AISSCE 2021

BATTLESHIP

PROJECT IN COMPUTER SCIENCE USING PYTHON

REGN NO:

NAME: POOJA PREMNATH

BATCH: 2020 – 2021

D.A.V SR. SECONDARY SCHOOL,

MOGAPPAIR, CHENNAI – 600 037

BONAFIDE CERTIFICATE

Register No. Internal Assessment

Certified to be the Bonafide Project work in COMPUTER SCIENCE done by POOJA PREMNATH of Class XII Section B of D.A.V. SR SECONDARY SCHOOL during the year 2020-2021.

_____ _____

Signature of Principal Signature of Subject Teacher

Designation : PGT/TGT

School Seal

Submitted for the Practical Examination held on _____ at _____.

Internal Examiner

Chief Superintendent

External Examiner

Date_____

ACKNOWLEDGEMENT

I sincerely thank my school principal Mrs. RADHA SUBRAMANIAN for inspiring me and providing all the required support during this pandemic situation.

I would like to take this opportunity to thank my Computer Science teacher Mrs. RADHIKA BABU for her guidance and support, without which my project would not have seen the light of completion.

I would also like to thank our lab assistant Mrs. M.KALPANA for her guidance. I extend my gratitude to my project partner S. VAISHNAVIY for her contribution. I also express my sincere and heartfelt thanks to everyone who helped me in completing this project.

Pooja Premnath

TABLE OF CONTENTS

INTRODUCTION

Introduction

This computer version of Battleship is based on the popular board game of the same name. It is a strategy -based guessing game, played on a grid, on which a fleet of ships is concealed. The objective of the game is to find all the ships with a minimum number of attempts.

Once the program code is run, the opening screen lets the player view the rules of play, the top scorers, or directs them to begin playing.

On deciding to play, a dynamic screen urges the player to either sign in or sign up for the game. The sign-in screen checks the database of players if a valid username and password have been entered, otherwise directs the player to enter a valid entry, or to sign up instead, if the user has not yet played the game.

Once signed in, the player is directed to the main game screen. The number of ships and their length is indicated alongside a 10x10 grid. The player can now begin play by using the Cartesian coordinate system to identify any coordinate on the grid, and then click on its corresponding square, to verify if a ship is concealed in that position. The five ships hidden on the board, are in positions randomly allocated by the program, oriented in either the horizontal or vertical direction. The five ships are the Aircraft Carrier (5 units), Battleship (4 units), Submarine (3 units), Cruiser (3 units) and Destroyer (2 units). If the coordinate chosen corresponds to the location of a ship, it is

illuminated in red, and the player can hear a loud boom, an indication that a part of the ship has been hit. If the player has taken a shot into empty waters, the box turns white, so that, no more attempts are made on the same box again.

When the player finds an entire ship, the ship finally comes into view on the grid, and a check mark appears next to the corresponding ship's picture to the right of the grid. The player can also note the number of attempts he/she has made, and the number of ships sunk from the tally kept on the top right of the screen.

At any point of the game, should the player feel the need to forfeit, he/she has the option to do so. However, if the player successfully finds all the ships hidden in the grid, a celebratory tune is played, and the player is congratulated. He/she can then choose to play again, or view the game leaderboard, with a colourful bar graph highlighting the top 5 scorers.

Battleship is enjoyable to play, and constantly keeps a player on their feet to accurately guess the position of the fleet of ships on the grid. The suspenseful and interactive manner of the game, urges users to keep playing.

SYSTEM REQUIREMENTS

System Requirements

Operating system : Windows7

Software : IDLE PYTHON 3.7

Processor : Intel(R)Core (TM)
i5-8265U
CPU@1.60GHz 1.80GHz

Hard Disk : 512 GB SSD

RAM : 8 GB

VDU : HP

Keyboard : HP Keyboard

Mouse : HP Wireless Mouse

SOFTWARE DESCRIPTION

Software Description

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was developed by Guido van Rossum in February 1991. Python is influenced by two programming languages: ABC language and Modula-3.

Its high-level built -in data structures, combined with dynamic typing and dynamic binding, makes it very attractive to use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available without charge for all major platforms, and can be freely distributed.

Python’s popularity is primarily because of its increased productivity. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: when the interpreter discovers an error, it raises an exception. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting

breakpoints, stepping through the code a line at a time, and so on.

Python is also used to build professional-quality software, both as standalone applications and as web services. Python may not be the fastest language, but what it lacks in speed, it makes up for in versatility.

Python is thus an ideal teaching language, and it lets newcomers pick it up quickly. As a result, developers spend more time thinking about the problem they're trying to solve and less time thinking about language complexities.

MySQL

MySQL is a relational database management system based on the Structured Query Language, which is a popular language for accessing and managing records in a database. MySQL is an open-source and free software under the GNU license. It is supported by the Oracle Company. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, facilitates database integrity and creation of backups. MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often MySQL is used with other programs to implement applications that need relational database capability.

MySQL’s popularity arises out of its following advantages:

- **Ease of Management** – It is easy to download and use the software.
- **High performance** – It provides fast loading utilities with memory cache.
- **Compatibility** – MySQL is compatible with all modern platforms like Windows, Linux, Unix.
- **Performance** – MySQL gives high-performance results without losing essential functionality.

- **Complete Data Security** – Only authorized users can access the database
- **Low Cost** – It is free to use.
- **Memory Efficiency** – MySQL has low memory leakage.

MySQL can also be integrated with Python, to retrieve records and execute queries programmatically.

Microsoft Word was also used in the development of this project's documentation.

PROGRAM ANALYSIS

Program Analysis

Modules and Packages Imported:

1. Tkinter :#Library to develop a GUI (Graphical User Interface)
2. PIL (Python Imaging Library) :#Library to open and manipulate image file formats
3. MySQL connector :#Software to support Python-MySQL interface
4. Matplotlib :#A plotting library to embed plots into applications
5. Pygame :#Library for computer graphics and embedding sounds into applications
6. Random Module :#A module in the Python Standard Library to generate random numbers

User Defined Functions:

1. firstscreen() :#To display the opening screen, and the buttons to transfer control to the rules, the leaderboard, to play or to quit the game.

2. secondscreen() :#To display the options of sign in, sign up and to quit the game

3. rules() :#To display the rules of the game

4. signup_entry() :#To facilitate the entry of a new valid username and password

5. signin_entry() :#To facilitate the entry of an existing username and valid password

6. db() :#Establishing Python-MySQL connectivity

   signup_check() :#To check if the entered username already exists in the database, and if not, insert it into the table of players

   signin_check() :#To check if the entered username and password are correct, by comparing their values with those in the database table

7. grid() :#To draw a 10x10 grid, label the x and y axes, and place images of

		the ships that have to be found, along with their length
8.	mousepos()	:#To identify the x and y coordinate on the board corresponding to the player's mouse click, and to check if the coordinate corresponds to the location of a ship, and accordingly change the colour of the box to red for a hit and white for a miss. #To increment the number of attempts, and the number of ships that have been sunk, and to display a check mark, if a ship has completely been found. #To show appropriate error messages if a shot is taken outside the grid
9.	gridcoordinates()	:#To create a dictionary with the key as ordered pairs in the cartesian product of AxA where A={1,2,3,4,5,6,7,8,9,10}, that correspond to all the possible coordinates on the grid, and the values as the screen coordinates of each square.
10.	assignshippos()	:#To assign the length of each type of ship

11.	shippos()	:#A function that makes use of the random module to arbitrarily assign a coordinate from the grid, and randomly choose the direction of the ship. The function then checks from the initial starting coordinate, along the direction specified, if all the coordinates required for a ship to be placed there are vacant or not, and if they can be placed within the limits of the defined grid. This process continues until 5 ships on the board are allocated random positions.
12.	scoretracking()	:#To calculate the total number of attempts made, and continually update the score on the screen.
13.	shiponboard()	:# To place the picture of the ship on the board after it has been sunk, by identifying the orientation of the ship, and the screen coordinates corresponding to the (x,y) coordinates on the board.
14.	forfeit()	:#To seek confirmation from the player before terminating the game play, displaying the correct answers, and then giving the user the choice to play again, or to quit.

15.	consolation()	:#To reinitialize all the variables to either 0 or to their empty state, after a game play, or after forfeiting the game.
16.	win()	:#To display a congratulatory message for the successful completion of the game, changing the colours of the ships, and to give the user the choice of playing again, quitting, or to view the leaderboard.
17.	leaderboard()	:#To fetch values of the players and their scores from the database, sort them, and then use the top five scores as the top players, and to draw a table to display the names, ranks and scores of these players.
18.	plot()	:#To use matplotlib.pyplot to draw a bar graph to represent the top five players.
19.	boomsound(), winsound()	:#To use the mixer available in pygame to add sound effects for the boom noises and after completion of the game.
20.	quitgame()	:#To generate a message box, asking for confirmation before terminating the game

21.	fsrfromcanvas2(), fsrfromrules(), fsrfromsignup(), fsrfromsignin(), signupfromsignin(), fsrfromleaderboard(),	:#To navigate between canvases, by erasing all the content, and packing the new canvas in place, with the required widgets.

PROGRAM CODE

```
#The purpose of this program is to create a
computerized version of the classical board game
Battleship


from tkinter import *
from PIL import ImageTk,Image
import mysql.connector as mc
import random
from tkinter import messagebox as mb
import matplotlib.pyplot as plt

#to establish connectivity between matplotlib
and tkinter
from matplotlib.backends.backend_tkagg import
FigureCanvasTkAgg,NavigationToolbar2Tk
from matplotlib.figure import Figure
import pygame
pygame.mixer.init()
root= Tk()
root.title('Battleship')

#home screen
def firstscreen():
    global canvas1,pic
    canvas1=Canvas(root,width=1000,height=800)
    canvas1.pack()
    image = Image.open('sea2.jpg')
    image = image.resize((1000, 800),
Image.ANTIALIAS)
    pic= ImageTk.PhotoImage(image)

canvas1.create_image(0,0,anchor=NW,image=pic)

canvas1.create_text(515,250,text='BATTLESHIP',fi
```

```
ll='azure',font=('century
schoolbook',60,'bold'))
    canvas1.create_text(800,600,text='Pooja
Premnath', fill='white',font=('century
schoolbook',15))
    canvas1.create_text(800,630,text='Vaishnaviy
S', fill='white',font=('century schoolbook',15))
    button1 =Button(text='PLAY',
command=secondscreen, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    button1.config(height=2,width=23)
    canvas1.create_window(510,370,
window=button1)
    button1a=Button(text='RULES', command=rules,
bg='floral white', fg='black',
font=('helvetica', 11, 'bold'))
    button1a.config(height=2,width=23)
    canvas1.create_window(510,430,
window=button1a)
    leader=Button(text='LEADERBOARD',
command=leaderboardfromfsr, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    leader.config(height=2,width=23)
    canvas1.create_window(510,490,
window=leader)
    endgame=Button(text='QUIT',
command=quitgame, bg='floral white', fg='black',
font=('helvetica', 11, 'bold'))
    endgame.config(height=2,width=23)
    canvas1.create_window(510,550,
window=endgame)

def secondscreen():
    canvas1.pack_forget()
    global canvas2
```

```
    canvas2=Canvas(root, width = 1000, height =
1000)
    canvas2.pack()
    image2 = Image.open('ship2.jpg')
    image2 = image2.resize((1000, 800),
Image.ANTIALIAS)
    global pic2
    pic2= ImageTk.PhotoImage(image2)

canvas2.create_image(0,0,anchor=NW,image=pic2)

canvas2.create_text(380,140,text="Battleship",fi
ll='black',font=('century
schoolbook',55,'bold'))
    global button2
    button2=Button(text='Home',
command=fsrfromcanvas2, bg='azure', fg='black',
font=('helvetica', 9, 'bold'))
    button2.config(height=2,width=10)
    canvas2.create_window(50,50, window=button2)
    signup= Button(text='SIGN  UP',
command=signup_entry, bg='azure', fg='black',
font=('century schoolbook', 11,'bold'))
    signup.config(height=2,width=30)
    canvas2.create_window(390,410,
window=signup)
    signin= Button(text='SIGN  IN',
command=signin_entry, bg='azure', fg='black',
font=('century schoolbook', 11,'bold'))
    signin.config(height=2,width=30)
    canvas2.create_window(390,330,
window=signin)
    bye= Button(text='QUIT', command=quitgame,
bg='azure', fg='black', font=('century
schoolbook', 11,'bold'))
```

```
    bye.config(height=2,width=30)
    canvas2.create_window(390,490, window=bye)

#for a new user to sign up
def signup_entry():
    global status
    status='signup'
    canvas2.pack_forget()
    global signupscreen
    signupscreen=Canvas(root, width = 1000,
height = 1000)
    signupscreen.pack()
    image = Image.open('sea.jpg')
    image = image.resize((1000, 800),
Image.ANTIALIAS)
    global pic5
    pic5= ImageTk.PhotoImage(image)

signupscreen.create_image(0,0,anchor=NW,image=pi
c5)
    global button2
    button2=Button(text='Home',
command=fsrfromsignup, bg='azure', fg='black',
font=('helvetica', 9, 'bold'))
    button2.config(height=2,width=10)
    signupscreen.create_window(50,50,
window=button2)
    signupscreen.create_text(450,180,text='Sign
Up',fill='azure',font=('century
schoolbook',60,'bold'))
    signupscreen.create_text(430,260,text="Let's
Get Started! Enter a username and
password...",fill='orchid1',font=('Lucida
Handwriting',15,'bold'))
    global entry1
```

```
    global entry2
    text=StringVar(value=('Times New Roman 30'))
    entry1 =Entry(root,font=text)
    entry2=Entry(root,font=text,show='*')
    entry1.configure(width=25)
    entry2.configure(width=25)
    signupscreen.create_window(500, 380,
window=entry1)

signupscreen.create_window(500,450,window=entry2
)

signupscreen.create_text(300,380,text="Username:
",fill='azure',font=('century
schoolbook',15,'bold'))

signupscreen.create_text(300,450,text="Password:
",fill='azure',font=('century
schoolbook',15,'bold'))
    submit= Button(text='Submit', command=db,
bg='azure', fg='black', font=('century
schoolbook', 11,'bold'))
    submit.config(height=2,width=23)
    signupscreen.create_window(500,540,
window=submit,tag='submitbutton')
    bye= Button(text='Quit', command=quitgame,
bg='azure', fg='black', font=('century
schoolbook', 11,'bold'))
    bye.config(height=2,width=23)
    signupscreen.create_window(870,660,
window=bye)

#for an existing user to sign up
def signin_entry():
```

```
    global
status,signinscreen,signinpic,button2,entry1,ent
ry2
    status='signin'
    canvas2.pack_forget()
    signinscreen=Canvas(root, width = 1000,
height = 1000)
    signinscreen.pack()
    image = Image.open('ship6.jpg')
    image = image.resize((1000, 800),
Image.ANTIALIAS)
    signinpic= ImageTk.PhotoImage(image)

signinscreen.create_image(0,0,anchor=NW,image=si
gninpic)
    button2=Button(text='Home',
command=fsrfromsignin, bg='azure', fg='black',
font=('helvetica', 9, 'bold'))
    button2.config(height=2,width=10)
    signinscreen.create_window(50,50,
window=button2)
    signinscreen.create_text(310,180,text='Sign
In',fill='black',font=('century
schoolbook',60,'bold'))
    signinscreen.create_text(280,260,text="Enter
your username and
password...",fill='navy',font=('Lucida
Handwriting',15,'bold'))
    text=StringVar(value=('Times New Roman 30'))
    entry1 =Entry(root,font=text)
    entry2=Entry(root,font=text,show='*')
    entry1.configure(width=25)
    entry2.configure(width=25)
    signinscreen.create_window(300, 380,
window=entry1)
```

```
signinscreen.create_window(300,450,window=entry2
)

signinscreen.create_text(100,380,text="Username:
",fill='black',font=('century
schoolbook',15,'bold'))

signinscreen.create_text(100,450,text="Password:
",fill='black',font=('century
schoolbook',15,'bold'))
    submit= Button(text='Submit', command=db,
bg='azure', fg='black', font=('century
schoolbook', 11,'bold'))
    submit.config(height=2,width=23)
    signinscreen.create_window(300,550,
window=submit)
    bye= Button(text='Quit', command=quitgame,
bg='azure', fg='black', font=('century
schoolbook', 11,'bold'))
    bye.config(height=2,width=23)
    signinscreen.create_window(870,660,
window=bye)

def rules():
    canvas1.pack_forget()
    global canvas3,pic3,button3
    canvas3=Canvas(root,width=1000,height=800)
    canvas3.pack()
    image = Image.open('ship5.jpg')
    image = image.resize((1000, 800),
Image.ANTIALIAS)
    pic3= ImageTk.PhotoImage(image)

canvas3.create_image(0,0,anchor=NW,image=pic3)
```

```
    canvas3.create_text(450,350,text='''\t\t*The
objective of the game is to sink all five of the
enemy's ships.
\n\t\t*There are five ships of varying size
placed on the grid.
\n\t\t*Take shots on the ships placed by
clicking on any of the squares on the grid.
\n\t\t*The positions in which ships have been
successfully hit are illuminated in red.
\n\t\t*If you have hit an empty area, the given
coordinate is illuminated in white.
\n\t\t*The lesser the number of hits taken to
sink all the ships, the better is your
score.''',fill='alice blue',font=('Arial',16))

canvas3.create_text(480,120,text='RULES',fill='g
old',font=('century schoolbook',60,'bold'))
    button3=Button(text='Home',
command=fsrfromrules, bg='azure', fg='black',
font=('helvetica', 9, 'bold'))
    button3.config(height=2,width=10)
    canvas3.create_window(50,50, window=button3)

#establishing Python-MySQL Connectivity to check
validity of details entered
def db():

conn=mc.connect(host='localhost',user='root',pas
sword='pooja',database='x')
    cur=conn.cursor()
    global f,p, entry1,entry2,submit
    f=entry1.get()
    p=entry2.get()
    def signup_check():
        global status
```

```
        cur.execute("select username from
score")
        rec=cur.fetchall()
        if f!='' and p!='':
            for i in rec:
                if i==(f,):

signupscreen.create_text(450,600,text="Oops..It
looks a user of the same name already exists.
Enter another
username.",fill='azure',font=('century
schoolbook',15,'bold'),tag='invaliduser')

signupscreen.delete('invalidpass')
                    break
            else:
                if p!='':
                    cur.execute("""insert into
score(username,password)
values("%s","%s")"""%(f,p))

signupscreen.create_text(450,600,text="Yay!
You're now ready to play. Click Play to
Begin.",fill='azure',font=('century
schoolbook',15,'bold'),tag='invalidpass')

entry1=Entry(root,state='disabled')

entry2=Entry(root,state='disabled')

signupscreen.delete('invaliduser')

signupscreen.delete('signupbutton')
                    conn.commit()
```

```
playbutton1=Button(text='PLAY', command=grid,
bg='azure', fg='black', font=('century
schoolbook', 11, 'bold'))

playbutton1.config(height=2,width=23)

signupscreen.create_window(500,540,
window=playbutton1,tag='play')
    def signin_check():
        global status
        cur.execute('select username,password
from score')
        rec=cur.fetchall()
        if f!='' and p!='':
            for i in rec:
                if i[0]==f:
                    if i[1]==p:

signinscreen.delete('label4')

signinscreen.delete('invalid')

signinscreen.create_text(300,650,text="You're
Back!..Click Play to
Begin",fill='black',font=('century
schoolbook',20,'bold'),tag='label3')

playbutton2=Button(text='PLAY', command=grid,
bg='azure', fg='black', font=('century
schoolbook', 11, 'bold'))

playbutton2.config(height=2,width=23)
```

```
signinscreen.create_window(300,550,
window=playbutton2,tag='play')
                        break

                    else:

signinscreen.create_text(300,650,text="Incorrect
Password.. Try
Again",fill='black',font=('century
schoolbook',20,'bold'),tag='label4')
                        break

            else:

signinscreen.create_text(400,640,text="You might
have entered an incorrect username, \nor you're
not in our records yet. Try Again or Sign up
instead.",fill='black',font=('century
schoolbook',15,'bold'),tag='invalid')
                signup= Button(text='Sign Up',
command=signupfromsignin, bg='azure',
fg='black', font=('century schoolbook',
11,'bold'))
                signup.config(height=2,width=23)

signinscreen.create_window(560,550,
window=signup)
                signinscreen.delete('label3')
                signinscreen.delete('label4')
    if status=='signup':
        signup_check()
    else:
        signin_check()
    conn.close()
```

```
#To draw the game grid
def grid():

   global
f,status,signupscreen,signinscreen,origin,canvas
4,pic,aircraftcarrier,battleship,submarine,cruis
er,destroyer,d,squares,gameover
   origin='game'

conn=mc.connect(host='localhost',user='root',pas
sword='pooja',database='x')
   cur=conn.cursor()
   cur.execute("""select bestscore from score
where username='%s'"""%(f))
   best=cur.fetchone()
   conn.close()
   if status=='signin':
       signinscreen.pack_forget()
   else:
       signupscreen.pack_forget()
   canvas4=Canvas(root,width=1000,height=800)
   canvas4.pack()
   image = Image.open('main2.jpg')
   image = image.resize((1000, 800),
Image.ANTIALIAS)
   pic= ImageTk.PhotoImage(image)
   canvas4.create_image(0,0,anchor=NW,image=pic)
   canvas4.create_text(800,50,text='Battleship',
fill='white',font=('century
schoolbook',30,'bold'))
   canvas4.create_text(750,100,text='Number of
ships sunk:', fill='white',font=('Times New
Roman',15,'bold'),tag='line1')
```

```
    canvas4.create_text(755,130,text='Number of
Attempts:', fill='white',font=('Times New
Roman',15,'bold'),tag='line2')
    canvas4.create_text(920,120,text='___',
fill='white',font=('Times New
Roman',15,'bold'),tag='blank')
    canvas4.create_text(770,160,text='Your Best
Score:', fill='white',font=('Times New
Roman',15,'bold'),tag='line3')
    if best!=None and best[0]!=100:

canvas4.create_text(920,160,text='{}'.format(bes
t[0]), fill='white',font=('Times New
Roman',15,'bold'),tag='line5')
    else:
        canvas4.create_text(920,160,text='NA',
fill='white',font=('Times New
Roman',15,'bold'),tag='line5')
    canvas4.create_text(350,30,text='{}, Try to
find all the ships hidden in as few attempts as
possible...'.format(f), fill='gold',font=('Times
New Roman',15))
    image =
Image.open('aircraftcarrierfinal.png')
    image = image.resize((300, 100),
Image.ANTIALIAS)
    aircraftcarrier= ImageTk.PhotoImage(image)

canvas4.create_image(750,220,image=aircraftcarri
er)
    canvas4.create_text(700,190,text='Aircraft
Carrier:5 units', fill='white',font=('Times New
Roman',10))
    image = Image.open('battleshipfinal.png')
```

```
    image = image.resize((250, 100),
Image.ANTIALIAS)
    battleship= ImageTk.PhotoImage(image)

canvas4.create_image(750,325,image=battleship)

canvas4.create_text(680,295,text='Battleship:4
units', fill='white',font=('Times New
Roman',10))
    image = Image.open('submarinefinal.png')
    image = image.resize((200, 100),
Image.ANTIALIAS)
    submarine= ImageTk.PhotoImage(image)
    canvas4.create_image(730,420,image=submarine)
    canvas4.create_text(680,390,text='Submarine:3
units', fill='white',font=('Times New
Roman',10))
    image = Image.open('cruiserfinal.png')
    image = image.resize((200, 100),
Image.ANTIALIAS)
    cruiser= ImageTk.PhotoImage(image)
    canvas4.create_image(730,520,image=cruiser)
    canvas4.create_text(680,475,text='Cruiser:3
units', fill='white',font=('Times New
Roman',10))
    image = Image.open('destroyerfinal.png')
    image = image.resize((150, 80),
Image.ANTIALIAS)
    destroyer= ImageTk.PhotoImage(image)
    canvas4.create_image(710,620,image=destroyer)
    canvas4.create_text(680,585,text='Destroyer:2
units', fill='white',font=('Times New
Roman',10))
    global d
    d=grid_coordinates()
```

```
    y=100
    for i in range(11):

canvas4.create_line(100,y,600,y,fill='white')
        y+=50
    x=100
    for i in range(11):

canvas4.create_line(x,100,x,600,fill='white')
        x+=50
    txc=130
    yc=80
    for i in range(1,11):

canvas4.create_text(txc,yc,text=str(i),fill='whi
te',font=('century schoolbook',30,'bold'))
        txc+=50
    lxc=75
    lyc=125
    for i in range(1,11):

canvas4.create_text(lxc,lyc,text=str(i),fill='wh
ite',font=('century schoolbook',30,'bold'))
        lyc+=50
    squares=[]
    canvas4.bind("<Button-1>",lambda event,
arg=shipdict: mousepos(event, arg))
    canvas4.bind("<Button-1>",lambda
event,arg=occupied: mousepos(event,arg))
    canvas4.pack()
    quitbutton=Button(text='FORFEIT',
command=forfeit, bg='floral white', fg='black',
font=('helvetica', 11, 'bold'))
    quitbutton.config(height=2,width=23)
```

```
    canvas4.create_window(890,680,
window=quitbutton,tag='forfeitbutton')
    gameover=False
    conn.close()

#To recognize the coordinates of a mouse click
def mousepos(event,arg):

conn=mc.connect(host='localhost',user='root',pas
sword='pooja',database='x')
    cur=conn.cursor()
    global
hitdict,f,squares,white,red,hit,miss,shipdict
    if gameover==True:
        return

    coordinate=(event.x,event.y)
    for i in d:
        if d[i][0]<coordinate[0]<d[i][2] and
d[i][1]<coordinate[1]<d[i][3]:

            if i in occupied:

canvas4.create_rectangle(d[i],fill='red',tag='re
ct')
                canvas4.delete('outseapos')

                squares.append(d[i])
                if d[i] not in red:
                    boomsound()
                    red.append(d[i])
                    hit+=1
                    scoretracking()
                    for j in shipdict:
                        for k in shipdict[j]:
```

```
                                if k==i:

hitdict[j].append(i)


                    sink=0
                    for i in shipdict:
                        if all(item in hitdict[i]
for item in shipdict[i]):
                            if i=='aircraft
carrier':
                                global check1
                                image =
Image.open('checkfinal.png')
                                check1=
ImageTk.PhotoImage(image)

canvas4.create_image(920,230,image=check1)
                                shiponboard(i)

                            elif i=='battleship':

                                global check2
                                image =
Image.open('checkfinal.png')
                                check2=
ImageTk.PhotoImage(image)

canvas4.create_image(920,330,image=check2)
                                shiponboard(i)

                            elif i=='submarine':

                                global check3
```

```
                                image =
Image.open('checkfinal.png')
                                check3=
ImageTk.PhotoImage(image)

canvas4.create_image(920,430,image=check3)
                                shiponboard(i)

                            elif i=='cruiser':

                                global check4
                                image =
Image.open('checkfinal.png')
                                check4=
ImageTk.PhotoImage(image)

canvas4.create_image(920,530,image=check4)
                                shiponboard(i)

                            elif i=='destroyer':
                                global check5
                                image =
Image.open('checkfinal.png')
                                check5=
ImageTk.PhotoImage(image)

canvas4.create_image(920,630,image=check5)
                                shiponboard(i)

                            sink+=1

                            canvas4.delete('sink')

canvas4.create_text(920,100,text=sink,
```

```
fill='white',font=('Times New
Roman',15,'bold'),tag='sink')

                if sink==5:
                    win()
                    winsound()
                    cur.execute("""update score
set bestscore=("%d") where username=("%s") and
bestscore>=("%d")"""%(hit+miss,f,hit+miss))
                    conn.commit()

            else:

canvas4.create_rectangle(d[i],fill='white',tag='
rect')
                canvas4.delete('outseapos')
                squares.append(d[i])
                if d[i] not in white:
                    white.append(d[i])
                    miss+=1
                    scoretracking()


            break

    else:
        canvas4.delete('seapos')

canvas4.create_text(350,650,text='Oops..You
tried to make a shot outside the
sea!',fill='red',font=('Times New
Roman',20,'bold'),tag='outseapos')

#a dictionary of coordinates available on the
grid
```

```
def grid_coordinates():
    global d
    d={}
    x1=100
    y1=100
    x2=150
    y2=150
    for i in range(1,11):
       for j in range(1,11):
           d[(j,i)]=(x1,y1,x2,y2)
           x1+=50
           x2+=50
       x1=100
       x2=150
       y1+=50
       y2+=50

    return d



occupied=[]
shipdict={}
direction=['horizontal','vertical']
def assignshippos():
    global shipname,count
    shipname=['aircraft
carrier','battleship','cruiser','submarine','des
troyer']
    count=0
    global shiplength
    shiplength=[5,4,3,3,2]
    while count<5:
        shippos()
```

```
#random generation of the position of the ships
def shippos():
    global count,ship,shiplength
    ship=[]

sc=(random.randrange(1,11),random.randrange(1,11
))
    shipdir=random.choice(direction)
    if sc in occupied:
        return
    else:

        if shipdir=='horizontal':
            for i in
range(sc[0]+1,(sc[0]+shiplength[count])):
                if (i,sc[1]) in occupied or
i>10:
                    return
            else:
                occupied.append(sc)
                ship.append(sc)
                for i in
range(sc[0]+1,sc[0]+shiplength[count]):
                    ship.append((i,sc[1]))
                    occupied.append((i,sc[1]))
                shipdict[shipname[count]]=ship
                count+=1
        else:
            for i in
range(sc[1]+1,(sc[1]+shiplength[count])):
                if (sc[0],i) in occupied or
i>10:
                    return
            else:
                occupied.append(sc)
```

```
                ship.append(sc)
                for i in
range(sc[1]+1,sc[1]+shiplength[count]):
                    ship.append((sc[0],i))
                    occupied.append((sc[0],i))

                shipdict[shipname[count]]=ship
                count+=1

def scoretracking():
    global hit,miss
    canvas4.delete('blank')
    canvas4.delete('score')
    canvas4.create_text(920,130,text=hit+miss,
fill='white',font=('Times New
Roman',15,'bold'),tag='score')

#placing the images of the ships on the board
def shiponboard(i):
    if i=='aircraft carrier':
        global ac,sub,bship,cship,dship
        if shipdict[i][0][0]==shipdict[i][1][0]:
            image =
Image.open('aircraftcarrierfinal.png')
            image = image.resize((300, 70),
Image.ANTIALIAS)

image=image.transpose(Image.ROTATE_90)
            ac= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][0]+20,d[s
hipdict[i][0]][3]+70,image=ac)

        else:
```

```
            image =
Image.open('aircraftcarrierfinal.png')
            image = image.resize((300, 70),
Image.ANTIALIAS)
            ac= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][2]+80,d[s
hipdict[i][0]][3]-30,image=ac)
    elif i=='battleship':
        if shipdict[i][0][0]==shipdict[i][1][0]:

            image =
Image.open('battleshipfinal.png')
            image = image.resize((200, 70),
Image.ANTIALIAS)

image=image.transpose(Image.ROTATE_90)
            bship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][0]+30,d[s
hipdict[i][0]][3]+50,image=bship)
        else:

            image =
Image.open('battleshipfinal.png')
            image = image.resize((200, 70),
Image.ANTIALIAS)
            bship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][2]+60,d[s
hipdict[i][0]][3]-20,image=bship)
    elif i=='submarine':
        if shipdict[i][0][0]==shipdict[i][1][0]:
```

```
            image =
Image.open('submarinefinal.png')
            image = image.resize((160, 70),
Image.ANTIALIAS)

image=image.transpose(Image.ROTATE_90)
            sub= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][0]+30,d[s
hipdict[i][0]][3]+30,image=sub)
        else:

            image =
Image.open('submarinefinal.png')
            image = image.resize((160, 70),
Image.ANTIALIAS)
            sub= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][2]+25,d[s
hipdict[i][0]][3]-20,image=sub)
    elif i=='cruiser':
        if shipdict[i][0][0]==shipdict[i][1][0]:

            image =
Image.open('cruiserfinal.png')
            image = image.resize((160, 70),
Image.ANTIALIAS)

image=image.transpose(Image.ROTATE_90)
            cship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][0]+20,d[s
hipdict[i][0]][3]+25,image=cship)
        else:
```

```
            image =
Image.open('cruiserfinal.png')
            image = image.resize((160, 70),
Image.ANTIALIAS)
            cship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][2]+30,d[s
hipdict[i][0]][3]-30,image=cship)

    elif i=='destroyer':
        if shipdict[i][0][0]==shipdict[i][1][0]:
            image =
Image.open('destroyerfinal.png')
            image = image.resize((110, 60),
Image.ANTIALIAS)

image=image.transpose(Image.ROTATE_90)
            dship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][0]+20,d[s
hipdict[i][0]][3]-3,image=dship)
        else:
            image =
Image.open('destroyerfinal.png')
            image = image.resize((110, 60),
Image.ANTIALIAS)
            dship= ImageTk.PhotoImage(image)

canvas4.create_image(d[shipdict[i][0]][2]+3,d[sh
ipdict[i][0]][3]-30,image=dship)


def forfeit():
    global d,gameover,squares,hit,miss
```

```
    if mb.askyesno('Forfeit Game','Do you really
want to quit? Your progress will be lost.'):
        gameover=True
        hit=miss=0
        for i in squares:
            canvas4.delete('rect')

colours=['purple1','gold','magenta2','pale
turquoise','SpringGreen2']
        count=0
        for i in shipdict:
            for j in range(len(shipdict[i])):

canvas4.create_rectangle(d[shipdict[i][j]],fill=
colours[count])
                shiponboard(i)
            count+=1
        canvas4.delete('forfeitbutton')
        canvas4.delete('outseapos')
        canvas4.create_text(350,650,text="It's
OK... \nBetter luck next
time!",fill='red',font=('Times New
Roman',20,'bold'))

        replay=Button(text='Play Again',
command=consolation, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
        replay.config(height=2,width=23)
        canvas4.create_window(890,680,
window=replay)
        bye= Button(text='Quit',
command=quitgame, bg='floral white', fg='black',
font=('century schoolbook', 11,'bold'))
        bye.config(height=2,width=23)
```

```
        canvas4.create_window(660,680,
window=bye)

#to reinitialize varibles to either 0 or to
their empty state, to play once again
def consolation():
    global
hitdict,squares,gameover,red,white,hit,miss,occu
pied,shipdict
    gameover=True
    canvas4.pack_forget()
    hitdict={'aircraft
carrier':[],'battleship':[],'cruiser':[],'submar
ine':[],'destroyer':[]}
    occupied=[]
    shipdict={}
    assignshippos()
    print("The dictionary of ships is",shipdict)
    squares=[]
    red=[]
    white=[]
    hit=0
    miss=0
    firstscreen()

#after winning the game
def win():
    global gameover,squares
    print('yay you have won the game')
    canvas4.delete('line1')
    canvas4.delete('line2')
    canvas4.delete('line3')
    canvas4.delete('score')
    canvas4.delete('sink')
    canvas4.delete('line5')
```

```
    canvas4.create_text(800,140,text="You
Win!",fill='gold',font=('Lucida
Handwriting',30))
    gameover=True
    canvas4.delete('forfeitbutton')
    canvas4.delete('outseapos')
    for i in squares:
        canvas4.delete('rect')

colours=['purple1','gold','magenta2','pale
turquoise','SpringGreen2']
        count=0
        for i in shipdict:
            for j in range(len(shipdict[i])):

canvas4.create_rectangle(d[shipdict[i][j]],fill=
colours[count])
            count+=1
    replay=Button(text='Play Again',
command=consolation, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    replay.config(height=2,width=23)
    canvas4.create_window(890,680,
window=replay)
    bye= Button(text='Quit', command=quitgame,
bg='floral white', fg='black',
font=('helvetica', 11,'bold'))
    bye.config(height=2,width=23)
    canvas4.create_window(430,680, window=bye)
    scorelist=Button(text='Leaderboard',
command=leaderboard, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    scorelist.config(height=2,width=23)
    canvas4.create_window(660,680,
window=scorelist)
```

```
def leaderboardfromfsr():
    global origin
    canvas1.pack_forget()
    origin='front'
    leaderboard()

#to draw a table to display the top scorers from
the database
def leaderboard():
    global origin,canvas5,pic5

conn=mc.connect(host='localhost',user='root',pas
sword='pooja',database='x')
    if origin!='front':
        canvas4.pack_forget()
    cur=conn.cursor()
    canvas5=Canvas(root, width = 600, height =
1000)
    canvas5.pack()
    replay=Button(text='Home',
command=fsrfromleaderboard, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    replay.config(height=2,width=20)
    canvas5.create_window(300,650,
window=replay)
    image = Image.open('leaderboard2.jpg')
    image = image.resize((1000, 800),
Image.ANTIALIAS)
    pic5= ImageTk.PhotoImage(image)

canvas5.create_image(0,0,anchor=NW,image=pic5)
    y=100
    for i in range(7):
```

```
canvas5.create_line(50,y,400,y,fill='black',widt
h=3)
       y+=75

canvas5.create_line(50,100,50,550,fill='black',w
idth=3)

canvas5.create_line(125,100,125,550,fill='black'
,width=3)

canvas5.create_line(290,100,290,550,fill='black'
,width=3)

canvas5.create_line(400,100,400,550,fill='black'
,width=3)

canvas5.create_text(300,40,text='Leaderboard',
fill='black',font=('century
schoolbook',30,'bold'))
    canvas5.create_text(80,130,text='Rank',
fill='black',font=('century
schoolbook',15,'bold'))
    canvas5.create_text(210,130,text='Username',
fill='black',font=('century
schoolbook',15,'bold'))
    canvas5.create_text(345,130,text='Best
Score', fill='black',font=('century
schoolbook',15,'bold'))
    cur.execute('select username,bestscore from
score order by bestscore')
    global top
    top=cur.fetchall()
    xcoordinate=80
    ycoordinate=200
```

```
    if len(top)>=5:
        for i in range(0,5):

canvas5.create_text(xcoordinate,ycoordinate,text
='{}'.format(i+1), fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+120,ycoordinate,
text='{}'.format(top[i][0]),
fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+260,ycoordinate,
text='{}'.format(top[i][1]),
fill='black',font=('century
schoolbook',16,'bold'))
            ycoordinate+=80
    else:
        for i in range(5):
            if i<len(top):

canvas5.create_text(xcoordinate,ycoordinate,text
='{}'.format(i+1), fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+120,ycoordinate,
text='{}'.format(top[i][0]),
fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+260,ycoordinate,
text='{}'.format(top[i][1]),
fill='black',font=('century
schoolbook',16,'bold'))
                ycoordinate+=80
```

```
            else:

canvas5.create_text(xcoordinate,ycoordinate,text
='{}'.format(i+1), fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+120,ycoordinate,
text='NA', fill='black',font=('century
schoolbook',16,'bold'))

canvas5.create_text(xcoordinate+260,ycoordinate,
text='NA', fill='black',font=('century
schoolbook',16,'bold'))
                ycoordinate+=80
    print(top)
    graph=Button(text='View Bar Graph of
Statistics', command=plot, bg='floral white',
fg='black', font=('helvetica', 11, 'bold'))
    graph.config(height=2,width=50)
    canvas5.create_window(300,600, window=graph)

#using matplotlib.pyplot to plot a bar graph
def plot():
    global top
    top_win = Toplevel(root)
    if len(top)>=5:
        x
=[top[4][1],top[3][1],top[2][1],top[1][1],top[0]
[1]]
    else:
        x=[]
        for i in range(5):
            if i<len(top):
                x.insert(0,top[i][1])
            else:
```

```
                x.insert(0,0)


y=[top[4][0],top[3][0],top[2][0],top[1][0],top[0
][0]]

c=['deeppink','mediumorchid','slateblue','turquo
ise','lime']
    fig = plt.figure(figsize=(8,8))
    plt.barh(y,x,0.5,color=c)
    plt.title('Leaderboard Rankings')
    plt.xlabel('Number of Attempts')
    plt.ylabel('Username')
    for x,y in zip(x,y):
        label="{:.2f}".format(x)

        plt.annotate(label,
                     (x,y),
                     textcoords="offset points",
                     xytext=(13,0),
                     ha='center')

    canvas = FigureCanvasTkAgg(fig,
master=top_win)
    canvas.draw()
    canvas.get_tk_widget().pack()

#using pygame to add sound effects
def boomsound():

pygame.mixer.music.load('C:\\Users\\Pooja\\Docum
ents\\Pooja\\Class XII\\Computer
Science\\Computer Science Project\\Boom.mp3')
    pygame.mixer.music.play(loops=0)
```

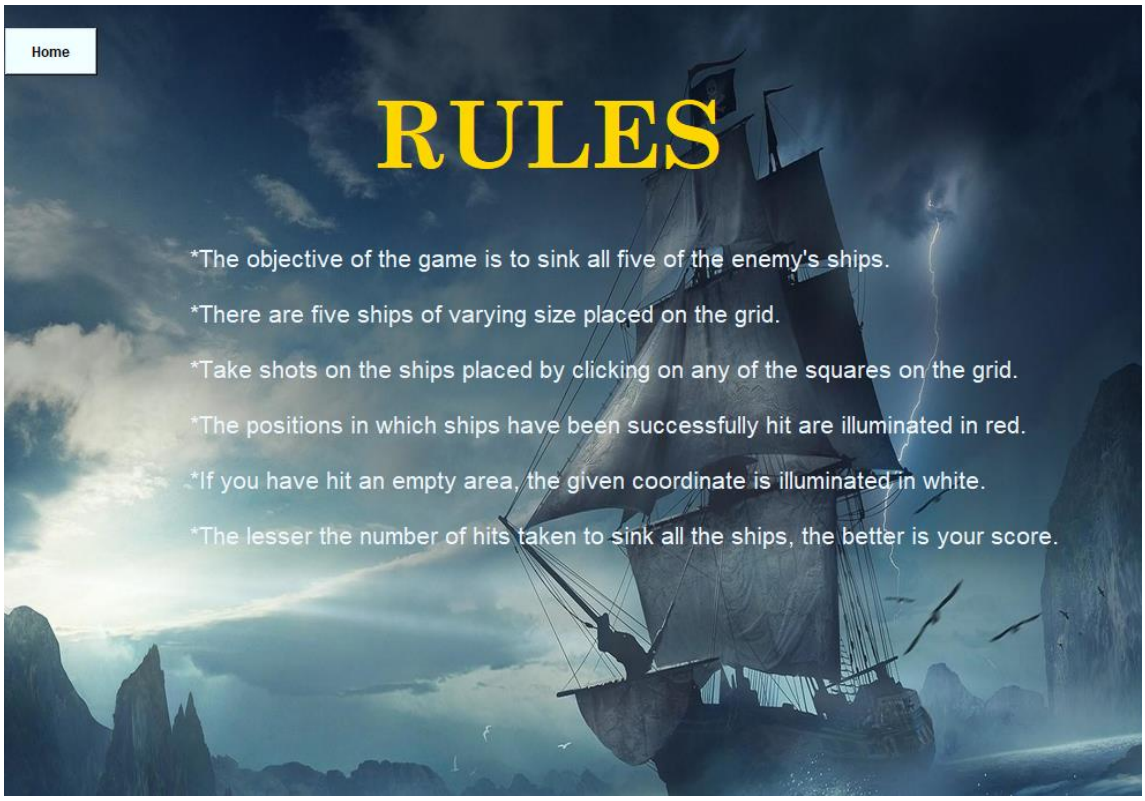
```
def winsound():

pygame.mixer.music.load('C:\\Users\\Pooja\\Docum
ents\\Pooja\\Class XII\\Computer
Science\\Computer Science
Project\\Victory2.mp3')
    pygame.mixer.music.play(loops=0)

#functions for navigation between screens
def fsrfromcanvas2():
    canvas2.pack_forget()
    firstscreen()

def fsrfromrules():
    canvas3.pack_forget()
    firstscreen()

def fsrfromsignup():
    signupscreen.pack_forget()
    firstscreen()

def fsrfromsignin():
    signinscreen.pack_forget()
    firstscreen()

def signupfromsignin():
    signinscreen.pack_forget()
    signup_entry()

def fsrfromleaderboard():
    global
hitdict,squares,gameover,red,white,hit,miss,occu
pied,shipdict
    canvas5.pack_forget()
    occupied=[]
```

```
    shipdict={}
    assignshippos()
    print("The dictionary of ships is",shipdict)
    squares=[]
    red=[]
    white=[]
    hit=0
    miss=0
    firstscreen()

def quitgame():
    if mb.askyesno('Quit Game','Do you really
want to quit the game?'):
        root.destroy()


#main body
red=[]
white=[]
hit=0
miss=0
hitdict={'aircraft
carrier':[],'battleship':[],'cruiser':[],'submar
ine':[],'destroyer':[]}
firstscreen()
assignshippos()
print("The dictionary of ships is",shipdict)
```

SCREEN LAYOUTS

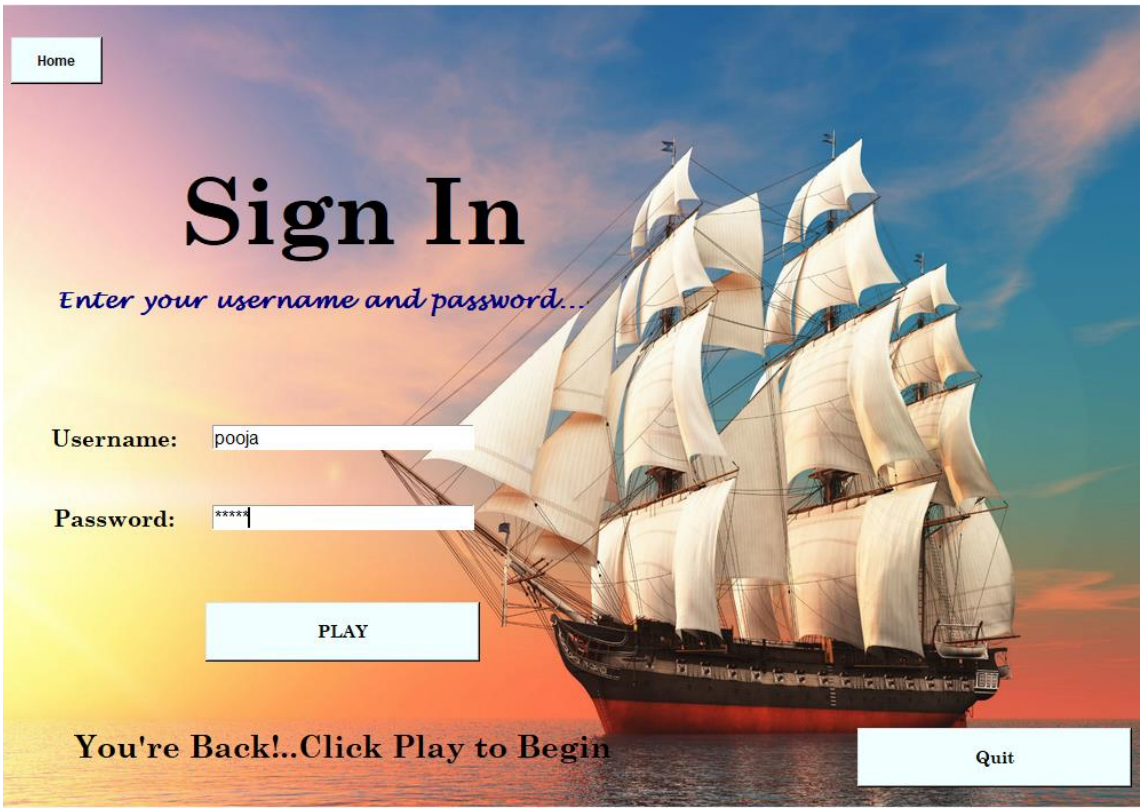
Home Page

Rules

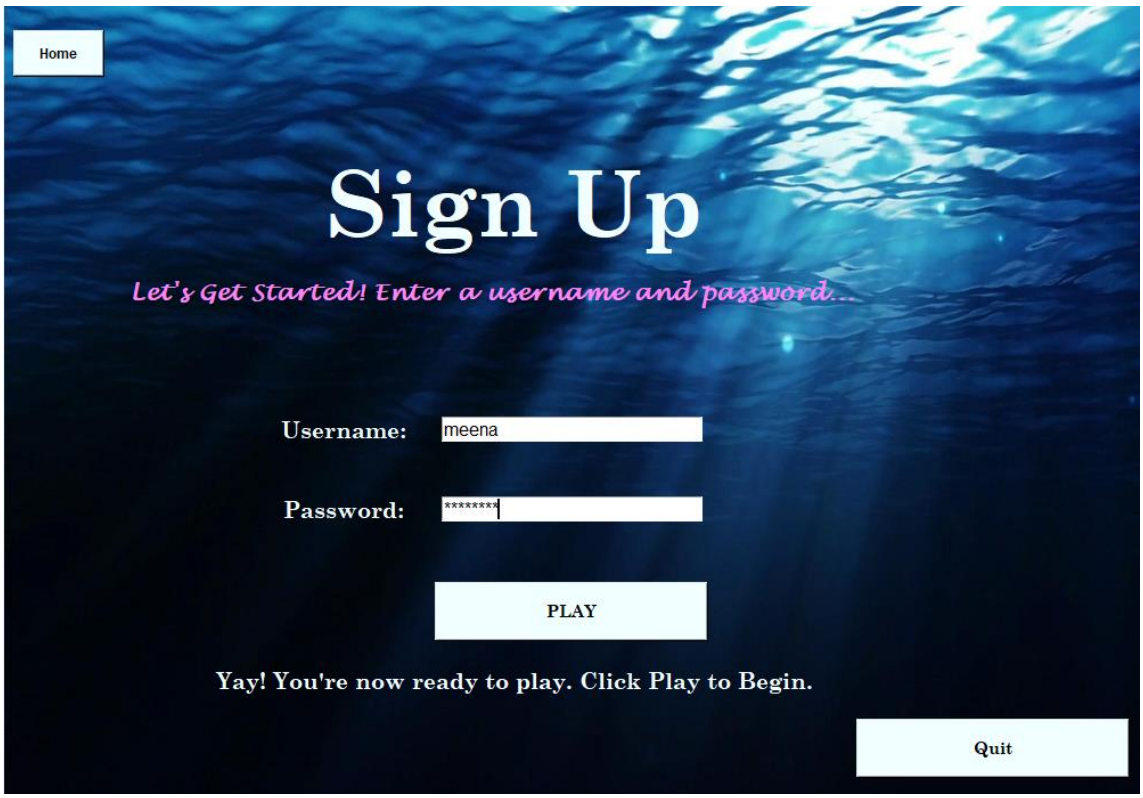


Sign In/ Sign Up

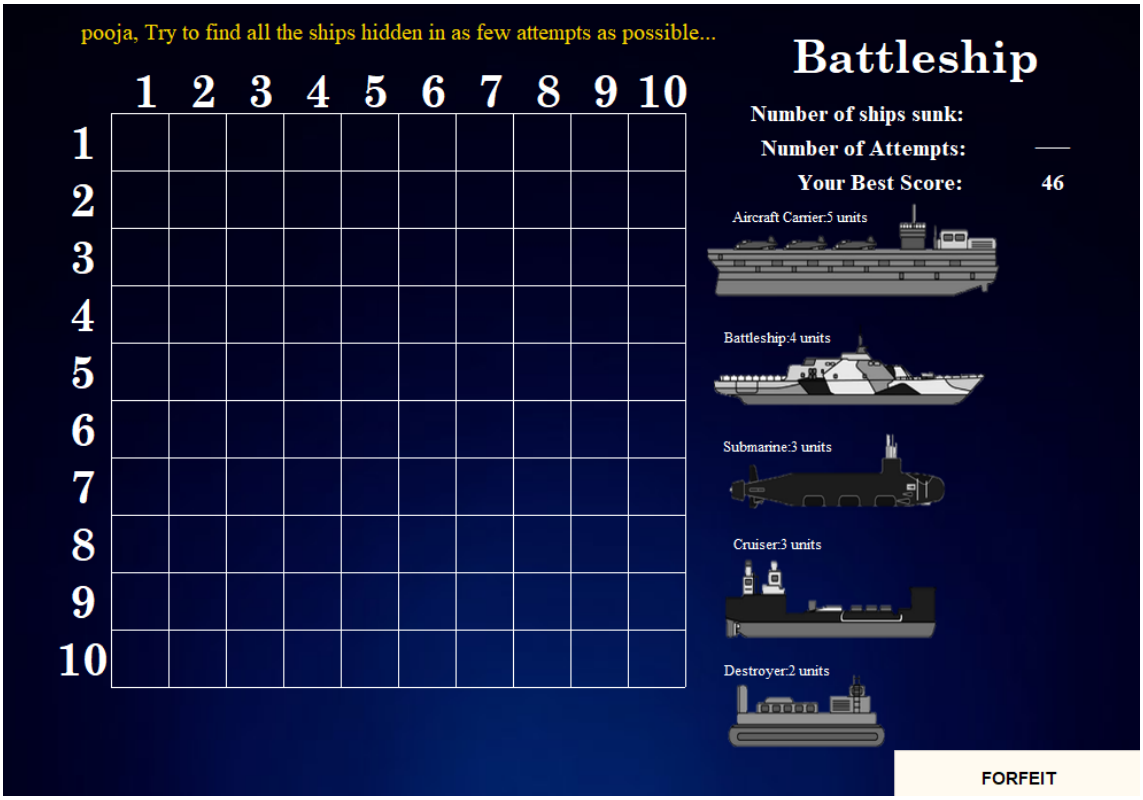
Sign In



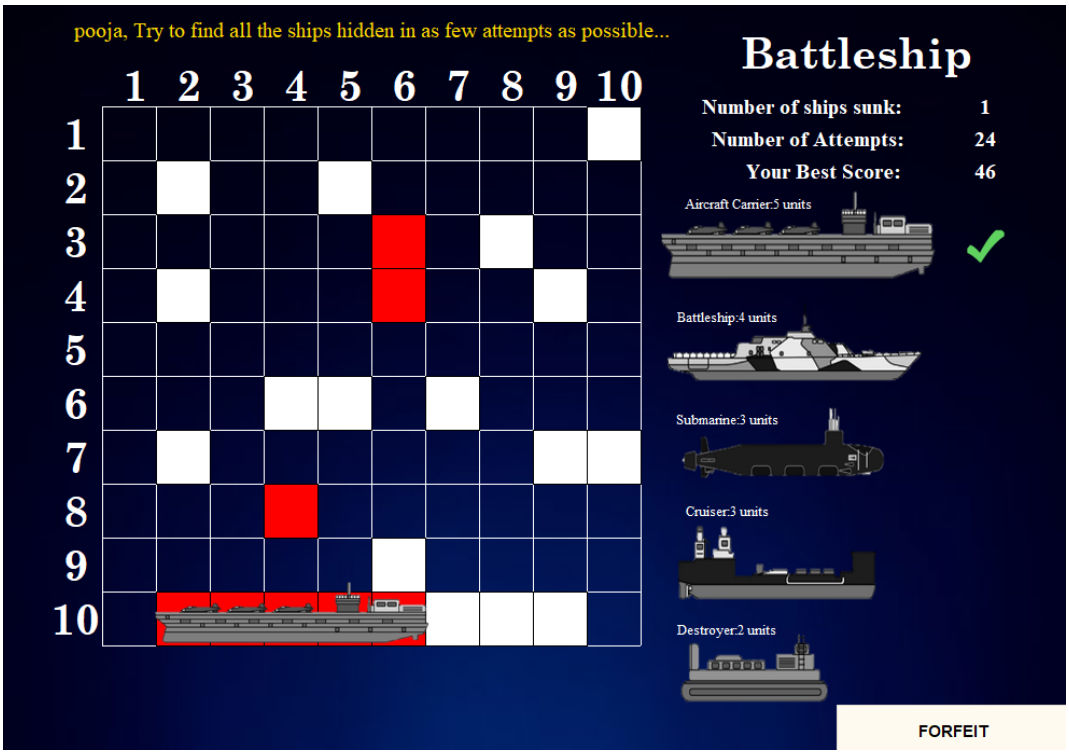
Sign Up



Main game screen



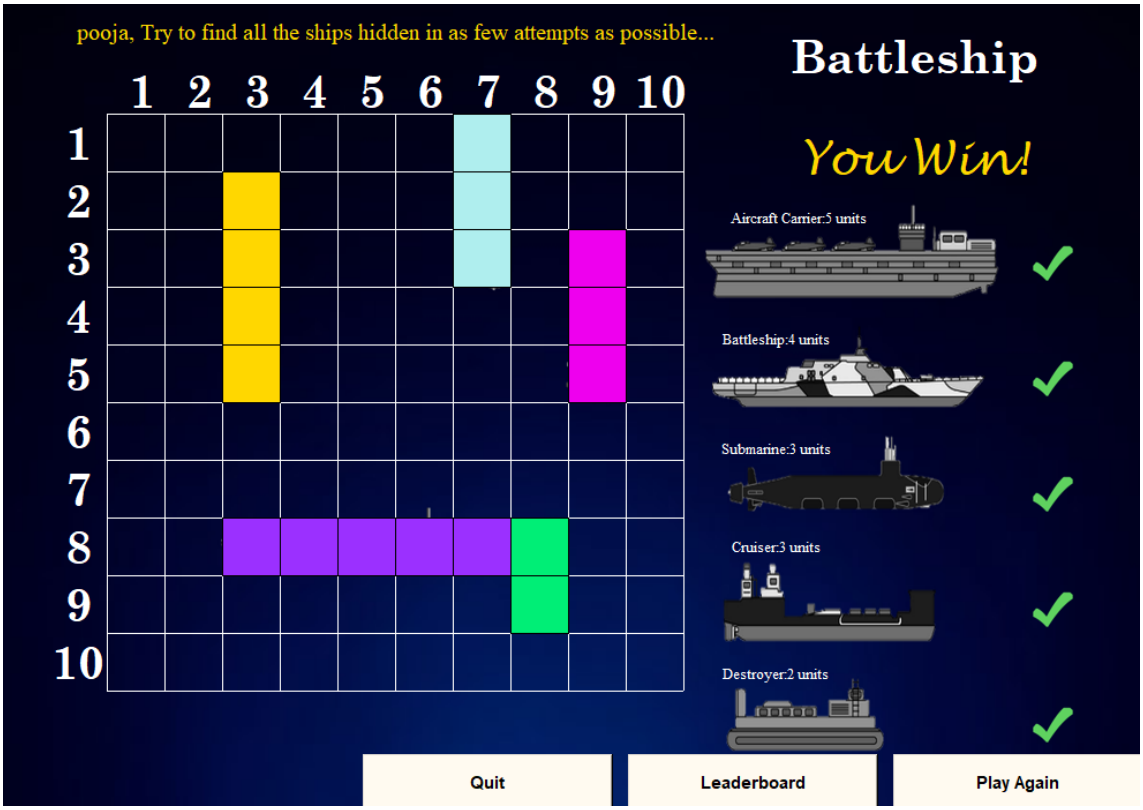
A game in progress



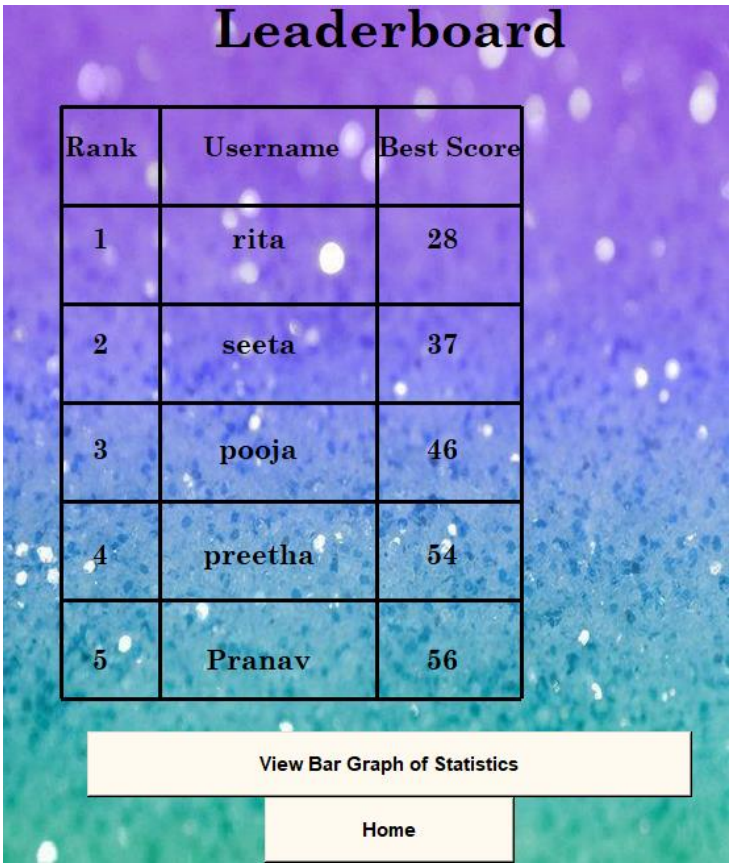
A shot made outside the game board

REPORTS

Winning the game

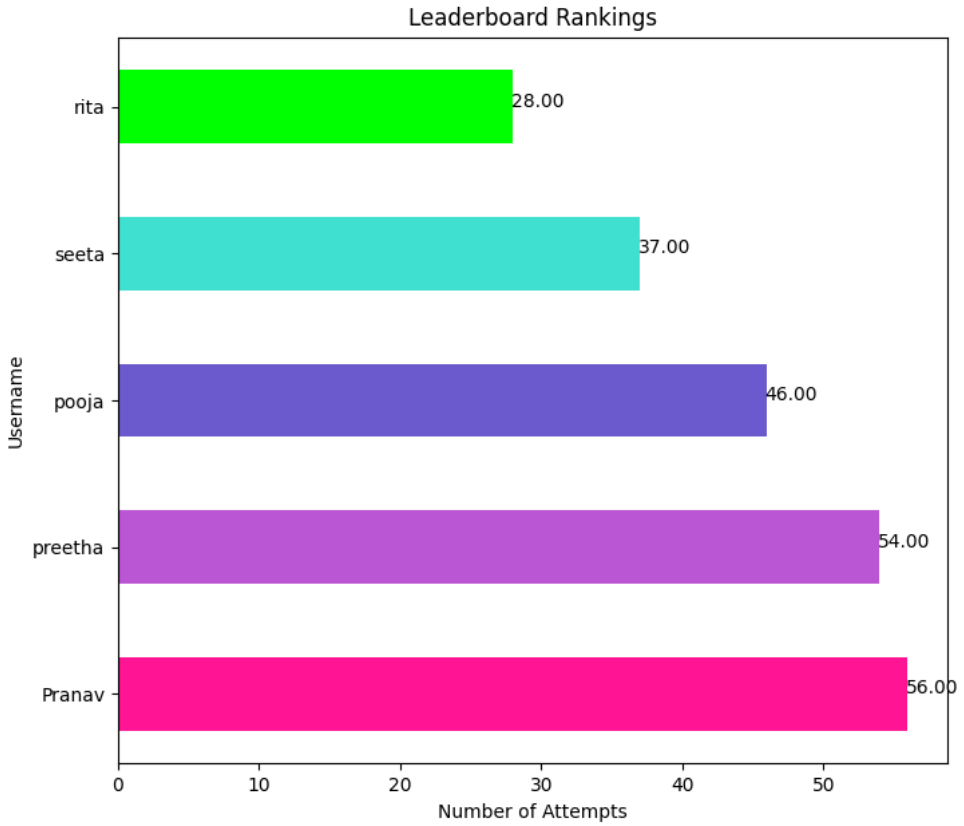


Leaderboard



Rank	Username	Best Score
1	rita	28
2	seeta	37
3	pooja	46
4	preetha	54
5	Pranav	56

Graphical representation of leaderboard



CONCLUSION

Conclusion

The game Battleship successfully runs using Python Programming Language. The salient features of this project are the integration of Python with MySQL, the use of tkinter and Pygame Libraries and PIL (Python Imaging Library) to incorporate computer graphics, and to retrieve image files from a local device. It extensively uses the random module from the Python Standard Library, to provide a unique game board for each play. The Matplotlib is also used to generate a graphical representation of the game's leaderboard.

The mysql.connector is used to establish a Python-MySQL connection to enable the storage and retrieval of a player's username, password and scores.

The tkinter Library is used to set up a Graphical User Interface and to create canvases with different widgets. The Python Imaging Library facilitates the use of custom images on canvases, for the background, and the ships on the playing board.

The Matplotlib Library helps to plot the graphs, depicting the leaderboard, used in this application.

The heart of the program is the random assignment of the five ships on the 10x10 board. It places the ships in a truly random orientation and position every time without overlap, and staying within the available coordinates.

The mouse clicks of the player are captured and are precisely identified to the coordinates they correspond to. The score is updated based on whether the click was a hit or a miss.

Once a ship has been fully identified, an image is placed on the board to mark its position, after systematic calculation of its position and orientation. The Pygame Library also enable the incorporation of sounds, upon the identification of a ship, and to celebrate a win. The process of error handling is taken care of, both for MySQL and screen activity, displaying appropriate messages to the user in each case.

This was an interesting and enlightening attempt to convert a classical board game into a computerized version.

BIBLIOGRAPHY

Bibliography

- https://wiki.python.org/
- https://docs.python.org/
- Computer Science with Python Class XI-Sumita Arora
- Computer Science with Python Class XII-Sumita Arora