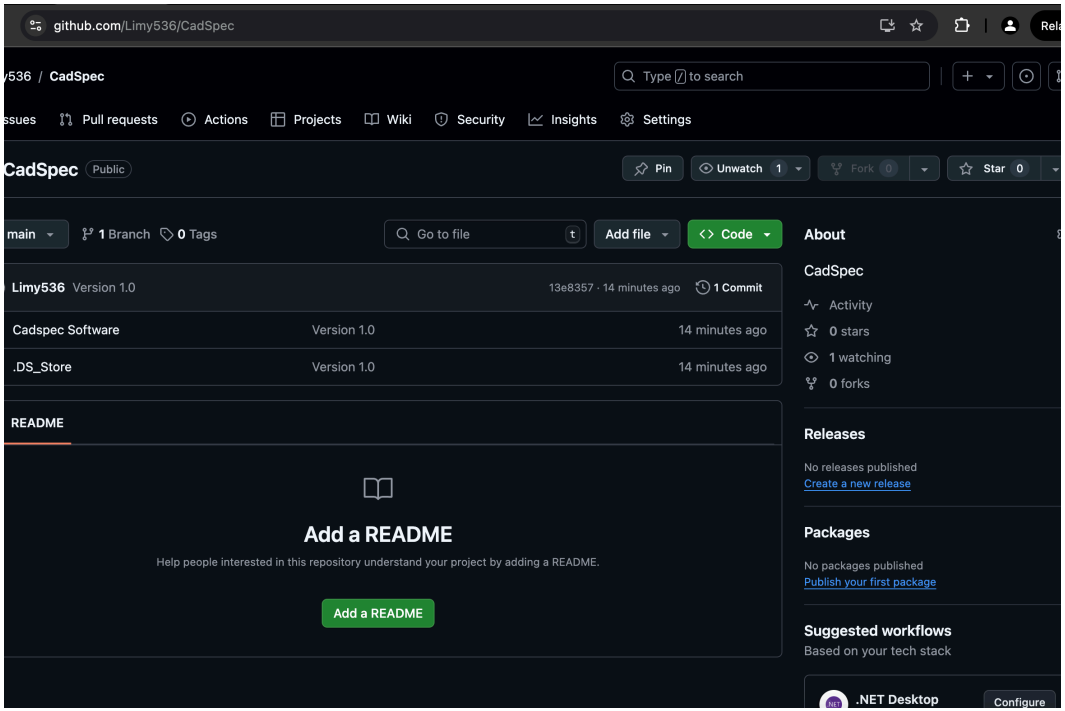
App Thought Process:

Version Control

Versions of the app as it is updated and expanded on will be managed through Git Hub, repository will be under the Cadspec name, public for others to view and will follow suit in matching the with the in app version for the branches off of the Main Branched, which will only be pushed to with the latest changes once all regressive testing is done.



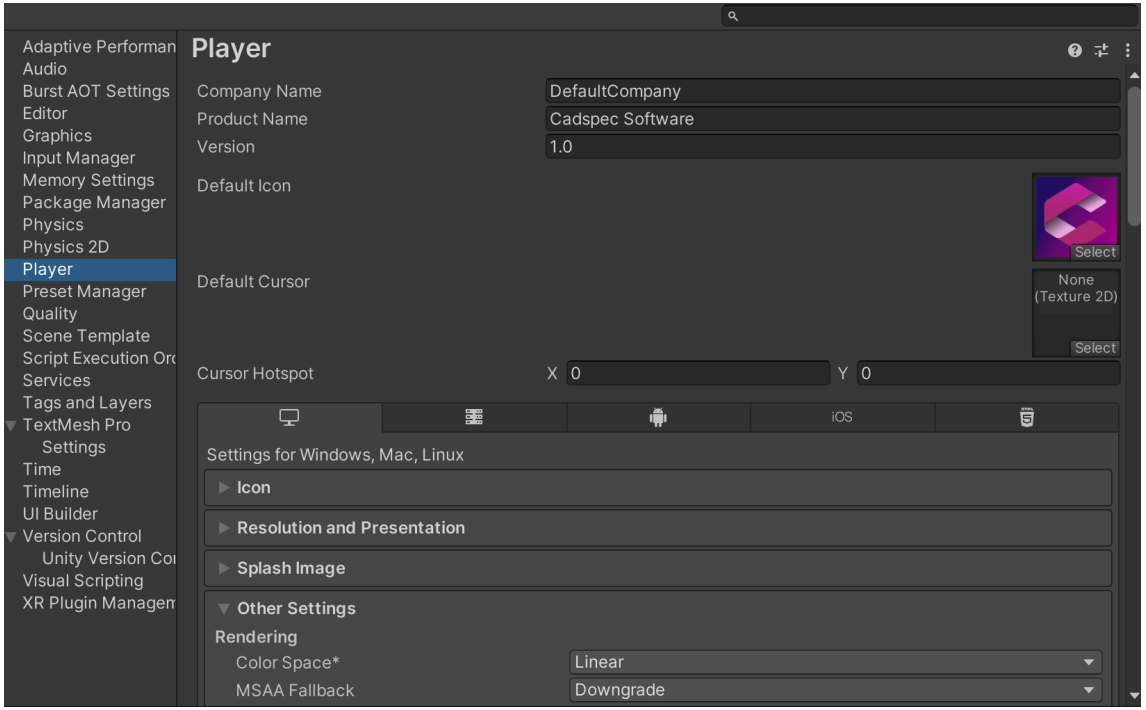
Git Hub Repository Set Up Ready for App Development

Link: https://github.com/Limy536/CadSpec.git

App Thought Process:

Software Package To Use:

Developing on a Mac computer, Visual Studios been deprecated for Mac and lacks ease of use of other software development packages that can also be used to create a simple app for multiple platforms, with quick setup. To ensure flexibility and speedy implementation I'm opting to use Unity. Easy to create a user interface, allows for simpler multi-platform development with ease to use in-house platform management as well as version control with Github and inbuilt time saving features to quickly implement interactive features.



Unity Platform and Version Controls

- Current version in use is 1.0
- Available on Mac, Windows, Linux, Android and IOS
- Will be tested and built on Mac fr interview

App Thought Process:

2D Assets Required to create App and Interface:

A list of 2D assets I'm going to need before creating the interactive interface using Gimp 2.10 in png format for alpha channel and reduce file size.

- App Icon incorporating the company logo to find and open the program on the given Platform.
- Company brand colour pallet set to standardise the look of the application with company website and other potential public software.
- Feather background and company logo with transparent background.

Assets For Interface

App Thought Process:

Basic Prototype of App:

- App only needs one scene, with search interface.
- User needs to be able to enter an address.
- User needs to be able to trigger a search.
- User needs to be able to exit app.

Prototype Design Layout

Interactive Components:

<- Close Button to exit App

<- Inputfield to enter address

<- Show number of occupants found

<- Results in a scrollview

<- Search Button to perform search

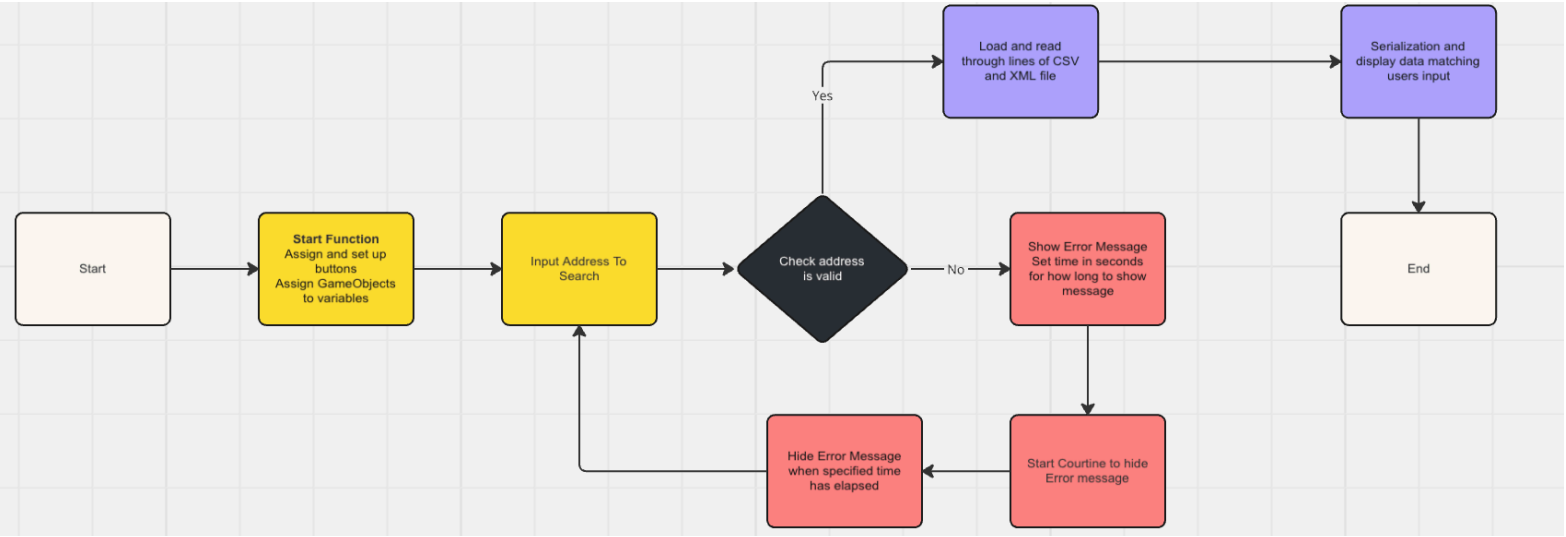
* *Note: User will need notifying of a search is in progress and prohibited from trying to re-submit a search to prevent compiled data becoming corrupted.*

* *Note: Specification given has an expected format for the address to valid, user will need notifying if the address they provided is not valid, a visual example of how it should appear in the input area will help reduce mistakes by the user.*

App Thought Process:

Search For Address Flow Diagram:

The main flow or process of the program while searching for the address



Process

Functionality:

Closing Application:

Close button to needs to end the application on onClick up of the mouse.

- Unity buttons have a number of inbuilt features that will automatically call the on click runtime event function.
- I will simply need to assign either code or in the editor the subsequent function called by onClick that I want to handle closing the application for PC devises.
- Alternative standard option for exiting application using the input escape key. Unity has inbuilt functions they checks for the input key presses with runtime events, I can use this to check regularly if the key is up after beeping pressed through the Update function (Function checked once per frame). If it has we again call out close function that will use the Application.Quit a function in Application class in the UnityEngine namespace with the value of 0.

App Thought Process:

Inputfield:
The user needs to be able to enter a address which the application can then check to see if it is valid.

- Unity provides a inbuilt input field object that contains the needed functionality to manage displaying inputted data. The Inputfield object also contains a placeholder child object that is displayed in the input text component until the user interacts with it. This is all handled by unity, so we can use the placeholder to give an example on how the user should enter the address.

Search Button:
The will be main drive behind most of the key things the add needs to do. It will need to call a function that checks the address provided is valid, then either display a error message if invalid or if valid, then call a function that opens the two files provided one at a time before call a function that reads through the lines of and serialises the data correctly dependent on it's file type. This data will then need to be stored ready to display. The data stored will also need to be sorted in the correct order from the oldest of occupants to the youngest, the data will also need to be checked for any duplicates and removed using System namespace functions. The result and the number of occupants found can then be displayed to the user.

- **Check Address Is Valid function** will check the input string contains numbers at the start by splitting the sting at the first whitespace and then try to parse the the string to an int. It will then need to check the street name has a reasonable length of chars in the string to be a valid address. The split string can also be store in variables to check against the data in the files. If the input data fails against these checks it can also notify the user if there was an issue with the address entered by them or inform them that the search has began. If the address is valid I can get the app to disable the search button from responding to further clicks until the search is finished to stop data corruption.
- **Search function** will be the function called when the user click search, this function will then call the Check Address Is Valid function as one of it first steps and check the returning result to know if to proceed with loading and reading the files by calling the Read File function. The files themselves will be store with the apps Resources folder, which gets baked into the app as part of a Unity feature. There room to develop this further and get the file via URL at a later date. Next it can then sort and check the returned matching data has no duplicates by call the Compare To List Age function and finally use namespace System.Linq to remove and doubles.
- **Read File function** will check which file is currently open and then read through each line and sort the information in the string, by splitting it up with special characters (',' or '=') into arrays, checking specific elements within the array against the values obtained on inputted house number and address. Matches can then have the occupants added to list of residents so they can be sorted in oldest to youngest order. The first file will be csv type file, the second is an xml file type and will need to sorted into a similar structure of the csv file first so all the relevant data is in scope before separated into an array to target specific elements (***Note: Check not overlooked simpler method to read data in xml file**).
- **Compare To File List Age function** will take the list of occupants, create an array of the references within the list and compare the ages of each the occupant, switching the occupants order in the array depending on who is older. This can then be then assigned back to a new instance of the list. The string within the array can be split by the special character (',') and on successful parse compare the two values.
- **IEnumerator HideMessage function** is a event function called every second until the set time is reached at which point it will execute the code beneath yield return WaitForSeconds, which in this case it will close the error message if the users hasn't inputed and submitted a new search before this time is up