My project is an app that is basically a really simplified version of a game called cookie clicker. The premise of my app is there’s a button and every time you click the button, you get a coin. You can then spend your coins on upgrades that help you get coins faster. There is no end goal other than getting as many coins as possible.

My app, like any app, starts with a storyboard. The storyboard is where the GUI is built and has a drag and drop editor. There are two different views in my storyboard. Almost all of my elements are either buttons or labels. They are embedded in a navigation controller, which is necessary when having multiple views. The ViewController (VC) is connected to the main view. To connect an element to the VC as an action or an outlet, the VC and storyboard must be opened up side by side and the element must be dragged to the VC. When the VC loads up the main view for the first time, it runs viewDidLoad, which it sets the click multiplier to 1, sets colors in the navigation bar, initiates a notification center that will be necessary for updating the bank text later, and shows an alert box with instructions. Every time the main view loads, it runs viewWillAppear, which updates the bank to the correct number from the other view controller, sets a control variable to true, and resets a variable that is used by the other view controller. When the main button is clicked, it increments the number of clicks by 1, increments the number of coins by the correct number (which depends on which upgrades have been purchased), and updates both labels. Many of the variables in VC and the menuViewController (MVC) are initiated in structures. Initiating these variables in structures allows the other view controller to access them. Clicking the menu button takes the user to the menu, which has a back button that takes them back to the main view. This transition does not exist anywhere in my code, but is instead something that was done using the storyboard editor. In MVC, viewWillAppear is unnecessary because viewDidLoad runs every time the menu loads. viewDidLoad switches the a control variable in VC to false, creates an array of all of the upgrade buttons, sets the MVC bank to the correct number of coins, checks which upgrades to show and hide, and initiates a timer that checks whether any timers are running in VC that would give the user free coins. When the view disappears, viewWillDisappear runs and invalidates that timer. When the first or third upgrade button is clicked, the function first checks to make sure the user has enough coins to buy the upgrade before proceeding. If the user does, that many coins are removed from their bank, the click multiplier in VC is updated accordingly, the bank label is updated, the button is hidden, and the right element in an array is set to true to indicate that the button has been used. The process for the timer buttons is identical except that the each initiate a timer in VC when clicked. Once a timer is initialized, it does an action one per interval until invalidated or paused. One timer in VC runs every 5 seconds and the other every 10 seconds, but both timers are otherwise identical. They increment the number of coins in the bank by one and then move into if/else statements that are dependent on a variable. If that variable is true, the bank label is updated. If the variable is false, they increment a variable by one. The timer in MVC checks the incrementing variable and increases the number of coins in the bank there if the variable has increased since it was last checked.

Please note that Xcode can only be run on a macOS and that Swift updates when Xcode updates. I developed using Xcode version 8.2.1. My project was developed for and tested on an iPhone 4 running ios 7.1.2, however, the app should work just fine on any ios device that is newer than that. Xcode 8 only supports ios 8 or later, but since the iPhone 4 can’t run any ios newer than ios 7, I had to manually add support for ios 7 so that I could run the app on my older iPhone. I did this by going into Xcode⇒Contents⇒Developer⇒Platforms⇒iPhoneOS.platform⇒DeviceSupport and adding the folder for ios 7.1 (which can be downloaded online).

To run the app, open the project file in Xcode and press the run button, which is in the upper left hand corner and looks like a play button. Xcode has a built in simulator for various ios devices. The user can select which simulator they would like to run code on by using a dropdown menu to the right of the play and stop buttons (I recommend using iPhone 5 for my project since there is no iPhone 4 simulator in Xcode 8). If an ios device is plugged into the computer, that device will appear on the dropdown and the user can select it as the build target. When the code is run, Xcode compiles the project, then builds the project, then runs the project. When an error is thrown, the user is told whether the error occurred during compile, build, or runtime.

In the future, I would like to add a bunch more upgrade buttons, hopefully in some sort of scrollbar such that purchased buttons vanish and the other ones move up to still fill the space. Better graphics would probably be nice too. The concept for this app involved running a theme park where clicking attracted a guest and there was an animation of the guests lining up for rides in the background, but that ended up being beyond what I was able to implement, so I think I’d like to do that in the future. It would be great to get all the upgrades themed around that too, so like the free coins every n seconds could be different advertising campaigns and the ones that increase clicking power could be purchasing new rides for the park. In terms of more advanced development on this app, many other incremental games have a soft reset where you start over, but get a bonus for how many resources you had during the previous game (cost of upgrades tends to increase exponentially), so that would be cool to implement, but wouldn’t make sense to add until I’ve added a lot more upgrades. If I did that, I think that would be an option to expand to a second location for your theme park, so you get some bonuses due to name recognition from your other ones, but you still have to start over building rides and advertising.

Demo video: <https://youtu.be/rKnCxpWNbfY>