Project Report
on
WanderUSA Travel Planner

Submitted By
Paridhi Garg
Anoushka Kumar
Class: XII D

Under the Guidance of

Ms. Angel Panesar
Department of Computer Science
Sardar Patel Vidyalaya

Lodhi Estate, New Delhi 110003

CERTIFICATE

This is to certify that Paridhi Garg and Anoushka Kumar of Class XII D have prepared the report on the Project entitled "WanderUSA travel planner". The report is the result of their efforts & endeavors. The report is found worthy of acceptance as final project report for the subject Computer Science of Class XII. They have prepared the report under my guidance.

Ms. Angel Panesar
Department of Computer Science
Sardar Patel Vidyalaya
Lodhi Estate, New Delhi 110003

DECLARATION

We hereby declare that the project work entitled “WanderUSA travel planner”, submitted to Department of Computer Science, Sardar Patel Vidyalaya, Lodhi Estate, New Delhi 110003, is prepared by us. The project work is result of our personal efforts.

Paridhi Garg
Anoushka Kumar
Class: XII D

CONTENTS

LIBRARIES USED

Plotly: Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST.

Plotly.graph_objects:

plotly.py provides a hierarchy of classes called "graph objects" that may be used to construct figures.
Graph objects provide precise data validation. So if you provide an invalid property name or an invalid property value, an exception will be raised with a helpful error message describing the problem.

1. Graph objects contain descriptions of each property as Python docstrings. You can use these docstrings to learn about the available properties as an alternative to consulting the *Full Reference*.
2. Properties of graph objects can be accessed using dictionary-style key lookup (e.g. fig["layout"]) or class-style property access (e.g. fig.layout).
3. Graph objects support higher-level convenience functions for making updates to already constructed figures, as described below.

Graph objects are stored in a hierarchy of modules under
the plotly.graph_objects package.

Pandas: Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured and time series data both easy and intuitive. Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.Pandas takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software such as excel.

Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Tkinter.ttk:

The Tk interface is located in a binary module named _tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

- These classes are provided for the purposes of organizing certain functions under one namespace. They aren't meant to be represented independently.
- The Tk class is meant to be instantiated only once in an application. Application programmers need not represent one explicitly, the system creates one whenever any of the other classes are instantiated.

Tkinter.messagebox:

The messagebox module is used to display the message boxes in the python applications. There are the various functions which are used to display the relevant messages depending upon the application requirements.
The tkinter.messagebox module provides a template base class as well as a variety of convenience methods for commonly used configurations. The message boxes are modal and will return a subset of (True, False, OK, None, Yes, No) based on the user's selection.

WORKING DESCRIPTION

The application is developed to be an assistance for planning travels according to user preferences such as temperature, precipitation and landforms. Currently the application focuses on USA only, however with provision of data for other continents it can be easily extended.

The GUI of application is developed using Tkinter widgets. The interface is designed with pleasing images and easy user interface that makes the application suitable for novice users as well.

Plotting of maps of USA for depicting information is done by using plotly library graph objects Choropleth. The maps show relevant information on mouse hover.

The data to be plotted by these maps is provided in the form of csv files.
Panda library is used for reading information from csv files
Data in csv files is prepared by referring to information from internet.

We have also added 'search history','wishlist' and user login features to incorporate mysql and python interfacing in our project. The login helps the user to create an account with an appropriate username and password and thus save material, revisit old searches and create wishlists. All of this is done by creating tables in sql and adding data to them.

Description of functions in application:

Signup and login functions:

loginClicked, signupClicked, userLoginClicked and signClicked functions are used to create accounts or to login to already existing ones. These add the username and corresponding password to a table created in sql (if signup is selected) or look for the correct username password combo if the account already exists.

Screen printing functions:

printScreen1, printScreen2, printScreen3, printScreen4, printScreen5 and printScreen6 functions are used for creating the screens for user interface. These screens use TkWidgets for showing images and Labels. User input is taken in text entry fields and buttons are used for taking command from user. clearScreen function is used for clearing the screen before creating new screen.

Style configurations are used for rendering the labels and texts in attractive font and colors.

Button Click Commands:

detailClicked, beginClicked, nextClicked, stateClicked and lastClicked functions are called back when a button is clicked. These functions process the user input.

User input that needs to be consumed in future function calls is stored in global variables

Plotting the maps:

plotTempMap, plotPptMap and plotLandMap functions are used for plotting the maps for showing information about temperature, precipitation and landforms for states in US.

The data for plotting in maps is read from csv files as per user selection. e.g. for showing temperature information for a specified month values are picked from the column for that month.

consolinfo:

This function is used for displaying consolidated information for the state the user selects for travel.

The data is read from csv file for the user mentioned state.

Search history:

This function adds every city/state selected to a table in sql where it is stored until asked to be displayed. If the button labeled search history is clicked, the user is shown a table containing all past records. This is done using python interfacing.

Wishlist:

Similar to the search history function, this function adds a selected record to a table from where it is displayed on command. Unlike the search history function, records that are added to the wishlist need to be selected by the user.

Further work:

This application can be further extended for including other continents to cover for wider travel choice. Also we can add more map options for covering other factors of relevance while planning travel

We can also add interesting graphs for selected destination showing the tourist trend for the travel month etc.

PROJECT CODE

```
from tkinter import *
from tkinter.ttk import *
from tkinter.messagebox import showerror

import plotly.graph_objects as go

# Load data frame and tidy it.
import pandas as pd
path="C:\\wander project\\"

window = Tk()

#global variables
name=""
month=""
state=""
info=""
style = Style()


#function to clear screen
def clearScreen():
    list = window.pack_slaves()
    for l in list:
l.destroy()

#functions called on button clicks
def detailClicked():
    global name
    global month

    name=txt1.get()
    month=txt2.get()
    month=month.lower()
    if month=="":
showerror(title='input  missing',message='Please  enter
month')
    else :
clearScreen()
        printScreen3()

def beginClicked():
clearScreen()
    printScreen2()

def nextClicked():
clearScreen()
    printScreen4()

def stateClicked():
    global state

    state=txt3.get()
    state=state.capitalize()

    if state=="" :
showerror(title='input  missing',message='Please  enter
state')
    else:
clearScreen()
        printScreen5()

def lastClicked():
clearScreen()
    printScreen6()

#functions for printing screens
def printScreen1():
```

```
style.configure("BW.TLabel",        foreground="white",
background="slate grey",
                           font= 'Helvatica 24 bold',
borderwidth=20,width= 24, anchor=CENTER,relief=RAISED)
    label = Label(top_frame, image = img1).pack()
btn = Button(bottom_frame, text = "Let the adventure
begin..",style="BW.TLabel",command=beginClicked).pack(
side = "bottom")


def printScreen2():
    label = Label(top_frame, image = img2).pack()
    # style configuration for label
style.configure("Label", font= 'Helvatica 10 bold')

    lbl1 = Label(bottom_frame, text="Please enter your
name ",style="Label").pack()
    txt1.pack()
    lbl2 = Label(bottom_frame, text="In which month do
you want to travel ",style="Label").pack()
    txt2.pack()

    # style configuration for button
style.configure("BW.TLabel",        foreground="white",
background="slate grey",
                          font= 'Helvatica 12 bold',
borderwidth=5,width= 10, anchor=CENTER,relief=RAISED)
btn               =               Button(bottom_frame,
text="Next..",style="BW.TLabel",
command=detailClicked).pack(side="bottom")

def printScreen3():
    label = Label(top_frame, image = img3).pack()
    # style configuration for button
style.configure("BW.TLabel",        foreground="white",
background="cyan4",
                           font= 'Helvatica 12 bold',
borderwidth=5,width= 40, anchor=CENTER,relief=RAISED)
      btn1 = Button(bottom_frame, text = "Temperature
pattern..",style="BW.TLabel",command=plotTempMap).pack
(side = "top")

     btn2 = Button(bottom_frame, text = "Precipitation
pattern..",style="BW.TLabel",command=plotPptMap).pack(
side = "top")

       btn3 = Button(bottom_frame, text = "Landform
Distribution..",style="BW.TLabel",command=plotLandMap)
.pack(side = "top")


            btn5 = Button(bottom_frame, text =
"Next..",style="BW.TLabel",command=nextClicked).pack(s
ide = "bottom")


def printScreen4():
    label = Label(top_frame, image = img4).pack()
    global name

    # style configuration for label
style.configure("Label", font= 'Helvatica 10 bold')
    text1= "Dear " +name + ",Enter name of state "
                    lbl1 = Label(bottom_frame,
text=text1,style="Label").pack()

    txt3.pack()
    # style configuration for button
style.configure("BW.TLabel",        foreground="white",
background="slate grey",
                           font= 'Helvatica 12 bold',
borderwidth=5,width= 10, anchor=CENTER,relief=RAISED)
```

```
btn                =                Button(bottom_frame,
text="Next..",style="BW.TLabel",
command=stateClicked).pack(side=BOTTOM)

def printScreen5():
    label = Label(top_frame, image = img5).pack()
    global state
    global info

     # style configuration for label
style.configure("Label",  font=  'Helvatica  10  bold',
space=4)
consolinfo()
       lbl1  =  Label(bottom_frame,  text="Consolidated
Information          about          "+state+"\n"+info,
style="Label").pack()

    # style configuration for button
style.configure("BW.TLabel",       foreground="white",
background="slate grey",
                           font=  'Helvatica  12  bold',
borderwidth=5,width= 10, anchor=CENTER,relief=RAISED)
btn                =                Button(bottom_frame,
text="Next..",style="BW.TLabel",command=lastClicked).p
ack(side=BOTTOM)

def printScreen6():
    label = Label(window, image = img6).pack()

#functions for plotting maps
def plotTempMap():
        global month

df = pd.read_csv(path+'US_monthly_temperature.csv')

        fig = go.Figure(data=go.Choropleth(
                      locations=df['code'],  #  Spatial
coordinates
            z = df[month], # Data to be color-coded
locationmode = 'USA-states', # set of locations match
entries in `locations`
colorscale = 'thermal',
colorbar_title = "Temperature(°F)",
hoverinfo= "all",
            text=df['state']
        ))

fig.update_layout(
title_text = 'US temperature by State',
geo_scope='usa', # limite map scope to USA
        )

fig.show()

def plotPptMap():
df = pd.read_csv(path+'US_monthly_precipitation.csv')

        fig = go.Figure(data=go.Choropleth(
                      locations=df['code'],  #  Spatial
coordinates
             z = df[month].astype(float), # Data to be
color-coded
locationmode = 'USA-states', # set of locations match
entries in `locations`
colorscale = 'teal',
colorbar_title = "Precipitation",
hoverinfo= "all",
            text=df['state']
        ))

fig.update_layout(
title_text = 'US precipitation by State',
geo_scope='usa', # limite map scope to USA
        )

fig.show()

def plotLandMap():
df = pd.read_csv(path+'US_landforms.csv')

        fig = go.Figure(data=go.Choropleth(
                      locations=df['code'],  #  Spatial
coordinates
            z = df['tc'], # Data to be color-coded
locationmode = 'USA-states', # set of locations match
entries in `locations`
colorscale = 'sunset',
colorbar_title = "Landforms",
hoverinfo= "all",
            text=df['Landform'],
showscale=False
        ))

fig.update_layout(
title_text = 'US landforms by State',
geo_scope='usa', # limite map scope to USA
        )

fig.show()



#function for fetching consolidated info for a state
def consolinfo():
        global info
df = pd.read_csv(path+'consolidated.csv')
         info="Landform type :" + df[state].values[0] +
"\n"    +    "Annual    high    temperature    :"    +
df[state].values[1]  +  "\n"+"Annual  low  temperature
```

```
:"+df[state].values[2]  +  "\n"+"Annual  precipitation
:"+df[state].values[3]  +  "\n"+"Best  place  to  visit
:"+df[state].values[4] + "\n"


window.title("Welcome to WanderUSA")
window.geometry('620x640')
window.resizable(0, 0)

top_frame = Frame(window).pack()
bottom_frame = Frame(window).pack(side = "bottom")

# show image in top frame
img1    =    PhotoImage(file    =    path+"\\pictures
project\\wonder wander repeat.png")
img2    =    PhotoImage(file    =    path+"\\pictures
project\\img2.png")
img3    =    PhotoImage(file    =    path+"\\pictures
project\\img3.png")
img4    =    PhotoImage(file    =    path+"\\pictures
project\\img4.png")
img5    =    PhotoImage(file    =    path+"\\pictures
project\\img5.png")
img6    =    PhotoImage(file    =    path+"\\pictures
project\\img6.png")

txt1 = Entry(window,width=20,justify=CENTER)
txt2 = Entry(window,width=20,justify=CENTER)
txt3 = Entry(window,width=20,justify=CENTER)
printScreen1()

window.mainloop()
```

OUTPUT SCREENS

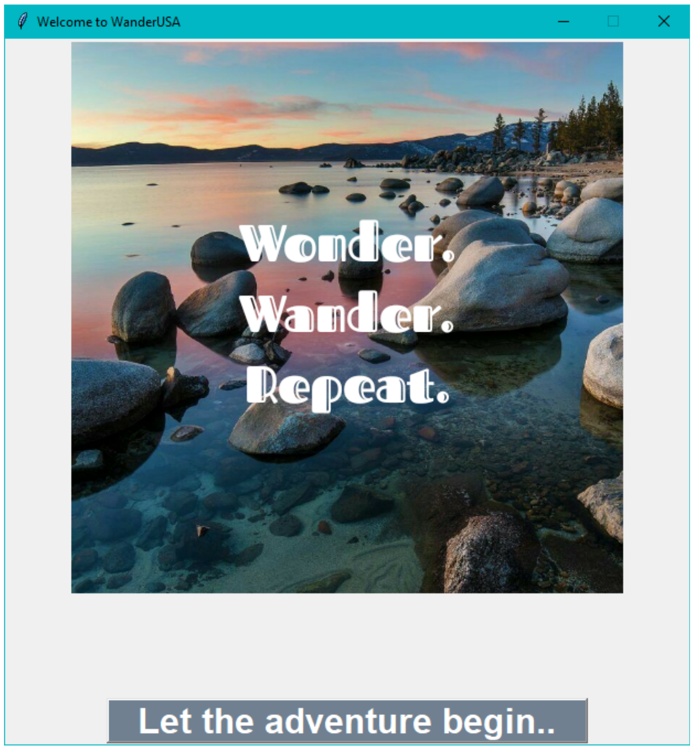
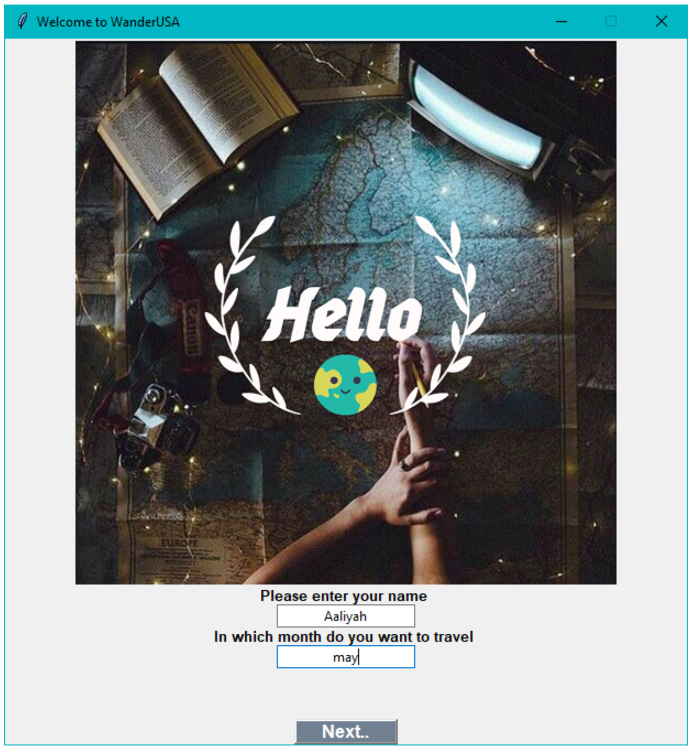


Figure 1 Welcome Screen



Please enter your name

Aaliyah

In which month do you want to travel

may

Figure 2 Entry screen for user data

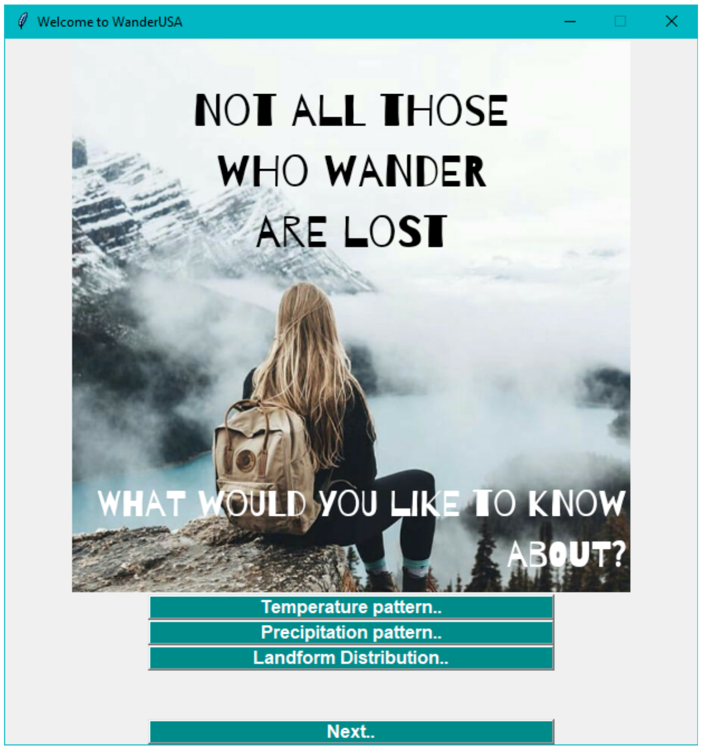


Figure 3 Plot Maps to understand about travel options

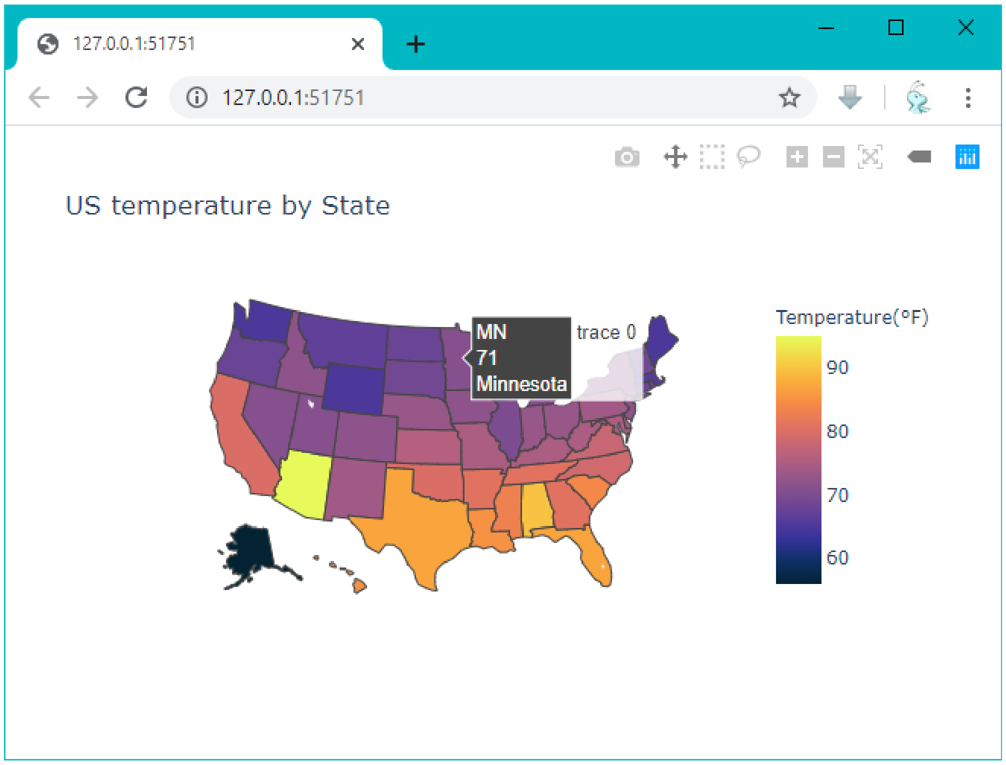


Figure 4 Map showing temperature distribution

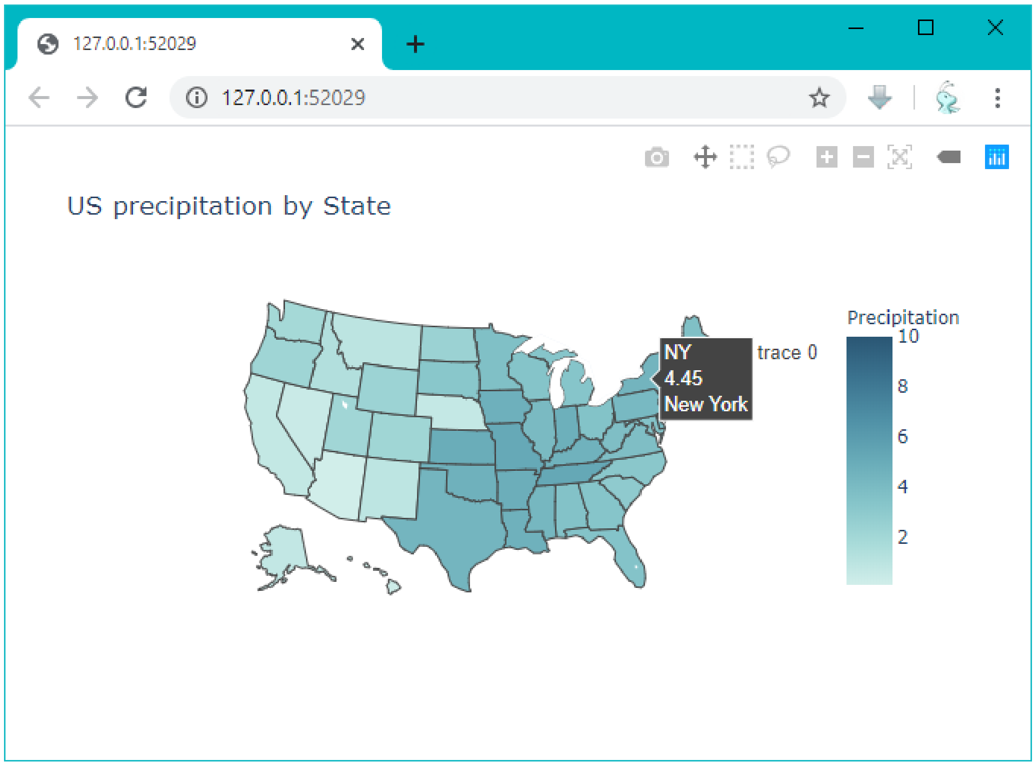
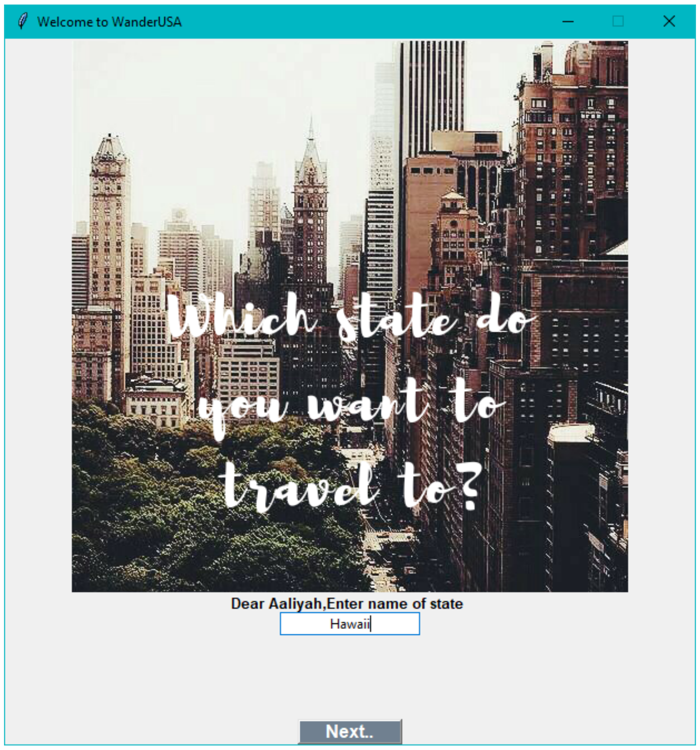


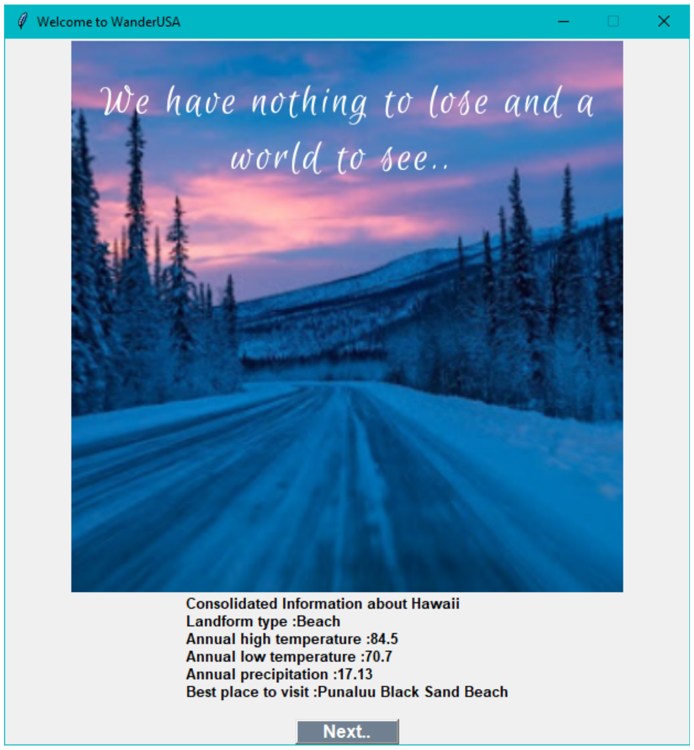
Figure 5 Map showing precipitation distribution



Dear Aaliyah,Enter name of state

Hawaii

Figure 6 User selected state for travel



Consolidated Information about Hawaii
Landform type :Beach
Annual high temperature :84.5
Annual low temperature :70.7
Annual precipitation :17.13
Best place to visit :Punaluu Black Sand Beach

Figure 7 Consolidated information for selected destination

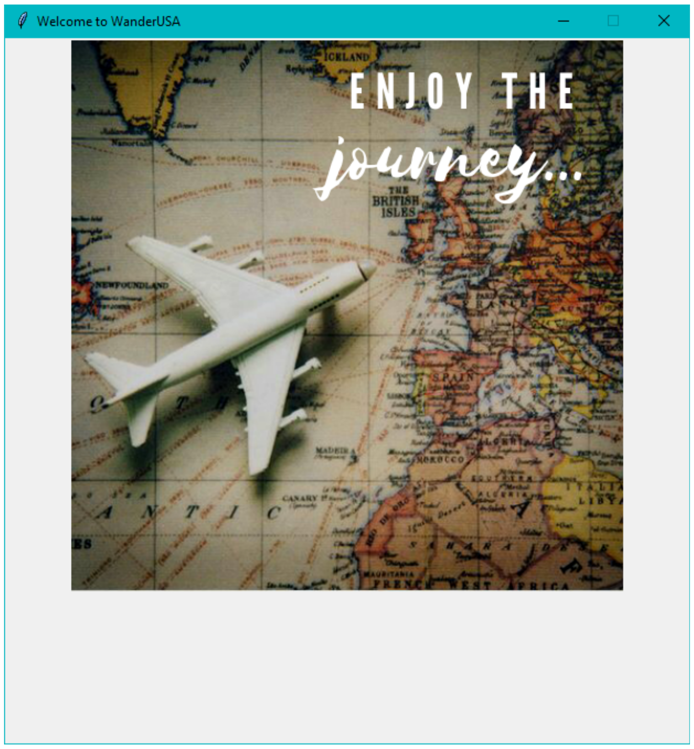


Figure 8 last screen

CONCLUSION

Our project, effectively, is a travel planning app. It shows the various essential pieces of information required to plan a trip. It takes an input from the user and then shows the relevant maps with temperature, precipitation etc. in the end, it shows the consolidated data such as temperature, precipitation and attraction of chosen destination

Therefore our project is a basic guide to tourism in the US and a helpful alternative to the hassle of various trip websites and planning troubles ☺

BIBLIOGRAPHY

1. Sumita Arora, *Computer Science with Python*, 2019
2. https://www.usclimatedata.com/
3. https://www.thrillist.com/travel/nation/grand-canyon-disney-world-wrigley-field-and-the-best-attractions-to-see-in-the-usa
4. https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States
5. https://www.tripadvisor.in/
6. https://plot.ly/python/choropleth-maps/
7. Hover Text and Formatting | Python | Plotly
8. https://pandas.pydata.org/pandas-docs/stable/user_guide/options.html
9. https://effbot.org/tkinterbook/tkinter-index.htm