Computer Engineering Program

MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

CNG 400

SUMMER PRACTICE REPORT

Name of Student **:** Fatma Erem Aksoy

ID Number **:** 2315075

Name of Company **:** İnnova Bilişim Çözümleri A.Ş

Project Title : AI Development with Google Dialogflow (Chatbot) and Python Projects

Date of Submission : 15 November 2022

ABSTRACT

This report has all the necessary information about my summer internship experience at Innova Bilişim Çözümleri A.Ş during 20 working days. The main objective for my practice was improving my Python skills, learning new programming languages, getting familiar with the web related concepts with these languages, and creating a chatbot which is a more basic version of a digital assistant. I started doing projects with Python first and used some libraries and modules I was not familiar with before. I created games and then continued doing different purposed websites and platforms with HTML and CSS. Learning HTML and CSS was an enjoying part of my practice as I am able to design the website according to my interests and play with the features using various types of built-in functions. By practicing more on Python, I learned how to write a cleaner and more efficient code, and debug more safely. Some of the tasks I worked on was focused directly on debugging, so I had some time to analyze my current debugging skills and improve them as much as possible because I know debugging is one of the most important concepts in programming. I am glad for the working experience, the skills I have improved and gained, and other crucial skills including analytical thinking, effective communication and problem solving. I think internship has given me a ton of new comprehensions from many perspectives because I've been given daily chores about various topics. I can list my gains as learning new languages, HTML and CSS, using a development tool, PyCharm, doing data labelling and getting familiar with the Google Dialogflow and learning to create a chatbot.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

1. DESCRIPTION OF THE COMPANY

In this section, detailed information about the company will be given. The information is divided into three parts as shown below.

1.1 Name, History, Location and Services

Innova Bilişim Çözümleri A.Ş is Turkey's leading IT solutions company with its professional staff of over 1400+ people with knowledge in different technologies. The name Innova, which comes from the root "innovate" meaning to renew, renew, symbolizes our mission to create new values for our customers by using innovative approaches and the possibilities of new technologies.

Offering platform-independent solutions to organizations in every sector, especially in the telecommunication, finance, production, public and service sectors since 1999, Innova has managed to export the solutions it produces in international standards to 37 countries in 4 continents so far.

Innova Bilişim Çözümleri A.Ş, which has been a subsidiary of Türk Telekom since 2007, continues its activities through a total of 14 offices spread over various regions of Turkey, together with its main offices in Istanbul and Ankara.

The services they provide are: Consultancy services, Application development, Managed services and Cloud services.

1.2 Organizational Structure

Professional staff of 1400+ people consisting of experienced software and solution experts, project managers, consultants and analysts who have managed and developed large-scale application development projects in Turkey.

Taking its strength from its team of experienced IT engineers and its working principles, Innova works by believing that every good service it offers will return as success, and sets its priorities by considering the benefits of its customers.

Innova, which constantly works and learns for new products and services that can meet these needs by anticipating the changing needs of its customers, attaches great importance to

teamwork within its organizational structure. As a basic principle, while supporting individual creativity, on the other hand, it carefully carries out teamwork, which is the quality assurance of the works. Innova contributes to international R&D projects that will shed light on the world of the future.

Its end-to-end solutions, which are spreading rapidly around the world, turn Innova into an international R&D company. Actively involved in R&D projects supported by important international organizations such as the European Union, İnnova contributes to projects carried out to shape the future of technology and set standards. Goal: To turn technology into a more efficient tool for everyone than ever before.

Serdar Toraman graduated from Istanbul University Electronics Engineering Department in 1999. Starting his career at IBM, Toraman worked in Sentim Information Technologies, ServisNET Telecommunication Services within NETAŞ, Eczacıbaşı Bilişim in various positions.

Toraman, who took on important roles in leading companies in the field of information technologies, worked in managerial positions at Avnet Technology (TechData), Borsa Istanbul and Information Technologies and Communications Authority. Later, Toraman worked as IT Director and Consultant at Türk Telekom subsidiary AssisTT Guidance and Customer Services company, and most recently, he served as the General Manager of PTT Bilgi Teknolojileri A.Ş. Serdar Toraman became the CEO of Innova as of December 7, 2020.

1.3 Personal Observations

During my summer internship, the office that I worked at was well-organized. The workers were always synchronized and able to solve problems effectively by helping each other. The way they were always willing to help made me feel comfortable in there and boosted my motivation. I have a couple of reasons to choose this company for the summer practice. I saw in the summer internship system that the company is available in ODTU(Ankara Campus) Teknokent. So I thought this company would be a great opportunity for me to improve myself. After checking the company website and seeing that the services they provide and the areas they work in match my interests, I applied for the internship. For more details about the company, you can check their website [1].

2. INTRODUCTION

I have done my summer internship at Innova Bilişim Çözümleri A.Ş, Ankara, Turkey. During my practice, the working environment there was so calming and encouraging as the office was big enough to support working individually and with teams as well. Everyone was willing to help each other in different projects to find solutions to others' problems as much as they can. I admire the knowledge everyone has and the way they work synchronously. I worked with Ersin Aksoy, the head of AI unit in the company, and Onur Odabaşı who guided me throughout the internship as my supervisor.

I was given daily or sometimes weekly tasks and expected to finish them based on a planned program. The first 2 weeks of my internship is focused on improving my practice with Python by doing multiple projects on different concepts, and data labeling for defect detection in Türk Telekom modems. I also learned HTML and CSS to help me do some of the projects. I used Jupyter notebook and labelImg for data labeling. In the data labeling project, I helped with the labeling part and it was a different field for me to work. That project was aiming to use a model and teach it how to find the issue in modem by looking at the labels that show which leds are on or off for which defect. I also worked on many minor projects in different fields and worked with some libraries and modules that I did not know before such as Turtle, Beautifulsoup, Pandas, Selenium and Requests. For the last 2 weeks, I worked with Google Dialogflow to create a chatbot like a basic version of a digital assistant. I learned the basics of Dialogflow to create a chatbot and then connect it to other applications, Facebook is this application in my case, by the integration method with some steps which will be further explained in related sections of the report.

I believe I did my best to benefit from this summer practice and gained as much experience as I can to improve my current knowledge and add some more to it. I am interested in AI field and I did not have any experience with it before my internship so it was a very fun and great opportunity for me to practice it other than focusing only on the research part of this area. I believe this practice was necessary for me to decide the area that I like working on the most for my future goals.

The details of the projects I did and my gains during the internship will be discussed in the rest of the report. I will give some pieces of information about the projects I worked on and most of the tasks I have given. I will also provide some figures to support my work. The problems I faced and their solutions, the ones I could solve with/out the help of my supervisor, will be

mentioned with details as well. A Conclusion section will be added to the very end to summarize all the work I did.

3. PROBLEM STATEMENT

I worked on several minor projects and a major project during my practice. I have started learning about HTML and CSS, and practicing in Python as I used these languages and also some libraries and modules related to them in the minor projects. I used PyCharm IDE as a tool to code on Python, HTML and CSS as well. I found PyCharm really easy to use and a very handy tool as I can use it with more than one language. The libraries and modules that I used for the minor projects are Turtle [2], Tkinter [3], Pandas [4], Selenium [5] and Requests [6]. I did some research and read documentations before using them in the projects (More details are provided with the links in the related part of the References section). After improving my Python skills and learning HTML and CSS, I was more comfortable with web based concepts. Then I learned the basics of Google Dialogflow to blend my knowledge of web based concepts with it to create a simple chatbot with some specific features and connect it to an application, Facebook, as the major project.

I also did some data labelling for defect detection for Türk Telekom modems. The company had a project to train models by using the labelled data so that the defect detection could be done without anyone checking for it but an application itself when a problem occurs in any of the modems. This area was quite different and interesting for me because it was my first time working on modems and labelling, so I really enjoyed the experience.

Finally, I can say that I worked on several tasks and different areas to broad my knowledge, and to be more versatile. I used many platforms and languages to test my knowledge and add up more to it. Sometimes I couldn’t solve the problems I faced but I asked for help and also did research to manage them as much as I can. I am quite satisfied with the result at the end of all these projects and this report will be covering all the necessary detailed information about them.

4. PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. I have chosen PyCharm as an IDE platform for my projects because I already had an experience using it for my interests from previous summer. I think it is very handy and user-friendly as its interface is simple and it provides so many useful options for the projects. It has visual debugging, version control integration and also code assistance which make it a smart and easy to use development environment. One of the other advantages is that I knew I was going to work with HTML and CSS as well, and PyCharm fully supports these languages. I have worked with version 2021.2 of PyCharm. More detailed information can be found on their website [7].

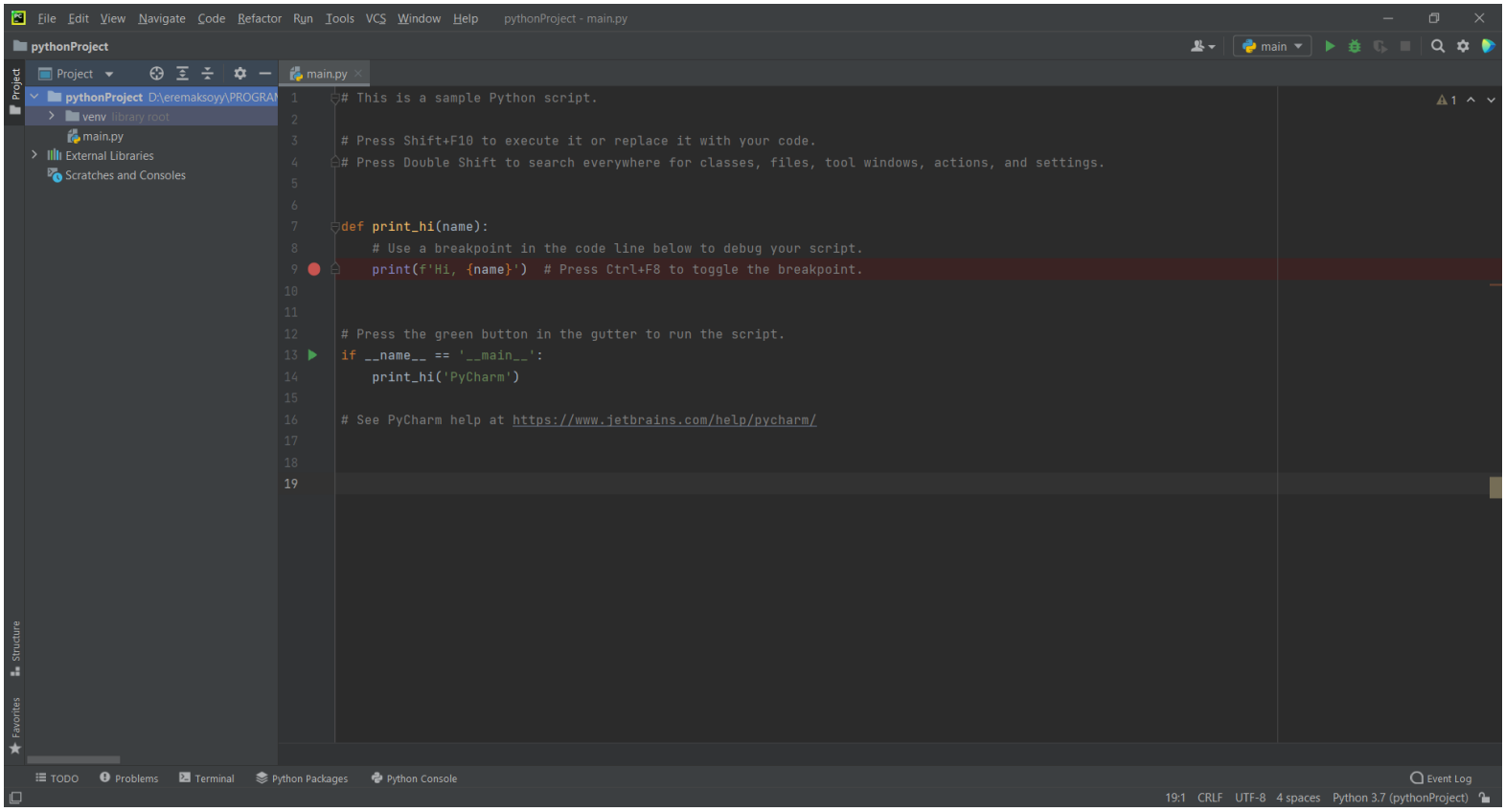


Figure 1: PyCharm 2021.2

4.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. It is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. [8]

There are many reasons that I have chosen Python such as it is open-source, powerful, fast, easy to use and integrate with other languages, and both its standard library and the community-contributed third-party modules create opportunities for endless possibilities. I also wanted to use its object oriented paradigm as well to practice the object oriented structure in another language other than C++. I have worked with Python 3.7 in my projects. You can find more detailed information about Python in their website [9].

4.2 HTML

HTML is the language in which most websites are written. HTML is used to create pages and make them functional. HTML stands for Hyper Text Markup Language. Hypertext means that the document contains links that allow the reader to jump to other places in the document or to another document altogether and a Markup Language is a way that computers speak to each other to control how text is processed and presented. To do this, HTML uses two things: tags and attributes.

Tags and attributes are the basis of HTML. Tags are used to mark up the start of an HTML element and they are usually enclosed in angle brackets, like <h1> for instance. Attributes on the other hand contain additional pieces of information. Attributes take the form of an opening tag and additional info is placed inside, such as `<img src="dog.jpg" alt="A photo of a dog.">` .

The reason I have chosen HTML is because it is easily customizable, beginner-friendly, and easy to learn. I found its syntax very easy to learn and work on with even though it is not very efficient. I also thought I could easily solve my problems when I faced one as there are so many sources about HTML in the internet, so the fact that it is a very used - common language helped me with the issues a lot. I used HTML5 for the projects. You can get more detailed information from the website [10].

4.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting [11].

I wanted to use CSS as it can be used with HTML and it has more advanced features comparing to HTML so I thought HTML and CSS would be a good combination of two languages to learn and practice together. For more information and examples about CSS, you can visit the website [12].

4.4 Libraries and Modules Used

I have used Turtle module, Pandas library, and Tkinter, Selenium and Requests packages in my minor projects. Some details and functions that I used from them will be discussed in this section of the report.

4.4.1 Turtle

Turtle is a Python library which used to create graphics, pictures, and games. It allows designing unique shapes, attractive pictures and various mini games. I have used it to design mini games and graphics such as Hirst Painting, Etch a Sketch, Turtle Coordinate System, Snake Game, Pong Game and Turtle Crossing Game. Some of the methods I used for these projects are forward(), backward(), right(), left(), penup(), up(), down(), goto(), color(), and shape(). To talk about the purpose of the methods I have used, I can explain their functions shortly as:

forward(): It moves the turtle forward by the specified amount

backward(): It moves the turtle backward by the specified amount

right(): It turns the turtle the turtle clockwise by the specified angle

left(): It turns the turtle the turtle counterclockwise by the specified angle

penup(): It picks up the turtle’s pen

up(): It picks up the turtle’s pen

down(): It picks down the turtle’s pen

goto(): it moves the turtle to position x, y entered as parameters

color(): It changes the color of the turtle’s pen

shape(): It changes the shape of the turtle’s pen. Should be ‘arrow’, ‘classic’, ‘turtle’ or ‘circle’ as they are defined as default values.

For more details about the Turtle library and other methods available, you can check the website [2].

4.4.2 Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Through pandas, you get acquainted with your data by cleaning, transforming, and analyzing it. For instance, pandas will extract the data from the CSV file that you have in your computer into a DataFrame, a table. The reason that I wanted to work with this library is that I wanted to try out different libraries from different fields and I am interested in data science so I thought it is a great opportunity for me to get familiar with this package and do some minor projects using it. The project I used pandas library is US State Game. All the details about the projects will be covered in the related section, and more details about the Pandas package can get from [4].

4.4.3 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 provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. A Tkinter user interface is made up of individual widgets. Each widget is

represented as a Python object, instantiated from classes like `ttk.Frame`, `ttk.Label`, and `ttk.Button`, and the ones I have used in my projects are Label, Button, Entry, Canvas and Photo Image.

I also used many methods such as grid(), config(), title(), destroy(), mainloop(), itemconfig(), after(), after_cancel() and minsize(). The most important ones are the grid() which is used to specify the relative layout (position) of the label within its containing frame widget, similar to how tables in HTML work; the destroy() which is the root window; the config() which updates multiple attrs subsequent to object creation; and finally the mainloop() which puts everything on the display, and respond to user input until the program terminates. The projects that I used Tkinter are Pomodoro Timer and Flash Card Project.

For more details, you can check the website [3].

5. SOLUTION

I have used different languages and platforms for my work. Each one of them is discussed with all the necessary details in this section. The code and the output graphics of the projects are available as figures as well.

5.1 Projects

5.1.1 Hirst Painting

This is the first project I used Turtle library and it takes an image that I chose from the internet and takes all the colors in that image, and uses those pixels to recreate a graphics with random dots of those colors in a random order. Dot number is user defined so it can be changed accordingly. First we add the jpg file to the project, the image we chose, then get the pixels of all the colors in the image with the proper lines of code. And at the end, we get the output of a specified number of dots with random colors from the image we added. The direction of the path is also defined by me so it can be changed as well by changing the value used in the forward() and setheading() methods.

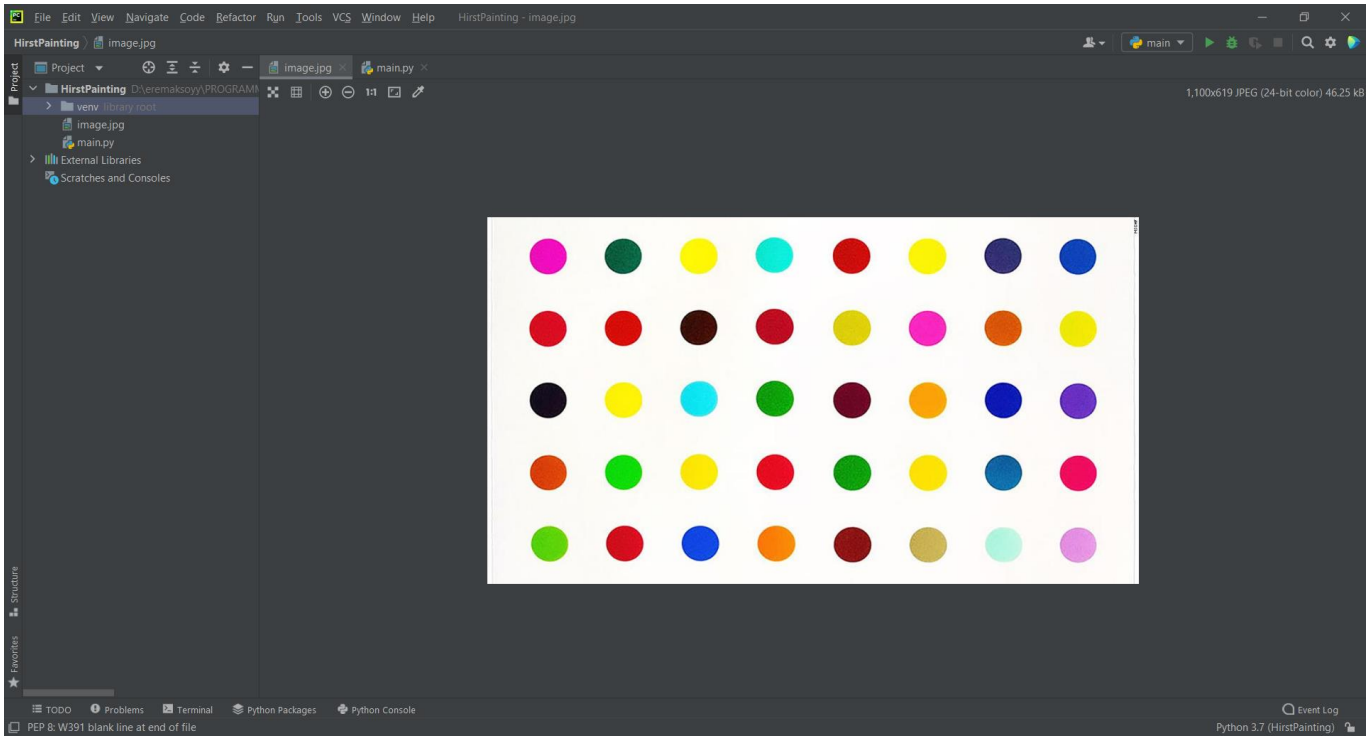


Figure 2: Chosen image to be used for its colors

image.jpg main.py

```
import turtle as turtle_module
import colorgram
import random

rgb_colors = []
colors = colorgram.extract('image.jpg', 30)

# or color in colors:
r = color.rgb.r
g = color.rgb.g
b = color.rgb.b
new_color = (r, g, b)
rgb_colors.append(new_color)

print(rgb_colors)

turtle_module.colormode(255)
tim = turtle_module.Turtle()
tim.speed("fastest")
tim.penup()
tim.hideturtle()

# getting the pixels in (r,g,b) format from the above code and create a list with those values and then comment the
# above code, except the libraries, to proceed with the code below.
color_list = [(234, 251, 243), (197, 13, 32), (249, 237, 21), (40, 76, 188), (39, 216, 69),
              (238, 227, 5), (228, 160, 48), (244, 247, 253), (28, 40, 155), (213, 75, 13), (16, 153, 16),
              (198, 15, 11), (242, 34, 163), (226, 19, 121), (75, 9, 31), (59, 15, 9), (224, 141, 208), (11, 97, 62),
              (220, 158, 10), (18, 18, 42), (49, 212, 232), (238, 156, 218), (11, 228, 239), (80, 74, 213),
              (75, 211, 166), (83, 234, 199), (58, 232, 241), (5, 67, 42)]

```

```
tim.setheading(225)
tim.forward(300)
tim.setheading(0)
number_of_dots = 100

for dot_count in range(1, number_of_dots + 1):
    tim.dot(20, random.choice(color_list))
    tim.forward(50)

    if dot_count % 10 == 0:
        tim.setheading(90)
        tim.forward(50)
        tim.setheading(180)
        tim.forward(500)
        tim.setheading(0)

screen = turtle_module.Screen()
screen.exitonclick()
```

Figure 3: Hitch Painting Main Code

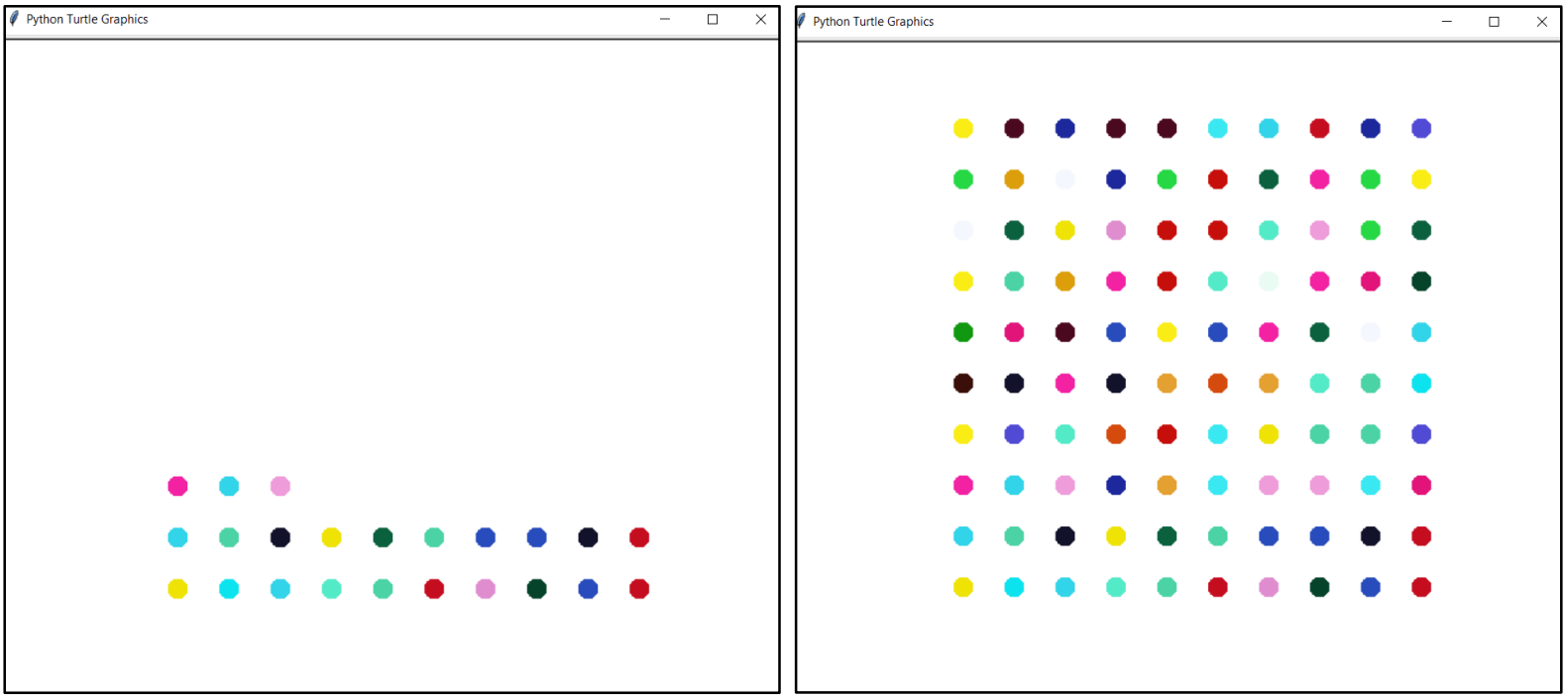


Figure 4: Hitch Painting Output at a random time and at the end

5.1.2 Etch a Sketch

In this mini project, the user should use the “w”, “s”, “a” and “d” letters on the keyboard to direction the pen, and “c” letter to clean the board. It is defined how many units the pen will be moved in each pick of letters so it can be changed by changed the value in the methods forward(), backward(), and left(). Random shapes will be drawn by the user according to their moves and the program will stop executing until the user exit the game.

main.py

```
from turtle import Turtle, Screen

tim = Turtle()
screen = Screen()

def move_forwards():
    tim.forward(20)
def move_backwards():
    tim.backward(20)
def turn_left():
    tim.left(20)
    # new_heading = tim.heading() + 20  #These 2 lines of code do the same thing as left() function
    # tim.setheading(new_heading)
def turn_right():
    # tim.right(20)
    new_heading = tim.heading() - 10
    tim.setheading(new_heading)
def clear():
    tim.clear()
    tim.penup()
    tim.home()
    tim.pendown()

screen.listen()
screen.onkey(move_forwards, "w")
screen.onkey(move_backwards, "s")
screen.onkey(turn_left, "a")
screen.onkey(turn_right, "d")
screen.onkey(clear, "c")
screen.exitonclick()
```

Figure 5: Etch a Sketch Main Code

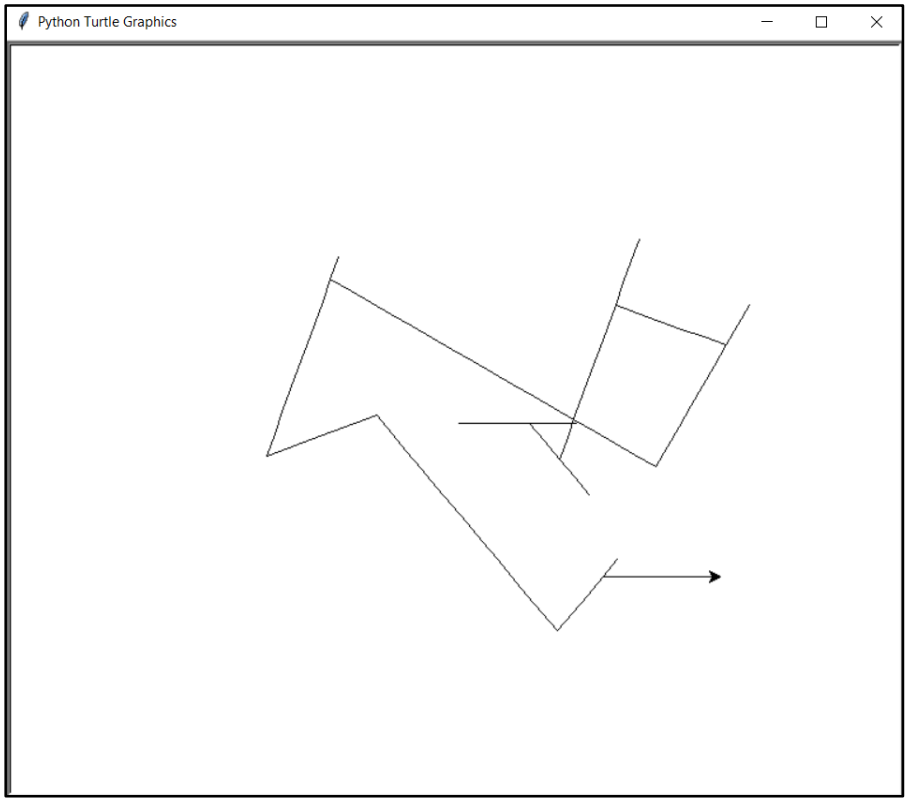


Figure 6: Etch a Sketch Output

5.1.3 Turtle Coordinate System

In this projects, there will be six turtles in different colors and they will be a competition between them. The speed of all the turtles will be different but it will not be predefined but randomly assigned after the program executes. The program will ask the user to enter a color of the turtle that they think will win the race. At the end of the race, the program will tell the user if their picked turtle won the race or not and if not then it will tell the color of the turtle who won. Their position assigned as they all will start and end the game at the same level (coordinates) so it can be changed which color will locate where.

main.py

```python
from turtle import *
import random

is_race_on = False
screen = Screen()
screen.setup(width=500, height=400)
user_bet = screen.textinput(title="Make your bet", prompt="Which turtle will win the race? Enter a color: ")
colors = ["red", "orange", "yellow", "green", "blue", "purple"]
y_positions = [-70, -40, -10, 20, 50, 80]
all_turtles = []
for index in range(len(colors)):
    new_turtle = Turtle(shape="turtle")
    new_turtle.penup()
    new_turtle.color(colors[index])
    new_turtle.goto(x=-230, y=y_positions[index])
    all_turtles.append(new_turtle)
if user_bet:
    is_race_on = True
while is_race_on:
    for turtle in all_turtles:
        if turtle.xcor() > 230:
            is_race_on = False
            winning_color = turtle.pencolor()
            if winning_color == user_bet:
                print(f"You've won! The {winning_color} turtle is the winner! ")
            else:
                print(f"You've lost! The {winning_color} turtle is the winner!")
        rand_distance = random.randint(0, 10)
        turtle.forward(rand_distance)
screen.exitonclick()
```

Figure 7: Turtle Coordinate System Main Code

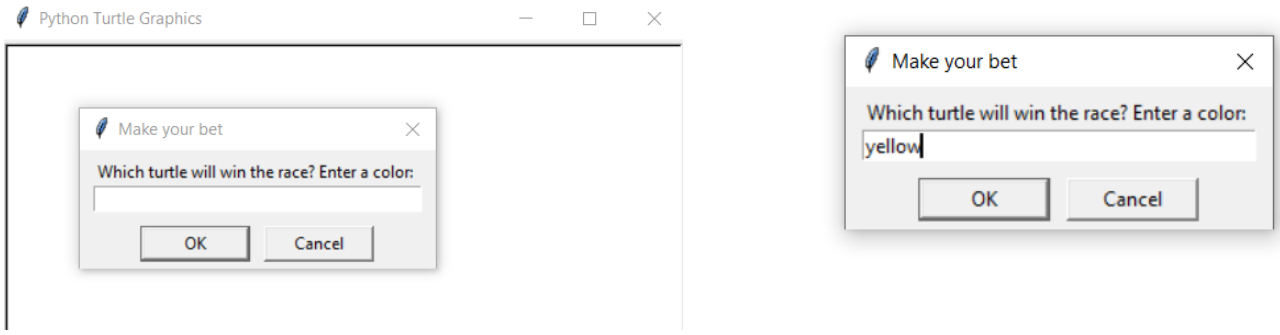


Figure 8: Message Box to Ask the User Which Turtle They Support

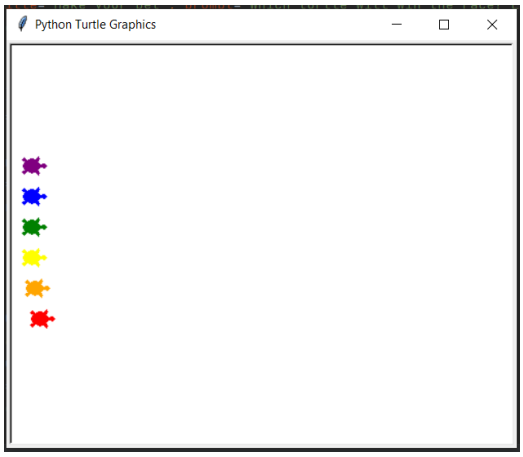


Figure 9: The race start

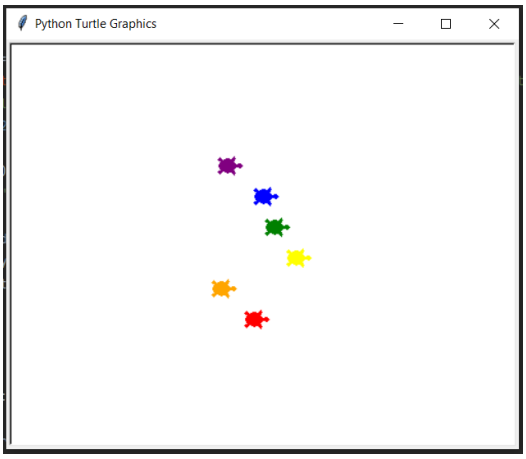


Figure 10: During the race

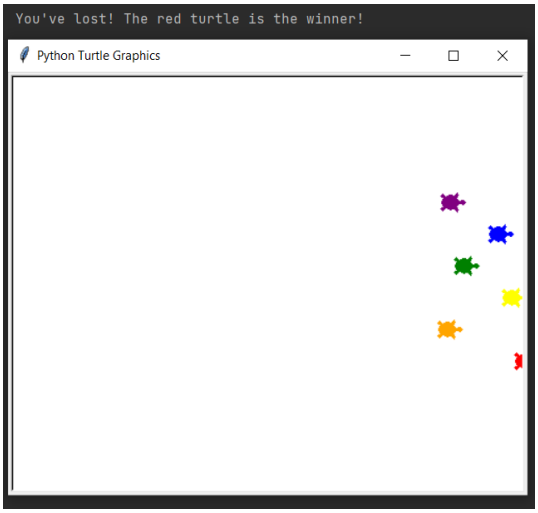


Figure 11: End of the Race and the Message

5.1.4 Snake Game

With this game, I have tried object oriented paradigm for the first time in Python. It was a great experience for me to integrate OOP concept here because it makes everything easier to understand and fix in case any problem occurs. I had some basic knowledge about OOP with C++ from CNG242 Programming Language course, and now I also used my knowledge to practice it in another language which I believe improved my Python skills.

This game is the classical snake game that the user directions the snake to collect the food and grow without touching the edges. When the snake touch any of the edges, in other words the out of the frame range, it starts moving from the starting point again. For a user to be game over, they needs to touch the snake itself, so if a snake touches its tail when it gets longer, then it is game over. The score will be shown during the game and it will also be saved to a txt file after the program executes, so the user can check the highest score they did from there as well. Also note that the score will be saved and start from 0 again for that level again, and also get back to its original size when the snake touched the edges. There will be three classes which are for snake, food, and score board.

The OOP concept is used with those classes and the methods for each of the operations in those classes. The classes and outputs are shown below with the figures.

main.py | snake.py | food.py | scoreBoard.py | data.txt

```python
from turtle import Turtle

STARTING_POSITIONS = [(0, 0), (-20, 0), (-40, 0)]
MOVE_DISTANCE = 20
UP = 90
DOWN = 270
LEFT = 180
RIGHT = 0

class Snake:
    def __init__(self):
        self.segments = []
        self.create_snake()
        self.head = self.segments[0]

    def create_snake(self):
        for position in STARTING_POSITIONS:
            self.add_segment(position)

    def add_segment(self, position):
        new_segment = Turtle("square")
        new_segment.color("white")
        new_segment.penup()
        new_segment.goto(position)
        self.segments.append(new_segment)

    def reset(self):
        for seg in self.segments:
            seg.goto(1000, 1000)
        self.segments.clear()
        self.create_snake()
        self.head = self.segments[0]

    def extend(self):
        self.add_segment(self.segments[-1].position())

    def move(self):
        for seg_num in range(len(self.segments) - 1, 0, -1):  # the parameters in the function in start-stop-step order.
            new_x = self.segments[seg_num - 1].xcor()
            new_y = self.segments[seg_num - 1].ycor()
            self.segments[seg_num].goto(new_x, new_y)
        self.head.forward(MOVE_DISTANCE)

    def up(self):
        if self.head.heading() != DOWN:
            self.head.setheading(UP)

    def down(self):
        if self.head.heading() != UP:
            self.head.setheading(DOWN)

    def left(self):
        if self.head.heading() != RIGHT:
            self.head.setheading(LEFT)

    def right(self):
        if self.head.heading() != LEFT:
            self.head.setheading(RIGHT)
```

Figure 12: Snake Class

main.py | snake.py | food.py | scoreBoard.py | data.txt

```python
from turtle import Turtle
import random


class Food(Turtle):

    def __init__(self):
        super().__init__()
        self.shape("circle")
        self.penup()
        self.shapesize(stretch_len=0.5, stretch_wid=0.5)
        self.color("blue")
        self.speed("fastest")
        random_x = random.randint(-280, 280)
        random_y = random.randint(-280, 280)
        self.goto(random_x, random_y)
        self.refresh()

    def refresh(self):
        random_x = random.randint(-280, 280)
        random_y = random.randint(-280, 280)
        self.goto(random_x, random_y)

```

Figure 13: Food Class

main.py | snake.py | food.py | scoreBoard.py | data.txt

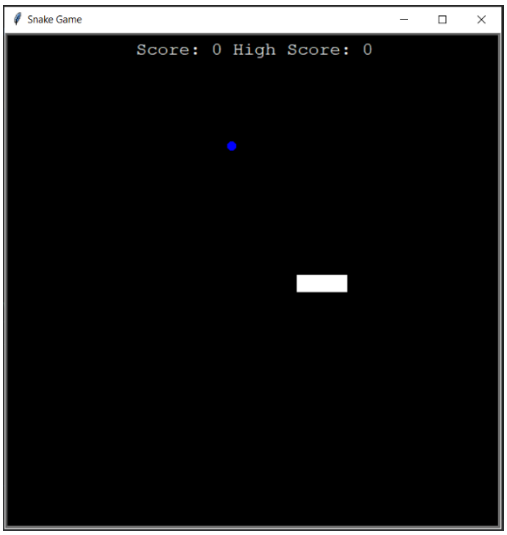
```python
from turtle import Turtle
ALIGNMENT = "center"
FONT = ("Courier", 16, "normal")

class ScoreBoard(Turtle):
    def __init__(self):
        super().__init__()
        self.score = 0
        with open("data.txt") as data:
            self.high_score = int(data.read())
        self.color("white")
        self.penup()
        self.goto(0, 270)
        self.hideturtle()
        self.update_scoreboard()

    def update_scoreboard(self):
        self.clear()
        self.write(f"Score: {self.score} High Score: {self.high_score}", align=ALIGNMENT, font=FONT)
    def reset(self):
        if self.score > self.high_score:
            self.high_score = self.score
            with open("data.txt", mode="w") as data:
                data.write(f"{self.high_score}")
        self.score = 0
        self.update_scoreboard()
    def increase_score(self):
        self.score += 1
        self.update_scoreboard()
```

Figure 14: Scoreboard Class

main.py | snake.py | food.py | scoreBoard.py | data.txt

```python
from turtle import Screen
from snake import Snake
from food import Food
from scoreBoard import ScoreBoard
import time

screen = Screen()
screen.setup(width=600, height=600)
screen.bgcolor("black")
screen.title("Snake Game")
screen.tracer(0)

snake = Snake()
food = Food()
scoreboard = ScoreBoard()

screen.listen()
screen.onkey(snake.up, "Up")
screen.onkey(snake.down, "Down")
screen.onkey(snake.left, "Left")
screen.onkey(snake.right, "Right")

game_is_on = True
while game_is_on:
    screen.update()
    time.sleep(0.1)
    snake.move()

    # Detect collision with food
    if snake.head.distance(food) < 15:  # As the food size is 10x10, we added a bit of buffer and check it with 15.
        food.refresh()
        snake.extend()
        scoreboard.increase_score()

    # Detect collision with wall.
    if snake.head.xcor() > 290 or snake.head.xcor() < -290 or snake.head.ycor() > 290 or snake.head.ycor() < -290:
        scoreboard.reset()
        snake.reset()

    # Detect collision with tail. (if head collides with any segment in the tail, trigger game_over())
    for segment in snake.segments[1:]:
        if snake.head.distance(segment) < 10:
            scoreboard.reset()
            snake.reset()

screen.exitonclick()
```

Figure 15: Snake Game Main Code

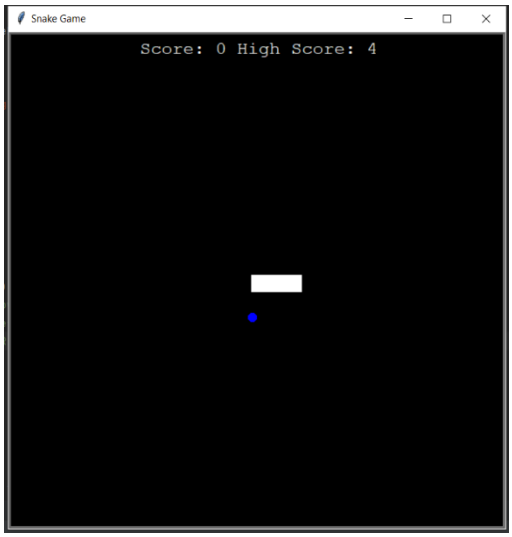


Here, the game starts when the snake is at the middle of the frame and moves to the right in default settings, so the user needs to direction the snake using the arrows in the keyboard. Food is represented by the blue dot so the user needs to try collecting as much food as possible to make the snake grow without touching the edges.

Figure 16: The beginning of the Snake Game

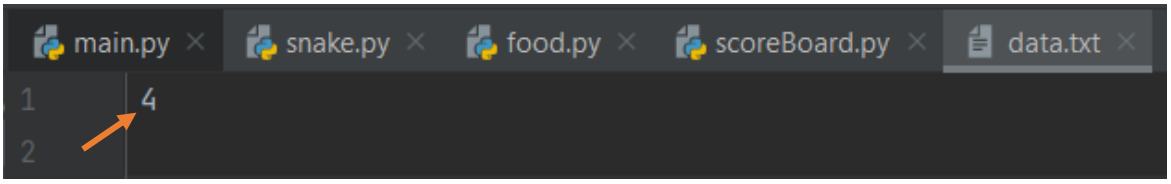
The score that the user made is saved as their score and the high score is still 0 as it will change according to their score in the next levels (the details are explained in the next figure).

Figure 17: User's Score Shown



So when the user touches the snake at one of the edges of the frame, then the score will be 0 and high score will be the score they just made before touching the edge, and that score will be saved in the data.txt file as their score. It means that the user is not game over but just in the next level. The snake will be back to its starting size when the next level is started. If the user scores less than the previous score where is kept in High Score, then that value will not be changed but if it is higher, then the value will be replaced with the new high score.

Figure 18: Highest Score Saved



The score is saved to the data.txt file successfully.

Figure 19: The Content of the data.txt File

5.1.5 Pong Game

In this project, I have built a game for 2-players. When one of the players can play the game with the arrows in the keyboard, the other user should use the letters “w” and “s” for up and down movement so that the user can direction the paddle, the left paddle to be more specific. The users should touch the ball by moving the paddles to score. Different scores will be saved for both left and right paddles. The ball will bounce back if it touches to up or down edge of the board but the user will lose that set if the paddle misses the ball on the left and right edges. The classes and the main code are shown below with the figures.

main.py | paddle.py | ball.py | scoreboard.py

```
from turtle import Turtle


class Paddle(Turtle):

    def __init__(self, position):
        super().__init__()
        self.shape("square")
        self.color("white")
        self.shapesize(stretch_wid=5, stretch_len=1)
        self.penup()
        self.goto(position)

    def go_up(self):
        new_y = self.ycor() + 20
        self.goto(self.xcor(), new_y)

    def go_down(self):
        new_y = self.ycor() - 20
        self.goto(self.xcor(), new_y)

```

Figure 20: Paddle Class

main.py | paddle.py | ball.py | scoreboard.py

```python
from turtle import Turtle

class Ball(Turtle):
    def __init__(self):
        super().__init__()
        self.color("white")
        self.shape("circle")
        self.penup()
        self.x_move = 10
        self.y_move = 10
        self.move_speed = 0.1
    def move(self):
        new_x = self.xcor() + self.x_move
        new_y = self.ycor() + self.y_move
        self.goto(new_x, new_y)
    def bounce_y(self):
        self.y_move *= -1
        self.move_speed *= 0.9
    def bounce_x(self):
        self.x_move *= -1
        self.move_speed *= 0.9
    def reset_position(self):
        self.goto(0, 0)
        self.move_speed = 0.1
        self.bounce_x()
```

Figure 21: Ball Class

main.py | paddle.py | ball.py | scoreboard.py

```python
from turtle import Turtle

class ScoreBoard(Turtle):

    def __init__(self):
        super().__init__()
        self.color("white")
        self.penup()
        self.hideturtle()
        self.l_score = 0
        self.r_score = 0
        self.update_scoreboard()

    def update_scoreboard(self):
        self.clear()
        self.goto(-100, 200)
        self.write(self.l_score, align="center", font=("Courier", 60, "normal"))
        self.goto(100, 200)
        self.write(self.r_score, align="center", font=("Courier", 60, "normal"))

    def l_point(self):
        self.l_score += 1
        self.update_scoreboard()

    def r_point(self):
        self.r_score += 1
        self.update_scoreboard()
```

Figure 22: Scoreboard Class

main.py | paddle.py | ball.py | scoreboard.py

```python
from turtle import Screen
from paddle import Paddle
from ball import Ball
from scoreboard import ScoreBoard
import time

screen = Screen()
screen.bgcolor("black")
screen.setup(width=800, height=600)
screen.title("Pong Game")
screen.tracer(0)

r_paddle = Paddle((350, 0))
l_paddle = Paddle((-350, 0))
ball = Ball()
scoreboard = ScoreBoard()

screen.listen()
screen.onkey(r_paddle.go_up, "Up")
screen.onkey(r_paddle.go_down, "Down")
screen.onkey(l_paddle.go_up, "w")
screen.onkey(l_paddle.go_down, "s")

game_is_on = True
while game_is_on:
    time.sleep(ball.move_speed)
    screen.update()
    ball.move()

    # Detect collision with wall.
    if ball.ycor() > 280 or ball.ycor() < -280:
        # Needs to bounce
        ball.bounce_y()

    # Detect collision with r_paddle
    if ball.distance(r_paddle) < 50 and ball.xcor() > 320 or ball.distance(l_paddle) < 50 and ball.xcor() < -320:
        ball.bounce_x()

    # Detect R paddle misses
    if ball.xcor() > 380:
        ball.reset_position()
        scoreboard.l_point()

    # Detect L paddle misses
    if ball.xcor() < -380:
        ball.reset_position()
        scoreboard.r_point()

screen.exitonclick()
```

Figure 23: Pong Game Main Code

5.1.6 Turtle Crossing Game

In this game, there will be a turtle trying to cross the road with randomly generated number of cars which are represented with horizontal paddles. The colors of the paddles will be given but the number of them will be randomly generated at any time. The user will try to move the turtle to the other end of the road by moving it with the arrows in the keyboard. If the turtle touches any of the cars, then the user will be game over, if not then they will jump to the next level. In each new level as the user proceeds, the speed of the cars will be increased so it will be harder to cross the road. Please also note that once the turtle moves forwards, it cannot move backwards again, so it can only move forwards not backwards. The content of all the classes are shown below with the figures.

main.py | car_manager.py | scoreboard.py | player.py

```python
from turtle import Turtle
import random

COLORS = ["red", "orange", "yellow", "green", "blue", "purple"]
STARTING_MOVE_DISTANCE = 5
MOVE_INCREMENT = 10

class CarManager:
    def __init__(self):
        self.all_cars = []
        self.car_speed = STARTING_MOVE_DISTANCE
    def create_car(self):
        random_chance = random.randint(1, 6)
        if random_chance == 1:
            new_car = Turtle("square")
            new_car.shapesize(stretch_wid=1, stretch_len=2)
            new_car.penup()
            new_car.color(random.choice(COLORS))
            random_y = random.randint(-250, 250)
            new_car.goto(300, random_y)
            self.all_cars.append(new_car)
    def move_cars(self):
        for car in self.all_cars:
            car.backward(self.car_speed)
    def level_up(self):
        self.car_speed += MOVE_INCREMENT
```

Figure 24: The Car Class

main.py | car_manager.py | scoreboard.py | player.py

```python
from turtle import Turtle

FONT = ("Courier", 20, "normal")


class Scoreboard(Turtle):

    def __init__(self):
        super().__init__()
        self.level = 1
        self.hideturtle()
        self.penup()
        self.goto(-280, 270)
        self.update_scoreboard()

    def update_scoreboard(self):
        self.clear()
        self.write(f"Level: {self.level}", align="left", font=FONT)

    def increase_level(self):
        self.level += 1
        self.update_scoreboard()

    def game_over(self):
        self.goto(0, 0)
        self.write(f"GAME OVER", align="center", font=FONT)
```

Figure 25: The Scoreboard Class

main.py | car_manager.py | scoreboard.py | player.py

```python
from turtle import Turtle

STARTING_POSITION = (0, -280)
MOVE_DISTANCE = 10
FINISH_LINE_Y = 280

# Player class will be the turtle we're using.
class Player(Turtle):

    def __init__(self):
        super().__init__()
        self.shape("turtle")
        self.penup()
        self.go_to_start()
        self.setheading(90)

    def go_up(self):
        self.forward(MOVE_DISTANCE)

    def go_to_start(self):
        self.goto(STARTING_POSITION)

    def is_at_finish_line(self):
        if self.ycor() > FINISH_LINE_Y:
            return True
        else:
            return False
```

Figure 26: The Player Class

main.py | car_manager.py | scoreboard.py | player.py

```python
import time
from turtle import Screen
from player import Player
from car_manager import CarManager
from scoreboard import Scoreboard

screen = Screen()
screen.setup(width=600, height=600)
screen.tracer(0)

player = Player()
car_manager = CarManager()
scoreboard = Scoreboard()


screen.listen()
screen.onkey(player.go_up, "Up")

game_is_on = True
while game_is_on:
    time.sleep(0.1)
    screen.update()
    car_manager.create_car()
    car_manager.move_cars()

    # Detect collision with car
    for car in car_manager.all_cars:
        if car.distance(player) < 20:
            game_is_on = False
            scoreboard.game_over()

    # Detect successful crossing
    if player.is_at_finish_line():
        player.go_to_start()
        car_manager.level_up()
        scoreboard.increase_level()


screen.exitonclick()
```

Figure 27: The Turtle Crossing Game Main Code

5.1.7 US State Game

In this game, the user tries to guess the state names from the map given and the map will be separated to its states with proper borders. The aim is to enter as many correct state names as possible. The names of the states, 50 states, will be taken from a csv file and to make the game more educational, a new file with the names of the states that are not entered or entered wrong until the end of the game will be created to check them later.

main.py | 50_states.csv | states_to_learn.csv | blank_states_img.gif

```
Arizona,-203,-40
Arkansas,57,-53
California,-297,13
Colorado,-112,20
Connecticut,297,96
Delaware,275,42
Florida,220,-145
Georgia,182,-75
Hawaii,-317,-143
Idaho,-216,122
Illinois,95,37
Indiana,133,39
Iowa,38,65
Kansas,-17,5
Kentucky,149,1
Louisiana,59,-114
Maine,319,164
Maryland,288,27
Massachusetts,312,112
Michigan,148,101
Minnesota,23,135
Mississippi,94,-78
Missouri,49,6
Montana,-141,150
Nebraska,-61,66
Nevada,-257,56
New Hampshire,302,127
New Jersey,282,65
New Mexico,-128,-43
New York,236,104
North Carolina,239,-22
North Dakota,-44,158
Ohio,176,52
Oklahoma,-8,-41
Oregon,-278,138
Pennsylvania,238,72
Rhode Island,318,94
South Carolina,218,-51
South Dakota,-44,109
Tennessee,131,-34
Texas,-38,-106
Utah,-189,34
Vermont,282,154
Virginia,234,12
Washington,-257,193
West Virginia,200,20
Wisconsin,83,113
Wyoming,-134,90
```

Figure 28: 50 US States csv File

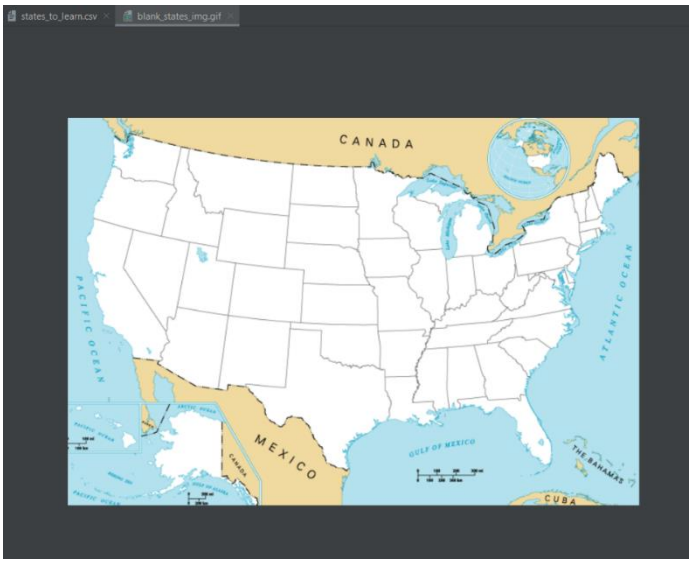
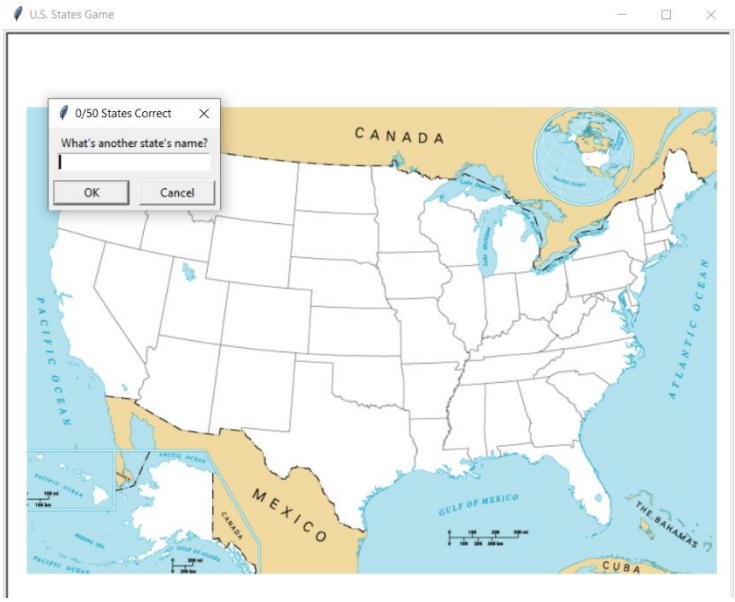


Figure 29: Blank US States Image



The user is expected to enter states names after the program executes. If the states name is spelled correctly, then the number of states will increase. If they enter a wrong state name, then it will not be counted, and a file named 'states to learn' will be created and the name of the state will be added to that file.

Figure 30: US States Game Interface when First Executes

The states that are entered correctly will be counted and the score will increase. The state will also will be shown in the map at its correct position. If the state name entered is wrong, then it'll not be counted.

Figure 31: The Output after Entering State Names

main.py states_to_learn.csv

```
import turtle
import pandas

screen = turtle.Screen()
screen.title("U.S. States Game")
image = "blank_states_img.gif"
turtle.addshape(image)
turtle.shape(image)

data = pandas.read_csv("50_states.csv")
all_states = data.state.to_list()
guessed_states = []

while len(guessed_states) < 50:
    answer_state = screen.textinput(title=f"{len(guessed_states)}/50 States Correct",
                                    prompt="What's another state's name?").title()
                                          # title() will make the 1st letter uppercase

    # For the if statement below, use list comprehension. It's given as comment lines where the if statement ends.
    #if answer_state == "Exit":
    #    missing_states = []
    #    for state in all_states:
    #        if state not in guessed_states:
    #            missing_states.append(state)
    #    new_data = pandas.DataFrame(missing_states)
    #    new_data.to_csv("states_to_learn.csv")
    #    break

    if answer_state == "Exit":
        missing_states = [state for state in all_states if state not in guessed_states]
        new_data = pandas.DataFrame(missing_states)
        new_data.to_csv("states_to_learn.csv")
        break

    if answer_state in all_states:
        guessed_states.append(answer_state)
        t = turtle.Turtle()
        t.hideturtle()
        t.penup()
        state_data = data[data.state == answer_state]
        t.goto(int(state_data.x), int(state_data.y))
        t.write(state_data.state.item())  # OR use this line -> t.write(answer_state)
```

Figure 32: US States Game Main Code

5.1.8 Pomodoro Timer

In this project, a pomodoro timer is created. With the tomato image at the background, this timer will start counting down from 25 minutes when the user pushed ‘start’ button and resets the timer when pushing the ‘reset’ button. 5 minutes break will be given after 35 minutes of working, so after the countdown is completed from 25 minutes, countdown from 5 minutes for the break will automatically start. The user can pause the timing if they want by pushing the ‘start’ button again. Not to confuse the user, ‘Break’ or ‘Work’ messages will be shown at the screen during the corresponding activity. The code and the screenshots for input/output are shown below with the figures.

main.py tomato.png

```python
from tkinter import *
import math

# ---------------------------- CONSTANTS ------------------------------- #
PINK = "#e2979c"
RED = "#e7305b"
GREEN = "#9bdeac"
YELLOW = "#f7f5dd"
FONT_NAME = "Courier"
WORK_MIN = 25
SHORT_BREAK_MIN = 5
LONG_BREAK_MIN = 20
reps = 0
timer = None

# ---------------------------- TIMER RESET ------------------------------- #


def reset_timer():
    window.after_cancel(timer)
    canvas.itemconfig(timer_text, text="00:00")
    title_label.config(text="Timer")
    check_marks.config(text="")
    global reps
    reps = 0
```

Figure 33: Constants and Timer Reset

```python
# ---------------------------- TIMER MECHANISM ------------------------------- #


def start_timer():
    global reps
    reps += 1
    work_sec = WORK_MIN * 60
    short_break_sec = SHORT_BREAK_MIN * 60
    long_break_sec = LONG_BREAK_MIN * 60

    if reps % 8 == 0:
        count_down(long_break_sec)
        title_label.config(text="Break", fg=RED)
    elif reps % 2 == 0:
        count_down(short_break_sec)
        title_label.config(text="Break", fg=PINK)
    else:
        count_down(work_sec)
        title_label.config(text="Work", fg=GREEN)
```

Figure 34: Timer Mechanism

```python
# ---------------------------- COUNTDOWN MECHANISM ------------------------------- #


def count_down(count):
    count_min = math.floor(count / 60)
    count_sec = count % 60
    if count_sec < 10:
        count_sec = f"0{count_sec}"
    if count_min < 10:
        count_min = f"0{count_min}"
    canvas.itemconfig(timer_text, text=f"{count_min}:{count_sec}")
    if count > 0:
        global timer
        timer = window.after(1000, count_down, count-1)
    else:
        start_timer()
        marks = ""
        work_sessions = math.floor(reps/2)
        for _ in range(work_sessions):
            marks += "✓"
        check_marks.config(text=marks)
```

Figure 35: Countdown Mechanism

```python
# ---------------------------- UI SETUP ------------------------------- #
window = Tk()
window.title("Pomodoro")
window.config(padx=100, pady=50, bg=YELLOW)

title_label = Label(text="Timer", fg=GREEN, bg=YELLOW, font=(FONT_NAME, 40))
title_label.grid(column=1, row=0)

canvas = Canvas(width=200, height=224, bg=YELLOW, highlightthickness=0)
tomato_img = PhotoImage(file="tomato.png")
canvas.create_image(100, 112, image=tomato_img)
timer_text = canvas.create_text(100, 130, text="00:00", fill="white", font=(FONT_NAME, 30, "bold"))
canvas.grid(column=1, row=1)

start_button = Button(text="Start", highlightthickness=0, command=start_timer)
start_button.grid(column=0, row=2)

reset_button = Button(text="Reset", highlightthickness=0, command=reset_timer)
reset_button.grid(column=2, row=2)

check_marks = Label(fg=GREEN, bg=YELLOW, font=(FONT_NAME, 10, "bold"))
check_marks.grid(column=1, row=3)


window.mainloop()
```

Figure 36: User Interface Setup Code

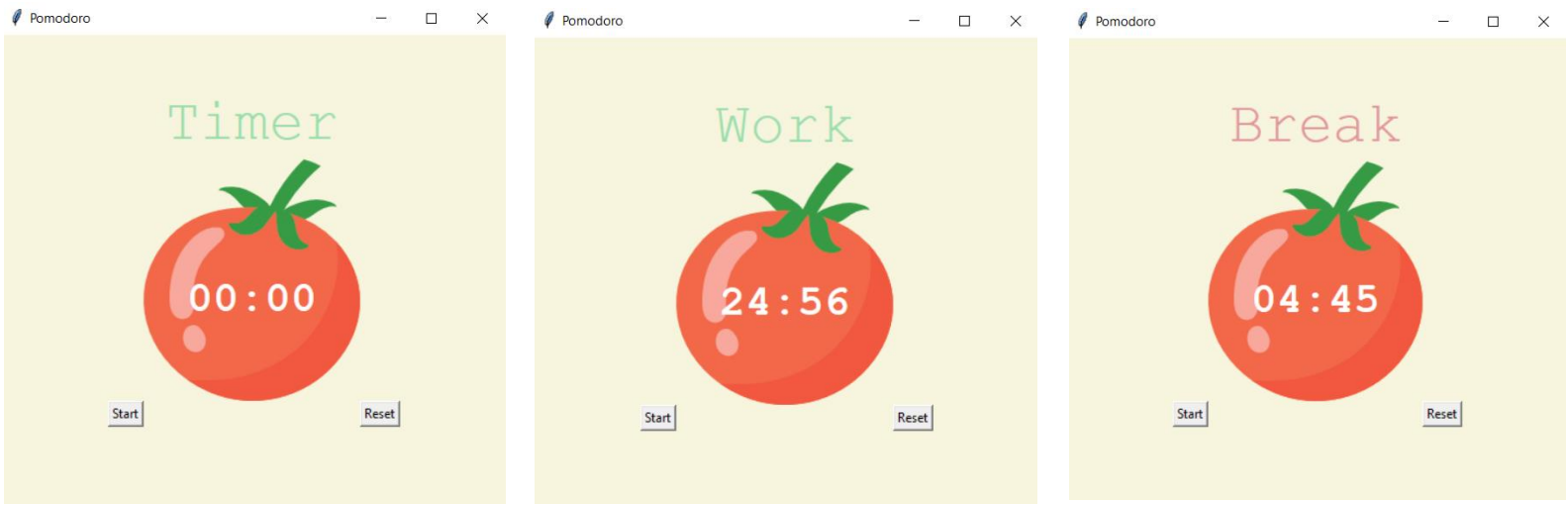


Figure 37: First Execution Output – Work Time Countdown – Break Time Countdown

5.1.9 Flash Card Project

In this project called Flashy, flash cards will be used for French vocabulary practice. There will be cards showing the French words and their back will be switched after some certain time (it can be changed as I assigned a random value I chose) for their English version. If the user does not want to see the English representation, they can click one of the cross or check mark shaped buttons. If the user knows the word, then they need to click the check mark button. If they do not, then they need to click the cross button, and a file named ‘words to learn’ will be created (if not already exists) and the word will be added to that file. The code and other details are shown with the figures below.

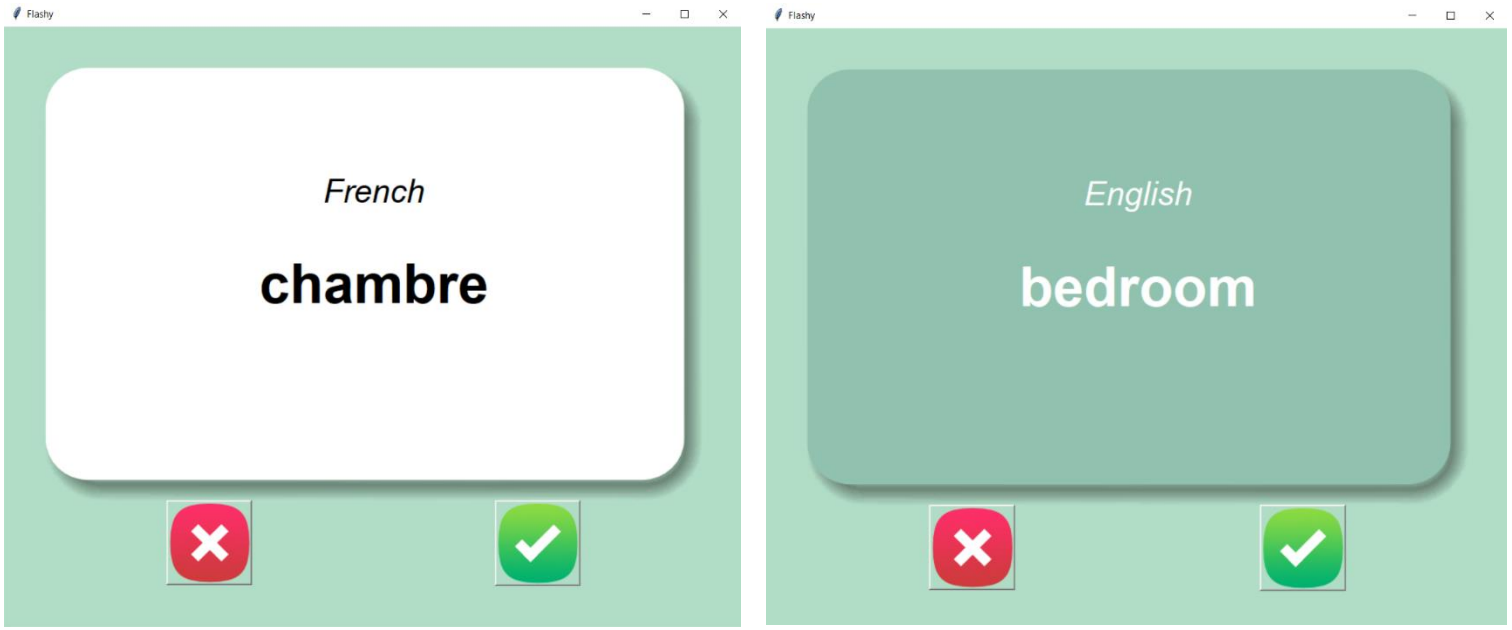


Figure 38: Front and Back Side of an Example Word Card

main.py

```python
from tkinter import *
import pandas
import random

BACKGROUND_COLOR = "#B1DDC6"
current_card = {}
to_learn = {}

try:
    data = pandas.read_csv("data/words_to_learn.csv")
except FileNotFoundError:
    original_data = pandas.read_csv("data/french_words.csv")
    to_learn = original_data.to_dict(orient="records")
else:
    to_learn = data.to_dict(orient="records")


def next_card():
    global current_card, flip_timer
    window.after_cancel(flip_timer)
    current_card = random.choice(to_learn)
    canvas.itemconfig(card_title, text="French", fill="black")
    canvas.itemconfig(card_word, text=current_card["French"], fill="black")
    canvas.itemconfig(card_background, image=card_front_img)
    flip_timer = window.after(3000, func=flip_card)

def flip_card():
    canvas.itemconfig(card_title, text="English", fill="white")
    canvas.itemconfig(card_word, text=current_card["English"], fill="white")
    canvas.itemconfig(card_background, image=card_back_img)
def is_known():
    to_learn.remove(current_card)
    data = pandas.DataFrame(to_learn)
    data.to_csv("data/words_to_learn.csv", index=False)
    next_card()
window = Tk()
window.title("Flashy")
window.config(padx=50, pady=50, bg=BACKGROUND_COLOR)

flip_timer = window.after(3000, func=flip_card)

canvas = Canvas(width=800, height=526)
card_front_img = PhotoImage(file="images/card_front.png")
card_back_img = PhotoImage(file="images/card_back.png")
card_background = canvas.create_image(400, 263, image=card_front_img)
card_title = canvas.create_text(400, 150, text="", font=("Ariel", 30, "italic"))
card_word = canvas.create_text(400, 263, text="", font=("Ariel", 50, "bold"))

canvas.config(bg=BACKGROUND_COLOR, highlightthickness=0)
canvas.grid(row=0, column=0, columnspan=2)

cross_image = PhotoImage(file="images/wrong.png")
unknown_button = Button(image=cross_image, highlightthickness=0, command=next_card)
unknown_button.grid(row=1, column=0)

check_image = PhotoImage(file="images/right.png")
known_button = Button(image=check_image, highlightthickness=0, command=is_known)
known_button.grid(row=1, column=1)

next_card()
window.mainloop()
```

Figure 39: Flash Card Game Main Code

5.1.10 Personal Website

In this project, I created a personal website that has some particular information about me. It is a very basic website using both HTML and very basic CSS to style the website a bit more. The details of the website is shown below with the figures.

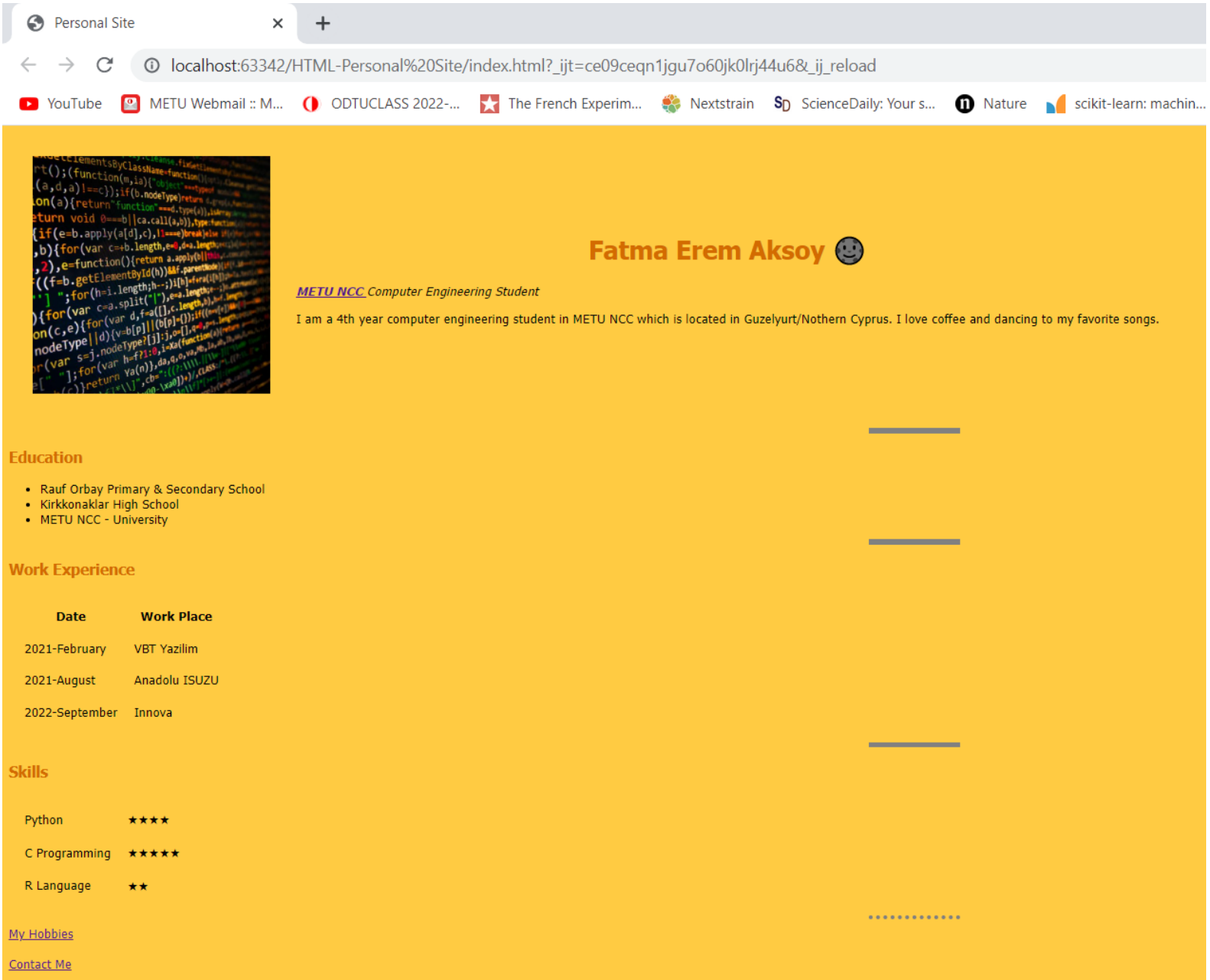


Figure 40: Personal Website Interface

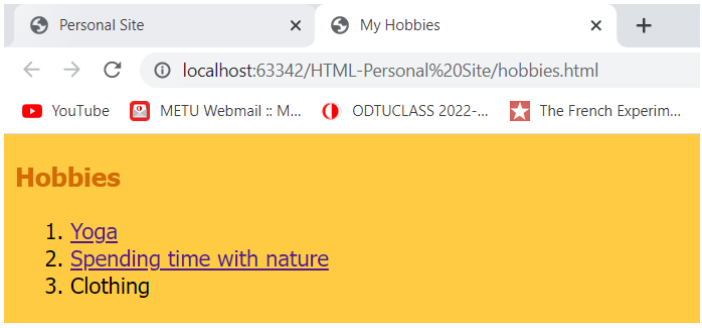


Figure 41: My Hobbies Section

After clicking 'My Hobbies' section in the main website, new window will be open to list my hobbies. My favorite yoga and nature sound videos on YouTube will be opened in a new window if the user clicks on them.

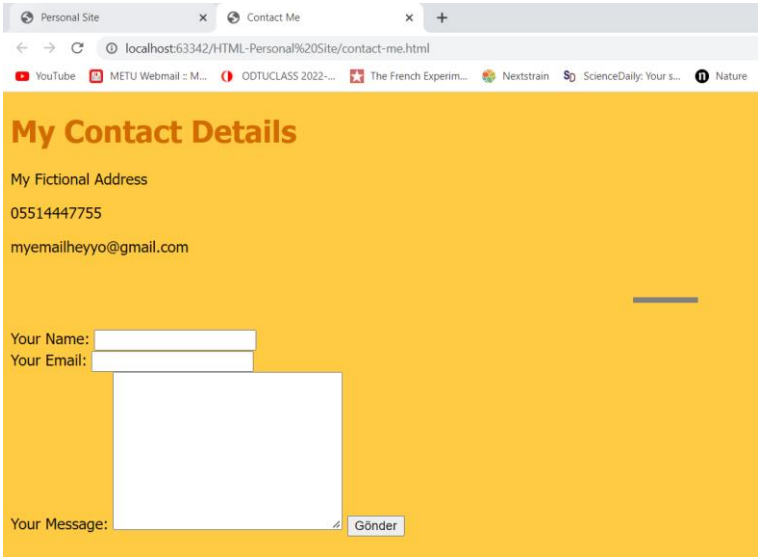


Figure 42: Contact Me Section

Random information is given for the contact details but this is how the Contact Me section looks. Users can send an email with the message that they want to give as well.

index.html | coding.png | styles.css | contact-me.html | hobbies.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My Hobbies</title>
    <link rel="stylesheet" href="CSS/styles.css">
</head>
<body>
  <h3> Hobbies </h3>
    <ol>
        <li>  <a href="https://www.youtube.com/c/yogawithkassandra"> Yoga </a> </li>
        <li> <a href="https://www.youtube.com/results?search_query=nature+sounds+relaxing+music"> Spending time with nature </a>  </li>
        <li> Clothing </li>
    </ol>
     <!-- ol stands for an ordered list, and it's lists the elements with numbers instead of bullet points. -->
</body>
</html>
```

Figure 43: Hobbies Section Code

index.html | coding.png | styles.css | contact-me.html | hobbies.html

```
<!DOCTYPE html>
<html lang="en">
<head>  <meta charset="UTF-8">
        <title>Contact Me</title>
        <link rel="stylesheet" href="CSS/styles.css">  </head>
<body>  <h1> My Contact Details </h1>
        <p> My Fictional Address </p>
        <p> 05514447755 </p>
        <p> myemailheyyo@gmail.com </p>
        <br><hr size="3" noshade=""><br>
        <form action="mailto:fatmaeremaksoy@gmail.com" method="post" enctype="text/plain">
            <label> Your Name: </label>
            <input type="text" name="yourName" value=""><br>
            <!-- <input type="color" name="" value=""><br>
            <label> Password: </label>
            <input type="password" name="" value=""><br>
            <label> Do you want to sign up to the email list? </label>
            <input type="checkbox" name="" value="">
            <input type="file" name="" value=""><br>
            <input type="date" name="" value=""><br>
            <input type="radio" name="" value=""><br>
            <input type="range" name="" value=""> -->
            <label> Your Email: </label>
            <input type="email" name="yourEmail" value=""><br>
            <label> Your Message: </label>
            <textarea name="name" rows="10" cols="30"> </textarea>
            <input type="submit" name="yourMessage"><br> </form>
</body>
</html>
```

Figure 44: Contact Me Section Code

index.html | coding.png | styles.css | contact-me.html | hobbies.html

```
body {
    background-color: #FFCB42;  }

h1 {
    color: #D36B00; }

h3 {
    color: #D36B00; }

hr {
    border-style: none;
    border-top-style: dotted;
    border-color: grey;
    border-width: 5px;
    width: 5%;  }

img {
    height: 300px   }
```

Figure 45: CSS File to Style

index.html | coding.png | styles.css | contact-me.html | hobbies.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title> Personal Site </title>
    <link rel="stylesheet" href="CSS/styles.css">
</head>

<body>
    <!-- This table below is created to show any text beside the picture. If we want to show the text under the picture
        but not beside, then we don't need to add table. We can simply write it as it is.-->
    <table cellspacing="30">
        <tr>
            <td> <img src="coding.png" alt="om symbol"> </td>
            <td> <h1> <center> Fatma Erem Aksoy 🧿 </center> </h1>
                <p><em> <strong> <a href="https://ncc.metu.edu.tr/">METU NCC </a> </strong> Computer Engineering Student </em></p>
                    <!-- Use <a> to highlight the word coming after and to direct the user to the mentioned link via clicking the word.
                        They are called 'anchor tags' and the links added are called hyperlinks.-->
                    <!-- <em> to make the text italic type and <strong> to make it bold.  -->
                <p> I am a 4th year computer engineering student in METU NCC which is located in Guzelyurt/Nothern Cyprus.
                        I love coffee and dancing to my favorite songs.</p>
            </td>
        </tr>
    </table>
    <hr size="4" noshade>
    <h3> Education </h3>
    <ul>
        <li> Rauf Orbay Primary & Secondary School </li>
        <li> Kirkkonaklar High School </li>
        <li> METU NCC - University </li>
    </ul>
    <!-- ul stands for an un-ordered list, and it's a bullet point list. We need to use <li> for each element. -->

    <hr size="4" noshade>
    <h3> Work Experience </h3>
  <!--  <p> 2021-February VBT Yazilim </p>
    <p> 2021-August Anadolu ISUZU </p>
    <p> 2022-September Innova </p>
   -->
    <table cellspacing="20">
        <thead>
            <tr>
                <th> Date </th>
                <th> Work Place </th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td> 2021-February </td>
                <td> VBT Yazilim </td>
            </tr>
            <tr>
                <td> 2021-August </td>
                <td> Anadolu ISUZU </td>
            </tr>
```

```
            <tr>
                <td> 2022-September </td>
                <td> Innova </td>
            </tr>
        </tbody>
    </table>
    <hr size="4" noshade>
    <h3> Skills </h3>
    <table cellspacing="20">
        <tr>
            <td> Python </td>
            <td> ★★★★ </td>
        </tr>
        <tr>
            <td> C Programming </td>
            <td> ★★★★★ </td>
        </tr>
        <tr>
            <td> R Language </td>
            <td> ★★ </td>
        </tr>
    </table>
    <hr>
    <a href="hobbies.html" > My Hobbies </a>
    <br> </br>
    <a href="contact-me.html" > Contact Me </a>
</body>
</html>
```

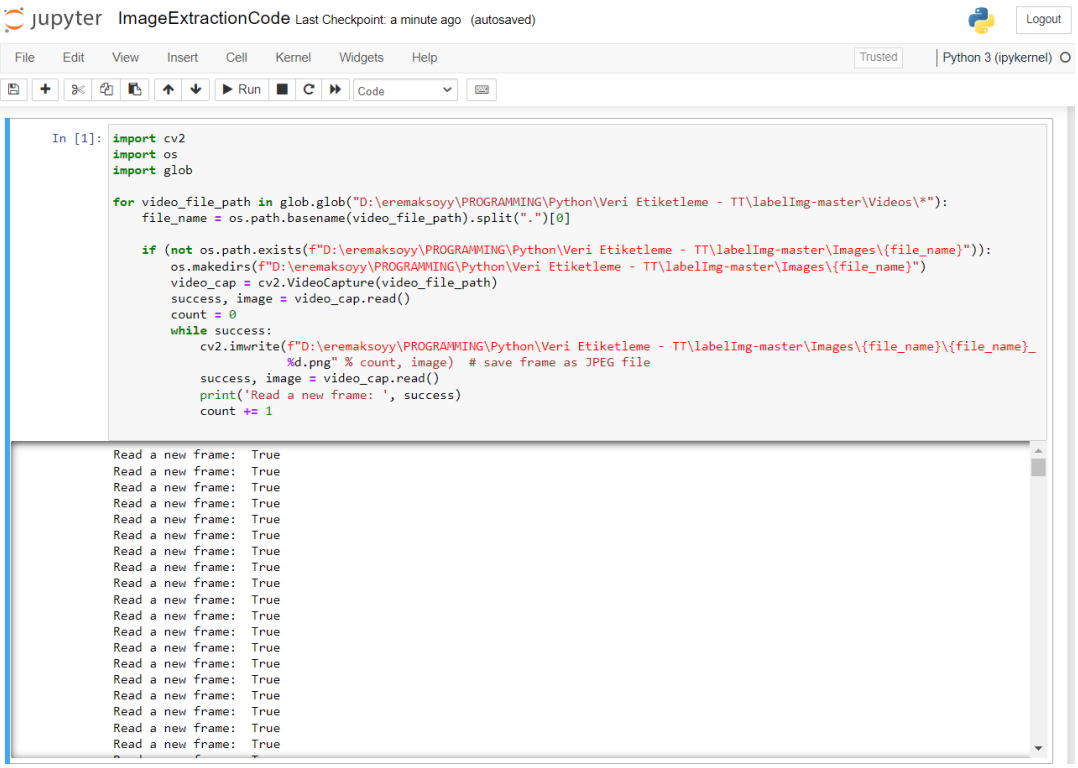
Figure 46: Personal Website Main Code

5.2 Data Labelling for Defect Detection

For this project, I helped with the data labelling for defect detection of Turk Telekom modems. There were videos taken from the modems showing which LEDs are on and off but to detect the defect, a model should have been trained so that the users of those modems would not even need to call someone to fix their problem when something goes wrong. The project aimed to fix the problems in the modems by using an application and when the costumers upload 5-10 seconds video of the modem, the problem could be fixed as the model for that function is already trained with the enough amount of data. So my part in this project was to get the frames from different angles from the videos taken and label them to be trained.

I used Python code on Jupyter Notebook with a few lines of code to get the frames from the videos, and then I extracted the ones that are not clear enough to train or the ones having the same angle. After I am done with it, I used labelImg to label the

frames. The reason I preferred to use labelImg is that it is easy to access from the Anaconda Navigator and its interface is very beginner friendly to use so I did not have any problem with understanding the platform. The code I used and the other details are given below with the support of figures as well.



The code here will extract the images from the videos in the given file path and save them JPEG file.

Figure 47: Image Extraction Code

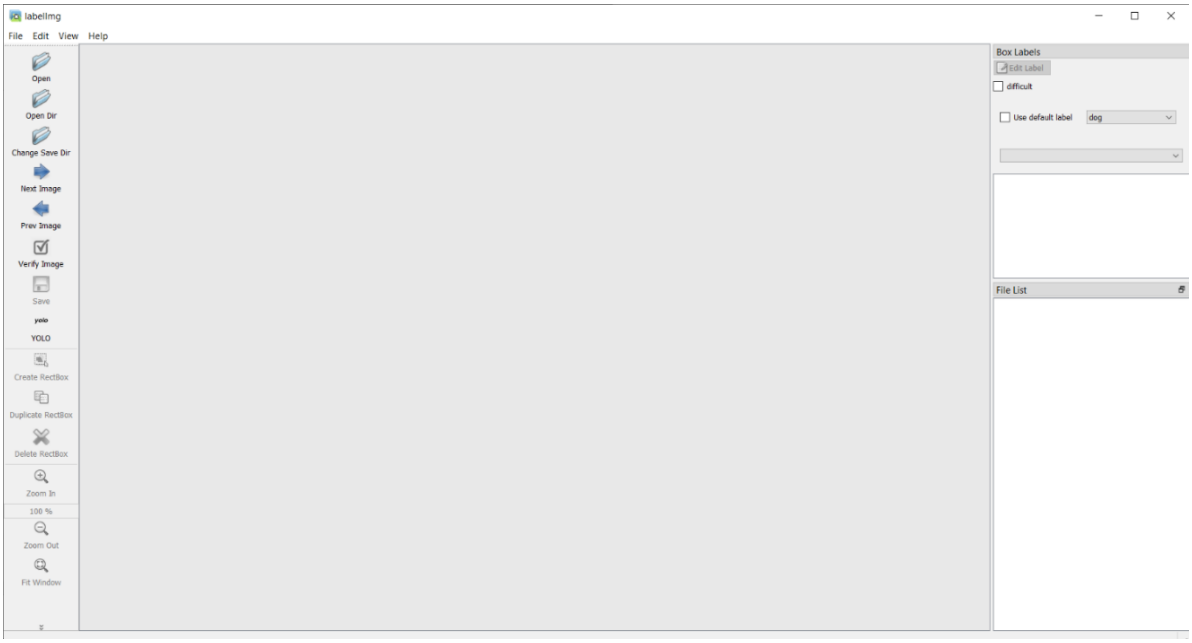
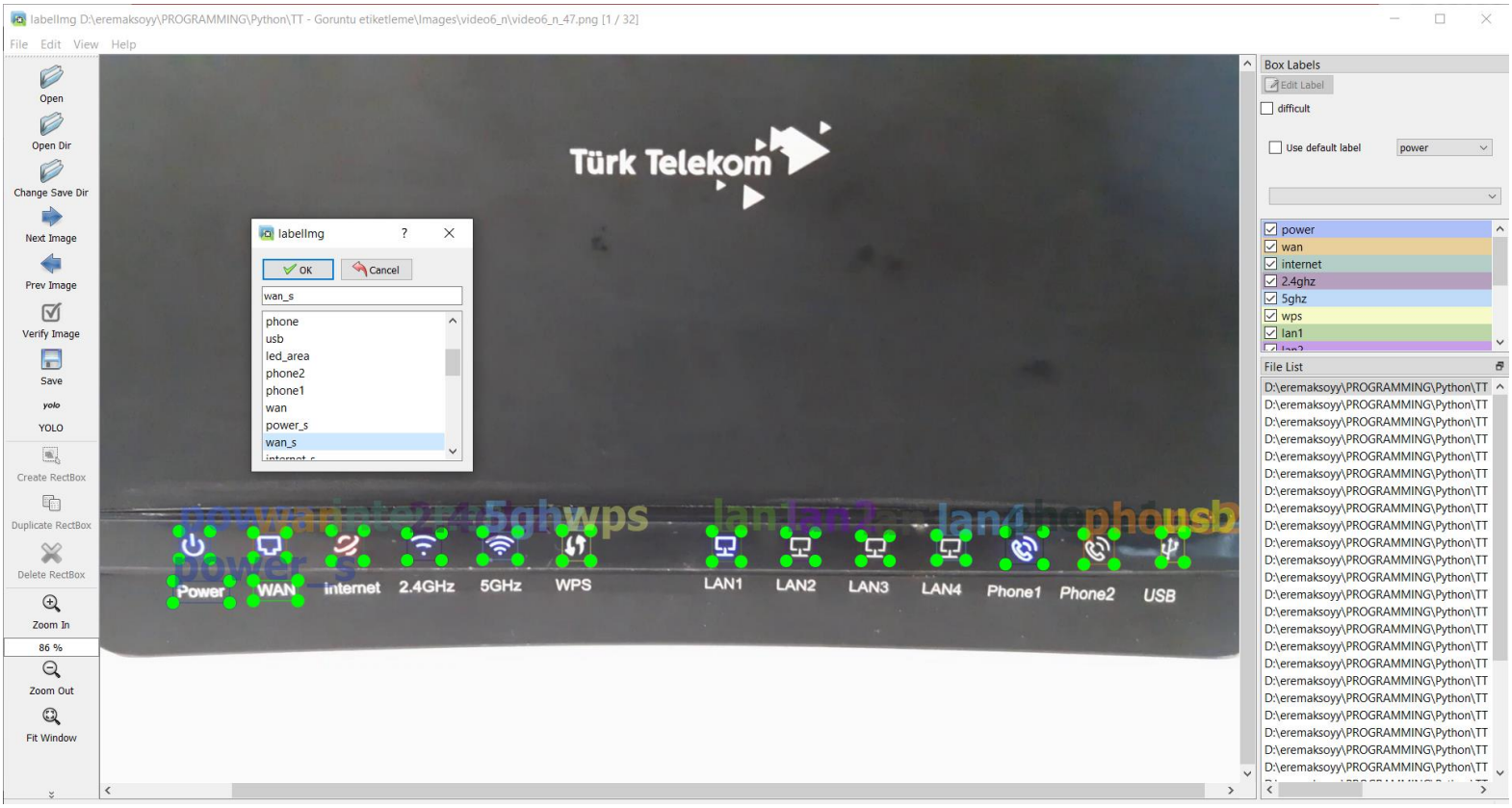


Figure 48: The Interface of labelImg



In the figure above, the LEDs are labeled according to their name, and the names of the labels are predefined in a txt file and then added to the labelImg folder to be able to see that will be the classes we will use. The content of that txt file is arranged by me, and I listed all the labels that will be necessary so it can be changed based on the concept of the images/data we will label. The content of the txt file can be seen below.

classes.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
power
broadband
internet
2.4ghz
5ghz
wps
phone
usb
led_area
phone2
phone1
wan
power_s
wan_s
internet_s
2.4ghz_s
5ghz_s
wps_s
phone1_s
phone2_s
```

(continues):

```
dsl
lan1
lan2
lan3
lan4
ww
5g_wlan
dsl_s
usb_s
lan1_s
lan4_s
lan2_s
lan3_s
inter
```

Figure 49: Example of a Different Modem and its Labels Selected

video16_0.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
29 0.314063 0.185648 0.029167 0.043519
20 0.351823 0.181019 0.043229 0.060185
19 0.399740 0.180556 0.042188 0.053704
17 0.451823 0.183796 0.031771 0.045370
16 0.496094 0.184722 0.031771 0.047222
15 0.537760 0.179167 0.040104 0.056481
14 0.583854 0.176389 0.040625 0.058333
13 0.638542 0.179630 0.031250 0.048148
12 0.682031 0.175926 0.035937 0.051852
7 0.324479 0.258796 0.017708 0.050926
9 0.371354 0.262037 0.019792 0.050000
10 0.414583 0.258333 0.013542 0.044444
5 0.461719 0.253241 0.020313 0.049074
4 0.508073 0.257407 0.024479 0.042593
3 0.552344 0.253704 0.026562 0.040741
2 0.598698 0.253241 0.025521 0.045370
11 0.644271 0.251852 0.027083 0.040741
0 0.692708 0.250463 0.025000 0.041667
```

After labeling the frames extracted, an XML file of the images is automatically created. After each labeling process, the XML file will be created there the location values (x and y locations) of the labels separately. For example, the XML file created for the frame and its labels looks like the figure at the left side.

5.3 Google Dialogflow

The platform I used for this project is Google Dialogflow, which is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your mobile app, web application, device, bot, interactive voice response system, and so on. More detailed information can be found in their website [13].

For the Google Dialogflow project, the task I was given was to create a chatbot that has some specific features and is able to perform some functions. Firstly, I created a basic chatbot to get familiar with the environment and I tried its integrations with both Web Demo and Dialogflow Messenger modes.

After creating a Google Dialogflow account, I created a new agent to get started with the chatbot. I added some intents from the “Intents” section, which represents the possible questions and dialogs between the chatbot and the user. After creating and clicking the intent I created, I added a couple of possible user expressions from the ‘Training phrases’ section. The example to this can be seen with the figure below:

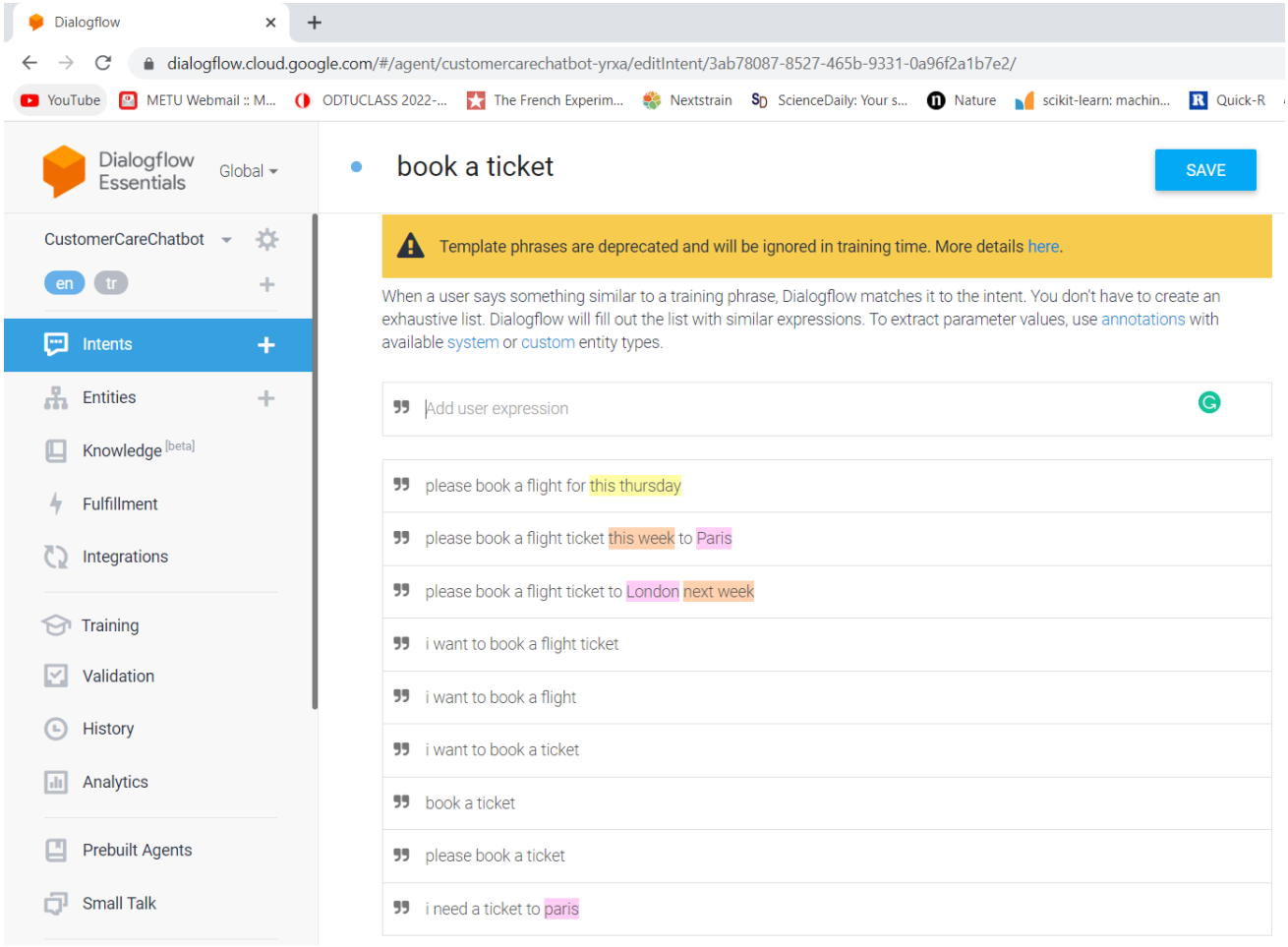
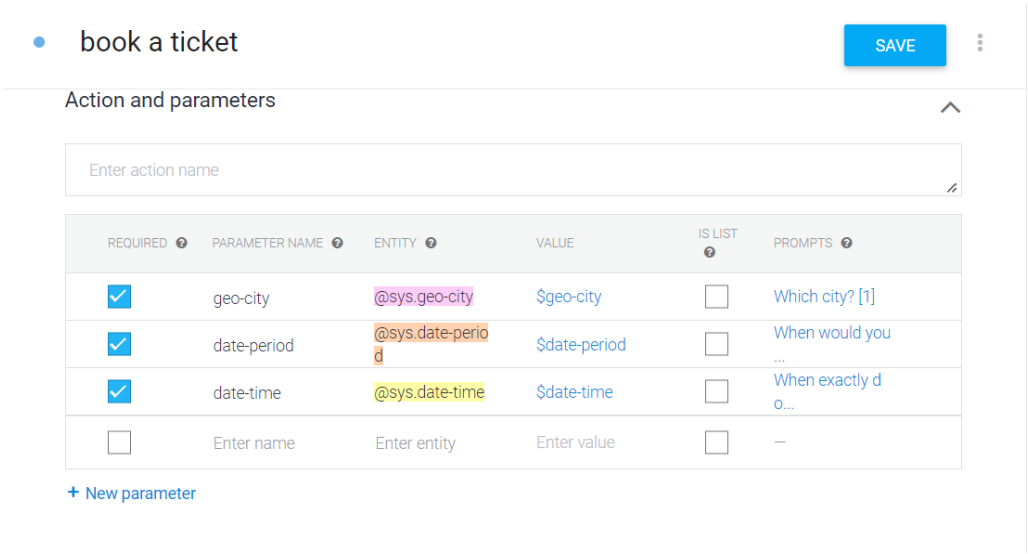


Figure 50: ‘book a ticket’ Intent

This figure shows the ‘book a ticket’ intent and some possible user expressions as the training phrases. I added some expressions for a case that the user wants to book a flight ticket to a certain place and certain time.



For the date-time and location selection of the ticket that the user wants to book, I used the appropriate parameters, and also added prompts in case the user does not include these necessary information to the expression, so that the program will ask with the prompts written there.

Figure 51: Actions and Parameters of 'book a ticket'

I entered some other intents as well like canceling the ticket, and I added follow-up intent for both booking and cancelling the flight ticket. Follow-up intents are basically a child of the parent intent, and after creating the follow-up intents, the output and input context of the follow-up intent is automatically added to the parent intent. The follow-up intents in this case are both "yes" as the user might say "yes" for booking a flight ticket or "yes" for cancelling the ticket. As an example how it looks like in Dialogflow, follow-up intent for "book a ticket" is shown in the figure below:

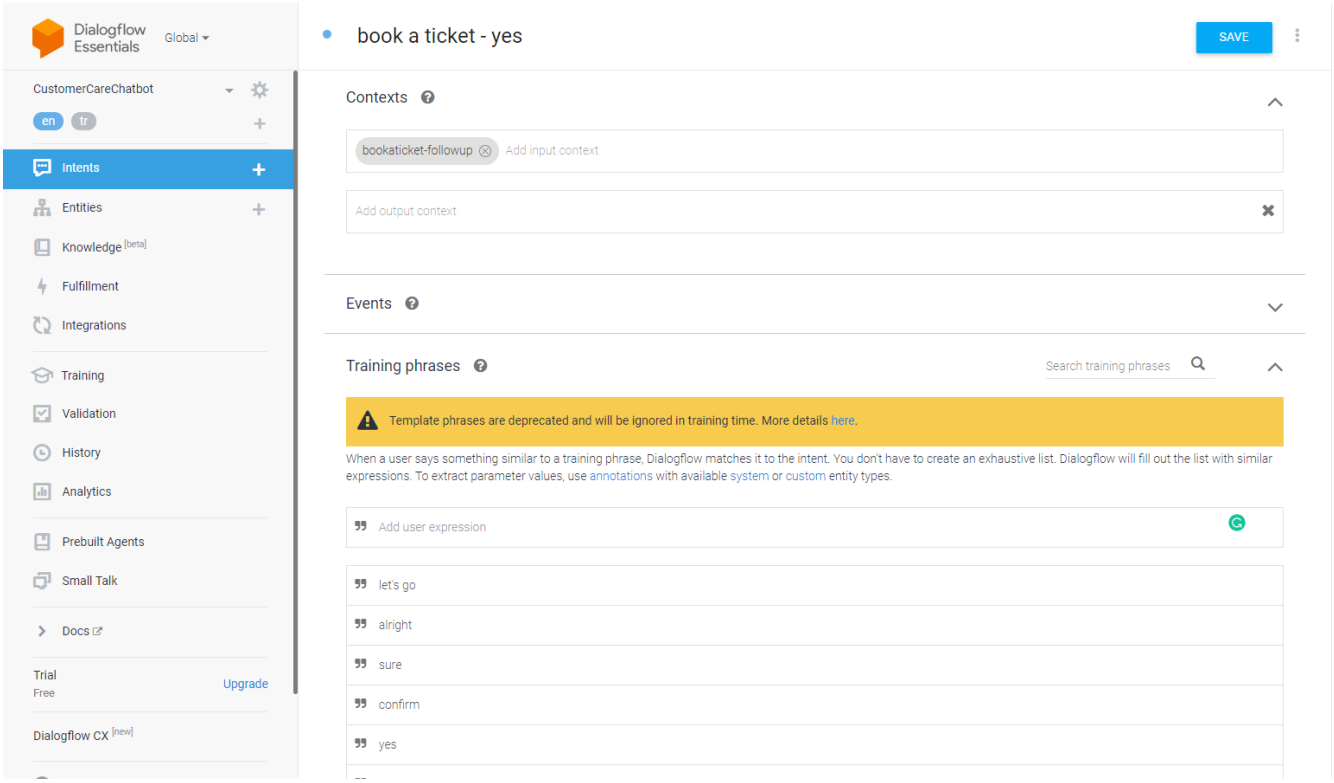
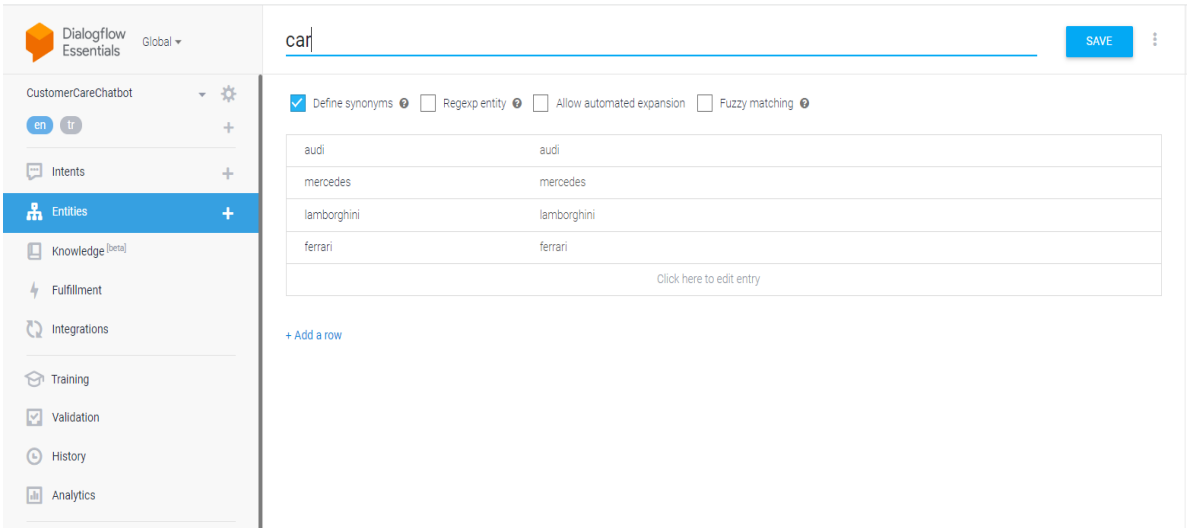


Figure 52: Book a ticket "yes" follow-up Intent

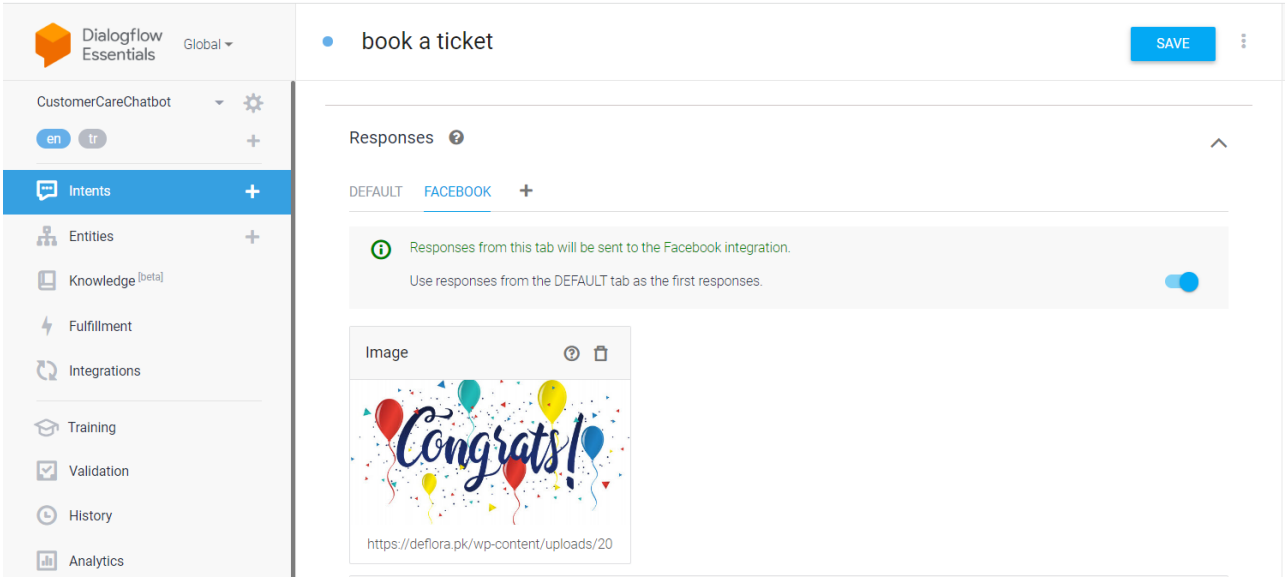
Apart from the entities coming by default, such as date-time, geo-city, date-period, as shown in the previous pages, Google Dialogflow also allows us to define our own entities to use them in the intents created as inputs or responses. I created an entity called ‘car’ to test this feature and the figure below shows how its implementation is:



Example values like Audi, Mercedes, Lamborghini, and Ferrari are given for the ‘car’ entitiy

Figure 53: ‘car’ Entity

Another feature I used adding different types of responses for the Responses section apart from the responses that manually added for the user expression. Custom payload option is one of them and it is used to respond differently based on the user action. I preferred buttons in custom payload option. Other options are also available such as image, card and text replies but they are available based on the application that we connect to such as Slack, Google Chat, Facebook, Telegram. I connected the chatbot I created with Facebook so I will be showing the available options for Facebook. I used image response and custom payload for that and how they are implemented/added is shown in the figure below. Further information about their looking on the actual chatbot will be discussed later.



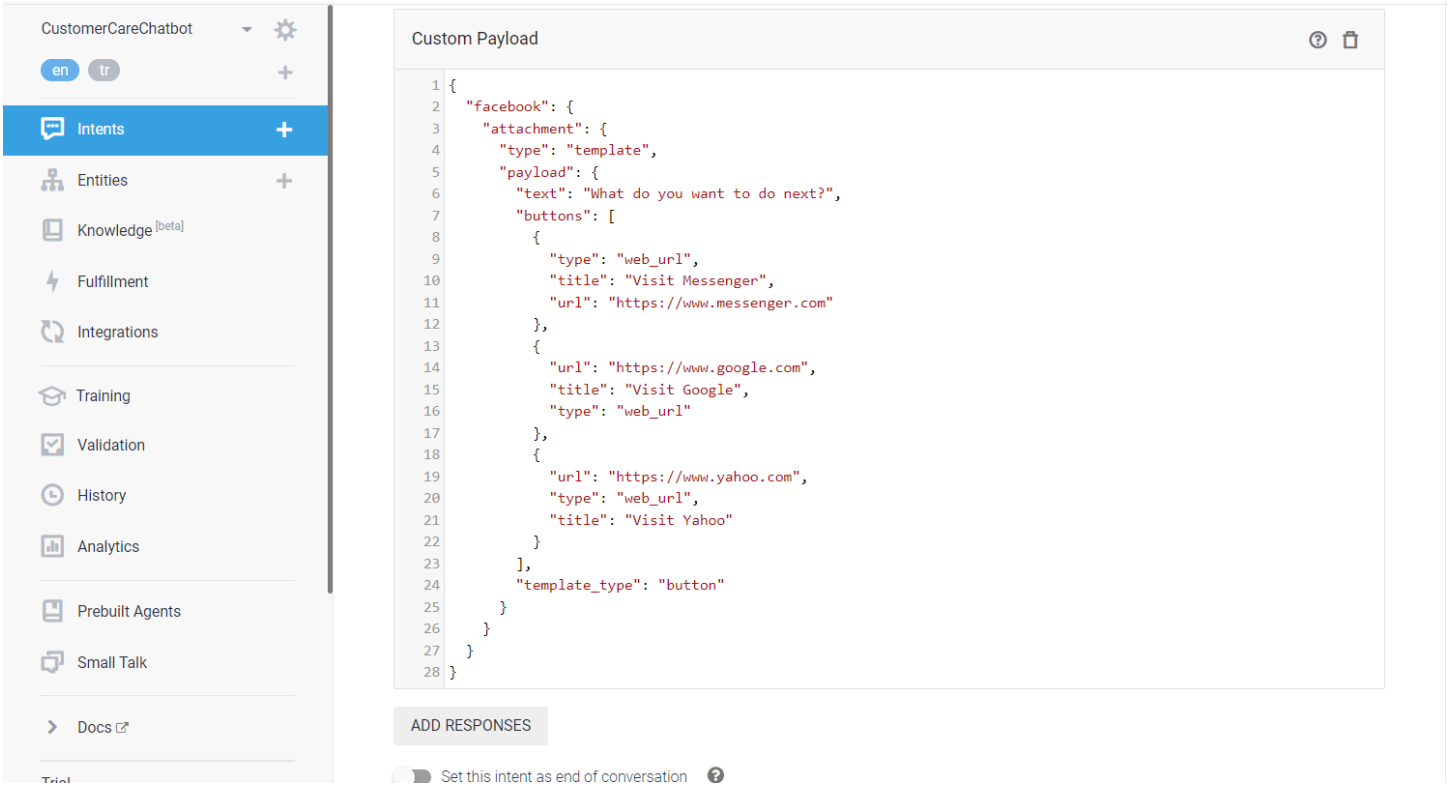


Figure 54: Custom Payload and Image Responses

For the next step, I connected the chatbot I created to a Facebook page that I created to use for this project. The page looks like this:

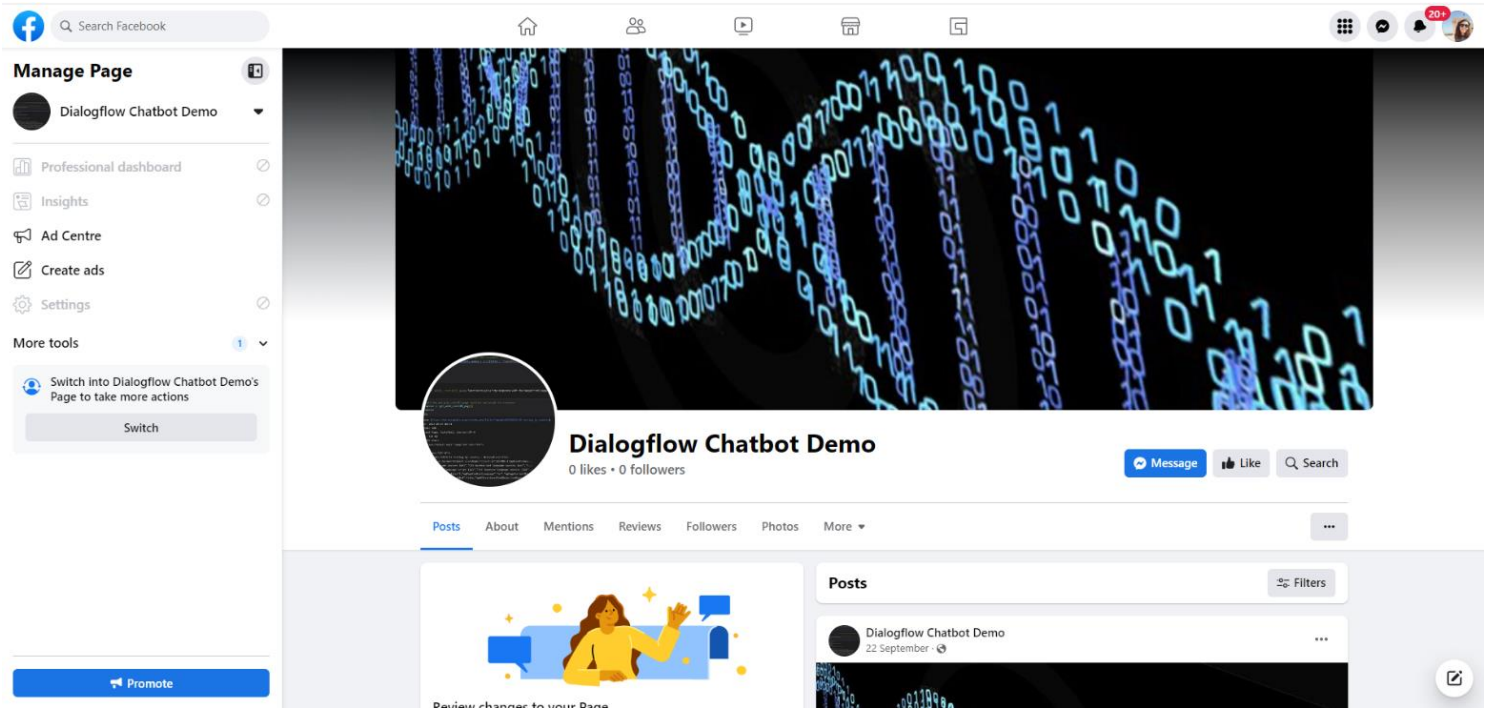


Figure 55: Facebook webpage to Test Chatbot

To connect Dialoglow Messenger and Facebook, I created an account in Mega for Developers and applied the required permission steps. First, I added a project there, which is the Facebook webpage that I created, and then go to the ‘Settings’ section for all the access permission steps. I added the page that created under the ‘Access Tokens’ section to generate a page access token to start using the platform APIs. Then I generated a token to use it under the ‘Webhooks’ section and after adding the page to this section as well, I selected 2 fields from the provided list as the features that I will need, which are messages and messaging_postbacks. Then to complete the connection, I did the necessary arrangements on Dialogflow as well. Under ‘Integrations’ -> ‘Test based’, I selected Messenger for Facebook option to connect them. I copied and pasted the access token that I generated on Mega Developers to ‘Page Access Token’ on the page appeared when I wanted to connect them (the page appeared is shown with figure 57 below). I selected a random password key for the ‘Verify Token’ part to use it on the ‘Verify Token’ section under the ‘Webhooks’ section on Mega for Developers page (this section is also shown with figure 56 below). After these arrangements are done, pages would look like the figures below:

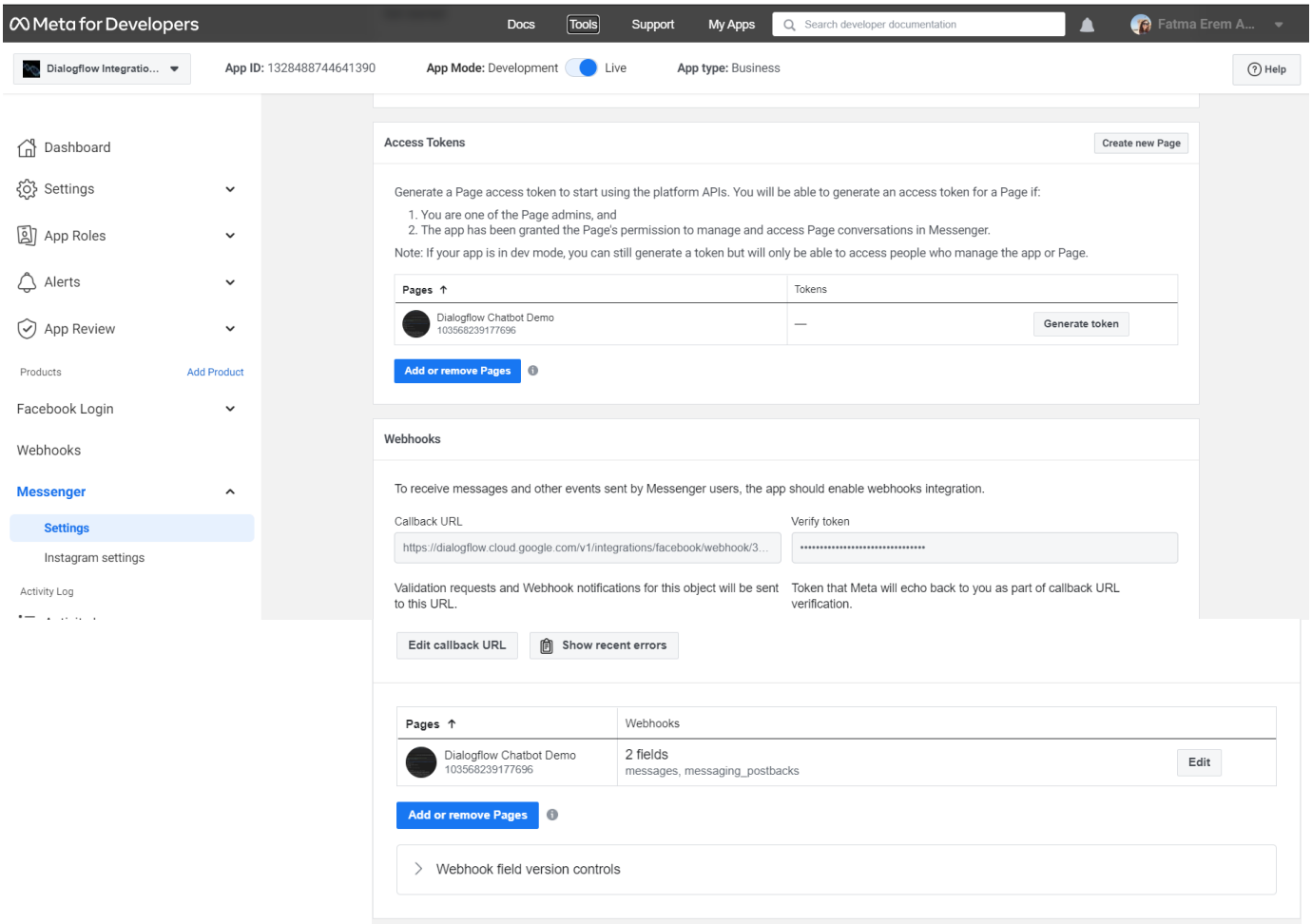


Figure 56: Mega for Developers Settings Page for Facebook Page Access

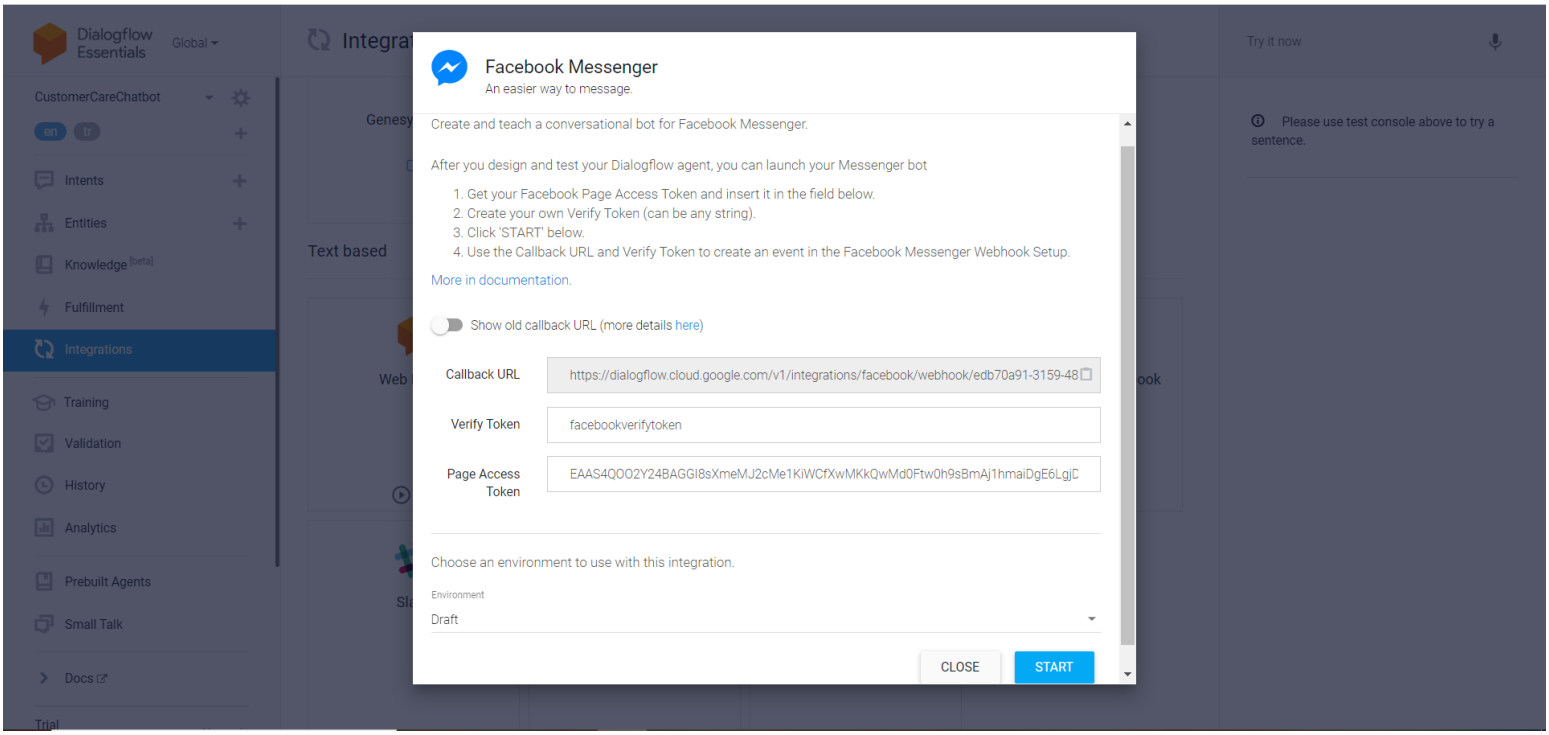
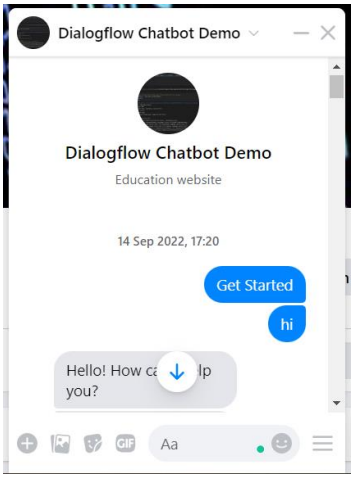


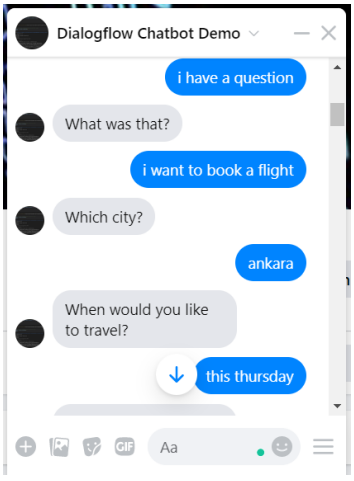
Figure 57: Dialogflow - Facebook Messenger Connection Access Permission Page

After these steps, the connection between Dialogflow and Facebook Messenger is created by clicking the 'Start' button on the page appearing in the figure above, then I opened Facebook webpage to test the connection and if the responses are as expected.

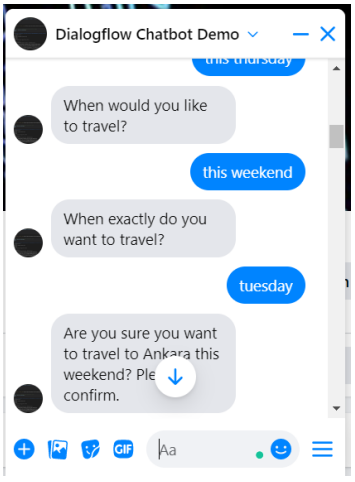
The dialog that I had with the chatbot on Facebook Messenger are shown with figures below. All the features that I mentioned till now for this project are tested and included there.



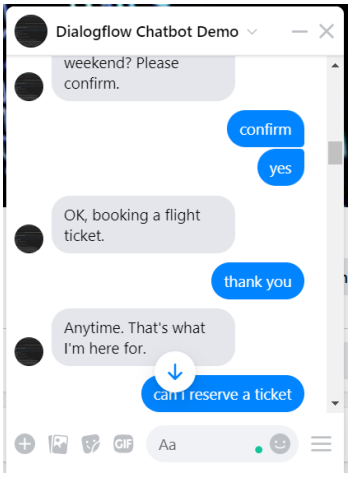
(1)



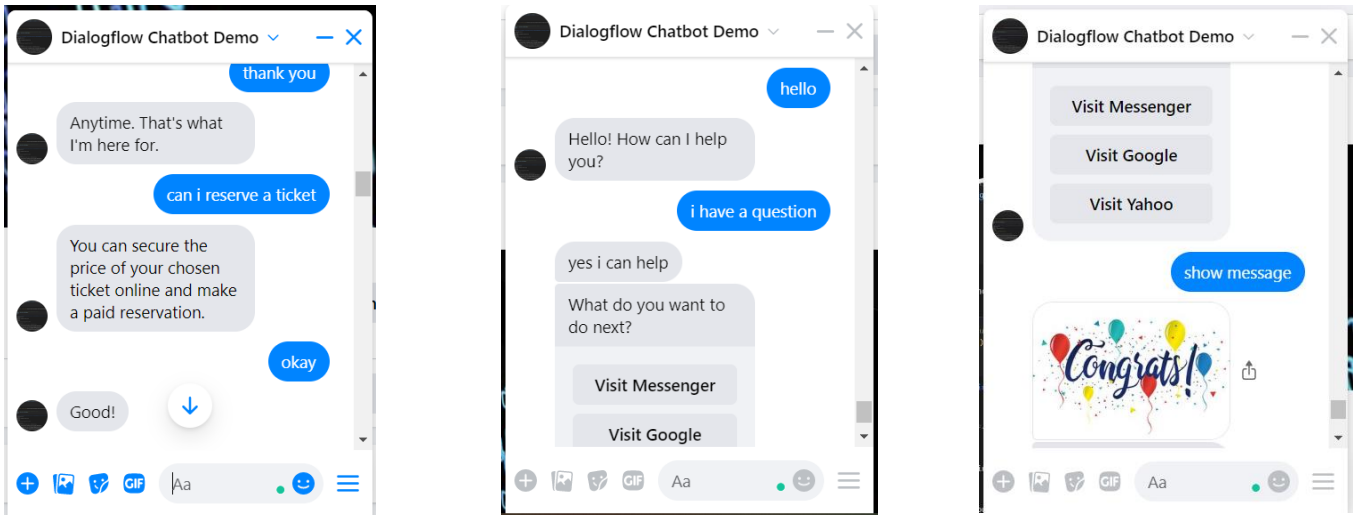
(2)



(3)



(4)



(5) (6) (7)

Figure 58: Test Conversation with the Chatbot on Facebook Messenger

A conversation is started with greeting words, such as hi and hello, and then some of the user expressions that added to the intents are tested. In picture (2) and (3), it can be seen that when the user gives the time of the flight as a day, then the chatbot does not take it as a valid date and asks for another valid input, which is ‘this weekend’ for example. This is because I added the entity as date-period in the parameters section (shown in figure 51). Then the conversation continues as expected according to the expressions and responses that I manually added. In the pictures (6) and (7), it can be seen that different types of responses, other than usual text response, are used. The buttons to visit some certain applications/websites are given as a response in picture (6), and an image is the response for the picture (7).

With this conversation, I tested the chatbot I have created and the responses that I was expecting to get for some certain user expressions, and saw that the chatbot is working well.

CONCLUSION

During my internship of 20 days, I worked on many Python projects, data labeling for defect detection on modems, and creating a basic chatbot on Google Dialogflow. I learned a new language, HTML, improved my existing knowledge of Python, and used a new platform to label data and create a chatbot. I had a chance to work on areas I have never experienced, such as data labeling and creating a chatbot, AI area. It made me realize that I like AI and want to learn more about its applications in different areas to improve myself and continue my academic career with a Master's degree after graduation on this path. I used many platforms during my work: PyCharm, Jupyter Notebook (Anaconda), labelImg, and Google Dialogflow. I did some research and watched tutorial videos to get familiar with those platforms before and during the time I used them.

I have gained so much valuable experience after my internship, and being in a real work environment where I can feel comfortable and welcomed improved my communication and group working skills. I also improved my critical thinking and problem-solving skills as I worked on many projects where I needed to solve the problems on my own in a limited time at some point. Apart from all these skills I gained and improved, the most crucial gain I had from my internship was making new connections with people who are very good in their field and working on different areas to help find my interests in the areas that I had no idea about.

The knowledge I had before my internship helped me a lot during my training, thanks to the courses I took until my fourth year. For instance, I had basic knowledge about Python from the CNG111 course and object-oriented programming from CNG 242 Programming Languages course, so I easily combined the existing knowledge I had and applied object-oriented programming to Python to write more efficient code for my projects. Also, as I noticed that I enjoyed working in the AI field, I checked the technical elective courses that our campus METU NCC provides and found out that a CNG409 Machine Learning course was going to offer for this Fall semester. As I found a related subject course in the area I like, I am taking a Machine Learning course this semester, and I am happy that what I learned about AI during my internship helps me understand the concept of this course better.

To sum up, I want to thank my supervisor Onur Odabaşı for his support and help during my training. He guided me in a very kind and wise way, and he was always ready to help me to improve myself by discussing the problems I faced instead of giving me the answer directly. I also want to thank Ersin Aksoy for giving me a chance to get an interview and accepting me to their AI department as an intern.

REFERENCES

1. The official website of Innova Bilişim Çözümleri A.Ş, https://www.innova.com.tr/tr

2. The documentation page for Turtle module Python, https://docs.python.org/3/library/turtle.html

3. The documentation page for Tkinter package Python, https://docs.python.org/3/library/tkinter.html

4. The documentation page for Pandas library Python, https://pandas.pydata.org/

5. The documentation page for Selenium package Python, https://pypi.org/project/selenium/

6. The documentation page for Requests package Python, https://pypi.org/project/requests/

7. The official website of PyCharm/Jet Brains, https://www.jetbrains.com/pycharm/

8. Wikipedia for Python, https://en.wikipedia.org/wiki/Python_(programming_language)

9. The official website of Python, https://www.python.org/

10. The official website of HTML, https://html.com/

11. Wikipedia for CSS, https://en.wikipedia.org/wiki/CSS

12. The website for CSS explanation and examples, https://www.w3schools.com/css/

13. Google Dialogflow Documentation, https://cloud.google.com/dialogflow/docs#:~:text=Dialogflow%20is%20a%20natural%20language,response%20system%2C%20and%20so%20on.