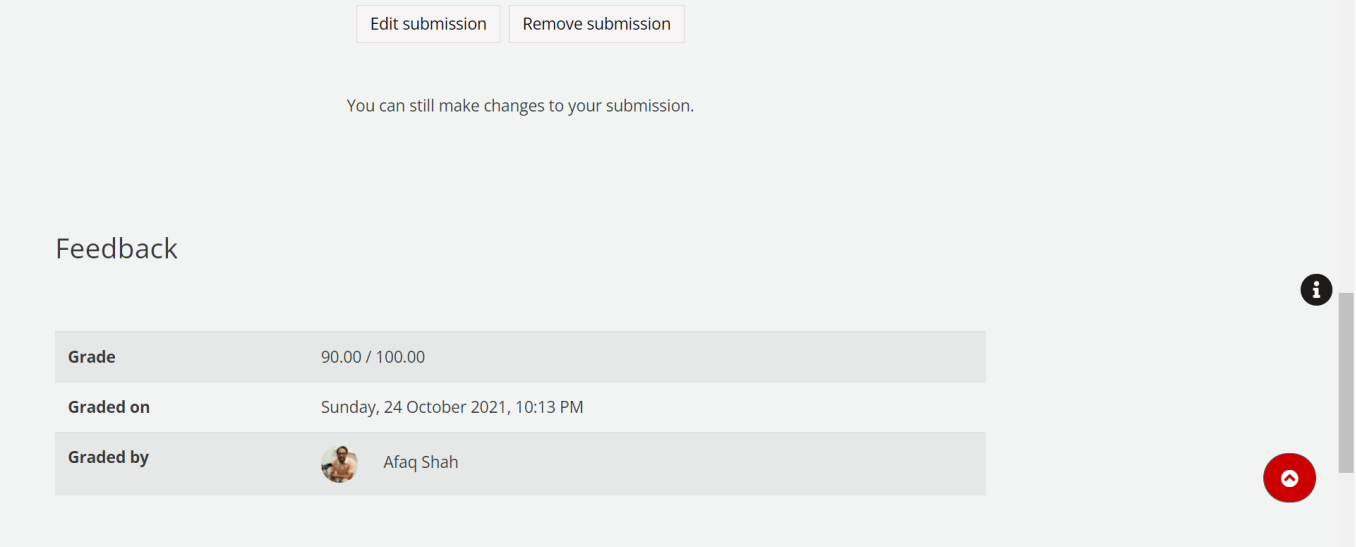


Assumptions

1. The collaborator(s) timeline in total is <200 tweets that are unread
2. /Icons/ ← Contains Map Icons
3. /Photos/ ← Contains full twitter photos

- To add anything to canvas two things must be done
    - Add to Eventdatabase
    - Click update map on home page (to refresh the canvas to see the added item on the canvas. Prompts are given)
- Clicking the update map in the home page will restore the clickable markers just in case the client closed the prompt allowing for the marker to be clicked

1)

ICT365 MAJOR ASSIGNMENT

CHECK LIST

Surname (Family Name):
____Chong_____

Given Names: ____Jin
cherng_____chong_____

Student Number:
__33170193_____

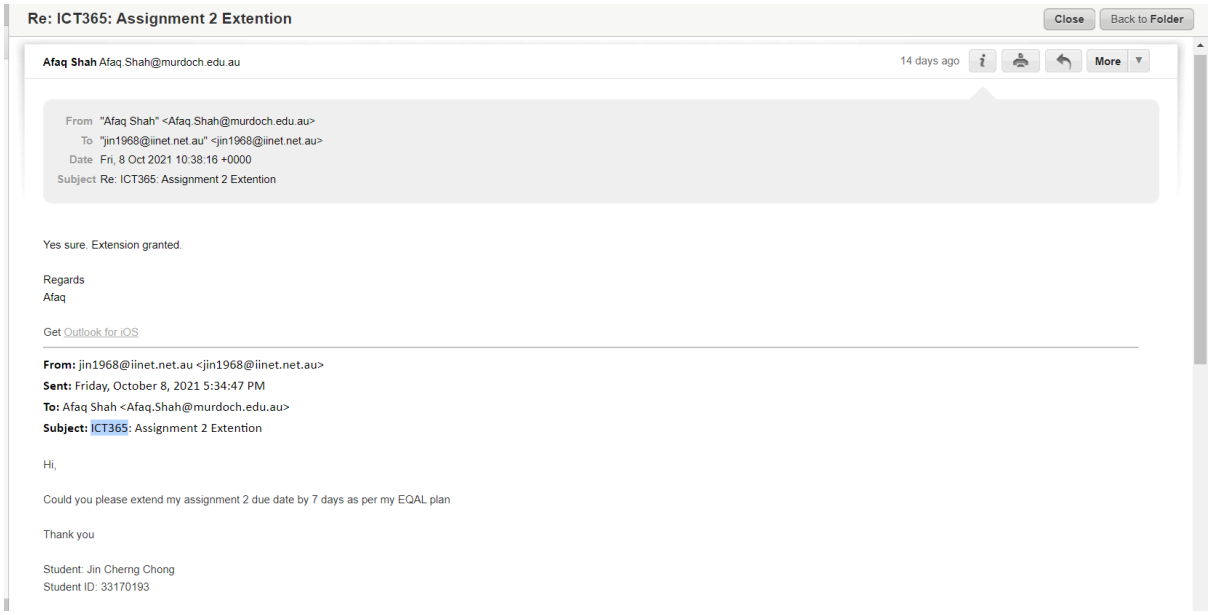
Tutor's Name: _____DR
Afaq_____

Assignment Due Date: _____22/10/2021_____ Date Submitted: ___22/10/2021___

Your assignment should meet the following requirements. Please confirm this (by ticking boxes) before submitting your assignment.

- ☐ [YES] All details above are complete.
- ☐ [YES] I acknowledge and agree that the assessor of this assignment may, for the purpose of assessing this assignment, reproduce this assignment and provide a copy to another academic staff member.
- ☐ [YES] I am aware of Murdoch University's assessment policy 8.1.8 that states that "Unit Coordinators have the right to submit any assignment to the University's plagiarism detection software if they suspect plagiarism."
- ☐ [YES] I have read and understood the requirements for submission of assignments specified in the Unit Information and Learning Guide.
- ☐ [YES] I have read and understood the requirements for documenting and submitting this assignment that are specified in the assignment question sheet of this assignment.
- ☐ [YES] I understand that the archive file should be submitted to ICT365 Unit LMS.
- ☐ [YES] I have kept a copy of this assignment, including the archive file.

2)



Re: ICT365: Assignment 2 Extention

Close Back to Folder

Afaq Shah Afaq.Shah@murdoch.edu.au 14 days ago More

From "Afaq Shah" <Afaq.Shah@murdoch.edu.au>
To "jin1968@iinet.net.au" <jin1968@iinet.net.au>
Date Fri, 8 Oct 2021 10:38:16 +0000
Subject Re: ICT365: Assignment 2 Extention

Yes sure. Extension granted.

Regards
Afaq

Get Outlook for iOS

From: jin1968@iinet.net.au <jin1968@iinet.net.au>
Sent: Friday, October 8, 2021 5:34:47 PM
To: Afaq Shah <Afaq.Shah@murdoch.edu.au>
Subject: ICT365: Assignment 2 Extention

Hi,

Could you please extend my assignment 2 due date by 7 days as per my EQAL plan

Thank you

Student: Jin Cherng Chong
Student ID: 33170193

3)

One Issue I faced was how to make the data structure available to all the files that needed it. I decided to utilize a singleton design pattern to create the EventDatabase class. This pattern fulfilled the requirement of only needing and allowing single instance of the database. Also, when designing the database class, I made sure not to return the whole eventDatabase. The reason being that would defeat the whole purpose of creating the class, since we are exposing the entire data structure making the encapsulation worthless. Instead, I opted to provide functions that would allow for the modification of the database in a controlled manner. These functions also support abstraction

Second issue I faced was in determining what data structure I was going to use to store the collaborators and tweets for the map. The data structure I decided to opt to was a dictionary. The key would always be unique since a GUID was used as the key. However, for the tweet class I decided to user twitters auto generated identifier as the key. However, one of the issues I noticed was that with the google map library (https://github.com/yeganehaym/GMap.Net) the key couldn't be all numbers. So, under the hood I added the character "s" in front of every number in all the data structure keys and when I needed to get the tweet, I would just remove the S

Thirdly another issue I faced was how to create clickable markers. The library I used to display the google map canvas https://github.com/linvi/tweetinvi provided non clickable markers creation. I managed to solve this by making sure during runtime each marker that was created would also create a button object assigned to it. Thus, the client could click on the button object to open up a form revealing the details of the marker

The technology I used to implement my assignment was ASP .net. This technology is primarily used for creating web applications. Specifically, ASP .net makes it easier to work with the backend and communicate with the data structures and database. In my assignment, the backend doesn't store long term only temporarily when the program runs using data structures such as dictionaries and list.

One advantage with ASP .net is that communicating with other web application such as twitter is easy. The reason being is the API end points of social media such as twitter are in JSON which is an extension of JavaScript. Since the web application has Javascript front end we can utilize that to communicate with the JSON endpoints of twitter. This allows us to use the API offered by twitter to

the full extent and not rely on C# libraries. On the other hand, Xamarin it's more difficult to integrate since Javascript is hidden away. One instance, where Xamarin is easier than ASP NET (in our context) is the integration and use of google map. The Xamarin.GooglePlayServices.Maps package was a library created by google.

For example, when dealing with clickable markers–

Clicking on Marker using Xamarin is inbuilt and simpler given google provided the function-

```
void MapOnMarkerClick(object sender, GoogleMap.MarkerClickEventArgs markerClickEventArgs)
{
   markerClickEventArgs.Handled = true;
}
```

Clicking on a marker using ASP .net is more difficult-

```
public void AddEventToCanvas(string tempKey, IAmEvent tempEvent)
 {
            Marker marker = new Marker("tempMarker");
            string str = "";


     str = @"<button type=""button"" id=""{0}"" onclick=""OpenDisplayTweetForm(this.id)"">Click Me!</button

      var createClickableMarker = String.Format(str, tempKey, tempKey);
            marker.InfoWindow = new InfoWindow("iw1")
            {
                Content = createClickableMarker
            };


 }
```

The ASP .net implementation required me to code buttons that would attach to unclickable markers to emulate the marker click. This is one of the downfalls of using a library not provided by the creator of the application.

A disadvantages of ASP .net is that while it is designed to work on all platforms through the web the implementation is more difficult on mobile. Xamarin adopts a mobile first approach where the application would be designed for the mobile. In order to make the ASP .net into a mobile application a possible design could the use of flexbox. Relies on putting everything in containers and

when the screen is enlarged or shrink the containers would shape in that size. The queries would be responsible for shifting the containers. With Xamarin a mobile canvas is provided where you drag and drop the components for specific mobile devices.

An advantage to ASP net is that is light weight and is intend to be used universally, given it runs on the web. On the other hand, WPF is limited to be used in windows thus there is some limitations in its application.

4)

Features-

- Singleton design pattern has been used for the EventDatabase, EventCanvas and ClientTwitter
- Designs principle have been implemented and commented in the solution file. Interface has also been used
- All classes that can be unit tested has been. Also, application tests have been performed
- EventIcon used as the icon on canvas must be formatted in correct size. Program does not resize it
- Used https://github.com/yeganehaym/GMap.Net for the google map
- Implemented clickable marker myself since the https://github.com/linvi/tweetinvi did not provide clickable marker. The library had non clickable markers
- Used https://github.com/linvi/tweetinvi for twitter api

Form for potential collaborator

- My application has three different collaborators roles- relative, carer, and friend
- My application allows clients to add collaborator level of support- collaborator's availability
- My application allows clients to add additional details of collaborator such as age, name, profile, and twitter name, location on canvas and icon on canvas

Display images and text

- My application displays a google map
- My application displays all the collaborators on the google map canvas with icon
- My application allows client to view collaborator details
- My application displays twitter messages pushed/pulled on google map canvas with icon

- My application allows client to view twitter message

Social media interaction

- My application interacts with twitter as it's social media stream
- When client pushes post on social media
    - The message will be sent/pushed to the client's twitter time line
    - The message will also be sent/pushed to collaborators (via twitter Direct Messaging) who are available to help
- Collaborators will post tweets on their twitter social media timeline
    - The message will be sent/pushed to the program
    - The message can then be read by the client and added to the google map canvas

- When client pull post from social media
    - All the collaborators' timelines tweets will be pulled to the program
    - The clients can add the unread/unadded tweets to the canvas
- Client can also pull/delete their own post from social media that they pushed using the application
- Client can also pull/delete collaborator tweets on the canvas but obviously not pull the tweet from collaborator own social media

5)

Task 1

- Create form that includes → Basic information + Levels of support/Availability

Task 2

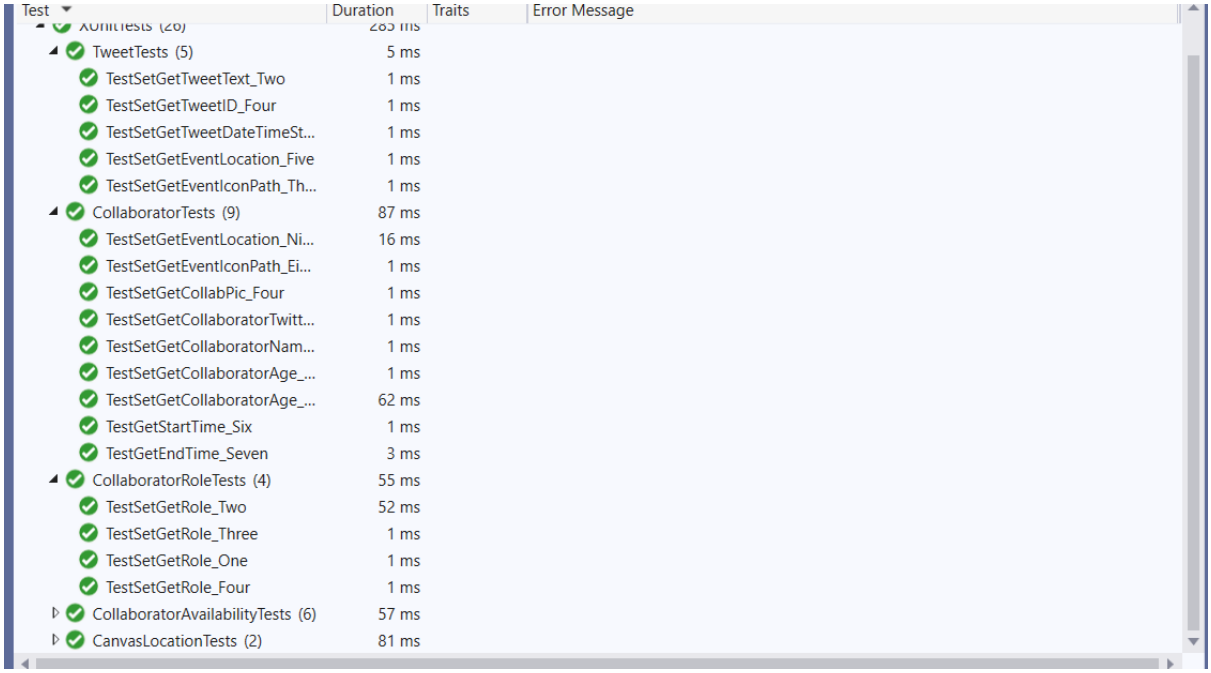
- Click on the Icon → Open up information for the collaborator in a form
- Click on the text message → Open tweet in the form

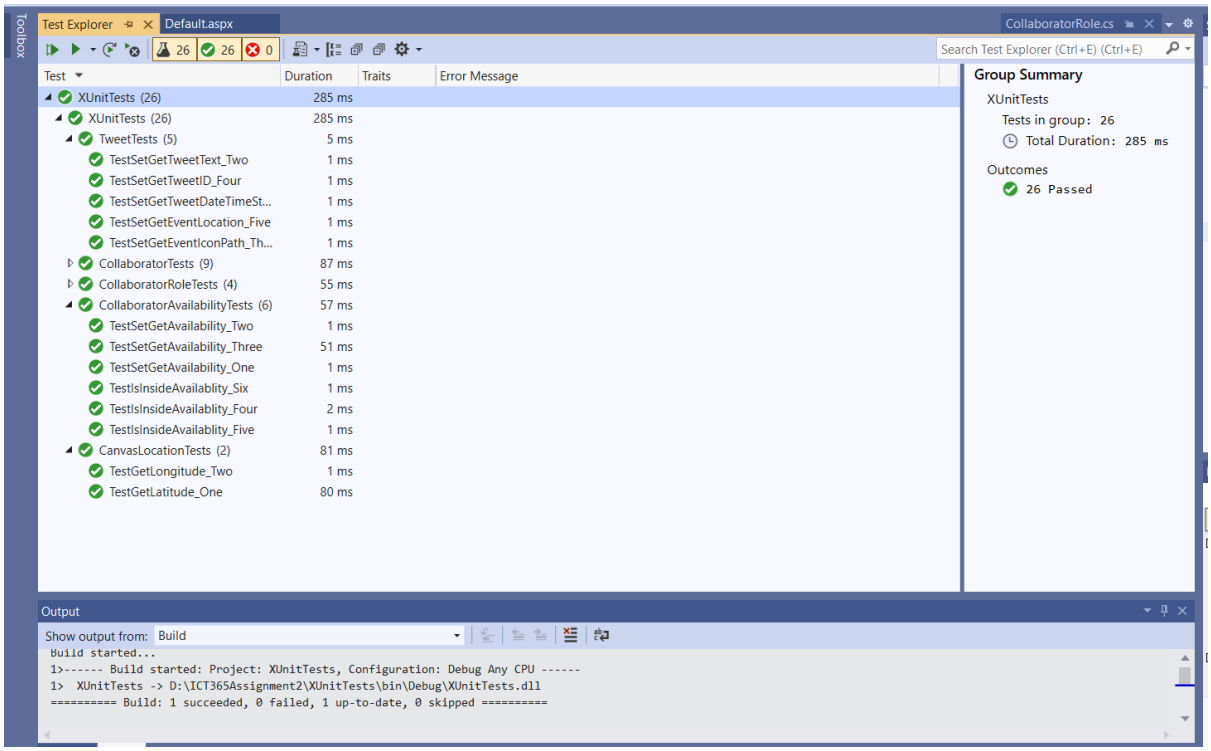
Task 3

- Client right click canvas to 'send message to collaborator' [PUSHING]
    - Write the message in textbox + Location where the help is needed
        - For example- "Help I can't find my car"
    - Add tweet message to canvas
    - Program pushes the message to available collaborates through twitter DM VIA TIME and sends the tweet on the clients timeline
        - Push via messages and general tweet with format- "ATTENTION: [Name] I need help"
- Collaborator sends a message on social media [PULLING]
    - Client reads the message
    - Adds message to canvas
    - Adds the icon to canvas
    - Creates the text object
- Client can also pull and remove their pushed twitter messages using the application.

6)

Unit testing has been completed-


Test
Duration
Traits
Error Message
TweetTests (5) 5 ms
TestSetGetTweetText_Two 1 ms
TestSetGetTweetID_Four 1 ms
TestSetGetTweetDateTimeSt... 1 ms
TestSetGetEventLocation_Five 1 ms
TestSetGetEventIconPath_Th... 1 ms
CollaboratorTests (9) 87 ms
TestSetGetEventLocation_Ni... 16 ms
TestSetGetEventIconPath_Ei... 1 ms
TestSetGetCollabPic_Four 1 ms
TestSetGetCollaboratorTwitt... 1 ms
TestSetGetCollaboratorNam... 1 ms
TestSetGetCollaboratorAge_... 1 ms
TestSetGetCollaboratorAge_... 62 ms
TestGetStartTime_Six 1 ms
TestGetEndTime_Seven 3 ms
CollaboratorRoleTests (4) 55 ms
TestSetGetRole_Two 52 ms
TestSetGetRole_Three 1 ms
TestSetGetRole_One 1 ms
TestSetGetRole_Four 1 ms
CollaboratorAvailabilityTests (6) 57 ms
CanvasLocationTests (2) 81 ms


Toolbox
Test Explorer
Default.aspx
CollaboratorRole.cs
26
26
0
Search Test Explorer (Ctrl+E) (Ctrl+E)
Test
Duration
Traits
Error Message
XUnitTests (26) 285 ms
XUnitTests (26) 285 ms
TweetTests (5) 5 ms
TestSetGetTweetText_Two 1 ms
TestSetGetTweetID_Four 1 ms
TestSetGetTweetDateTimeSt... 1 ms
TestSetGetEventLocation_Five 1 ms
TestSetGetEventIconPath_Th... 1 ms
CollaboratorTests (9) 87 ms
CollaboratorRoleTests (4) 55 ms
CollaboratorAvailabilityTests (6) 57 ms
TestSetGetAvailability_Two 1 ms
TestSetGetAvailability_Three 51 ms
TestSetGetAvailability_One 1 ms
TestIsInsideAvailablity_Six 1 ms
TestIsInsideAvailablity_Four 2 ms
TestIsInsideAvailablity_Five 1 ms
CanvasLocationTests (2) 81 ms
TestGetLongitude_Two 1 ms
TestGetLatitude_One 80 ms
Group Summary
XUnitTests
Tests in group: 26
Total Duration: 285 ms
Outcomes
26 Passed
Output
Show output from: Build
Build started...
1>------ Build started: Project: XUnitTests, Configuration: Debug Any CPU ------
1> XUnitTests -> D:\ICT365Assignment2\XUnitTests\bin\Debug\XUnitTests.dll
========== Build: 1 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========

Twitter account used-

https://twitter.com/Student33170193 @Student33170193

@_Mew2_

Test #1

Test Objective: Check adding a *relative* type collaborator and their level of support will be displayed on canvas

Test Steps:

- Right click on the canvas
- Select “Register a collaborator to database”
- The form for the collaborator is filled out with the collaborator role- relative and level of support (start and end time)
- Display the created event on canvas

Expected result: Added collaborator should appear on canvas

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #2

Test Objective: Check adding a *friend* type collaborator and their level of support will be displayed on canvas

Test Steps:

- Right click on the canvas
- Select “Register a collaborator to database”
- The form for the collaborator is filled out with the collaborator role- friend and level of support (start and end time)
- Display the created event on canvas

Expected result: Added collaborator should appear on canvas

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #3

Test Objective: Check adding a *carer* type collaborator and their level of support will be displayed on canvas

Test Steps:

- Right click on the canvas
- Select “Register a collaborator to database”
- The form for the collaborator is filled out with the collaborator role- friend and level of support (start and end time)
- Display the created event on canvas

Expected result: Added collaborator should appear on canvas

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #4

Test Objective: Check client pushes post to social media

Test Steps:

- Right click cavas
- Client clicks send messages to collaborator

Expected result: Should appear on client twitter feed and DM collaborator available to help

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #5

Test Objective: Check client pushes post to social media but collaborator unavailable

Test Steps:

- Right click cavas
- Client clicks send messages to collaborator

Expected result: Should appear on client twitter feed and collaborator should not be DMed

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #6

Test Objective: Check client can also pull/delete their own post from social media that they pushed using the application

Test Steps:

- Client clicks on marker
- Client clicks removes

Expected result: Twitter tweet should be removed

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #7

Test Objective: Check client can pull collaborator post from social media and add to canvas

Test Steps:

- Right click on the canvas
- Client clicks read unread messages
- Client add message

Expected result: Unread twitter tweet should now be read and appear on canvas

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Test #8

Test Objective: Check client can pull different collaborator posts from social media and add to canvas

Test Steps:

- Right click on the canvas
- Client clicks read unread messages
- Client add message

Expected result: Unread twitter tweet should now be read and appear on canvas

Pass/Failed: Passed

Actual result-

https://www.youtube.com/playlist?list=PLUktbenWQI5jqjo8MTcLOZa3Xew9KF-Dd

Design pattern implemented is singleton pattern

Image 1: Design pattern code singleton pattern EventDatabase.cs

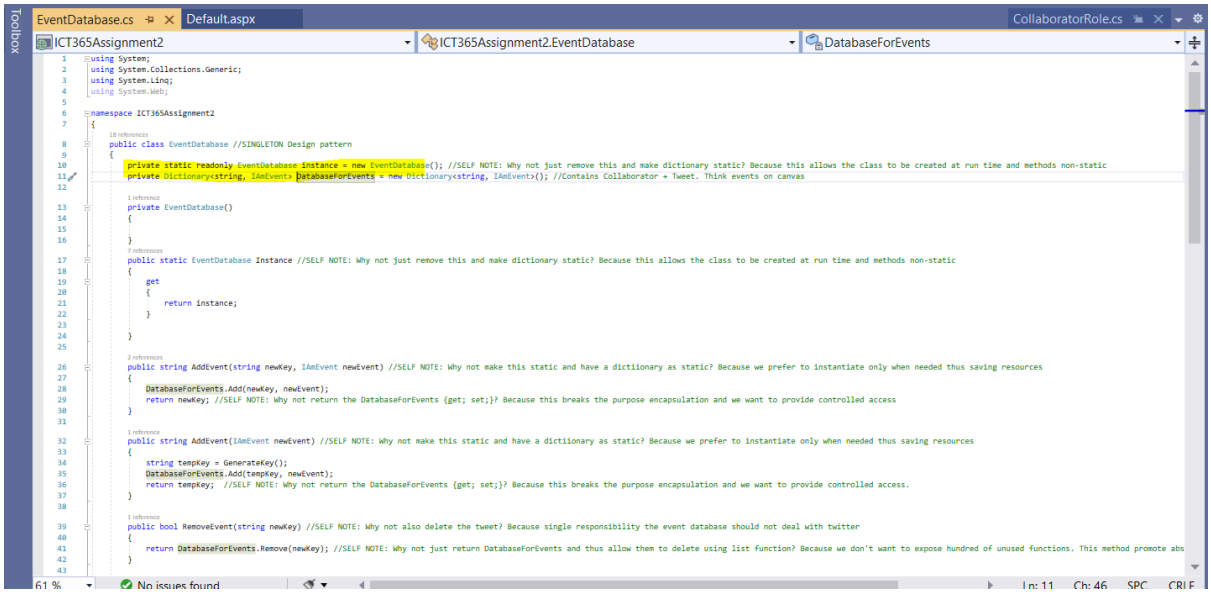


Image 2: Design pattern code singleton pattern EventCanvas.cs

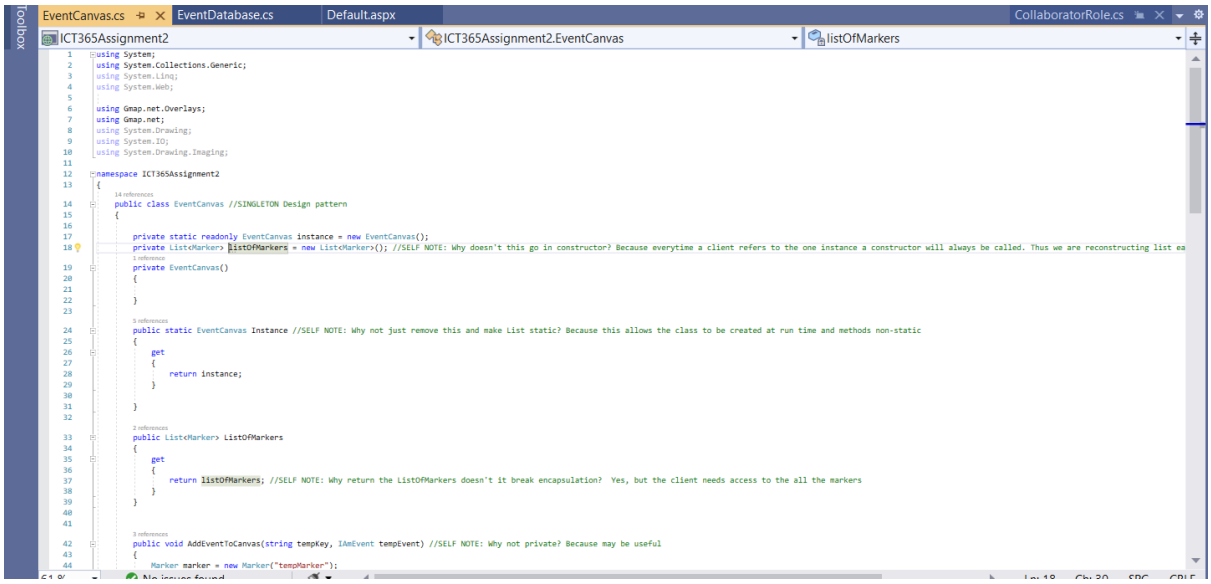
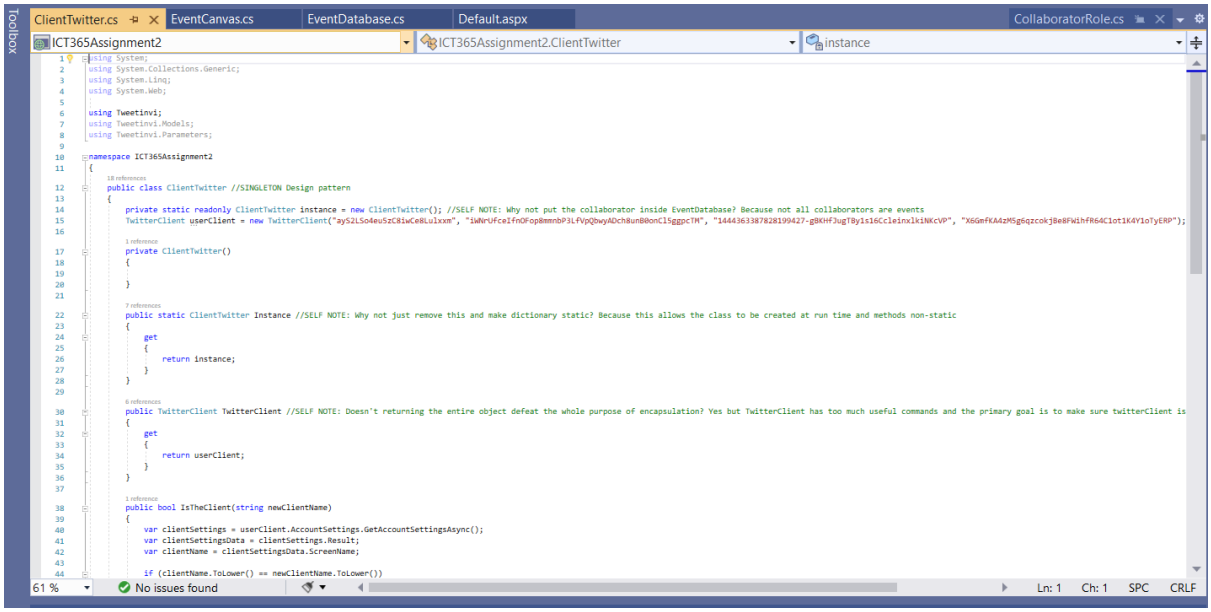


Image 3: Design pattern singleton pattern ClientTwitter.cs


Toolbox
ClientTwitter.cs
EventCanvas.cs
EventDatabase.cs
Default.aspx
CollaboratorRole.cs
ICT365Assignment2
ICT365Assignment2.ClientTwitter
instance
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Tweetinvi;
using Tweetinvi.Models;
using Tweetinvi.Parameters;
namespace ICT365Assignment2
{
18 references
public class ClientTwitter //SINGLETON Design pattern
{
private static readonly ClientTwitter instance = new ClientTwitter(); //SELF NOTE: Why not put the collaborator inside EventDatabase? Because not all collaborators are events
TwitterClient userClient = new TwitterClient("ayS2LSo4eu5zC8iwCe8Lulxxn", "iWNrUfceIfnOFop8mmnbP3LfVpQbwyADch8unB0onCl5ggpcTM", "1444363387828199427-gBKHFJugfBy1s16CcleinxlkiNkcVP", "X6GmfKA4zMSg6qzcokjBe8FWihfR64C1ot1K4YloTyERP");
1 reference
private ClientTwitter()
{
}
7 references
public static ClientTwitter Instance //SELF NOTE: Why not just remove this and make dictionary static? Because this allows the class to be created at run time and methods non-static
{
get
{
return instance;
}
}
6 references
public TwitterClient TwitterClient //SELF NOTE: Doesn't returning the entire object defeat the whole purpose of encapsulation? Yes but TwitterClient has too much useful commands and the primary goal is to make sure twitterClient is
{
get
{
return userClient;
}
}
1 reference
public bool IsTheClient(string newClientName)
{
var clientSettings = userClient.AccountSettings.GetAccountSettingsAsync();
var clientSettingsData = clientSettings.Result;
var clientName = clientSettingsData.ScreenName;
if (clientName.ToLower() == newClientName.ToLower())
61 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF