Group: ChadGPT
Professor: Dr. Muhammad Lutfor Rahman
Course: CS 481
Date: 12 May 2023
Group Members:
Sean Perry
Bradley Tran
Jamison Coombs
Trevor Kvanvig
Edward Uriarte

Synapflow Final Project Documentation

Abstract

The purpose of this project is to create a simple productivity timer and event planner app. To study via the Pomodoro Technique, studying for 25 minutes, taking a break for 5. While also creating events to keep track of and to be alerted on to make sure that you are staying on track. We believe there is a benefit for an app like this one, removing extra distractions and creating exactly what is needed to keep you motivated and on track for success.

Introduction

SynapFlow is an app that is built to be a simple way of keeping track of events like exams, graduation, or scheduling time to sit down and study a certain subject. Giving you an interactive calendar to view your tasks to make sure you can plan ahead. The app is meant to help you stay productive and on top of your schedule, while also giving you a tool to help you sit down and study for a period of time, while giving you breaks.

The app also rewards you with trophies to help feed a sense of accomplishment for each completed study session. The app also gives you the capability of viewing some analytics for your last week and last month's amount of study sessions, in the form of a line chart for the last week, and a bar chart for the last month, binning each group as a week in the last four.

Description of App

SynapFlow is a time management app that allows you to track your study times and keep track of important events like exams, birthdays, or meetings. Features include:

- **Timer:** Primarily structured around the Pomodoro Method (25 minutes of focused work, 5 minute break intervals). Users have the option to pause, resume, or completely stop a timed session.

- **Analytics:** Tracks users' study sessions progress by tracking completed number of study sessions over the last few days, and understanding your study sessions over the last month which are visually displayed through user friendly chart graphics.
- **Tasks:** Creating event tasks with short descriptions and date/timeframes to be alerted when you need to be working on something.
- **Rewards:** Users can acquire achievement badges that serve as reminders of progress and accomplishments which instill a sense of motivation to study consistently and log more sessions for greater milestones.
- **Calendar:** Clean user friendly calendar which serves as a hub to display all previous and newly created tasks which can be marked as complete or deleted. Dates within the calendar containing any tasks are represented by a blue dot visual so that users can identify which days contain tasks to be completed.

Discussion

Challenges

Notifications and AlarmManager: Getting notifications to work with the Android AlarmManager class proved to be a challenge as the way to do this has changed many times over the development of Android and its many different API versions. Getting notifications themselves to show up was not a problem or a challenge but I had difficulty getting the AlarmManager and BroadcastReceiver to work properly. It ended up being all due to a single line of code that was using the wrong function call. Originally I was trying to get it to work with the AlarmManger.SetExact() function, which was the function that I most commonly saw in the documentation to do this sort of thing. I then eventually stumbled across the setExactAndAllowWhileIdle() function which solved all of my problems after trying to get it to work for a couple of hours. It seems to always be the small things that cause all of the problems.

Firestore Implementation: Firestore's implementation was a struggle to grasp. The concept of adding, deleting, updating properly was a bit complicated to work with at first. The first hurdle we encountered was properly getting the data out of Firestore and making sure each fragment properly had access to them. We eventually landed on a global *fsdb* variable that could store lists inside of the class implementation, so that as things processed it could request firestore to update the lists and maps. The next issue we encountered was properly updating the data, and storing it. We eventually found out that you could store new collections inside documents, and that was the game changer we needed to make our implementation work.

Innovation

The innovation of our app is the simplicity in how it brings together useful features that a user would need to effectively study, track progress through logged sessions, as well as store important tasks which can alert the user through notifications based on priority and chosen target date/times. All listed features of our comprehensive study tool play a crucial role in keeping the

user on track through the promotion of productivity, motivation, and organization. Although this is a powerful tool to enhance a user's study experience or desired learning goals, the overall design was developed with simplicity in mind so that anyone can easily pick up and use our application without facing any technological barriers.

Merit

- **Interactive:** The app engages you in a variety of activities, such as studying, analytics, and tracking tasks. Analytics serve as an effective visual tool that user's can understand by seeing the progression of logged study sessions over varying periods of time. Providing users the ability to create tasks which are automatically allocated to the calendar allows greater flexibility to easily set and revisit. To ensure users do not miss or forget to manage a task (complete/delete), our app provides automatic notifications to act as helpful reminders to the user.

- **Simple:** There isn't a lot of interface to bog down the application, so that way it's lightweight in resource utilization as well as being straight to the point of the user's experience. Our vision was to create a painless to use study app that still managed to be robust in providing the user with functional features that would improve their overall productivity and organization.

- **Useful:** The app makes studying enjoyable and motivates users to be consistent with the rewards system in place as well as review their progress through the analytics graphics. Provides users a timer that is designed to operate based on the most effective studying technique known (Pomodoro) to provide a structured study session. Task creation is simple to use and is automatically stored and organized through the calendar making it easy to manage tasks and deadlines effectively.

<u>List of Implemented Concept</u>

1.Navigation

- The navigation bar exists inside our home activity, where it also displays the current fragment. The bar allows you to access the home timer, the analytics page, task create page, trophies, and the calendar.

2.Fragments

- Near the entirety of our application is ran inside of fragments. The only page that is not, is the main login page. We utilized Fragments for the timer, analytics, task, trophy, and calendar pages. Both plots as well are fragments inside of the analytics fragment.

3.Firebase

- ➢ Firebase was the core of our project's information. The first portion we implemented was the Google Authentication for the users to login with. This decision was so that we could easily deal with accounts and login systems and not have to worry about passwords and lost accounts. We then used the token UID generated as the users ID for the firestore database.

- ➢ In Firestore we created a database, using the UID as keys for each user, we then saved some general information, like a count of how many completed study sessions the user has completed, a long with a map of dates, with a string of the time they completed a task, so that this information could be later accessed. We didn't fully utilize the times finished, we could have instead used a map with an int counter, but this was a part of something we attempted with the analytics.

4.API

- ➢ AndroidChart API: We discovered Android chart as an alternative to AndroidPlot. The formatting, code setup, and design was significantly cleaner and we decided we wanted the format of AndroidChart for our design.

- ➢ Material Progress Bar: The progress bar 'wheel' was something we found while researching how to do the timer, via the walkthrough I began with. The wheel for the timer counts down/gradually depletes as the time counts down and ultimately disappears on completion. Functional visual element that provides users a quick and easy way to monitor the time remaining during a study session.

- ➢ Calendar API: The calendar API was a github repository we found that allowed us to integrate date information to display for the user. It also was very simple for us to add tasks to it due to it being simple enough to take in a date as a key, so that we could display the information in the recycler view.

5. Notification

- ➢ AddTaskFragment: After a task is created in the add task fragment we create a pending intent to the NotificationBroadcastReciever (see below) with the task details. We then used the Android Alarm Manager class to set an alarm at the date and time set in the fragment in the OS to trigger that broadcast to the NotificationBroadcastReciever.

- ➢ NotificationBroadcastReciever: When this class receives a broadcast from the OS it builds a new notification with our channel and task details. Once that is done it will then send the notification to the user that they have a task to complete.

6. Recycler view

- Recycler view was used as a way to display tasks inside the calendar. Inside the calendar at the bottom, we designed it so that it displays the name, description time and if the task is completed. That way the user can keep track of the information the had described the task with. Along with being able to complete the task and delete the task from that window as well.

7. Material Design

- Material Design was a trickier implementation for us. We implemented our own image in the home screen of our app. We also utilized trophy images, along with icons for the navigation bar. We also changed the default color and appearance of the app, attempting to make it look more completed.

8. Permission

- Permissions were implemented to allow for both notifications and we also implemented it to access the users location. The notifications need the permissions to be able to run properly. While implementing the extra logic for the location was to make sure we could do as much as we can with the functionality.

9. LiveData

- LiveData is used to store all of the variable information for the timer data. We utilized LiveData so that we can store the timer and state of the project without fear of losing the data while we access other portions of the app. Once you return the timer fragment, the app will reinitialize the timer and either continue it, continue to have it paused, or finish the timer and run the code related to the timer finishing.

10. Unit Testing

- Plot Unit Testing: For the plots, we created some unit tests to make sure the lines and plots were created as expected based on the map that is stored inside the function.

- Task Unit Testing: For the tasks, we created some tests to make sure that each of the variables within the Task object are updated and stored correctly.

- Calendar Unit Testing: For fun dayBinderIsCalledOnDayChanged(), this test checks to make sure that the dayBinder is working properly to update any changes made to a selected day. For fun programmaticScrollWorksAsExpected(), this test checks to confirm that the scrollToMonth correctly scrolls to the specified month.

Limitations

We had a few complications throughout the project. During the final few weeks of the project, a member had some medical problems so that they could no longer fully contribute to the project. Leaving us to have to make up a quarter of the work in the last week. Brad and Sean focused on taking over this aspect of the code and managed to get it complete in a few days. In doing so we were unable to make the Work Manager work within our scope, and ultimately had to drop it. The Work Manager runs tasks in the background, but you cannot retrieve the tasks once they start. So you can't run the timer in the background, and then retrieve it once you reopen the app. So we had to scrap this idea. We also had trouble implementing LiveData, we eventually realized we were creating a new LiveData object every time we accessed the fragment, rather than using a persistent one. We spent at least ten to twenty hours over two days researching both LiveData and Work Manager attempting to fully understand what they're capable of before we were able to get our proper implementation.

Our team's schedule was not aligned at all which contributed to a complicated experience with time management and communication. A few of us work during the week, others were overloaded full time students that worked the weekend, and all of us had other projects outside of this one with similar issues to the one I discussed in the last paragraph. We managed to reconcile these complexities by employing the use of a group trello board as guidance for project checkpoints/deadlines. Additionally, multiple remote voice chat meetings were held after designating a timeframe for when we all had a half hour to sit down, look at the app and discuss in detail scope of work to be divided, progress over time, and potential blockers encountered. Sean tried to solve minor problems, since he had the least amount of classes this semester so that we didn't have to wait for others to be free. And we all spoke regularly in our group discord server to share any issues we found, or let people know when we can hop in to solve a problem. So even though we had very different schedules, we managed to make it work and complete the project on time.

Contributor	**Contributions**
Sean Perry	➢ Created the color design for the app. ➢ Created the Analytics classes, fragments, and functions to draw and display the data to the app. ➢ Created the test cases for the Plot class. ➢ Implemented the Firebase authentication. Implemented the main Firestore app, and created the General data types. ➢ Supported Brad a small amount on the Calendars with the portion that reads in the Firestore data, and helped with the recycler view. ➢ Created the timer functionality, UI for the timer, the LiveData class that .

	➢ Helped with minor problems, since I had a higher availability.
Bradley Tran	➢ Built the calendar fragment ➢ Helped support Timer development ➢ Researched and implemented the API for the calendar app ➢ Added the test cases for the Calendar code ➢ Built the functionality for bottom navigation menu ➢ Implemented permissions requests to respect users privacy and preferences
Trevor Kvanvig	➢ Implemented the rewards fragment ➢ Built the integration into firebase to help the rewards functionality. ➢ Supported Firestore development ➢ Helped clean up on other fragments UI ➢ Created the logic for each of the rewards
Jamison Coombs	➢ Created the AddTaskFragment implementation and UI to create a new task. ➢ Created the Task class that stores all of the task data. ➢ Created functions in the FireStoreData class for all of the different Firestore functionality including adding a task, updating a task, deleting a task and completing a task. ➢ Created the notifications for when a task is due. ➢ Created test cases for the tasks. ➢ Contributed in making the slides and report.
Edward Uriarte	➢ Contributed the Synapflow image ➢ Helped build the base timer ➢ Helped build the base LiveData ➢ Helped General UI layout

Conclusion

We believe our app is a great solution to those who need something that can keep them busy studying and keep track of their tasks in one place. Using the Pomodoro Method is proven to be a strong method for studying. Viewing your completed sessions per day is valuable to keep you motivated. Tasks help you stay organized, and you can plan ahead your tasks. Rewards help you stay motivated and give you that much more to work towards. Having your schedule mapped out on a calendar is a great way to visually know what is going on.

In conclusion this app allowed us to work as a team and implement an app to our expectations. We ran into issues and problems that we did not anticipate in the beginning of the project, and that caused us to scrap some portions entirely, like the work manager. We quickly

adapted as a team to find a new solution and were able to bring our idea to life. We were able to avoid scope creep, and managed to keep every expectation under manageable deadlines. Our team dynamic was great throughout the entire project, and we learned a lot of lessons as a group that we plan on taking with us into our new careers as Software Engineers.

References

Forrester, Alex. *How to Build Android Apps with Kotlin - A Hands on Guide to Developing,*. PACKT PUBLISHING LIMITED, 2023.

Griffiths, Dawn. *Head First Android Development*. O'Reilly, 2022.

Kizitonwose. "Kizitonwose/Calendar: A Highly Customizable Calendar View and Compose Library for Android." *GitHub*, github.com/kizitonwose/Calendar. Accessed 12 May 2023.

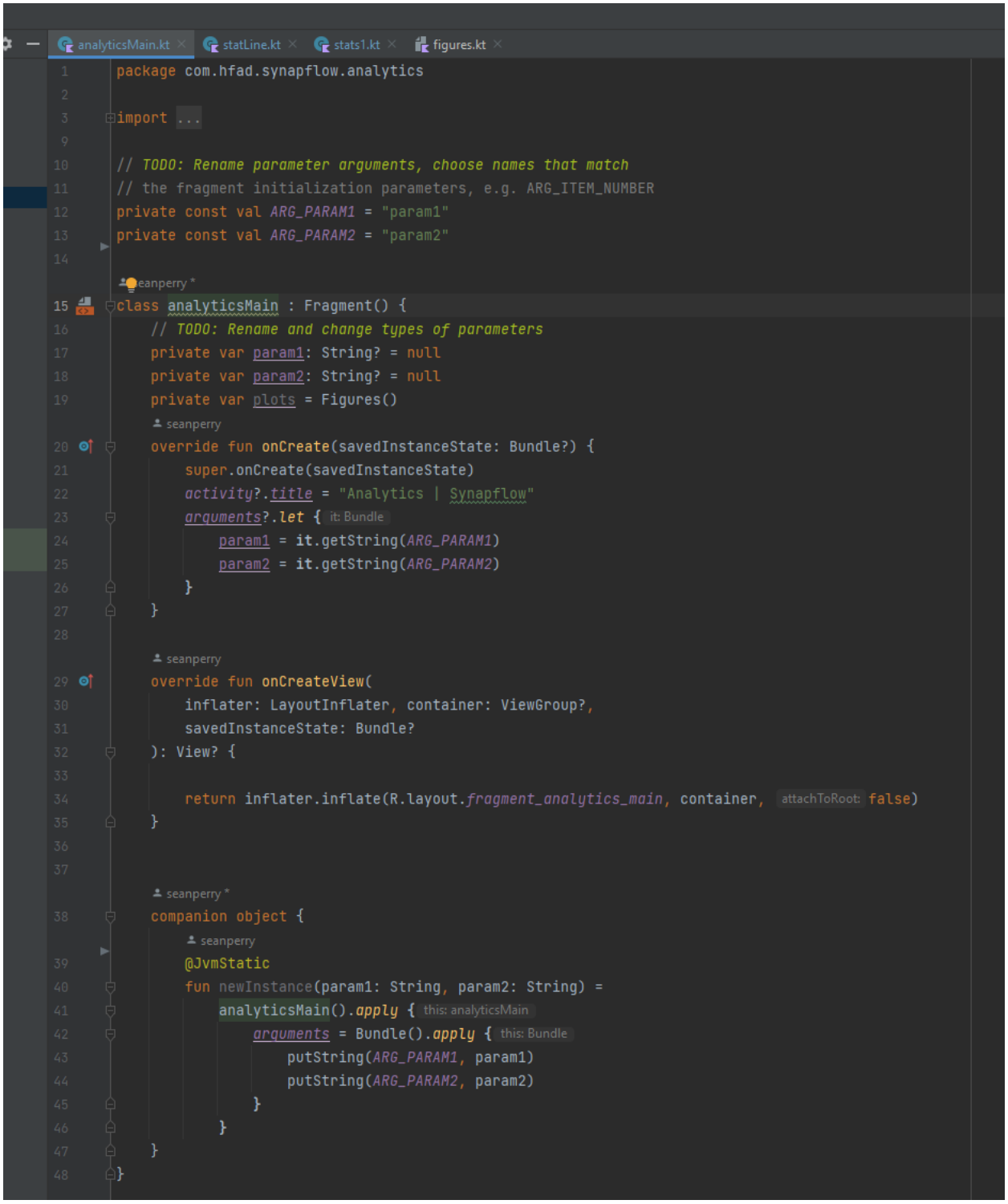
PhilJay. "Philjay/MPAndroidChart: A Powerful 🚀 Android Chart View / Graph View Library, Supporting Line- Bar- Pie- Radar- Bubble- and Candlestick Charts as Well as Scaling, Panning and Animations." *GitHub*, github.com/PhilJay/MPAndroidChart. Accessed 12 May 2023.

YouTube, YouTube, 5 Jan. 2018, https://www.youtube.com/watch?v=VnRxq4Fpl64&t=1210s&ab_channel=ResoCoder. Accessed 12 May 2023.

Zhanghai. "Zhanghai/Materialprogressbar: Material Design Progressbar with Consistent Appearance." *GitHub*, github.com/zhanghai/MaterialProgressBar. Accessed 12 May 2023.

Appendix

AnalyticMain.kt



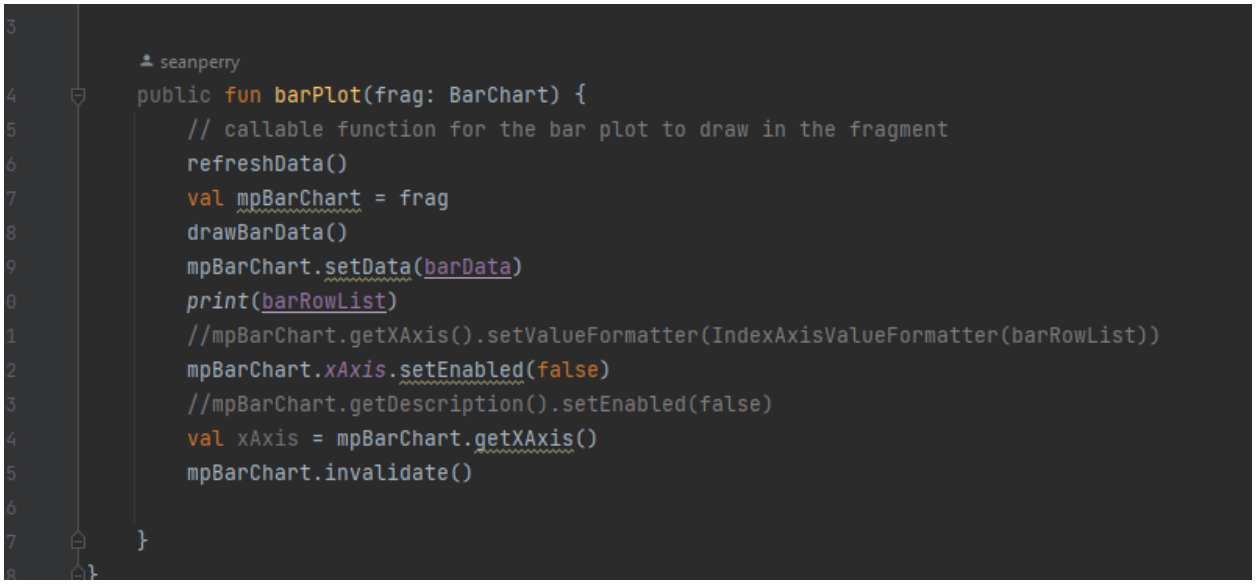
```
analyticsMain.kt | statLine.kt | stats1.kt | figures.kt

1  package com.hfad.synapflow.analytics
2
3  import ...
9
10 // TODO: Rename parameter arguments, choose names that match
11 // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
12 private const val ARG_PARAM1 = "param1"
13 private const val ARG_PARAM2 = "param2"
14
   eanperry *
15 class analyticsMain : Fragment() {
16     // TODO: Rename and change types of parameters
17     private var param1: String? = null
18     private var param2: String? = null
19     private var plots = Figures()
       seanperry
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         activity?.title = "Analytics | Synapflow"
23         arguments?.let { it: Bundle
24             param1 = it.getString(ARG_PARAM1)
25             param2 = it.getString(ARG_PARAM2)
26         }
27     }
28
       seanperry
29     override fun onCreateView(
30         inflater: LayoutInflater, container: ViewGroup?,
31         savedInstanceState: Bundle?
32     ): View? {
33
34         return inflater.inflate(R.layout.fragment_analytics_main, container, attachToRoot: false)
35     }
36
37
       seanperry *
38     companion object {
           seanperry
39         @JvmStatic
40         fun newInstance(param1: String, param2: String) =
41             analyticsMain().apply { this: analyticsMain
42                 arguments = Bundle().apply { this: Bundle
43                     putString(ARG_PARAM1, param1)
44                     putString(ARG_PARAM2, param2)
45                 }
46             }
47     }
48 }
```

figures.kt

```kotlin
import ...

seanperry
class Figures {
    //private lateinit var fsdb : FirestoreData
    private lateinit var dateData : MutableMap<String, Any>
    private lateinit var lineData: LineData
    private lateinit var barData: BarData
    private lateinit var dateList: MutableList<String>
    private lateinit var barRowList: MutableList<String>
    seanperry
    constructor() {
        // Initialize plots
        dateData = mutableMapOf<String, Any>()
        dateList = mutableListOf<String>()
        barRowList = mutableListOf<String>()
    }

    seanperry
    public fun createTestData() {
        // Used for test cases, cant access firestore to fill data
        val today = LocalDate.now()
        dateData = mutableMapOf<String, Any>()
        for (i in 0 ≤ .. ≤ 31) {
            dateData[today.minusDays(i.toLong()).toString()] = mutableListOf<String>("1", "2", "3", "4")
        }
    }

    seanperry
    public fun getTestData() : MutableMap<String, Any>{
        // Returns dateData for the test functions
        return dateData
    }
    seanperry
    public fun refreshData() {
        // Re pulls the data from firestore to draw
        dateData = fsdb.getCompletionDates()
        println(dateData)
    }
```

```
47
                seanperry
48      fun getDayCounts(date : String) : Float {
49          // Creates a loop to count the dates
50          val data = dateData?.get(date)
51          if (data is List<*>) {
52              println(date)
53              println(data.filterIsInstance<String>().size.toFloat())
54              return data.filterIsInstance<String>().size.toFloat()
55          }
56          return 0f
57      }
58
59
                seanperry
60      public fun getWeekLine(weekAgo: Int = 0) : MutableList<Entry> {
61          // Draws a line for a week, returns the list for it
62          val test = mutableListOf<Entry>()
63          dateList = mutableListOf<String>()
64          val today = LocalDate.now()
65          for (i in 0 ≤ .. ≤ 7) {
66              val date = today.minusDays(i.toLong())
67              dateList.add("${date.monthValue}-${date.dayOfMonth}")
68              test.add(Entry(i.toFloat(), getDayCounts(date.toString())))
69          }
70          return test
71      }
                seanperry
72      public fun getBarCount() : MutableList<BarEntry>{
73          // Get counts for each bar for the last 7, 7-14, 15-21, 22-28 days
74          var weekIncr = 4
75          barRowList = mutableListOf<String>()
76          val test = mutableListOf<BarEntry>()
77          val today = LocalDate.now()
78          for (j in 4 ≥ downTo ≥ 1) {
79              var counter: Float = 0f
80              for (i in 7 ≥ downTo ≥ 1) {
81                  val date = today.minusDays(((j * 7) - i).toLong()).toString()
82                  counter += getDayCounts(date)
83              }
84              weekIncr--
85              val date =  today.minusDays(((j * 7) - 1).toLong())
86              test.add(BarEntry(j.toFloat(), counter))
87              barRowList.add("Wk of ${date.monthValue}-${date.dayOfMonth}")
88          }
89          return test
90      }
91
```

```
seanperry
public fun linePlot(frag: LineChart) {
    // Callable function to draw a linePlot to a fragment
    refreshData()
    // Needs to be the line chart fragment passed in
    val mpLineChart =  frag
    drawLineData()
    mpLineChart.setData(lineData)

    mpLineChart.getXAxis().setValueFormatter(IndexAxisValueFormatter(dateList))

    mpLineChart.getDescription().setEnabled(false)
    mpLineChart.invalidate()
}
seanperry
private fun drawLineData() {
    // Get the line (x,y) points. for the line

    refreshData()
    val test = getWeekLine()
    // Now the weird part add to a list, to another list to another..
    var lineDs = LineDataSet(test, label: "Daily Study Counts")
    lineDs.setCircleColor(0xFFF7D86D.toInt())
    lineDs.setColors(mutableListOf<Int>(0xFF47C6B9.toInt()))
    var dataSet = mutableListOf<ILineDataSet>()

    dataSet.add(lineDs)
    lineData = LineData(dataSet)
}


seanperry
private fun drawBarData() {
    // Logic for frawing thr bars
    val barList = getBarCount()

    var barDs = BarDataSet(barList, label: "Last Months Study Sessions by Week")
    barDs.setColors(mutableListOf<Int>(0xFF4C4C4C.toInt(), 0xFFF7D86D.toInt(),0xFF9BD5A9.toInt(), 0xFF47C6B9.toInt()))
    var barDataSet = mutableListOf<IBarDataSet>() // B G Y GR

    barDataSet.add(barDs)
    barData = BarData(barDataSet)
}
```



```
    seanperry
    public fun barPlot(frag: BarChart) {
        // callable function for the bar plot to draw in the fragment
        refreshData()
        val mpBarChart = frag
        drawBarData()
        mpBarChart.setData(barData)
        print(barRowList)
        //mpBarChart.getXAxis().setValueFormatter(IndexAxisValueFormatter(barRowList))
        mpBarChart.xAxis.setEnabled(false)
        //mpBarChart.getDescription().setEnabled(false)
        val xAxis = mpBarChart.getXAxis()
        mpBarChart.invalidate()

    }
}
```

statsLine.kt

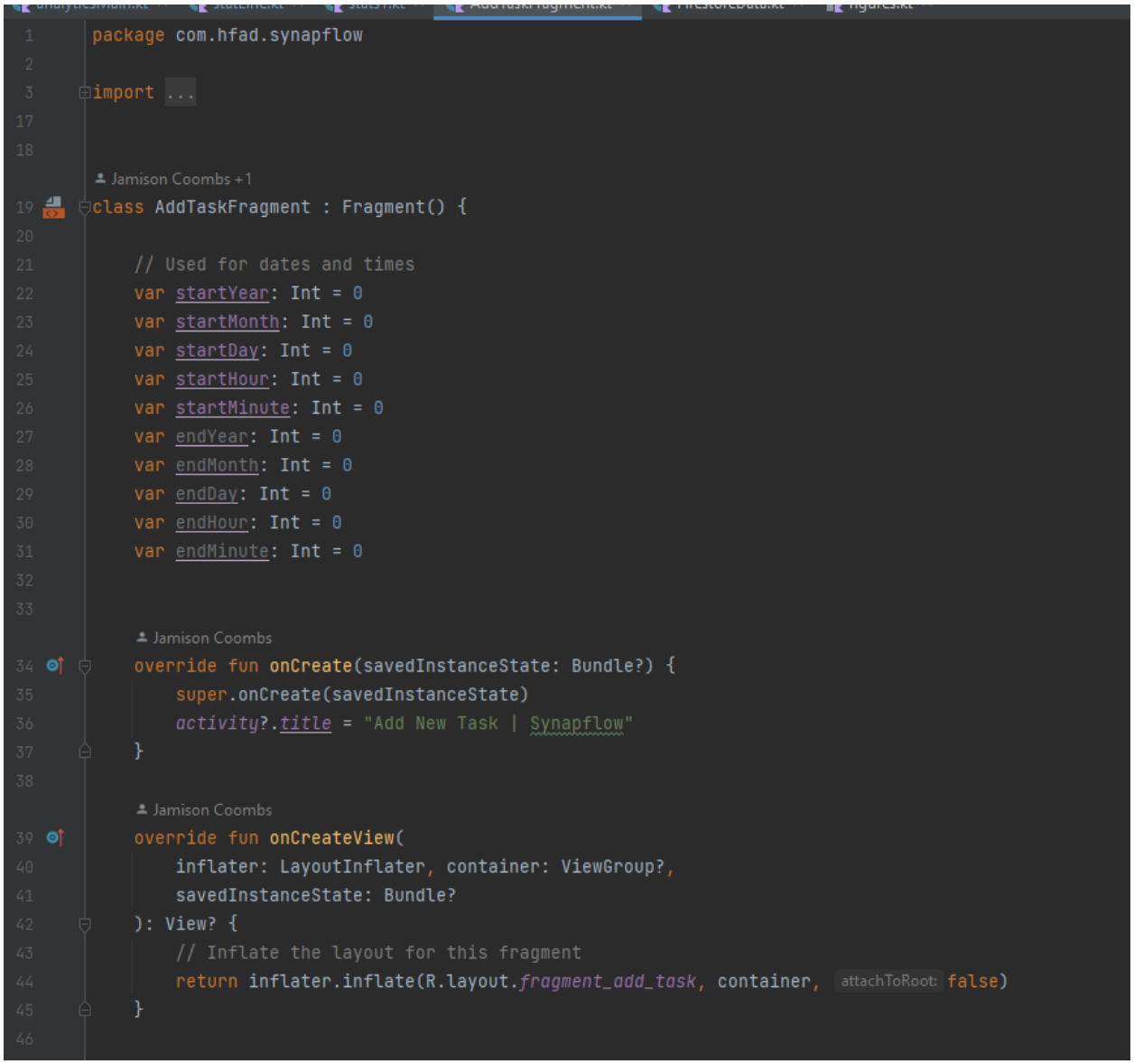
```kotlin
// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"
seanperry *
class statLine : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    //private lateinit var plots : Figures
    seanperry
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

    seanperry
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_stat_line, container, attachToRoot: false)
    }

    seanperry *
    companion object {
        // TODO: Rename and change types and number of parameters
        seanperry
        @JvmStatic
        fun newInstance(param1: String, param2: String) =
            statLine().apply { this: statLine
                arguments = Bundle().apply { this: Bundle
                    putString(ARG_PARAM1, param1)
                    putString(ARG_PARAM2, param2)
                }
            }
    }
    seanperry
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Draws barplot to fragment
        plots.barPlot(view?.findViewById(R.id.plot2)!!)

    }

}
```

stats1.kt

analyticsMain.kt | statLine.kt | stats1.kt | figures.kt

```kotlin
package com.hfad.synapflow.analytics

import ...

// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

seanperry *
class stats1 : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null
    //private var plots = Figures()
    seanperry
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }

    }

    seanperry
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_stats1, container, attachToRoot: false)
    }

    seanperry
    companion object {
        seanperry
        @JvmStatic
        fun newInstance(param1: String, param2: String) =
            stats1().apply { this: stats1
                arguments = Bundle().apply { this: Bundle
                    putString(ARG_PARAM1, param1)
                    putString(ARG_PARAM2, param2)
                }
            }
    }
    seanperry *
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        plots.linePlot(view?.findViewById(R.id.plot1)!!)
    }



}
```

AddTaskFragment.kt



```kotlin
package com.hfad.synapflow

import ...


Jamison Coombs +1
class AddTaskFragment : Fragment() {

    // Used for dates and times
    var startYear: Int = 0
    var startMonth: Int = 0
    var startDay: Int = 0
    var startHour: Int = 0
    var startMinute: Int = 0
    var endYear: Int = 0
    var endMonth: Int = 0
    var endDay: Int = 0
    var endHour: Int = 0
    var endMinute: Int = 0


    Jamison Coombs
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        activity?.title = "Add New Task | Synapflow"
    }

    Jamison Coombs
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_add_task, container, attachToRoot: false)
    }

```

```
Jamison Coombs +1
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    // Set up the spinner for the task's category
    val spinner = view.findViewById<Spinner>(R.id.category_spinner)
    val options = resources.getStringArray(R.array.taskCategoryOptions)
    val adapter = ArrayAdapter(requireContext(), android.R.layout.simple_spinner_item, options)
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    spinner.adapter = adapter

    // Set up the date and time picker dialogs for the start date and time EditTexts
    setStartDateTimeListener(view)

    // Set up the SeekBar for the priority
    val prioritySeekBar = view.findViewById<SeekBar>(R.id.priority_seekbar)
    val priorityLabel = view.findViewById<TextView>(R.id.priorityLabel)

    // Set up the text labels for each SeekBar position
    val priorityLabels = listOf("Very Low", "Low", "Medium", "High", "ASAP")

    // Listen for changes to the SeekBar position
    prioritySeekBar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {
        override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {
            // Update the text label to reflect the current SeekBar position
            priorityLabel.text = priorityLabels[progress]
        }
        override fun onStartTrackingTouch(seekBar: SeekBar?) {}
        override fun onStopTrackingTouch(seekBar: SeekBar?) {}
    })

    view.findViewById<Button>(R.id.addTaskButton).setOnClickListener { it: View!
        // Get the task name
        val taskName = view.findViewById<EditText>(R.id.name_txt).text.toString()
        val taskDescription = view.findViewById<EditText>(R.id.description_txt).text.toString()
        // Get the task category
        val taskCategory = spinner.selectedItem.toString()
        // Get the task priority
        val taskPriority = prioritySeekBar.progress.toLong()
        val startDateTime = Date( year: startYear - 1900, startMonth, startDay, startHour, startMinute)

        val startDateStamp : Timestamp;
        val endDateStamp : Timestamp;

        if(startYear > 1900) {
            startDateStamp =  Timestamp(startDateTime);
        } else {
            startDateStamp = Timestamp(Date())
        }
```

```
95
96              // Create a new task object
97              val task = Task(taskName, taskDescription, taskCategory, startDateStamp, taskPriority)
98
99              // Add the task to the database
100             FirestoreData().addTask(task)
101
102             val alarmManager = activity?.getSystemService(Context.ALARM_SERVICE) as AlarmManager
103             val notificationIntent = Intent(context, NotificationBroadcastReceiver::class.java)
104             notificationIntent.addFlags(Intent.FLAG_RECEIVER_FOREGROUND)
105             val pendingIntent = PendingIntent.getBroadcast(context,  requestCode: 0, notificationIntent,  flags: PendingIntent.FLAG_IMMUTABLE or PendingIntent.FLAG_UPDATE_CURRENT)
106
107             val calendar = Calendar.getInstance()
108             calendar.set(Calendar.YEAR, startYear)
109             calendar.set(Calendar.MONTH, startMonth)
110             calendar.set(Calendar.DAY_OF_MONTH, startDay)
111             calendar.set(Calendar.HOUR_OF_DAY, startHour)
112             calendar.set(Calendar.MINUTE, startMinute)
113             calendar.set(Calendar.SECOND, 0)
114
115             Log.d( tag: "calendar", calendar.time.toString())
116
117             // Set the alarm to trigger at the specified date and time
118             alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, calendar.timeInMillis, pendingIntent)
119             fsdb.getTaskMap()
120             // Exit fragment
121             requireActivity().supportFragmentManager.popBackStackImmediate()
122
123         }
124     }
125
```

```
Jamison Coombs
private fun setStartDateTimeListener(view: View) {
    val dateEditText = view.findViewById<EditText>(R.id.startDateEditText)
    val timeEditText = view.findViewById<EditText>(R.id.startTimeEditText)
    val calendar = Calendar.getInstance()
    // Set up the date picker dialog
    val datePickerDialog = DatePickerDialog(
        requireContext(),
        { _, year, month, dayOfMonth ->
            calendar.set(year, month, dayOfMonth)
            startYear = year
            startMonth = month
            startDay = dayOfMonth
            val dateFormat = SimpleDateFormat( pattern: "MMM dd, yyyy", Locale.getDefault())
            dateEditText.setText(dateFormat.format(calendar.time))
        },
        calendar.get(Calendar.YEAR),
        calendar.get(Calendar.MONTH),
        calendar.get(Calendar.DAY_OF_MONTH)
    )
    // Set up the time picker dialog
    val timePickerDialog = TimePickerDialog(
        requireContext(),
        { _, hourOfDay, minute ->
            calendar.set(Calendar.HOUR_OF_DAY, hourOfDay)
            calendar.set(Calendar.MINUTE, minute)
            startHour = hourOfDay
            startMinute = minute
            val timeFormat = SimpleDateFormat( pattern: "hh:mm a", Locale.getDefault())
            timeEditText.setText(timeFormat.format(calendar.time))
        },
        calendar.get(Calendar.HOUR_OF_DAY),
        calendar.get(Calendar.MINUTE),
        is24HourView: false
    )
    // Set the date picker dialog to open when the date EditText is clicked
    dateEditText.setOnClickListener { it: View!
        datePickerDialog.show()
    }
    // Set the time picker dialog to open when the time EditText is clicked
    timeEditText.setOnClickListener { it: View!
        timePickerDialog.show()
    }
}


}
```

CalendarFragment.kt

analyticsMain.kt | statLine.kt | stats1.kt | AddTaskFragment.kt | CalendarFragment.kt | FirestoreData.kt | figures.kt

```kotlin
package com.hfad.synapflow

import ...

seanperry
data class Event(
        val id: String,
        val time: String,
        val text: String,
        val desc: String,
        val date: LocalDate,
        var completed: Boolean
    )
Bradley T +1
class CalendarEventsAdapter(val onClick: (Event) -> Unit) :
    RecyclerView.Adapter<CalendarEventsAdapter.CalendarEventsViewHolder>() {

    val events = mutableListOf<Event>()

    Bradley T
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CalendarEventsViewHolder {
        return CalendarEventsViewHolder(
            CalendarEventItemViewBinding.inflate(parent.context.layoutInflater, parent, attachToParent: false),
        )
    }

    Bradley T
    override fun onBindViewHolder(viewHolder: CalendarEventsViewHolder, position: Int) {
        viewHolder.bind(events[position])
    }

    Bradley T
    override fun getItemCount(): Int = events.size

    Bradley T +1
    inner class CalendarEventsViewHolder(private val binding: CalendarEventItemViewBinding) :
        RecyclerView.ViewHolder(binding.root) {

        Bradley T
        init {
            itemView.setOnClickListener { it: View!
                onClick(events[bindingAdapterPosition])
            }
        }

```

```
69
                seanperry +1
70          fun bind(event: Event) {
71              // Format from FB event.
72              binding.cardTitle.text = event.text
73              binding.description.text = event.desc
74              binding.time.text = event.time
75              if (event.completed)
76                  binding.completed.text = "Complete"
77              else
78                  binding.completed.text = "Not Complete"
79          }
80      }
81  }
82
            Bradley T +2
83  class CalendarFragment : Fragment(R.layout.fragment_calendar) {
84
85      //private val fsdb = FirestoreData()
86      private var taskMap = mutableMapOf<String, Task>()
87
88
89      private val eventsAdapter = CalendarEventsAdapter { it: Event
90          AlertDialog.Builder(requireContext()) AlertDialog.Builder
91              .setMessage("Complete this task?") AlertDialog.Builder!
92              .setPositiveButton("Complete") { _, _ ->
93                  // ISend an update to FB to complete the task.
94                  completeEvent(it)
95              }
96              .setNeutralButton("Delete") { _, _ ->
97                  fsdb.deleteTask(it.id)
98                  deleteEvent(it)
99              }
100             .setNegativeButton("Close", listener: null)
101             .show()
102     }
103
104     private val inputDialog by lazy {
105         val editText = AppCompatEditText(requireContext())
106         val layout = FrameLayout(requireContext()).apply { this: FrameLayout
107             // Setting the padding on the EditText only pads the input area
108             // not the entire EditText so we wrap it in a FrameLayout.
109             val padding = dpToPx( dp: 20, requireContext())
110             setPadding(padding, padding, padding, padding)
111             addView(editText, FrameLayout.LayoutParams(
112                 ViewGroup.LayoutParams.MATCH_PARENT,
113                 ViewGroup.LayoutParams.WRAP_CONTENT
114             ))
115         }
```

```
        AlertDialog.Builder(requireContext()) AlertDialog.Builder
            .setTitle("Enter event title") AlertDialog.Builder!
            .setView(layout)
            .setPositiveButton("Save") { _, _ ->
                saveEvent(editText.text.toString())
                // Prepare EditText for reuse.
                editText.setText("")
            }
            .setNegativeButton("Close", listener: null)
            .create() AlertDialog
            .apply { this: AlertDialog
                setOnShowListener { it: DialogInterface!
                    // Show the keyboard
                    editText.requestFocus()
                    context.inputMethodManager
                        .toggleSoftInput(InputMethodManager.SHOW_FORCED, hideFlags: 0)
                }
                setOnDismissListener { it: DialogInterface!
                    // Hide the keyboard
                    context.inputMethodManager
                        .toggleSoftInput(InputMethodManager.HIDE_IMPLICIT_ONLY, hideFlags: 0)
                }
            } ^lazy
    }

    private var selectedDate: LocalDate? = null
    private val today = LocalDate.now()

    private val selectionFormatter = DateTimeFormatter.ofPattern( pattern: "d MMM yyyy")
    private val events = mutableMapOf<LocalDate, List<Event>>()

    private lateinit var binding: FragmentCalendarBinding


    Bradley T +2
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        activity?.title = "Calendar | Synapflow"

        addStatusBarColorUpdate(R.color.calendar_statusbar_color)
        binding = FragmentCalendarBinding.bind(view)
        binding.calendarRv.apply { this: RecyclerView
            layoutManager = LinearLayoutManager(requireContext(), RecyclerView.VERTICAL, reverseLayout: false)
            adapter = eventsAdapter
            addItemDecoration(DividerItemDecoration(requireContext(), RecyclerView.VERTICAL))
        }

        val daysOfWeek = daysOfWeek()
        val currentMonth = YearMonth.now()
        val startMonth = currentMonth.minusMonths( monthsToSubtract: 50)
        val endMonth = currentMonth.plusMonths( monthsToAdd: 50)
```

```
166         val endMonth = currentMonth.plusMonths( monthsToAdd: 50)
167
168         configureBinders(daysOfWeek)
169         binding.calendarView.apply { this: CalendarView
170             setup(startMonth, endMonth, daysOfWeek.first())
171             scrollToMonth(currentMonth)
172         }
173
174
175         if (savedInstanceState == null) {
176             // Show today's events initially.
177             binding.calendarView.post { selectDate(today) }
178         }
179         //binding.eventAddButton.setOnClickListener { inputDialog.show() }
180
181         getTaskList()
182         getTasksForCalender()
183     }
184
        Bradley T
185     private fun selectDate(date: LocalDate) {
186         if (selectedDate != date) {
187             val oldDate = selectedDate
188             selectedDate = date
189             oldDate?.let { binding.calendarView.notifyDateChanged(it) }
190             binding.calendarView.notifyDateChanged(date)
191             updateAdapterForDate(date)
192         }
193     }
194
        Bradley T + 1
195     private fun getTaskList() {
196         taskMap = fsdb.getTaskMap()
197     }
198
        Bradley T + 1
199     public fun getTasksForCalender() {
200         /*
201             Updates the task in Firestore to complete the task.
202         */
203         // Helper function to convert date.
204         fun asLocalDate(date: Date): LocalDate {
205             return Instant.ofEpochMilli(date.time).atZone(ZoneId.systemDefault()).toLocalDate()
206         }
207         clearEvents()
208         // Create time parameter
209         for ((key, task) in taskMap) {
210             var date = asLocalDate(task.startTimeStamp.toDate())
211             var intHrs = task.startTimeStamp.toDate().hours.toInt()
212             var intMins = task.startTimeStamp.toDate().minutes.toInt()
213             var strMins = task.startTimeStamp.toDate().minutes.toString()
214             var strTime = ""
```

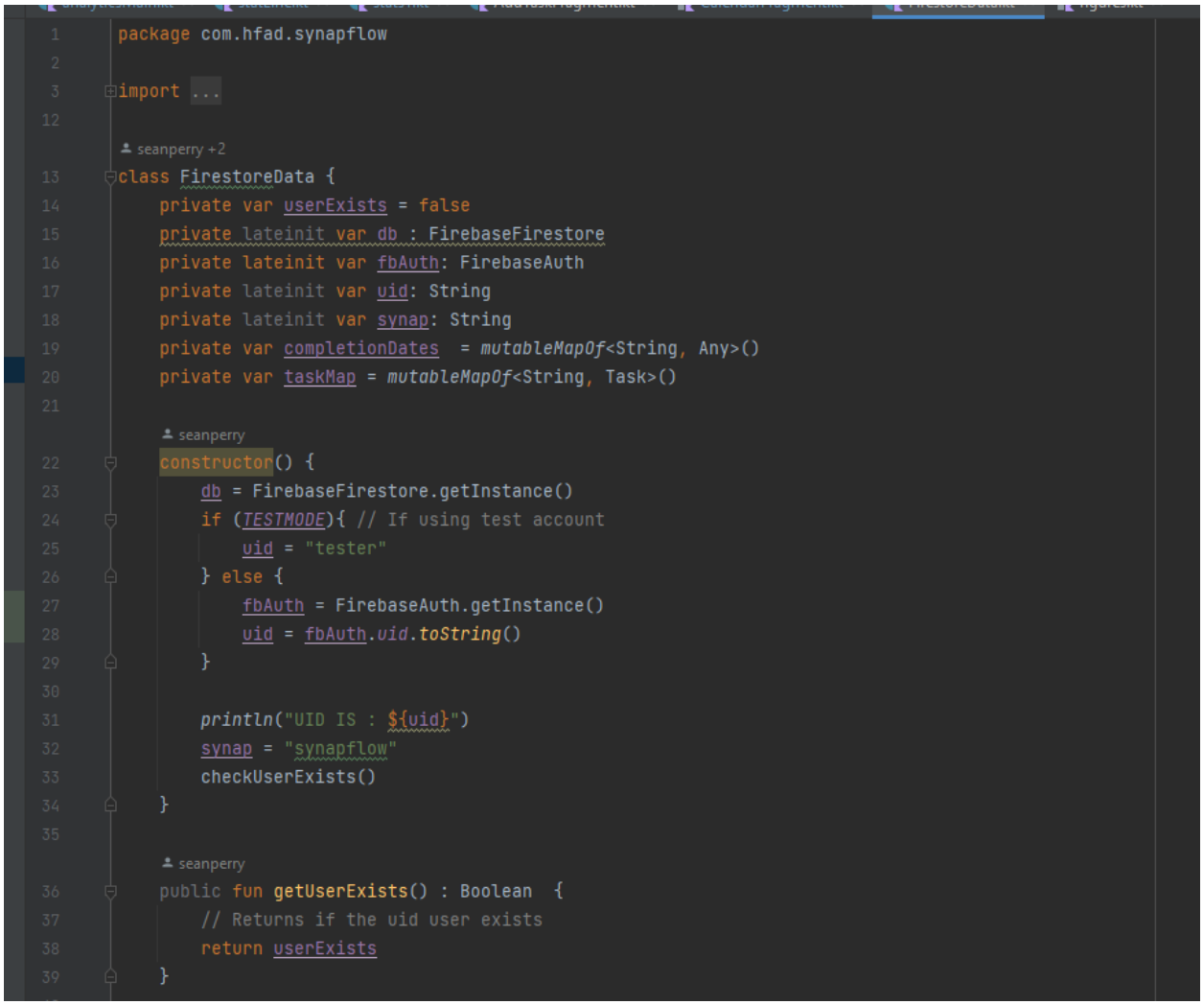
```
            //Check timestamps minutes to account for example: 1:9 PM
            //which should be 1:09 PM
            if (strMins.length < 2) {
                strMins = "0" + strMins
            }
            else {
                strMins
            }

            //Checks for converting military time format to
            //standard time format
            if (intHrs in 13 ≤ .. ≤ 23) {
                strTime = (intHrs - 12).toString() + ":" + strMins + " PM"
            }
            else if (intHrs in 1 ≤ .. ≤ 11) {
                strTime = intHrs.toString() + ":" + strMins + " AM"
            }
            else if (intHrs.equals(12) && intMins >= 0) {
                strTime = intHrs.toString() + ":" + strMins + " PM"
            }
            else if (intHrs.equals(0) && intMins >= 0) {
                strTime = (intHrs + 12).toString() + ":" + strMins + " AM"
            }

            events[date] =
                events[date].orEmpty().plus(Event(
                    key,
                    strTime,
                    task.name,
                    task.description,
                    date,
                    task.completed
                ))
            updateAdapterForDate(date)
        }
    }

    // DOESNT WORK FOR NOW
    seanperry +1
    private fun saveEvent(text: String) {
        // DOESNT WORK FOR NOW
        /*
        DOESNT WORK FOR NOW

         */
        if (text.isBlank()) {
            Toast.makeText(requireContext(), "Text is empty", Toast.LENGTH_LONG)
                .show()
        } else {
            selectedDate?.let { it: LocalDate
                //events[it] =
                    //events[it].orEmpty().plus(Event(UUID.randomUUID().toString(), text, it))
                //updateAdapterForDate(it)
            }
        }
```

```
seanperry +1
private fun clearEvents() {
    /*
    Clears events to redraw
     */
    for (eventList in events.values) {
        for (event in eventList){
            deleteEvent(event)
        }
    }
}

seanperry
private fun completeEvent(event: Event) {
    // Tells Firestore to update the event.
    fsdb.markTaskCompleted(event.id)
    event.completed = true
    // Request the taskListUpdate.
    getTaskList()
}
Bradley T +1
private fun deleteEvent(event: Event) {
    val date = event.date
    events[date] = events[date].orEmpty().minus(event)
    updateAdapterForDate(date)
}

Bradley T
private fun updateAdapterForDate(date: LocalDate) {
    eventsAdapter.apply { this: CalendarEventsAdapter
        events.clear()
        events.addAll(this@CalendarFragment.events[date].orEmpty())
        notifyDataSetChanged()
    }
    binding.currentDateText.text = selectionFormatter.format(date)
}

Bradley T
private fun configureBinders(daysOfWeek: List<DayOfWeek>) {
    Bradley T
    class DayViewContainer(view: View) : ViewContainer(view) {
        lateinit var day: CalendarDay // Will be set when this container is bound.
        val binding = CalendarDayLayoutBinding.bind(view)

        Bradley T
        init {
            view.setOnClickListener { it: View!
                if (day.position == DayPosition.MonthDate) {
                    selectDate(day.date)
                }
            }
        }
```

```kotlin
binding.calendarView.dayBinder = object : MonthDayBinder<DayViewContainer> {
    override fun create(view: View) = DayViewContainer(view)
    @SuppressLint("ResourceAsColor")
    override fun bind(container: DayViewContainer, data: CalendarDay) {
        container.day = data
        val textView = container.binding.calendarDayText
        val dotView = container.binding.calendarDotView

        textView.text = data.date.dayOfMonth.toString()

        if (data.position == DayPosition.MonthDate) {
            textView.makeVisible()
            when (data.date) {
                today -> {
                    textView.setTextColorRes(R.color.calendar_white)
                    textView.setBackgroundResource(R.drawable.calendar_today_bg)
                    dotView.makeInVisible()
                }
                selectedDate -> {
                    textView.setTextColorRes(R.color.calendar_blue)
                    textView.setBackgroundResource(R.drawable.calendar_selected_bg)
                    dotView.makeInVisible()
                }
                else -> {
                    textView.setTextColorRes(R.color.calendar_black)
                    textView.background = null
                    dotView.isVisible = events[data.date].orEmpty().isNotEmpty()
                }
            }
        } else {
            //textView.makeInVisible()
            textView.setTextColor(R.color.calendar_grey_light)
            dotView.makeInVisible()
        }
    }
}

val currentMonth = YearMonth.now()
val startMonth = currentMonth.minusMonths( monthsToSubtract: 50)
val endMonth = currentMonth.plusMonths( monthsToAdd: 50)
val firstDayOfWeek = WeekFields.of(Locale.getDefault()).firstDayOfWeek

binding.calendarView.setup(startMonth, endMonth, firstDayOfWeek)
binding.calendarView.scrollToMonth(currentMonth)
```

```
        val titlesContainer = view?.findViewById<ViewGroup>(R.id.titlesContainer)
        if (titlesContainer != null) {
            titlesContainer.children Sequence<View>
                .map { it as TextView } Sequence<TextView>
                .forEachIndexed { index, textView ->
                    val dayOfWeek = daysOfWeek[index]
                    val title = dayOfWeek.getDisplayName(TextStyle.SHORT, Locale.getDefault())
                    textView.text = title
                }
        }

        Bradley T
        class MonthViewContainer(view: View) : ViewContainer(view) {
            val titlesContainer = view as ViewGroup
        }
        binding.calendarView.monthHeaderBinder =
            object : MonthHeaderFooterBinder<MonthViewContainer> {
                override fun create(view: View) = MonthViewContainer(view)
                override fun bind(container: MonthViewContainer, data: CalendarMonth) {
                    // Setup each header day text if we have not done that already.
                    if (container.titlesContainer.tag == null) {
                        container.titlesContainer.tag = data.yearMonth
                        container.titlesContainer.children.map { it as TextView }
                            .forEachIndexed { index, tv ->
                                tv.text = daysOfWeek[index].name.first().toString()
                                tv.setTextColorRes(R.color.calendar_black)
                            }
                    }
                }
            }
    }
}
```

FirestoreData.kt



```
package com.hfad.synapflow

import ...

seanperry +2
class FirestoreData {
    private var userExists = false
    private lateinit var db : FirebaseFirestore
    private lateinit var fbAuth: FirebaseAuth
    private lateinit var uid: String
    private lateinit var synap: String
    private var completionDates  = mutableMapOf<String, Any>()
    private var taskMap = mutableMapOf<String, Task>()

    seanperry
    constructor() {
        db = FirebaseFirestore.getInstance()
        if (TESTMODE){ // If using test account
            uid = "tester"
        } else {
            fbAuth = FirebaseAuth.getInstance()
            uid = fbAuth.uid.toString()
        }

        println("UID IS : ${uid}")
        synap = "synapflow"
        checkUserExists()
    }

    seanperry
    public fun getUserExists() : Boolean  {
        // Returns if the uid user exists
        return userExists
    }
```

```
41    public fun checkUserExists(){
42        // If the user exists, good and update data maps and lists. Otherwise create account
43        val userRef = db.collection(synap).document(uid).collection( collectionPath: "General").document( documentPath: "CompletionCount")
44        userRef.get()
45            .addOnSuccessListener { doc ->
46                if (doc.getData() == null) {
47                    println(doc)
48                    println("Doc not found, creating user...")
49                    createUser()
50                } else {
51                    println("Doc Found!")
52                    userExists = true
53                    updateCompletionDates(completionDates)
54                    getTaskMap()
55                }
56            }
57            .addOnFailureListener { doc ->
58                println("err")
59            }
60    }
61
62    /* Insert data */
63
64    /**
65     * Adds a task to the database, under the current user
66     */
          Jamison Coombs +1
67    fun addTask(task: Task) {
68        // Adds a task into firestore
69        val data: MutableMap<String, Any?> = HashMap()
70        data["name"] = task.name
71        data["description"] = task.description
72        data["category"] = task.category
73        data["startTimeStamp"] = task.startTimeStamp
74        data["priority"] = task.priority
75        data["completed"] = false
76
77        // Add task to Tasks subcollection of user with a random id
78        db.collection(synap).document(uid) DocumentReference
79            .collection( collectionPath: "Tasks").document()
80            .set(data) Task<Void!>
81            .addOnSuccessListener { it: Void!
82                Log.d( tag: "dbfirebase",  msg: "save ${data}\n\n\n")
83            }
84            .addOnFailureListener { it: Exception
85                Log.d( tag: "dbfirebase",  msg: "FAIL ${data}\n\n\n")
86            }
87    }
```

```kotlin
/* Update Variables */
seanperry
public fun onTimerCompletion() {
    /*
    Handles all logic for communicating to firestore and updating variables
    Once a study timer session has finished.
     */

    // Increment that a study cycle has completed.
    db.collection(synap).document(uid)
        .collection( collectionPath: "General").document( documentPath: "CompletionCount")
        .update( field: "CompletionCount", FieldValue.increment( l: 1))

    // Todays Date
    val dateKey = LocalDate.now().toString()
    // Current time.
    val now = LocalDateTime.now().format(DateTimeFormatter.ofPattern( pattern: "HH:mm:ss"))

    // Add the timers completion time to the DB
    db.collection(synap).document(uid)
        .collection( collectionPath: "General").document( documentPath: "listOfCompletedSessions")
        .update(dateKey.toString(), FieldValue.arrayUnion(now))

    plots.refreshData()
}

seanperry
public fun getCompletionDates() : MutableMap<String, Any> {
    // Returns the completionDates map for access
    updateCompletionDates(completionDates)
    return completionDates
}
seanperry
public fun getTaskMap() : MutableMap<String, Task> {
    // Returns the task map, returning all tasks to the calendar
    getTasksAsMap(taskMap)
    return taskMap
}
seanperry
private fun resetTaskMap() {
    // Gets a fresh tasks map (used when task is deleted)
    taskMap = mutableMapOf<String, Task>()
    getTasksAsMap(taskMap)
}
```

```
130
        seanperry
131     public fun updateCompletionDates(dataMap : MutableMap<String, Any>) {
132         // Updates the dataMap to have the new completion dates
133         db.collection(synap).document(uid) DocumentReference
134             .collection( collectionPath: "General").document( documentPath: "listOfCompletedSessions")
135             .get() Task<DocumentSnapshot!>
136             .addOnSuccessListener { document ->
137                 if (document != null) {
138                     for (data in document.data?.toMutableMap()!!) {
139                         dataMap[data.key] = data.value
140                     }
141                 println(dataMap)
142                 } else {
143                     Log.d( tag: "dbfirebase",  msg: "No such document")
144                 }
145             }
146             .addOnFailureListener { exception ->
147                 Log.d( tag: "dbfirebase",  msg: "get failed with ", exception)
148             }
149     }
        seanperry
150     fun getTasksAsMap(tasks : MutableMap<String, Task>){
151         // Gets the task as a map for display on the calendar
152         db.collection(synap).document(uid) DocumentReference
153             .collection( collectionPath: "Tasks") CollectionReference
154             .get() Task<QuerySnapshot!>
155             .addOnSuccessListener { documents ->
156                 for (document in documents) {
157                     val task = Task(
158                         document.data["name"].toString(),
159                         document.data["description"].toString(),
160                         document.data["category"].toString(),
161                         document.data["startTimeStamp"] as Timestamp,
162                         document.data["priority"] as Long,
163                         document.data["completed"] as Boolean,
164                         document.id
165                     )
166                     tasks[document.id] = task
167                 }
168             }
169             .addOnFailureListener { exception ->
170                 Log.w( tag: "dbfirebase",  msg: "Error getting documents: ", exception)
171             }
172     }
173
```

```
174     fun getCompletionCount(completion: (Long) -> Unit) {
175         // Async call to get the completion count for trophies
176         db.collection(synap).document(uid) DocumentReference
177             .collection( collectionPath: "General").document( documentPath: "CompletionCount")
178             .get() Task<DocumentSnapshot!>
179             .addOnSuccessListener { document ->
180                 if (document != null) {
181                     val completionCount = document.getLong( field: "CompletionCount")
182                     completionCount?.let { it: Long
183                         completion(it)
184                     }
185                 } else {
186                     Log.d( tag: "dbfirebase",  msg: "No such document")
187                 }
188             }
189             .addOnFailureListener { exception ->
190                 Log.d( tag: "dbfirebase",  msg: "get failed with ", exception)
191             }
192     }
193
194     // Update existing task
        Jamison Coombs +1
195     fun updateTask (task: Task) {
196         // Used to update in task
197         val data: MutableMap<String, Any?> = HashMap()
198         data["name"] = task.name
199         data["description"] = task.description
200         data["category"] = task.category
201         data["startTimeStamp"] = task.startTimeStamp
202         data["priority"] = task.priority
203         data["completed"] = task.completed
204
205         db.collection(synap).document(uid) DocumentReference
206             .collection( collectionPath: "Tasks").document(task.getUID())
207             .update(data) Task<Void!>
208             .addOnSuccessListener { it: Void!
209                 Log.d( tag: "dbfirebase",  msg: "save ${data}\n\n\n")
210             }
211             .addOnFailureListener { it: Exception
212                 Log.d( tag: "dbfirebase",  msg: "FAIL ${data}\n\n\n")
213             }
214     }
215
216
        seanperry
217     public fun deleteTask(taskID: String) {
218         // Deletes a task from Firestore.
219         db.collection(synap).document(uid) DocumentReference
220             .collection( collectionPath: "Tasks").document(taskID)
221             .delete() Task<Void!>
222             .addOnSuccessListener { it: Void!
223                 Log.d( tag: "dbfirebase",  msg: "Task Deleted")
224                 resetTaskMap()
225             }
226             .addOnFailureListener { Log.d( tag: "dbfirebase",  msg: "Task failed to delete.") }
```

apflow

```
}

Jamison Coombs +1
fun markTaskCompleted(taskID: String) {
    // Marks a task parameter as complete
    val data: MutableMap<String, Any?> = HashMap()
    data["completed"] = true

    db.collection(synap).document(uid) DocumentReference
        .collection( collectionPath: "Tasks").document(taskID)
        .update(data) Task<Void!>
        .addOnSuccessListener { it: Void!
            Log.d( tag: "dbfirebase",  msg: "save ${data}\n\n\n")
        }
        .addOnFailureListener { it: Exception
            Log.d( tag: "dbfirebase",  msg: "FAIL ${data}\n\n\n")
        }
}

Jamison Coombs +1
fun undoTaskCompleted(taskID: String) {
    // Undos a task completion back to incomplete (Not used)
    val data: MutableMap<String, Any?> = HashMap()
    data["completed"] = false

    db.collection(synap).document(uid) DocumentReference
        .collection( collectionPath: "Tasks").document(taskID)
        .update(data) Task<Void!>
        .addOnSuccessListener { it: Void!
            Log.d( tag: "dbfirebase",  msg: "save ${data}\n\n\n")
        }
        .addOnFailureListener { it: Exception
            Log.d( tag: "dbfirebase",  msg: "FAIL ${data}\n\n\n")
        }
}

/* Get Data */
```

```
Jamison Coombs +1
fun getTask(uid: String) : Task {
    // Gets a single task
    var task = Task()
    db.collection(synap).document(uid) DocumentReference
        .collection( collectionPath: "Tasks").document(uid)
        .get() Task<DocumentSnapshot!>
        .addOnSuccessListener { document ->
            if (document != null) {
                task = Task(
                    document.data!!["name"].toString(),
                    document.data!!["description"].toString(),
                    document.data!!["category"].toString(),
                    document.data!!["startTimeStamp"] as Timestamp,
                    document.data!!["priority"] as Long,
                    document.data!!["completed"] as Boolean,
                    document.id
                )
                Log.d( tag: "dbfirebase",  msg: "No such document")
            } else {
                Log.d( tag: "dbfirebase",  msg: "No such document")
            }
        }
        .addOnFailureListener { exception ->
            Log.d( tag: "dbfirebase",  msg: "get failed with ", exception)
        }
    return task
}

/**
 * Get all tasks from user and return as a list of Task objects
 */
```

```
297     fun getTasks() : MutableList<Task> {
298         // Gets tasks as a list
299         val tasks = mutableListOf<Task>()
300
301         db.collection(synap).document(uid) DocumentReference
302             .collection( collectionPath: "Tasks") CollectionReference
303             .get() Task<QuerySnapshot!>
304             .addOnSuccessListener { documents ->
305                 for (document in documents) {
306                     val task = Task(
307                         document.data["name"].toString(),
308                         document.data["description"].toString(),
309                         document.data["category"].toString(),
310                         document.data["startTimeStamp"] as Timestamp,
311                         document.data["priority"] as Long,
312                         document.data["completed"] as Boolean,
313                         document.id
314                     )
315                     tasks.add(task)
316                 }
317             }
318             .addOnFailureListener { exception ->
319                 Log.w( tag: "dbfirebase",  msg: "Error getting documents: ", exception)
320             }
321         return tasks
322     }
323
324     /* User Creation */
325
        seanperry
326     private fun createUser(){
327         // Creates a user.
328         initGlobalVariables()
329     }
330
```

```
    seanperry
    private fun iterLoadDataset(collectionName : String, dataset: MutableMap<String, Any?>) {
        /*
            Reads in a dataset as a collections.document.collections with each dataset key
            Being the document for the file. Since there is no builtin function to do this
            We have to iterate over the dataset to add them individually so we'll see a schema like
            "SynapFlow(col)" -> "Uid(doc)" -> "Global(col)" -> "completedSessions(doc)" -> 0
            initialized in firebase.
            This is also optimal for having lists of objects.

         */
        for ((docName, docVal) in dataset) {
            db.collection(synap).document(uid) DocumentReference
                .collection(collectionName).document(docName)
                .set(hashMapOf(docName to docVal)) Task<Void!>
                .addOnSuccessListener { it: Void!
                    Log.d( tag: "dbfirebase",  msg: "save ${dataset}\n\n\n")
                    updateCompletionDates(completionDates)
                    getTaskMap()
                }
                .addOnFailureListener { it: Exception
                    Log.d( tag: "dbfirebase",  msg: "FAIL ${dataset}\n\n\n")
                }
        }
    }

    // Review adding data: https://firebase.google.com/docs/firestore/manage-data/add-data
    seanperry
    private fun initGlobalVariables(){
        /*
        Initializes the global/general variables we will be accessing accross the board, like
        Completed tasks and incremental values
         */
        val dataset: MutableMap<String, Any?> = HashMap()
        dataset["CompletionCount"] = 0
        dataset["listOfCompletedSessions"] = true
        iterLoadDataset( collectionName: "General", dataset)
    }

}
```

HomeActivity.kt

```kotlin
package com.hfad.synapflow

import ...
public val fsdb = FirestoreData()
public val plots = Figures()

Bradley T +2
class HomeActivity : AppCompatActivity() {

    private lateinit var fb: FirebaseAuth
    //val fsdb = FirestoreData()

    seanperry +2
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_home)

        title = "Synapflow"

        //Request user permissions upon accessing via Login/Skip Login
        permissionRequest()

        // Create notification channel
        createNotificationChannel()

        val calendarFragment = CalendarFragment()
        val analyticsFragment = analyticsMain()
        val rewardsFragment = Rewards()
        val timerFragment = TimerFragment()
        val bottomNav = findViewById<BottomNavigationView>(R.id.bottom_nav)

        fsdb.checkUserExists()
        if (fsdb.getUserExists()){
            fsdb.getTaskMap()
            plots.refreshData()
        }
        calendarFragment.getTasksForCalender()
        // Navbar navigation
        bottomNav.setOnItemSelectedListener {
            item -> when(item.itemId) {
                R.id.nav_home -> {
                     // Comment this out if you see it:
                    //fsdb.getCompletionDates()
                    changeFragment(timerFragment)
                    true ^setOnItemSelectedListener
                }
                R.id.nav_calendar -> {
                    changeFragment(calendarFragment)
                    true ^setOnItemSelectedListener
                }
                R.id.nav_analytics -> {
                    plots.refreshData()
                    changeFragment(analyticsFragment)
                    true ^setOnItemSelectedListener
                }
```

```
                }
                R.id.nav_rewards -> {
                    changeFragment(rewardsFragment)
                    true ^setOnItemSelectedListener
                }
                else -> false ^setOnItemSelectedListener
            }
        }
        // Go to add task fragment when add button is clicked
        bottomNav.findViewById<FloatingActionButton>(R.id.nav_add_item).setOnClickListener { it: View!
            bottomNav.isSelected = false
            openAddTaskFragment()
        }
    }
    Bradley T +1
    fun changeFragment(fragment: Fragment) {
        supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
            replace(R.id.fragmentContainerView, fragment)
            commit()
        }
    }

    Jamison Coombs
    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val name = "Synapflow"
            val descriptionText = "Synapflow Notifications"
            val importance = NotificationManager.IMPORTANCE_DEFAULT
            val channel = NotificationChannel( id: "C10", name, importance).apply { this: NotificationChannel
                description = descriptionText
            }
            val notificationManager: NotificationManager =
                getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
            notificationManager.createNotificationChannel(channel)
        }
    }


    // Separate function to open add task fragment so the fragment gets reset when returning
    Jamison Coombs
    private fun openAddTaskFragment() {
        val addTaskFragment = AddTaskFragment()
        // Check if fragment is running, if so then remove it before adding it again
        if(addTaskFragment.isAdded) {
            supportFragmentManager.beginTransaction().remove(addTaskFragment).commit()
        }
        supportFragmentManager.beginTransaction().apply { this: FragmentTransaction
            replace(R.id.fragmentContainerView, addTaskFragment)
            addToBackStack( name: null)
            commit()
        }
    }
```

```
    Bradley T
    private fun permissionRequest() {
        var permissionList = mutableListOf<String>()
        if (!(ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED)) {
            permissionList.add(Manifest.permission.ACCESS_COARSE_LOCATION)
        }
        if (!(ActivityCompat.checkSelfPermission( context: this, Manifest.permission.POST_NOTIFICATIONS) == PackageManager.PERMISSION_GRANTED)) {
            permissionList.add(Manifest.permission.POST_NOTIFICATIONS)
        }
        if (permissionList.isNotEmpty()) {
            ActivityCompat.requestPermissions( activity: this, permissionList.toTypedArray(),  requestCode: 0)
        }
    }
    //When user grants or denies permission, this function is called
    Bradley T
    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if (requestCode == 0 && grantResults.isNotEmpty()) {
            for (i in grantResults.indices) {
                if (grantResults[i] == PackageManager.PERMISSION_GRANTED) {
                    Log.d( tag: "MainActivityPerms",  msg: "${permissions[i]} granted.")
                }
            }
        }
    }
}
```

MainActivity.kt

```kotlin
package com.hfad.synapflow

import ...
// Used by firestore in case we skip the google login
var TESTMODE = false

seanperry +3
class MainActivity : AppCompatActivity() {

    private lateinit var auth : FirebaseAuth
    private lateinit var googleSignInClient : GoogleSignInClient

    seanperry +3
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        title = "Synapflow"

        auth = FirebaseAuth.getInstance()
        val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken("333016502251-lml7av23goontra8iikrqd0r1c620vhl.apps.goog...")
            .requestEmail()
            .build()

        googleSignInClient = GoogleSignIn.getClient( activity: this, gso)


        // Activate the googleAccount class
        findViewById<Button>(R.id.btnGoogSignIn).setOnClickListener { it: View!
            signInGoogle()
        }

        // Used to Skip the google sign in
        findViewById<Button>(R.id.skipbtnGoogSignIn).setOnClickListener { it: View!
            // Has to be an Intent to go to the HomeActivity otherwise the content view will always
            // be stuck on the setContentView of layout activity_home when going through this skip option
            // which is why attempting to load fragments from the bottom nav button menu were failing
            // to display
            TESTMODE = true
            //setContentView(R.layout.activity_home)
            Intent( packageContext: this, HomeActivity::class.java)
                .also { homeIntent ->
                    startActivity(homeIntent)
                }
        }
    }
```

```
    // Logic for Google Login
    private val launcher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
            result ->
        Log.d( tag: "dbfirebase", result.data.toString())
        if (result.resultCode == Activity.RESULT_OK) {
            val task = GoogleSignIn.getSignedInAccountFromIntent(result.data)
            handleResult(task)
        }
    }

    seanperry
    private fun handleResult(task: Task<GoogleSignInAccount>) {
        if (task.isSuccessful) {
            val account : GoogleSignInAccount? = task.result
            if (account != null) {
                updateUI(account)
            }
        } else {
            Toast.makeText( context: this, task.exception.toString(), Toast.LENGTH_SHORT).show()
        }
    }

    seanperry
    private fun updateUI(account: GoogleSignInAccount) {
        val credential = GoogleAuthProvider.getCredential(account.idToken,  accessToken: null)
        auth.signInWithCredential(credential).addOnCompleteListener { it: Task<AuthResult!>
            if (it.isSuccessful) {
                // Goes to the Home Activity
                val intent: Intent = Intent( packageContext: this, HomeActivity::class.java)
                startActivity(intent)
            } else {
                Toast.makeText( context: this, it.exception.toString(), Toast.LENGTH_SHORT).show()
            }
        }
    }

    seanperry
    private fun signInGoogle() {
        val signInIntent = googleSignInClient.signInIntent
        launcher.launch(signInIntent)
    }
}
```

Rewards.kt

```kotlin
package com.hfad.synapflow

import ...



trevork
class Rewards : Fragment() {
    trevork
    data class RewardItem(val icon: Int, val description: String)
    private val fd = FirestoreData()
    // create list to hold items in Horizontal scroll view
    private val rewardsToEarn = mutableListOf<RewardItem>()
    private val earnedRewards = mutableListOf<RewardItem>()

    // used to keep track if the lists have already been changed for number of completions
    private var didInit = false
    private var didInit1 = false
    private var didInit5 = false
    private var didInit10 = false
    private var didInit15 = false
    private var didInit20 = false
    private var didInit25 = false
    private var didInit30 = false
    private var didInit35 = false




    trevork
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        activity?.title = "Rewards | SynapFlow"
    }

    trevork
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_rewards, container, attachToRoot: false)
    }
    trevork
```

```
trevork
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    // get refrence to both Layouts
    val toEarnRewardsLinearLayout = view.findViewById<LinearLayout>(R.id.clRewardToEarn)
    val earnedRewardsLinearLayout = view.findViewById<LinearLayout>(R.id.clEarnedRewardItem)

    // remove what was created inside of layouts if there is already layouts in from before
    toEarnRewardsLinearLayout.removeAllViews()
    earnedRewardsLinearLayout.removeAllViews()

    // add trophies to screen
    populateRewards(earnedRewardsLinearLayout, earnedRewards)
    populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)

    //place all trophies in to earn section if not done already
    if(!didInit){
        initRewards()
        didInit = true
    }

    // get completion count from firebase
    checkCompletionCount { cc ->
        // if completion count from firebase is 1 or more
        if(cc >= 1){
            // if not done already
            if(!didInit1){
                // create reward Item to add in completed
                val rewardItem = RewardItem(R.drawable.trophy,  description: "Completed 1 Study Session")
                //add created reward in earned rewards list
                earnedRewards.add(rewardItem)
                // remove the reward corresponding with the one added to other list
                rewardsToEarn.removeAt( index: 0)

                //reset view
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                // mark done
                didInit1 = true
            }
        }
```

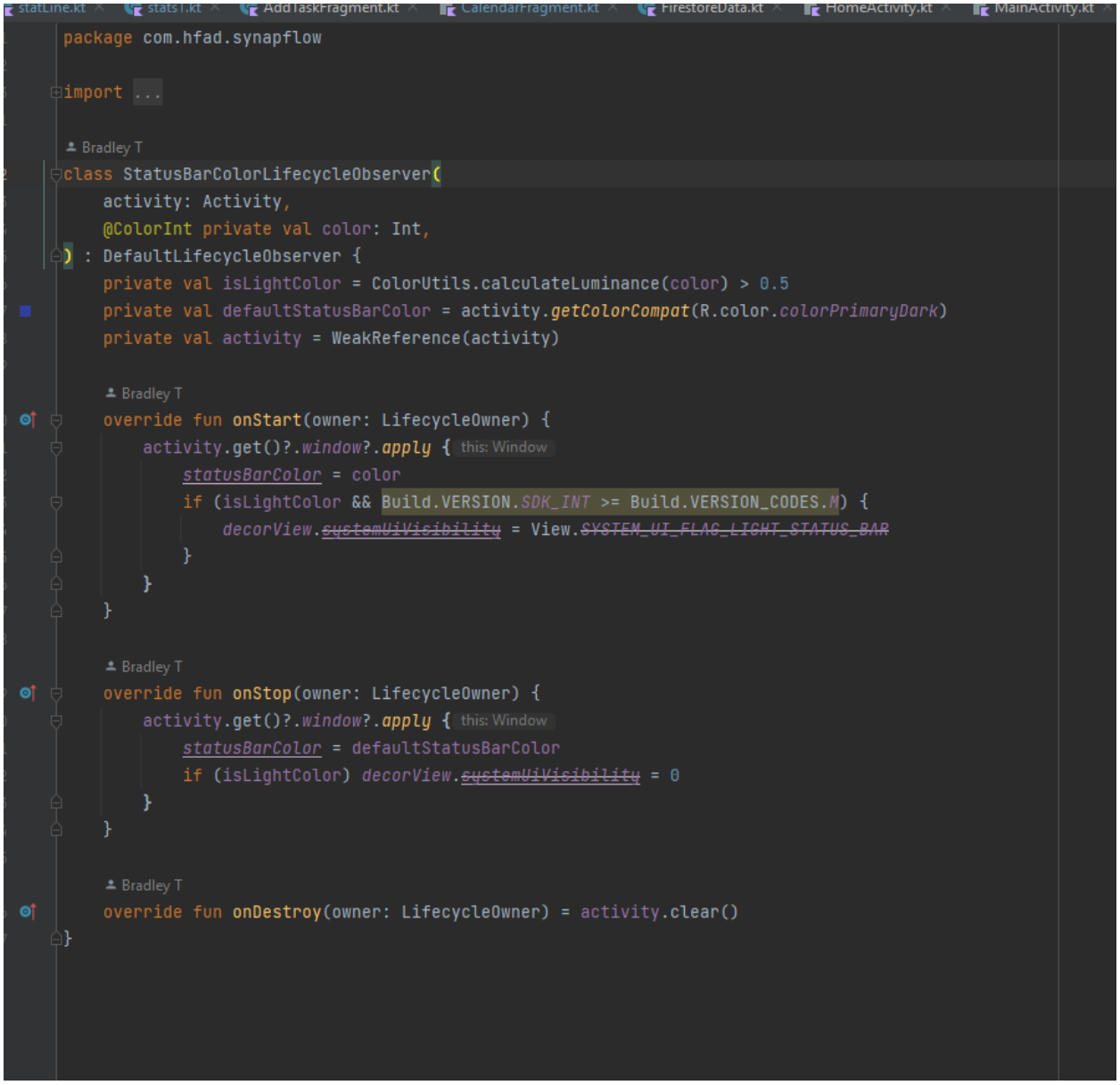
```
        if(cc >= 5){
            if(!didInit5){
                val rewardItem = RewardItem(R.drawable.trophy1,  description: "Completed 5 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit5 = true
            }
        }
        // same functionality as first if statement
        if(cc >= 10){
            if(!didInit10){
                val rewardItem = RewardItem(R.drawable.trophy2,  description: "Completed 10 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit10 = true
            }
        }
        // same functionality as first if statement
        if(cc >= 15){
            if(!didInit15){
                val rewardItem = RewardItem(R.drawable.trophy3,  description: "Completed 15 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit15 = true
            }
        }
        // same functionality as first if statement
        if(cc >= 20){
            if(!didInit20){
                val rewardItem = RewardItem(R.drawable.trophy4,  description: "Completed 20 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit20 = true
            }
```

```
        } // same functionality as first if statement
        if(cc >= 25){
            if(!didInit25){
                val rewardItem = RewardItem(R.drawable.trophy5,  description: "Completed 25 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit25 = true
            }
        } // same functionality as first if statement
        if(cc >= 30){
            if(!didInit30){
                val rewardItem = RewardItem(R.drawable.trophy6,  description: "Completed 30 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit30 = true
            }
        } // same functionality as first if statement
        if(cc >= 35){
            if(!didInit35){
                val rewardItem = RewardItem(R.drawable.trophy7,  description: "Completed 35 Study Session")
                rewardsToEarn.removeAt( index: 0)
                earnedRewards.add(rewardItem)
                toEarnRewardsLinearLayout.removeAllViews()
                earnedRewardsLinearLayout.removeAllViews()
                populateRewards(earnedRewardsLinearLayout, earnedRewards)
                populateRewards(toEarnRewardsLinearLayout, rewardsToEarn)
                didInit35 = true
            }
        }
    }

}
```

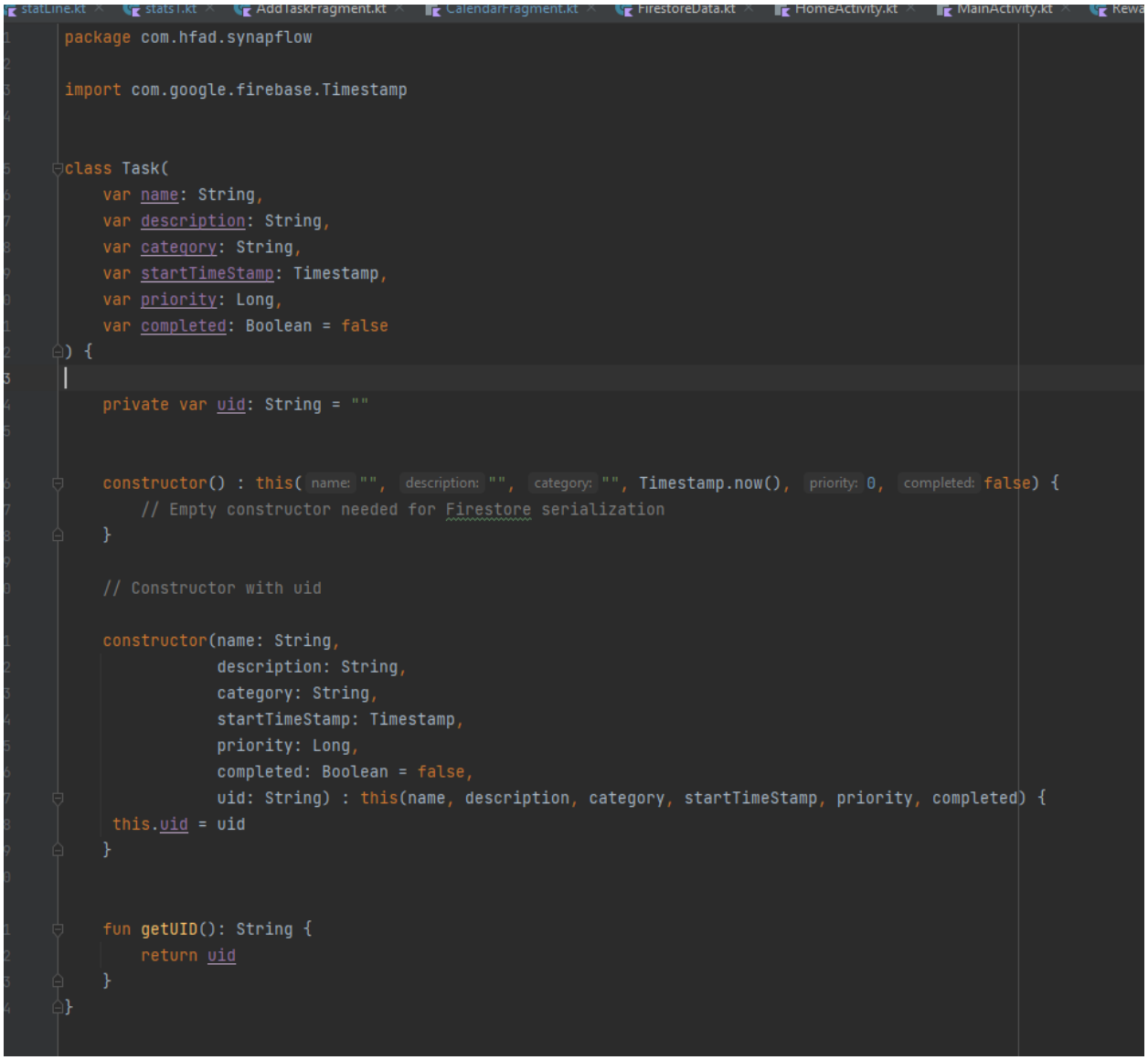
```
trevork
private fun populateRewards(RewardsLL: LinearLayout, rewardsList: List<RewardItem>) {
    //go through every created reward
    for (reward in rewardsList) {
        val displayMetrics = resources.displayMetrics
        val dpToPx = { dp: Int -> TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp.toFloat(), displayMetrics).toInt() }

        // Create ImageView to display the reward icon
        val iconImageView = ImageView(requireContext()).apply { this: ImageView
            // set parameters for layout
            layoutParams = ConstraintLayout.LayoutParams(dpToPx(64), dpToPx(53)).apply { this: ConstraintLayout.LayoutParams
                topToTop = ConstraintLayout.LayoutParams.PARENT_ID
                startToStart = ConstraintLayout.LayoutParams.PARENT_ID
                endToEnd = ConstraintLayout.LayoutParams.PARENT_ID
                //bottomToTop = R.id.earnedRewardsDescription
                horizontalBias = 0.5f
                verticalBias = 0.5f
                setMargins( left: 10, dpToPx(15),  right: 0, dpToPx(10))
            }
            adjustViewBounds = false
            cropToPadding = false
            scaleType = ImageView.ScaleType.CENTER_INSIDE

            setImageResource(reward.icon)
        }

        // create text view
        val descriptionTextView = TextView(context).apply { this: TextView
            // set parameters
            layoutParams = ConstraintLayout.LayoutParams(dpToPx(122), dpToPx(49)).apply { this: ConstraintLayout.LayoutParams
                bottomToBottom = ConstraintLayout.LayoutParams.PARENT_ID
                endToEnd = ConstraintLayout.LayoutParams.PARENT_ID
                startToStart = ConstraintLayout.LayoutParams.PARENT_ID
                topToTop = ConstraintLayout.LayoutParams.PARENT_ID
                verticalBias = 0.849f
            }
            gravity = Gravity.CENTER
            text = reward.description
        }
```

```
            // Add ImageView and TextView to LinearLayout with created parameters
            val rewardLayout = LinearLayout(requireContext())
            rewardLayout.orientation = LinearLayout.VERTICAL
            rewardLayout.gravity = Gravity.CENTER
            rewardLayout.addView(iconImageView)
            rewardLayout.addView(descriptionTextView)

            val layoutParams = LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT
            )
            layoutParams.marginEnd = 8dp
            RewardsLL.addView(rewardLayout, layoutParams)
        }

    }

    trevork
    private fun checkCompletionCount(completion: (Long) -> Unit) {
        // get completion count from firebase asynchronously
        fd.getCompletionCount { completionCount ->
            Log.d( tag: "Completion Count",  msg: "The current completion count is: $completionCount")
            completion(completionCount)
        }
    }

    trevork
    private fun initRewards(){
        // create all rewards for initial display when completion count is 0
        val rewardItem = RewardItem(R.drawable.trophy,  description: "Complete 1 Study Session")
        val rewardItem1 = RewardItem(R.drawable.trophy1,  description: "Complete 5 Study Session")
        val rewardItem2 = RewardItem(R.drawable.trophy2,  description: "Complete 10 Study Session")
        val rewardItem3 = RewardItem(R.drawable.trophy3,  description: "Complete 15 Study Session")
        val rewardItem4 = RewardItem(R.drawable.trophy4,  description: "Complete 20 Study Session")
        val rewardItem5 = RewardItem(R.drawable.trophy5,  description: "Complete 25 Study Session")
        val rewardItem6 = RewardItem(R.drawable.trophy6,  description: "Complete 30 Study Session")
        val rewardItem7 = RewardItem(R.drawable.trophy7,  description: "Complete 35 Study Session")
        rewardsToEarn.add(rewardItem)
        rewardsToEarn.add(rewardItem1)
        rewardsToEarn.add(rewardItem2)
        rewardsToEarn.add(rewardItem3)
        rewardsToEarn.add(rewardItem4)
        rewardsToEarn.add(rewardItem5)
        rewardsToEarn.add(rewardItem6)
        rewardsToEarn.add(rewardItem7)
    }
}
```

statusBarColorLifecycleObserver.kt


statLine.kt
stats1.kt
AddTaskFragment.kt
CalendarFragment.kt
FirestoreData.kt
HomeActivity.kt
MainActivity.kt
package com.hfad.synapflow
import ...
Bradley T
class StatusBarColorLifecycleObserver(
activity: Activity,
@ColorInt private val color: Int,
) : DefaultLifecycleObserver {
private val isLightColor = ColorUtils.calculateLuminance(color) > 0.5
private val defaultStatusBarColor = activity.getColorCompat(R.color.colorPrimaryDark)
private val activity = WeakReference(activity)
Bradley T
override fun onStart(owner: LifecycleOwner) {
activity.get()?.window?.apply { this: Window
statusBarColor = color
if (isLightColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
decorView.systemUiVisibility = View.SYSTEM_UI_FLAG_LIGHT_STATUS_BAR
}
}
}
Bradley T
override fun onStop(owner: LifecycleOwner) {
activity.get()?.window?.apply { this: Window
statusBarColor = defaultStatusBarColor
if (isLightColor) decorView.systemUiVisibility = 0
}
}
Bradley T
override fun onDestroy(owner: LifecycleOwner) = activity.clear()
}

Task.kt



TimerFragment.kt

```kotlin
package com.hfad.synapflow

import ...


seanperry +2
class TimerFragment : Fragment() {
    seanperry
    enum class TimerState {
        // Used to store the status of the states
        Stop, Pause, Play
    }

    //setting up viewModel
    private val viewModel : TimerViewModel by activityViewModels()
    private lateinit var timer: CountDownTimer
    private lateinit var myContext: Context

    private  lateinit var play : FloatingActionButton
    private  lateinit var pause : FloatingActionButton
    private  lateinit var stop : FloatingActionButton
    private lateinit var progress_countdown: MaterialProgressBar
    private  lateinit var  countdown: TextView
    private  lateinit var  title: TextView

    val fsdb = FirestoreData()
    //-----------------------------------------------------------------//

    seanperry +1
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        activity?.title = "Timer | Synapflow"
        }
    //-----------------------------------------------------------------//
    trevork +1
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        return inflater.inflate(R.layout.fragment_timer, container,  attachToRoot: false)
    }
    //-----------------------------------------------------------------//
```

```kotlin
    seanperry
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    myContext = view.context
    play = view.findViewById<FloatingActionButton>(R.id.fab_play)
    pause = view.findViewById<FloatingActionButton>(R.id.fab_pause)
    stop = view.findViewById<FloatingActionButton>(R.id.fab_stop)
    progress_countdown = view.findViewById<MaterialProgressBar>(R.id.progress_countdown)
    countdown = view.findViewById<TextView>(R.id.countdown)
    title = view.findViewById<TextView>(R.id.timerTitle)

    // Update UI
    updateButtons()
    updateCountdownUI()
    // If we have timer data, update it with this if the timer is still going.
    if ((viewModel.secondsRemaining.value!! > 0) and (viewModel.timerState.value!! == TimerFragment.TimerState.Play)) {
        val elapsedTime = (System.currentTimeMillis() - viewModel.lastCurrentTime.value!!) / 1000
        val newSecondsRemaining = viewModel.secondsRemaining.value!! - elapsedTime

        // If the timer finished while off the screen
        if (newSecondsRemaining <= 0) {
            viewModel.secondsRemaining.setValue(0)
            updateCountdownUI()
            onTimerFinished()
        } else { // Restart the timer with the new time remaining
            viewModel.secondsRemaining.setValue(newSecondsRemaining)
            viewModel.timer.setValue(object : android.os.CountDownTimer( millisInFuture: (viewModel.secondsRemaining.value?: 0) * 1000,  countDownInterval: 1000) {
                override fun onFinish() = onTimerFinished()
                override fun onTick(millisUntilFinished: kotlin.Long) {
                    viewModel.lastCurrentTime.setValue(System.currentTimeMillis())
                    viewModel.secondsRemaining.setValue(millisUntilFinished / 1000)
                    updateCountdownUI()
                }
            })
            viewModel.timer.value!!.start()
            progress_countdown.max = if (viewModel.onStudySession())  viewModel.studyTimerMax.value!!.toInt() else viewModel.breakTimerMax.value!!.toInt()
        }

    } else { // Otherwise reset the wheel to make sure it draws.
        progress_countdown.max = 1500
        progress_countdown.progress = 1499
    }

    // Initial setup
    play.setOnClickListener { it: View!
        viewModel.timerState.setValue(TimerFragment.TimerState.Play)
        startTimer()
```

```
        }
        pause.setOnClickListener { it: View!
            viewModel.timerState.setValue(TimerFragment.TimerState.Pause)
            updateButtons()
            viewModel.timer.value!!.cancel()
        }
        stop.setOnClickListener { it: View!
            viewModel.timerState.setValue(TimerFragment.TimerState.Stop)
            viewModel.timer.value!!.cancel()
            onTimerFinished()
        }
    }
    //-----------------------------------------------------------------//
    seanperry
    override fun onResume() {
        // Used on resuming the timer
        super.onResume()
        initTimer()
    }

    seanperry
    override fun onPause() {
        // Information for pausing the timer.
        super.onPause()
        if (viewModel.timerState.value ==  TimerFragment.TimerState.Play){ // If timer is playing cancel it
            viewModel.timer.value!!.cancel()
        } else if (viewModel.timerState.value == TimerFragment.TimerState.Pause) {  // If paused, reset the length
            PrefUtil.setPrevTimerLength((viewModel.timerLengthSeconds.value?: 0), myContext)
            PrefUtil.setTimeRemaining((viewModel.secondsRemaining.value?: 0), myContext)
        }
    }
    seanperry +1
    private fun initTimer() {
        // Initialized the timer
        if (viewModel.getTimerState() == TimerFragment.TimerState.Stop) { // If  the timers state is stopped do a study session
            setNewTimerLength()
        } else { // Otherwise set it with the previous timer length
            setPreviousTimerLength()
        }// Update the secondsremaining data.
        if (viewModel.getTimerState() == TimerFragment.TimerState.Play || viewModel.getTimerState() == TimerFragment.TimerState.Pause) {
            viewModel.secondsRemaining.setValue(getTimeRemaining(myContext))
        } else {
            viewModel.secondsRemaining = viewModel.timerLengthSeconds
        }
    }
```

```
seanperry + 1
private fun onTimerFinished() {
    // Logic for when the timer finishes
    //check if timer has ended
    if(viewModel.getTimerState() == TimerFragment.TimerState.Play && viewModel.secondsRemaining.value!! <= 0L){
        // Output a message
        Toast.makeText(context, text: "Time is up!", Toast.LENGTH_SHORT).show()

        // Update the study session count, only if its a finished study session
        if (viewModel.onStudySession())
            fsdb.onTimerCompletion()
        // Change between study/break session
        viewModel.updateSession()
    }
    viewModel.timerState.setValue(TimerState.Stop) // Turn off the timer
    setNewTimerLength() // Change the timer length
    progress_countdown.progress = 0 // Update UI wheel
    PrefUtil.setTimeRemaining((viewModel.timerLengthSeconds.value?: 0), myContext)
    viewModel.secondsRemaining = viewModel.timerLengthSeconds
    // Update UI
    updateButtons()
    updateCountdownUI()
}


seanperry
private fun startTimer() {
    // Creates our overriden timer
    viewModel.timerState.setValue( TimerFragment.TimerState.Play)
    viewModel.timer.setValue(object : CountDownTimer( millisInFuture: (viewModel.secondsRemaining.value?: 0) * 1000, countDownInterval: 1000) {
        override fun onFinish() = onTimerFinished() // Calls func on finish
        override fun onTick(millisUntilFinished: Long) { // Updates LiveModel Data
            viewModel.lastCurrentTime.setValue(System.currentTimeMillis())
            viewModel.secondsRemaining.setValue(millisUntilFinished / 1000)
            updateCountdownUI()
        }
    })
    viewModel.timer.value!!.start() // Starts the timer
    updateButtons() // Updates the UI
}


seanperry
private fun setNewTimerLength() {
    // Sets the logic for the new timer
    var lengthInMinutes : Long // Start a study session Timer
    if (viewModel.onStudySession())
        lengthInMinutes = PrefUtil.getTimerLength()
    else // Starts a break timer
        lengthInMinutes = PrefUtil.getBreakTimerLength()
    viewModel.timerLengthSeconds.setValue(lengthInMinutes * 60L)  // Updates the timer logic
    progress_countdown.max = (viewModel.timerLengthSeconds.value!!).toInt() // updates the timer wheel
}
```

```
seanperry
private fun setPreviousTimerLength() {
    // Updates prev timer
    viewModel.timerLengthSeconds.setValue(getPrevTimerLength(myContext))
}

seanperry
private fun updateCountdownUI() {
    // Updates the countdown UI data
    title.text = if (viewModel.onStudySession())  "Study Time!" else "Break Time!" // Updates the text on top
    countdown.text = viewModel.tSecondsRemainingStr() // updates timer
    progress_countdown.progress = viewModel.secondsRemaining.value!!.toInt() // updates countdown wheel data
}

seanperry +1
private fun updateButtons() {
    // Changes what buttons are active based on the state.
    when (viewModel.timerState.value) {
        TimerFragment.TimerState.Play -> {
            play.isEnabled  = false
            pause.isEnabled = true
            stop.isEnabled  = true
        }
        TimerFragment.TimerState.Pause -> {
            play.isEnabled  = true
            pause.isEnabled = false
            stop.isEnabled  = true
        }
        TimerFragment.TimerState.Stop -> {
            play.isEnabled  = true
            pause.isEnabled = false
            stop.isEnabled  = false
        }
        null -> {
            //Null parameter
        }
    }

}
```

```
230
231 }
232 //------------------------------------------------------------------//
    seanperry +1
233 class PrefUtil {
234     /*
235     Helper function for timer.
236      */
        seanperry
237     companion object {
238         // Id data
239         private const val PREV_TIMER_ID = "com.hfad.synapflow.timer.prev_timer_id"
240         private const val TIMER_STATE_ID = "com.hfad.synapflow.timer.state_id"
241         private const val TIME_REMAINING_ID = "com.hfad.synapflow.timer.remaining_id"
            seanperry
242         fun getTimerLength(): Long {
243             // Timer length for study
244             return 25
245         }
            seanperry
246         fun getBreakTimerLength(): Long {
247             // Timer length for break
248             return 5
249         }
250
            seanperry
251         fun getPrevTimerLength(context: Context): Long {
252             // gets prev timer length
253             val pref = PreferenceManager.getDefaultSharedPreferences(context)
254             return pref.getLong(PREV_TIMER_ID, 0)
255         }
            seanperry
256         fun setPrevTimerLength(time: Long, context: Context) {
257             // Sets prev timer lenght
258             val pref = PreferenceManager.getDefaultSharedPreferences(context).edit()
259             pref.putLong(PREV_TIMER_ID, time).apply()
260         }
            seanperry
261         fun getTimeRemaining(context: Context): Long {
262             // Gets the  timer length
263             val pref = PreferenceManager.getDefaultSharedPreferences(context)
264             return pref.getLong(TIME_REMAINING_ID, 0)
265         }
            seanperry
266         fun setTimeRemaining(time: Long, context: Context) {
267             // Sets the timer length
268             val pref = PreferenceManager.getDefaultSharedPreferences(context).edit()
269             pref.putLong(TIME_REMAINING_ID, time).apply()
270         }
271     }
272 }
273
```

TimerViewModel.kt

```kotlin
package com.hfad.synapflow
import android.os.CountDownTimer
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel


seanperry
class TimerViewModel :ViewModel() {
    /*
        Live data model for saving data in the timer while accessing other elements.
     */
    var studyTimerMax = MutableLiveData<Long>( value: PrefUtil.getTimerLength() * 60)
    var breakTimerMax = MutableLiveData<Long>( value: PrefUtil.getBreakTimerLength() * 60)
    var timerLengthSeconds = MutableLiveData<Long>( value: 0)
    var timerState = MutableLiveData<TimerFragment.TimerState>(TimerFragment.TimerState.Stop)
    var secondsRemaining = MutableLiveData<Long>(studyTimerMax.value!!)
    var secondsRemainingStr = MutableLiveData<String>()
    var studySessionState = MutableLiveData<Boolean>( value: true)
    var lastCurrentTime = MutableLiveData<Long>()
    var timer = MutableLiveData<CountDownTimer>()


    seanperry
    fun updateSession() {
        // Flips thethe session to true or false based on what it previously was
        if (studySessionState.value!!)
            studySessionState.setValue(false)
        else
            studySessionState.setValue(true)
    }

    seanperry
    fun onStudySession() : Boolean {
        // Returns if the study session is on or if its break
        if (studySessionState.value?: true)
            return true
        return false
    }

    seanperry
    fun tSecondsRemainingStr() : String? {
        // Returns the formatted time string. To be a timer like 24:59
        val minutesUntilFinished = (secondsRemaining.value?: 0) / 60
        val secondsInMinutesUntilFinished = (secondsRemaining.value?: 0) - minutesUntilFinished * 60
        val secondsStr = secondsInMinutesUntilFinished.toString()
        var newStr = "$minutesUntilFinished:${ // Adds a 0 in the case otherwise 24:50 might be 24:5
            if (secondsStr.length == 2) secondsStr
            else "0"+ secondsStr}"
        secondsRemainingStr.setValue(newStr)
        return secondsRemainingStr.value?.toString()
    }

    seanperry
    fun getTimerState(): TimerFragment.TimerState? {
        // Gets the state of the timer.
        return timerState.value
    }
}
```

Utils.kt

CalendarFragment.kt | FirestoreData.kt | HomeActivity.kt | MainActivity.kt | Rewards.kt | StatusBarColorLif

```kotlin
package com.hfad.synapflow

import ...

Bradley T
fun View.makeVisible() {
    visibility = View.VISIBLE
}

Bradley T
fun View.makeInVisible() {
    visibility = View.INVISIBLE
}

Bradley T
fun View.makeGone() {
    visibility = View.GONE
}

Bradley T
fun dpToPx(dp: Int, context: Context): Int =
    TypedValue.applyDimension(
        TypedValue.COMPLEX_UNIT_DIP,
        dp.toFloat(),
        context.resources.displayMetrics,
    ).toInt()

internal val Context.layoutInflater: LayoutInflater
    get() = LayoutInflater.from( context: this)

internal val Context.inputMethodManager
    get() = this.getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager

Bradley T
internal fun Context.getDrawableCompat(@DrawableRes drawable: Int): Drawable =
    requireNotNull(ContextCompat.getDrawable( context: this, drawable))

Bradley T
internal fun Context.getColorCompat(@ColorRes color: Int) =
    ContextCompat.getColor( context: this, color)

Bradley T
internal fun TextView.setTextColorRes(@ColorRes color: Int) =
    setTextColor(context.getColorCompat(color))

Bradley T
fun Fragment.addStatusBarColorUpdate(@ColorRes colorRes: Int) {
    view?.findViewTreeLifecycleOwner()?.lifecycle?.addObserver(
        StatusBarColorLifecycleObserver(
            requireActivity(),
            requireContext().getColorCompat(colorRes),
        ),
    )
}
```

Drawable XMLS

Activity_home.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeActivity">

    <!--
        android:name="com.hfad.synapflow.HomePage"
        android:name="com.hfad.synapflow.analytics.analyticsMain"
    -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:weightSum="100">

        <androidx.fragment.app.FragmentContainerView
            android:id="@+id/fragmentContainerView"
            android:name="com.hfad.synapflow.TimerFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:layout_constraintBottom_toTopOf="@+id/bottom_nav"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:layout="@layout/fragment_home_page"
            android:layout_weight="10" />


            <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_nav"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginBottom="2dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintVertical_bias="1.0"
            app:menu="@menu/bottom_nav_menu"
            android:layout_weight="90">

            <com.google.android.material.floatingactionbutton.FloatingActionButton
                android:id="@+id/nav_add_item"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:backgroundTint="@color/Teal"
                android:clickable="true"
                app:srcCompat="@android:drawable/ic_input_add"
                app:tint="@color/white"
                tools:ignore="SpeakableTextPresentCheck" />
        </com.google.android.material.bottomnavigation.BottomNavigationView>

        -->

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >


    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="match_parent"
        android:layout_height="401dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/newlogo" />

    <Button
        android:id="@+id/btnGoogSignIn"
        android:layout_width="267dp"
        android:layout_height="46dp"
        android:layout_marginTop="80dp"
        android:text="Google Signin"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView2" />

    <Button
        android:id="@+id/skipbtnGoogSignIn"
        android:layout_width="168dp"
        android:layout_height="41dp"
        android:layout_marginTop="60dp"
        android:text="Skip Google Signin"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnGoogSignIn" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

calendar_day_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="2dp">

    <TextView
        android:id="@+id/calendarDayText"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:layout_margin="8dp"
        android:gravity="center"
        android:textColor="@color/black"
        android:textSize="16sp"
        tools:text="22" />

    <View
        android:id="@+id/calendarDotView"
        android:layout_width="4.5dp"
        android:layout_height="4.5dp"
        android:layout_marginBottom="10dp"
        android:layout_gravity="bottom|center_horizontal"
        android:background="@drawable/calendar_today_bg" />

</FrameLayout>
```

calendar_day_legend_container.xml

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:weightSum="7"
    android:orientation="horizontal"
    android:background="#EEEEEE"
    android:paddingTop="12dp"
    android:paddingBottom="12dp">

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

    <include layout="@layout/calendar_day_legend_text" />

</LinearLayout>
```

calendar_day_legend_test.xml

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/legendText1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="center"
    android:textColor="@color/black"
    android:textSize="14sp"
    android:textStyle="bold"
    android:textAllCaps="true"
    tools:text="SUN" />
```

calendar_event_item_view.xml

```xml
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="20dp"
    android:background="?attr/selectableItemBackground"
    xmlns:tools="http://schemas.android.com/tools">


    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:cardCornerRadius="8dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/cardTitle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Title"
                android:textSize="18sp"
                android:textStyle="bold"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.025"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/description"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginVertical="8dp"
                android:layout_marginEnd="8dp"
                android:maxLines="2"
                android:textSize="12sp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.0"
                app:layout_constraintStart_toStartOf="@id/cardTitle"
                app:layout_constraintTop_toBottomOf="@id/cardTitle"
                app:layout_constraintVertical_bias="0.0"
                tools:text="@tools:sample/lorem/random" />

```

```
            <TextView
                android:id="@+id/time"
                android:layout_width="123dp"
                android:layout_height="15dp"
                android:layout_marginVertical="8dp"
                android:maxLines="1"
                android:textSize="12sp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@+id/completed"
                app:layout_constraintHorizontal_bias="0.0"
                app:layout_constraintStart_toStartOf="@id/description"
                app:layout_constraintTop_toBottomOf="@id/description"
                app:layout_constraintVertical_bias="0.421"
                tools:text="At 8:00am" />

            <TextView
                android:id="@+id/completed"
                android:layout_width="217dp"
                android:layout_height="15dp"
                android:layout_marginVertical="8dp"
                android:maxLines="1"
                android:textSize="12sp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.948"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/description"
                tools:text="Complete" />
        </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
    <!--

        <TextView
        android:id="@+id/itemEventText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        tools:text="Event Reminder Test 1" />
    -->

</FrameLayout>
```

fragment_add_task.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".AddTaskFragment">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center|center_horizontal"
        android:orientation="vertical">



<!--        <TextView-->
<!--            android:id="@+id/title_text"-->
<!--            android:layout_width="wrap_content"-->
<!--            android:layout_height="wrap_content"-->
<!--            android:layout_marginTop="16dp"-->
<!--            android:layout_marginBottom="8dp"-->
<!--            android:text="Add Task"-->
<!--            android:textAppearance="@style/TextAppearance.AppCompat.Display1" />-->

        <androidx.cardview.widget.CardView
            android:id="@+id/card_view"
            android:layout_gravity="center"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            app:cardElevation="4dp"
            app:cardUseCompatPadding="true"
            app:cardCornerRadius="24dp">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginTop="16dp"

                android:layout_marginBottom="16dp"
                android:orientation="vertical">

                <EditText
                    android:id="@+id/name_txt"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:ems="10"
                    android:hint="Name"
                    android:inputType="text"
                    android:minHeight="48dp"
                    android:textAlignment="center" />
```

```
52
53              <EditText
54                  android:id="@+id/description_txt"
55                  android:layout_width="match_parent"
56                  android:layout_height="wrap_content"
57                  android:ems="10"
58                  android:gravity="center|top"
59                  android:hint="Description"
60                  android:inputType="textMultiLine"
61                  android:minHeight="48dp" />
62          </LinearLayout>
63      </androidx.cardview.widget.CardView>
64
65
66      <androidx.cardview.widget.CardView
67          android:layout_gravity="center"
68          android:layout_width="match_parent"
69          android:layout_height="wrap_content"
70          android:layout_marginBottom="16dp"
71          app:cardUseCompatPadding="true"
72          app:cardElevation="4dp"
73          app:cardCornerRadius="24dp">
74      <LinearLayout
75          android:layout_width="match_parent"
76          android:layout_height="match_parent"
77          android:padding="8dp"
78          android:orientation="vertical">
79
80          <TextView
81              android:id="@+id/spinner_label"
82              android:layout_width="match_parent"
83              android:layout_height="wrap_content"
84              android:layout_marginBottom="8dp"
85              android:text="Category"
86              android:textAlignment="center"
87              android:textSize="18sp" />
88
89          <Spinner
90              android:id="@+id/category_spinner"
91              android:layout_width="match_parent"
92              android:layout_height="wrap_content"
93              android:minHeight="48dp"
94              android:spinnerMode="dropdown"
95              android:textAlignment="center" />
96
97      </LinearLayout>
98      </androidx.cardview.widget.CardView>
```

```
    <androidx.cardview.widget.CardView
        android:layout_gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        app:cardUseCompatPadding="true"
        app:cardCornerRadius="24dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="end"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
            android:gravity="end"
            android:layout_width="0dp"
            android:layout_weight="20"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:text="Start:"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/startDateEditText"
            android:layout_width="0dp"
            android:layout_weight="45"
            android:layout_height="wrap_content"
            android:focusable="false"
            android:drawableEnd="@drawable/ic_calendar"
            android:drawableTint="@color/purple_700"
            android:hint="Date"
            android:inputType="none"
            android:minHeight="48dp" />
```

```
    </LinearLayout>
    </androidx.cardview.widget.CardView>

    <!--...-->

    <androidx.cardview.widget.CardView
        android:layout_gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        app:cardUseCompatPadding="true"
        app:cardCornerRadius="24dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="8dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:gravity="center"
            android:text="Priority"
            android:textSize="18sp" />

        <SeekBar
            android:id="@+id/priority_seekbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:max="4"
            android:progress="2" />

        <TextView
            android:id="@+id/priorityLabel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Medium"
            android:textAlignment="center" />

    </LinearLayout>
```

```
        </androidx.cardview.widget.CardView>

        <Button
            android:id="@+id/addTaskButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginStart="100dp"
            android:layout_marginEnd="100dp"
            android:background="@drawable/customborder"
            android:text="Add Task" />

        <!--            android:thumb="@drawable/ic_seek_thumb"-->
        <!--            android:progressDrawable="@drawable/seekbar_progress" -->

    </LinearLayout>

</ScrollView>
```

fragments_analytics_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/bgcard"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    tools:context=".analytics.analyticsMain">


    <androidx.cardview.widget.CardView
        android:id="@+id/Title1"
        android:layout_width="400dp"
        android:layout_height="30dp"
        android:layout_marginTop="4dp"

        app:cardBackgroundColor="@color/Teal"
        app:cardCornerRadius="10dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.545"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView

            android:id="@+id/textTitle1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Last Months Sessions Count" />
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/Title2"
        android:layout_width="400dp"
        android:layout_height="30dp"
        android:layout_marginTop="8dp"
        app:cardBackgroundColor="@color/Teal"
        app:cardCornerRadius="10dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.545"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fragmentContainerView">
```

```
        <TextView
            android:id="@+id/textTitle2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Last Weeks Completed Study Sessions" />
    </androidx.cardview.widget.CardView>

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView"
        android:name="com.hfad.synapflow.analytics.statLine"
        android:layout_width="375dp"
        android:layout_height="250dp"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/Title1"
        tools:layout="@layout/fragment_stats1" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView2"
        android:name="com.hfad.synapflow.analytics.stats1"

        android:layout_width="355dp"
        android:layout_height="237dp"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.49"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/Title2"
        app:layout_constraintVertical_bias="0.0"
        tools:layout="@layout/fragment_stats1" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_calendar.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/white"
    android:clickable="true"
    android:focusable="true"
    tools:context=".CalendarFragment">

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <include
                android:id="@+id/titlesContainer"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                layout="@layout/calendar_day_legend_container" />

            <com.kizitonwose.calendar.view.CalendarView
                android:id="@+id/calendarView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:translationY="6dp"
                app:cv_dayViewResource="@layout/calendar_day_layout" />
```

```
            <TextView
                android:id="@+id/currentDateText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:background="#EEEEEE"
                android:fontFamily="sans-serif-medium"
                android:paddingStart="20dp"
                android:paddingTop="12dp"
                android:paddingEnd="20dp"
                android:paddingBottom="12dp"
                android:textAllCaps="true"
                android:textColor="@color/black"
                android:textSize="14sp"
                tools:text="20 April, 2023" />

            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/calendarRv"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:nestedScrollingEnabled="false" />

        </LinearLayout>

    </androidx.core.widget.NestedScrollView>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/eventAddButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:layout_margin="20dp"
        android:translationY="-60dp"
        android:visibility="gone"
        app:backgroundTint="@color/Teal"
        app:srcCompat="@drawable/nav_event_add"
        app:tint="@color/white"
        tools:ignore="SpeakableTextPresentCheck" />

</FrameLayout>
```

fragment_home_page.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomePage">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="hello_home_page" />

</FrameLayout>
```

fragments_rewards.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Rewards">


    <TextView
        android:id="@+id/tvRewardsFragmentTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="147dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="147dp"
        android:layout_marginBottom="71dp"
        android:text="Rewards"
        android:textSize="30sp"
        app:layout_constraintBottom_toTopOf="@+id/tvEarnedRewards"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvEarnedRewards"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="156dp"
        android:layout_marginTop="180dp"
        android:layout_marginEnd="153dp"
        android:layout_marginBottom="551dp"
        android:text="Earned Rewards"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

```

```
    <TextView
        android:id="@+id/tvToEarnRewards"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="153dp"
        android:layout_marginTop="390dp"
        android:layout_marginEnd="152dp"
        android:layout_marginBottom="322dp"
        android:text="Rewards To Earn"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <HorizontalScrollView
        android:id="@+id/hsvEarnedRewards"
        android:layout_width="393dp"
        android:layout_height="175dp"
        android:layout_marginStart="9dp"
        android:layout_marginTop="197dp"
        android:layout_marginEnd="9dp"
        android:layout_marginBottom="359dp"
        android:background="@color/ap_gray"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

            <LinearLayout
                android:id="@+id/clEarnedRewardItem"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_margin="10dp"
                android:gravity="center_horizontal"
                android:orientation="horizontal"
                android:padding="10dp">

            </LinearLayout>


    </HorizontalScrollView>
```

```
    <HorizontalScrollView
        android:id="@+id/hsvToEarnRewards"
        android:layout_width="393dp"
        android:layout_height="175dp"
        android:layout_marginStart="9dp"
        android:layout_marginTop="430dp"
        android:layout_marginEnd="9dp"
        android:layout_marginBottom="126dp"
        android:background="@color/ap_gray"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">


    <LinearLayout
            android:id="@+id/clRewardToEarn"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_margin="10dp"
            android:gravity="center_horizontal"
            android:orientation="horizontal"
            android:padding="10dp">

    </LinearLayout>

    </HorizontalScrollView>


</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_stat_line.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:ap="http://schemas.android.com/apk/res-auto"
    tools:context=".HomeActivity">


    <com.github.mikephil.charting.charts.BarChart
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/plot2">
    </com.github.mikephil.charting.charts.BarChart>

</FrameLayout>
```

fragment_stats1.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:ap="http://schemas.android.com/apk/res-auto"
    tools:context=".HomeActivity">


    <com.github.mikephil.charting.charts.LineChart
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/plot1">
    </com.github.mikephil.charting.charts.LineChart>

</FrameLayout>
```

fragment_timer.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".TimerFragment">

        <!-- TODO: Update blank fragment layout -->

        <androidx.cardview.widget.CardView
            android:layout_width="300dp"
            android:layout_height="70dp"
            app:cardBackgroundColor="@color/Teal"
            app:cardCornerRadius="20dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.495"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.08">

            <TextView
                android:id="@+id/timerTitle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Study Time!"
                android:textSize="40sp"
                android:textStyle="bold"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.025"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />


        </androidx.cardview.widget.CardView>
```

```
    <TextView
        android:id="@+id/countdown"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="70sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="25:00"
        />
    <!--
    tools:text="10:00"
    -->

    <me.zhanghai.android.materialprogressbar.MaterialProgressBar
        android:id="@+id/progress_countdown"
        style="@style/Widget.MaterialProgressBar.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minWidth="310dp"
        android:minHeight="310dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints" />
        <!--


        -->
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_stop"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="64dp"
    android:clickable="true"
    android:src="@drawable/ic_stop"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/fab_play"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/countdown"
    tools:ignore="SpeakableTextPresentCheck" />

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_play"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="64dp"
    android:clickable="true"
    android:src="@drawable/ic_play"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/countdown"
    tools:ignore="MissingConstraints,SpeakableTextPresentCheck" />

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_pause"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="64dp"
    android:clickable="true"
    android:src="@drawable/ic_pause"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/fab_play"
    app:layout_constraintTop_toBottomOf="@+id/countdown"
    tools:ignore="SpeakableTextPresentCheck" />
```

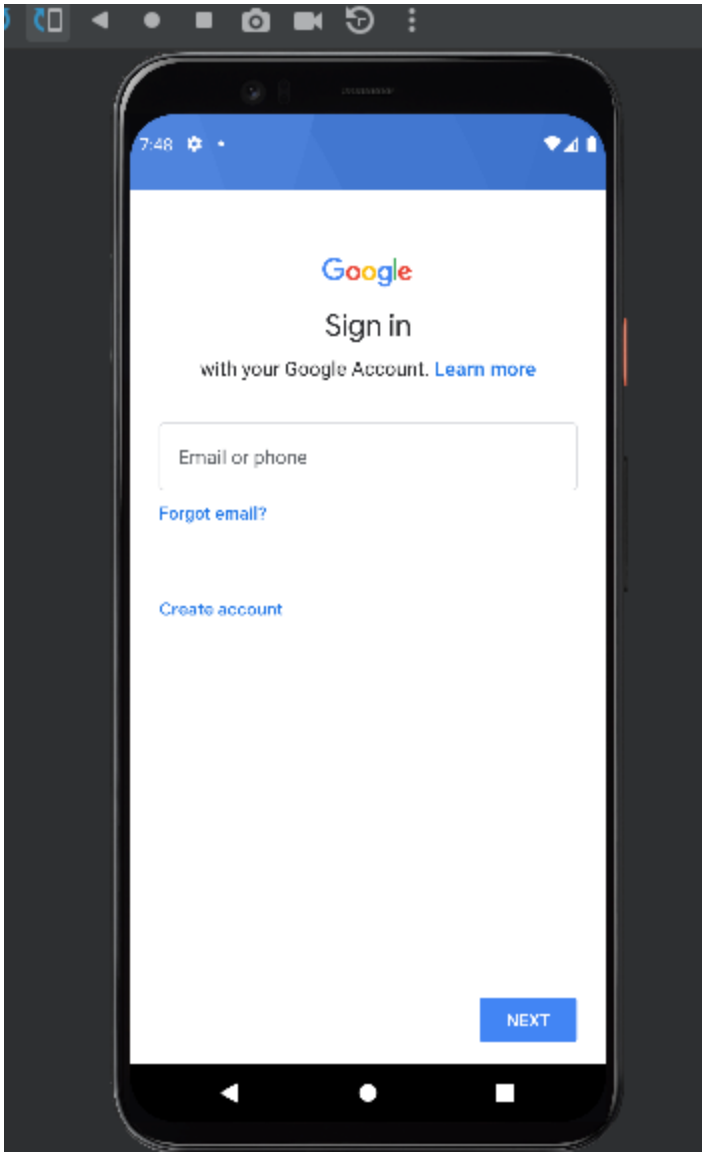
```
        <!--...-->

        />


    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

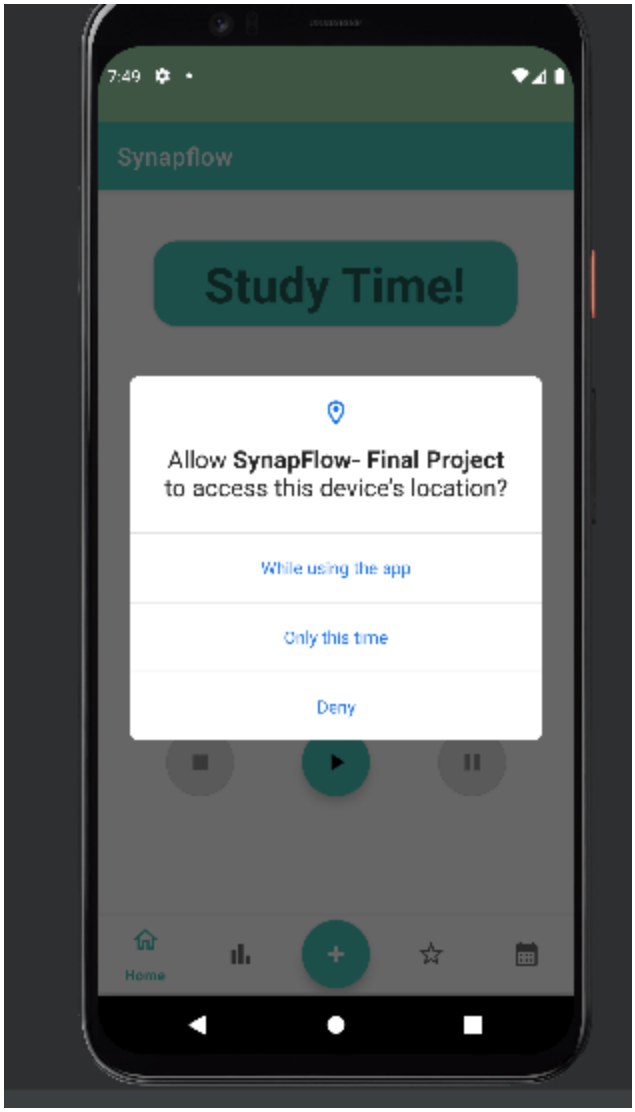
rewards_view_layout.xml

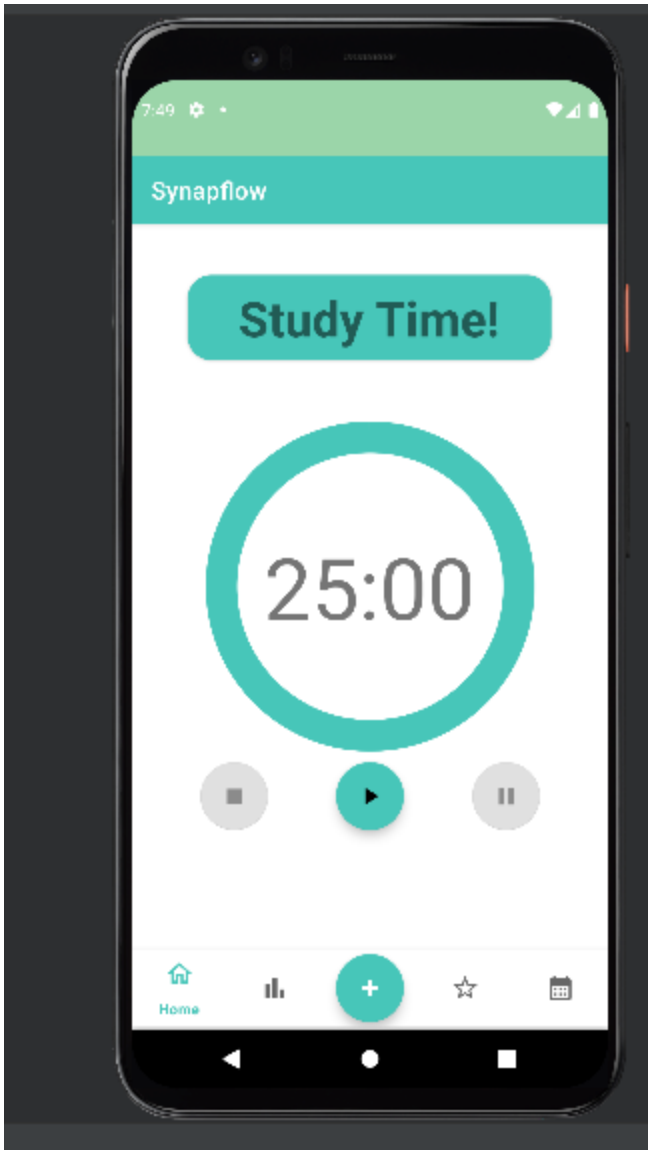
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:padding="16dp">

    <ImageView
        android:id="@+id/iconImage"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:src="@drawable/trophy"/>


    <TextView
        android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reward Description"/>

</LinearLayout>
```

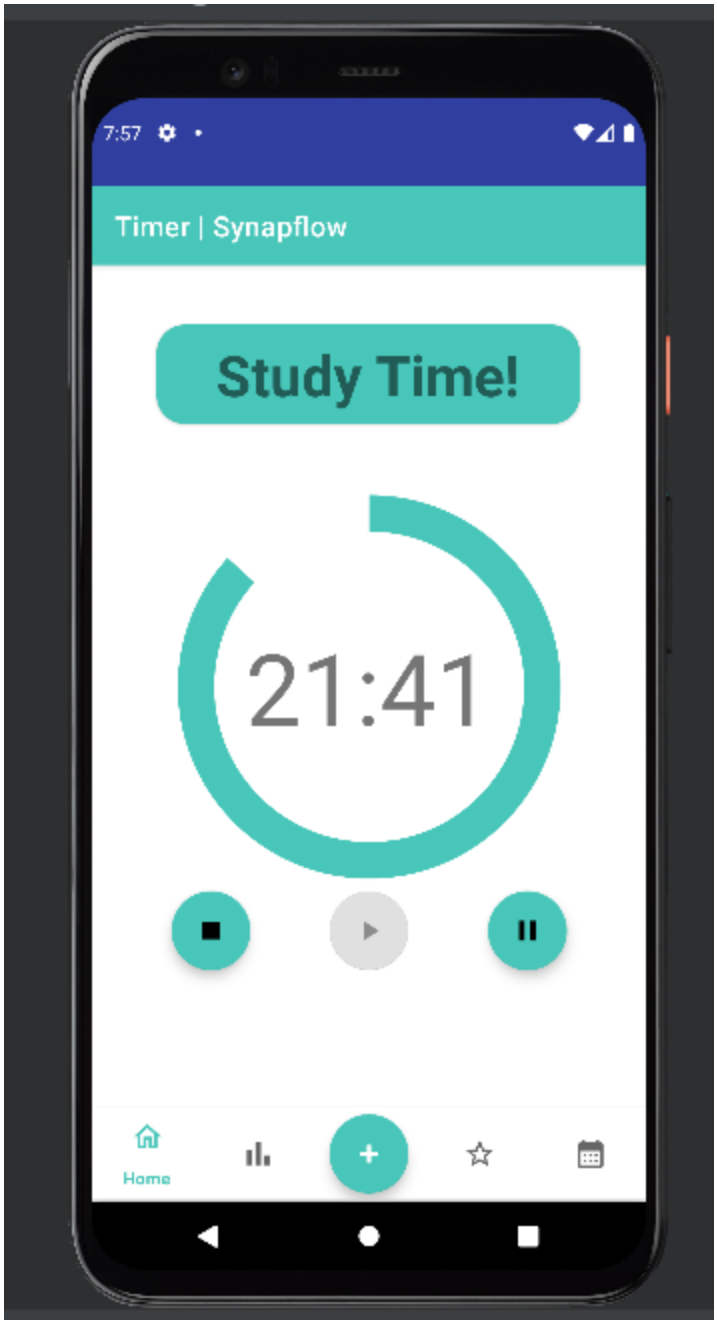
App Screenshots

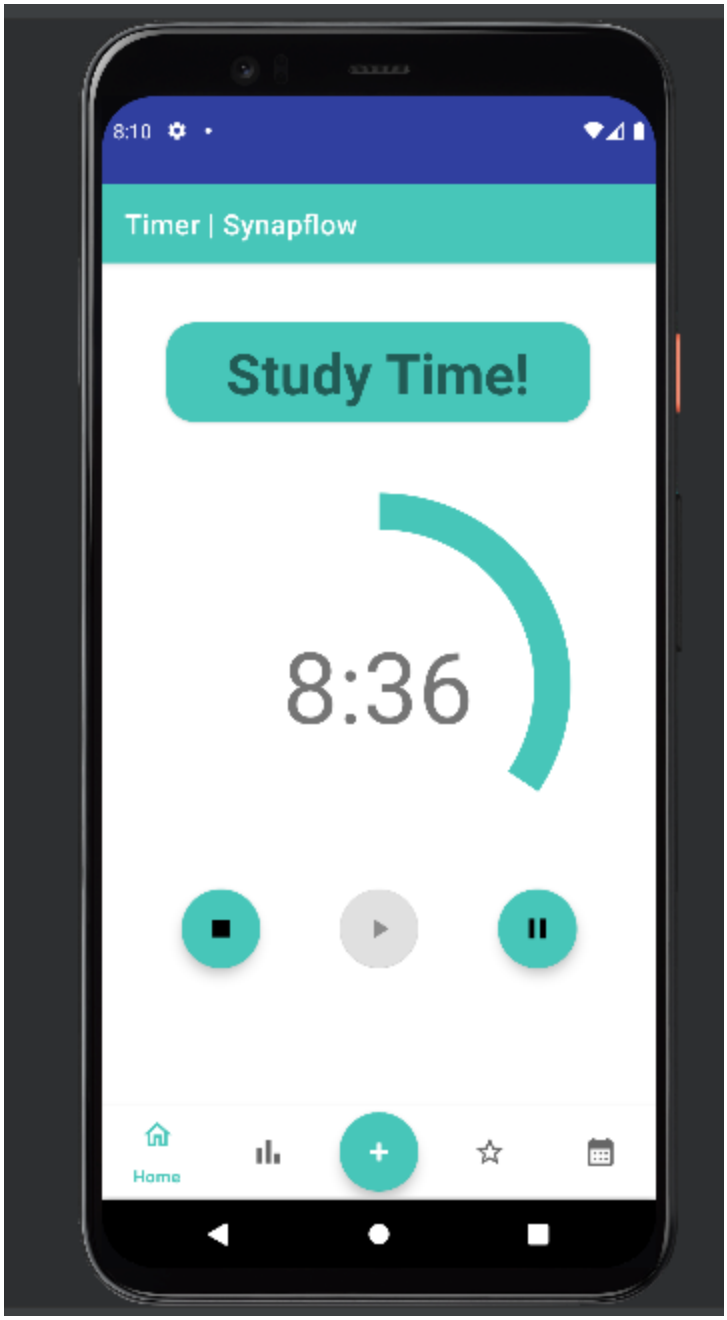

7:47
Synapflow
SynapFlow
GOOGLE SIGNIN
SKIP GOOGLE

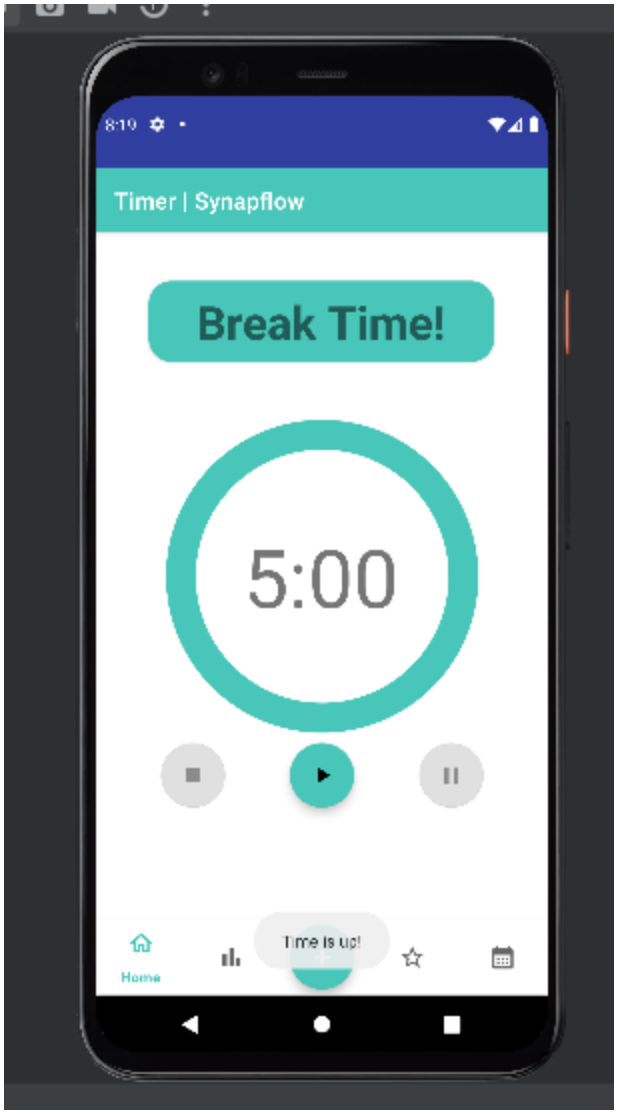

7:48
Google
Sign in
with your Google Account. Learn more
Email or phone
Forgot email?
Create account
NEXT

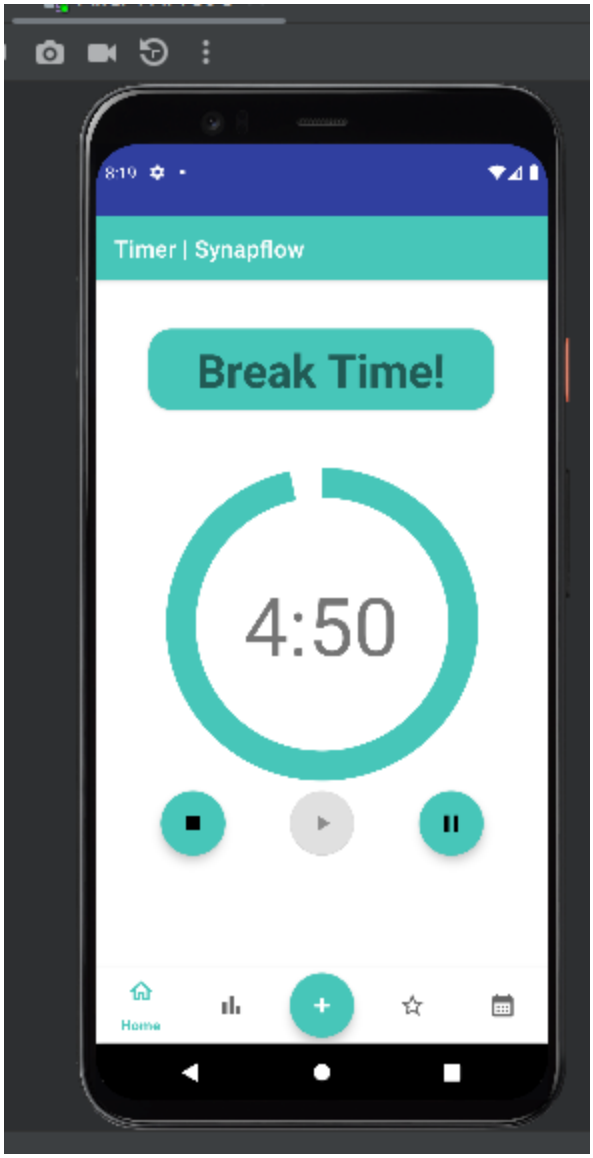

7:49
Synapflow
Study Time!
Allow SynapFlow- Final Project to access this device's location?
While using the app
Only this time
Deny
Home

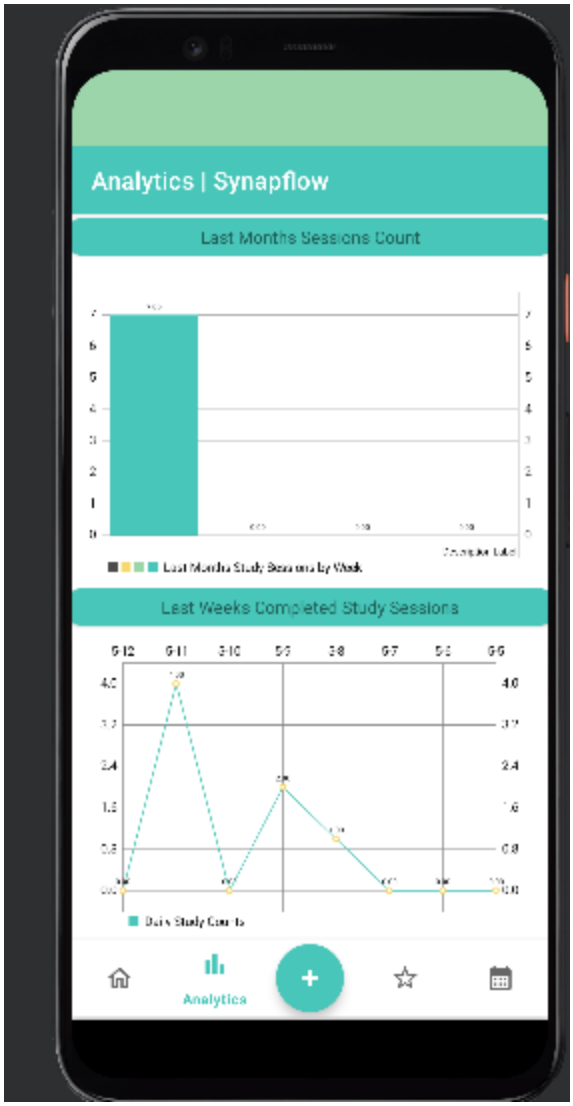

7:49
Synapflow
Study Time!
25:00
Home

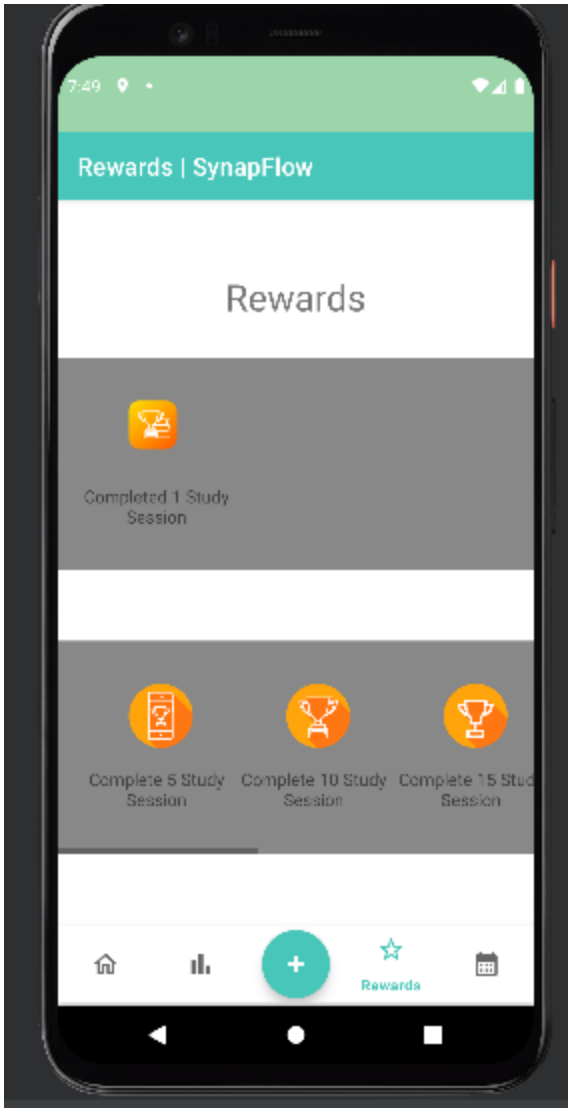

7:57
Timer | Synapflow
Study Time!
21:41
Home

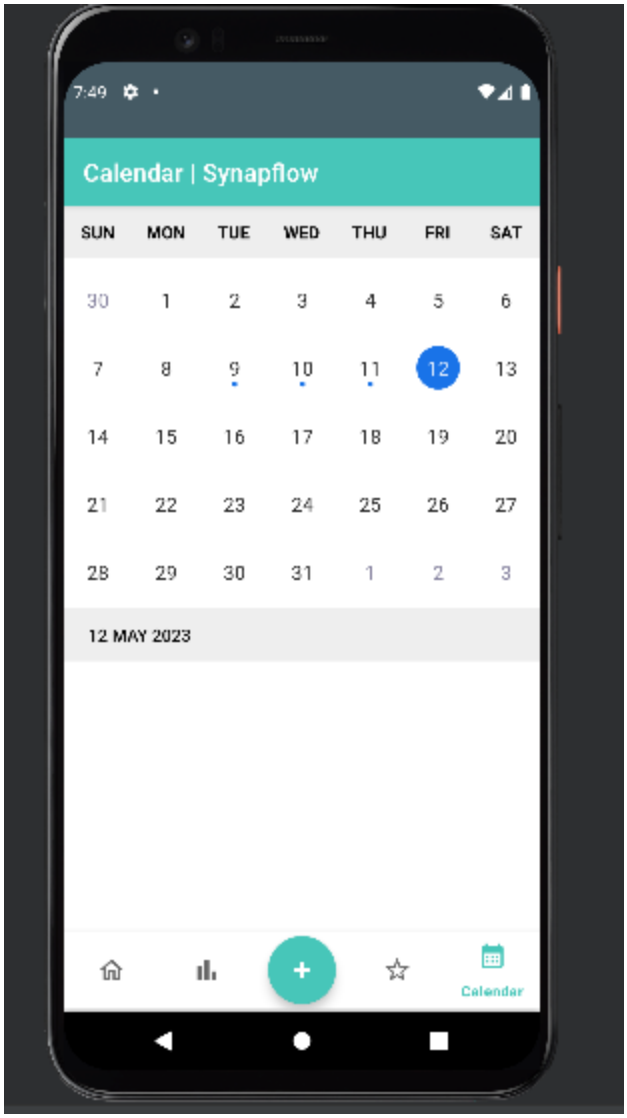

8:10
Timer | Synapflow
Study Time!
8:36
Home

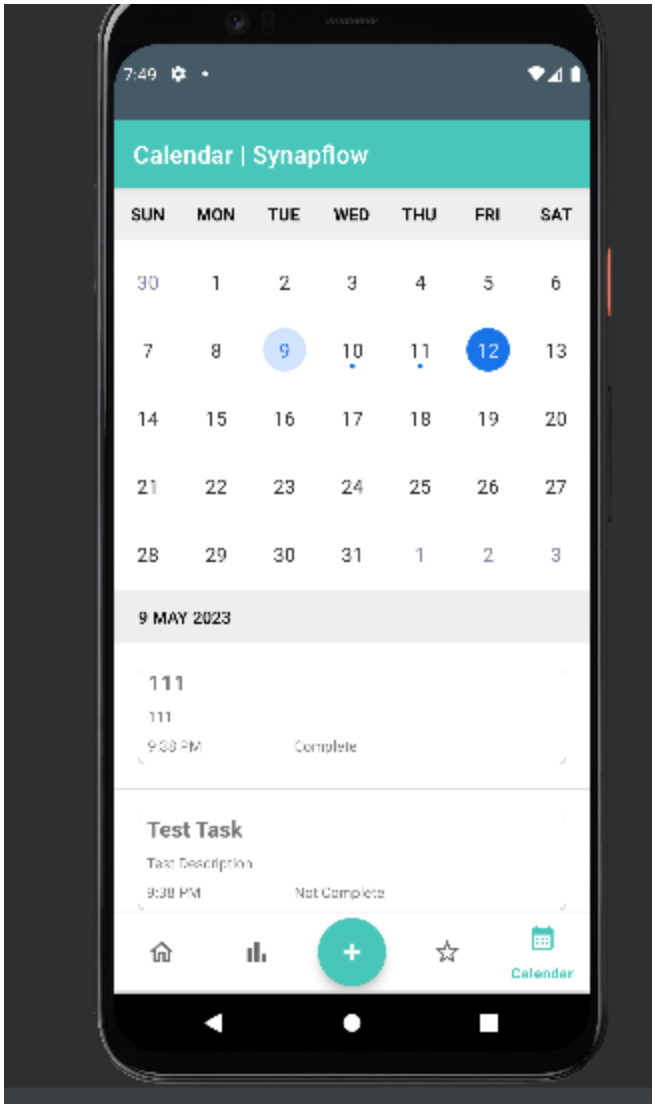

8:19
Timer | Synapflow
Break Time!
5:00
Time is up!
Home

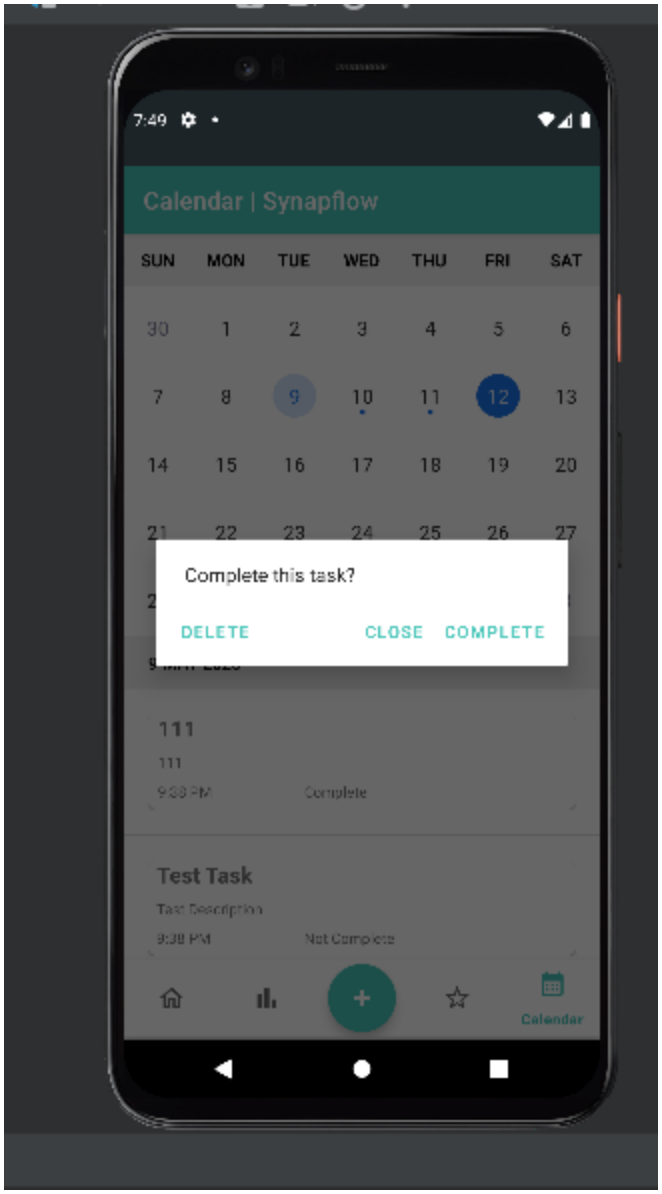

8:19
Timer | Synapflow
Break Time!
4:50
Home

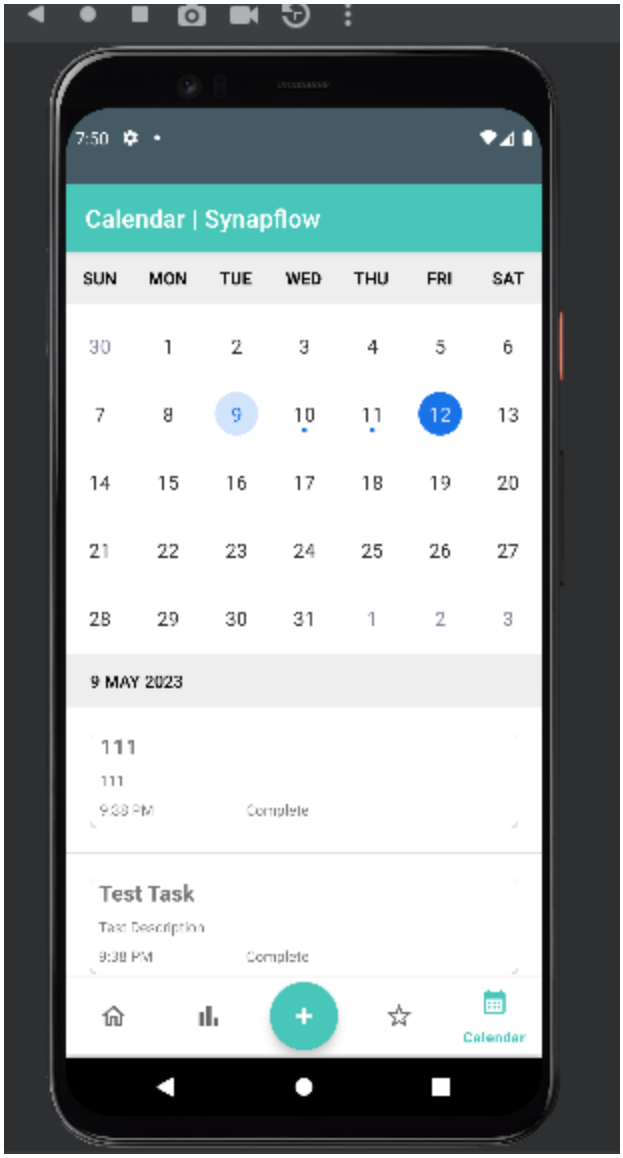

Analytics | Synapflow
Last Months Sessions Count
Last Weeks Completed Study Sessions
Analytics

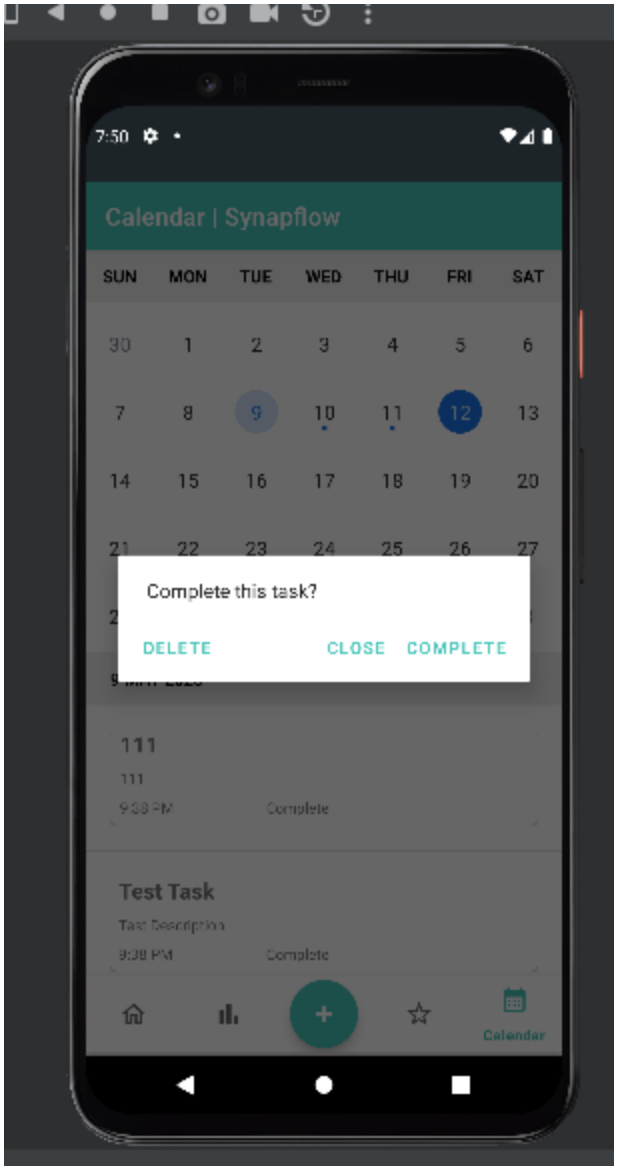

7:49
Rewards | SynapFlow
Rewards
Completed 1 Study Session
Complete 5 Study Session
Complete 10 Study Session
Rewards

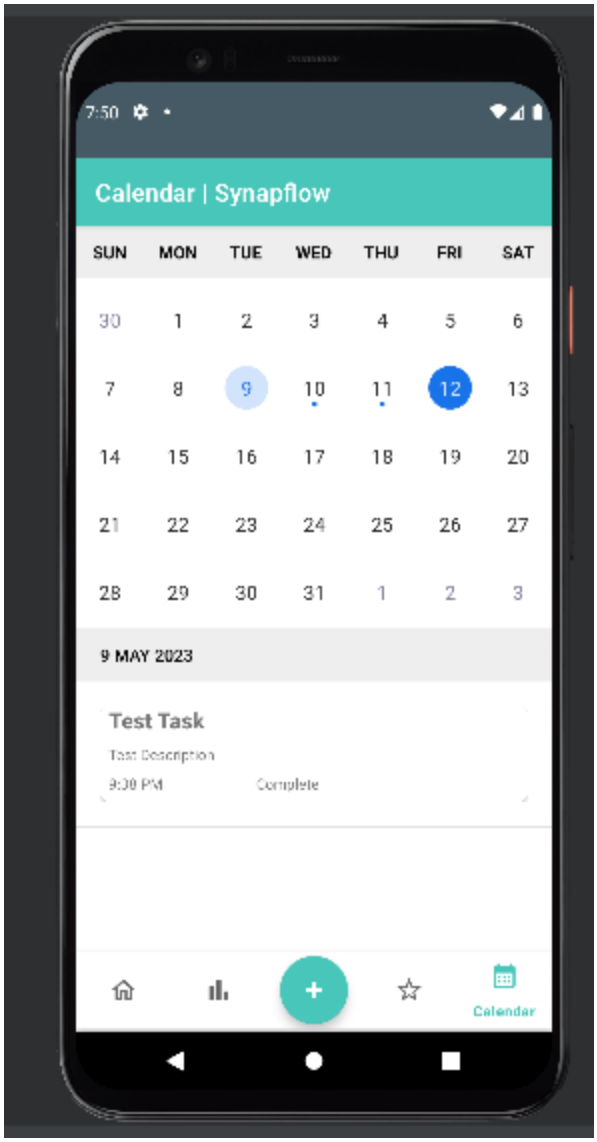

7:49
Calendar | Synapflow
SUN
MON
TUE
WED
THU
FRI
SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3
12 MAY 2023
Calendar

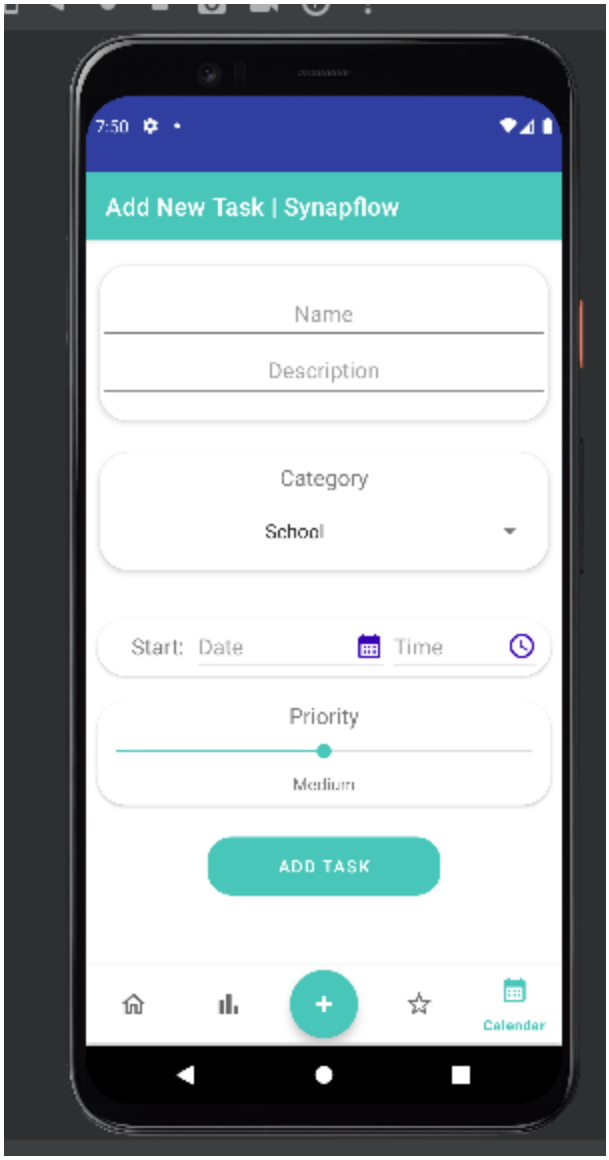

7:49
Calendar | Synapflow
SUN MON TUE WED THU FRI SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3
9 MAY 2023
111
111
9:38 PM
Complete
Test Task
Task Description
9:38 PM
Not Complete
Calendar

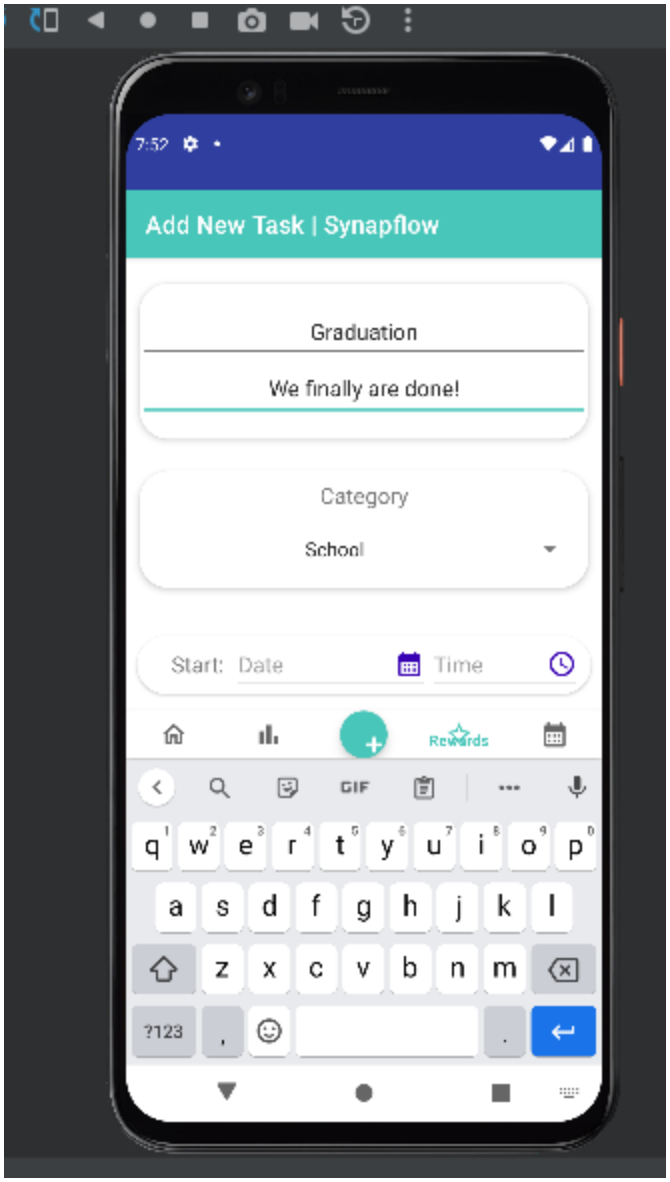

7:49
Calendar | Synapflow
SUN MON TUE WED THU FRI SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
Complete this task?
DELETE
CLOSE
COMPLETE
111
111
Complete
Test Task
Not Complete
Calendar

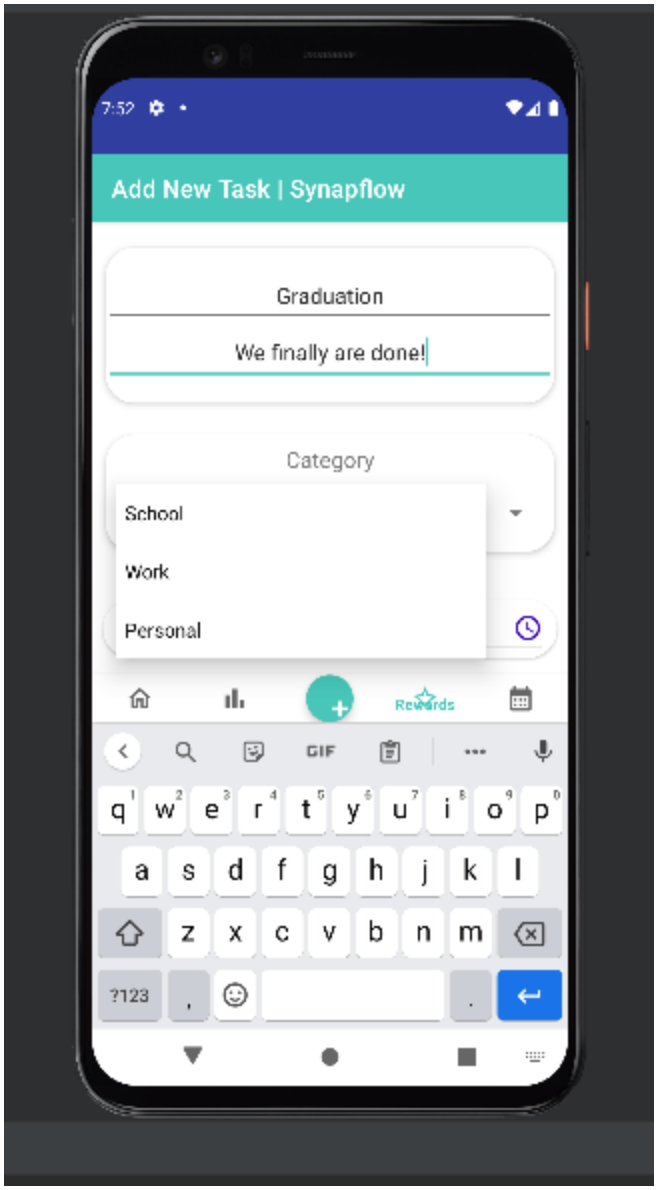

7:50
Calendar | Synapflow
SUN MON TUE WED THU FRI SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3
9 MAY 2023
111
111
9:38 PM
Complete
Test Task
Task Description
9:38 PM
Complete
Calendar

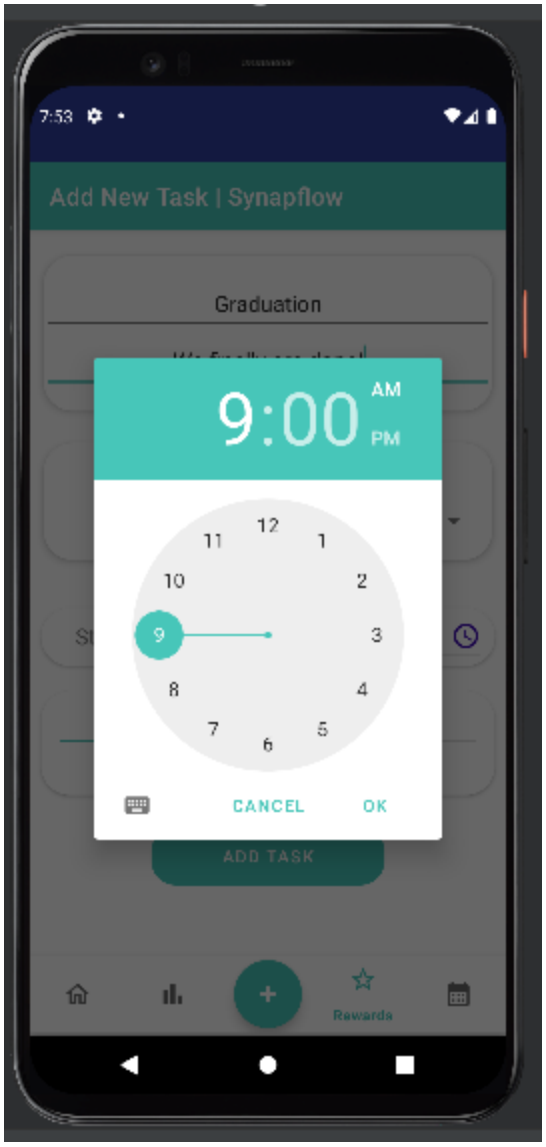

7:50
Calendar | Synapflow
SUN MON TUE WED THU FRI SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
Complete this task?
DELETE
CLOSE
COMPLETE
111
111
Complete
Test Task
Complete
Calendar

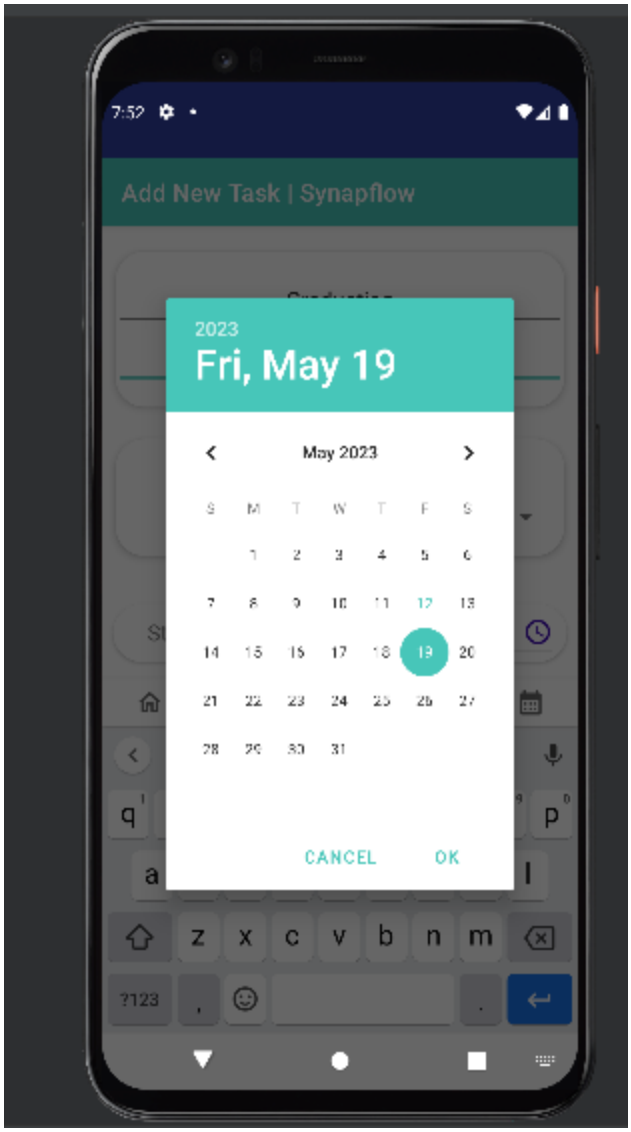

7:50
Calendar | Synapflow
SUN MON TUE WED THU FRI SAT
30 1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31 1 2 3
9 MAY 2023
Test Task
Test Description
9:30 PM
Complete
Calendar

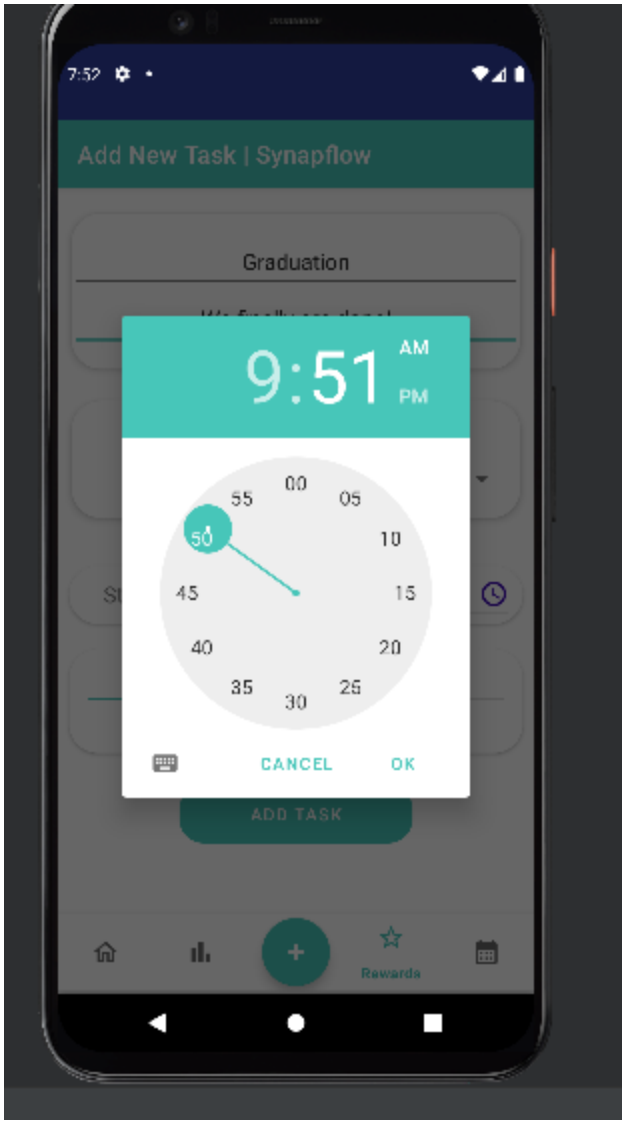

7:50
Add New Task | Synapflow
Name
Description
Category
School
Start:
Date
Time
Priority
Medium
ADD TASK
Calendar

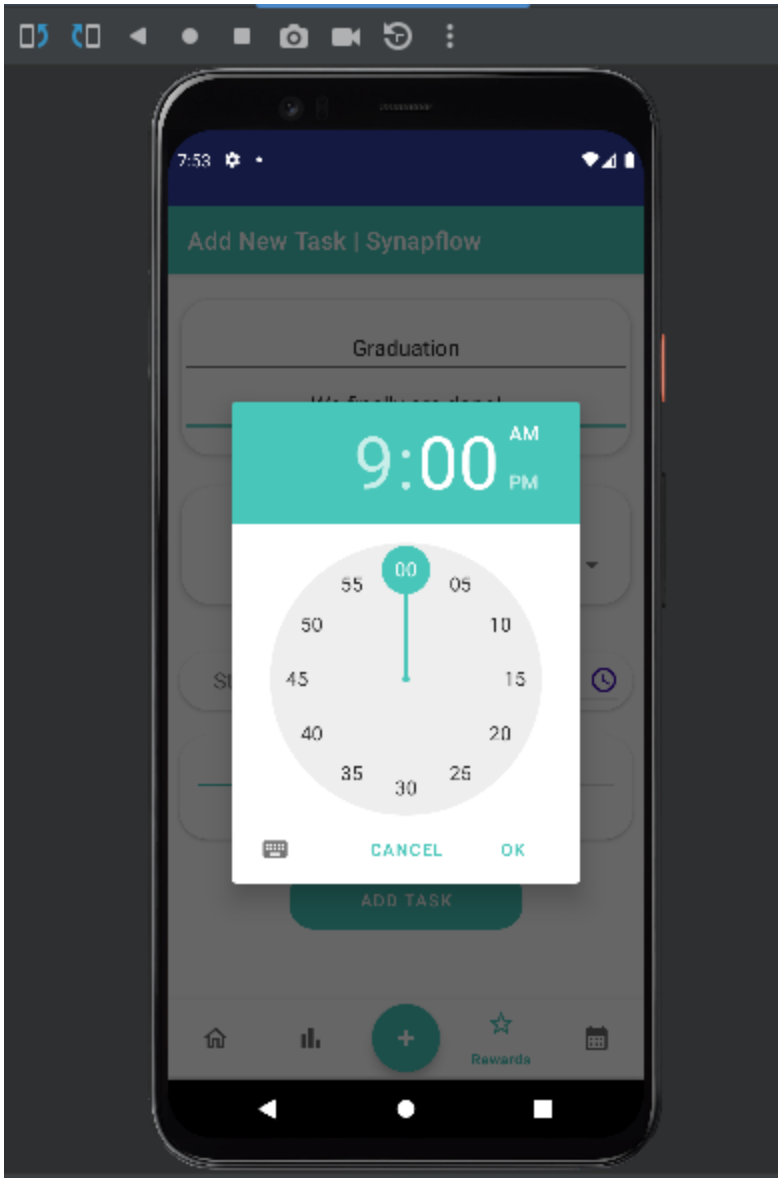

7:52
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Start: Date
Time
Rewards
GIF

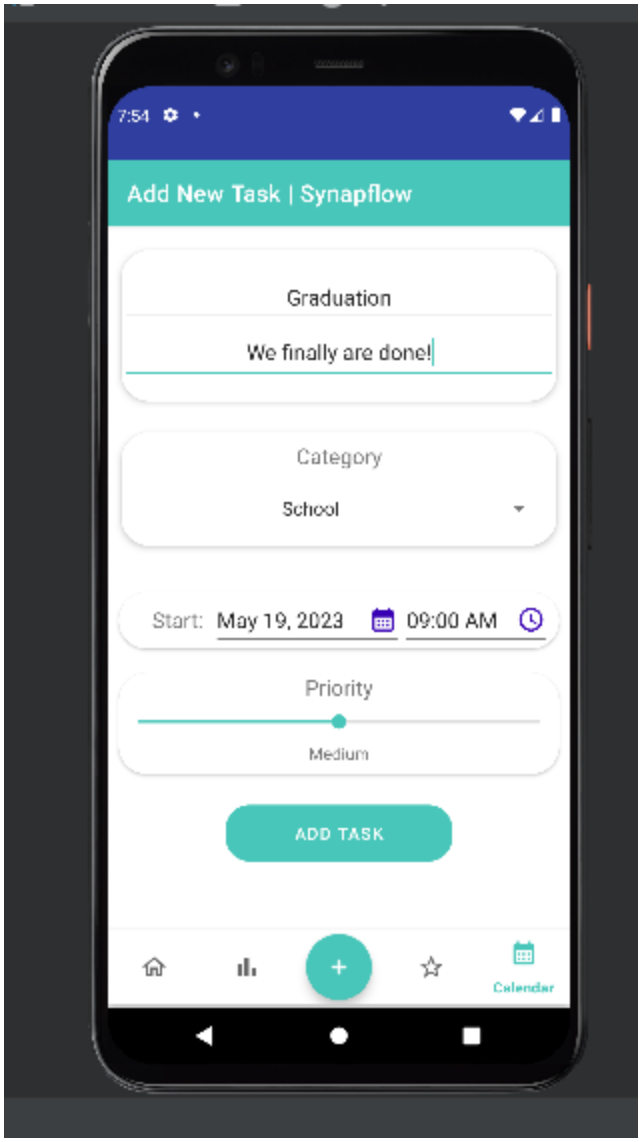

7:52
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Work
Personal
Rewards
GIF
?123

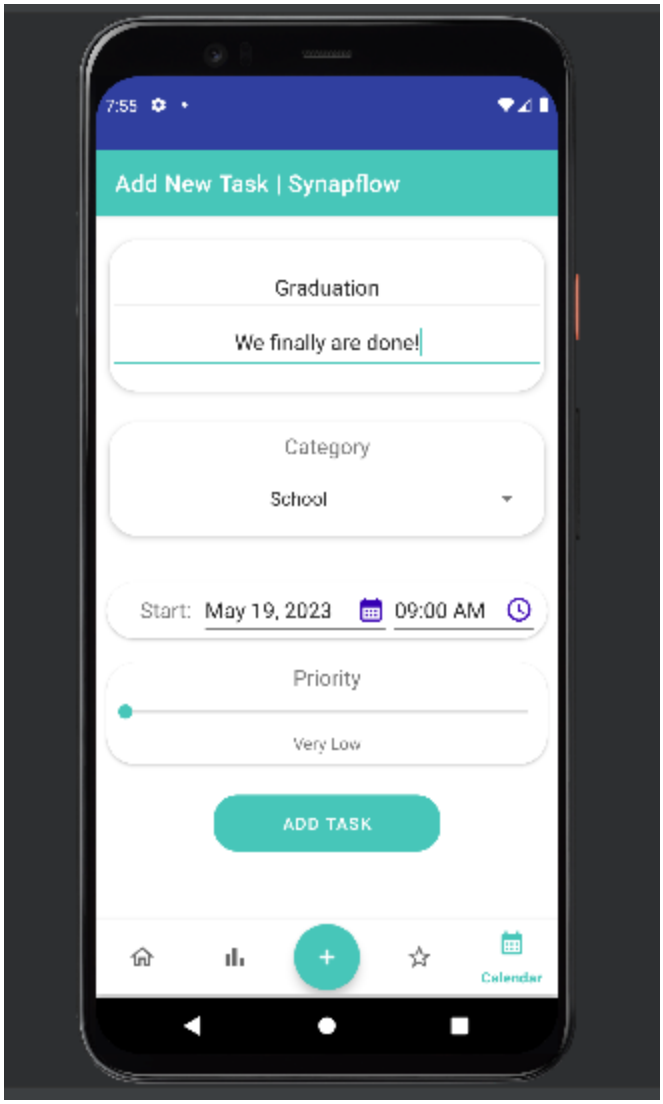

7:53
Add New Task | Synapflow
Graduation
9:00
AM
PM
12
1
2
3
4
5
6
7
8
9
10
11
CANCEL
OK
ADD TASK
Rewards

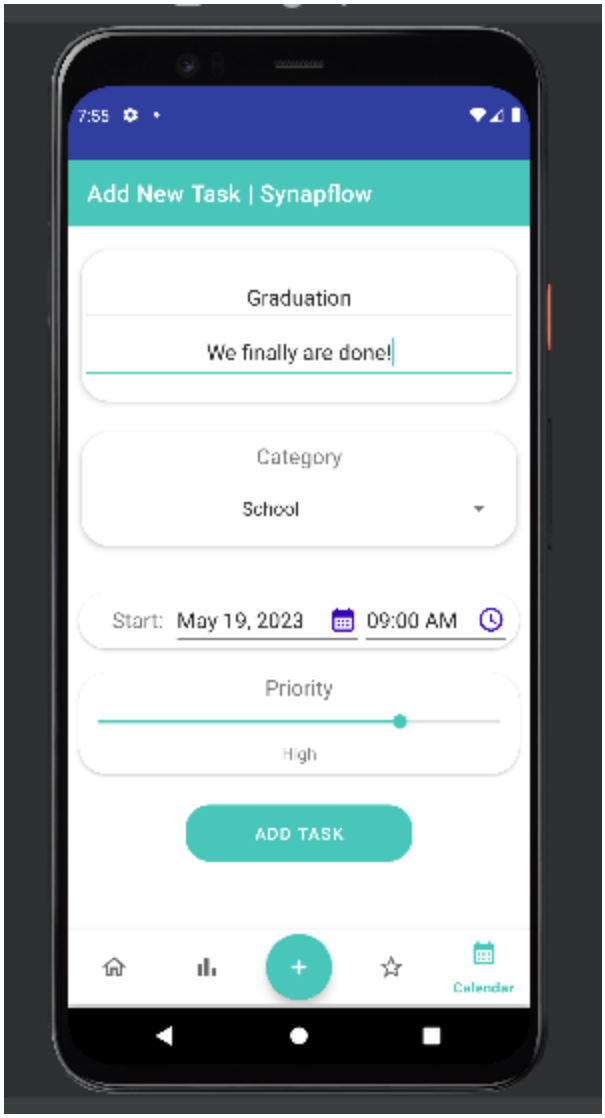

7:52
Add New Task | Synapflow
2023
Fri, May 19
May 2023
S M T W T F S
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
CANCEL
OK

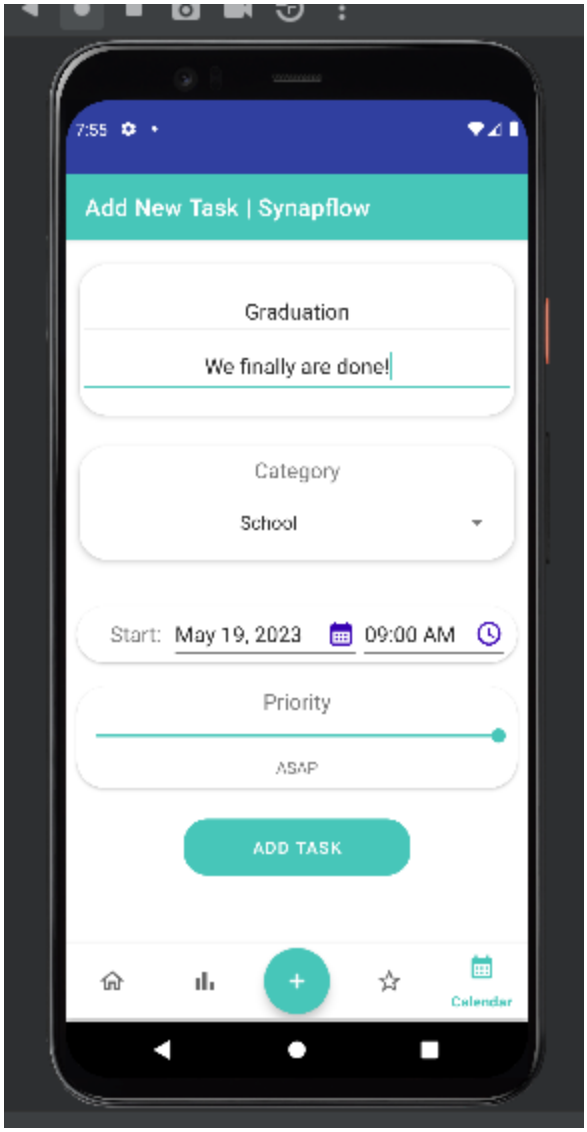

7:52
Add New Task | Synapflow
Graduation
9:51
AM
PM
00
05
10
15
20
25
30
35
40
45
50
55
CANCEL
OK
ADD TASK
Rewards

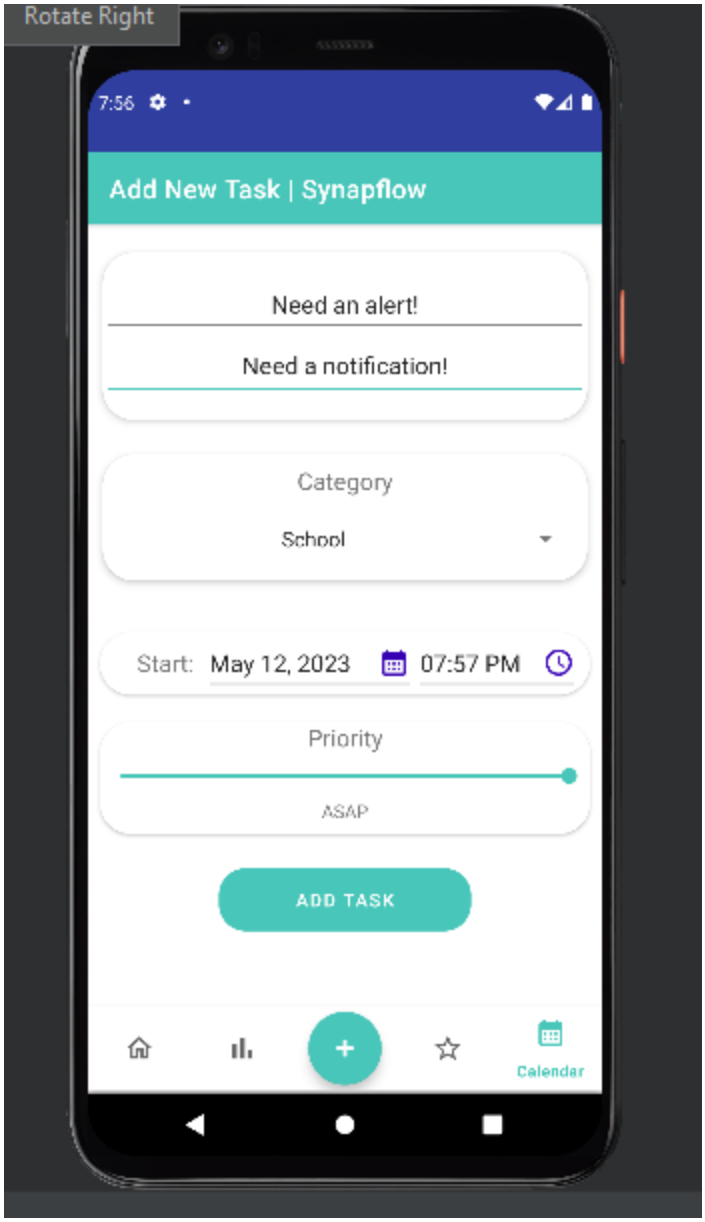

7:53
Add New Task | Synapflow
Graduation
9:00
AM
PM
00
05
10
15
20
25
30
35
40
45
50
55
CANCEL
OK
ADD TASK
Rewards

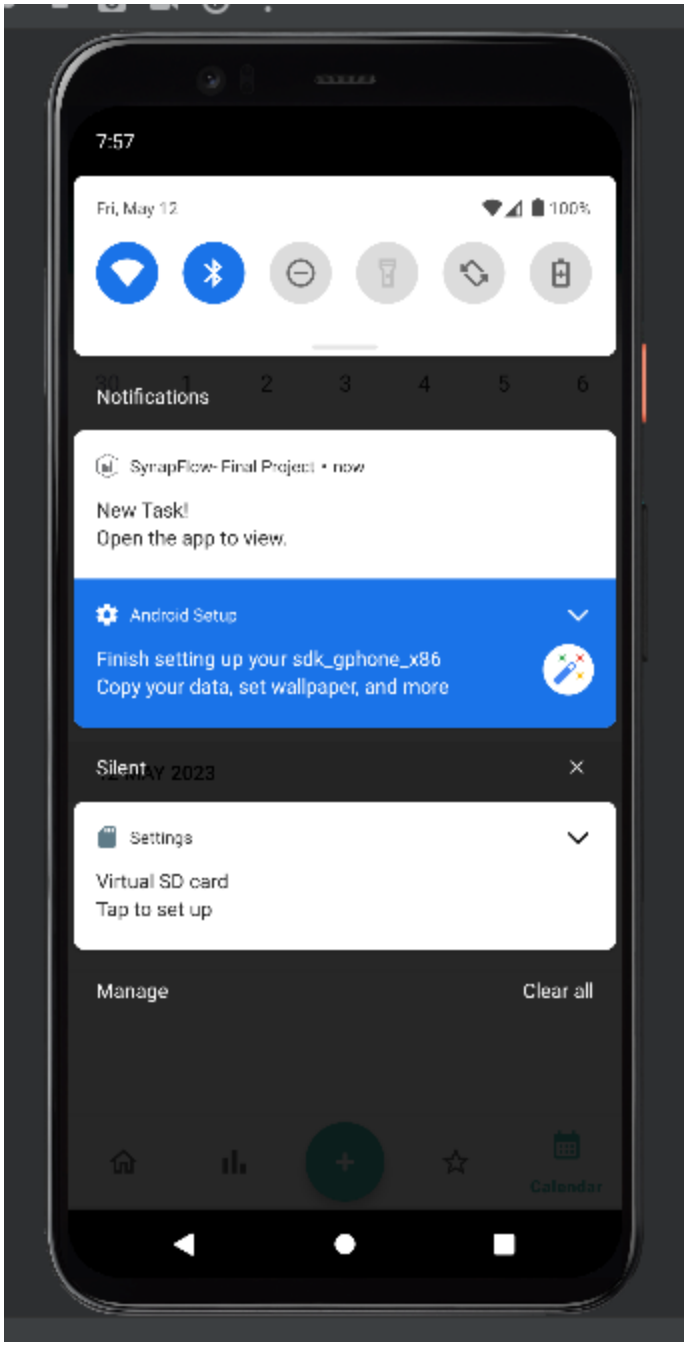

7:54
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Start: May 19, 2023 09:00 AM
Priority
Medium
ADD TASK
Calendar


7:55
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Start: May 19, 2023 09:00 AM
Priority
Very Low
ADD TASK
Calendar


7:55
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Start: May 19, 2023 09:00 AM
Priority
High
ADD TASK
Calendar

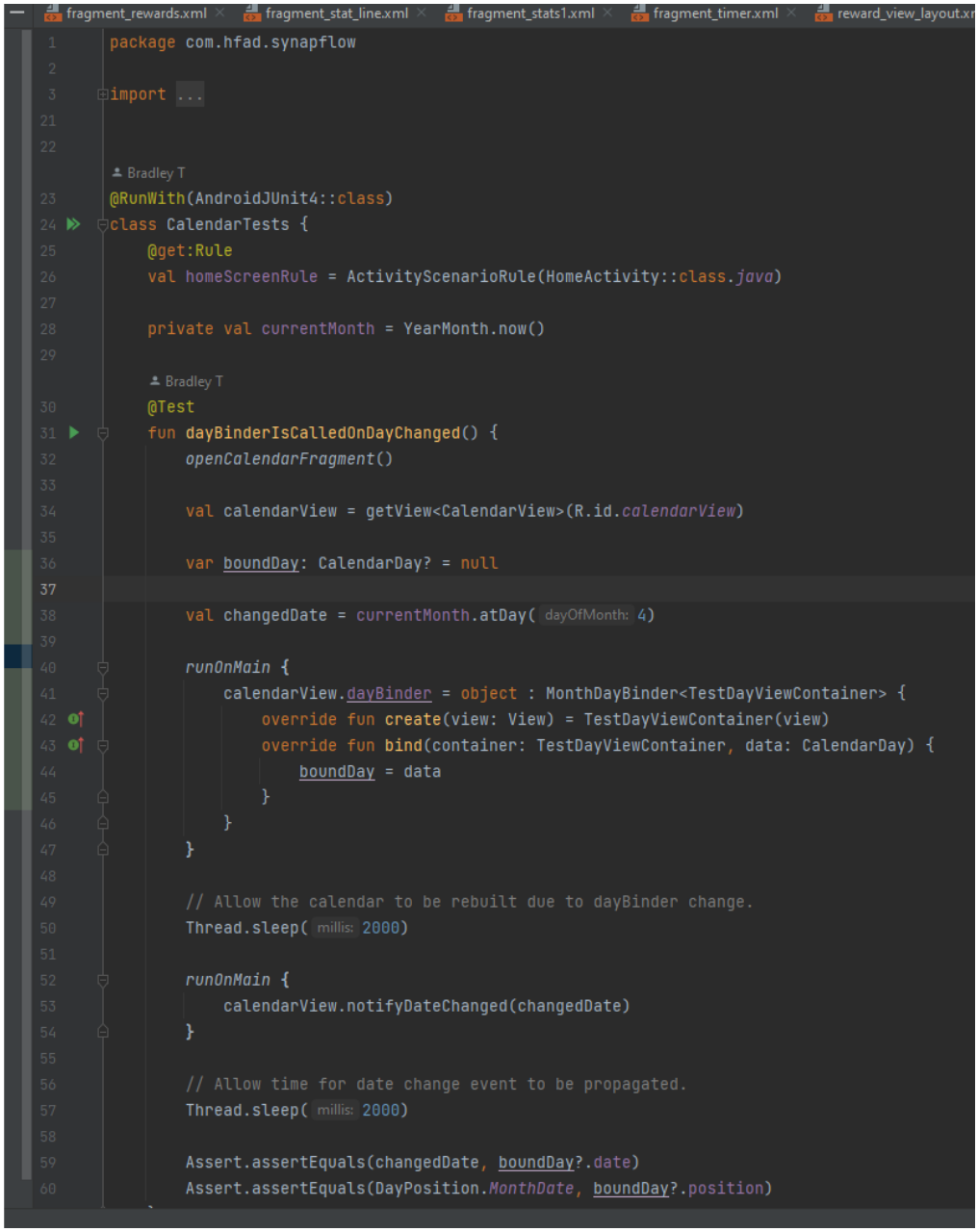

7:55
Add New Task | Synapflow
Graduation
We finally are done!
Category
School
Start: May 19, 2023 09:00 AM
Priority
ASAP
ADD TASK
Calendar


Rotate Right
7:56
Add New Task | Synapflow
Need an alert!
Need a notification!
Category
School
Start: May 12, 2023 07:57 PM
Priority
ASAP
ADD TASK
Calendar

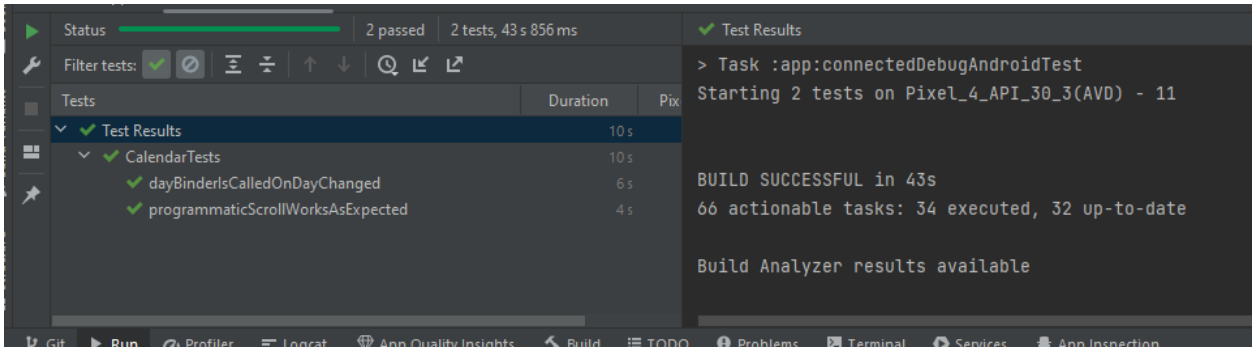


Test Outputs and Test files

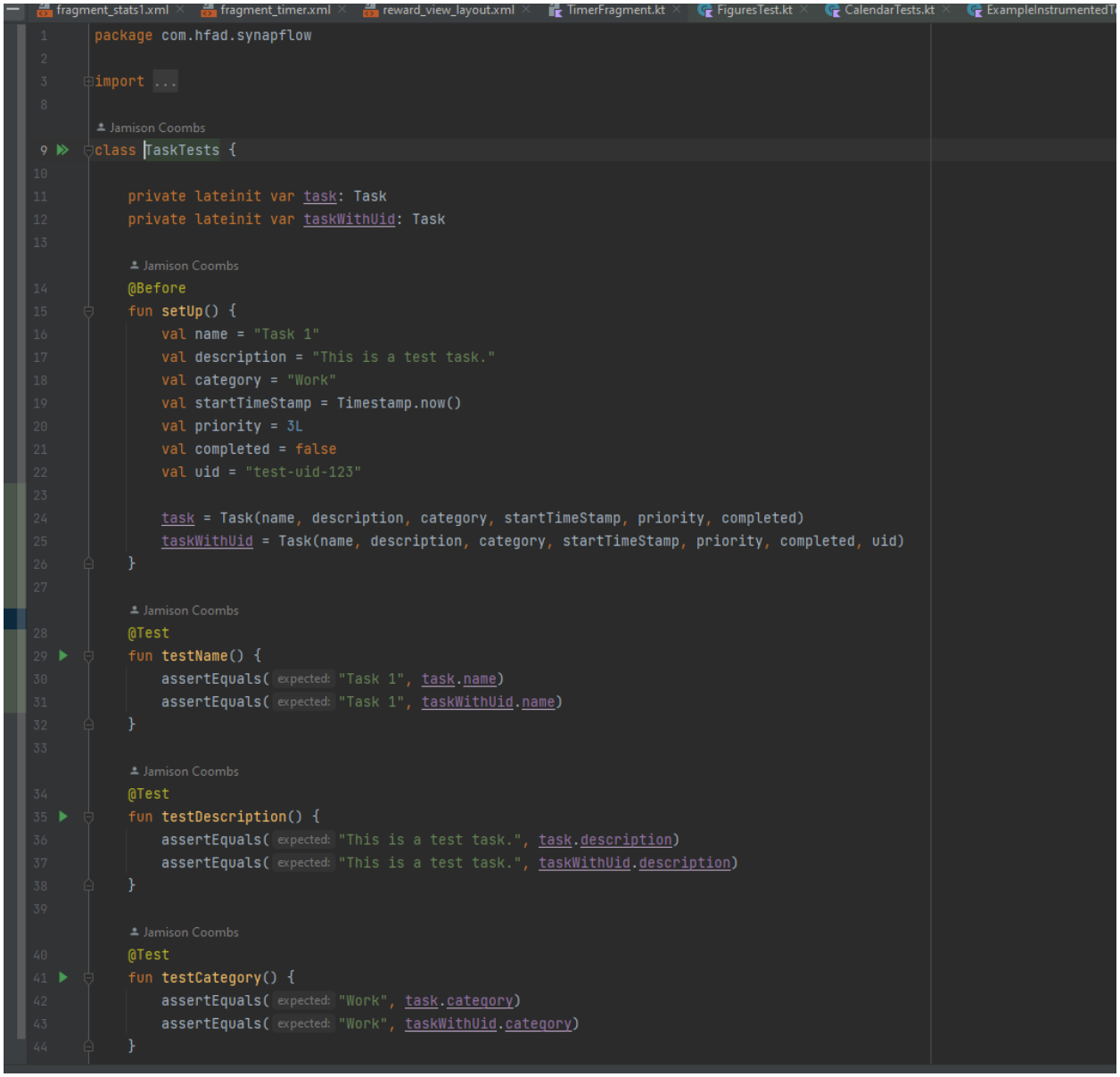
CalendarTests


fragment_rewards.xml
fragment_stat_line.xml
fragment_stats1.xml
fragment_timer.xml
reward_view_layout.xml
package com.hfad.synapflow
import ...
Bradley T
@RunWith(AndroidJUnit4::class)
class CalendarTests {
@get:Rule
val homeScreenRule = ActivityScenarioRule(HomeActivity::class.java)
private val currentMonth = YearMonth.now()
Bradley T
@Test
fun dayBinderIsCalledOnDayChanged() {
openCalendarFragment()
val calendarView = getView<CalendarView>(R.id.calendarView)
var boundDay: CalendarDay? = null
val changedDate = currentMonth.atDay(dayOfMonth: 4)
runOnMain {
calendarView.dayBinder = object : MonthDayBinder<TestDayViewContainer> {
override fun create(view: View) = TestDayViewContainer(view)
override fun bind(container: TestDayViewContainer, data: CalendarDay) {
boundDay = data
}
}
}
// Allow the calendar to be rebuilt due to dayBinder change.
Thread.sleep(millis: 2000)
runOnMain {
calendarView.notifyDateChanged(changedDate)
}
// Allow time for date change event to be propagated.
Thread.sleep(millis: 2000)
Assert.assertEquals(changedDate, boundDay?.date)
Assert.assertEquals(DayPosition.MonthDate, boundDay?.position)

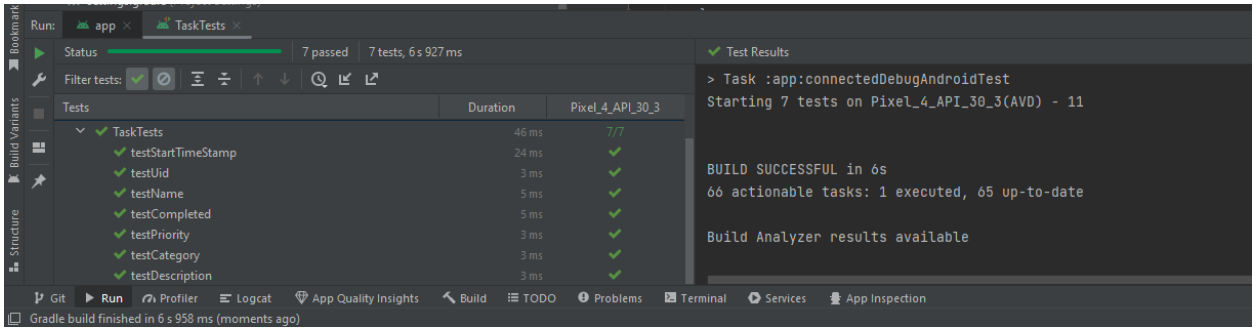
```
    }

    Bradley T
    @Test
    fun programmaticScrollWorksAsExpected() {
        openCalendarFragment()

        val calendarView = getView<CalendarView>(R.id.calendarView)

        Assert.assertNotNull(calendarView.findViewWithTag(currentMonth.atDay( dayOfMonth: 1).hashCode()))

        val nextFourMonths = currentMonth.plusMonths( monthsToAdd: 4)

        runOnMain {
            calendarView.scrollToMonth(nextFourMonths)
        }

        Thread.sleep( millis: 2000)

        Assert.assertNull(calendarView.findViewWithTag(currentMonth.atDay( dayOfMonth: 1).hashCode()))
        Assert.assertNotNull(calendarView.findViewWithTag(nextFourMonths.atDay( dayOfMonth: 1).hashCode()))
    }

    Bradley T
    private fun <T : View> getView(@IdRes id: Int): T = homeScreenRule.getView(id)
}
```



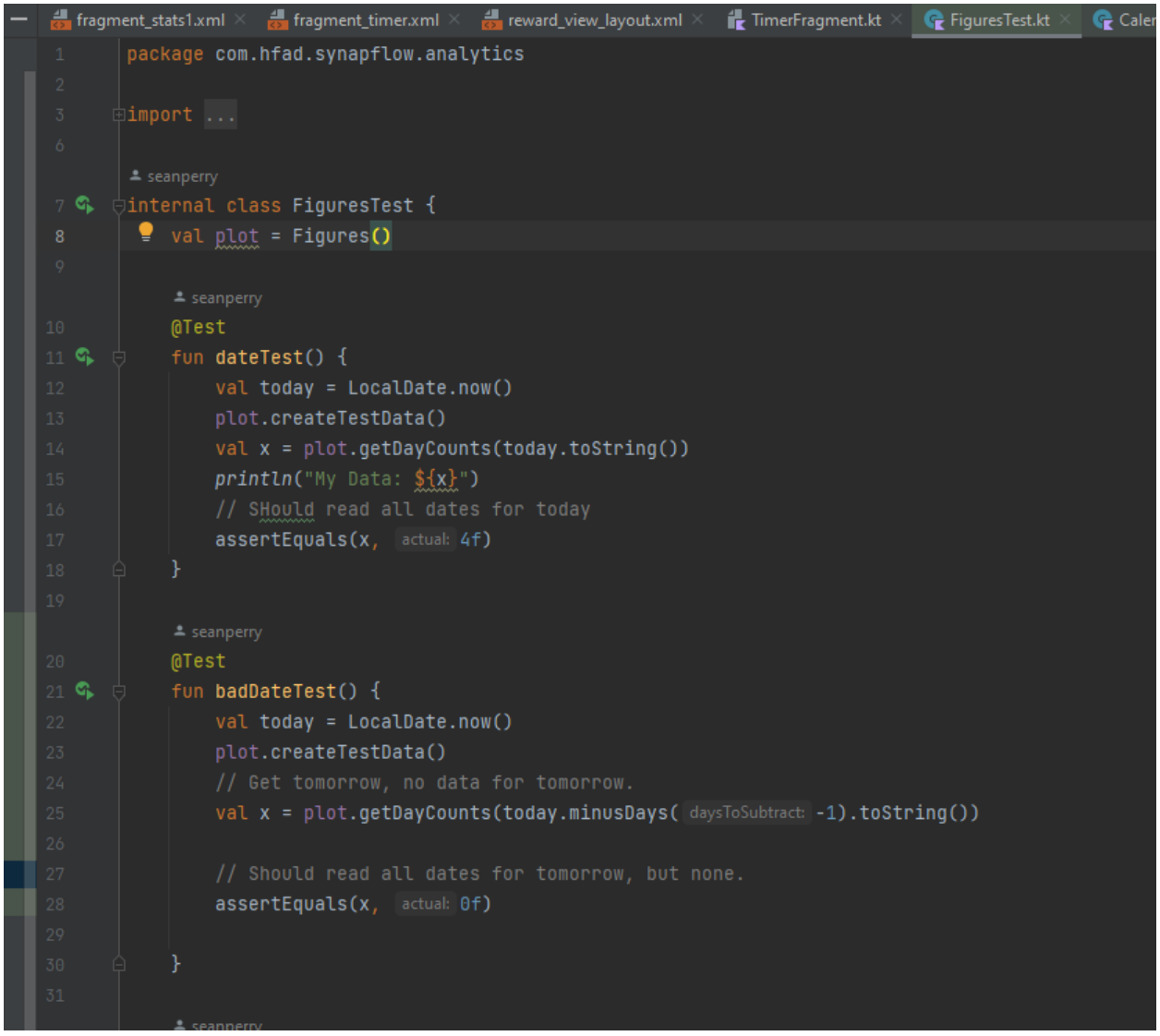
TaskTests


fragment_stats1.xml
fragment_timer.xml
reward_view_layout.xml
TimerFragment.kt
FiguresTest.kt
CalendarTests.kt
ExampleInstrumentedT
package com.hfad.synapflow
import ...
Jamison Coombs
class TaskTests {
private lateinit var task: Task
private lateinit var taskWithUid: Task
Jamison Coombs
@Before
fun setUp() {
val name = "Task 1"
val description = "This is a test task."
val category = "Work"
val startTimeStamp = Timestamp.now()
val priority = 3L
val completed = false
val uid = "test-uid-123"
task = Task(name, description, category, startTimeStamp, priority, completed)
taskWithUid = Task(name, description, category, startTimeStamp, priority, completed, uid)
}
Jamison Coombs
@Test
fun testName() {
assertEquals(expected: "Task 1", task.name)
assertEquals(expected: "Task 1", taskWithUid.name)
}
Jamison Coombs
@Test
fun testDescription() {
assertEquals(expected: "This is a test task.", task.description)
assertEquals(expected: "This is a test task.", taskWithUid.description)
}
Jamison Coombs
@Test
fun testCategory() {
assertEquals(expected: "Work", task.category)
assertEquals(expected: "Work", taskWithUid.category)
}

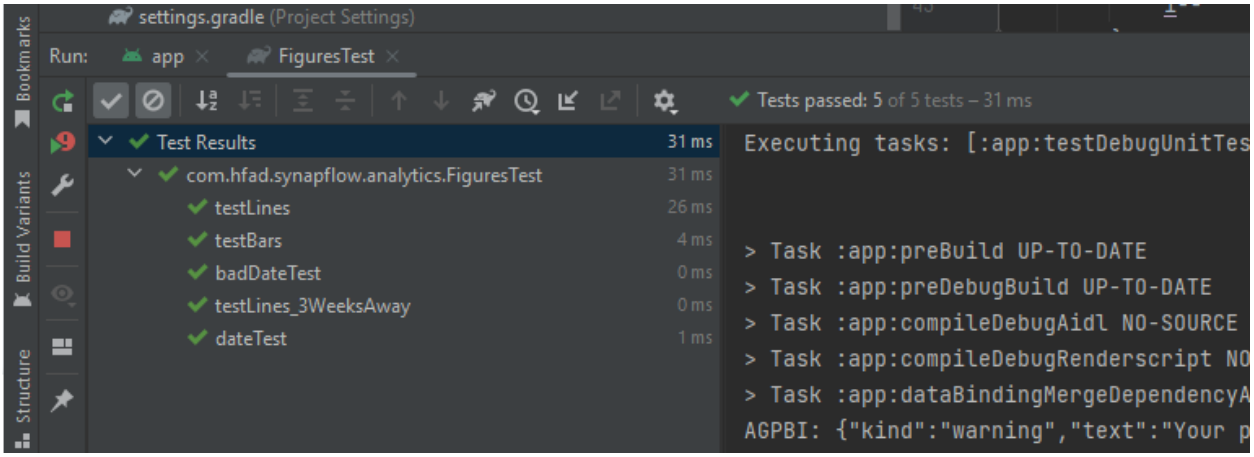
```
45
        Jamison Coombs
46      @Test
47      fun testStartTimeStamp() {
48          assertEquals(Timestamp.now().seconds, task.startTimeStamp.seconds)
49          assertEquals(Timestamp.now().seconds, taskWithUid.startTimeStamp.seconds)
50      }
51
        Jamison Coombs
52      @Test
53      fun testPriority() {
54          assertEquals( expected: 3L, task.priority)
55          assertEquals( expected: 3L, taskWithUid.priority)
56      }
57
        Jamison Coombs
58      @Test
59      fun testCompleted() {
60          assertFalse(task.completed)
61          assertFalse(taskWithUid.completed)
62      }
63
        Jamison Coombs
64      @Test
65      fun testUid() {
66          assertEquals( expected: "", task.getUID())
67          assertEquals( expected: "test-uid-123", taskWithUid.getUID())
68      }
69  }
```



FiguresTest


fragment_stats1.xml
fragment_timer.xml
reward_view_layout.xml
TimerFragment.kt
FiguresTest.kt
Caler
package com.hfad.synapflow.analytics
import ...
seanperry
internal class FiguresTest {
val plot = Figures()
seanperry
@Test
fun dateTest() {
val today = LocalDate.now()
plot.createTestData()
val x = plot.getDayCounts(today.toString())
println("My Data: ${x}")
// SHould read all dates for today
assertEquals(x, actual: 4f)
}
seanperry
@Test
fun badDateTest() {
val today = LocalDate.now()
plot.createTestData()
// Get tomorrow, no data for tomorrow.
val x = plot.getDayCounts(today.minusDays(daysToSubtract: -1).toString())
// Should read all dates for tomorrow, but none.
assertEquals(x, actual: 0f)
}
seanperry

```
32      @Test
33      fun testBars() {
34
35          plot.createTestData()
36
37          val x = plot.getBarCount()
38          var i = 4 // keeps track of x value
39          println(plot.getTestData())
40          for (z in x) {
41              assertEquals(z.x, i.toFloat())
42              assertEquals(z.y,  actual: 4f * 7)
43              i--
44          }
45
46      }
47
        seanperry
48      @Test
49      fun testLines() {
50          plot.createTestData()
51          var i = 0
52          val x = plot.getWeekLine()
53          for (z in x) {
54              assertEquals(z.x, i.toFloat())
55              assertEquals(z.y,  actual: 4f)
56              i++
57          }
58      }
59
        seanperry
60      @Test
61      fun testLines_3WeeksAway() {
62          plot.createTestData()
63          var i = 0
64          val x = plot.getWeekLine( weekAgo: 3)
65          for (z in x) {
66              assertEquals(z.x, i.toFloat())
67              assertEquals(z.y,  actual: 4f)
68              i++
69          }
70      }
71  }
```


settings.gradle (Project Settings)
Run:
app
FiguresTest
Tests passed: 5 of 5 tests – 31 ms
Test Results 31 ms
com.hfad.synapflow.analytics.FiguresTest 31 ms
testLines 26 ms
testBars 4 ms
badDateTest 0 ms
testLines_3WeeksAway 0 ms
dateTest 1 ms
Executing tasks: [:app:testDebugUnitTes
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:compileDebugAidl NO-SOURCE
> Task :app:compileDebugRenderscript NO
> Task :app:dataBindingMergeDependencyA
AGPBI: {"kind":"warning","text":"Your p
Bookmarks
Build Variants
Structure