Flow Manager

Ahmed Ali
Robert Paud
Ernest Rising
Adam Salinggih

CS 481

Summary

The Flow Manager project is the unique approach to a Todo List application, such that it helps visualize the data of the workloads required for tasks/projects, in doing so, this transition from a simple Todo list becomes more of a project management application. This paper will provide a thorough discussion of the various aspects of the Flow Manager project, such as various obstacles the team overcame to complete the project. Such obstacles include the software engineering design that propagated from, lack of libraries developed for this specific application. Also, the various points of the development life cycle, that include, the initial proposal of the project, the design aspect of the project, the implementation of the project, and the final release features. Thirdly, a brief description/list of features, such as, the main function of the Gantt Chart, with a couple of interactive futures with it, followed by a Google Calendar interaction with the Flow Manager App. And lastly, limitations the project had to face due to time constraint and the limitations of the required concepts that had to be implemented to make this application possible. By the end of this discussion, there will be a complete overview, background, history, release, and discussion of future updates discussed within this essay about the project, Flow Manager.

Introduction

This paper will be a discussion of the Flow Manager app project that was designed to implement a Gantt Chart to help visualize meaningful data, such as the number hours needed to work, for various tasks around a given period. This paper will go over the app description, and the various features that have been implemented, a brief discussion of the challenges, merits, and the innovations the project offers over similar apps and ideas, the limitations of the work period and various aspects of the project that requires more time to implement, and a brief overview of the topics covered by the project and how they were worked on.

App Description

This section of the paper will be a brief explanation of the app's selling points, a few of its features, and how the app stands apart from its similar competitors within the same market.

Flow Manager is an application meant to help visualize the workload over several weeks by providing an easy to interpret bar graph that clearly illustrates the amount of work to be done. This is done through the clever use of the app's bar graph by displaying the day of the week, through the use of one being Sunday and seven being Sunday, along the x-axis, the number of hours pertaining to that day, along the y-axis, and the what week its associated with by the color of the bar itself. The application also offers easy navigation to, and from, the user's Google Calendar to allow easy synchronization of the tasks inputted into the app and the user's daily calendar. Lastly, the user can easily add and delete workload tasks through the app's navigation of the add and delete windows.

Flow Manager's scope and projection is quite narrow, but even so, we have been able to develop several features that help build this app's portfolio of functionality. Besides the main feature of overseeing your tasks for the week, this app also has direct, and easy, access to the Google Calendar where the user can manually synchronize the events with what is in the Gantt Chart. Besides this, the app is also interfaced with a simple notification system that will remind the user to check the app's overview screen to see how the week is looking. These are simply the starting points of this app, and the app still can use more development time and engineering to really set up a fully functioning application, but as it stands, it is a unique approach to handling the week-to-week work of a project manager.

Discussion

In this section, there will be a thorough discussion on the major features of this application, on a much more technical level, by having a deeper understanding of what the application does, how its achieved, and lastly how it can be expanded upon in the future.

To start with, as described in the App Description section, we will talk about the major features of the app, such as, the data visualization, the Google Calendar interface, and the notifications. Data visualization is the core foundational component of the app, in which, we use a view Model between the main applications, and Firebase, to ensure a fast and secure method of storing/reading data/tasks of the user. The main component used to make this happen is the

Halfhp Simple XY Android plot, please refer to reference [1] for further details about this, as this was used to construct the bar graphs that are seen within the main app window. This is done through the library's clever use of number typed arrays that allow for unique generation of bar graphs through their bar formatter tool. This bar formatter instance allows for great customization and organization of the data, such as, complete customization of bar organization, whether that be orientation, grouping, width, or colors, and complete control of the exact scaling and placement of the bars on both the X and Y axis. This library also gives control over labeling the various axes, titling the graph, and auto generates a legend of the graph based on the naming conventions of the bar formatting instances. With this, the team was able to implement a Gantt chart within the constraints of a bar chart with the labeling associating the Y-axis with the number of hours in a day and the X-axis being associated with the day of the week, such that, one represents Sunday and seven represents the following Saturday. With this, the app also offers Google Calendar implementation so that the user can quickly transverse and synchronize their task information with their personal calendar, which also interlays with another minor feature of the app which is the in-app notifications. The in-app notifications are simple reminders that can be given permissions at any point by pressing the notification button on the splash screen of the page to initialize. Whether the user accepts permissions will not affect usability of the app in any way, because the in-app notifications are meant to remind the user to return to Flow Manager to check on their progress.

Secondly, we will discuss some of the background features that run the app such as the Firebase integration, some reasoning on why we chose to choose fragments with the app navigation, and lastly some of the background operations for expandability. The use of Firebase, in conjunction with a view Model class, is to allow data to be loaded in from the API and stored locally while the main activity is still alive, thus reducing the number of resources required for switching between the various functions of the app. The Firebase is also used to store the information so that upon launch of the application, the user is not required to reapply all the tasks that were stored into the app upon the previous session, making it more user friendly and allowing for reusability for the future tasks.

Lastly, we will have a brief discussion about the expandability of the app and what sort of features future updates could bring, such as more visualization of the data, and the integration of user contacts. Through some brief research, the visualization of the data could be offered in a myriad of ways that could be added through development of libraries and or more extensive use of the Simple Android XY Plot used for the bar graph. The plot library offers several options for statistical analysis, such as, the use of pie charts, line graphs, candle graphs, live Data upload graphs, and scatter plots. With just these alone, Flow Manager's features could be expanded upon in future updates to use more of these functions, for example, instead of just associating colored graphs to weeks/time periods, with the integration of the user's contacts, various lines, wedges, or bars could be associated with people the user added. While the current Flow Manager is much more in the prototypical phase of development, it still stands as steppingstones for going further in the future.

Limitations

In the Limitations section, there will be a thorough discussion about the constraints the Flow Manager project faced such as time constraints within the class, the known issues within the app, unfinished tasks that require more research and development than the allotted time in class and forced changes the project had to go through for some version of the app to be released.

To start, through development of the app, there has only been one notable bug that caused us to keep one feature omitted from the release version, which was proper integration of in-app notifications. Unbeknownst to the developers on the team, through following the book and lecture material, certain keywords of the notification build methods could not handle/allow the use of certain keywords. And as such, even though technical googling, the app's notification system was unfortunately not able to be fully developed within the project's life cycle.

This project unfortunately had to be released with one known unfinished task, which was a more proper version of a Gantt chart. As mentioned earlier, in the Discussion section of the essay, the Gantt chart was not fully realized as it should have been at the time of the proposal. While the bar graph functions as intended, and provides the user with the necessary information, in the light of the Flow Manager's original scope and hope, this task will be as a technical unfinished task.

The project was forced to undertake several known changes after the proposal, such as handling multiple contacts/dynamically allocated arrays of information of people, proper integration of the contacts list into the app, and a scale back of the initial design of the Gantt chart's main feature. As previously mentioned in the above sections and parts of the Limitations section, several of the application's features, UI, and integration plans were unfortunately scaled back due to not only the scope of the class but also the lack of documention and or accessible libraries for Kotlin. In the future, as Kotlin becomes more widely used and developed, more libraries will be developed that can help contain the scope of the original project's intentions.

List of Implemented Concepts in Project

Group Member	Tasks
Adam	Firebase, API
Ahmed	UI, Navigation, Fragments, Material Design
Ernest	Gantt Chart Implementation, partial UI, partial Firebase Database, MVVM, Repository Pattern, View Model/Live Data, paper author.
Rob	Notifications, Google Calendar Implementation, Google Calendar API, Permissions, Notifications

Implemented Tasks

- API

- Fragments
- Firebase
- Notifications
- Permissions
- Navigation
- Material Design
- MVVM
- Live Data
- Repository

The API in this application is a combination of the Firebase and google calendar API, the strengths of using these both is to allow for an easily portable/cloud-based storage system, with internet connection, that will follow the user with their associated account.

Fragments were used for fast navigation between the three main windows of the application, which were the main application/task overview, the add task page, and the delete task page. The reason we decided to use fragments was to ensure a responsive feel to the app, and efficient resource management, given the amount of background processing that can occur when adding/deleting a task. The reason for the immense amount of background processes is the fact that the app not only updates the firebase information/task, of what tasks are currently in the system, but also updates the current view Model of the arrays that are in use that power the main data visualization that is shared between the fragments because of the activity.

Firebase was the core database project used for developing this app, for two core reasons. The first reason was so that we could use the API functionality of the app to ensure a much more secure way of saving our data within the system, while also having an ease of use/fluency amongst all developers on the team since Firebase was much more familiar to many of us. The supporting reason for using Firebase was because of the fluency that all members of the team had strong developer skills that helped in the debugging process when it came to the software engineering side of the development.

Notifications were an optional feature that allowed our user the ability to engage with this function, by accepting the required permissions, if they wished to have reminders about checking the tasks on the application. The reason notifications were so valuable for the development of the project was because of how well it meshed with our use of permissions since the workflow of permissions reinforce the idea of making the app still highly functional even if the user denies the permission requests.

Navigation for our app was designed for ease of use with one hand with a bottom navigation menu, with appealing graphics that make our app more comfortable and soothing to the eyes. Also, with the use of a bottom navigation, we can ensure a snappy and comfortable navigation between our three main fragments of the app, that also allow for expansion of ideas/features with simply adding more navigation options below.

Material design for the project was meant for functionality first, which is apparent with how rigid and technical some of the windows of the application are. Although, the development stages have allowed us to experiment with some nice graphic designs as a theme to extend into with the app.

The Model-View-ViewModel (MVVM) was used for code organization and functionality for the UI and easy navigation/documentation of some of the interlaced aspects of the code with the various team members. For example, by implementing and using the MVVM, this allowed the database developer work freely and conflicting with the UI developer, which allowed easy expansion for the Google Calendar API developer which made the project organizer's position much easier and efficient. Overall, using this pattern became a great tool for a more complete organization of all the various concepts and ideas this app was working with.

The repository architecture became a core facet of designing the code organization when the various developers had begun to push their code. Since the application ran three distinct sources of information, two different databases/pages within the Firebase database and a third, redundant, local information stored from within the view Model. So, with the repository pattern, this allowed us to virtualize, abstract, and organize our sources of data manipulation from various tasks/calls/variables to one convenient source that became consistent throughout the entire application, thus, made navigating code much more efficient for debugging purposes.

The above-mentioned concepts were implemented as key choices to allow our application to function, given the project idea and proposal originally made. While some aspects had to be changed and various ideas had to be adjusted, the ten core concepts mentioned above did allow us to meld them together into a cohesive idea.

Conclusion

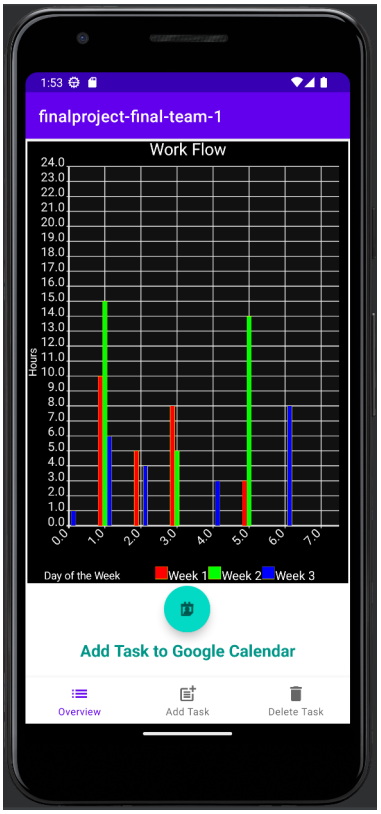
In conclusion, the Flow Manager project brought together many ideas, concepts, and tough decisions when it came to its development. This essay was a thorough discussion of what the application's initial intentions were, the development lifecycle of those intentions, and where the application's realistic expectation came to materialize in the final product. Given the unique constraints of the project, such as the lack of frontloaded knowledge, initial expectations vs realistic standards, time, and the required ten out of fourteen concepts, the project was indeed a massive success of software ingenuity and perseverance through the many obstacles this project had presented throughout the semester.

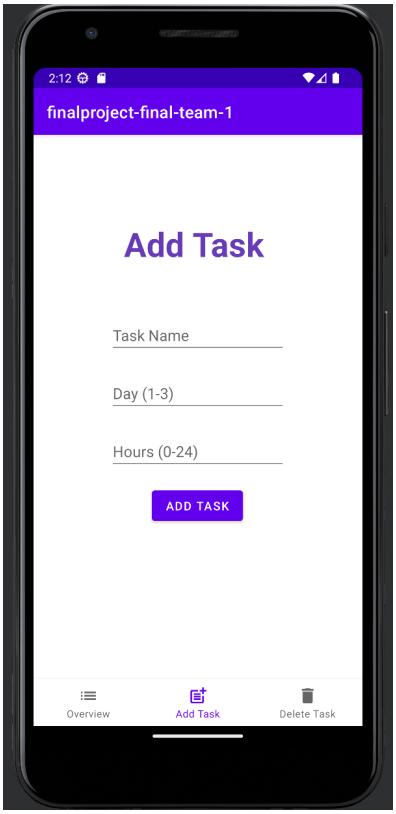
References

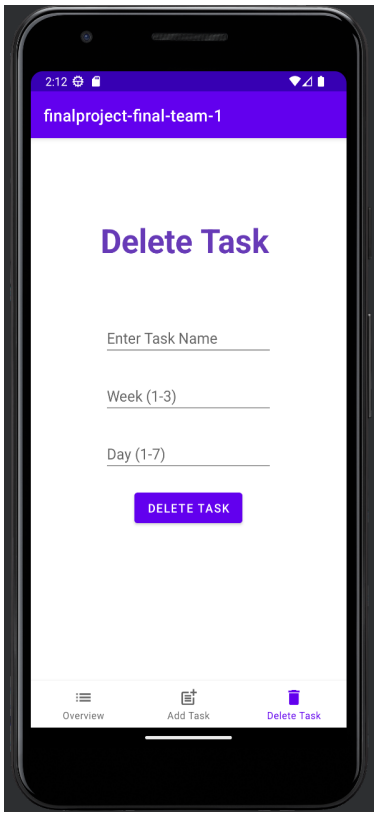
[1] Halfhp, "Androidplot/index.md at master · HALFHP/androidplot," *GitHub*. [Online]. Available: https://github.com/halfhp/androidplot/blob/master/docs/index.md.

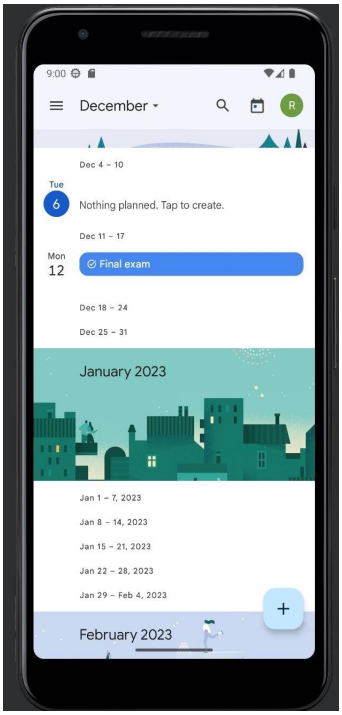
Appendix

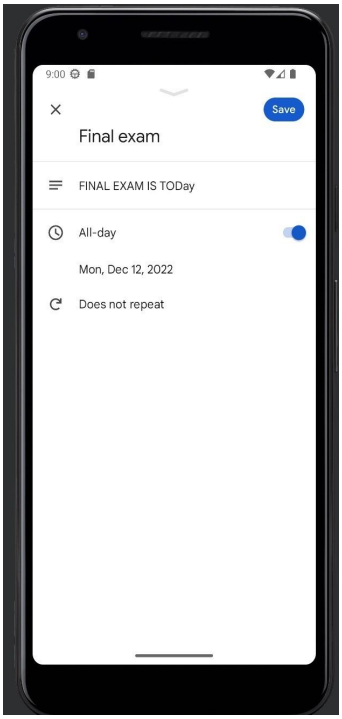

1:53
finalproject-final-team-1
WorkFlow
johndoe@email.com
.....
LOGIN
NOTIFICATION

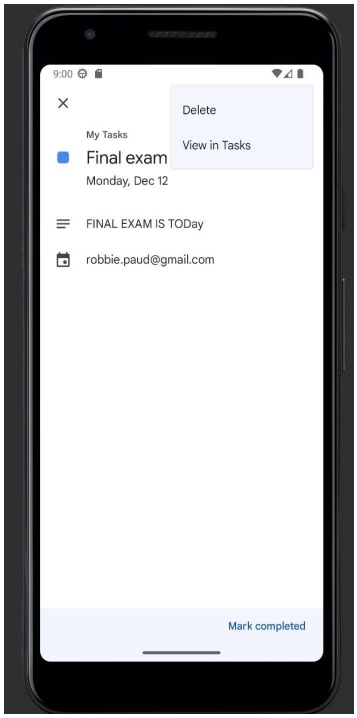

1:53
finalproject-final-team-1
Work Flow
Hours
Day of the Week
Week 1
Week 2
Week 3
Add Task to Google Calendar
Overview
Add Task
Delete Task

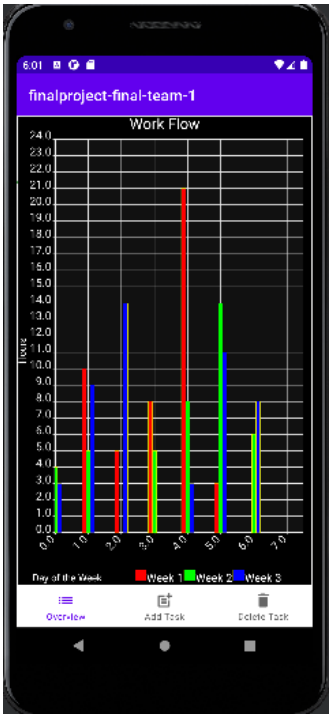

2:12
finalproject-final-team-1
Add Task
Task Name
Day (1-3)
Hours (0-24)
ADD TASK
Overview
Add Task
Delete Task


2:12
finalproject-final-team-1
Delete Task
Enter Task Name
Week (1-3)
Day (1-7)
DELETE TASK
Overview
Add Task
Delete Task


9:00
December
Dec 4 – 10
Tue
6
Nothing planned. Tap to create.
Dec 11 – 17
Mon
12
Final exam
Dec 18 – 24
Dec 25 – 31
January 2023
Jan 1 – 7, 2023
Jan 8 – 14, 2023
Jan 15 – 21, 2023
Jan 22 – 28, 2023
Jan 29 – Feb 4, 2023
February 2023


9:00
Save
Final exam
FINAL EXAM IS TODay
All-day
Mon, Dec 12, 2022
Does not repeat


9:00
Delete
View in Tasks
My Tasks
Final exam
Monday, Dec 12
FINAL EXAM IS TODay
robbie.paud@gmail.com
Mark completed


6:01
finalproject-final-team-1
Work Flow
Tasks
Week 1
Week 2
Week 3
Overview
Add Task
Delete Task