Andrew Baird

Professor Ssamra

CS 441

December 1st, 2019

Midterm One WriteUp

1. Prior to this class, I had very little knowledge of Python and its capabilities. Because everything in our project is written in Python, I had to quickly refresh and learn the language so I could begin programming. Luckily, I found writing code in python to be fairly self explanatory, but when I ran into a problem with something like syntax, I would just try to look up code examples and snippets. For example, I had to look up how to call public methods from another class in another file because I wasn't sure how to do that in python.
2. The main thing that I helped contribute to the group proposal were the use cases, and helping create the sprint tasks and outlines. Below are two sprint outline tasks that I helped create during class:

Sprint Ends October 4th.

Main Goals:
Get a working environment with a simple UI that when you press a button says Hello World.
Have this code pushed to the GitHub (make sure we have a .gitignore for any unwanted files) and make sure everyone can push/pull.
Create 40 sets of dummy data (20 per person)
Create parser that can read in and dummy data and print it out

/*Look at Google Calendar API and how to access with correct credentials.
Set up User Class*/

Notes:
Program will be done in Python?
Use pycharm to compile with.
Using tkinter for GUI.
Program will be a desktop application for windows.

Tasks:

Andrew:UI
Al: Parser
Mauricio: UI
Abdullah: Data
Yazeed: Data
Leo: Parser

Sprint Ends 10/21:

Allow parser to read input from a file:
Leo
Al

Begin constructing web scraper:
Yazeed
Mauricio

Begin constructing basic calendar:
Abdullah
Andrew

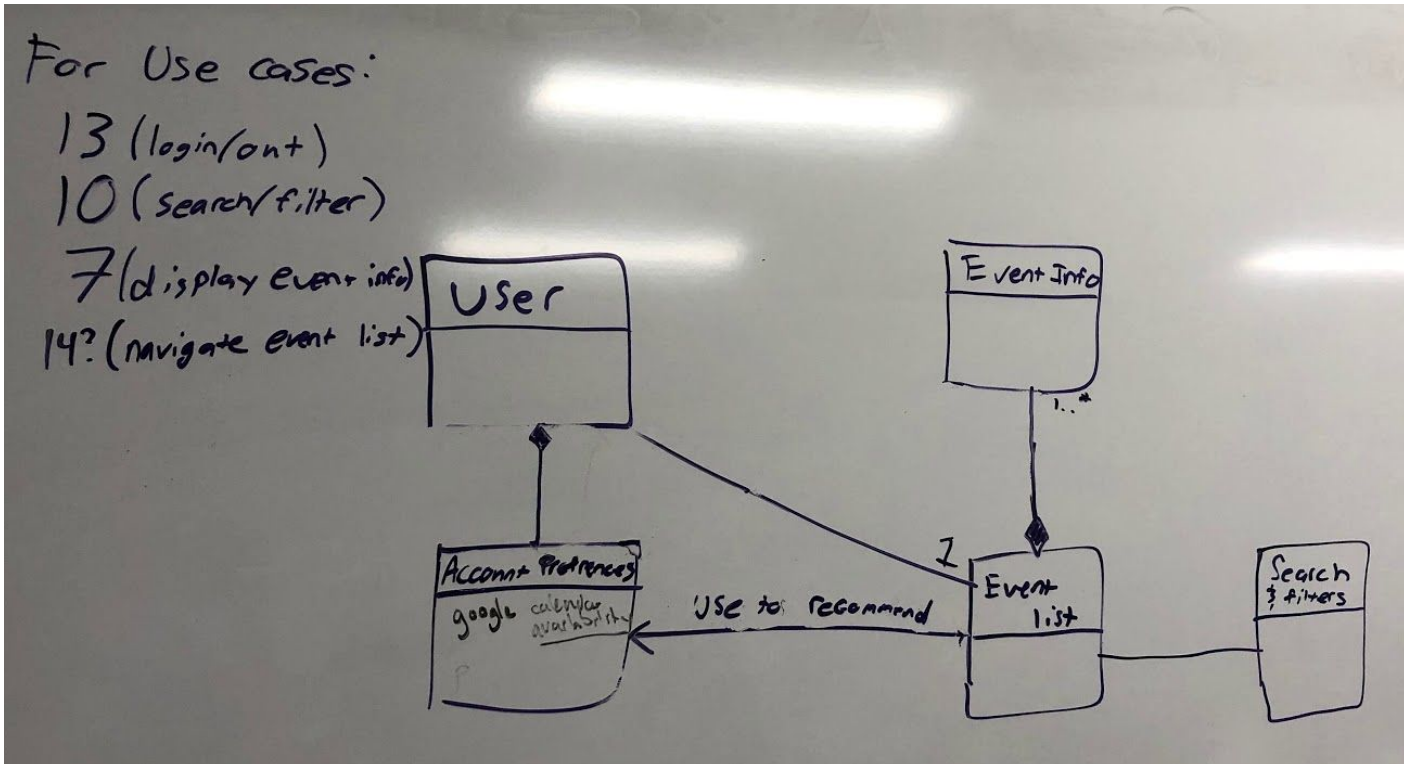
3. Below are three example use cases that we created and implemented in our final design:

2. Favorite event:
   - 2.1. Users searches for an event
   - 2.2. System shows a list of events
   - 2.3. User clicks on an event
   - 2.4. User can then favorite an event
   - 2.5. Favorite event shows up on a favorites page

7. Event Information
   - 7.1 User clicks on event
   - 7.2 System load information about event (photos, description, address, price)
   - 7.3 User can add event to calendar by clicking "Apply to Calendar"
     - 7.3.1(see 5.4.2.6 - 5.4.2.9)

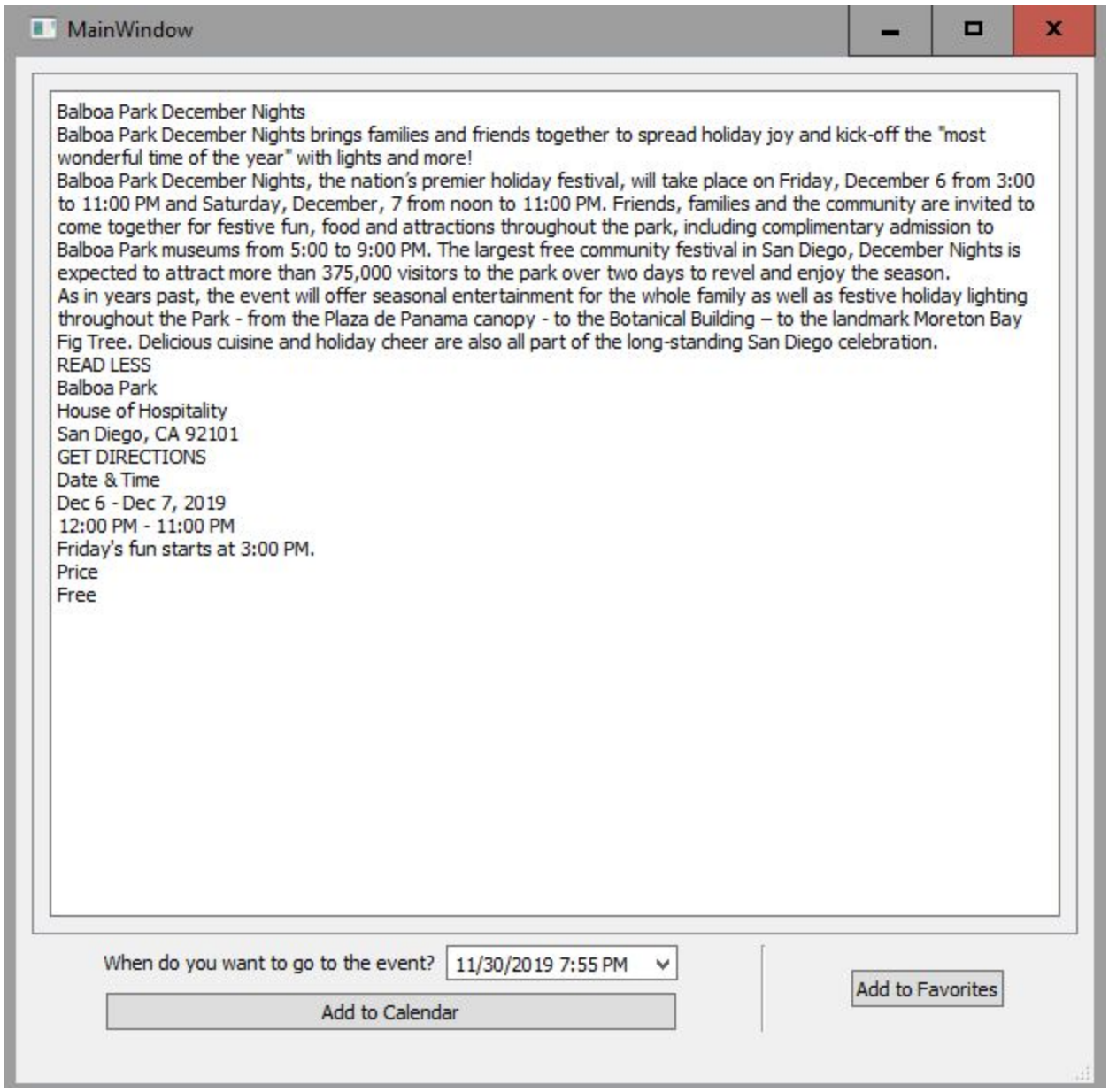
15.
   - UI changes
   - 15.1 User logs in and goes to a settings section
   - 15.2 User gets prompted with different UI environments (light mode/dark mode/etc.)
   - 15.3 User chooses the option they want
   - 15.4 System updates the UI to the desired environment.

4. Below is an example UML diagram that we created for the different use cases we designed:



5. The main thing that I developed was the google calendar part of our program. I first started with a quickstart example that could only print the first 10 events. Next, I allowed the user to add an event based on a hard coded string variable. After, I allowed the add event function to accept our Event class as a parameter and it would add the information from that event the google calendar based off of our Event class. Finally, I made it so that when the user is viewing an event, they can choose a date and time they want to attend, and it will add that events information to the user's calendar. An example of this

is shown below:



Also below is a snippet of how the program communicates with the Google Calendar API. Some of this was taken from their quickstart and some was added by me:

```
def printEvents(service, numOfEvents, outputBox):
    # Call the Calendar API
    now = datetime.datetime.utcnow().isoformat() + 'Z'  # 'Z' indicates UTC time
    outputBox.append('Getting the upcoming ' + str(numOfEvents) + ' events')
    events_result = service.events().list(calendarId='primary', timeMin=now,
                                          maxResults=numOfEvents, singleEvents=True,
                                          orderBy='startTime').execute()
    events = events_result.get('items', [])

    #  if not events:
    #     outputBox.insert(tk.END, links.get_text())
    for event in events:
        start = event['start'].get('dateTime', event['start'].get('date'))
        outputBox.append(event['summary'])


def AddEvent(service, outputBox, Event, startDateTime):
    # Refer to the Python quickstart on how to setup the environment:
    # https://developers.google.com/calendar/quickstart/python
    # Change the scope to 'https://www.googleapis.com/auth/calendar' and delete any
    # stored credentials.
    # startDateTime = Event.date + 'T' + Event.startTime
    # endDateTime = Event.date + 'T' + Event.endTime
    Event.description = Event.description + '\n' + Event.link
    event = {
        'summary': Event.name,
        'location': Event.location,
        'description': Event.description,
        'start': {
            # 'dateTime': '2019-11-30T09:00:00-07:00',
            'dateTime': startDateTime,
            'timeZone': 'America/Los_Angeles',
        },
        'end': {
            'dateTime': startDateTime,
            # 'dateTime': '2019-11-30T09:00:00-07:00',
            'timeZone': 'America/Los_Angeles',
        },
        'reminders': {
            'useDefault': False,
            'overrides': [
                {'method': 'email', 'minutes': 24 * 60},
                {'method': 'popup', 'minutes': 10},
            ],
        },
    }

    event = service.events().insert(calendarId='primary', body=event).execute()
    outputBox.append('Event created: %s' % (event.get('htmlLink')))
```

6. There isn’t a lot of time left, but if there were more time, I would allow the user to enter how many events they wanted printed on the calendar screen, and I would show a confirmation box after the event has been added to their calendar.