and refined them into a feature rich and efficient cross platform flutter app. Like my prototype, my flutter app was an efficiency first design, making sure the most common use cases can be done with as little clicks as possible, while still staying intuitive and enjoyable to use. While I am happy with the outcome, I was once again reminded how much of a balancing act it can be to try and make the app as easy to understand as possible while still being quick and minimal. There were no missed features this time round and thanks to the discord for doing some user testing for me and finding some bugs, I am fairly confident that my app is stable and reliable in most if not all case. This app really does feel like a cumulation of all the skill learnt so far in this unit and has left me with an app I am very proud of.

References

1. MapInTime. (2013). Don Norman's Design Principles. [online] Available at: https://williamgrimes12.wordpress.com/2013/01/22/don-normans-design-principles/ [Accessed 12 June. 2021].