Trey Stone

CS 441

1 December 2019

Notifi Messaging App Report

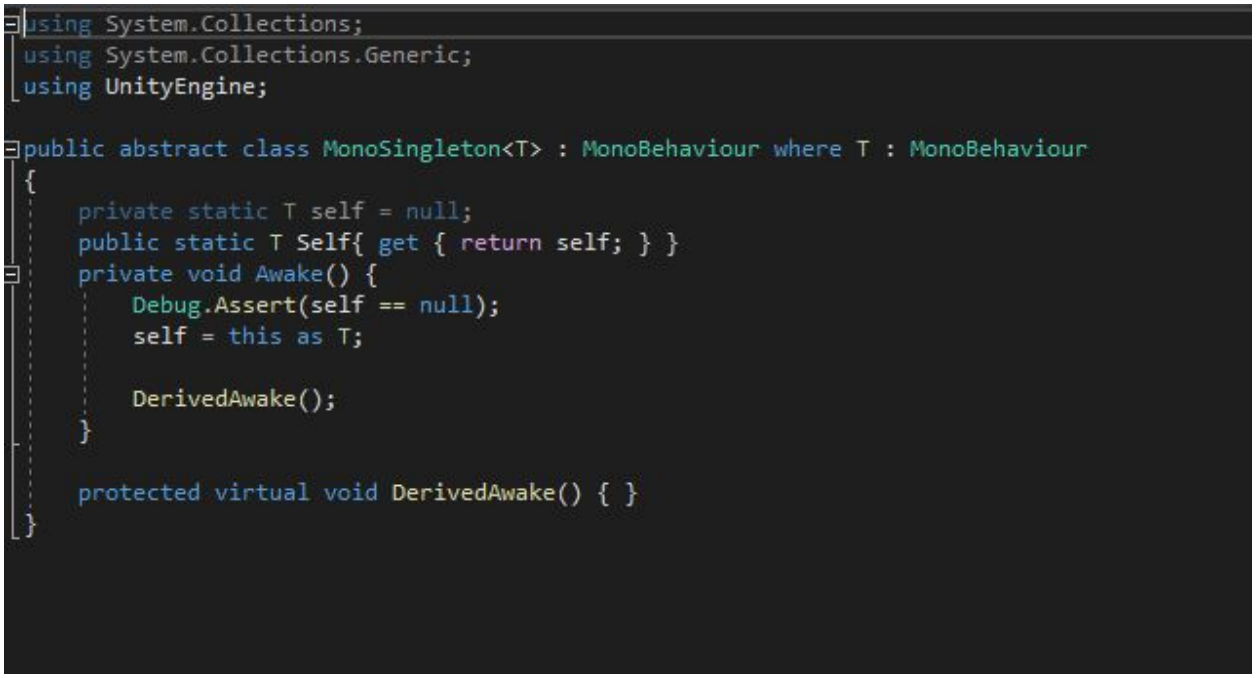
When we originally decided on making an app for our semester project, Unity became an obvious choice due to its high flexibility toward different development environments. Additionally, it's a program that I have substantial experience with, originally beginning back in high school. Most of what I'd learned before this semester had been self-taught, with a heap of Stack Overflow posts being the closest thing to formal education I'd ever received in Unity development. This project served as an excellent opportunity to focus on aspects of programming I was not as experienced with, particularly network programming and UI design. I had only ever tentatively used these systems in Unity, so I started with my base understanding of how the app *should* work, then researched how to make that possible with the tools available. Everything ended up as I had hoped, with only minor hiccups along the way.

Our original app was a group-messaging app that specialized in quick multiple-choice selection of predetermined responses. I had originally thought of this over the Summer during my internship after realizing I'd typed up and sent "Just got off. On my way." about a hundred times. Having this as a prebaked message that could be sent with a simple gesture or IFTTT-style patterns would be a notable convenience. We discussed the idea as a group, and brainstormed a number of potential uses, including class groups, contractor work, and any number of social groups. Over the course of

development, I proposed small changes that would simplify the app into something more manageable for the semester. Through this work, I designed and implemented several use-cases that are core to any group-messaging app. Among these are the ability to create groups, name and edit them, and send messages to other group members. Despite being completely obvious for a messaging app, their exact specifications and implementation ended up setting the frame for the rest of the programming I did this semester.

While developing the core framework of the app, I determined an efficient design of how to structure our groups, accounts, profiles, and messages. All of these were presented to the group, at which point we drafted the UML design for each of these specific features. This early finalization of the design was extremely helpful in preventing unnecessary refactoring of the app's framework as I pieced it together. As the lead programmer, I drafted and implemented most of the framework and features of the application client.

I have learned numerous tricks from my years of working with Unity, so I was able to apply a lot of this knowledge to producing higher-quality, more sustainable code. Many of these habits were also utilized when practicing with the systems I was less familiar with. This made learning and practicing UI and networking developing considerably easier. My PageController classes handled all of the UI heavy-lifting, while the use of my various Manager singletons controlled everything from instantiation to network interfacing. Leveraging Unity's robust tools for animations, graphics, and coroutines furthered simplified the overall design of the app.



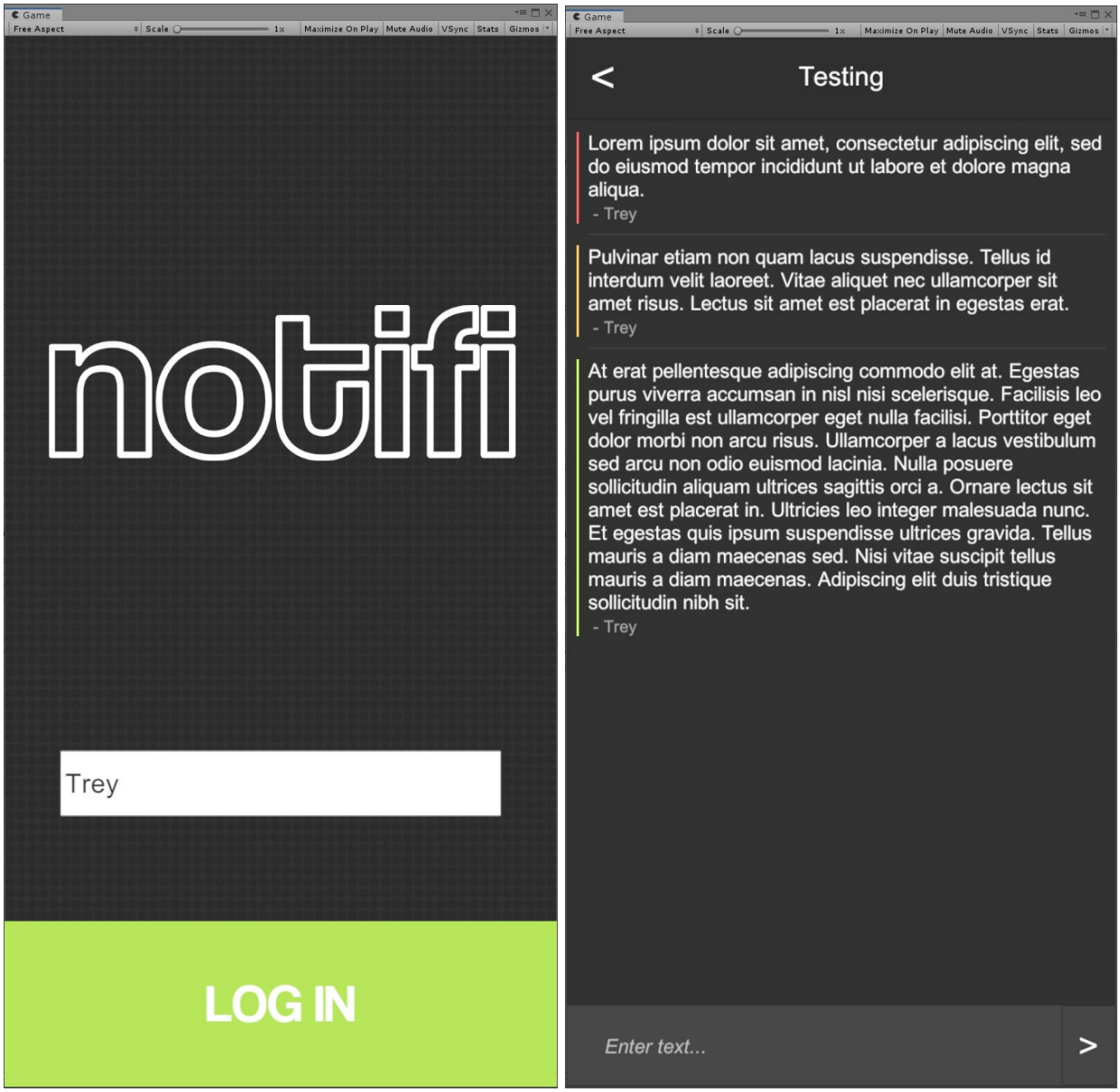
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public abstract class MonoSingleton<T> : MonoBehaviour where T : MonoBehaviour
{
    private static T self = null;
    public static T Self{ get { return self; } }
    private void Awake() {
        Debug.Assert(self == null);
        self = this as T;

        DerivedAwake();
    }

    protected virtual void DerivedAwake() { }
}
```

The MonoSingleton base class provides singleton design to Unity’s MonoBehaviours



Screenshots of the Login screen and a group’s message viewer