CSSE1001

29/05/2014

Assignment 3 – Final Design Document

Name: Bodhi Connolly
Student Number: s4359741
Project Title: Day One Python Client

Contents

1. Description

The popular iPhone app *Day One* allows users to create diary entries with text and an image, and attaches their current location and the weather into each entry. The app will sync these entries over Dropbox, but unless you use OS X there is no way to view, edit or create these entries.

This project is a multiplatform Python application that can view, edit and create these entries while syncing to the iPhone app over Dropbox. It can add photos to entries from the webcam. It allows the user to input their location and interfaces with Google Maps, while downloading the nearest weather information using the OpenWeatherMap API. All of this functionality is present while retaining 100% compatibility with the iPhone app.

2. User Interface

Figure 1. The splash screen.

The splash screen is what greets users upon launch. Before the app is ready for interaction it must update the necessary files from Dropbox. This takes a few seconds, and the splash screen reassures the user that nothing has frozen and that the app is still loading.

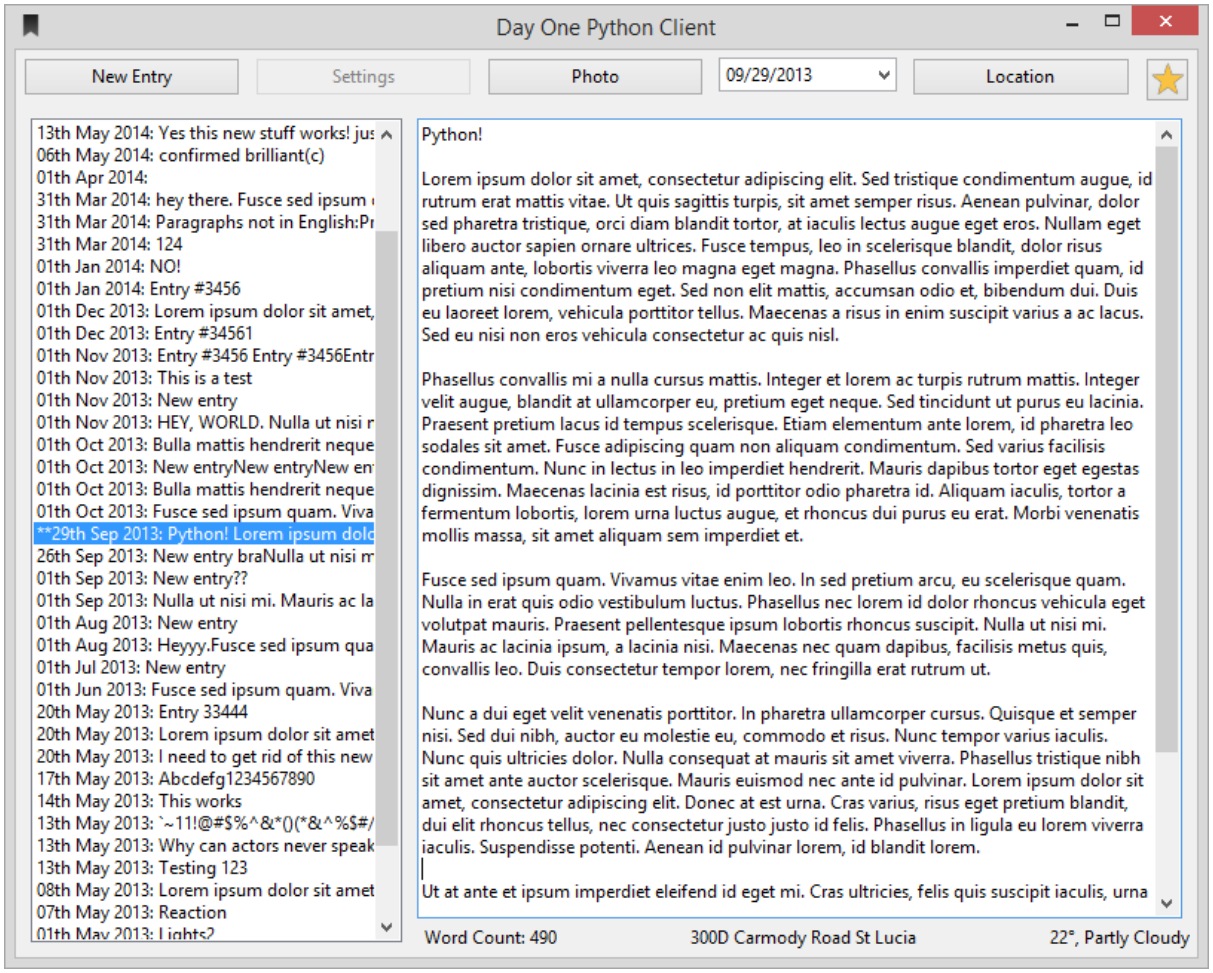

Figure 2. The main window.

The main window is the first chance the user has to interact with the program. On the left is a list of all entries, sorted by date. By default the first entry is selected. On the right is the body of the selected entry. This is editable. Along the top are buttons that provide the basic functionality.

The 'New Entry' button creates a new entry with the current date.

The 'Settings' button is disabled – the functionality was not finished, however Dropbox and directory settings can be changed by editing 'settings.txt'.

The 'Photo' button opens up a window that allows the user to take a photo with their webcam (if applicable, Windows only) shown in figure 3.

Clicking on the date opens a calendar dropdown, which allows the user to change the date of the entry to any date in the past (figure 4).

The 'Location' button opens a window that allows users to add/change the location and weather for an entry (figure 5).

Clicking on the star button stars the entry as an important one (toggle).

The labels along the bottom of the main window show some entry metadata such as location, weather and word count.

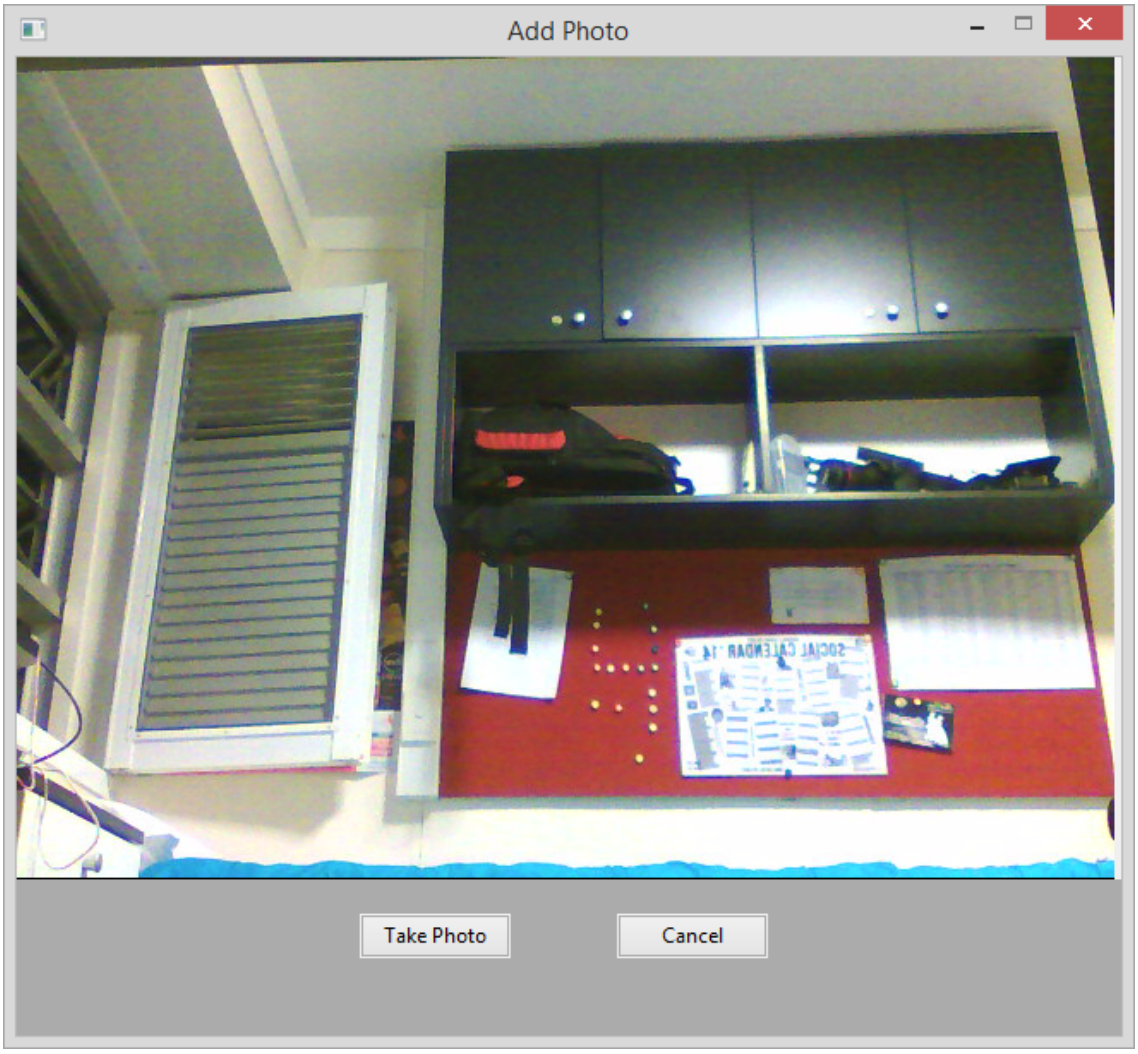

Figure 3. The webcam capture window.

While displaying the feed the webcam image is flipped, however when a photo is taken it is saved with the correct reflection.

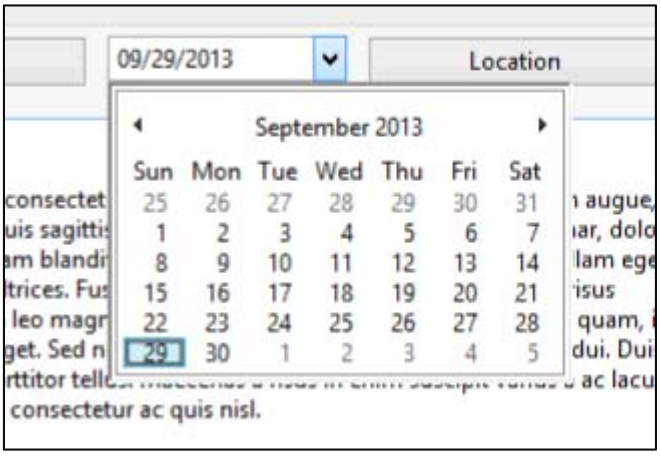

Figure 4. The calendar dropdown.

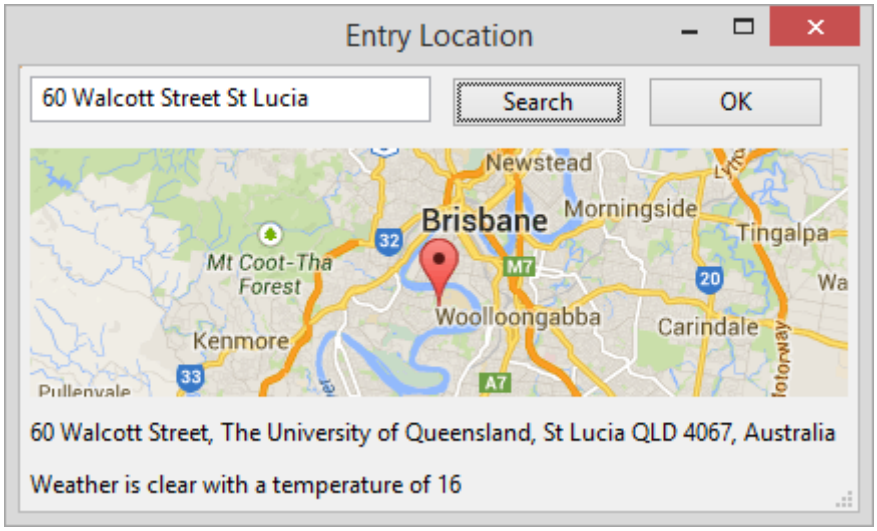

Figure 5. The location input window.

The user can enter their location in the text box, and then pressing enter or clicking search will query Google for a matching place. If found, the matching place will be displayed on a map and the weather at the location will also be printed. Clicking the OK button will then add this information to the current entry (if location has not been searched when OK button is pressed, the search will be performed and then automatically added to entry if valid).

3. Design

The application is built with four modules and a total of ten classes. The interaction of the GUI classes can be seen in figure 6.

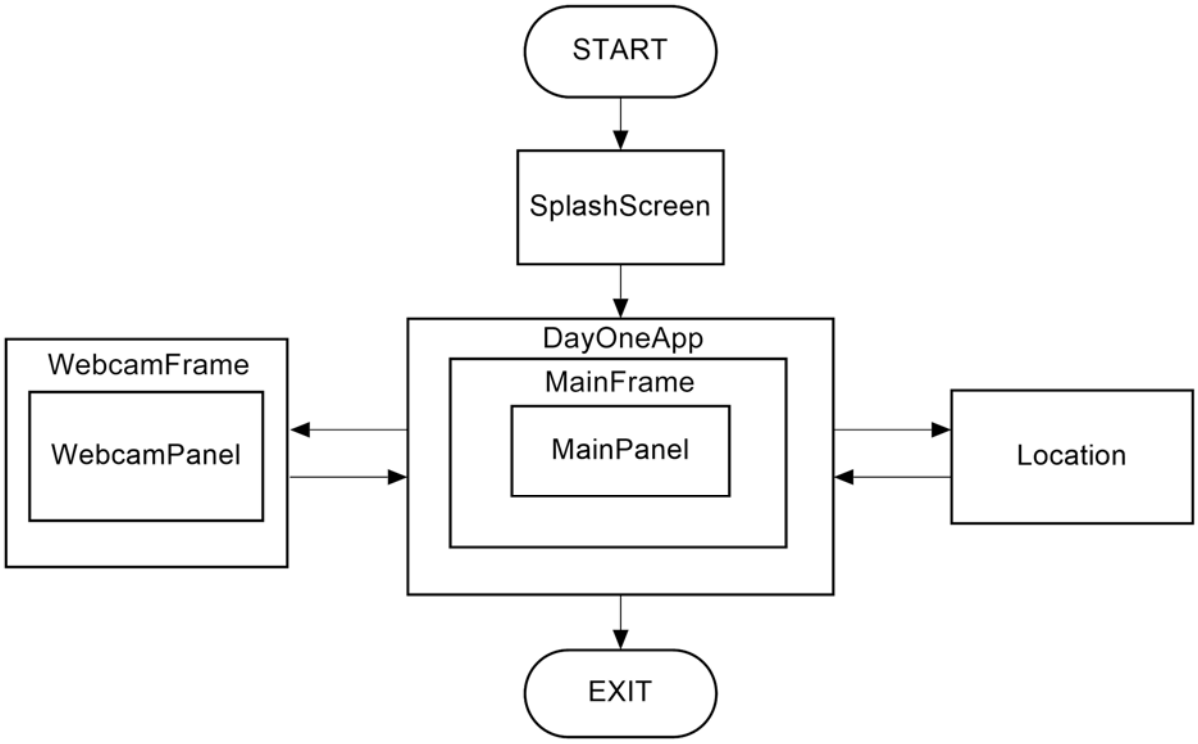

Figure 6. A visualisation of the GUI classes.

Module: manage_entries

This module provides the backend of the application, providing file access and Dropbox syncing. It contains the following classes

Class Entry(dict):

This class holds a single entry.

The functions in this class include:

uuid(self):

returns the entry's UUID

been_edited(self):

sets the entry as edited

is_edited(self):

returns whether entry has been edited

__lt__(self,other):

set compare method to use entry creation date

has_saved(self):

remove the edit flag from entry

load_entry(self,uuid):

load entry from file

save_entry(self):

save entry to file

Class EntryList(object):

A class to store a list of entries.

The functions in this class include:

__init__(self):

initialise with an emprty list

add_entry(self):

add an empty entry with the current date

load_entry(self,uuid):

add entry from file (returns None if file invalid)

load_list(self):

iterate over directory and attempt to load each file

get_all(self):

return a sorted list of all entries

save_list(self):

writes all edited entris to file

__getitem__(self,key):

return key entry from list

Class Dbox(object):

A class for uploading and downloading files from Dropbox.

The functions in the class include:

__itit__(self):

creates dropbox connection with app code, app secret and auth code

download_entry(self,filepath):

download entry file from Dropbox and write it to directory

update_dir(self):

downloads entries that need updating from Dropbox

get_changes(self,cursor=None,path_prefix=None):

gets list of files that need updating

upload_entry(self,filename,data):

upload a local entry to Dropbox

upload_changes(self):

uploads any modified files to Dropbox using threading

upload_changes_nothread(self):

uploads any modified files to Dropbox (not using threading)

upload_photo(self,uuid,file,ext='.jpg')

upload an image to Dropbox

Module: day_one

This module provides the main GUI and the majority of the operating logic.

It includes the following classes:

Class MainPanel(wx.Panel):

The main panel that hosts all the widgets in the main interface.

Its functions include:

__init__(self,parent):

initialise the main panel

display_list(self):

displays the sorted list of entries in the listbox

append_entry(self,entry):

appends an entry to the listbox

update_entry(self,pos1,pos2):

updates one entry and then selects another entry

selection(self,evt=None):

saves open entry and updates main text from new selection

edited(self,evt=None):

updates word count, marks entry as edited and saves entry text

new_entry(self,evt=None):

saves old entry and adds new entry at current date, deselecting all entries

open_webcam(self,evt=None):

opens webcam window set to current entry, unless no entry is selected or no webcam module

set_open_key(self,key,value):

sets key of open entry to value

set_open_key_key(self,key1,key2,value):

sets key2 within key1 of open entry to value

get_location(self,evt=None):

if entry selected, opens window to get user location and weather using Google Maps and OpenWeatherMap and uses them to set current entry

word_count(self,text):

returns a rudimentary count of the number of words in a string

date_change(self,evt=None):

update_wordcount(self):

updates the word count label to current word count

update_location_weather(self):

updates the location and weather label to the current entry

make_starred(self,evt=None):

toggles the star flag of open entry

on_close(self,evt=None):

saves the text from the open entry and uploads all un-uploaded changes to Dropbox

bg_saving(self):

uploads edited files to Dropbox every 15 seconds

save_upload(self,entries):

saves changes to file and uploads modified entries to Dropbox

save_upload(self,entries):

saves changes to file and uploads modified entries to Dropbox without threading

Class MainWindow(wx.Frame):

A class that hosts a MainPanel.

Its functions include:

__init__(self,parent):

initialises the frame with a MainPanel

on_close(self,evt=None):

activates the MainPanel closing procedure then closes the frame

Class SplashScreen(wx.SplashScreen):

A loading screen that updates files from Dropbox before opening the main window.

The functions include:

__init__(self,Parent=None):

initialises the splash screen and starts downloading changes

OnExit(self,evt=None):

closes the splash screen

Class DayOneApp(wx.App):

An app class that opens the splash screen and then a MainFrame.

Its functions include:

__init__(self):

initialises the app and launches the SplashScreen followed by the MainFrame

Module: location

A module that hosts the weather and location functions and windows.

Class Location(wx.Dialog):

A dialog to get the user's location and local weather via Google Maps and OpenWeatherMap.

Its functions include:

__init__(self,parent):

initialises the items in the dialog

button_click(self,evt=None):

finds the coordinates from the user entry text and get the weather from these coordinates

get_coordinates(self):

searches Google Maps for the location entered and returns the results, returns None if no results

get_weather(self,coordinates):

searches OpenWeatherMap for the weather at specified coordinates and sets variables based on this result for adding to entry, also loads image of coordinates from Google Maps Static Image API

submit(self,evt=None):

closes the dialog if user has already searched, else search and then close the dialog

Module: webcam

This module includes a window to view a feed from a webcam and take a photo.

Class WebcamFrame(wx.Frame):

A frame that hosts a panel showing the webcam feed.

Its functions include:

__init__(self,*args,kw):**

initialises frame with buttons and webcam panel

to_exit(self):

exit if picture has been taken

leave(self,evt=None):

close frame

Class WebcamPanel(wx.Panel):

A panel that displays a feed from the webcam.

Its functions include:

__init__(self,*args,kw):**

initialise panel with constantly refreshing feed

TakePhoto(self,evt=None):

set takepicture to True, telling the OnPaint function to take a photo on next refresh

CalcPic(self):

get frame from webcam ready to display

setUUID(self,uuid):

set UUID to associated entry

OnPaint(self,evt=None):

display latest webcam frame and take photo if takepicture is True

4. Support Modules

wx

The library used to generate the GUI.

dropbox

The library used to upload and download files from Dropbox.

pygeocoder

The library used to request location information from Google Maps.

VideoCapture (windows only)

The library used to access the webcam.