# P5

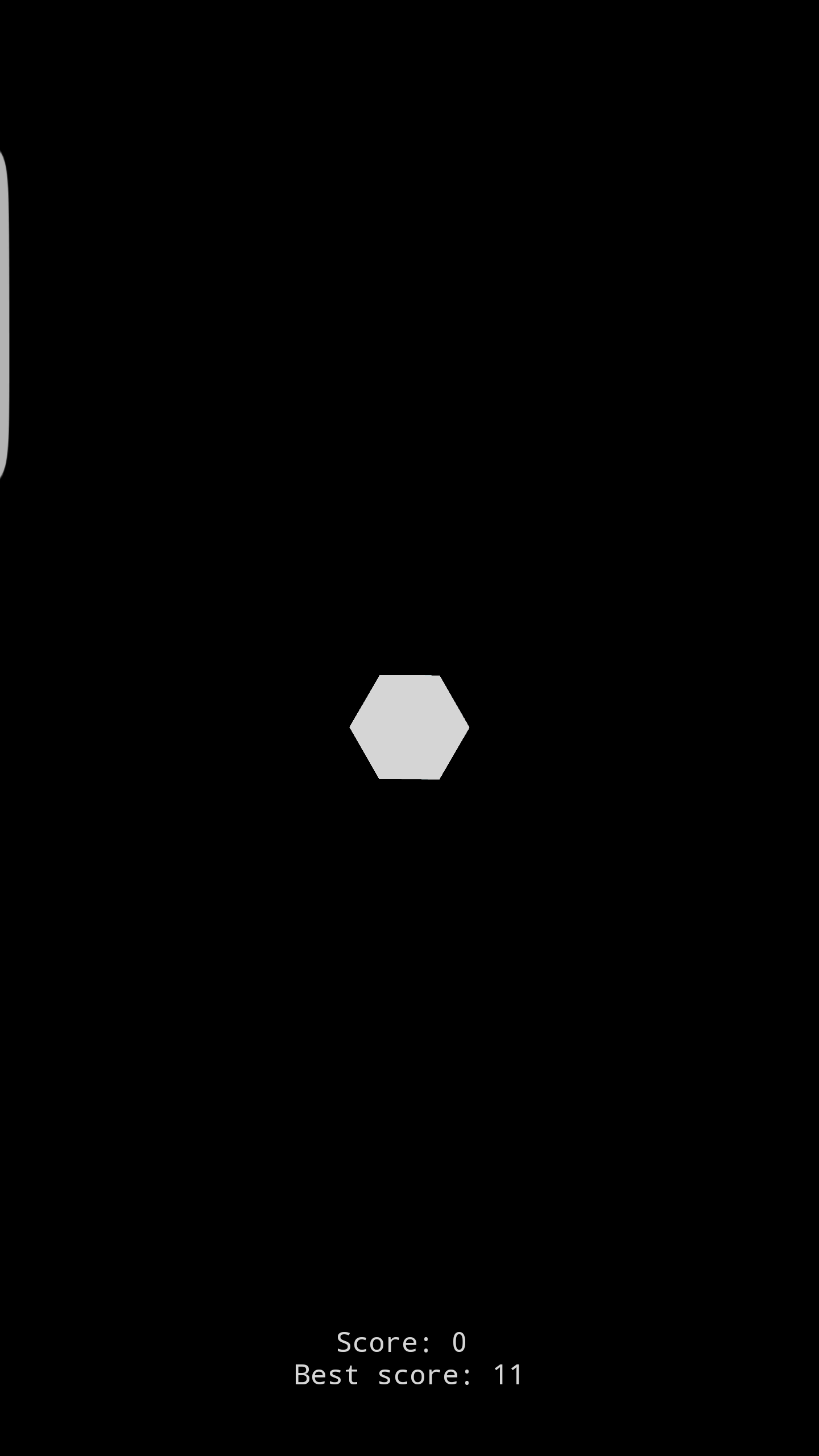
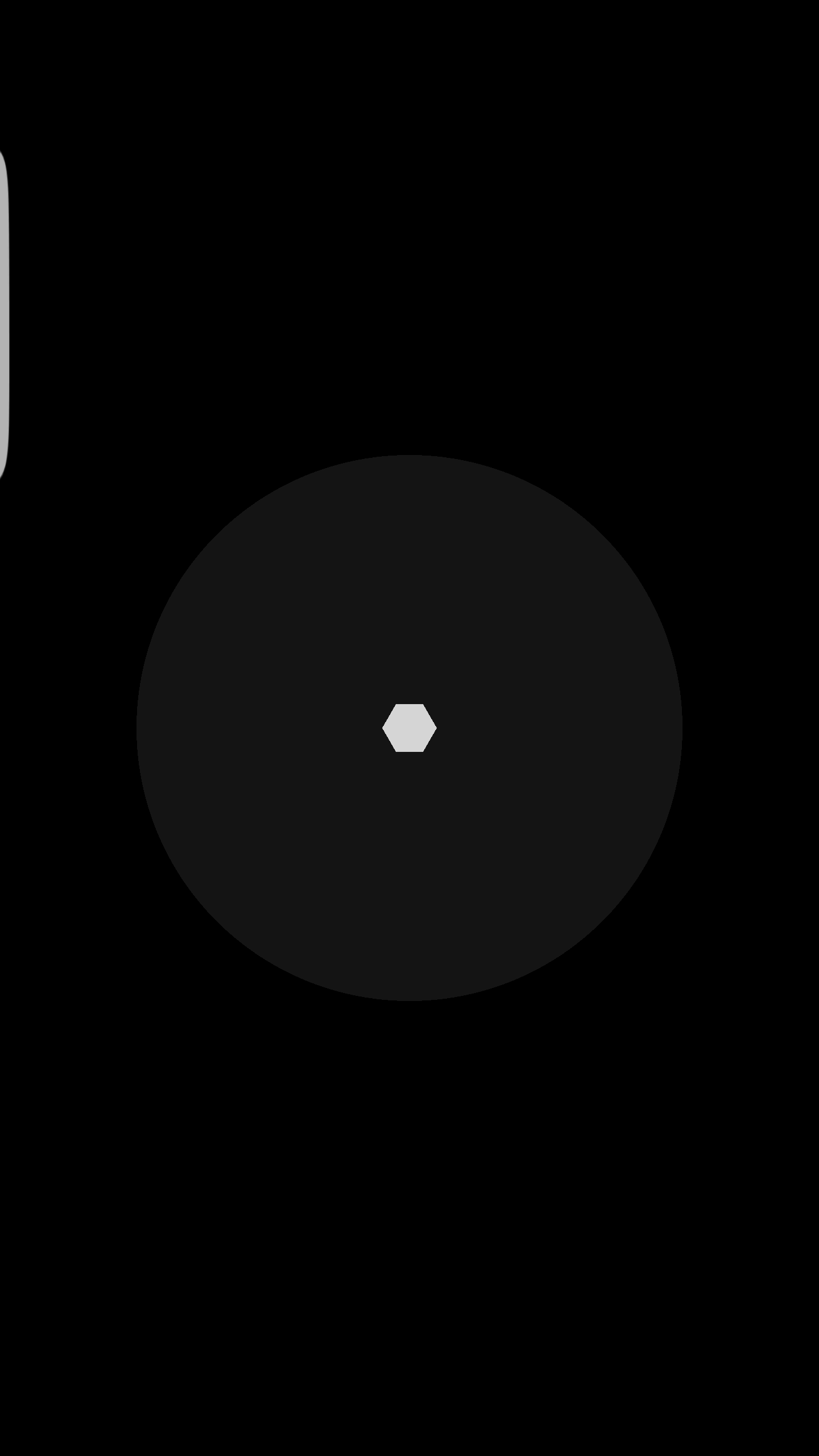
Ian Davis

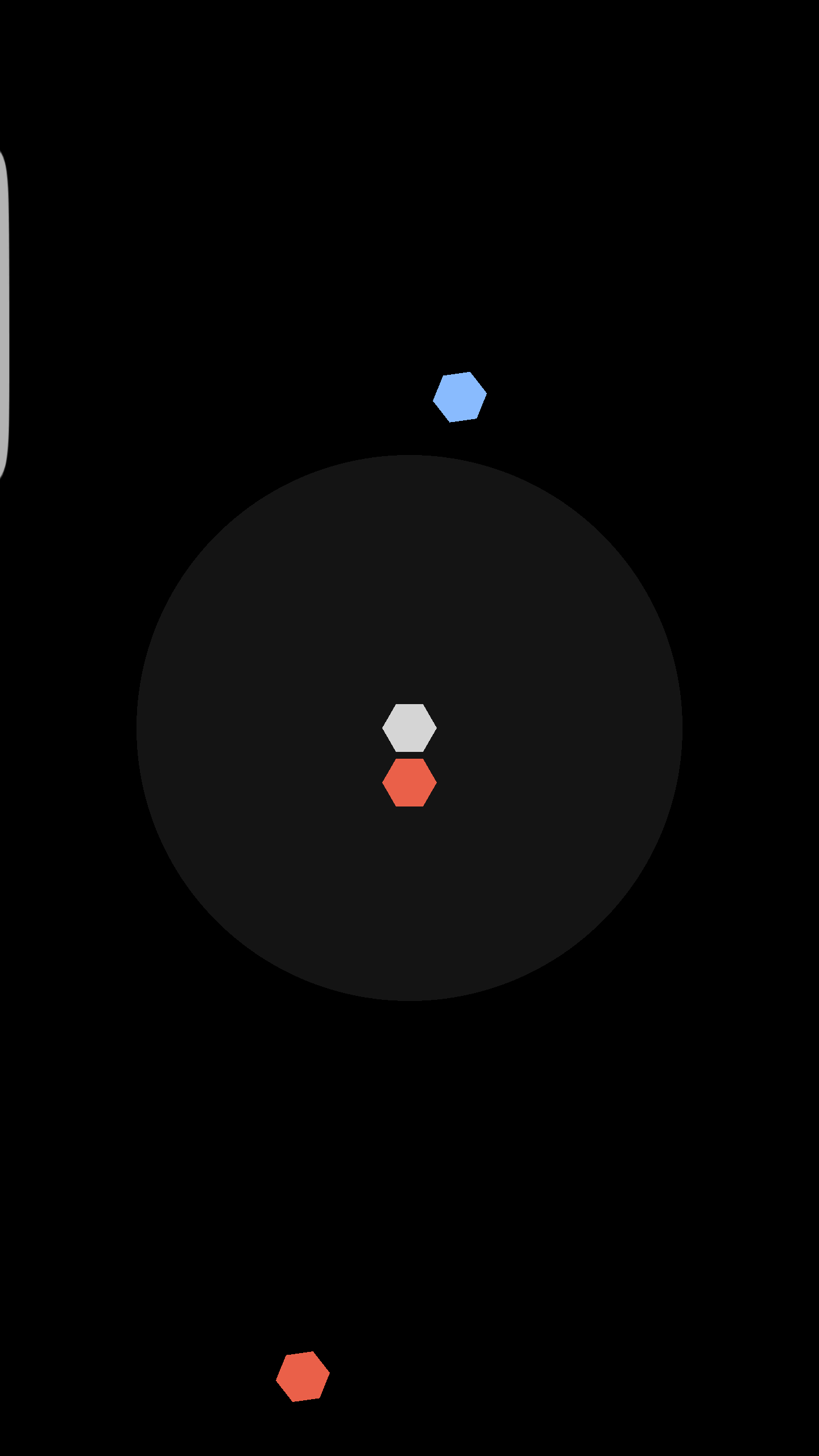
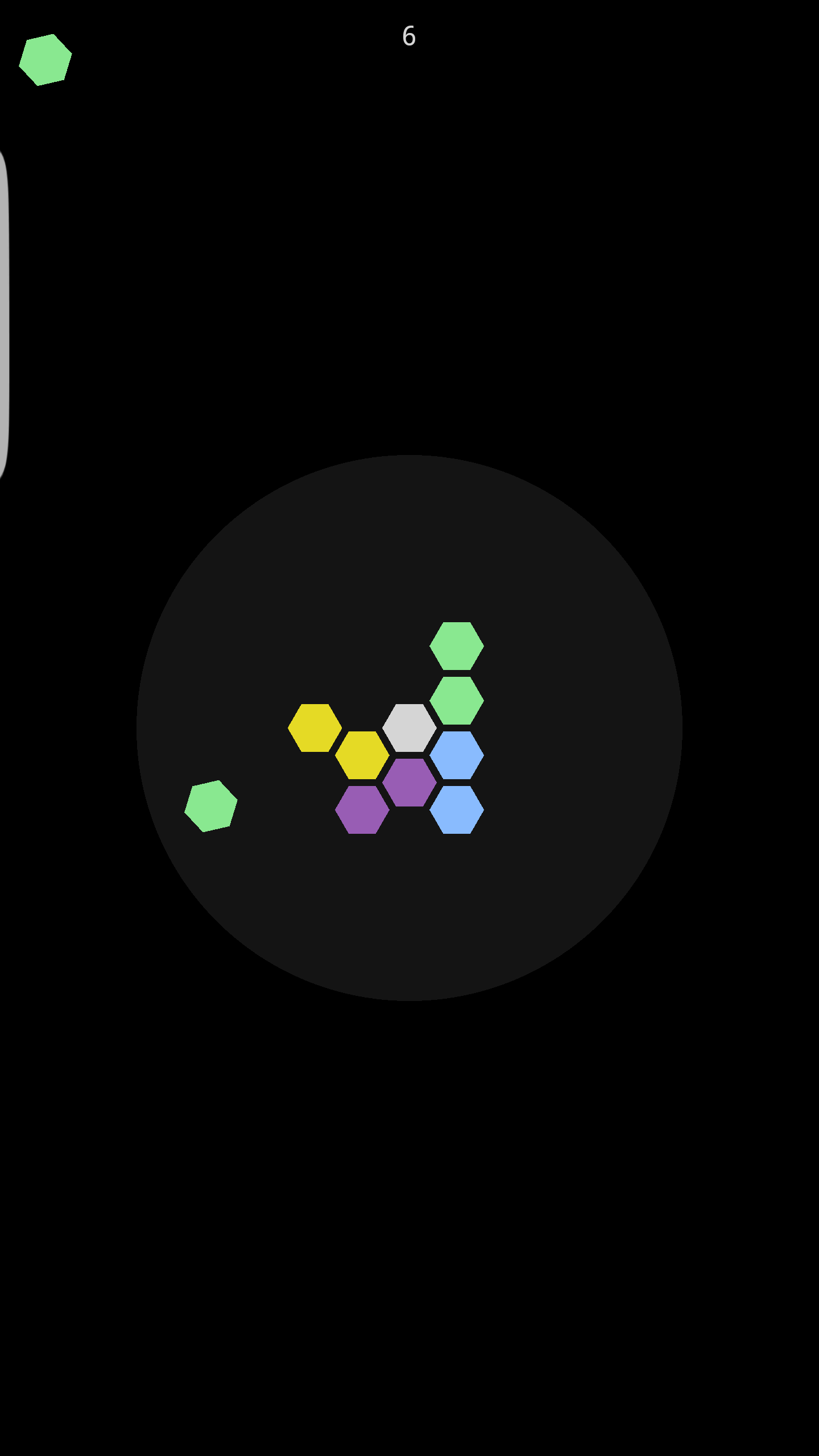
W002icd

Github URL: <https://github.com/IanDavis1995/P5>

# Task 1

* Screenshots pg. 2
* Status Report pg. 3
* Experience Report pg. 3

## Status Report

Some documentation may not be completely accurate, and some areas are lacking/vague in detail. Overall, mostly complete. The user’s manual is rather short, as the general functioning of the game seemed self-explanatory.

## Experience Report

Most of this code was easy to follow, however I struggled to put together the functioning of the GraphGameHexagon, GameHexagon and GameField classes at first. Some debugging/stepping through update calls shed some light on how this all worked together and helped flesh out some documentation on this code. Being thorough with this required a lot of effort. I did not do very much with the MyGLRenderer class and Hexagon base class as a lot of this was some complicated OpenGL code with vertex shaders and projection matrices and buffers galore. I gave it my best effort but was not sure where to begin with understanding what all was exactly going on besides a very high level overview of what it seemed to do.

# Task 2

* Screenshots pg. 4
* Status report pg. 5
* Experience report pg. 5

## Screenshots ../../../../../Desktop/P5-screens/photouploader/screen-main ../../../../../Desktop/P5-screens/photouploader/screen-preview../../../../../Desktop/P5-screens/photouploader/screen-uploads ../../../../../Desktop/P5-screens/photouploader/screen-after-upload

## Status Report

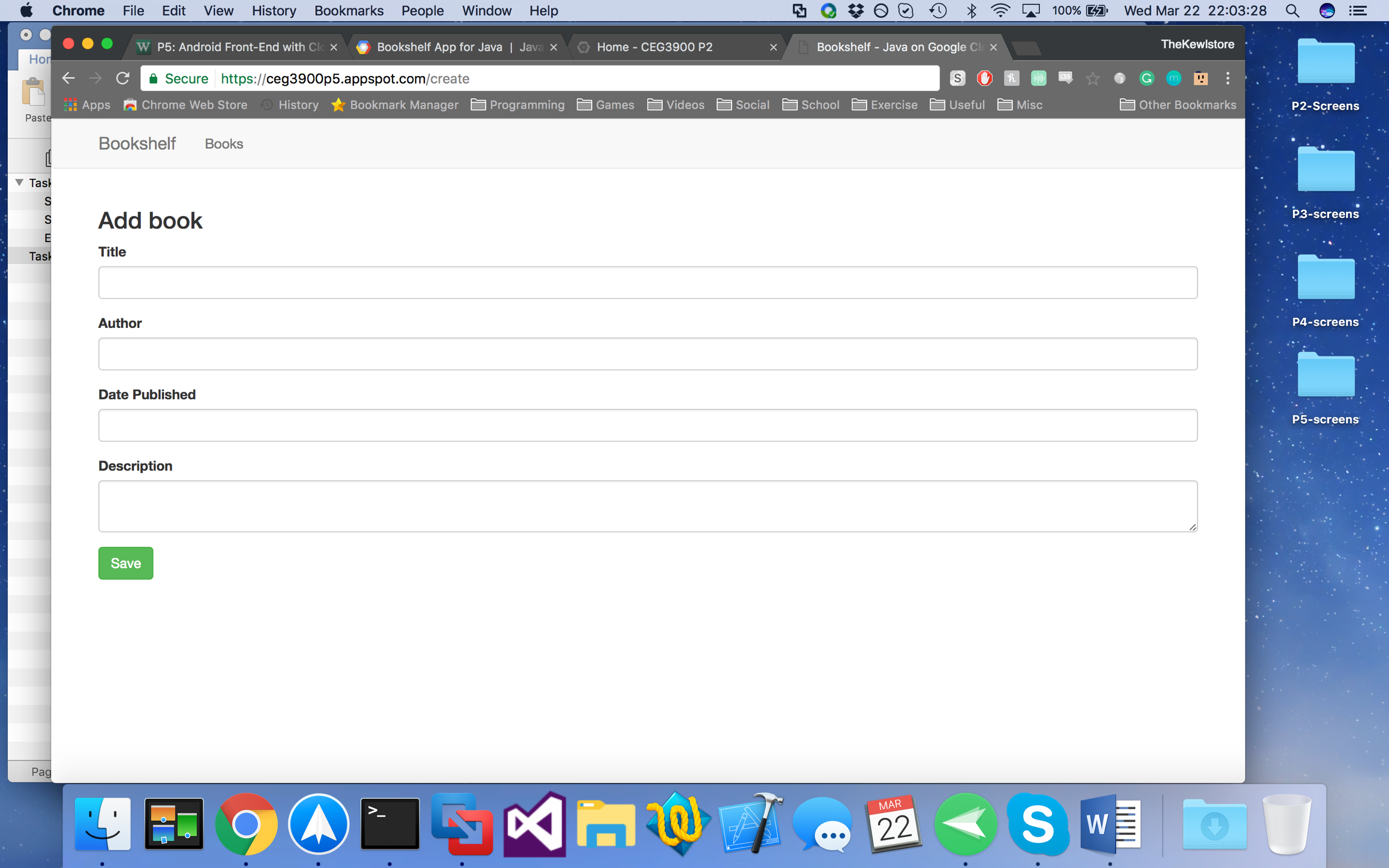
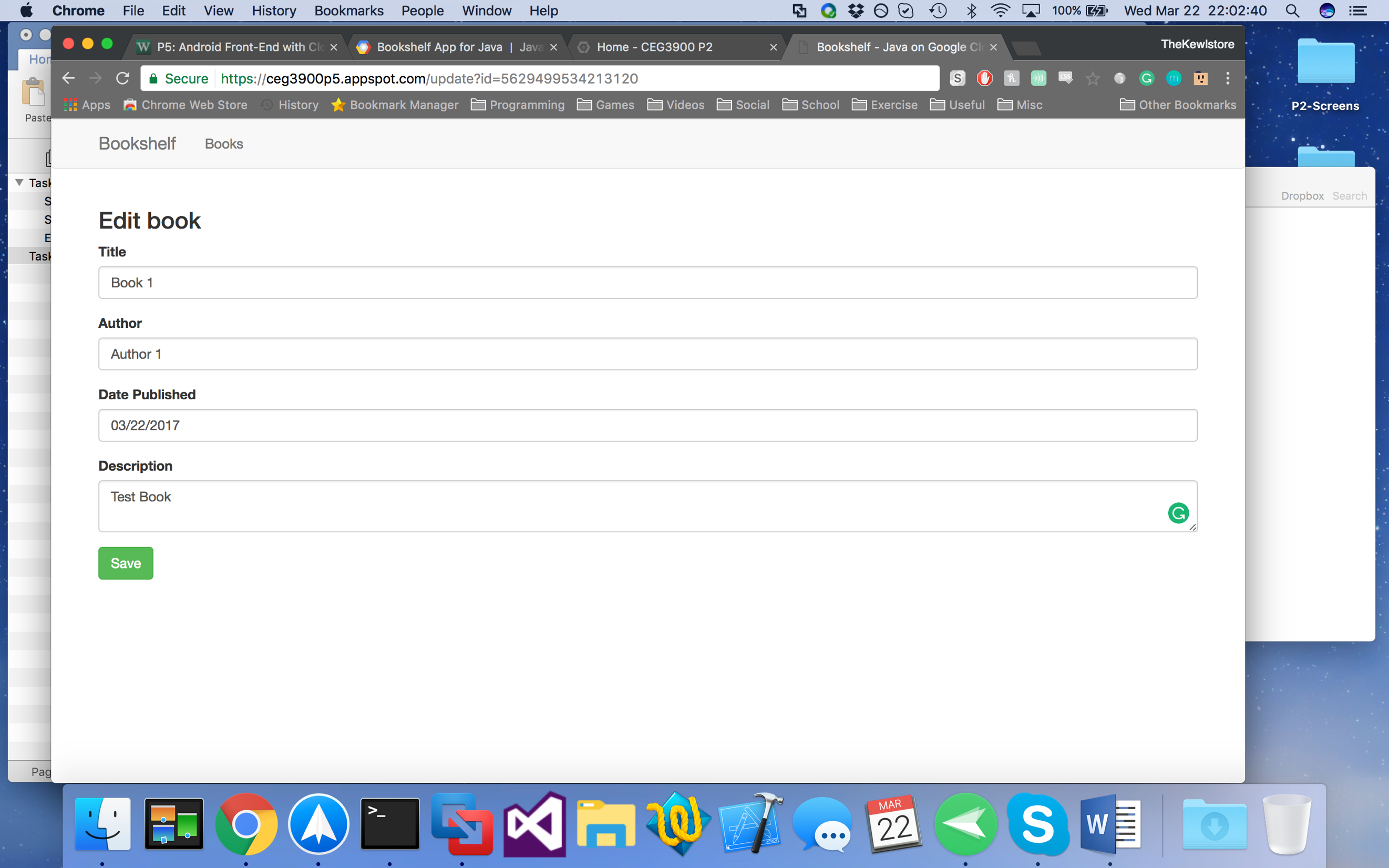
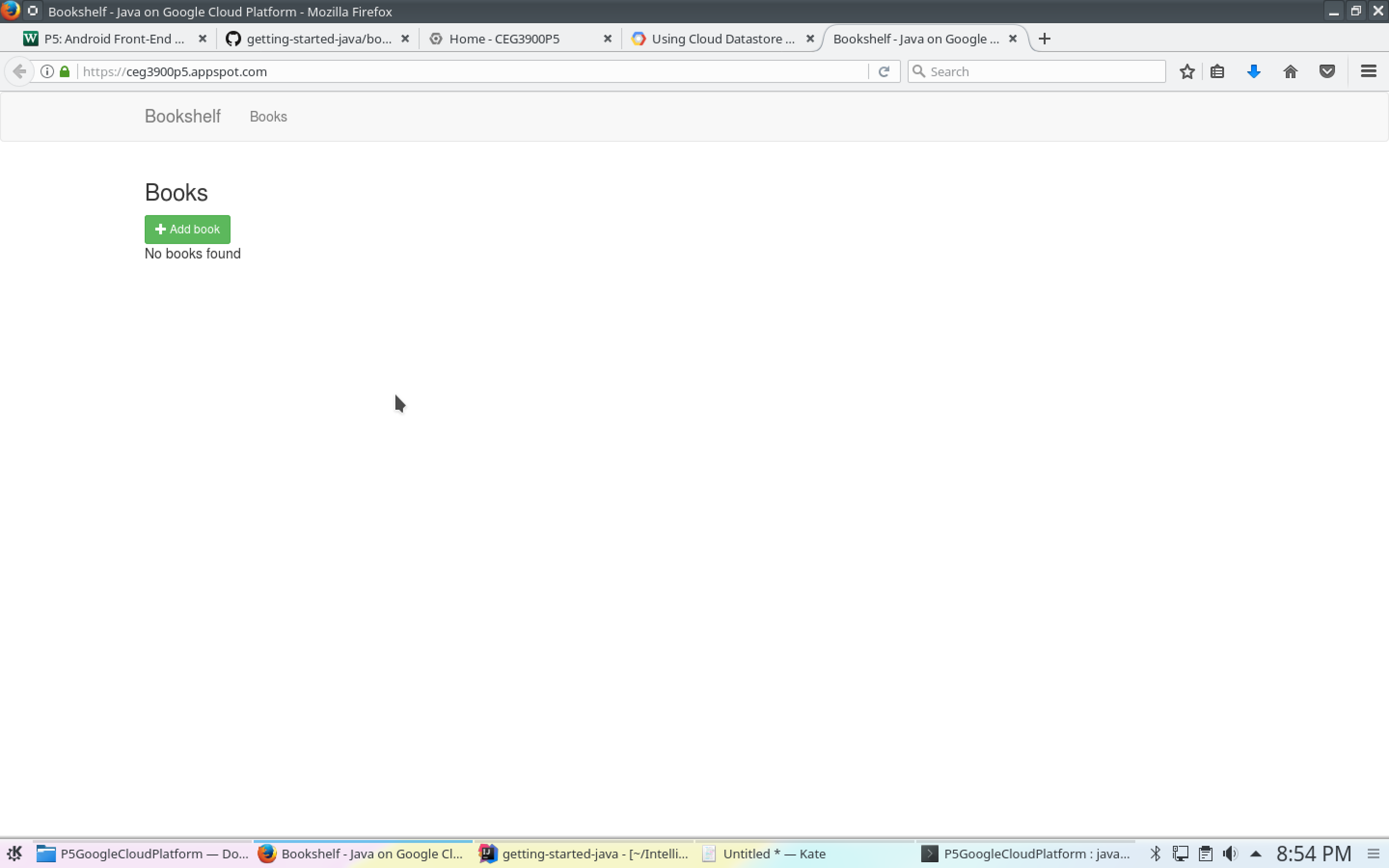
Fully Working, no issues found.

## Experience Report

Setting up this APK and the Azure Blob Storage was very easy and user friendly. The tutorial didn’t do the best job explaining that the ACCOUNT refers to a Blob Storage account on the Azure Portal, but it wasn’t difficult to figure out.

# Task 3

* Screenshots pg. 7
* Status Report pg. 8
* Experience Report pg. 8



## Status Report

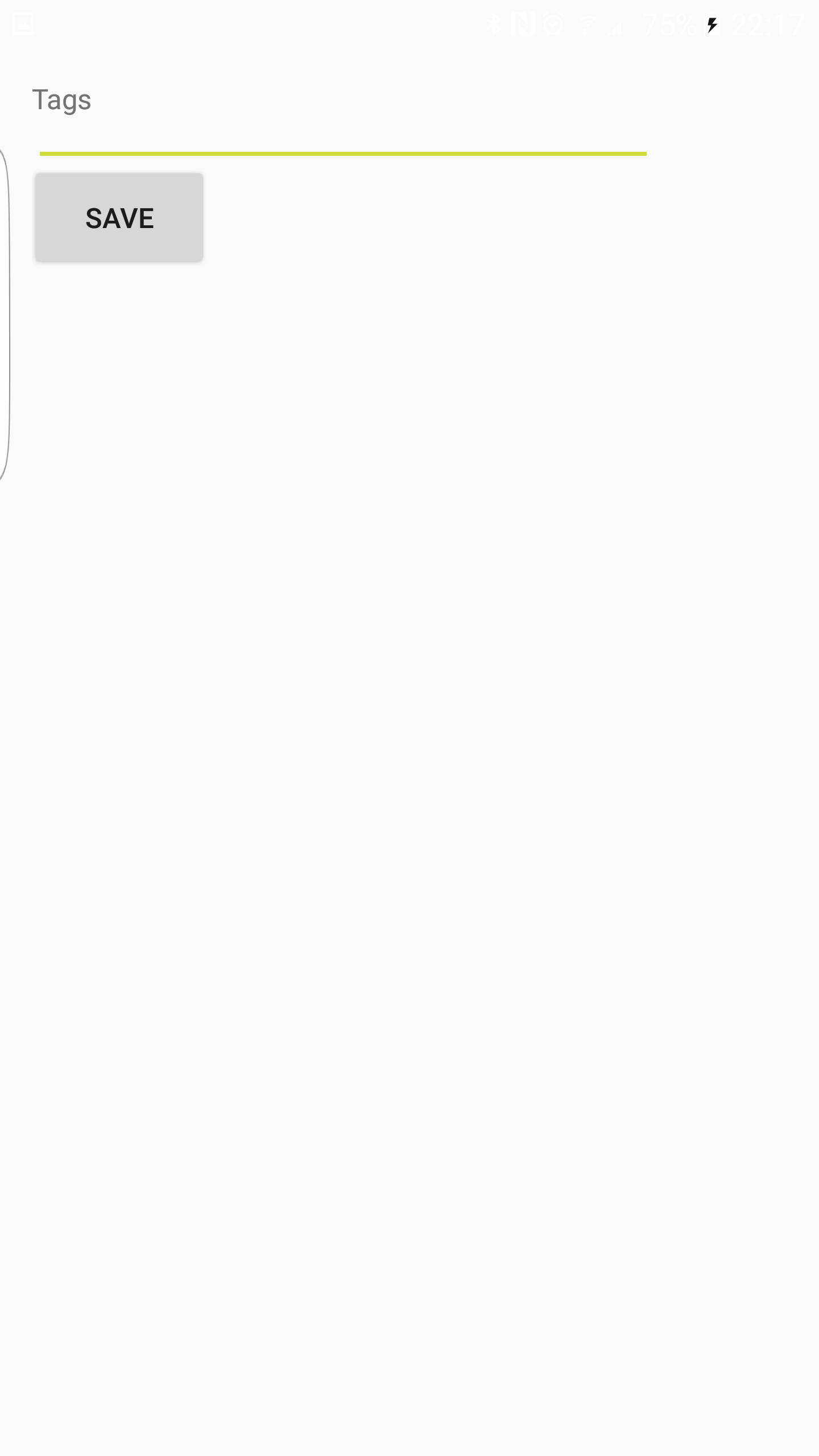
Mostly working, but images cannot be added to new books.

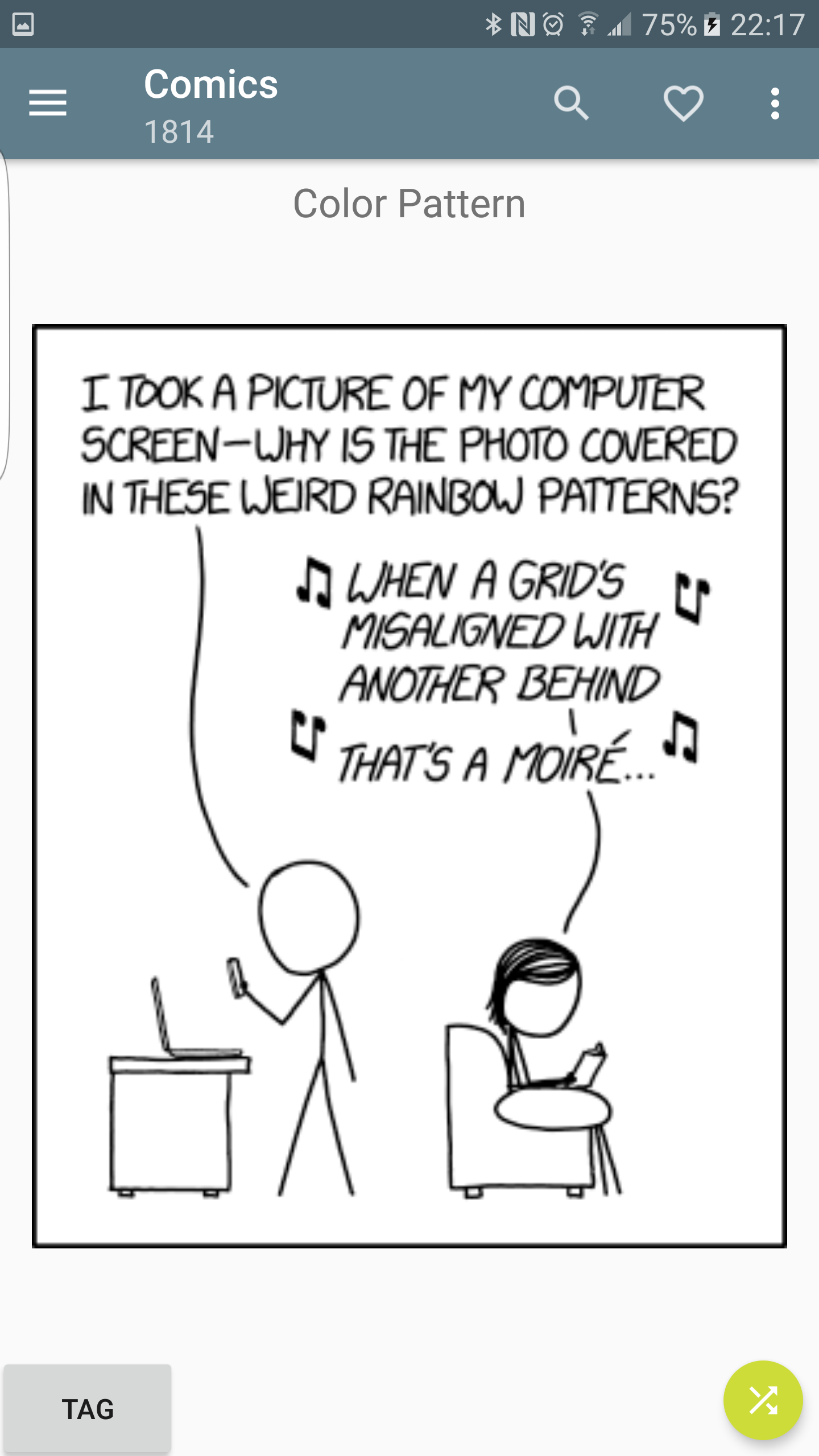
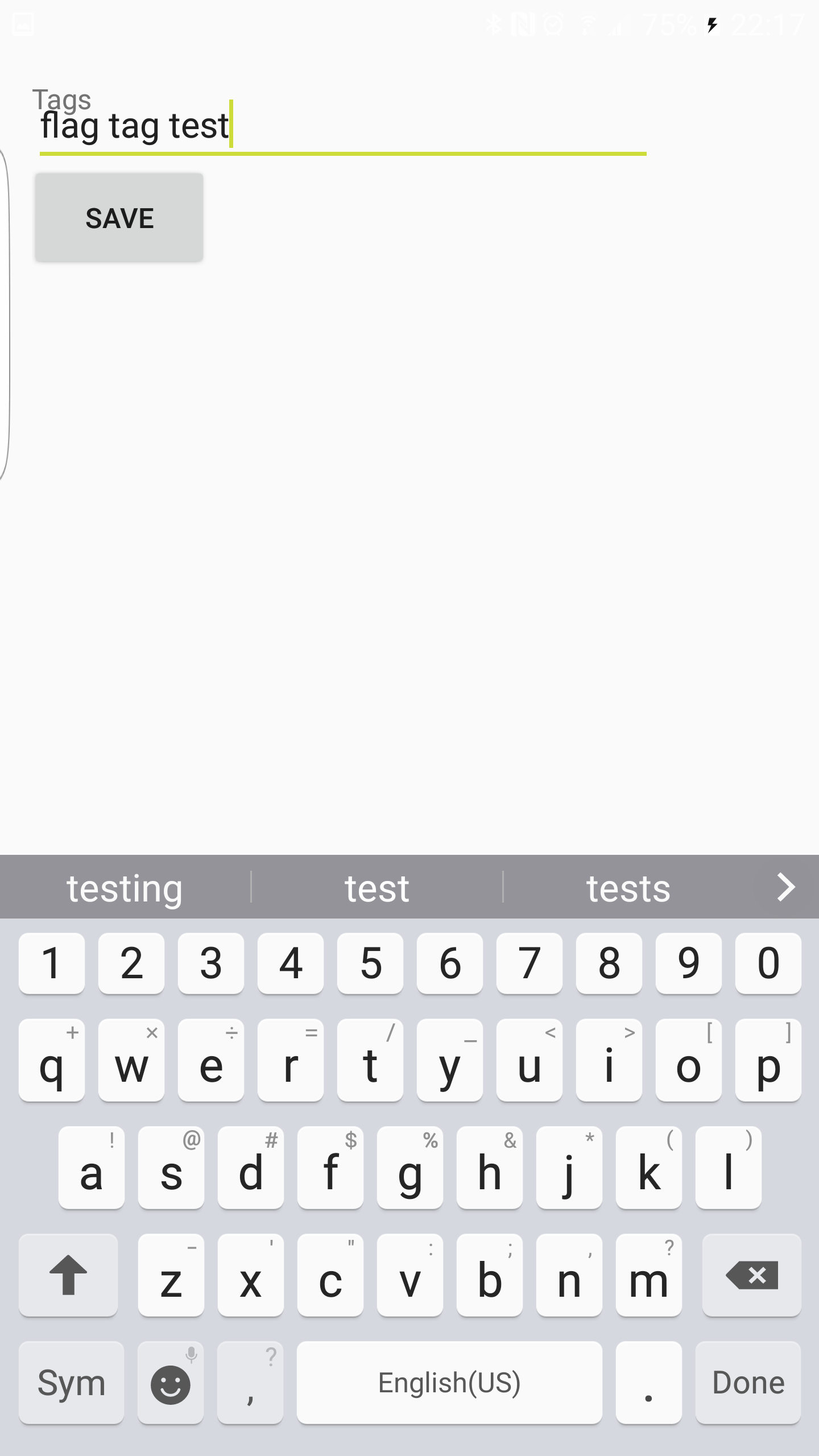
## Experience Report

I struggled understanding how to get the permissions of the Blob Storage with the Client ID and Client Secret, the only way I could manage to successfully create new books was to set the default permissions to read/write. I also couldn’t get the application to deploy locally, as I kept getting a weird chain of errors. Deploying it to the web worked throughout, however.

# Task 4

* Screenshots pg. 10
* Status Report pg. 11
* Experience Report pg. 11

 ****

## Status Report

Mostly Functional. Tags are stored as a string for each comic id and loaded in when the tag button is pressed on a comic. They are saved when the save button is pressed. Tags are only stored on a per comic level, and comics cannot be looked up based on tags.

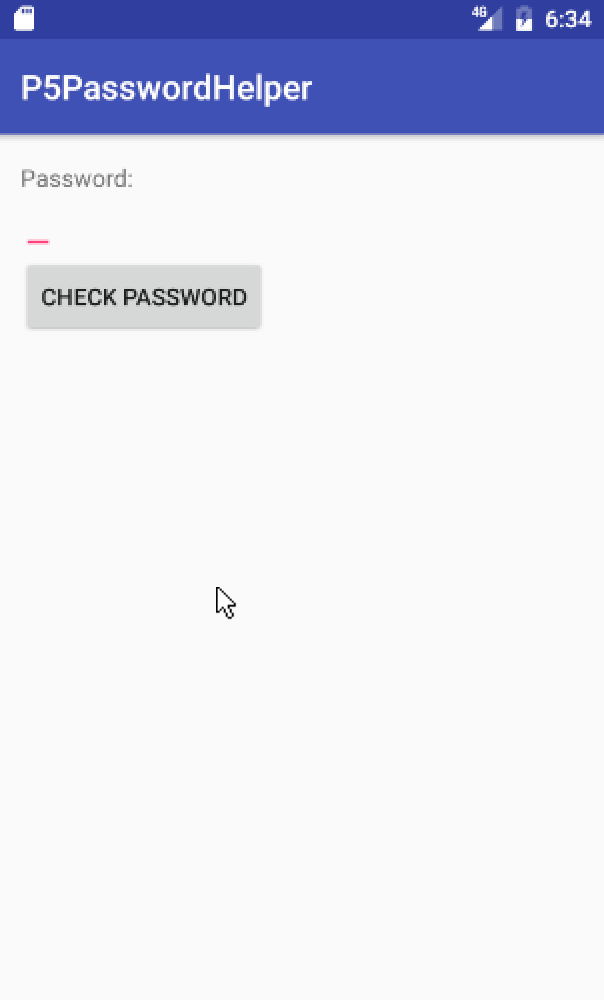
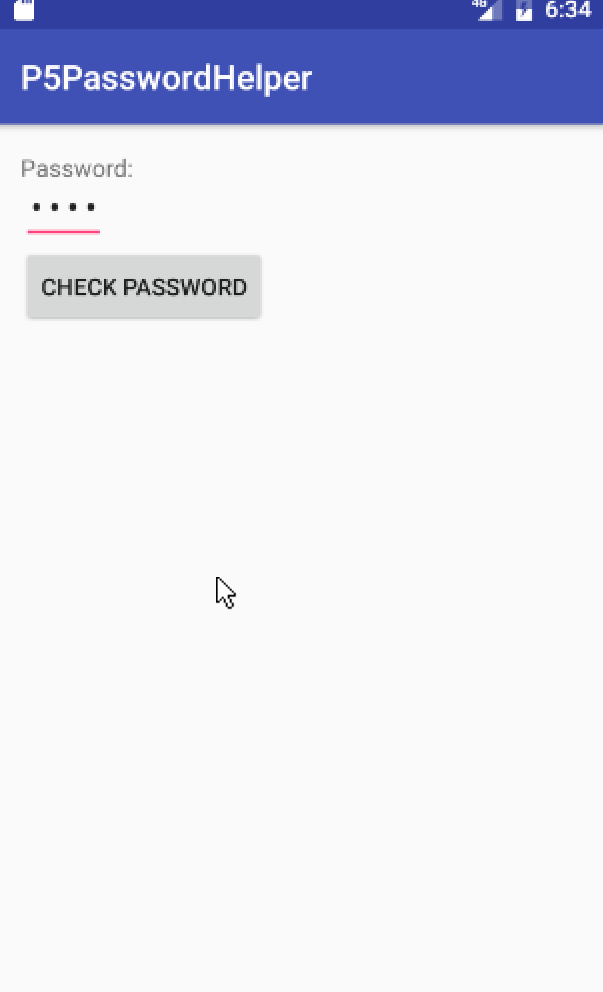
## Experience Report

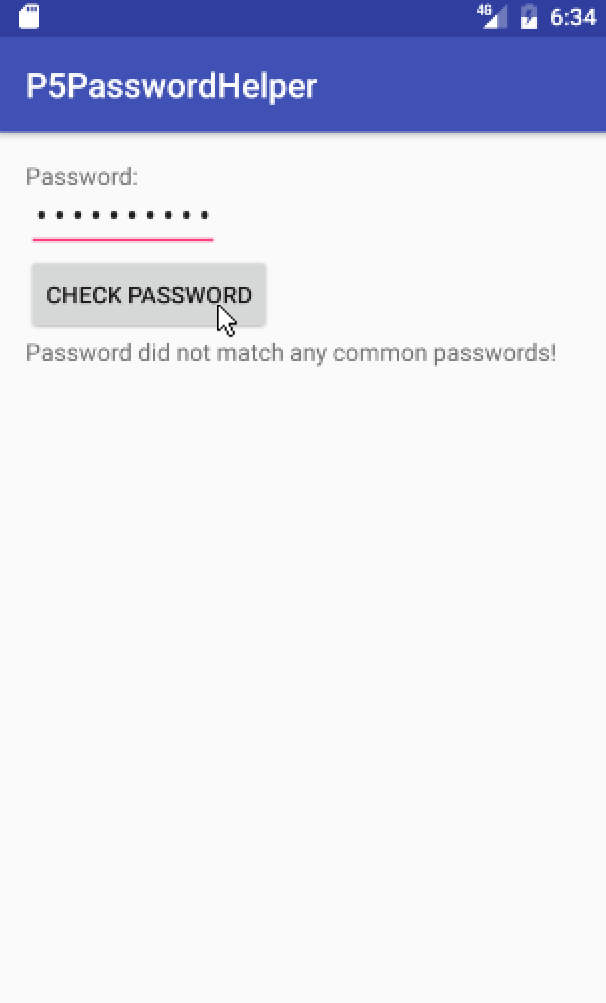
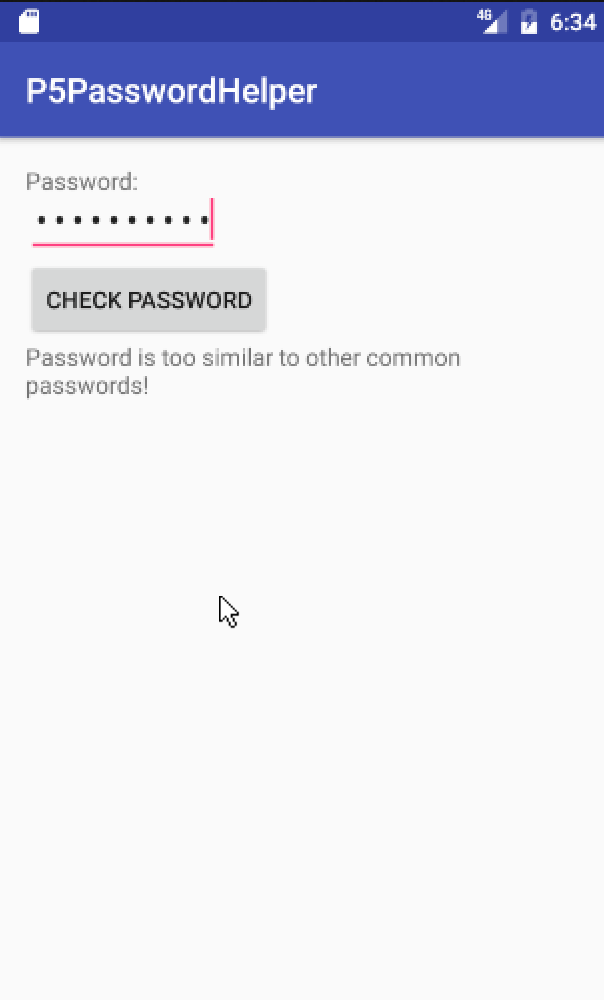
I had some trouble with getting the Easy XKCD Source up and running. The Android appcompat library was complaining about MenuItemCompat.setOnActionExpandListener being deprecated, even though all online references suggested using exactly that. I’m not sure if this was because of versioning or what, but ultimately, I had to comment out these calls. This did not seem to alter any noticeable behavior of the options menu, however. After these build issues were resolved, this task was straightforward. I chose to implement it with a Firebase database, storing tags as a string mapped to their xkcd comic id. Other than some issues getting the firebase libraries added into the easy xkcd project properly, this was a pretty smooth implementation. I struggled slightly to figure out how the xkcd viewer implemented the actual comic viewing pane through its fragments, as it was far from a trivial implementation of the UI.

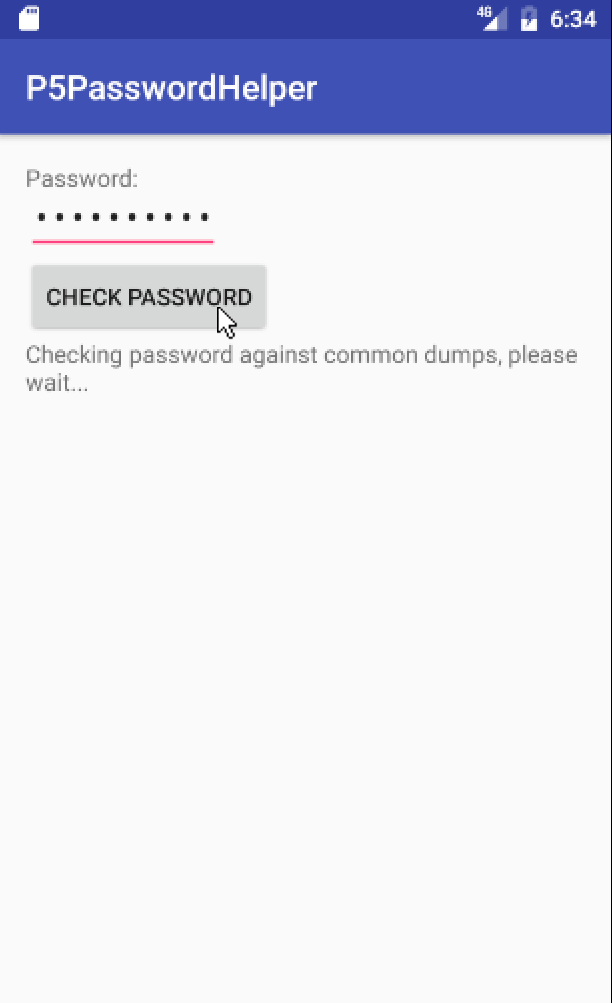
# Task 5

* Screenshots pg. 12-13
* Status Report pg. 13
* Experience Report pg. 13-14
* Definition of Similarity pg. 14
* Description of APK Benefits/Help pg. 14

## Screenshots



## Status Report

Functional, not well optimized. Worst case computation time is about 6 seconds for a password with no commonality. Some of the most “common” phrases/words will be caught quickly, however.

## Experience Report

I implemented this app using a google app engine HttpServlet backend, and sent Http get requests along, passing the password entered from the android application. The backend had a resource file of the top 1 million common passwords taken from the SecList github repository. It parses this file, reading in each password and checks for similarities based on the rules defined in the Definition of Similarity section of this report. I had a lot of issues trying to get the app engine servlet up and running, and had to hunt down lots of tutorials of how to set everything up properly. I considered several different options for implementing this in the cloud, but decided on google app engine servlets as it seemed the most flexible to fit this specific need. The other main choice was Amazon EMR with an Apache Hadoop job, but I was even more lost on where to begin with that implementation.

## Definition of Similarity

For this APK, I defined a password as being too similar based on the following rules

(NOTE: For this list, a “common password” is used to mean a password found in the common password dumps available in the SecList github repository):

* A common password exists such that 60% of its characters exist (in the same order) as the password given
* Said common password is no longer than the password given
* Said common password is no more than 3 characters shorter than the password given.

Several different tweaks of these rules were tested, but this gave the best balance between being decently “tough” on password uniqueness, but also not matching any conceivable password possible, without getting into some much more complex comparison algorithms.

Description of APK Benefits/Help  
Beyond simply giving suggestions about the security of certain passwords, this app could also be useful to determine if a password already in use were already cracked or compromised on a public password dump. As exploits like this are increasingly common, an app that could scour some of the most common password dumps and determine if your password is already cracked, or even further, how easy it would be to crack, could be invaluable to end users.