Milestone 7 - Team 8: Focus

(formerly known as RMP)

Members:
Adam Tamariello
Akash Gaonkar
Byron Becker
Joseph Jennings
Luis Rodriguez
Nurin Syukrina Mustapha Kamal

Project Report

Goals:

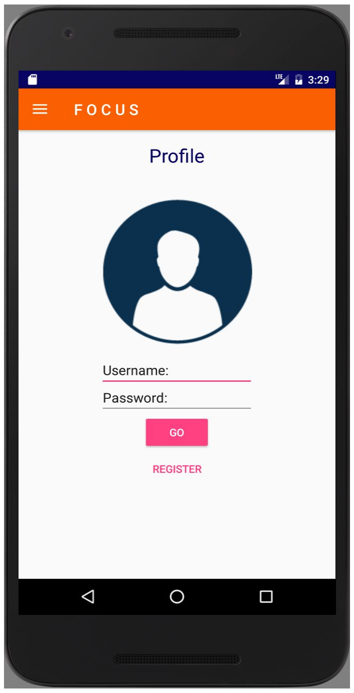
Clean, non-Cluttered UX/UI
Application Usage Access
Use Database to Store/Load Data
User Authentication
Maintainable Code

Accomplishments:

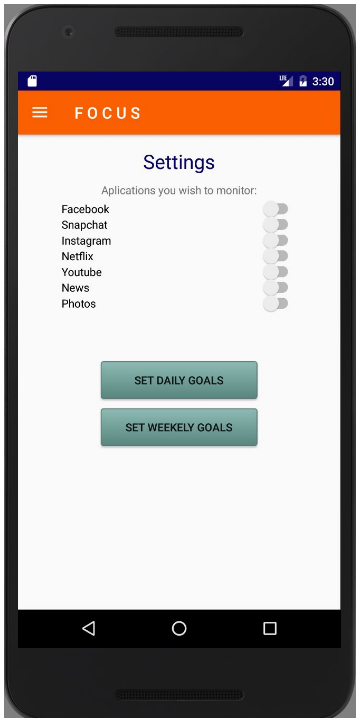
Clean, Non-Cluttered UX/UI:

We were very successful in this component of the project. We were able to create an easy to use, clean UX/UI, starting with our login page. From here you can go to either the weekly or daily usage graphs. We also have pages set up to set which applications you would like to track, and allows you to set both daily and weekly limits. We have the first time user to register their profile.
It is located in the “Focusing” Android Studio folder of our repository.

Login Page:



Settings Page (to set goals and application tracking):

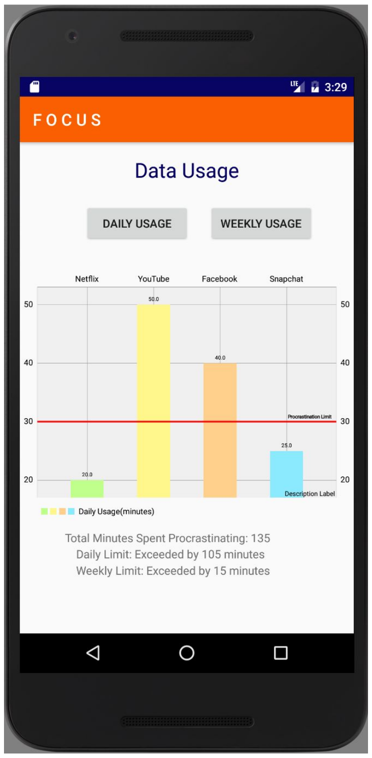


Settings

Aplications you wish to monitor:

Facebook
Snapchat
Instagram
Netflix
Youtube
News
Photos

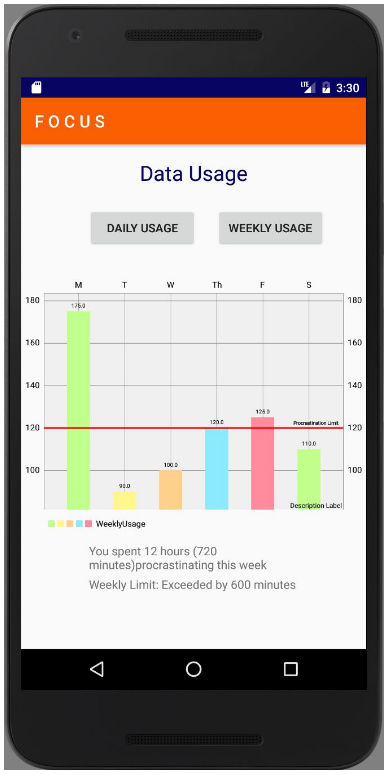
Daily Usage:



Data Usage

Total Minutes Spent Procrastinating: 135
Daily Limit: Exceeded by 105 minutes
Weekly Limit: Exceeded by 15 minutes

Weekly Usage:



You spent 12 hours (720 minutes)procrastinating this week

Weekly Limit: Exceeded by 600 minutes

Application Usage Access:

We were very successful for the most part in this component of the project. We were able to develop a background listener that could get access to the usage data even when the app wasn't open. We were able to use the Google API Usage stats to access how much time each application was used.

Database and Server:

We were successfully able to set up an ubuntu EC2 instance (virtual server) using Amazon Web Services. On this server, we set up a NoSql database using Couchdb, and installed a flask server to redirect API calls and correctly manipulate the usage data coming to and from the android client. We created methods that would be called when various api routes were taken, such as creating a new document when a new user was added, as well as a totalling function that would sum the totals of all applications for the day, as well as totals of individual applications. On the front end side, we were able to create methods of connecting to the server, as well as both PUT and GET methods. We also created a JSON parser to help access and transfer the data to the graphing functions.

User Authentication:

We were able to create and implement user authentication on the back end using the bcrypt module. Users could be created using a username and a password, which would be hashed with a randomly generated salt and stored (only the hash -- no plaintext). When a user logs in, it takes the password hash associated with their name and the password they entered and verifies the password (the password hash contains the salt appended to it, this salt is used to hash the password, and the two values are compared). If successful, logging in returns a basic authentication token. This will smooth the transition to OAuth, where authentication tokens are also used, albeit with slightly more complexity.

Outstanding Issues:

Usage data - The only part we weren't able to fully finish here was how often the background listener would run. We initially had it set for 2 hours, and had the interval shorten as the user got closer to their limit. However, we had to set the final interval to 10 minutes, as the closer it got to the limit, the more it would poll using more and more battery and cpu from the phone. We were not able to come up with a solution around this problem.

Authentication - There were issues when trying to implement and integrate authentication with the front end, especially since this complicating connecting to the database. We also need to use a SSL certificate and switch to OAuth.

Running Tasks Asynchronously - we have had errors pop up from time to time saying that too much of a load is running on the main thread, and are currently looking into the async library for android to help distribute the client load better.

Future Plans:

The first thing that we would like to implement in the future is the user authentication. We were pretty close with this during the project, and would like to finish it. We would also like to add push notifications to update the user on their progress. From here we would like to develop the application on different operating systems. We have plans to create another mobile application for IOS, as well as a windows application. A stretch goal for us would be to implement premium features such as focus (rewards) points and adding in user geolocation data tracking to see where you procrastinate the most or are most productive.

Project Reflection

We used a variety of methods of tools and methods throughout this process — this reflection will critique the use of them throughout the semester.

Agile Methodology:

We used agile development methodology very successfully throughout the semester. With our first two sprints being performed by the teams within our group, we were able to make really good progress in all areas of the project at once. The last sprint was a really good opportunity for us to troubleshoot the finish the various teams' tasks, and then integrate all of our work into one functional application. Unfortunately, we ended up underestimating the time it takes to integrate different areas from a project, and should have dedicated an entire sprint to this.

Slack:

Slack was our main communication tool during the project. We used it in our individual groups to communicate about what we were working on, as well as to update the group on any upcoming milestones. We also had several different main channels devoted to announcements, project discussion, and troubleshooting that all members could contribute to.

Google Hangouts:

This was a vital tool for our group as our team members were often very busy and off campus. Therefore, Hangouts allowed for everyone to contribute to our project retrospective, during our Thanksgiving meeting, and periodically on weeknights throughout the semester. The main feature that made Hangouts so appealing was the screenshare option that allowed us to share code and troubleshoot errors remotely.

Google Calendars:

We used trello for individual tasks, but utilized Google Calendars to create a broader scope of the project in making a project calendar. Our project calendar had more deadlines of when we wanted to get things done, as well as information on when all of the milestones were due.

Trello:

We underutilized Trello for the most part. We would add new tasks that we needed to work on, but were not very good at keeping it updated. We would mostly update the task all at once, instead of when they were actually completed. At the end of the project when merging parts of the project, keeping this updated more frequently would have been helpful to team members in knowing what roadblocks remained ahead of them being able to successfully merge their code into the application.

GitHub:

Because so many of us were new to GitHub, this is another area we could have used better. In the beginning of the project, we had a GitHub tutorial meeting to institute a formality by which we push to our forks, pull request, and then wait for 2 people to review the code before merging to the master. This however proved to be a hurdle preventing many from pushing frequently in the beginning of the project, and definitely impacted the cohesiveness of the branches and slowed down the project. For example, up until the very last week we had two different versions of the app on the GitHub, where it would have saved time if we were on the same page with GitHub. Although it was a bit of a struggle, I would say that everyone is significantly more familiar and comfortable with GitHub after this project than before it.

Amazon Web Services & CouchDB (NoSQL Database):

Although we originally planned on using Firebase, our database branch decided to venture out and host the database on our own private server. This allowed the database branch the educational opportunity to learn more about how server connections, creating apis, and NoSQL databases such as CouchDB work, while also allowing us more freedom and computing power than Firebase would. Although setting up the server and database took slightly longer time that anticipated, having our own server made everything much easier to troubleshoot. Couchdb proved to be straightforward, and although we have not scaled up the app, it has been very reliable thus far.

Flask & Python:

We implemented an api using Flask and Python to receive and forward REST requests containing JSON to and from the database. It took us two tries to find a library we liked in couchdb-python, but once found, implementing database interacting functions was not too difficult.

Android Studio, Java, & Xml:

We used Android Studio to create, format, edit, and evaluate our user interface and user experience. We made four applications in total, each of these had different layouts and different features, by comparison we chose the best two and worked on them until the end of our sprints where we voted on the best one that reflected our intention and initial idea of the interface. Through the use of xml to create our functions we were able to ignore the predetermined layouts and applications in Android Studio and were to mold and play with our own.

XBox DVR:

This was very useful when editing the video of our demo that we showed in our presentation. XBox DVR is an application on microsoft tablets and computers for originally recording when one is playing video games but we manipulated it to work with recording android studio.