A
PROJECT
ON

"FLAPPY BIRD & PLATFORMER"

Submitted to

Shiksha Mandal's
G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR
(AUTONOMOUS)
In the Partial Fulfillment of

B.Com. (Computer Application) Final Year

Submitted by
Chandan Sharma
Sanket Malve

Under the Guidance of

Pravin J. Yadao

Shiksha Mandal's
G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR
(AUTONOMOUS)
2021-2022

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR (AUTONOMOUS)

CERTIFICATE

(2021 - 2022)

This is to certify that Mr. Chandan Dharampal Sharma & Sanket Naresh Malve has completed their project on the topic of "FLAPPY BIRD & PLATFORMER" prescribed by G. S. College of Commerce & Economics, Nagpur (Autonomous) for B.Com. (Computer Application) – Semester-VI.

Date:

Place: Nagpur

Pravin J. Yadao

Project Guide

External Examiner **Internal Examiner**

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude and whole hearted thanks to project guide Prof. Pravin Yadao, Coordinator for his guidance throughout this work. We are very much thankful to him for his constant encouragement, support and kindness.

We are also grateful to our teachers Prof. Rahul Tiwari, Prof. Sushma Gawande, Prof. Preethi Ragnar, Prof. Prajakta Deshpande and Prof. Haresh Naringin for their encouragement, help and support from time to time.

We also wish to express our sincere thanks to Principal Dr. N. Y. Khandi for providing us wide range of opportunities, facilities and inspiration to gather professional knowledge and material without which this project could not have been completed.

Chandan Dharmapala

Sanket Naresh Malve

Date:

Place: Nagpur

DECLARATION

We **Chandan Dharampal Sharma & Sanket Naresh Malve** hereby honestly declare that the work entitled **“FLAPPY BIRD & PLATFORMER”** submitted by us at G. S. College of Commerce & Economics, Nagpur (Autonomous) in partial fulfillment of requirement for the award of B.Com. (Computer Application) degree by Attractant Tukituki Maharaj, Nagpur University, Nagpur has not been submitted elsewhere for the award of any degree, during the academic session 2021-2022.

The project has been developed and completed by us independently under the supervision of the subject teacher and project guide.

Chandan Dharampal Sharma

Sanket Naresh Malve

Date:

Place: Nagpur

INDEX

INTRODUCTION

FLAPPYBIRD

Flappy Bird is an arcade -style game in which the player controls the bird Feby, which moves persistently to the right. The player is tasked with navigating Feby through pairs of pipes that have equally sized gaps placed at random heights. Each successful pass through a pair of pipes awards the player one point. Colliding with a pipe or the ground ends the gameplay. During the game over screen, the player is awarded a bronze medal if they reached ten or more points, a silver medal from twenty points, a gold medal from thirty points, and a platinum medal from forty points. Flappy Bird was originally released on May 24, 2013, with support for the iPhone 5. The game was subsequently updated for iOS 7 in September 2013. Although originally unsuccessful, the game received a massive influx of players after being reviewed by the Swedish YouTuber PewDiePie.] In January 2014, it topped the Free Apps chart in the US and Chinese App Stores, and later that month topped the same section of the UK App Store where it was touted as "the new Angry Birds". It ended January as the most downloaded App on the App Store. The Android version of Flappy Bird was released to the Google Play store on January 30, 2014.

Flappy Bird was created and developed by Nguyen in two to three days. The bird character, Feby, was originally designed in 2012 for a cancelled platform. The gameplay was inspired by the act of bouncing a ping pong ball against a paddle for as long as possible. Initially the game was significantly easier than it became in the final version, however Nguyen said he found this version to be boring and subsequently tightened up the difficulty. He described the business plan of a free download with in-game advertisements as "very common in the Japanese market. The Mechanics of Flappy Bird •

Flappy Bird is based around a simple interaction: press screen to flap wings. But it is clear that Nguyen has spent time working out the exact vertical lift achieved by this single

input; just as he has got the gap between pipes exactly right. Creating Flappy Birds in Scratch A good way to introduce yourself to Programming is to use SCRATCH. Scratch was created by MIIT to get young people (as young as five years old!) interested in the concept of programming and howsequences can get sprites to do clever things. Can Flappy Birds be created in Scratch

PLATFORMER

A platform game (often simplified as platformer or jump 'n' run games) is a video game genre and subgenre of action game in which the core objective is to move the player character between points in a rendered environment. Platform games are characterized by their level design featuring uneven terrain and suspended platforms of varying height that requires use of the player character’s abilities, such as jumping and climbing, to navigate the player's environment and reach their goal. Other acrobatic maneuvers may factor into the gameplay as well, such as swinging from objects such as vines or grappling hooks, jumping off walls, air dashing, gliding through the air, being shot from cannons or bouncing from springboards or trampolines.

A platform game requires the player to maneuver their character across platforms, to reach a goal, while confronting enemies and avoiding obstacles along the way. These games are either presented from the side view, using two-dimensional movement, or in 3D with the camera placed either behind the main character or in isometric perspective. Typical platforming gameplay tends to be very dynamic and challenges a player's reflexes, timing, and dexterity with controls.

While commonly associated with console game, there have been many prominent platform games released for video arcades, as well as for handled game consoles and home computers.

During the peak of platform games' popularity in the late 1980s and early 1990s, platform games were estimated to consist of between a quarter and a third of all console games, but have since been supplanted by first-person shooters. In 2006, the genre experienced a decline in popularity, representing a 2% market share as compared to 15% in 1998; however, the genre still exists in commercial environment, with a number of games selling in the millions of units.Most games of this genre consist of multiple levels of increasing difficulty, that may also be interleaved by boss encounters, where the character has to defeat a particularly dangerous enemy in order to progress. Usually the level order is pre-determined, but some games also allow players to navigate freely throughout the game world, or may feature different paths to take at certain points. Simple logical puzzles to resolve and skill trials to overcome are another common element in the genre.

Objective

Objective of Flappy Bird Game

Goal

It's a simple game of the infinite level type. In this case, you control a bird: pressing a button makes it flap its wings which increases its height. Gravity is also acting on the bird, causing it to fall over time. To complicate things, an unending series of random obstacles enter from the left side of the screen, which must be avoided by flying over, under, or through holes in them. If the bird hits an obstacle, it's "game over".

A simple game: one input (press any button or shake), one character to move up and down, and a series of obstacles that enter from the right and move across the screen and leave on the left.

Some complications can be thrown in: the obstacles are random in size and structure, and the speed at which they move can increase over time.

Use

When the system starts, it presents the user with a choice of using the accelerometer (shake) to flap, or pressing buttons. Choice is shown by filling the left half of the Trellis with yellow, and the right half with blue.

Pressing a yellow button selects the accelerometer while blue selects using buttons.

The game then starts. The bird (yellow pixel) will gradually fall under the effects of the game's gravity. Flapping (by shaking or pressing any button) will cause the bird to move up. Over time green posts will entry from the right and move across the display, disappearing off the left side. The player's job is to maneuver the bird to avoid hitting the posts. As the game progresses, the speed of the posts increases. Initially posts extend from the bottom of the screen. In time they can extend from the top as well. Eventually there will be posts that extend from both top and bottom and the player has to guide the bird between the two pieces of the post.

Entertainment

Our main objective is to provide entertainment, fun and to refresh the mind of the user. It also increases the level of concentration, ability to think and focus of the user.

Design

Let's write this as an object-oriented system. So, the first question is: "What are the things?"

Let's start by having a class that implements the rules of play, initial state, win conditions (or in this case, loose conditions).

Next we have a class to implement the player character: the class. That's the player character in the game. It's what the player has (some) control over.

Finally we need a class to implement the obstacles, which is the class, because they look sort of like posts.

The object is the central thing, it has a single instance and zero or more objects that are on the screen. has a run function that loops until the gamer is over, i.e. until the bird collides with a post.

Each time though the run function's loop, it will look at input and make adjustments to the bird's altitude based on gravity and user activity.

Occasionally the scene will be advanced, moving posts to the right. It's at this point that a collision between the bird and a post is checked for and dealt with. Even more infrequently, a new post will be added to the right side of the screen**.**
Apply previous Skill to put together basis blocks within scratch to create an app style game
Extend Knowledge about variable and how you can affect the difficulty within a game
Some complication can be thrown in the obstacles are random in size and structure and the speed at which they move can increase over time.

Flexible

This software is quite flexible as it allows one to add-on new updations or changes that one find suitable. The updation can be made according to the requirement which makes this project more flexible

Objective of Platformer game

A platform game (often simplified as platformer or jump 'n' run games) video game genre and subgenre action game in which the core objective is to move the player character between points in a rendered environment.

Remember that gameplay trumps animation -- your priority is to make it feel good first, then make it look good. For example, from an artistic point of view you might want your character to crouch before they jump, but for a responsive platformer having a delay before they leave the ground is usually not desirable.

Platformers are curated so much because: -They are satisfying, relaxing and almost everybody likes them. -They have simple controls. -Infinite creativity can be used on them.

Develop a data driven cross platform game engine supporting
Desktop
XBOX One
Android

Develop a game using the engine and produce playable builds for all supported
platforms.

Maintaining such type of project does not cost much and don't require any specialization and can be maintained easily.

This software is compatible for every system. It does not require any specifications. Installing lengthy and bulky software are not necessary for installing this software.

Project category

In this project “Flappy bird and platformer” we use python and for backend mysql language .

Fronted:-

PYTHON:-

➢ Programming Language Used In Project

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical
constructions than other languages.

• Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

• Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

• Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties
at the National Research Institute for Mathematics and Computer Science in the
Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++,
Algol-68, Smalltalk, and Unix shell and other scripting languages.
Python is copyrighted. Like Perl, Python source code is now available under the
GNU General Public License (GPL).
Python is now maintained by a core development team at the institute, although
Guido van Rossum still holds a vital role in directing its progress.
Python Features
Python's features include −

• Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• Easy-to-read − Python code is more clearly defined and visible to the eyes.

• Easy-to-maintain − Python's source code is fairly easy-to-maintained.

• A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

• Interactive Mode − Python has support for an interactive mode which allows
interactive testing and debugging of snippets of code.

• Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
• Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• Databases − Python provides interfaces to all major commercial databases.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable − Python provides a better structure and support for large programs
than shell scripting.
Apart from the above-mentioned features, Python has a big list of good features,
few are listed below −

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking.

• It supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

List of why Python is popular:

• The Python framework also has modules and packages, which facilitates code
reusability.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable − Python provides a better structure and support for large programs
than shell scripting.
Apart from the above-mentioned features, Python has a big list of good features,
few are listed below −

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking.

• It supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
List of why Python is popular:• The Python framework also has modules and packages, which facilitates code reusability.

Backend:-

MySQL=MySQL is an open-source relational database management system. 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 is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for mySQL however, is for the purpose of a web database.

History of mysql

MySQL was created by a Swedish company, MySQL AB, founded by Swedes David Axmark, Allan Larsson and Finland Swede Michael "Monty" Widenius. Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995.

Repository: github.com/mysql/mysql-server

Original author(s): MYSQL LAB

Initial release: 23 May 1995; 26 years ago

Operating system: LINUX, SOLARIES, MAC OS, Win.

MySQL is a system that helps store and manage data efficiently. Database generally stores data in a structured fashion. It is written in C and C++, and it has been tested with a variety of compilers to check for bugs and inconsistencies.

Main features associated with MySQL

Open-Source

MySQL is open-source, which means this software can be downloaded, used and modified by anyone. It is free-to-use and easy-to-understand. The source code of MySQL can be studied, and changed based on the requirements. It uses GPL, i.e. GNU General Public license which defines rules and regulations regarding what can and can’t be done using the application.

Quick and Reliable

MySQL stores data efficiently in the memory ensuring that data is consistent, and not redundant. Hence, data access and manipulation using MySQL is quick.

Scalable

Scalability refers to the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines, and so on. MySQL server was developed to work with large databases.

Data Types

It contains multiple data types such as unsigned integers, signed integers, float (FLOAT), double (DOUBLE), character (CHAR), variable character (VARCHAR), text, blob, date, time, datetime, timestamp, year, and so on.

Character Sets

It supports different character sets, and this includes latin1 (cp1252 character encoding), German, Ujis, other Unicode character sets and so on.

Secure

It provides a secure interface since it has a password system which is flexible, and ensures that it is verified based on the host before accessing the database. The password is encrypted while connecting to the server.

Support for large databases

It comes with support for large databases, which could contain about 40 to 50 million records, 150,000 to 200,000 tables and up to 5,000,000,000 rows.

Client and Utility Programs

MySQL server also comes with many client and utility programs. This includes Command line programs such as 'mysqladmin' and graphical programs such as 'MySQL Workbench'. MySQL client programs are written in a variety of languages. Client library (code encapsulated in a module) can be written in C or C++ and would be available for clients that have C bindings.

SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATIONS

Hardware

To play A Flappy Bird in Real Life you will need a minimum CPU equivalent to an Intel Core 2 Duo Q6867. A Flappy Bird in Real Life system requirements state that you will need at least 512 MB of RAM. The cheapest graphics card you can play it on is an NVIDIA GeForce 7200 GS.

• 4 GB RAM and Above

• 320 GB HARDDISK and Above

• Keyboard • Mouse

• Processor (CPU) with 2 gigahertz (GHz) frequency or Above

• Monitor Resolution 1024 * 768 or Abov

SOFTWARE

Software can be termed as the group of instruction or command used by the computer to accomplish the given task. In today's world generation software is ever ending. It is an evolution of dignified technology.

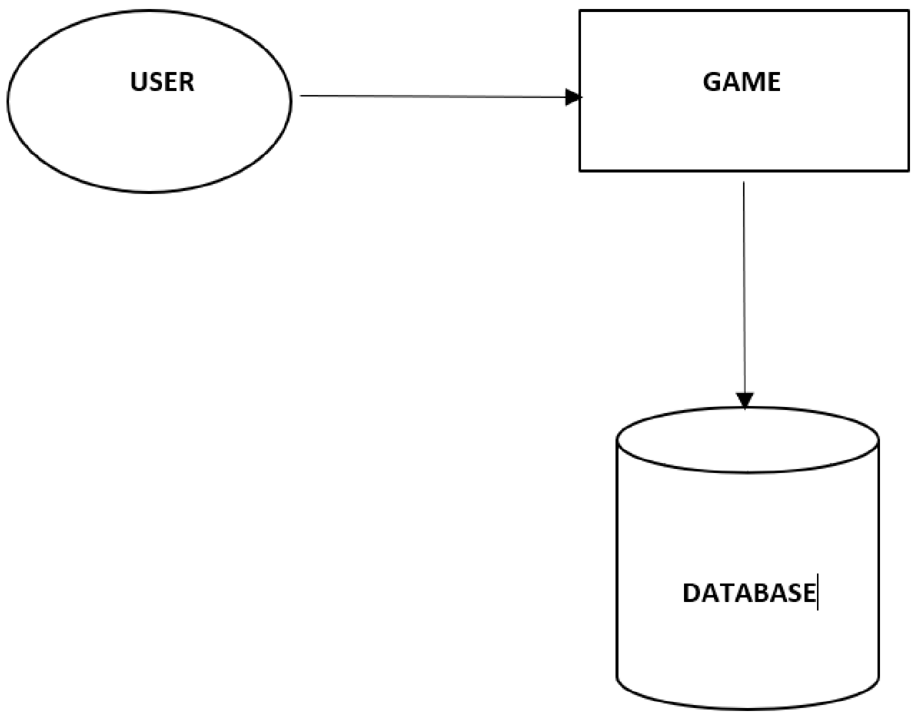
• OPERATING SYSTEM: Windows 10

• LANGUAGES (FRONT END): Python

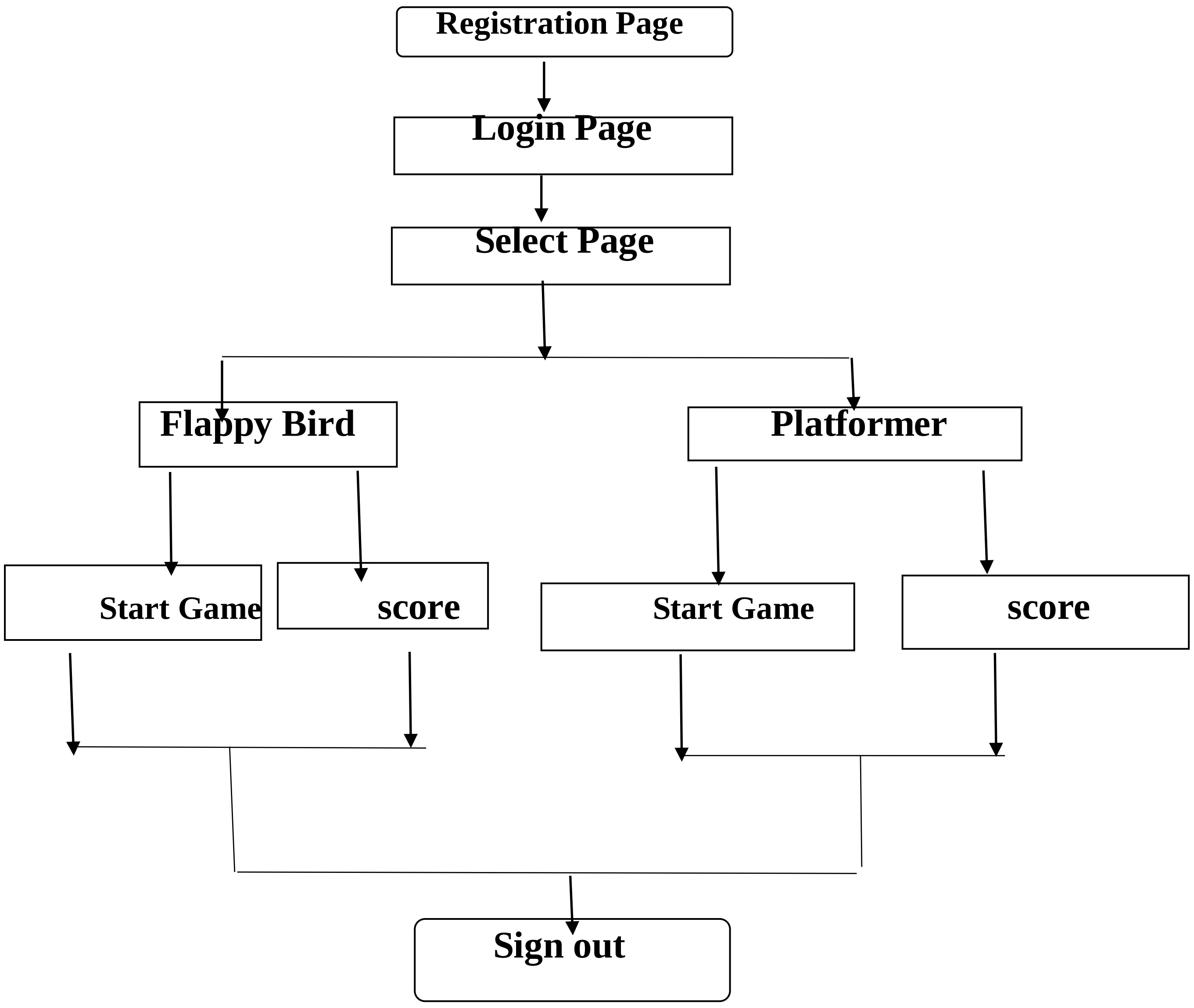
• (BACK END): Mysql

DETAILED SYSTEM ANALYSIS

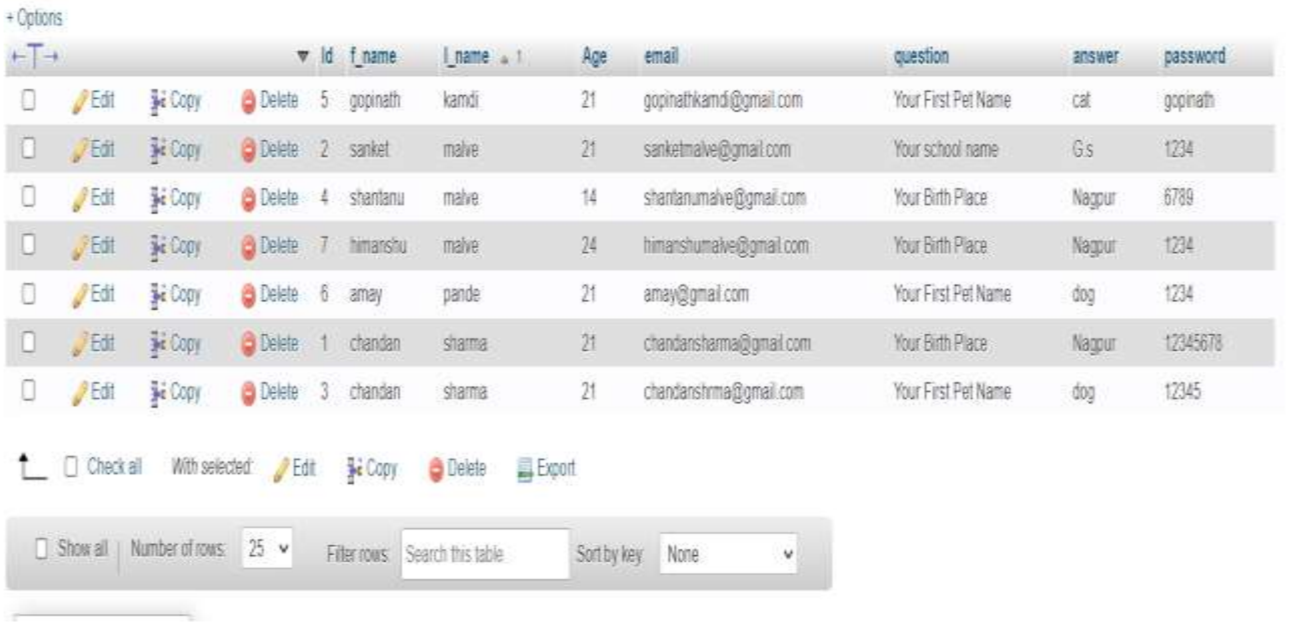
Data Flow Diagram



Structure of Application



Data Tables



+ Options

				Id	f_name	l_name	Age	email	question	answer	password
☐	Edit	Copy	Delete	5	gopinath	kamdi	21	gopinathkamdi@gmail.com	Your First Pet Name	cat	gopinath
☐	Edit	Copy	Delete	2	sanket	malve	21	sanketmalve@gmail.com	Your school name	G.s	1234
☐	Edit	Copy	Delete	4	shantanu	malve	14	shantanumalve@gmail.com	Your Birth Place	Nagpur	6789
☐	Edit	Copy	Delete	7	himanshu	malve	24	himanshumalve@gmail.com	Your Birth Place	Nagpur	1234
☐	Edit	Copy	Delete	6	amay	pande	21	amay@gmail.com	Your First Pet Name	dog	1234
☐	Edit	Copy	Delete	1	chandan	sharma	21	chandansharma@gmail.com	Your Birth Place	Nagpur	12345678
☐	Edit	Copy	Delete	3	chandan	sharma	21	chandanshrma@gmail.com	Your First Pet Name	dog	12345

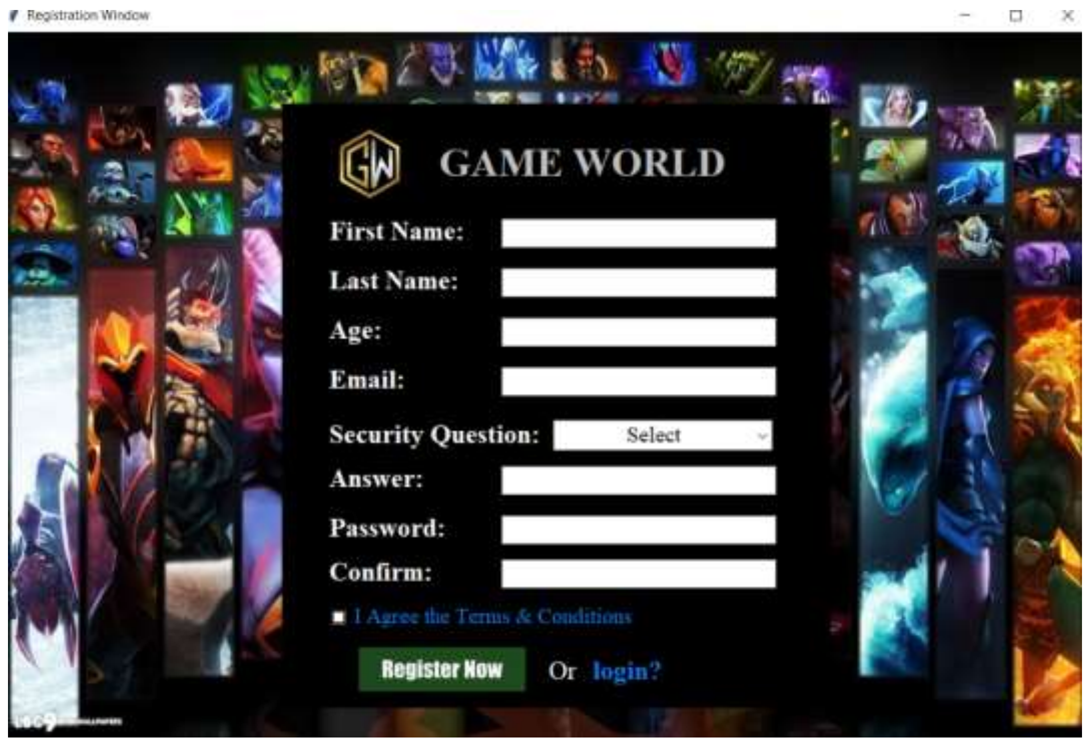
Check all
With selected: Edit Copy Delete Export
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None

Databases play a vital role in game design and development. They store player data, game states, information on performance, and maintain the environments that developer teams have put so much effort into. Without a good database, games can't function properly. Databases support good data access because: Large volumes of data can be stored in one place. Multiple users can read and modify the data at the same time. Databases are searchable and sortable, so the data you need can be found quick and easily. MYSQL is a very good tool to use due to is capability of handling lot of data storage, most especially when you're developing an internet base football manager game, it will be capable of holding your various team data. A database is needed if you have multiple processes (users/servers) modifying the data. Then the database serves to prevent them from overwriting each others changes. You also need a database when your data is larger than memory.

SYSTEM DESIGN

Form Design

Registration form



First Name:

Last Name:

Age:

Email:

Security Question:

Answer:

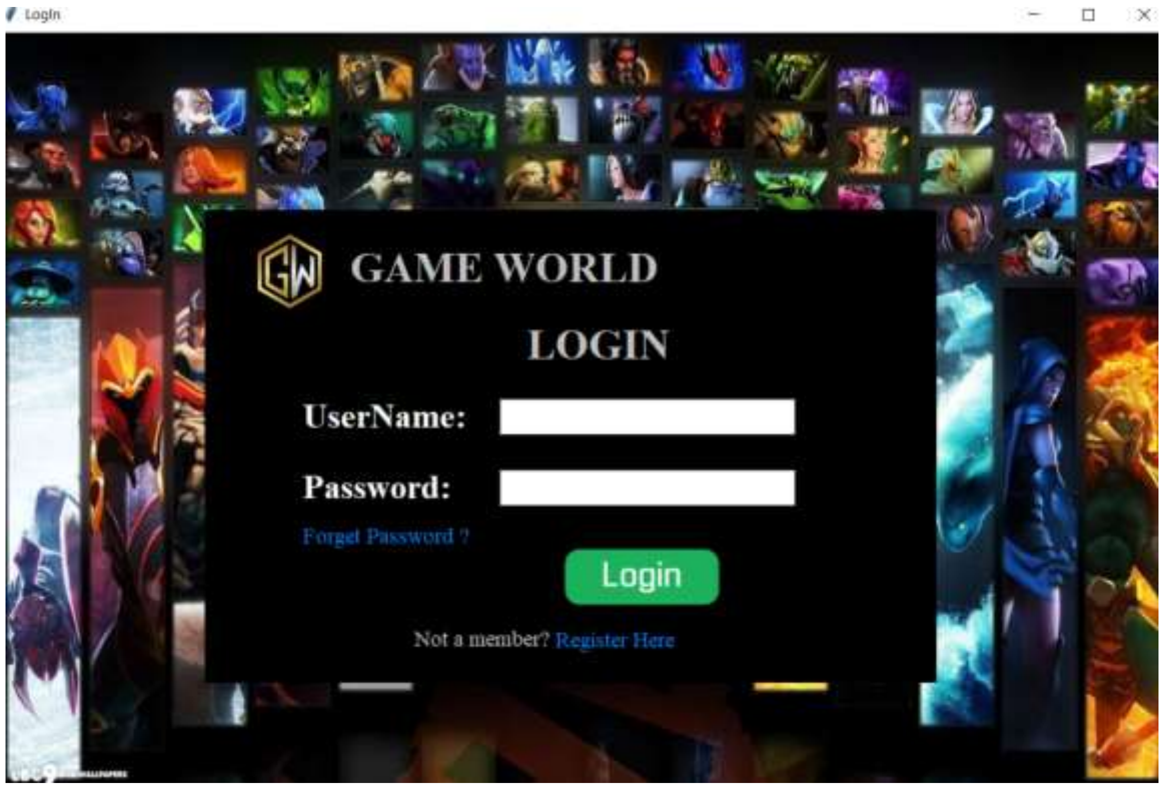
Password:

Confirm:

I Agree the Terms & Conditions

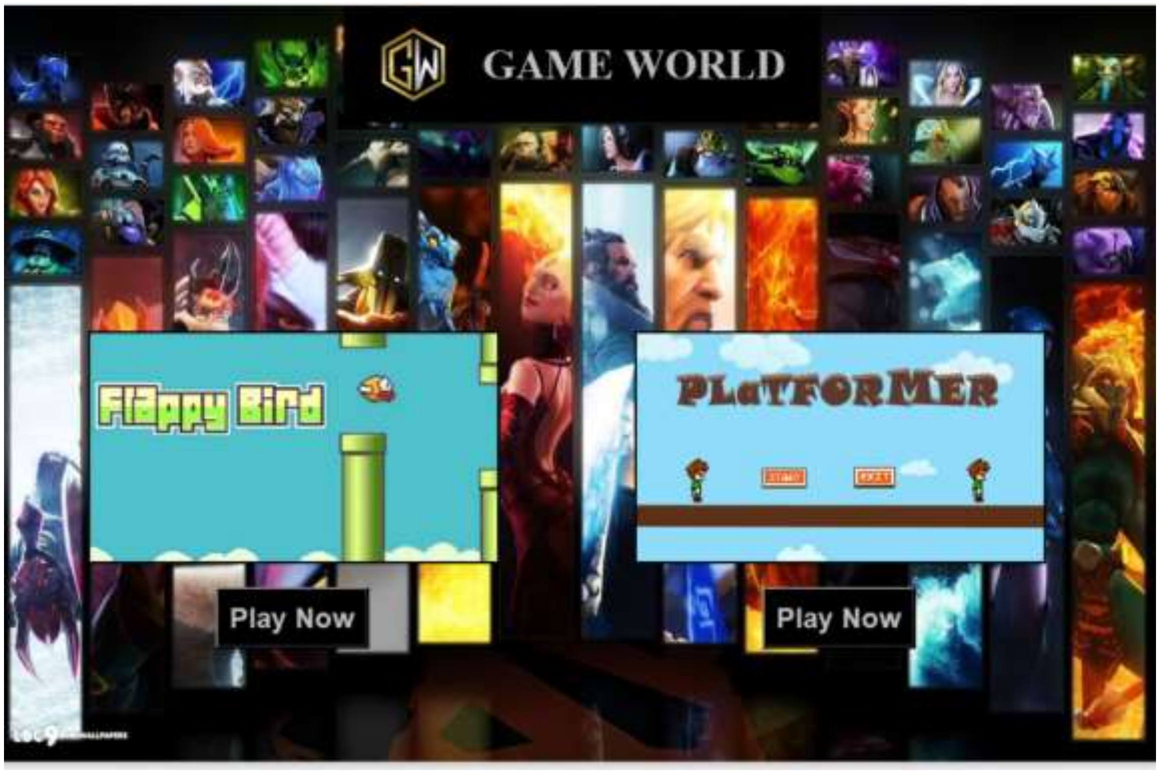
A signup page (also known as a registration page) enables users and organizations to independently register and gain access to your system. It is common to have multiple signup pages depending on the types of people and organizations you want to register. User registration exists to make consumers' life easier when they return to the site. However, some shoppers may be using a site for a specific one-off purchase. Being required to sign up for an account can also deter first-time customers.

Login form



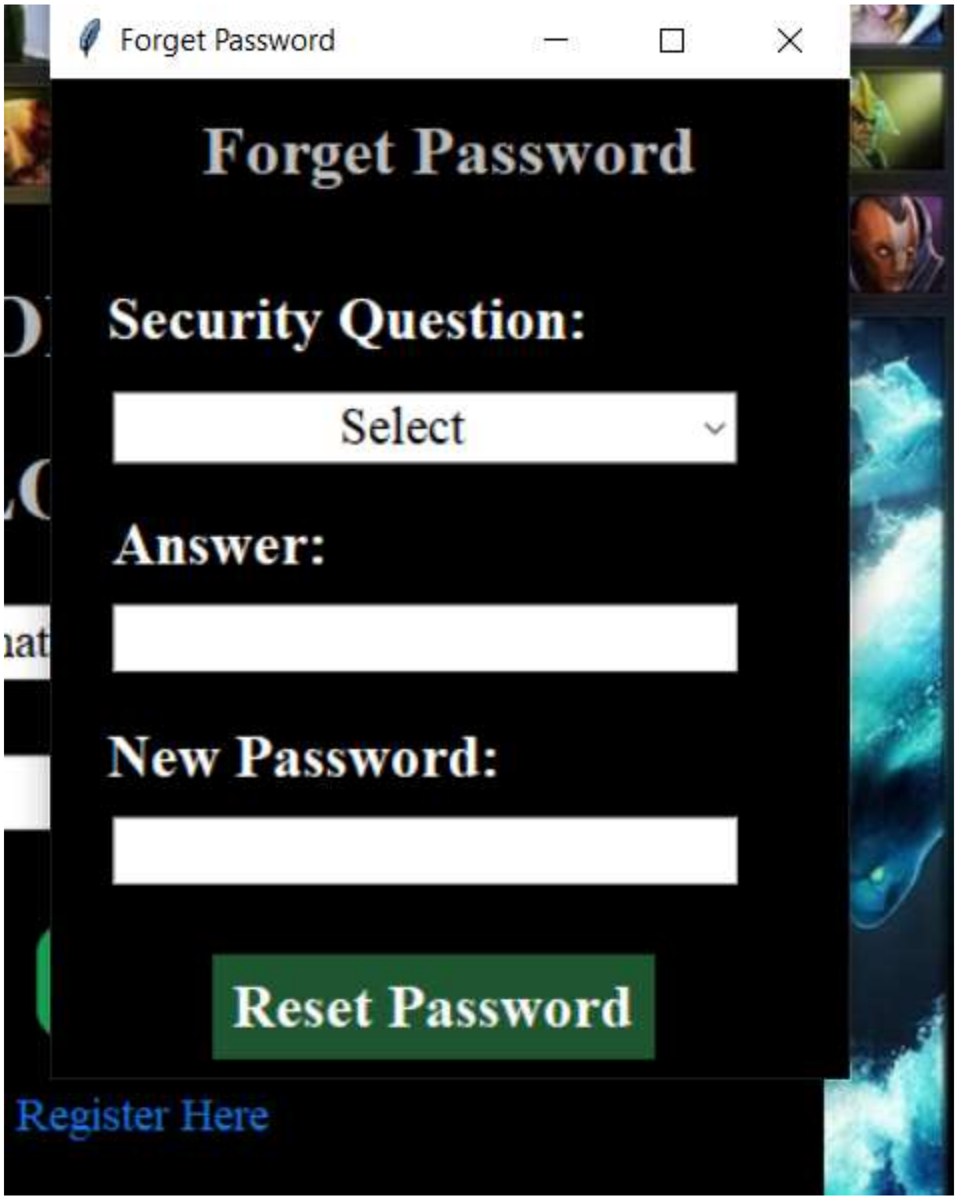
A login page is a web page or an entry page to a website that requires user identification and authentication, regularly performed by entering a username and password combination. Logins may provide access to an entire site or part of a website.

Select form



One of the easiest games of this sort is 'This or That'. This or that questions game is an amazing conversation game where players choose between two items they prefer.

Forget password form



Most websites that require a user to log in provide a link titled forgot password or another similar phrase feature. This link allows users who have forgotten their password to unlock, retrieve, or reset it, usually by answering account secret questions or sending them an e-mail.

Source Code

Source code of registration form

```
from tkinter import*
from tkinter import ttk
from functools import partial
from tkinter import messagebox
from PIL import Image,ImageTk
import pymysql
root = Tk()
class register():
    def __init__(self,root):
        self.root=root
        self.root.title("Registration Window")
        self.root.geometry("1000x650+0+0")
        self.bg=ImageTk.PhotoImage(file="img/Registration/Game.png")
        bg=Label(self.root,image=self.bg).place(x=0,y=0,relheight=1,relwidth=1)
        #registration Form
        frame1=Frame(self.root,bg="black")
        frame1.place(x=260,y=65,width=500,height=550)
        title=Label(frame1,text="GAME WORLD",font=("times new
roman",27,"bold"),bg="black",fg="silver").place(x=140,y=30)
        #Logo
        self.logo=ImageTk.PhotoImage(file="img/Registration/Logo.png")
        logo=Label(frame1,image=self.logo,bg="white").place(x=20,y=5,height=100,width=120)
        #First Name
        self.txt_fname=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_fname.place(x=200,y=105,width=250)
        fname=Label(frame1,text="First Name:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=100)
        #Last Name
        self.txt_lname=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_lname.place(x=200,y=150,width=250)
        lname=Label(frame1,text="Last Name:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=145)
```

```
        #Age
        self.txt_Age=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_Age.place(x=200,y=195,width=250)
        Age=Label(frame1,text="Age:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=190)
        #E-mail
        self.txt_Email=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_Email.place(x=200,y=240,width=250)
        Email=Label(frame1,text="Email:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=235)
        #security Question
        self.txt_Que=ttk.Combobox(frame1,font=("times new
roman",15),state="readonly",justify=CENTER)
        self.txt_Que['values']=("Select","Your First Pet Name","Your Birth Place","Your school
name")
        self.txt_Que.place(x=247,y=288,width=200)
        Que=Label(frame1,text="Security Question:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=285)
        self.txt_Que.current(0)
        #Answer
        self.txt_Ans=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_Ans.place(x=200,y=330,width=250)
        Ans=Label(frame1,text="Answer:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=325)
        #Password
        self.txt_Pass=Entry(frame1,show='*',font=("times new roman",15),bg="white")
        self.txt_Pass.place(x=200,y=375,width=250)
        PASS=Label(frame1,text="Password:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=370)
        #confirm Password
        self.txt_CP=Entry(frame1,show='*',font=("times new roman",15),bg="white")
        self.txt_CP.place(x=200,y=415,width=250)
        CP=Label(frame1,text="Confirm:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=40,y=410)
        #checkbox TC
        self.var_chk=IntVar()
        chk=Checkbutton(frame1,text="I Agree the Terms &
Conditions",variable=self.var_chk,onvalue=1,offvalue=0,bg="black",fg="#1178F2",font=("times
new roman",15)).place(x=40,y=450)
        #Button
        self.btn_img=ImageTk.PhotoImage(file="img/Registration/hehe.png")

btn_reg=Button(frame1,image=self.btn_img,bd=0,cursor="hand2",command=self.register_data,b
g="black").place(x=70,y=495,height=40,width=152)
        #or
        lor=Label(frame1,text="Or",font=("times new
roman",18),bg="black",fg="white").place(x=240,y=500)
        #Sign-in button
        btn_login=Button(frame1,text="login?",font=("times new
roman",18,"bold"),bd=0,command=self.split,cursor="hand2",bg="black",fg="#1178F2").place(x
=275,y=495)
    def clear(self):
        self.txt_fname.delete(0,END)
        self.txt_lname.delete(0,END)
        self.txt_Age.delete(0,END)
        self.txt_Email.delete(0,END)
        self.txt_Ans.delete(0,END)
        self.txt_Que.delete(0,END)
        self.txt_Pass.delete(0,END)
        self.txt_CP.delete(0,END)
    def split(self):
        self.root.destroy()
        import LogIn
    def register_data(self):
        if self.txt_fname.get()=="" or self.txt_lname.get()=="" or self.txt_Email.get()=="" or
self.txt_Age.get()=="":
            messagebox.showerror("Error","All Feilds Are Required",parent=self.root)
        elif self.txt_Pass.get()!=self.txt_CP.get():
            messagebox.showerror("Error","Password & Comfirm Password should be
same",parent=self.root)
```

```
        elif self.var_chk.get()==0:
            messagebox.showerror("Error","Please Agree our Trems and Condition",parent=self.root)
        else:
            try:

con=pymysql.connect(host="localhost",user="root",password="",database="flappydata")
                cur=con.cursor()
                cur.execute("select * from fgdata where email=%s",self.txt_Email.get())




row=cur.fetchone()
                print(row)
                if row!=None:
                    messagebox.showerror("Error","User already exist , Please try with another
email",parent=self.root)

                else:
                    cur.execute("insert into fgdata(f_name,l_name,Age,email,question,answer,password)
values(%s,%s,%s,%s,%s,%s,%s)",
                                (self.txt_fname.get(),
                                self.txt_lname.get(),
                                self.txt_Age.get(),
                                self.txt_Email.get(),
                                self.txt_Que.get(),
                                self.txt_Ans.get(),
                                self.txt_Pass.get()
                                ))

                    con.commit()
                    con.close()
                    self.clear()
                    messagebox.showinfo("Success","Register Successful",parent=self.root)
            except Exception as es:
                messagebox.showerror("Error",f"Error due to:{str(es)}",parent=self.root)
```

```
obj=register(root)
root.mainloop()
```

Source code of login form

```
from tkinter import*
from tkinter import messagebox,ttk
from PIL import Image,ImageTk
import pymysql

root = Tk()


class Login():
    def __init__(self,root):
        self.root=root
        self.root.title("LogIn")
        self.root.geometry("1000x650+0+0")
        self.bg=ImageTk.PhotoImage(file="img/Login/Game.png")
        bg=Label(self.root,image=self.bg).place(x=0,y=0,relheight=1,relwidth=1)
        #login form
        frame1=Frame(self.root,bg="black")
        frame1.place(x=180,y=150,width=620,height=400)
        title=Label(frame1,text="LOGIN",font=("times new
roman",27,"bold"),bg="black",fg="silver").place(x=270,y=90)
        #title
        self.FP=ImageTk.PhotoImage(file="img/Login/Logo.png")
        FP=Label(frame1,image=self.FP,bg="black").place(x=5,y=-15)
        title=Label(frame1,text="GAME WORLD",font=("times new
roman",27,"bold"),bg="black",fg="silver").place(x=120,y=25)
        #Username
        self.txt_Email=Entry(frame1,font=("times new roman",15),bg="white")
        self.txt_Email.place(x=250,y=160,width=250,height=30)
        lname=Label(frame1,text="UserName:",font=("times new
roman",22,"bold"),bg="black",fg="white").place(x=80,y=155)
        #password
        self.txt_Pass=Entry(frame1,show='*',font=("times new roman",15),bg="white")
        self.txt_Pass.place(x=250,y=220,width=250,height=30)
        lname=Label(frame1,text="Password:",font=("times new
roman",22,"bold"),bg="black",fg="white").place(x=80,y=215)
        #forget password
        forget=Button(frame1,text="Forget Password
?",bg="black",command=self.forget_passwordwindow,cursor="hand2",fg="#1178F2",bd=0,font
=("times new roman",14)).place(x=78,y=260)
        #Not a member
        title=Label(frame1,text="Not a member?",font=("times new
roman",14,),bg="black",fg="silver").place(x=175,y=350)
        signup=Button(frame1,text="Register
Here",command=self.split,bg="black",fg="#1178F2",cursor="hand2",bd=0,font=("times new
roman",14,)).place(x=292,y=348)
        #Login button
        self.btn_log=ImageTk.PhotoImage(file="img/Login/log.png")

btn_login=Button(frame1,image=self.btn_log,command=self.login,bd=0,bg="black",cursor="han
d2").place(x=300,y=280,width=140)
    def split(self):
        self.root.destroy()
        import REGISTRATION
    def jump(self):
        self.root.destroy()
        import selectthegame

    def reset(self):
        self.txt_Que.current(0)
        self.txt_newpass.delete(0,END)
        self.txt_Ans.delete(0,END)
        self.txt_Pass.delete(0,END)
        self.txt_Email.delete(0,END)
    def forget_password(self):
        if self.txt_Que.get()=="Select" or self.txt_Ans.get()=="" or self.txt_newpass.get()=="":
            messagebox.showerror("Error","All feilds are required",parent=self.root2)
        else:
            try:

con=pymysql.connect(host="localhost",user="root",password="",database="flappydata")
                cur=con.cursor()
                cur.execute("select * from fgdata where email=%s and question=%s and
answer=%s",(self.txt_Email.get(),self.txt_Que.get(),self.txt_Ans.get()))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please select the Correct security Question / Enter
Answer",parent=self.root2)
                else:
                    cur.execute("update fgdata set password=%s where
email=%s",(self.txt_newpass.get(),self.txt_Email.get()))
                    con.commit()
                    con.close()
                    messagebox.showinfo("Success","Your password is updated",parent=self.root2)
                    self.reset()
                    self.root2.destroy()
                row=cur.fetchone()
            except Exception as es:
                messagebox.showerror("Error",f"Error Due to:{str(es)}",parent=self.root)



    def forget_passwordwindow(self):
        if self.txt_Email.get()=="":
            messagebox.showerror("Error","Please enter email address to reset your
password",parent=self.root)
        else:
             try:
```

```
con=pymysql.connect(host="localhost",user="root",password="",database="flappydata")
                cur=con.cursor()
                cur.execute("select * from fgdata where email=%s",(self.txt_Email.get()))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Please enter a vaild email address to reset your
password",parent=self.root)
                else:
                    con.close()
                    self.root2=Tk()



 self.root2.title
("Forget Password")
                    self.root2.geometry("320x400+490+100")
                    self.root2.config(bg="black")
                    self.root2.focus_force()
                    self.root2.grab_set()
                    t=Label(self.root2,text="Forget Password",font=("times new
roman",20,"bold"),bg="black",fg="silver").place(x=0,y=10,relwidth=1)
                    #security Question
                    self.txt_Que=ttk.Combobox(self.root2,font=("times new
roman",16),state="readonly",justify=CENTER)
                    self.txt_Que['values']=("Select","Your First Pet Name","Your Birth Place","Your
school name")
                    self.txt_Que.place(x=25,y=125,width=250)
                    Que=Label(self.root2,text="Security Question:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=20,y=80)
                    self.txt_Que.current(0)
                    #Answer
                    self.txt_Ans=Entry(self.root2,font=("times new roman",16),bg="white")
                    self.txt_Ans.place(x=25,y=210,width=250)
                    Ans=Label(self.root2,text="Answer:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=22,y=170)
```

```
                    #Newpassword
                    self.txt_newpass=Entry(self.root2,show='*',font=("times new
roman",16),bg="white")
                    self.txt_newpass.place(x=25,y=295,width=250)
                    Newp=Label(self.root2,text="New Password:",font=("times new
roman",18,"bold"),bg="black",fg="white").place(x=20,y=255)
                    #Resetpassword
                    btn_Rpassword=Button(self.root2,text="Reset
Password",command=self.forget_password,font=("times new
roman",18,"bold"),bd=0,cursor="hand2",bg="#1E5631",fg="white").place(x=65,y=350)









except Exception as es:
                messagebox.showerror("Error",f"Error Due to:{str(es)}",parent=self.root)
    def login(self):
        if self.txt_Email.get()==""or self.txt_Pass.get()=="":
            messagebox.showerror("Error","All Field Are Required",parent=self.root)
        else:
            try:

con=pymysql.connect(host="localhost",user="root",password="",database="flappydata")
                cur=con.cursor()
                cur.execute("select * from fgdata where email=%s and
password=%s",(self.txt_Email.get(),self.txt_Pass.get()))
                row=cur.fetchone()
                print(row)

                if row==None:
                    messagebox.showerror("Error","Invalid Username or Password",parent=self.root)
                else:
```

```
                # messagebox.showinfo("Success","Login sucessful",parent=self.root)
                self.root.destroy()
                import selectthegame

            except Exception as es:
                messagebox.showerror("Error",f"Error Due to:{str(es)}",parent=self.root)
obj=Login(root)
root.mainloop()
```

Source code of flappy bird game

```
from tkinter import *
from tkinter import messagebox
from PIL import Image,ImageTk
import pymysql
import pygame
from pygame.locals import *
import random

pygame.init()
screen_width=850
screen_height=650
clock=pygame.time.Clock()
fps=60
screen=pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption('Flappy Bird')
#Define font
font=pygame.font.SysFont('Bauhaus 93',60)
#Define font colours
white=(255,255,255)


#game variables
ground_scroll=0
scroll_speed=4
fly=False
game_over=False
```

```
pipe_gap=180
pipe_freq=1500
last_pipe=pygame.time.get_ticks()-pipe_freq
score=0
pipe_pass=False
#images
bg=pygame.image.load('img/Flappybird/BG1.png')
ground_img=pygame.image.load('img/Flappybird/ground.png')
button_img=pygame.image.load('img/Flappybird/restart.png')

def draw_text(text,font,text_col,x,y):
    img=font.render(text,True,text_col)
    screen.blit(img,(x,y))

def reset_game():
    pipe_group.empty()
    flappy.rect.x=100
    flappy.rect.y=int(screen_height/2)
    score=0
    return score

class Bird(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.images=[]
        self.index=0
        self.counter=0
        for num in range(1,4):
            img=pygame.image.load(f'img/Flappybird/bird{num}.png')
            self.images.append(img)
        self.image=self.images[self.index]
        self.rect=self.image.get_rect()
        self.rect.center=[x,y]
        self.vel=0
        self.clicked=False
    def update(self):
        if fly==True:
            self.vel += 0.5
            if self.vel > 8:
                self.vel = 8
            if self.rect.bottom < 530:
                self.rect.y += int(self.vel)
        if game_over==False:
            #jump
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                self.clicked = True
                self.vel = -10
            if pygame.mouse.get_pressed()[0] == 0:
                self.clicked = False
            # handle the animation
            self.counter += 1
            flap_cooldown = 5
            if self.counter > flap_cooldown:
                self.counter = 0
                self.index += 1
                if self.index >= len(self.images):
                    self.index = 0
            self.image = self.images[self.index]
            # Rotate the img
            self.image = pygame.transform.rotate(self.images[self.index], self.vel * -2)
        else:
            self.image = pygame.transform.rotate(self.images[self.index], -90)
class Pipe(pygame.sprite.Sprite):
    def __init__(self,x,y,position):
        pygame.sprite.Sprite. __init__(self)
        self.image=pygame.image.load('img/Flappybird/pipe.png')
        self.rect=self.image.get_rect()
        if position ==1:
            self.image = pygame.transform.flip(self.image, False, True)
            self.rect.bottomleft = [x, y-int(pipe_gap/2)]
        if position==-1:
            self.rect.topleft = [x, y+int(pipe_gap/2)]
    def update(self):
        self.rect.x-=scroll_speed
        if self.rect.right<0:
            self.kill()

class Button():
    def __init__(self,x,y,image):
        self.image=image
        self.rect=self.image.get_rect()
        self.rect.topleft=(x,y)
    def draw(self):
        action=False
        #get mouse position
        pos=pygame.mouse.get_pos()
        #check mouse is over button
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0]==1:
                action=True

            screen.blit(self.image,(self.rect.x,self.rect.y))
            return action
bird_group=pygame.sprite.Group()
pipe_group=pygame.sprite.Group()
flappy=Bird(100,int(screen_height/2))
bird_group.add(flappy)

# restart button instance
button=Button(screen_width//2-50,screen_height//2-100,button_img)

run=True
while run:
    clock.tick(fps)
    screen.blit(bg,(0,-190))
    bird_group.draw(screen)
    bird_group.update()
    pipe_group.draw(screen)

    #Draw the ground
    screen.blit(ground_img, (ground_scroll, 530))
    #check the score
    if len(pipe_group)>0:
        if bird_group.sprites()[0].rect.left>pipe_group.sprites()[0].rect.left\
            and bird_group.sprites()[0].rect.right<pipe_group.sprites()[0].rect.right\
            and pipe_pass==False:
            pipe_pass=True
        if pipe_pass==True:
            if bird_group.sprites()[0].rect.left > pipe_group.sprites()[0].rect.right :
                score+=1
                pipe_pass=False
   draw_text(str(score),font,white,int(screen_width/2),20)
    #look for collision
    if pygame.sprite.groupcollide(bird_group,pipe_group,False,False)or flappy.rect.top<0:
        game_over=True

    #check if bird is hit the ground
    if flappy.rect.bottom>=530:
        game_over=True
        fly=False

    if game_over==False and fly==True:
        #generate pipe
        time_now=pygame.time.get_ticks()
        if time_now-last_pipe>pipe_freq:
            pipe_height= random.randint(-100,+100)
            btm_pipe = Pipe(screen_width, int(screen_height / 2)+pipe_height,-1)
            top_pipe = Pipe(screen_width, int(screen_height / 2)+pipe_height,1)
            pipe_group.add(btm_pipe)
            pipe_group.add(top_pipe)
            last_pipe=time_now
            ground_scroll -= scroll_speed
        if abs(ground_scroll) > 35:
            ground_scroll = 0
        pipe_group.update()
#check for game over and reset
    if game_over == True:
        if button.draw()== True:
            game_over=False
            score=reset_game()
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            run=False
        if event.type==pygame.MOUSEBUTTONDOWN and fly==False and game_over==False:
            fly=True
pygame.display.update()
pygame.quit()
```

Source code of select game

```
from tkinter import *
from tkinter import messagebox
from PIL import Image,ImageTk
import random

root=Tk()
class level():
    def __init__(self,root):
        self.root=root
        self.root.title("Select the level")
        self.root.geometry("1000x650+0+0")
        self.bg=ImageTk.PhotoImage(file="img/selectthegame/Game.png")
        bg=Label(self.root,image=self.bg).place(x=0,y=0,relheight=1,relwidth=1)

        #Logo
```

```
  title=Label(self.root,text="GAME WORLD",font=("times new
roman",27,"bold"),bg="black",fg="silver").place(x=390,y=1,height=100,width=320)
        self.logo=ImageTk.PhotoImage(file="img/selectthegame/Logo.png")
        logo=Label(self.root,image=self.logo,bg="white").place(x=300,y=1,height=100,width=120)
        #Flappybird
        self.flappy=ImageTk.PhotoImage(file="img/selectthegame/frontimg.png")
        flappy=Label(self.root,image=self.flappy,bg="black").place(x=80,y=280)
        btn_1=Button(text="Play Now",font=("Bauhaus
93",18,"bold"),bd=3,command=self.Game1,cursor="hand2",bg="black",fg="silver").place(x=190
,y=500)
        #Platformer
        self.Platf=ImageTk.PhotoImage(file="img/selectthegame/front.png")
        Platf=Label(self.root,image=self.Platf,bg="black").place(x=550,y=280)
        btn_2=Button(text="Play Now",font=("Bauhaus
93",18,"bold"),bd=3,command=self.Game2,cursor="hand2",bg="black",fg="silver").place(x=660
,y=500)
    #game button
    def Game1(self):
      self.root.destroy()
      import Flappybird
    def Game2(self):
      self.root.destroy()
      import Platformer
obj = level(root)
root.mainloop()
```

Source code of platformer game

```
import pygame
from pygame.locals import *
from pygame import mixer
import pickle
from os import path
pygame.mixer.pre_init(44100,-16,2,512)
mixer.init()
pygame.init()
clock=pygame.time.Clock()
fps=50
screen_width=990
screen_height=650

screen=pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption('Platformer')

#define font
font=pygame.font.SysFont('Bauhaus 93',70)
font_score=pygame.font.SysFont('Bauhaus 93',30)
#Game variables
tile_size=30
game_over=0
main_menu=True
level=1
max_levels=7
score=0
#define color
white=(255,255,255)
blue=(0,0,255)
#load Background Image
bg_img=pygame.image.load('img/Platformer/sky.png')
restart_img=pygame.image.load('img/Platformer/restart_btn.png')
start_img=pygame.image.load('img/Platformer/start_btn.png')
exit_img=pygame.image.load('img/Platformer/exit_btn.png')
```

```
#Load sound
pygame.mixer.music.load('img/Platformer/music.wav')
pygame.mixer.music.play(-1,0.0,5000)
coin_fx=pygame.mixer.Sound('img/Platformer/coin.wav')
coin_fx.set_volume(0.5)
jump_fx=pygame.mixer.Sound('img/Platformer/jump.wav')
jump_fx.set_volume(0.5)
gameover_fx=pygame.mixer.Sound('img/Platformer/game_over.wav')
gameover_fx.set_volume(0.5)




def draw_text(text,font,text_col,x,y):
    img=font.render(text,True,text_col)
    screen.blit(img,(x,y))

def reset_level(level):
    player.reset(100,screen_height-130)
    blob_group.empty()
    lava_group.empty()
    exit_group.empty()
    if path.exists(f'img/Platformer/level{level}_data'):
        pickle_in=open(f'img/Platformer/level{level}_data','rb')
        world_data=pickle.load(pickle_in)
    world=World(world_data)
    return world
class Button:
    def __init__(self,x,y,image):
        self.image=image
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.clicked=False
    def draw(self):
        action=False
        pos=pygame.mouse.get_pos()
        #hek mouseover and liked onditions
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0]==1 and self.clicked==False:
                action=True
                self.clicked=True
        if pygame.mouse.get_pressed()[0]==0:
            self.clicked=False
        screen.blit(self.image,self.rect)
        return action
        for tile in self.tile_list:
            screen.blit(tile[0],tile[1])
class Player():
    def __init__(self,x,y):
        self.reset(x, y)

    def update(self,game_over):
        dx=0
        dy=0
        walk=4

        if game_over==0:
            #get keypresses
            key=pygame.key.get_pressed()
            if key[pygame.K_SPACE] and self.jumped==False and self.in_Air==False:
                jump_fx.play()
                self.vel_y=-14
                self.jumped=True
            if key[pygame.K_SPACE]==False:
                self.jumped=False
            if key[pygame.K_LEFT]:
                dx-=5
                self.counter+=1
                self.Directioin=-1
```

```
if key[pygame.K_RIGHT]:
    dx+=5
    self.counter+=1
    self.Directioin=1
if key[pygame.K_LEFT]==False and key[pygame.K_RIGHT]==False:
    self.counter=0
    self.index=0
    if self.Directioin==1:
        self.image=self.image_right[self.index]
    if self.Directioin==-1:
        self.image=self.image_left[self.index]

#Animation
if self.counter>walk:
    self.counter=0
    self.index+=1
    if self.index>=len(self.image_right):
        self.index=0
    if self.Directioin==1:
        self.image=self.image_right[self.index]
    if self.Directioin==-1:
        self.image=self.image_left[self.index]

#add gravity
self.vel_y+=1
if self.vel_y>10:
    self.vel_y=10
dy+=self.vel_y

#check for collision
self.in_Air=True
for tile in world.tile_list:
    #Check for collision in x direction
    if tile[1].colliderect(self.rect.x+dx,self.rect.y,self.width,self.height):
        dx=0
```

```
            #check for collision in y direction
            if tile[1].colliderect(self.rect.x,self.rect.y+dy,self.width,self.height):
                #check if below the ground i.e jumping
                if self.vel_y<0:
                    dy=tile[1].bottom-self.rect.top
                    self.vel_y=0
                #check if above the ground i.e falling
                elif self.vel_y>=0:
                    dy=tile[1].top-self.rect.bottom
                    self.vel_y=0
                    self.in_Air=False
        #check for collision with enemies
        if pygame.sprite.spritecollide(self,blob_group,False):
            game_over=-1
            gameover_fx.play()
        #check for collision with lava
        if pygame.sprite.spritecollide(self,lava_group,False):
            game_over=-1
            gameover_fx.play()
        #check for collision with exit
        if pygame.sprite.spritecollide(self,exit_group,False):
            game_over=1

        #update player cordinates
        self.rect.x+=dx
        self.rect.y+=dy
    elif game_over==-1:
        self.image=self.dead_image
        draw_text('GAME OVER!', font,blue,(screen_width//2)-200,screen_height//2)
        if self.rect.y>100:
            self.rect.y-=5
    #draw player on screen
    screen.blit(self.image,self.rect)
    #pygame.draw.rect(screen,(225,225,225),self.rect,2)
```

```
        return game_over

    def reset(self,x,y):
        self.image_right=[]
        self.image_left=[]
        self.index=0
        self.counter=0
        for num in range(1,5):
            img_right=pygame.image.load(f'img/Platformer/guy{num}.png')
            img_right=pygame.transform.scale(img_right,(30,40))
            img_left=pygame.transform.flip(img_right,True,False)
            self.image_left.append(img_left)
            self.image_right.append(img_right)
        self.dead_image=pygame.image.load('img/Platformer/ghost.png')
        self.image=self.image_right[self.index]
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.width=self.image.get_width()
        self.height=self.image.get_height()
        self.vel_y=0
        self.jumped=False
        self.Directioin=0
        self.in_Air=True

class World():
    def __init__(self,data):
        self.tile_list=[]

        dirt_img=pygame.image.load('img/Platformer/Dirt.png')
        grass_img=pygame.image.load('img/Platformer/grass.png')



        row_count=0
        for row in data:
```

```
col_count=0
for tile in  row:
    if tile ==1:
        img=pygame.transform.scale(dirt_img,(tile_size,tile_size))
        img_rect=img.get_rect()
        img_rect.x=col_count * tile_size
        img_rect.y=row_count * tile_size
        tile=(img,img_rect)
        self.tile_list.append(tile)
    if tile ==2:
        img=pygame.transform.scale(grass_img,(tile_size,tile_size))
        img_rect=img.get_rect()
        img_rect.x=col_count * tile_size
        img_rect.y=row_count * tile_size
        tile=(img,img_rect)
        self.tile_list.append(tile)
    if tile==3:
        blob=Enemy(col_count * tile_size,row_count * tile_size-5)
        blob_group.add(blob)
    if tile==4:
        platform=Platform(col_count * tile_size,row_count *tile_size)
        platform_group.add(platform)
    if tile==5:
        platform=Platform(col_count * tile_size,row_count *tile_size)
        platform_group.add(platform)
    if tile==6:
        lava=Lava(col_count*tile_size,row_count*tile_size+(tile_size//2))
        lava_group.add(lava)
    if tile==7:
        coin=Coin(col_count*tile_size+(tile_size//2),row_count*tile_size+(tile_size//2))
        coin_group.add(coin)
    if tile==8:
        exit=Exit(col_count*tile_size, row_count*tile_size-(tile_size//2))
        exit_group.add(exit)
    col_count+=1
```

```
            row_count+=1
    def draw(self):
        for tile in self.tile_list:
            screen.blit(tile[0],tile[1])
            #pygame.draw.rect(screen,(225,225,225),tile[1],2)
class Enemy(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        self.image=pygame.image.load('img/Platformer/blob.png')
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.move_direction=1
        self.move_counter=0
    #move the blob left and right
    def update(self):
        self.rect.x+=self.move_direction
        self.move_counter+=1
        if self.move_counter>50:
            self.move_direction*=-1
            self.move_counter*=-1

class Platform(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        img=pygame.image.load('img/Platformer/platform.png')
        self.image=pygame.transform.scale(img,(tile_size,tile_size//2))
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y


class Lava(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        img=pygame.image.load('img/Platformer/lava.png')
        self.image=pygame.transform.scale(img,(tile_size,tile_size//2))

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
class Coin(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        img=pygame.image.load('img/Platformer/coin.png')
        self.image=pygame.transform.scale(img,(tile_size//2,tile_size//2))
        self.rect=self.image.get_rect()
        self.rect.center=(x,y)
class Exit(pygame.sprite.Sprite):
    def __init__(self,x,y):
        pygame.sprite.Sprite.__init__(self)
        img=pygame.image.load('img/Platformer/exit.png')
        self.image=pygame.transform.scale(img,(tile_size,int(tile_size * 1.5)))
        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y


player=Player(100,screen_height-130)
blob_group=pygame.sprite.Group()
platform_group=pygame.sprite.Group()
lava_group=pygame.sprite.Group()
coin_group=pygame.sprite.Group()
exit_group=pygame.sprite.Group()
#DUMMY COIN
score_coin=Coin(tile_size//2,tile_size//2)
coin_group.add(score_coin)
#loading world data and creating it
if path.exists(f'img/Platformer/level{level}_data'):
    pickle_in=open(f'img/Platformer/level{level}_data','rb')
    world_data=pickle.load(pickle_in)
world=World(world_data)
```

```
#buttons
restart_button=Button(screen_width//2-50,screen_height//2+100,restart_img)
start_button=Button(screen_width//2-350,screen_height//2,start_img)
exit_button=Button(screen_width//2+150,screen_height//2,exit_img)

#def draw_grid():
 #   for line in range(0,34):
  #      pygame.draw.line(screen,(255,255,255),(0,line*tile_size),(screen_width,line*tile_size))
   #     pygame.draw.line(screen,(255,255,255),(line*tile_size,0),(line*tile_size,screen_height))
run=True
while run:
    clock.tick(fps)
    screen.blit(bg_img,(0,0))
    #screen.blit(sun_img,(100,100))
    #draw_grid()

    if main_menu==True:
        if exit_button.draw():
            run=False
        if start_button.draw():
            main_menu=False

    else:

        world.draw()
        #draw_grid()
        if game_over==0:
            blob_group.update()
            #update score
            if pygame.sprite.spritecollide(player,coin_group,True):
                score+=1
                coin_fx.play()
            draw_text('X '+str(score),font_score,white,tile_size-5,0)
        blob_group.draw(screen)
        platform_group.draw(screen)
        lava_group.draw(screen)
        coin_group.draw(screen)
        exit_group.draw(screen)
        game_over=player.update(game_over)
        #if player has died
        if game_over==-1:



  if
restart_button.draw():
                world_data=[]
                world=reset_level(level)
                game_over=0
                score=0
        #if player is won
        if game_over==1:
            #load the next level
            level+=1
            if level<=max_levels:
                #reset level
                world_data=[]
                world=reset_level(level)
                game_over=0
            else:
                draw_text('YOU WIN!',font,blue,(screen_width//2)-140,screen_height//2)
                if restart_button.draw():
                    level=1
                    world_data=[]
                    world=reset_level(level)
                    game_over=0
                    score=0
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            run=False
    pygame.display.update()

pygame.quit()
```

Source code of Level editor

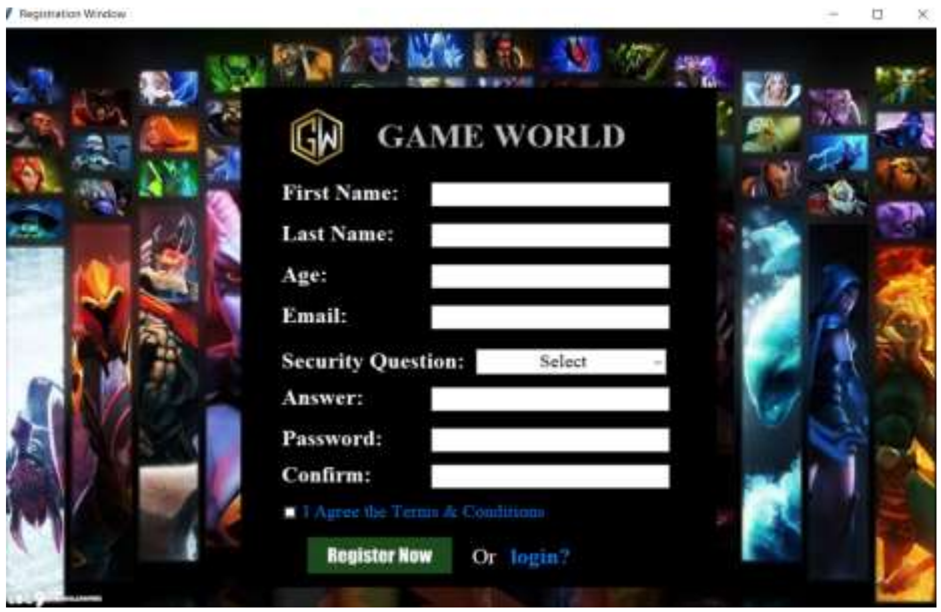
```
import pygame
import pickle
from os import path


pygame.init()

clock = pygame.time.Clock()
fps = 60

#game window
tile_size = 30
cols = 34
margin = 0
screen_width = tile_size * cols
screen_height = 700#(tile_size * cols) + margin

screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Level Editor')


#load images
bg_img = pygame.image.load('img/Platformer/sky.png')
bg_img = pygame.transform.scale(bg_img, (screen_width, screen_height - margin))
dirt_img = pygame.image.load('img/Platformer/dirt.png')
grass_img = pygame.image.load('img/Platformer/grass.png')
blob_img = pygame.image.load('img/Platformer/blob.png')
platform_x_img = pygame.image.load('img/Platformer/platform_x.png')
platform_y_img = pygame.image.load('img/Platformer/platform_y.png')
```

```
lava_img = pygame.image.load('img/Platformer/lava.png')
coin_img = pygame.image.load('img/Platformer/coin.png')
exit_img = pygame.image.load('img/Platformer/exit.png')
save_img = pygame.image.load('img/Platformer/save_btn.png')
load_img = pygame.image.load('img/Platformer/load_btn.png')


#define game variables
clicked = False
level = 1

#define colours
white = (255, 255, 255)
green = (144, 201, 120)

font = pygame.font.SysFont('Futura', 24)

#create empty tile list
world_data = []
for row in range(34):
    r = [0] * 34
    world_data.append(r)

#create boundary
for tile in range(0, 34):
    world_data[21][tile] =1
    world_data[0][tile] = 1
    world_data[tile][0] = 1
    world_data[tile][32] = 1

#function for outputting text onto the screen
def draw_text(text, font, text_col, x, y):
    img = font.render(text, True, text_col)
    screen.blit(img, (x, y))

def draw_grid():
    for c in range(35):
        #vertical lines
        pygame.draw.line(screen, white, (c * tile_size, 0), (c * tile_size, screen_height - margin))
        #horizontal lines
        pygame.draw.line(screen, white, (0, c * tile_size), (screen_width, c * tile_size))


def draw_world():
    for row in range(34):
        for col in range(34):
            if world_data[row][col] > 0:
                if world_data[row][col] == 1:
                    #dirt blocks
                    img = pygame.transform.scale(dirt_img, (tile_size, tile_size))
                    screen.blit(img, (col * tile_size, row * tile_size))
                if world_data[row][col] == 2:
                    #grass blocks
                    img = pygame.transform.scale(grass_img, (tile_size, tile_size))
                    screen.blit(img, (col * tile_size, row * tile_size))
                if world_data[row][col] == 3:
                    #enemy blocks
                    img = pygame.transform.scale(blob_img, (tile_size, int(tile_size * 0.75)))
                    screen.blit(img, (col * tile_size, row * tile_size + (tile_size * 0.25)))
                if world_data[row][col] == 4:
                    #horizontally moving platform
                    img = pygame.transform.scale(platform_x_img, (tile_size, tile_size // 2))
                    screen.blit(img, (col * tile_size, row * tile_size))
                if world_data[row][col] == 5:
                    #vertically moving platform
                    img = pygame.transform.scale(platform_y_img, (tile_size, tile_size // 2))
                    screen.blit(img, (col * tile_size, row * tile_size))
                if world_data[row][col] == 6:
                    #lava
                    img = pygame.transform.scale(lava_img, (tile_size, tile_size // 2))
                    screen.blit(img, (col * tile_size, row * tile_size + (tile_size // 2)))
                if world_data[row][col] == 7:
                    #coin
                    img = pygame.transform.scale(coin_img, (tile_size // 2, tile_size // 2))
                    screen.blit(img, (col * tile_size + (tile_size // 4), row * tile_size + (tile_size // 4)))
                if world_data[row][col] == 8:
                    #exit
                    img = pygame.transform.scale(exit_img, (tile_size, int(tile_size * 1.5)))
                    screen.blit(img, (col * tile_size, row * tile_size - (tile_size // 2)))



class Button():
    def __init__(self, x, y, image):
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)
        self.clicked = False

    def draw(self):
        action = False

        #get mouse position
        pos = pygame.mouse.get_pos()

        #check mouseover and clicked conditions
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                action = True
                self.clicked = True

        if pygame.mouse.get_pressed()[0] == 0:
            self.clicked = False

        #draw button
        screen.blit(self.image, (self.rect.x, self.rect.y))

        return action
```

```
#create load and save buttons
save_button = Button(screen_width // 2 - 150, screen_height - 50, save_img)
load_button = Button(screen_width // 2 + 150, screen_height - 50, load_img)

#main game loop
run = True
while run:

    clock.tick(fps)

    #draw background
    screen.fill(green)
    screen.blit(bg_img, (0, 0))
    #screen.blit(sun_img, (tile_size * 2, tile_size * 2))

    #load and save level
    if save_button.draw():
        #save level data
        pickle_out = open(f'img/Platformer/level{level}_data', 'wb')
        pickle.dump(world_data, pickle_out)
        pickle_out.close()
    if load_button.draw():
        #load in level data
        if path.exists(f'img/Platformer/level{level}_data'):
            pickle_in = open(f'img/Platformer/level{level}_data', 'rb')
            world_data = pickle.load(pickle_in)


    #show the grid and draw the level tiles
    draw_grid()
    draw_world()
    #text showing current level
    draw_text(f'img/Platformer/Level: {level}', font, white, tile_size, screen_height - 60)
    draw_text('Press UP or DOWN to change level', font, white, tile_size, screen_height - 40)
    #event handler
    for event in pygame.event.get():
        #quit game
        if event.type == pygame.QUIT:
            run = False
        #mouseclicks to change tiles
        if event.type == pygame.MOUSEBUTTONDOWN and clicked == False:
            clicked = True
            pos = pygame.mouse.get_pos()
            x = pos[0] // tile_size
            y = pos[1] // tile_size
            #check that the coordinates are within the tile area
            if x < 33 and y < 34:
                #update tile value
                if pygame.mouse.get_pressed()[0] == 1:
                    world_data[y][x] += 1
                    if world_data[y][x] > 8:
                        world_data[y][x] = 0
                elif pygame.mouse.get_pressed()[2] == 1:
                    world_data[y][x] -= 1
                    if world_data[y][x] < 0:
                        world_data[y][x] = 8
        if event.type == pygame.MOUSEBUTTONUP:
            clicked = False
        #up and down key presses to change level number
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                level += 1
            elif event.key == pygame.K_DOWN and level > 1:
                level -= 1
    #update game display window
    pygame.display.update()

pygame.quit()
```

Input And Output Screen

Input screen



Registration Window

GAME WORLD

First Name:

Last Name:

Age:

Email:

Security Question: Select

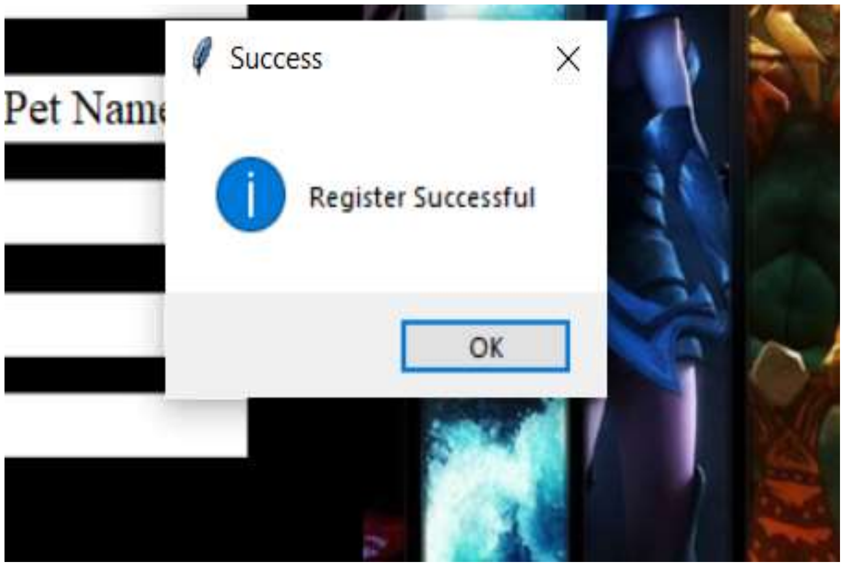
Answer:

Password:

Confirm:

I Agree the Terms & Conditions

Output screen

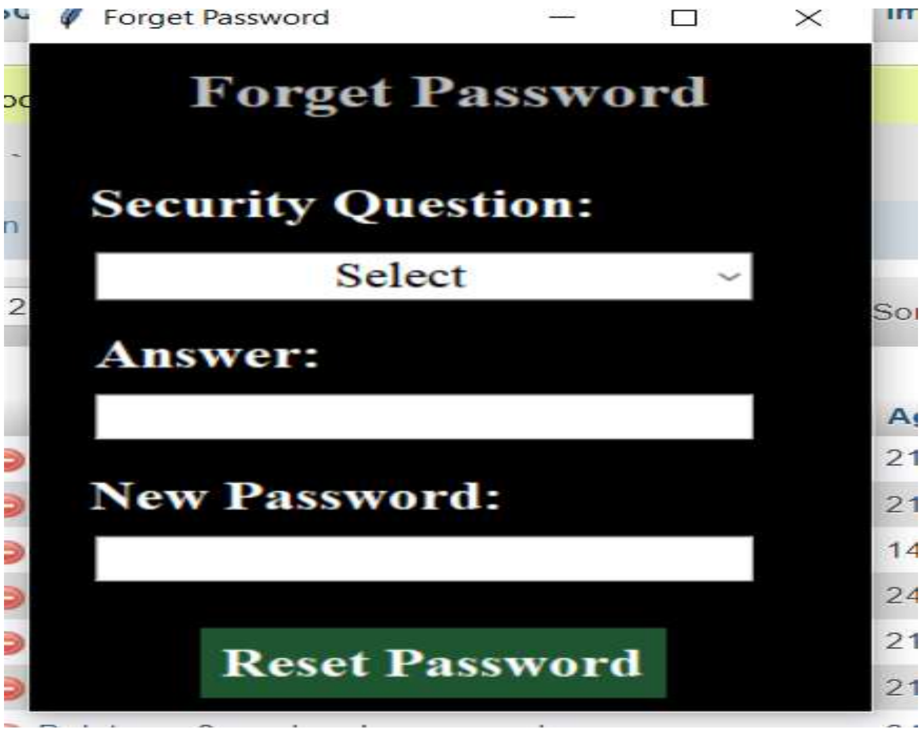


Pet Name

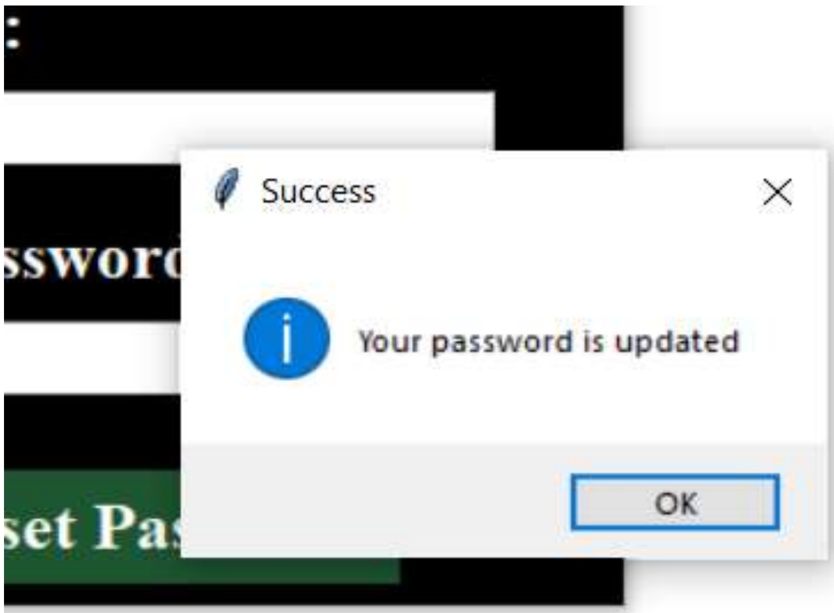
Success

Register Successful

Input screen

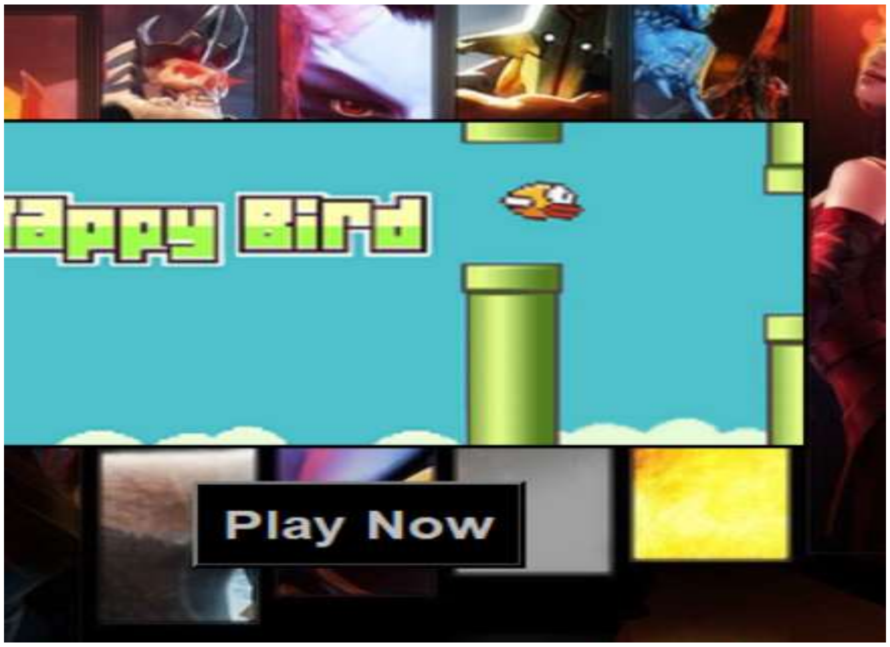

Forget Password

Output screen

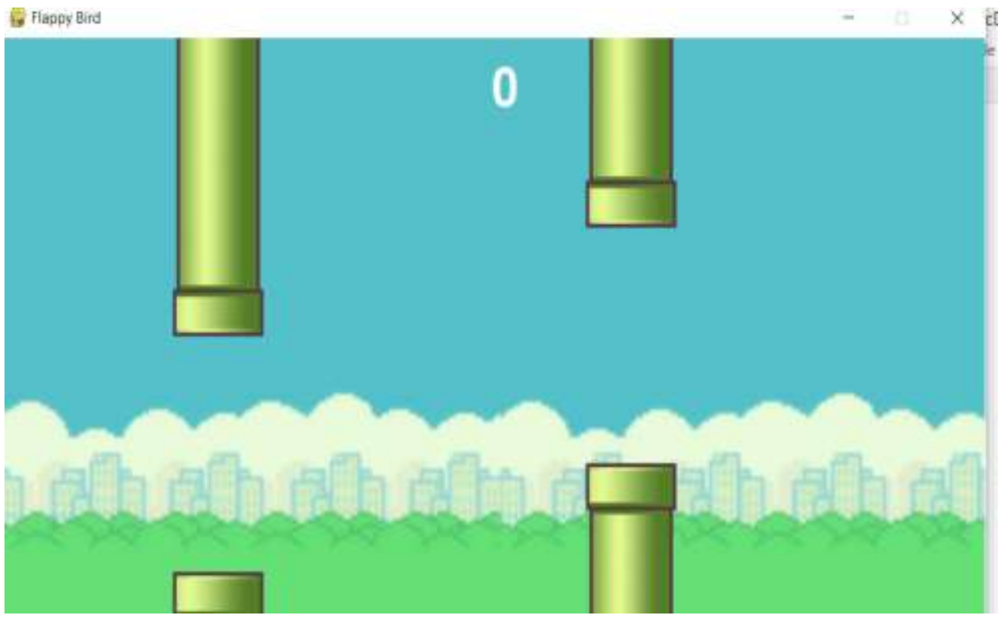

Success

Your password is updated

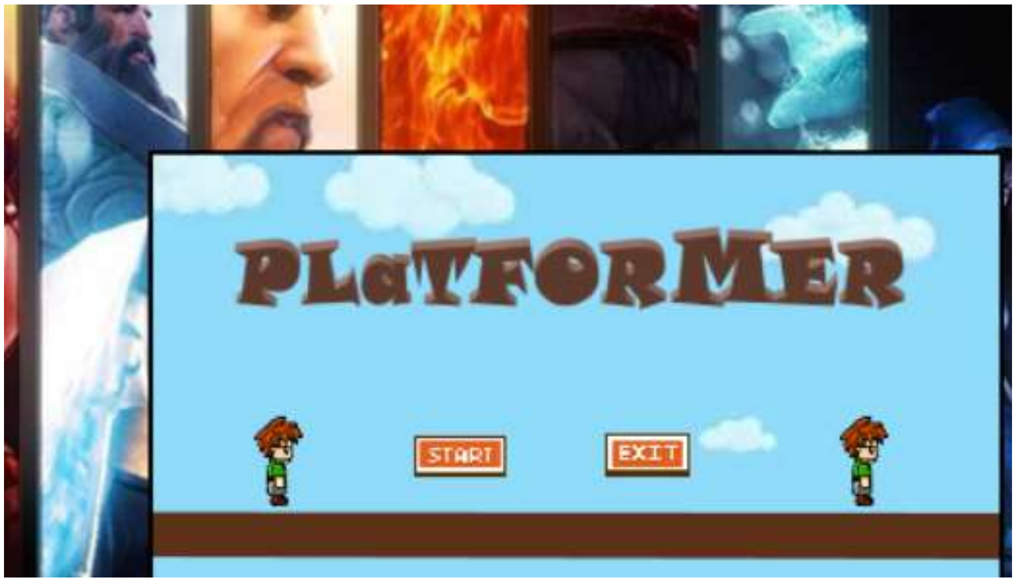
Input screen



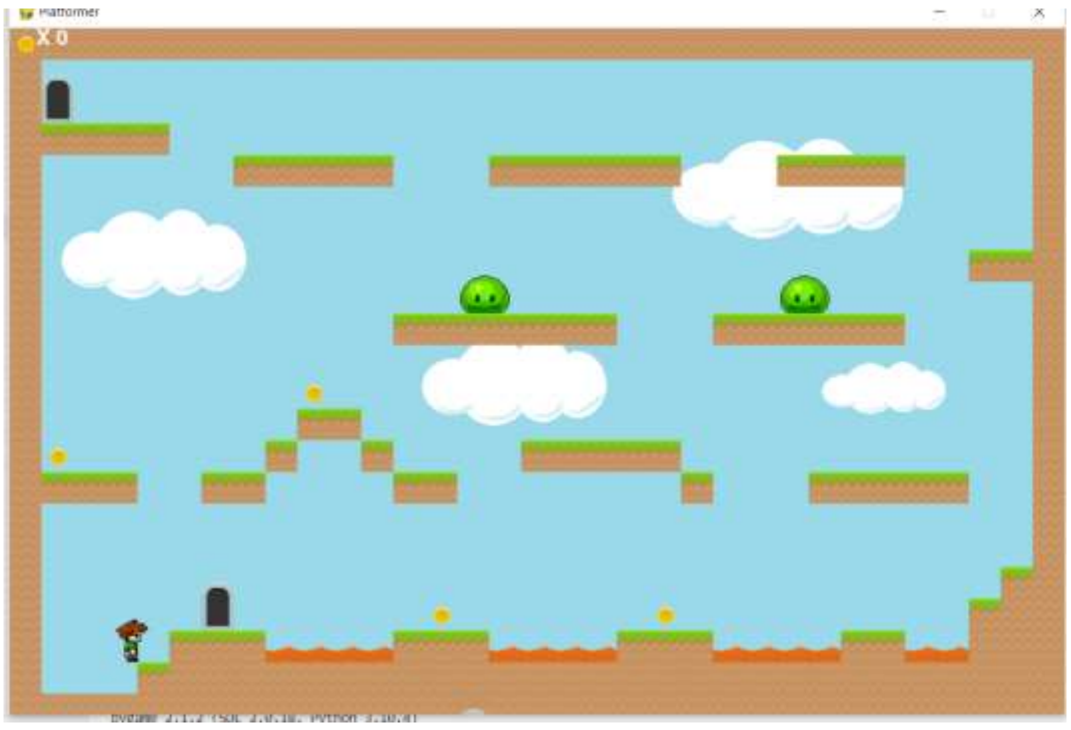
Output screen



Input screen



Output screen



TESTING AND VALIDATION

TESTING

Testing plays important role to identify the quality of any software. Testing actually refers to detecting errors in the system. Before testing can begin, a test plan needs to be developed. Test plan actually includes the type of testing that has to be performed on the code, resources for testing, how the software will be tested. There are several types of testing during the test phase, that includes quality assurance testing (QAT), System Integration testing (SIT), and user acceptance testing (UAT).

• Quality Assurance (QA) Testing: In this the procedures and processes are checked. This means whether the instructions are executed as per the user requirements and commands.

• System Integration Testing (UAT): It verifies proper execution of software components and proper interfacing between components within the solution. the objective behind this testing is to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.

• User Acceptance Testing (UAT): This is the last phase of the software testing procedure. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications. UAT is one of the final and critical software project procedures that must occur before newly developed software is rolled out to actual use. Before testing the system, we need to consider following questions in our mind:

• What is the actual problem?

  • How critical the problem is?
  • Measures should be taken for the upcoming problems or errors?

Testing gives chance to upgrade or to improve if any drawbacks prevail in the application.

Testing is generally done at two levels,
testing of individual modules and testing entire system.

During system testing, the system is used experimentally to ensure that the software does not fall. that it will run according to its specification and in the way users expect. Testing is done throughout system development at various stages.
Following are the type of testing done in the project:

1. Program Testing: In this, we have to concentrate on the software part, system software should be free from errors. whether it is syntax error or logical error. In this system, we have done software testing and the output of this test is satisfactory. It fulfills all the conditions, which was required for the program testing.

2. Stress Testing: this test is conducted to check the performance of the system in main hours. It finds out how much workload the system can bear. In stress testing of this system, we come to know that this software can work easily and accurately at any condition. The concentration is made on the

performance of the system by checking the giving input and their expected outputs.

3. Documentation Testing: this testing work to find out that whatever document supplied is satisfactory or there is a need to supply further document. In this system, all the documents which are supplied are satisfactory

Different types of Testing

1) Unit Testing
Unit testing is a type of software testing which is done on an individual unit or
component to test its corrections. Typically, Unit testing is done by the developer at
the application development phase. Each unit in unit testing can be viewed as a
method, function, procedure, or object. Developers often use test automation tools
such as Unknit, Unit, JUnit for the test execution.

Unit testing is important because we can find more defects at the unit test level.

For example, there is a simple calculator application. The developer can write the
unit test to check if the user can enter two numbers and get the correct sum for

addition functionality.

a) White Box Testing

White box testing is a test technique in which the internal structure or code
of an
application is visible and accessible to the tester. In this technique, it is easy
to find
loopholes in the design of an application or fault in business logic. Statement
coverage and decision coverage/branch coverage are examples of white box
test
techniques.

b) Gorilla Testing

Gorilla testing is a test technique in which the tester and/or developer test the
module
of the application thoroughly in all aspects. Gorilla testing is done to check
how
robust your application is.

2) Integration Testing
Integration testing is a type of software testing where two or more modules
of an
application is logically grouped together and tested as a whole. The focus of
this
type of testing is to find the defect on interface, communication, and data
flow

among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. For example, a user is buying a flight ticket from any airline website.
Users can see flight details and payment information while buying a ticket, but flight
details and payment processing are two different systems. Integration testing should
be done while integrating of airline website and payment processing system.

a) gray box testing

As the name suggests, gray box testing is a combination of white-box testing and
black-box testing. Testers have partial knowledge of the internal structure or code of
an application.

3) System Testing
System testing is types of testing where tester evaluates the whole system against
the specified requirements.

a) End to End Testing

It involves testing a complete application environment in a situation that mimics
real-world use, such as interacting with a database, using network communications,
or interacting with other hardware, applications, or systems if appropriate.

For example, a tester is testing a pet insurance website. End to End testing involves
testing of buying an insurance policy, LPM, tag, adding another pet, updating credit
card information on users' accounts, updating user address information, receiving
order confirmation emails and policy documents.
b) Black Box Testing

Blackbox testing is a software testing technique in which testing is performed
without knowing the internal structure, design, or code of a system under test.
Testers should focus only on the input and output of test objects.
Detailed information about the advantages, disadvantages, and types of Black Box
testing can be found here.

c) Smoke Testing Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level.

4) Acceptance Testing

Acceptance testing is a type of testing were client/business/customer test the software with real time business scenarios.

The client accepts the software only when all the features and functionalities work
as expected, this is the last phase of testing, after which the software goes into
production. This is also called User Acceptance Testing (UAT).

a) Alpha Testing

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers.

b) Beta Testing

Beta Testing is a type of software testing which is carried out by the clients/customers. It is performed in the Real Environment before releasing the
product to the market for the actual end-users.

Beta Testing is carried out to ensure that there are no major failures in the software
or product, and it satisfies the business requirements from an end-user perspective.
Beta Testing is successful when the customer accepts the software.

Non-Functional Testing

There are four main types of functional testing.

1) Security Testing
It is a type of testing performed by a special team. Any hacking method can
penetrate
the system.

Security Testing is done to check how the software, application, or website
is secure
from internal and/or external threats. This testing includes how much
software is
secure from malicious programs, viruses and how secure & strong the
authorization
and authentication processes are.

2) Performance Testing
Performance testing is testing of an application's stability and response time
by
applying load.

The word stability means the ability of the application to withstand in the
presence
of load. Response time is how quickly an application is available to users.
Performance testing is done with the help of tools. Loader.IO, JMeter,
LoadRunner,

etc. are good tools available in the market.

3) Usability Testing

Usability testing is testing an application from the user's perspective to check the
look and feel and user-friendliness.

For example, there is a mobile app for stock trading, and a tester is performing
usability testing. Testers can check the scenario like if the mobile app is easy to
operate with one hand or not, scroll bar should be vertical, background color of the
app should be black and price of and stock is displayed in red or green color.

4) Compatibility testing

This is a testing type in which it validates how software behaves and runs in a
different environment, web servers, hardware, and network environment.
Compatibility testing ensures that software can run on different configuration,
different databases, different browsers, and their versions. The testing team performs
compatibility testing.

Other Types of Testing

Back-end Testing

Whenever an input or data is entered on the front-end application, it is stored in the
database and the testing of such database is known as Database Testing or Backend
Testing.

There are different databases like SQL Server, MySQL, Oracle, etc. Database
Testing involves testing of table structure, schema, stored procedure, data structure,
and so on. In Back-end Testing, GUI is not involved, the testers are directly
connected to the database with proper access and testers can easily verify data by
running a few queries on the database.

Black Box Testing

Internal system design is not considered in this type of testing. Tests are based on
the requirements and functionality.

Detailed information about the advantages, disadvantages, and types of Black Box testing can be found here.

Browser Compatibility Testing

This is a sub-type of Compatibility Testing (which is explained below) and
is
performed by the testing team.

Browser Compatibility Testing is performed for web applications and
ensures that
the software can run with a combination of different browsers and operating
systems.
This type of testing also validates whether a web application runs on all
versions of
all browsers or not.

Backward Compatibility Testing

It is a type of testing that validates whether the newly developed software or
updated
software works well with the older version of the environment or not.

Backward Compatibility Testing checks whether the new version of the
software
works properly with the file format created by an older version of the
software. It

also works well with data tables, data files, and data structures created by the older

version of that software. If any of the software is updated, then it should work well
on top of the previous version of that software.

VALIDATION CHECKS

Data validation is the process of ensuring, at least as far as is possible, that the data given to a program by a user or from a file (essentially, the system's input) is of the correct type, and in the correct format.

There are however measures that can be taken to restrict the program's input to valid data. such measures involve the application of validation rules to any data being input to the program. In this system, Data validation rules can also make this system more user friendly, since they enable the program to warn the user immediately when there is a problem rather than simply allowing them to continue entering data until the program crashes or some other problem occurs.

In this proposed system, we have introduced the following data validation rules:

1. Value entered check: this is used for things like required fields in online forums where the user must enter some data (for example their username and password) and must not leave the field blank.

2. Permitted character check: it is useful for determining whether an input string contains valid characters. For example, a phone number may include the digits 0- 9.

3. Limit check: It is used for numeric values that must either be greater than or equal to some lower limit, or less than or equal to some upper limit. For example, the limited number that a user can enter as a phone number is 10.

4. Confirmation check: At the time of creating an account in this system, it is used for determining whether the enter password and confirm password are same or not.

5. Email address check: At the time of creating an account, the system can only accept a valid email id. For example, “@gmail.com”

SYSTEM AND SECURITIES MEASURES

SYSTEM SECURITY MAJORS

The objective of system security is the protection of information and property from theft, corruption and other types of damage, while allowing the information and property to remain accessible and productive. System security includes the development and implementation of security countermeasures. There are a number of different approaches to system security.

In this proposed system, we have provided the following security majors:

1. Password: the most widely method to prevent unauthorized access is to use passwords. The password needs to be kept secret and is only intended for the specific user. In this system, each password is associated with a specific username since many individuals may be accessing the same system.

2. Accessibility: In the game, with the help of admin username and password, only the admin has the right to update the game such as add, delete and update the products.

IMPLEMENTATION EVALUATION AND MAINTENANCE

IMPLEMENTATION

Implementation refers to that stage of project during which the theory is turned into practice i.e., converting soft ideas into actual process. In this stage physical system specifications are converted into working and reliable solution. This is where the system is developed. It is followed by testing and then again implementation.

Implementation phases: • Coding: this includes implementation of the design document into executable programming language code. The output of the coding phase is the source code for the software that acts as input to the testing and maintenance phase.

• Integration and Testing: It includes detection of errors in the software. The testing process starts with a test plan recognizes test-related activities, such as test case generation, testing criteria and resource allocation of testing. The code is tested and mapped against the design document created in the design phase.

• Installation: New system is installed and rolled out.

The steps involved in this phase are:

1. Acquisition and installation of hardware and software.

2. Conversion: It actually means to convert the old data to new format for proper functioning of the application in the new system.

3. User Training: User in this case has to be trained to use the system properly so that it is easy for them to grab control over the use of the application.

4. Documentation: This provides details of how to operate the given software, application and website.

The hardware and relevant software required for running the application must be installed and fully checked before implementation. In this phase conversion plays a crucial role. It actually means to convert the old data to a new format for proper functioning of the application in the new system. During the phase all the required programs are loaded onto user's computer. User must be trained.

The documentation is a complete description of the system from the user's point of view as it provides details of how to operate the given software and application. It also includes certain error messages that a user is expected to encounter during its usage and solution to the expected problems. It involves detained and step by step information of the project development so as to modify or update as per the new user requirements.

EVALUATION

Evaluation phase is the next to the implementation and it evaluates whether or not the system has met its requirements by comparing with the standards that were set before its actual development. The evaluation process includes the study of the current system and their drawbacks (if any) and various alternatives to improve and solve those prevailing problems. Evaluation is done by keeping the preliminary requirements of the user in mind.

Evaluation is included as a part of the final phase, but practically, evaluation takes place during each and every phase. The concentration should be on the satisfying the primary requirement of the users. The system is evaluated on the basis of following points:

• System Availability: whether the required system is available or not. • Compatibility: whether the application is compatible with the system or not.

• Cost: whether the developed application is affordable and has low maintenance cost. • Performance: it basically checks the efficiency of the application. Efficiency in handling the rush and fired queries simultaneously. It evaluates whether the application generates result at same speed when load is given to it as when it is stress free.

• Usability: whether the developed application is easily accessible and user-friendly. Evaluation in this system is done as follows:

• The errors generated in the code due to compatibility issues are debugged.
• Ease of installation and training.

• Adequacy and cost of hardware maintenance.

• Performance and its efficiency to handle the stress.

• Low maintenance cost. In this proposed system, evaluation is made on existing system, what are their drawbacks what improvement can be made to provide facility to users. Collecting the information required for improvement in the project and then implementing it in real use

MAINTENANCE

Maintenance is the final stage after the development process. After the system is installed, it must be maintained means that the computer programs must be modified and kept up to date. The average amount of time spent on maintenance is 60% of the total time. Estimates of the time spent by departments on maintenance have ranged from 48 to 60 percent of the total time spent developing systems. As the number of programs written increases, so does the amount of maintenance they require.

Maintenance covers a wide range of activities including correcting, coding, designing errors and updating user support. the project needs maintenance in further if any enhancements are made, maintenance of the hardware and software is also required.
The maintenance phase occurs once the system is operational. It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed

at the customer location. The maintenance phase also includes handling the residual errors that may exist in the software even the testing phase.

The maintenance phase also monitors system performance, rectifies bugs and requested changes are made.

Maintenance is performed for two reasons:

• First is to correct software errors. no matter how thoroughly the system is tested, bugs or errors creep into the computer programs. Bugs in commercial PC software are often documented as “known anomalies’’ and are corrected when new versions of the software are released or in an interim release. In custom software (also called bespoke software), bugs must be corrected as they are detected.

• Second, for performing system maintenance is to enhance the software’s capabilities in response to changing organizational needs, generally involving one of the following three situations:

1. Users often request additional features after they become familiar with the computer system and its capabilities.

3. Hardware and software are changing at an accelerated pace. In summary, Maintenance is an ongoing process over the life cycle of a system. After the application is installed, maintenance usually takes the form of correcting previously undetected program errors. Once these are corrected, the system approaches a steady state, providing dependable service to its users.

FUTURE SCOPE OF PROJECT

• Extra Features: Success of this application provides the extra ordinary features to the user.

• Simple and Easy: This project procedure is user-friendly, easy, simple and error free, it makes the software attractive.

• Reliability: This application can be run or expected into the current operating software's also.

. platform: Flappy bird game and platformer game will arcade- style in game platform.

. Use: The game use as an environment for Neuroevolutionary algorithms tests is an excellent study.

. Vision: The vision of the project, in terms of understanding scope, requires detailed foresight about what they will be its scope, requires of hours of gameplay, graphics quality, number of players supported, number of levels, quality of the AL, and other aspects that would be thought out fully in a game design document.

CONCLUSION

• Premium Quality: The main motive behind developing this project is to provide good quality software.

• Fulfilment of User Satisfaction: The basic concern behind developing this software is to provide good quality of game and provide high level of satisfaction to the user.

• Refreshing and Enhances Mind Power: This project is made basically to refresh the mind of people as well as it also enhances the concentration power of the user.

• when games are played in moderation and with mindfulness, they are a viable source of stress relief as well as a catalyst for mental health improvement and development of social skills.

▪ We have created this application using python language which is very popular.

▪ Our application also has many pros and cons which will be improved as per
the user’s preferences.

BIBLIOGRAPHY

References: -

1**) http://www.tutorialspoint.com/**

2) https://www.seleniumhq.org/download/

3) www.Google.co.in

4) YouTube

5) Python Programming Book

APPROVED COPY OF SYNOPSIS

A

PROJECT SYNOPSIS

ON

Flappy Bird Game and Platformer

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

AUTONOMOUS

In the Partial Fulfillment of

B.Com. (Computer Application) Final Year

Submitted by

Chandan Sharma
Sanket Malve

Under the Guidance of

Pravin J. Yadao

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

AUTONOMOUS

2021-2022

1. Introduction:

Flappy Game introduction

Flappy bird rose to popularity as an easy to play and addictive game at the end of 2013 It is a game where the user must navigate the bird through a series of pipes. It is stated that the creator (Dong Nguyen) made the game in 3 days, and has made up to $50,000 a day from advertising in the game. He is also sad that the game's popularity has taken over his otherwise "simple" life Now it is your chance to see how easy it is to construct a game like this

Platformer game introduction

A platformer, or platform video game, is one that traditionally features two-dimensional graphics in which players control characters who jump or climb between different platforms on the screen. It's a subgenre of the Action category, which is one of the many different types of video games.

2.Objective of Flappy bird game project

1: -Apply previous skill to put together basis blocks within scratch to create an app style game

2: -Extend knowledge about variables and how you can affect the difficulty within a game

3: -A simple game: one input (press any button or shake), one character to move up and down, and a series of obstacles that enter from the right and move across the screen and leave on the left.

4: -Some complications can be thrown in: the obstacles are random in size and structure, and the speed at which they move can increase over time.

Objective of platformer game project

1: - A platform game (often simplified as platformer or jump 'n' run games) is a video game genre and subgenre of action games in which the core objective is to move the player character between points in a rendered environment.

2: -Develop a data-driven cross platform game engine supporting
Desktop
XBOX One
Android

3: - Develop a game using the engine and produce playable builds for all supported platforms

3. Project Category: GAME

4. Tools/ Platform/ Languages to be used: Python

5. Scope of future application:

1: - We Will increase more level

2: - We will create new features in this project.

3: - The game use as an environment for Neuroevolutionary algorithms tests is an excellent study.

4: - Flappy bird game environment will using a minimal strategy that in addition to finding a simple agent to play the game and finds its quickly.

5: - The both games will popular in the world.

6: - flappy bird game and platformer game will arcade-style game in game platform.

Submitted by,	**Approved by,**
Name and Signature of the student	**Prof. Pravin Yadao**
1: - Sanket Malve	**Project Guide**
2: - Chandan Sharma	