REPORT FOR GAMERBUX

As a project work for Course

PYTHON PROGRAMMING (INT 213)

Name : Karan Jaikishan Chandwani.

Registration Number : 12101051

Name : Salik Aziz Khan.

Registration Number : 12100392

Name : Aryan Chand.

Registration Number : 12101154

Program : CSE B.Tech (3rd Semester)

School of Computer Science and Engineering (CSE)

Date of submission :18th November 2021

LOVELY PROFESSIONAL UNIVERSITY

GAMERBUX

11TH NOVEMBER 2021

ABSTRACT :

We provide lag free gaming, accept query with chatbot, using database to store user detail, login and signup. We have used Chatbots to collect user queries and responses. The login page is not tied to the dashboard of the application. It is simply a page on the application where visitors can enter their email address and password to gain access to restricted areas or content on our application. We have a Forgot Password page so that if you are unable to access your account, you can just click on the link above to reset your password.

We built this platform to help gamers play a lot of games that are free and accessible by everyone. We also have a customer support to help users give us feedback and suggestions. It makes our application user friendly as well.

ACKNOWLEDGEMENT:

We would like to thank our mentor - Prof. Sagar Pande for his advice and inputs on this project. A big thanks to our friends, who spent countless hours to listen and provide feedbacks.

TABLE OF CONTENTS

2. INTRODUCTION :

2.1 - CONTEXT :

This project has been done as part of our course for CSE at Lovely Professional University. Supervised by Sagar Pande, I have three months to fulfill the requirements in order to succeed the module.

2.2 – MOTIVATION:

Being extremely interested in everything having a relation with the app development, the group project was a great occasion to give us the time to learn and confirm our interest for this field. Our interest in gaming gave us a nod to build an application development.

2.3 – IDEA :

For gaming, we normally don't get sites and applications where we can give suggestions and interact with the backend and also there are may games which have to be paid for to play. Hence we got the idea that we'll implement all these suggestions into our project !

3.TEAM MEMBERS

3.1 TEAM LEADER :

1. Karan Jaikishan Chandwani

- CONTRIBUTIONS :

- Game
- Report
- Assets
- GUI

2. Salik Aziz Khan

- CONTRIBUTIONS :

- Database
- Coding(joined)
- Report
- Authentication (Login, Signup, Forgot password)
- GUI

3. Aryan Chand

- CONTRIBUTIONS :

- Chatbot
- GUI
- Report
- Research

4. LIBRARIES

1. PYREBASE :

Through Pyrebase we can access Realtime database of firebase in Python. We can also use any one of the authentication system(Google Sign in, Email and Password, etc) from it.

2. JARVIS :

Jarvis AI is a Python Module which is able to perform task like Chatbot, Assistant etc. It provides base functionality for any assistant application. This Jarvis AI is built using Tensorflow, Pytorch, Transformers and other opensource libraries and frameworks.

3. PYGAME :

- Pygame is a cross-platform set of Python modules which is used to create video games.
- It consists of computer graphics and sound libraries designed to be used with the Python programming language.

5. PROPOSED MODULES

1. AUTHENTICATION SYSTEM :

- Login
- Signup
- Forgot Password
- Dashboard

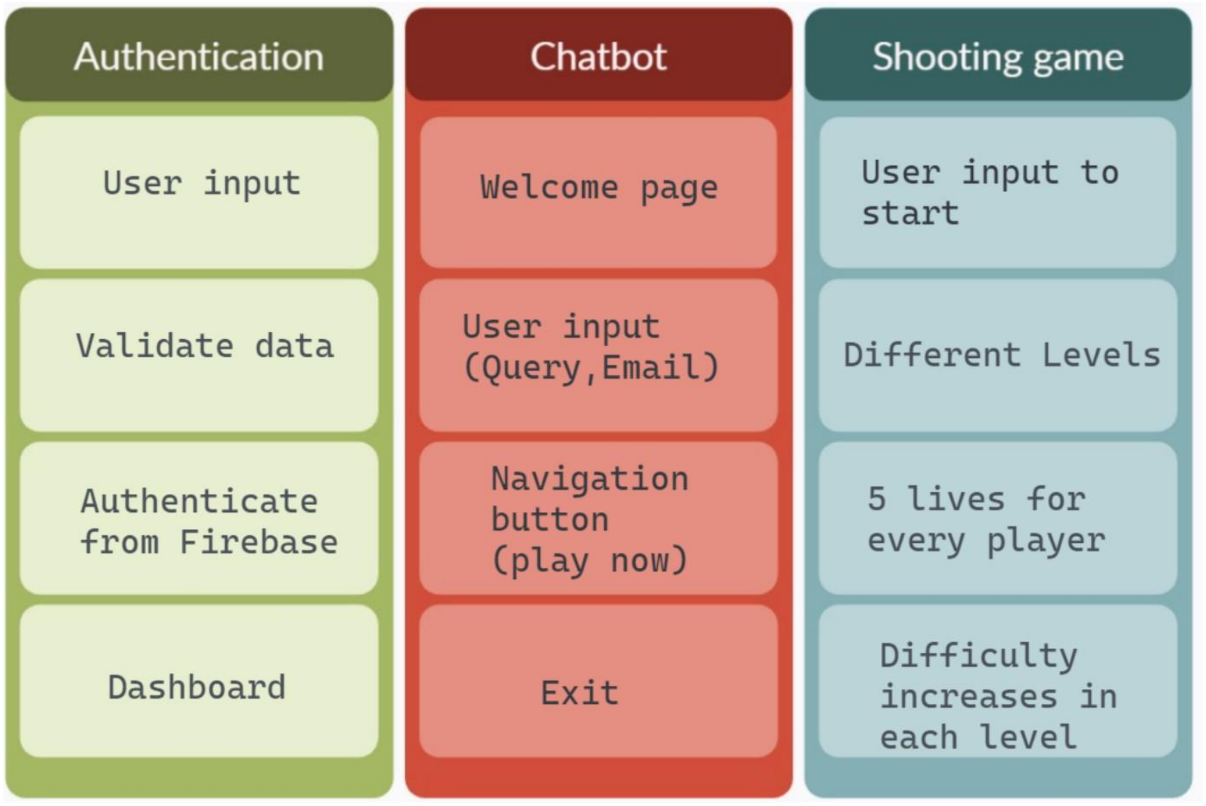
2. CHATBOT :

- Welcome Page
- Raise a query
- Navigate to Game
- Exit

3. SHOOTING GAME :

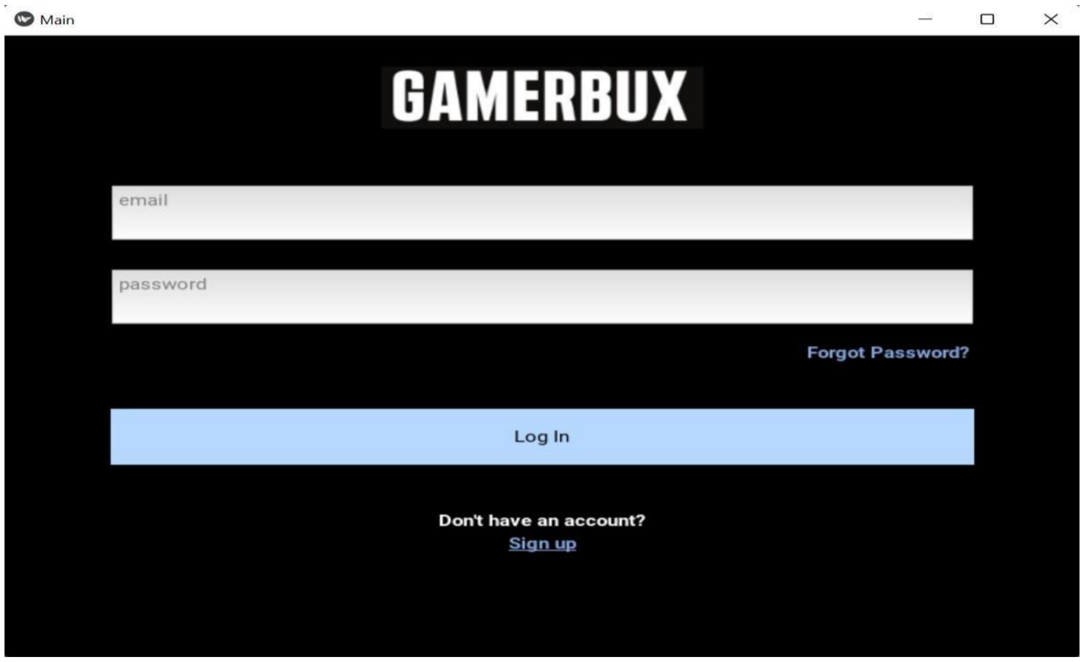
- User input to start game.
- There are 5 lives for the player.
- There are 15 levels.
- After every level, difficulty increases.

Basic layout for GamerBux.

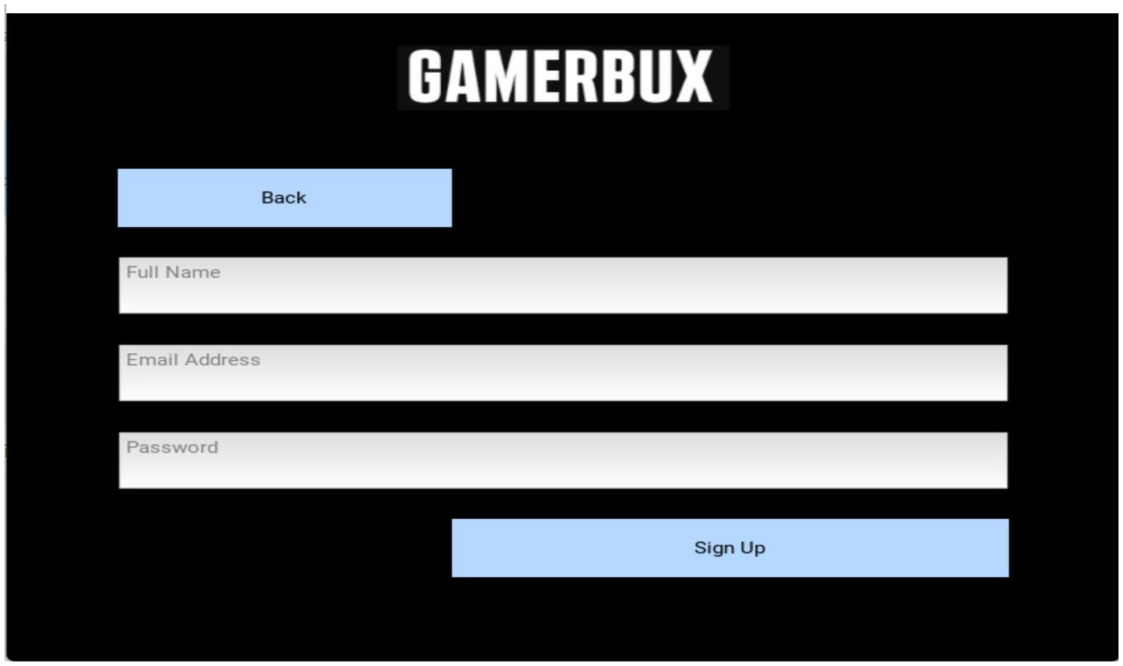


6. SCREENSHOTS.

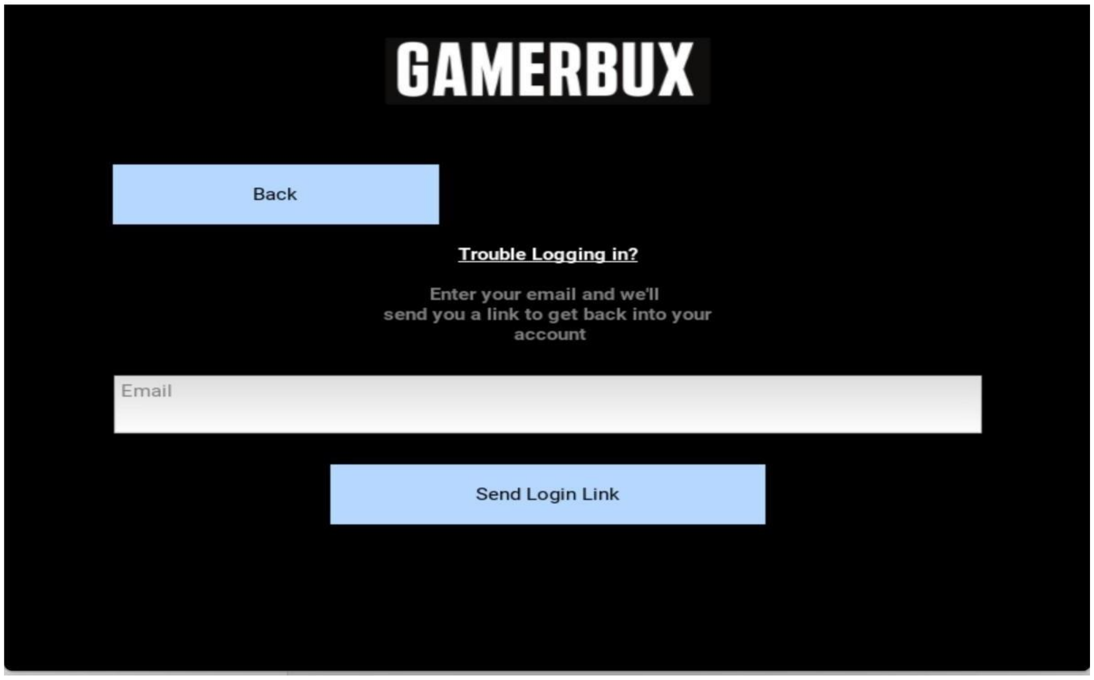
1. Main page / Login :



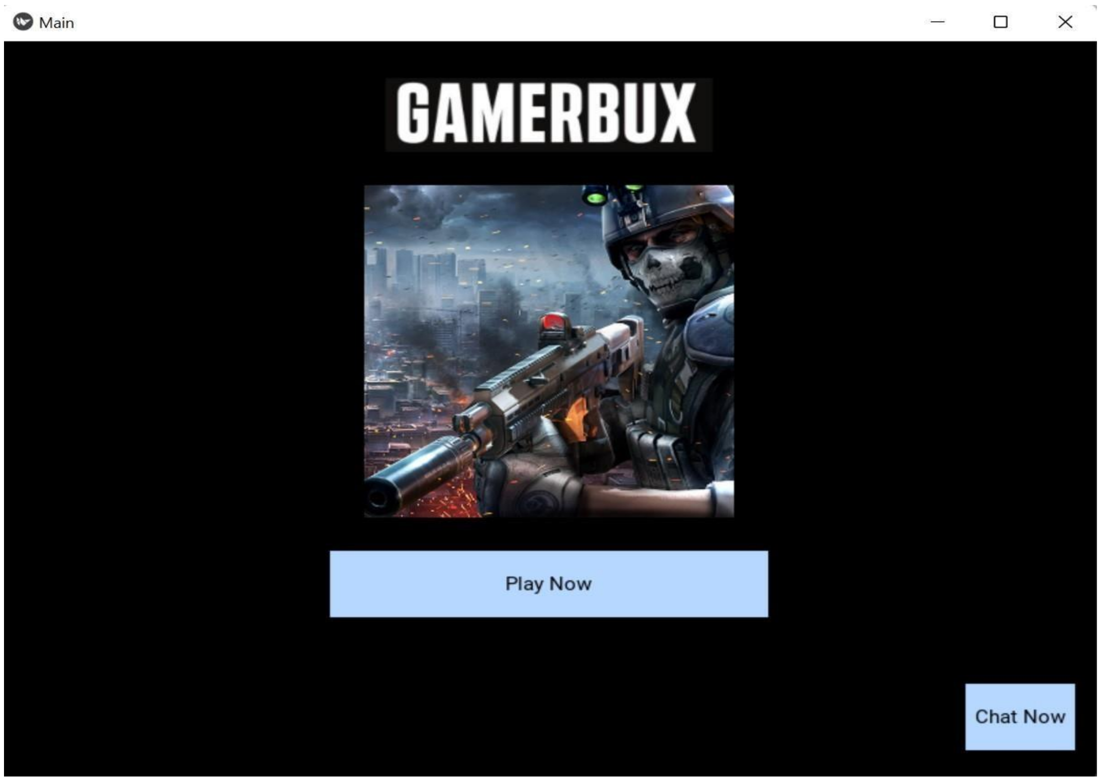
2. SIGN UP :



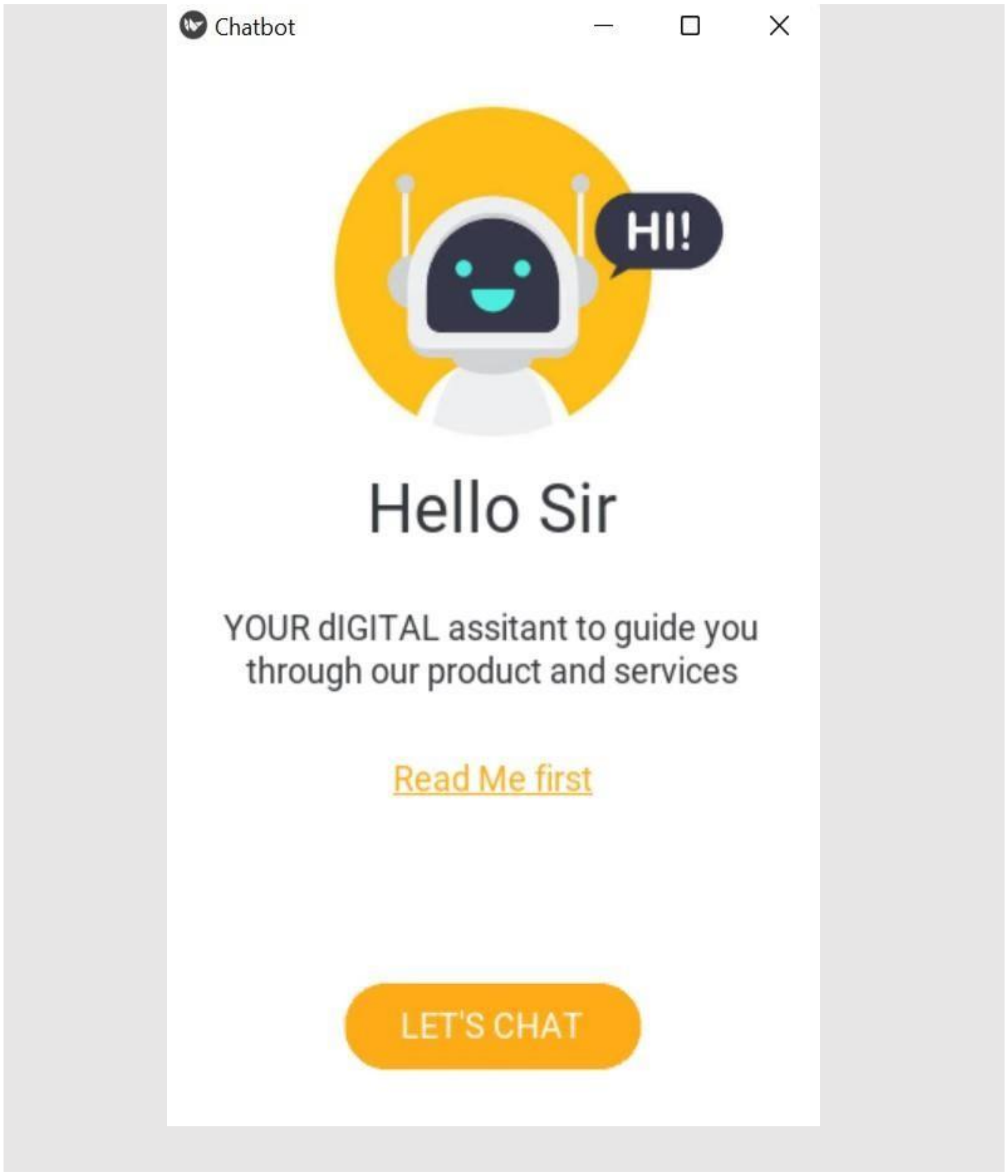
3. FORGOT PASSWORD :


GAMERBUX
Back
Trouble Logging in?
Enter your email and we'll
send you a link to get back into your
account
Email
Send Login Link

4. DASHBOARD :



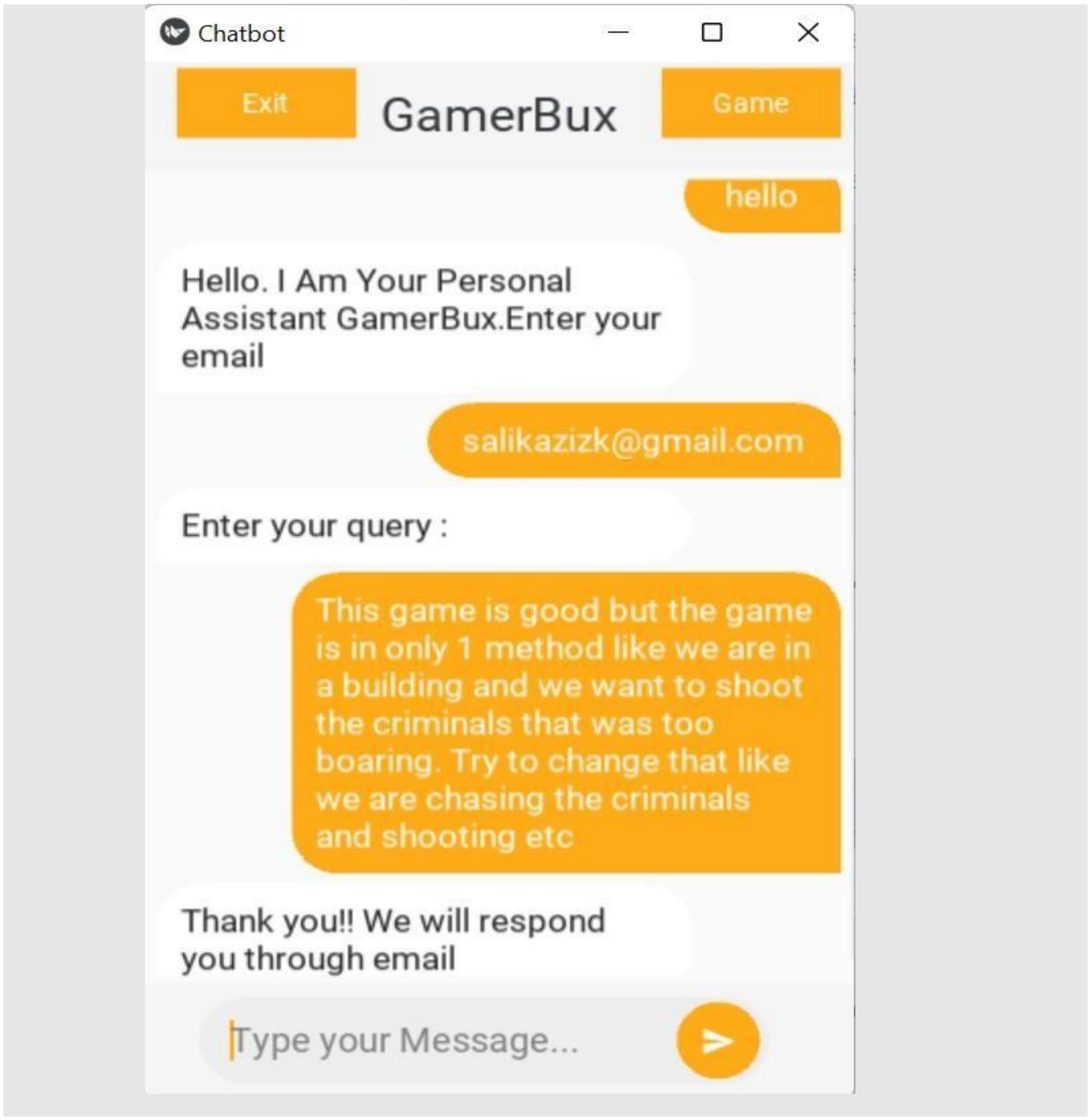
5. CHATBOT HOME PAGE:



Hello Sir

YOUR dIGITAL assitant to guide you through our product and services

6. CHATBOT QUERIES AND INTERACTION:



hello

Hello. I Am Your Personal Assistant GamerBux.Enter your email

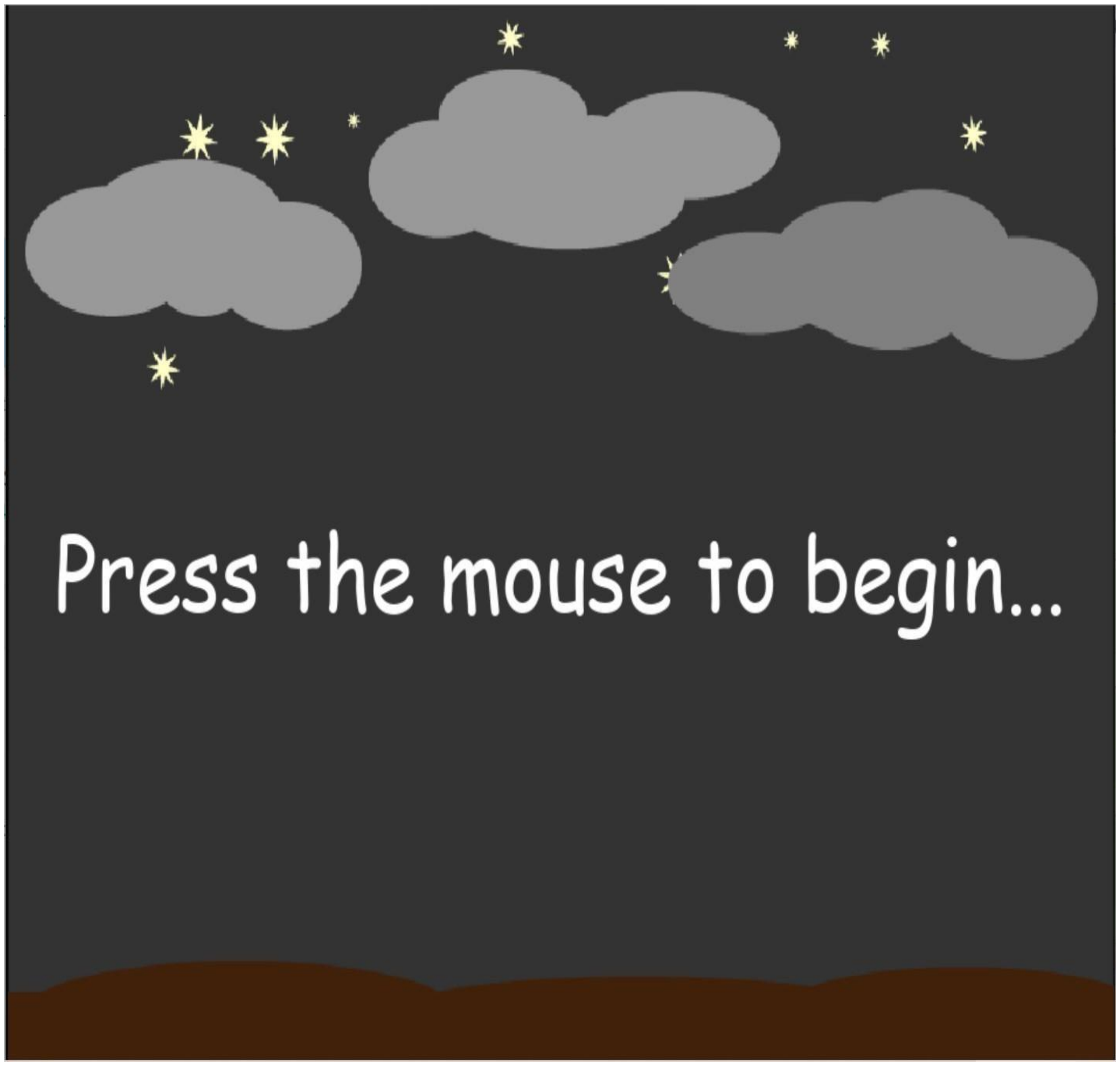
salikazizk@gmail.com

Enter your query :

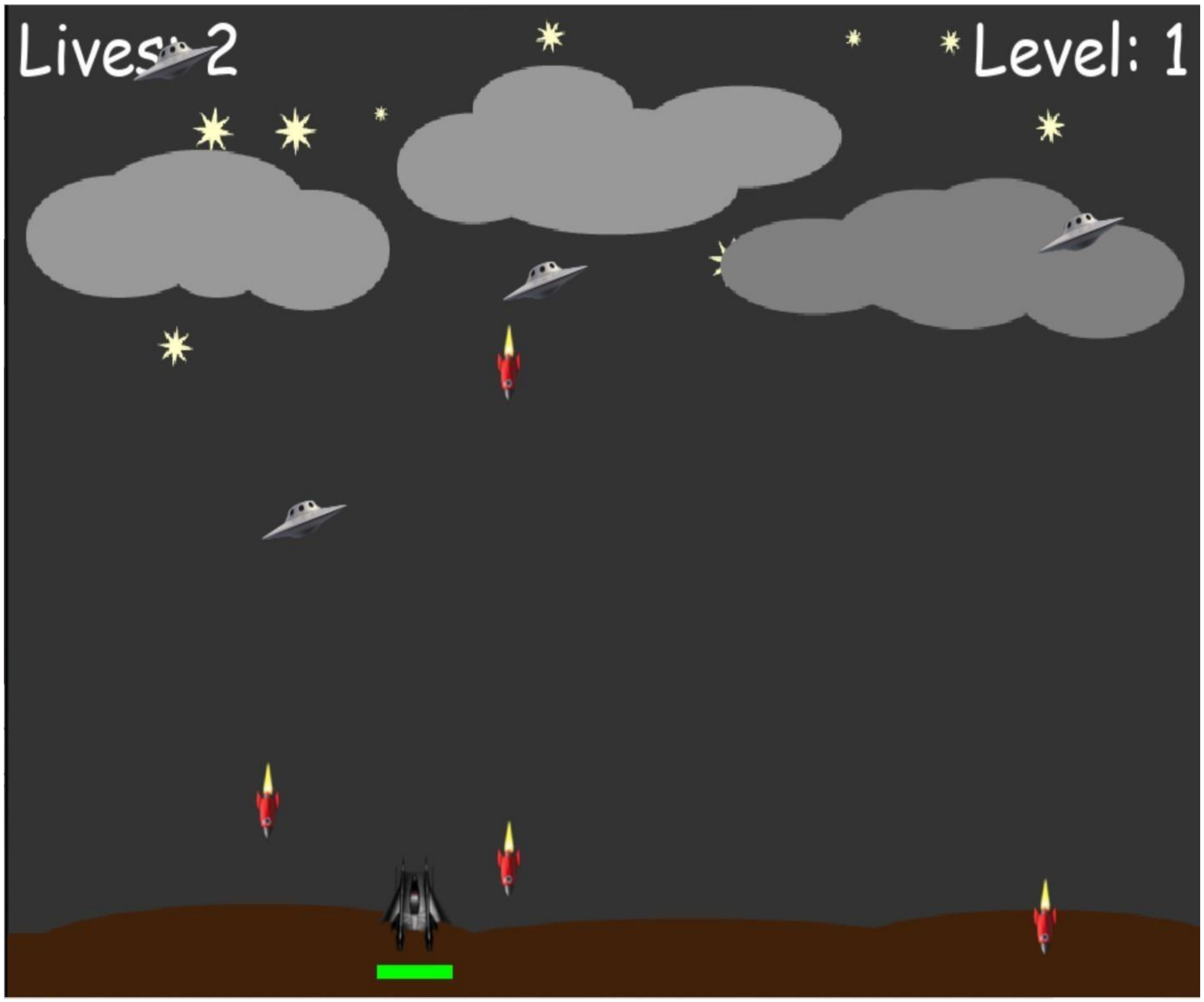
This game is good but the game is in only 1 method like we are in a building and we want to shoot the criminals that was too boaring. Try to change that like we are chasing the criminals and shooting etc

Thank you!! We will respond you through email

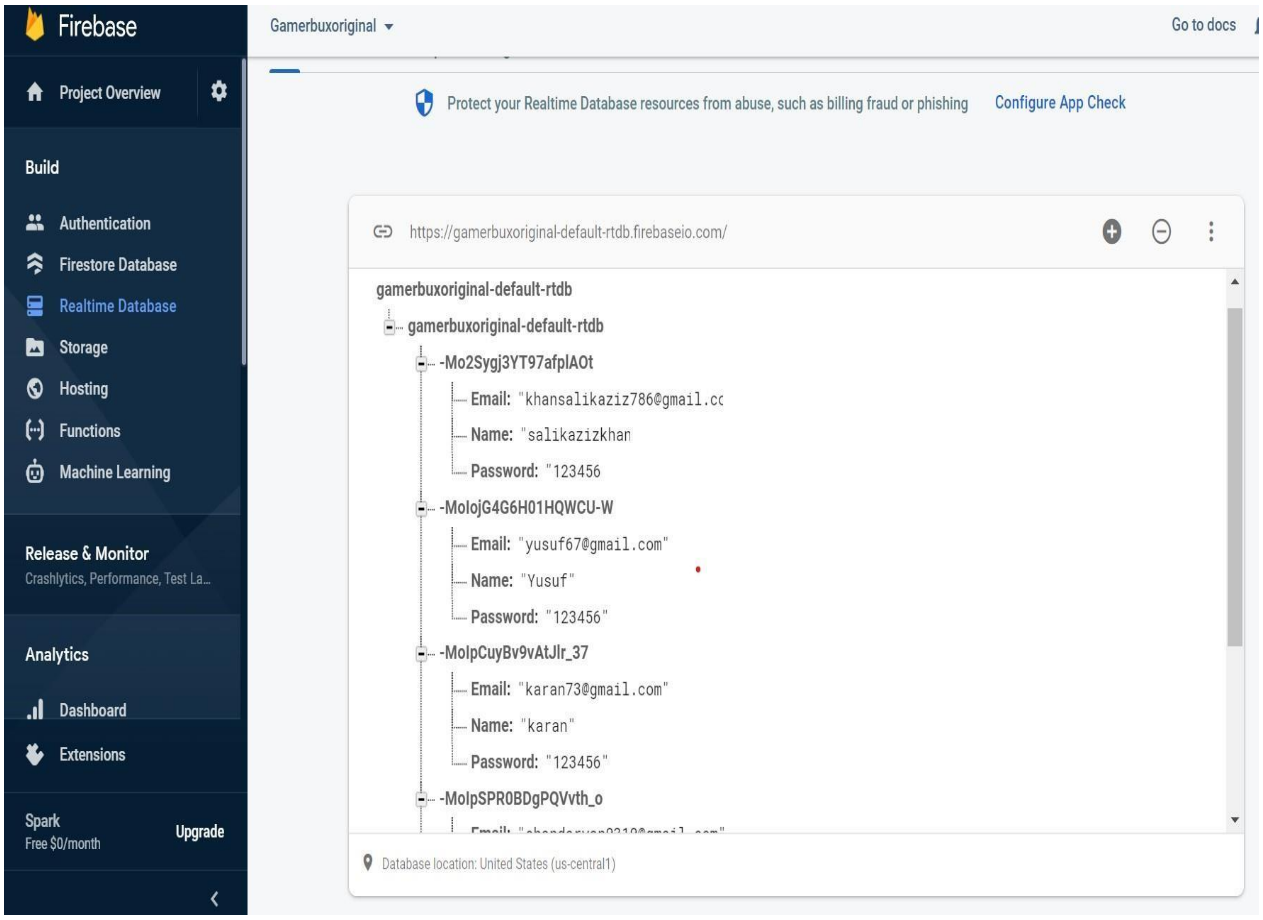
7. SHOOTING GAME :



CONT.



REAL-TIME DATABASE :



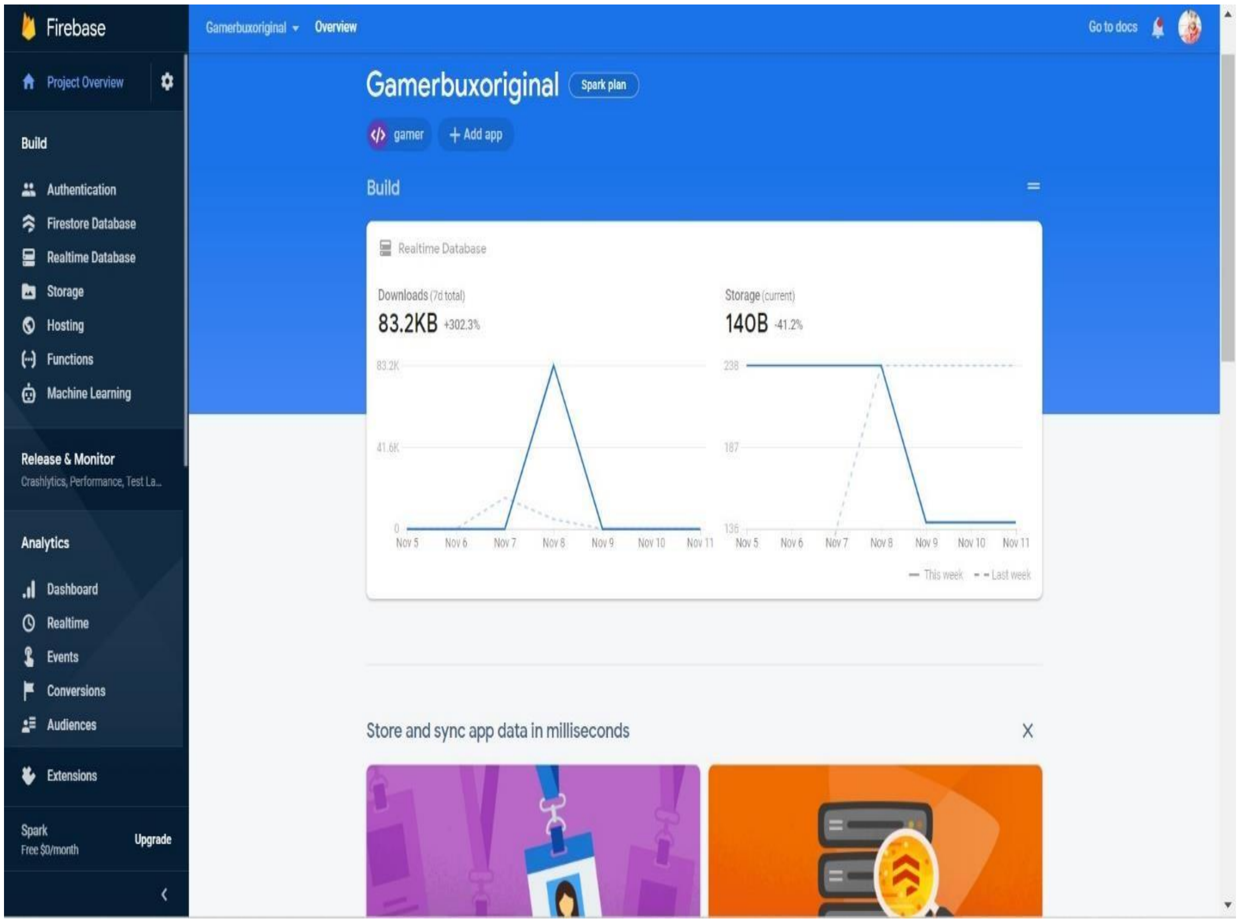
Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://gamerbuxoriginal-default-rtdb.firebaseio.com/

```
gamerbuxoriginal-default-rtdb
 - gamerbuxoriginal-default-rtdb
    - -Mo2Sygj3YT97afplAOt
        Email: "khansalikaziz786@gmail.cc
        Name: "salikazizkhan
        Password: "123456
    - -MolojG4G6H01HQWCU-W
        Email: "yusuf67@gmail.com"
        Name: "Yusuf"
        Password: "123456"
    - -MolpCuyBv9vAtJlr_37
        Email: "karan73@gmail.com"
        Name: "karan"
        Password: "123456"
    - -MolpSPR0BDgPQVvth_o
```

Database location: United States (us-central1)

ANALYTICS :


Firebase
Gamerbuxoriginal
Overview
Go to docs
Project Overview
Build
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
Release & Monitor
Crashlytics, Performance, Test La...
Analytics
Dashboard
Realtime
Events
Conversions
Audiences
Extensions
Spark
Free $0/month
Upgrade
Gamerbuxoriginal
Spark plan
gamer
Add app
Build
Realtime Database
Downloads (7d total)
83.2KB +302.3%
Storage (current)
140B -41.2%
This week
Last week
Store and sync app data in milliseconds

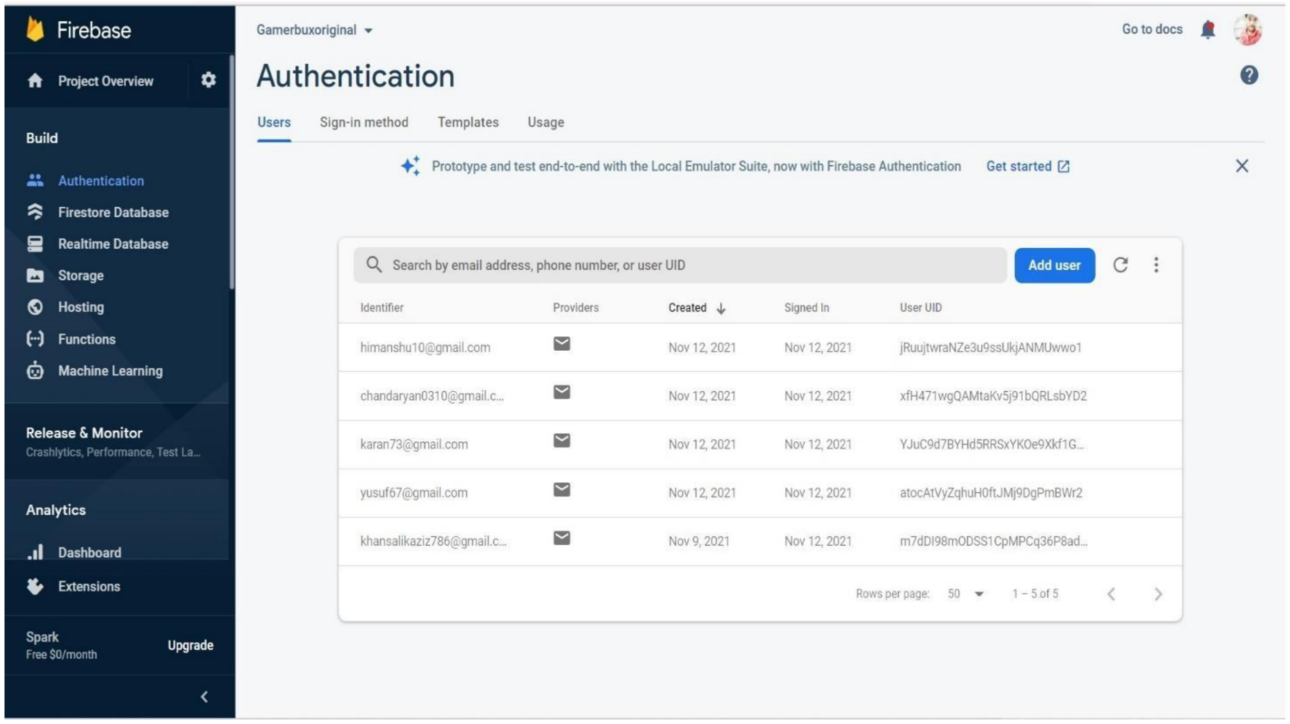
7. AUTHENTICATION SYSTEM

We are using firebase as our database and we are using Pyrebase library to use it's feature. The main reason for using firebase is that Firebase is a platform developed by Google (so there will not any security issues) and it is providing realtime access to data.

For GUI we are using Kivy and purpose of using it is that Kivy is a free and open source Python framework for developing desktop application and other multitouch application software with a natural user interface.

This authentication system includes several activities-:

1) Main page /Login Page
   - ⇨ On this page there are 2 input fields which collect email and password from user.
   - ⇨ There is a login button which performs the authentication process from firebase database on click event(onclick) and if authentication is successful then user can access dashboard and if it is unsuccessful, it will display an error message.
   - ⇨ And there are two links, the first will navigate user to forgot password page and the second one will navigate user to signup page.
2) **Forgot password page**
   - ⇨ This page contains back button on top which will navigate user to main page on button click(onclick).
   - ⇨ There is an input field which collect user email to reset password and there is a button (send login link-: button name) which will send link to reset password on that email.
3) **Signup page**
   - ⇨ This page contains back button on top which will navigate user to main page on button click(onclick).
   - ⇨ On this page there are three input fields which collects username, email and password respectively.
   - ⇨ There is a signup button which check if all details are valid or not on click event(onclick) and if details are valid, it will display a signup success message and if details are invalid it will display an error message.



Authentication

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication

Identifier	Providers	Created	Signed In	User UID
himanshu10@gmail.com	✉	Nov 12, 2021	Nov 12, 2021	jRuujtwraNZe3u9ssUkjANMUwwo1
chandaryan0310@gmail.c...	✉	Nov 12, 2021	Nov 12, 2021	xfH471wgQAMtaKv5j91bQRLsbYD2
karan73@gmail.com	✉	Nov 12, 2021	Nov 12, 2021	YJuC9d7BYHd5RRSxYKOe9Xkf1G...
yusuf67@gmail.com	✉	Nov 12, 2021	Nov 12, 2021	atocAtVyZqhuH0ftJMj9DgPmBWr2
khansalikaziz786@gmail.c...	✉	Nov 9, 2021	Nov 12, 2021	m7dDI98mODSS1CpMPCq36P8ad...

Rows per page: 50 1 – 5 of 5

1)Main.py

```
import pyrebase
import os
from kivy.app import App
from kivy.core.window import Window
from kivy.lang import Builder
from kivy.uix.screenmanager import Screen
from kivy.uix.button import ButtonBehavior
from kivy.uix.label import Label

from firebaseaut import MyFirebase


class HomeScreen(Screen):
    pass


class SettingsScreen(Screen):
    pass


class LoginScreen(Screen):
    pass


class SignUpScreen(Screen):
    pass


class ForgotPasswordScreen(Screen):
    pass


class Dashboard(Screen):
    pass
```

```
class LabelButton(ButtonBehavior, Label):
    pass


GUI = Builder.load_file("main.kv")


class MainApp(App):
    def build(self):
        self.my_firebase = MyFirebase()
        return GUI

    def change_screen(self, filename):
        screen_manager = self.root.ids["screen_manager"]
        screen_manager.current = filename
        pass
    def visitchatbot(self):
       print("Button click")
       App.get_running_app()
       Window.close()
       os.system('python chatbot.py')

    def visitgame(self):
       print("Button click")
       App.get_running_app()
       Window.close()
       os.system('python game.py')

MainApp().run()
```

2)Firebaseaut.py

```
import pyrebase
from firebase import firebase
import json
from kivy.app import App

config = {
  "apiKey": "AIzaSyCTaHJug8updPNZv_b-ULJDbwVC4grjDKo",
  "authDomain": "gamerbuxoriginal.firebaseapp.com",
  "databaseURL": "https://pythonfirebase-d3500-default-rtdb.firebaseio.com",
  "projectId": "gamerbuxoriginal",
  "storageBucket": "gamerbuxoriginal.appspot.com",
  "messagingSenderId": "524291599220",
  "appId": "1:524291599220:web:8f460720995e3280b6da21",
  "measurementId": "G-QVCF99NZ5P"
}

firebase_auth = pyrebase.initialize_app(config)

#Enter Your firebase RealTime database url here
firebase_data = firebase.FirebaseApplication("https://gamerbuxoriginal-default-
rtdb.firebaseio.com/" , None)

class MyFirebase() :
    def sign_up(self , fullname , email , password):

        try :
            # RealTime Database
            data = {
                "Name": fullname,
                "Email": email,
                "Password": password
            }
            result = firebase_data.post("gamerbuxoriginal-default-rtdb", data)
#Enter your database table name here

            # Creating User
            signup_auth = firebase_auth.auth()
            user_signup = signup_auth.create_user_with_email_and_password(email,
password)
            print("SignUp Successfully")

App.get_running_app().root.ids["signup_screen"].ids["signup_screen"].text =
"[b][color=#FF0000]Signup Succesfully.[/color][/b]"
        except:

App.get_running_app().root.ids["signup_screen"].ids["signup_screen"].text =
"[b][color=#0000FF]Please enter correct details.[/color][/b]"




    def sign_in(self , email , password):
        signin_auth = firebase_auth.auth()

        try :
            user_login =
signin_auth.sign_in_with_email_and_password(email,password)
            print("Login Successfully !!!")
            print(user_login["registered"])
            path_to_home = user_login["registered"]

            if path_to_home == True :
                print("hello")

App.get_running_app().root.ids["login_screen"].ids["login_message"].text = ""
                App.get_running_app().change_screen("dashboard")

#App.get_running_app().root.ids["home_screen"].ids["passing_email"].text =
"[b]%s[/b]" %email

        except :

App.get_running_app().root.ids["login_screen"].ids["login_message"].text =
"[b]Invalid Email or Password[/b]"

    def forgot_password(self , email) :
        try:
            auth = firebase_auth.auth()
            auth.send_password_reset_email(email)

App.get_running_app().root.ids["forgot_password_screen"].ids["forgot_message"].te
xt = "[b]Thanks! Please check your email .[/b]"
        except:

App.get_running_app().root.ids["forgot_password_screen"].ids["forgot_message"].te
xt = "[b][color=#FF0000]Please Enter Correct Email !.[/color][/b]"
```

8. JARVIS AI CHATBOT

Chatbot is a form of Artificial Intelligence (AI) used in messaging apps. This tool helps add convenience for customers—they are automated programs that interact with customers like a human would and cost little to nothing to engage with.

For GUI we are using Kivy and purpose of using is that Kivy is that it is a free and Open source Python framework for developing desktop and other multitouch application software with a natural user interface.

The chatbot includes several activities:

1). DASHBOARD PAGE :

On this page one image and some instructions are there.

Then there is let's chat button. when we click on button it will redirect to next page which is main chat page. where customer interact with ai chatbot.

2). CHATTING PAGE :

This page has 3 Buttons. First button on the top-left side which is an exit button. When we click on it, it will redirect to login page.

Next button is on right-top side which is a play now button. When we click on it, it will redirect to game interface.

Third button is message sending button which is placed on right-down side. After typing the messages, we simply click on it.

1)Chatbot.py

```
from firebase import firebase
from kivy.clock import Clock
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.core.window import Window
from kivy.uix.screenmanager import ScreenManager
from kivymd.uix.label import MDLabel
from kivy.properties import StringProperty, NumericProperty
from kivy.core.text import LabelBase
import os
import pyrebase
Window.size = (350, 550)

config = {
  "apiKey": "AIzaSyCTaHJug8updPNZv_b-ULJDbwVC4grjDKo",
  "authDomain": "gamerbuxoriginal.firebaseapp.com",
  "databaseURL": "https://pythonfirebase-d3500-default-rtdb.firebaseio.com",
  "projectId": "gamerbuxoriginal",
  "storageBucket": "gamerbuxoriginal.appspot.com",
  "messagingSenderId": "524291599220",
  "appId": "1:524291599220:web:8f460720995e3280b6da21",
  "measurementId": "G-QVCF99NZ5P"
}
firebase_auth = pyrebase.initialize_app(config)

#Enter Your firebase RealTime database url here
firebase_data = firebase.FirebaseApplication("https://gamerbuxoriginal-default-
rtdb.firebaseio.com/" , None)
class Command(MDLabel):
    text = StringProperty()
    size_hint_x = NumericProperty()
    halign = StringProperty()
    font_size = 17

class Response(MDLabel):
    text = StringProperty()
    size_hint_x = NumericProperty()
    halign = StringProperty()
    font_size = 17







class Chatbot(MDApp):
    def change_screen(self, name):
        screen_manager.current = name

    def build(self):
       global screen_manager
       screen_manager = ScreenManager()
       screen_manager.add_widget(Builder.load_file("err.kv"))
       screen_manager.add_widget(Builder.load_file("Chats.kv"))
       return screen_manager

    def bot_name(self):
        if screen_manager.get_screen('err').bot_name.text != "":
            screen_manager.get_screen('chats').bot_name.text =
screen_manager.get_screen('err').bot_name.text
```

```
            screen_manager.current = "chats"

    def response(self, *args):
        response = ""
        count = 0
        for i in value:
            count = count + 1
        if value == "Hello" or value == "hello":
            response = f"Hello. I Am Your Personal Assistant
{screen_manager.get_screen('chats').bot_name.text}.Enter your email"
            #response1 = "Please enter your Email Address"
            #response2 ="Enter your query :"
            #response3 = "Invalid input"

        elif value.endswith("@gmail.com"):
            response = "Enter your query :"
            firebase_data.post("Raise ticket/email",value)


        elif count > 30:
         firebase_data.post("Raise ticket/query",value)
         response = "Thank you!! We will respond you through email"

        elif value == "How are you?" or "how are you?":
          response = "Sorry could you say that again?"

        else:
            response = "Sorry could you say that again?"

screen_manager.get_screen('chats').chat_list.add_widget(Response(text=response,
size_hint_x=.75))

#screen_manager.get_screen('chats').chat_list.add_widget(Response(text=response1,
size_hint_x=.75))


    def reve(self):
        print("Button click")
        MDApp.get_running_app()
        Window.close()
        os.system('python main.py')
    def game(self):
        print("Button click")
        MDApp.get_running_app()
        Window.close()
        os.system('python game.py')


    def send(self):
        global size, halign, value
        if screen_manager.get_screen('chats').text_input != "":
            value = screen_manager.get_screen('chats').text_input.text
            if len(value) < 6:
                size = .22
                halign = "center"
            elif len(value) < 11:
                size = .32
                halign = "center"
            elif len(value) < 16:
                size = .45
                halign = "center"
            elif len(value) < 21:
                size = .58
                halign = "center"
            elif len(value) < 26:
                size = .71
                halign = "center"
            else:
                size = .77
                halign = "left"

screen_manager.get_screen('chats').chat_list.add_widget(Command(text=value,
size_hint_x=size, halign=halign))
            Clock.schedule_once(self.response, 0.5)
            screen_manager.get_screen('chats').text_input.text = ""





if __name__ == '__main__':

    Chatbot().run()
```

9. SHOOTING GAME

It is a simple game where the player (defender) defends himself from the spaceships coming to kill him. You will be given a total of 5 lives and you will have unlimited bullets to shoot the opponent.

It tests the player’s spatial awareness, reflexes and speed of the player. This shooting game focuses almost entirely on the defeat of the character's enemies using the weapons given to the player.
The game is developed based on a space environment to make it feel even more realistic.

We have used Pygame which makes it easier to build games and provides a lot of options as well. Then we have added assets for the player to use. We have used different classes and functions to divide the game so it functions properly. We have written the code in such a way that it is easy for the other coders to read and understand.

We have used shortcuts for the player to use for shooting bullets, moving the rocket, and also for going back to the Main Menu.

- The “A” letter is a shortcut for making the player’s rocket go LEFT.
- The “D” letter is a shortcut for making the player’s rocket go RIGHT.
- The “W” letter is a shortcut for making the player’s rocket go UP.
- The “S” letter is a shortcut for making the player’s rocket go DOWN.
- The “SPACEBAR” letter is a shortcut for making the player’s rocket shoot.
- The “X” letter is a shortcut for Exit.

1)Game.py

```python
import pygame
import os
import time
import random
import keyword
import os

from kivy.core.window import Window
from pynput import keyboard
pygame.font.init()

WIDTH, HEIGHT = 950, 750
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Space Shooter")

# Load images
RED_SPACE_SHIP = pygame.image.load(os.path.join("assets", "ufo.png"))
GREEN_SPACE_SHIP = pygame.image.load(os.path.join("assets", "ufo.png"))
BLUE_SPACE_SHIP = pygame.image.load(os.path.join("assets", "ufo.png"))

# Player player
YELLOW_SPACE_SHIP = pygame.image.load(os.path.join("assets", "raider.png"))

# Lasers
RED_LASER = pygame.image.load(os.path.join("assets", "missile111.png"))
GREEN_LASER = pygame.image.load(os.path.join("assets", "missile111.png"))
BLUE_LASER = pygame.image.load(os.path.join("assets", "missile111.png"))
YELLOW_LASER = pygame.image.load(os.path.join("assets", "bullet111.png"))

# Background
BG = pygame.transform.scale(pygame.image.load(os.path.join("assets",
"background.png")), (WIDTH, HEIGHT))

class Laser:
    def __init__(self, x, y, img):
        self.x = x
        self.y = y
        self.img = img
        self.mask = pygame.mask.from_surface(self.img)

    def draw(self, window):
        window.blit(self.img, (self.x, self.y))

    def move(self, vel):
        self.y += vel

    def off_screen(self, height):
        return not(self.y <= height and self.y >= 0)

    def collision(self, obj):
        return collide(self, obj)


class Ship:
    COOLDOWN = 30

    def __init__(self, x, y, health=100):
        self.x = x
        self.y = y
        self.health = health
        self.ship_img = None
        self.laser_img = None
        self.lasers = []
        self.cool_down_counter = 0

    def draw(self, window):
        window.blit(self.ship_img, (self.x, self.y))
        for laser in self.lasers:
            laser.draw(window)

    def move_lasers(self, vel, obj):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            elif laser.collision(obj):
                obj.health -= 10
                self.lasers.remove(laser)

    def cooldown(self):
        if self.cool_down_counter >= self.COOLDOWN:
            self.cool_down_counter = 0
        elif self.cool_down_counter > 0:
            self.cool_down_counter += 1

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1

    def get_width(self):
        return self.ship_img.get_width()

    def get_height(self):
        return self.ship_img.get_height()


class Player(Ship):
    def __init__(self, x, y, health=100):
        super().__init__(x, y, health)
        self.ship_img = YELLOW_SPACE_SHIP
        self.laser_img = YELLOW_LASER
        self.mask = pygame.mask.from_surface(self.ship_img)
        self.max_health = health

    def move_lasers(self, vel, objs):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            else:
                for obj in objs:
                    if laser.collision(obj):
                        objs.remove(obj)
                        if laser in self.lasers:
                            self.lasers.remove(laser)

    def draw(self, window):
        super().draw(window)
        self.healthbar(window)

    def healthbar(self, window):
        pygame.draw.rect(window, (255,0,0), (self.x, self.y +
self.ship_img.get_height() + 10, self.ship_img.get_width(), 10))
        pygame.draw.rect(window, (0,255,0), (self.x, self.y +
self.ship_img.get_height() + 10, self.ship_img.get_width() *
(self.health/self.max_health), 10))


class Enemy(Ship):
    COLOR_MAP = {
                "red": (RED_SPACE_SHIP, RED_LASER),
                "green": (GREEN_SPACE_SHIP, GREEN_LASER),
                "blue": (BLUE_SPACE_SHIP, BLUE_LASER)
                }

    def __init__(self, x, y, color, health=100):
        super().__init__(x, y, health)
        self.ship_img, self.laser_img = self.COLOR_MAP[color]
        self.mask = pygame.mask.from_surface(self.ship_img)

    def move(self, vel):
        self.y += vel

    def shoot(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x-20, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1


def collide(obj1, obj2):
    offset_x = obj2.x - obj1.x
    offset_y = obj2.y - obj1.y
    return obj1.mask.overlap(obj2.mask, (offset_x, offset_y)) != None

def main():
    run = True
    FPS = 60
    level = 0
    lives = 5
    main_font = pygame.font.SysFont("comicsans", 50)
    lost_font = pygame.font.SysFont("comicsans", 60)

    enemies = []
    wave_length = 5
    enemy_vel = 1

    player_vel = 5
    laser_vel = 5

    player = Player(300, 630)

    clock = pygame.time.Clock()

    lost = False
    lost_count = 0

    def redraw_window():
        WIN.blit(BG, (0,0))
        # draw text
        lives_label = main_font.render(f"Lives: {lives}", 1, (255,255,255))
        level_label = main_font.render(f"Level: {level}", 1, (255,255,255))

        WIN.blit(lives_label, (10, 10))
        WIN.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))

        for enemy in enemies:
            enemy.draw(WIN)

        player.draw(WIN)

        if lost:
            lost_label = lost_font.render("You Lost!!", 1, (255,255,255))
            WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))

        pygame.display.update()

    while run:
        clock.tick(FPS)
        redraw_window()

        if lives <= 0 or player.health <= 0:
            lost = True
            lost_count += 1

        if lost:
            if lost_count > FPS * 3:
                run = False
            else:
                continue

        if len(enemies) == 0:
            level += 1
            wave_length += 5
            for i in range(wave_length):
                enemy = Enemy(random.randrange(50, WIDTH-100), random.randrange(-
1500, -100), random.choice(["red", "blue", "green"]))
                enemies.append(enemy)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit()

        keys = pygame.key.get_pressed()
        if keys[pygame.K_x]:
            Window.close()
            os.system('python main.py')
        if keys[pygame.K_a] and player.x - player_vel > 0: # left
            player.x -= player_vel
        if keys[pygame.K_d] and player.x + player_vel + player.get_width() <
WIDTH: # right
            player.x += player_vel
        if keys[pygame.K_w] and player.y - player_vel > 0: # up
            player.y -= player_vel
        if keys[pygame.K_s] and player.y + player_vel + player.get_height() + 15
< HEIGHT: # down
            player.y += player_vel
        if keys[pygame.K_SPACE]:
            player.shoot()


        for enemy in enemies[:]:
            enemy.move(enemy_vel)
            enemy.move_lasers(laser_vel, player)

            if random.randrange(0, 2*60) == 1:
                enemy.shoot()

            if collide(enemy, player):
                player.health -= 10
                enemies.remove(enemy)
            elif enemy.y + enemy.get_height() > HEIGHT:
                lives -= 1
                enemies.remove(enemy)

        player.move_lasers(-laser_vel, enemies)

def main_menu():
    title_font = pygame.font.SysFont("comicsans", 70)
    run = True
    while run:
        WIN.blit(BG, (0,0))
        title_label = title_font.render("Press the mouse to begin...", 1,
(255,255,255))
        WIN.blit(title_label, (WIDTH/2 - title_label.get_width()/2, 350))
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                main()
    pygame.quit()


main_menu()
```

10. CONCUSION

This document will be of huge help with understanding our project as we have used a different approaches and skillsets. We have implemented everything we wanted to into the project and is working well.

REFERENCES

To conduct this project the following tools have been used :

- Pycharm and Kivy
- Pygame (Library) : https://www.pygame.org/docs/tut/newbieguide.html
- Pyrebase (Library) : https://firebase.google.com/docs
- Jarvis (Library) : https://jarvis.readthedocs.io/en/latest/

1.1 - Stackoverflow:

We have used this site for solving our different errors and implementing new functionalities.
https://stackoverflow.com/

1.2 – Javatpoint :

We have used this site for understanding basic functionalities and code of all topics.
https://www.javatpoint.com/python-tutorial

1.3 – RealPython :

We used this site for all the knowledge regarding Python.
https://realpython.com/