SynapFlow

- Bradley Tran
- Jamison Coombs
- Sean Perry
- Trevor Kvanvig
- Edward Uriarte

SynapFlow Quick Overview

- SynapFlow is an app created from scratch using the Pomodoro technique.
  - The Pomodoro Technique uses a 25 minute study time, 5 minute break schedule.
  - We wanted to create something to help us study in a reasonable time frame
- We wanted to help keep track of Tasks like exams, meetings, and plan times to study.
- Trophy system to award our users for having multiple study sessions
- Analytics to keep track of the users sessions by the day, and group them over the last few weeks.
- We utilized Google Authentication for login information
- Firebase to store user data
- Multiple API's for Analytics, Timer, Trophies, and the calendar

Concepts Overview

- Navigation
- Fragments
- Firebase
- API
- Notification
- Recyclerview
- Material Design
- Unit Testing
- Permission
- ViewModel

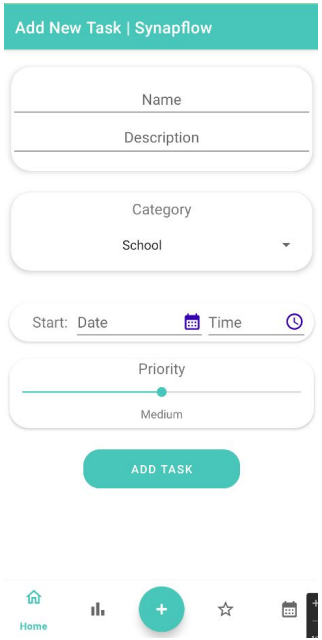
Demo

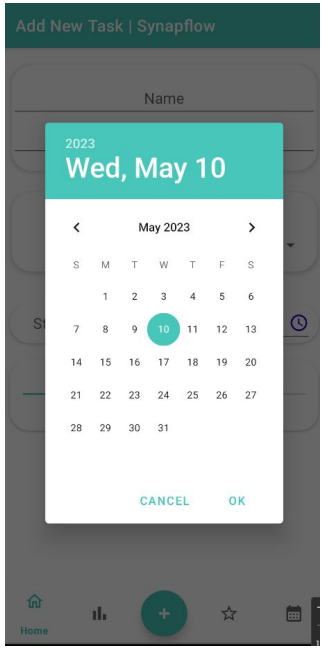
Jamison Coombs - Tasks

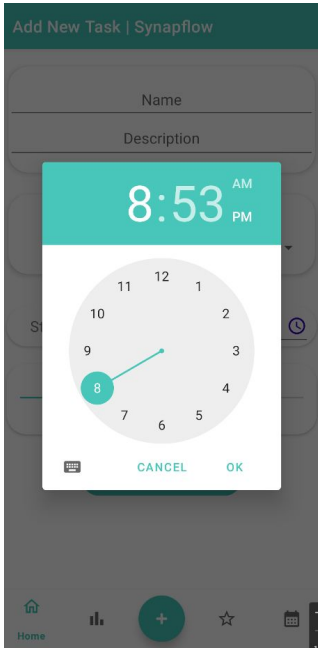
<u>Add Task Fragment</u>

Contains the layout to add a new task to the database with the following fields:

- Name
- Description
- Category
- Date
- Time
- Priority







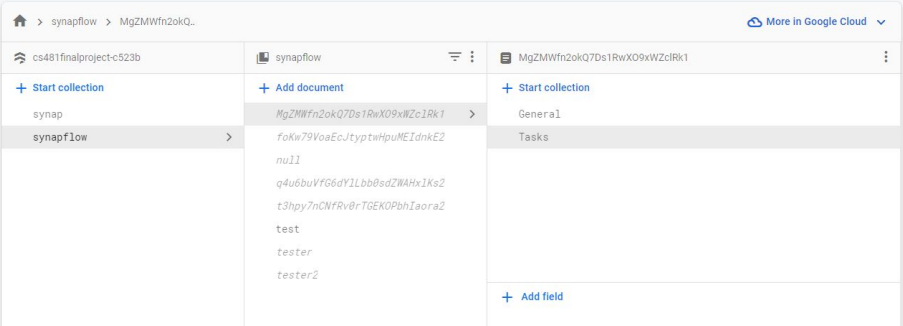
Jamison Coombs - Tasks (cont.)

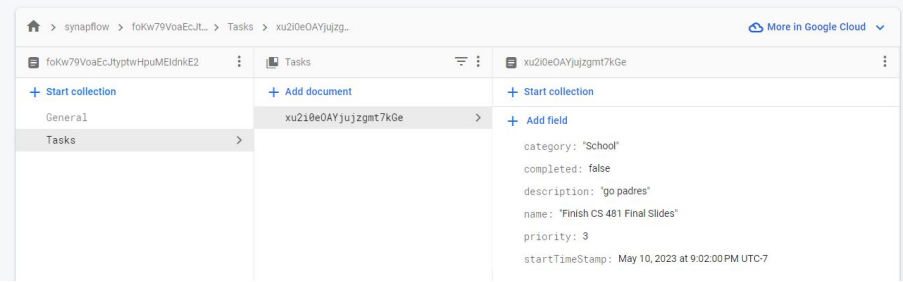
<u>Task Firestore Implementation</u>

Created functions in our firestore service class so that my teammates can easily call them to interact with the tasks on the backend.

Functions:

- Add Task
- Complete Task
- Get All Tasks
- Get Task Info
- Delete Task
- Update Task





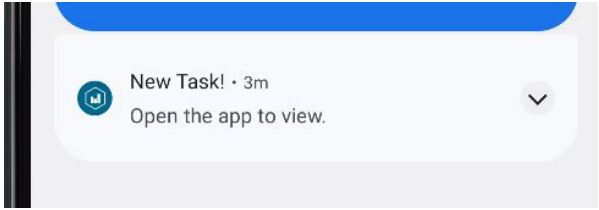
Jamison Coombs - Tasks (cont.)

(This one managed to give me the biggest headache all due to one line of code) *It be like that

Task Notifications

- AlarmManager to set an alarm in the Android OS at the set time and date of the task to trigger a broadcast to the app's BroadcastReceiver

- When a signal is received by the BroadcastReceiver it will build a notification and then notify the user that they have a task to complete



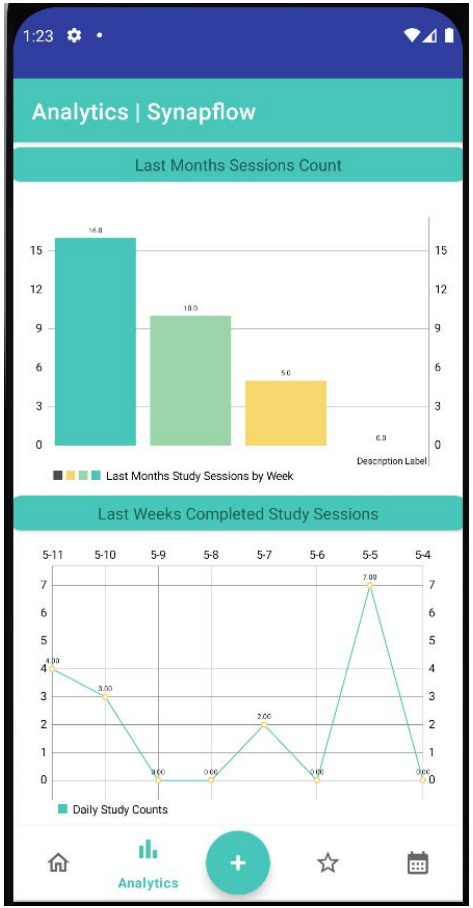

```
Jamison Coombs
class NotificationBroadcastReceiver : BroadcastReceiver() {

    Jamison Coombs
    override fun onReceive(context: Context, intent: Intent) {
        // Create an explicit intent for the app's main activity
        val homeIntent = Intent(context, HomeActivity::class.java).apply { this: Intent
            flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
        }

        val pendingIntent = PendingIntent.getActivity(context, requestCode: 0, homeIntent, flags: Pendi
        val notification = NotificationCompat.Builder(context, channelId: "C10")
            .setSmallIcon(R.drawable.logo_only)
            .setContentTitle("New Task!")
            .setContentText("Open the app to view.")
            .setContentIntent(pendingIntent)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .build()

        // Display the notification
        val notificationManager =
            context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
        notificationManager.notify( id: 0, notification)
    }
}
```

Sean Perry - Analytics

<u>Analytics</u>

- Utilized the Android Chart API to display the different analytics information.
- Displays a bar chart of the count per week of the amount of completed study sessions.
- Displays a line chart displaying the count the users last 7 days of study sessions

- The data sessions count is gathered and stored inside firebase.

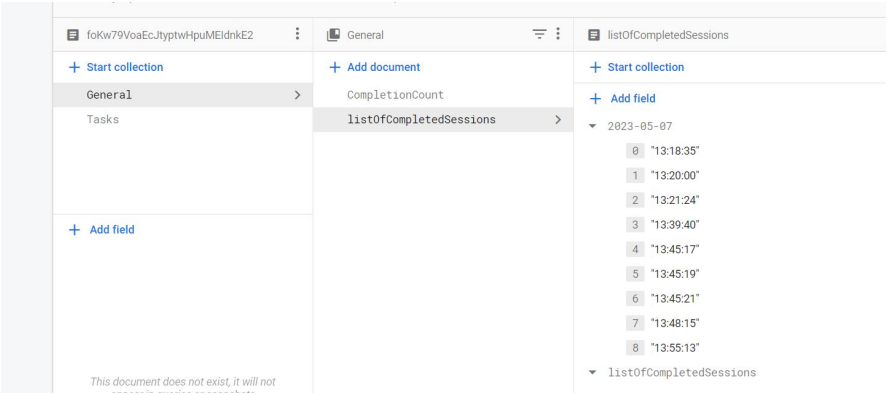


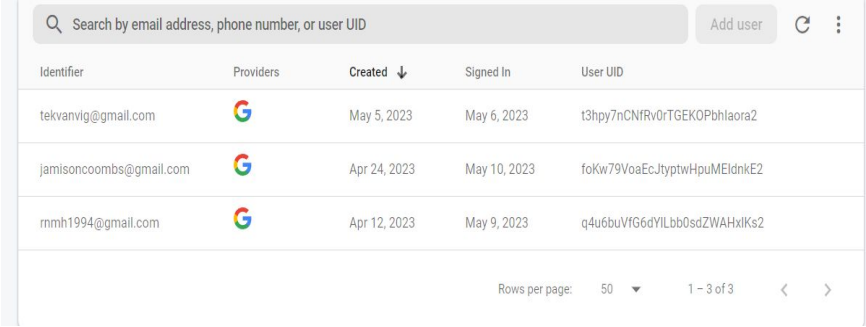
Sean Perry - Firestore / Google Authenticator

<u>Firestore</u>

- Google Authentication for user login
- General Firestore user setup
- General Firestore functionality
- Interfaced Firestore with the Analytics and the timer.

- Firestore was a struggle for me to grasp the concepts on.



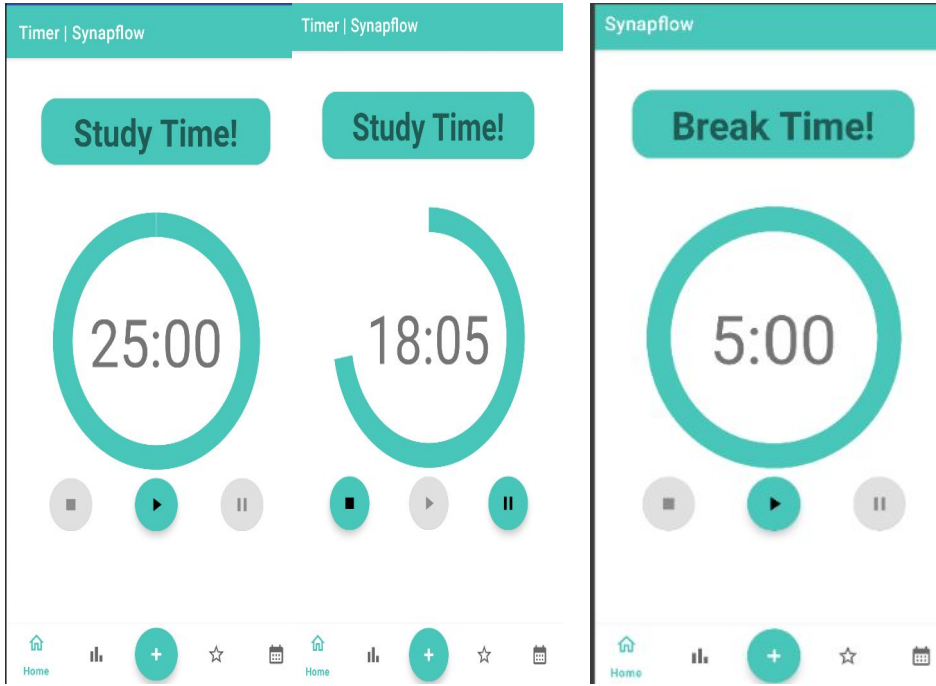


Sean Perry - Timer

Timer

- Implemented an overridden timer class that works with a LiveView model
- Added a Graphic API for the wheel animation
- Start, Stop, Pause functionality, persistent through the app.
- Session is 25 study, 5 minute break intervals

- LiveData was as struggle, due to some misunderstanding of the general setup
- Failed to be able to utilize WorkManager



Edward Uriarte

Support

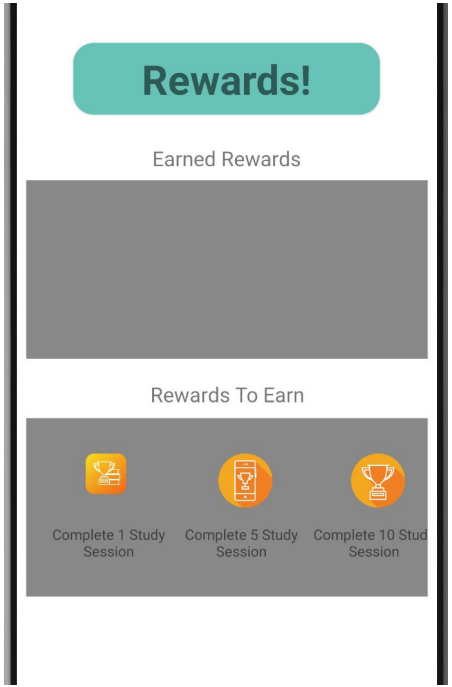
- Helped the groundwork
- Helped make the UI for the Timer/Intro Page, along with the Synapflow image
- Initial groundwork with the Timer, Fragments, and ViewModel.
- Helping on the final paper.

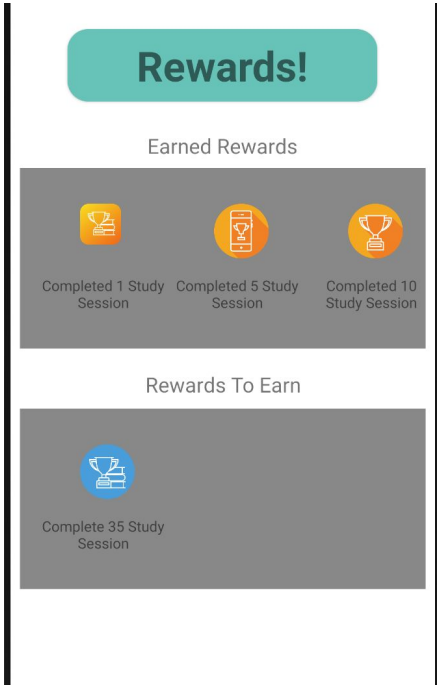
Trevor Kvanvig - Rewards

<u>Rewards</u>

Rewards Tab is used to keep track of user progress and keep them motivated to stay focused. Based on the number of completed study sessions they will earn trophies in trophy case in real time.

- Contains 2 horizontal scroll views that get dynamically loaded based on information changes from other parts of app making changes to firestore
- Once study session goals are hit. Trophies get moved into the Earned scroll view





Trevor Kvanvig - Rewards (cont.)

Rewards

- created a function to asynchronous return the Completed study session count from firestore
- Had to write kotlin code to dynamically create image and description elements then place it in linear layout to then add to the correct horizontal scroll view

```
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
```

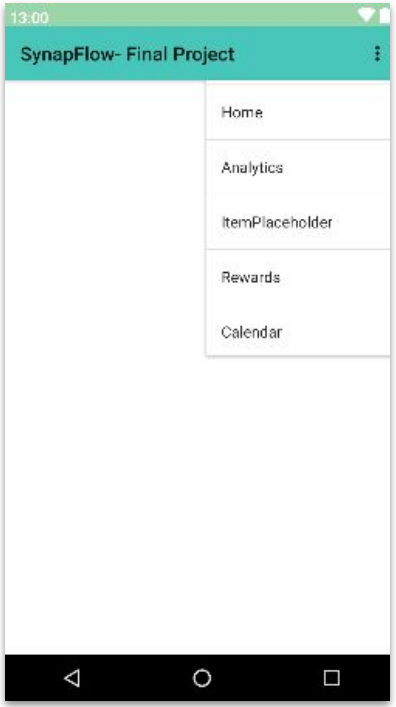
Bradley Tran - Bot. Nav. Menu

<u>Bottom Navigation Menu</u>

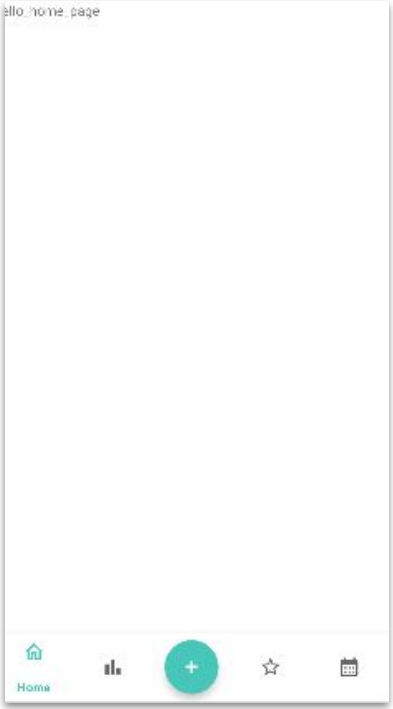
Primary navigation source for accessing various feature fragments. Button/items are set to display fragments through our fragmentContainerView depending on user interaction/selection.

Accessible Items/Fragments:

- Home
- Analytics
- Task Creation
- Rewards
- Calendar



menu xml design



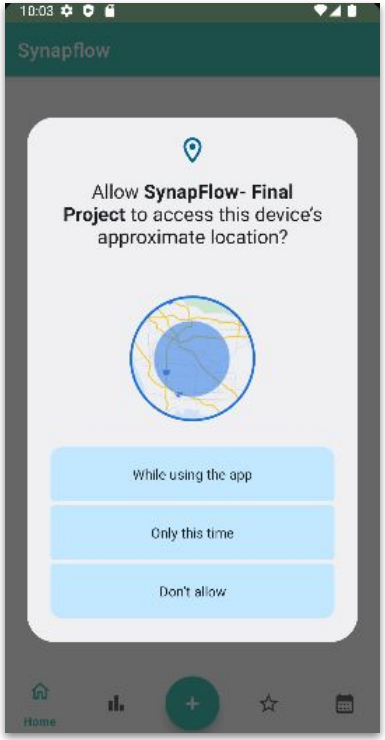
HomeActivity xml design

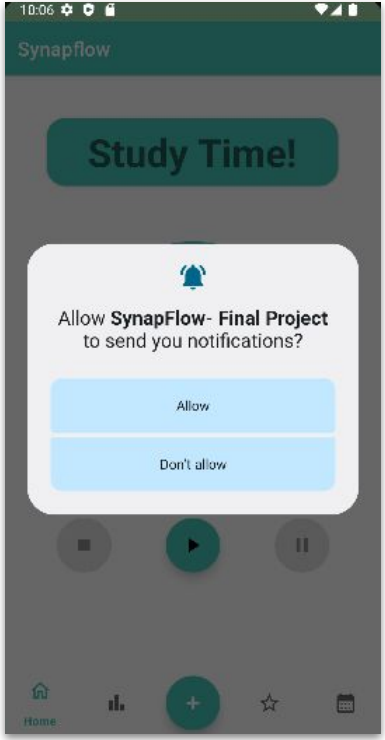
Bradley Tran - Permissions

Permissions

Query user permissions upon application access to help support user privacy by protecting access to the following:

- ACCESS_COARSE_LOCATION
  - Allow an app to access approximate location.
- POST_NOTIFICATIONS
  - Allow an app to post notifications.





Bradley Tran - Calendar

Calendar System

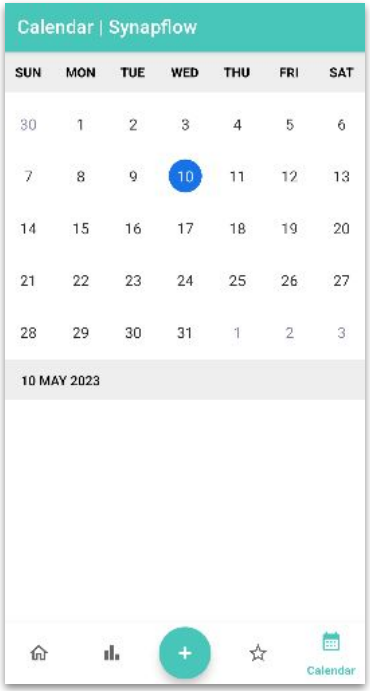
Implemented via open-source calendar library (kizitonwose) which provided a wider range of options for customization.

- Utilizes a subset of the java.time API for obtaining necessary calendar related values (daysOfWeek, currentMonth, date, etc.).

Challenges

Reviewing open-source library codebase in order to develop a greater understanding of bringing conceptual vision/ideas to an authentic functioning application.

xml design



functioning app

Innovation and Merit - Conclusion

- Our project was fairly simple all things considered but we were being realistic and simple.
- The app is well rounded, it completes what it sets out to do.
  - Help the user designate time and keep track of their events.
  - The app helps you keep track of your past study sessions.
  - The app is a simple study tool, allowing you to utilize a simple yet effective study/break interval.