

PROJECT REPORT  
ON  
DESERT HAWK  
(UNDER THE PARTIAL FULFILLMENT OF THE UNIVERSITY  
FOR THE COURSE OF T.Y.BSC  
COMPUTER SCIENCE)

DESIGNED AND DEVELOPED  
BY  
Mr. VAIBHAV M. KAMTEKAR

GUIDED BY  
Ms. BHUMIKA NAKUM  
DEPARTMENT OF COMPUTER SCIENCE  
PARLE TILAK VIDYALAYA ASSOCIATION'S  
MULUND COLLEGE OF COMMERCE S.N.ROAD,  
MULUND (WEST) MUMBAI-80

UNIVERSITY OF MUMBAI  
2022-23

## **ACKNOWLEDGEMENT**

I like to extend my gratitude to Dr. Sonali Pednekar, our principal and all staff of Mulund College of Commerce for providing us moral support, conducive work environment and the much needed inspiration to complete this project on time.

I also take this opportunity to thank our Course Coordinator Dr. Reena Nagda and all the faculty members of Department of Computer Science for giving us the most needed guidance and continuous encouragement throughout the duration of the programme.

I wish to extend my deepest gratitude and special thanks to my project guide Ms. Bhumika Nakum, for giving their generous support, necessary inputs and companionship during my project work.

I would like to convey my special thanks to the Management and all the staff of the college for providing the required infrastructure and resource to enable the completion and enrichment of my project.

I am extremely grateful to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally, I thank all my fellow friends who have directly or indirectly helped me in completing my project.

VAIBHAV M. KAMTEKAR

## Plagiarism Checker X Originality Report



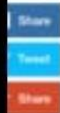
Plagiarism Quantity: 8% Duplicate

Date	Wednesday, March 01, 2023
Words	191 Plagiarized Words / Total 2290 Words
Sources	More than 28 Sources Identified.
Remarks	Low Plagiarism Detected - Your Document needs Optional Improvement.

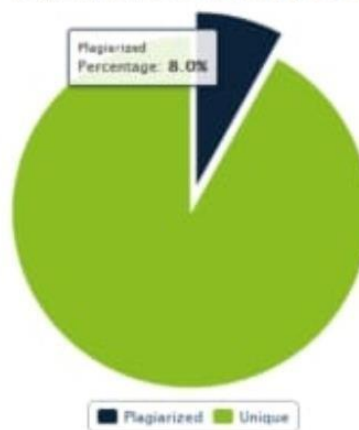
[Start](#)[Results](#)[Summary](#)

### Summary Report

are Score with friends



PlagiarismCheckerX Summary Report



**INDEX**

<b>SR. NO.</b>	<b>Topic</b>	<b>Page NO.</b>
1	TITLE	5
2	INTRODUCTION	6
3	REQUIREMENT SPECIFICATION	9
4	SYSTEM DESIGN DETAILS	11
5	SYSTEM IMPLEMENTATION	20
6	RESULTS	40
7	CONCLUSION AND FUTURE ENHANCEMENTS	44
8	REFERENCES	44
9	ANNEXURE	45

# 1. Desert Hawk

## 2.Introduction

### 2.1. Objective of the project:

- This is a 2D single player game designed for Android and PCs.
- This gaming project is easy to operate and understood.
- Talking about the gameplay, User has to dodge the hawk from enemies and destroy in order to gain score points.
- Recently, the video game market appears to be of an unprecedented stage, which means the springing up of more platforms lead to more competition.
- The video game market is not just serviced for PC, PS3 and Xbox. The mobile platforms basis on iOS, Android and Windows Phone rise sharply. As a result, “cross-platform” come into people’s eyes.

## **2.2. Description of the current system:**

- The language used for the development of this simple game is " C# language ".
- This project mainly deals with the development of a 3D game application with the Unity 4 game engine for Windows OS.
- The game engine used for the project was Unity 4. Unity 4 was developed by the UnityTechnologies.
- The theme throughout the game project is the word mechanic to relate to the actions which are taking place in games from the internal operations of animation and programming to the interactions between the environment and the player.

## **2.3. Limitations of current system:**

- Integrated Gamming PC's are required.
- Puts so much load on the devices.
- Not enough smooth gameplay.

## **2.4 Description of proposed system:**

- The game can be played without any special training.
- There are different stages available for the player.
- The bullets are fired automatically after the game
- PC controls are (Left, Right Arrow Keys formovement].
- The game will be more accurate and reliable than the current one.
- The language used for this simple game is C# .
- User has to dodge the hawk from enemies and destroy in order to gain score points.

## **2.5 Advantages of proposed system**

- Reduced time consumption.
- No extra equipments are necessary.
- No loss of records.
- Can find what you need quickly.



## **2. Requirement Specification**

### **3.1 Software Requirement**

- Operating System – Microsoft Windows2010
- Application Software – Front end: C#  
Back end: SQL
- Platform – Unity.

### **3.2 Hardware Requirement**

- Processor - 1.6 GHz
- RAM - 4GB or more
- Disk Space – 3 GB or less

### **3.3 Data Requirement:**

- Unique Id.

### **3.4 Fact finding questions:**

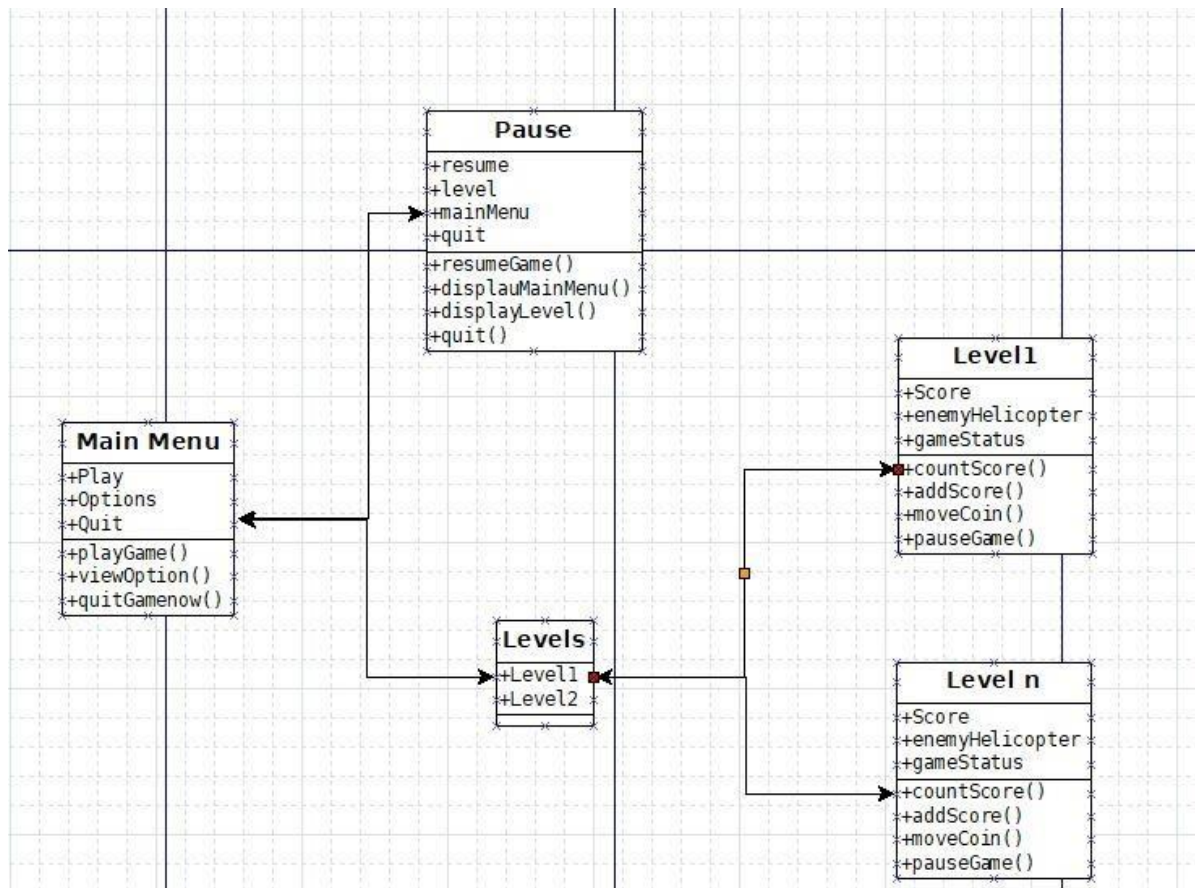
- Why do you want a game with Helicopter shooting?
- Is it expected to have any future scope?
- What hardware and software will the system product be set up on?
- How hard a game level can be?

### **3. System Design** **Details**

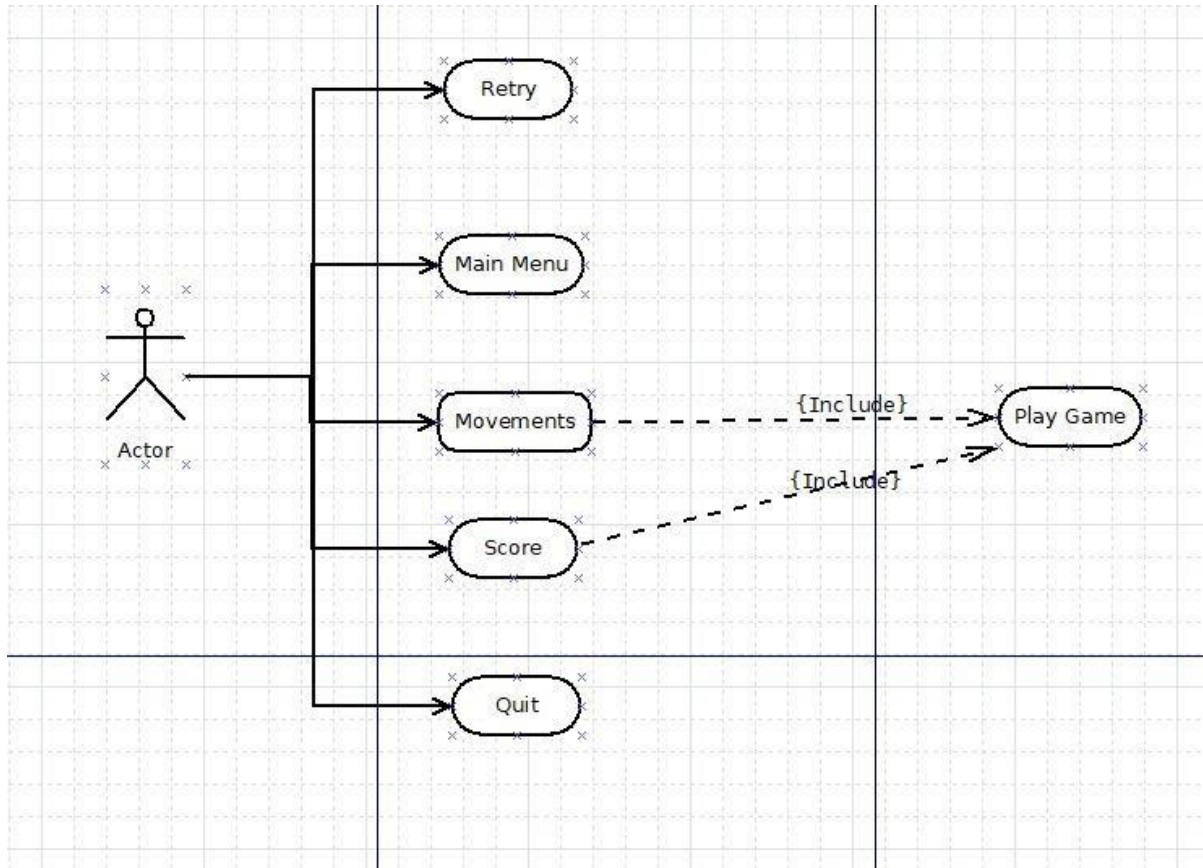
#### **4.1 Event Table:**

Sr. No	Event	Source	Activity	Response
1	Touch	User	Moves the helicopter right/left	Moves the helicopter right/left
2	Quit	User	Stops all activity	Exists the game
3	Restart	User	Restarts all the activities in game	Restarts the game from the beginning
4	Main Menu	User	Redirects to a specific page	Redirects to the Main menu

## 4.2 Class Diagram:

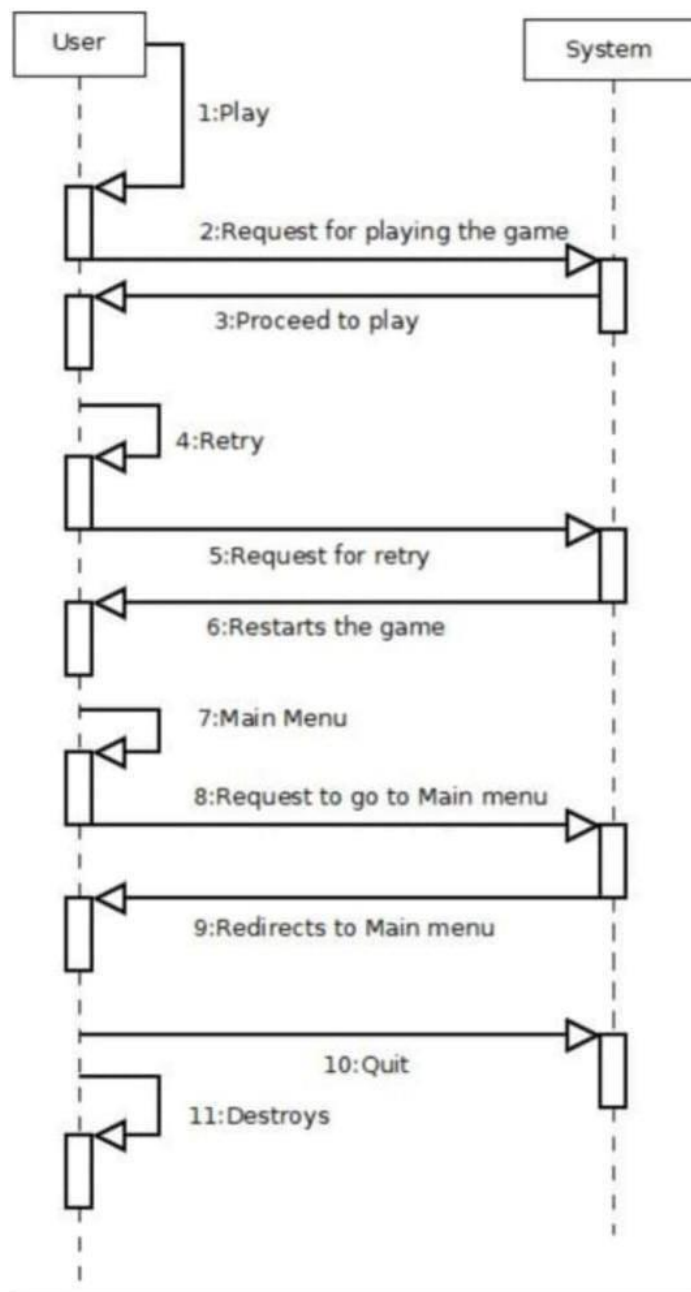


### 4.3 Use Case Diagram:

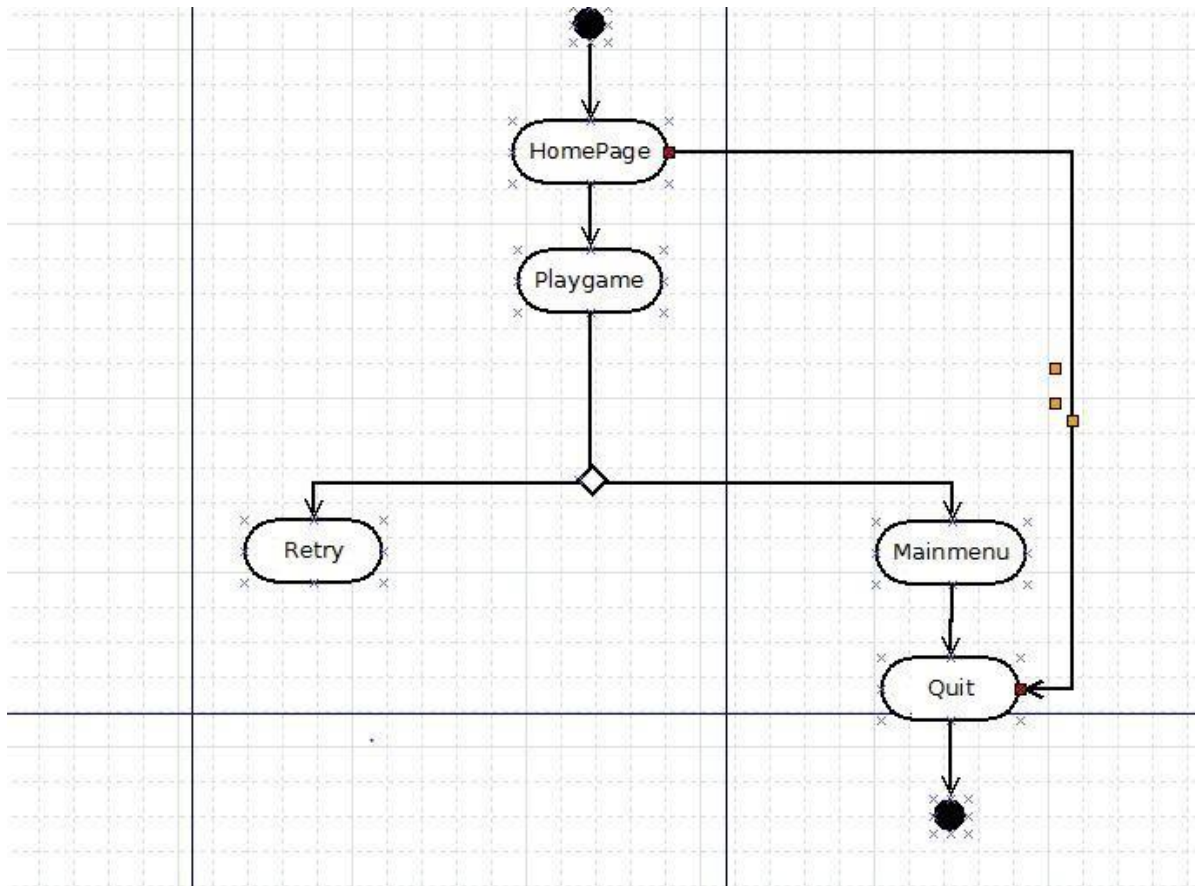


## 4.4 Sequence Diagram:

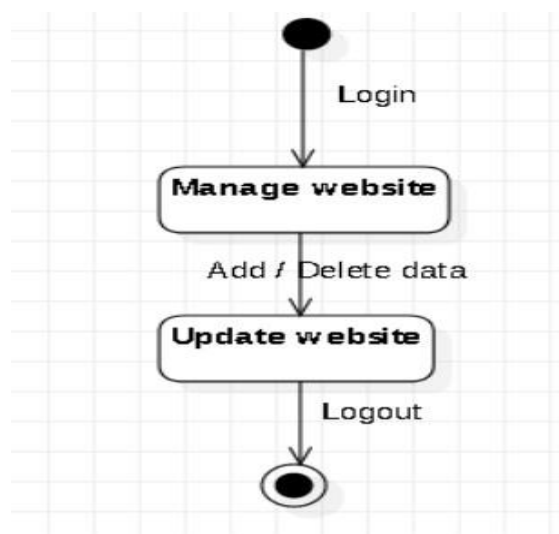
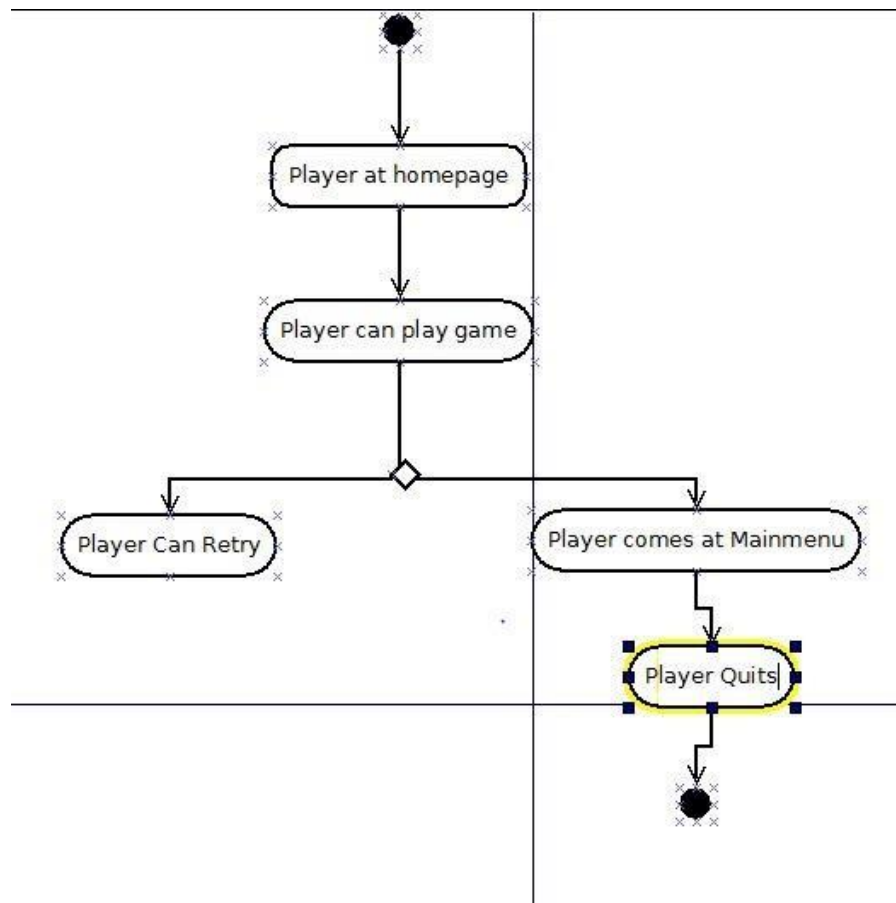
Sequence Diagram



## 4.5 Activity Diagram:



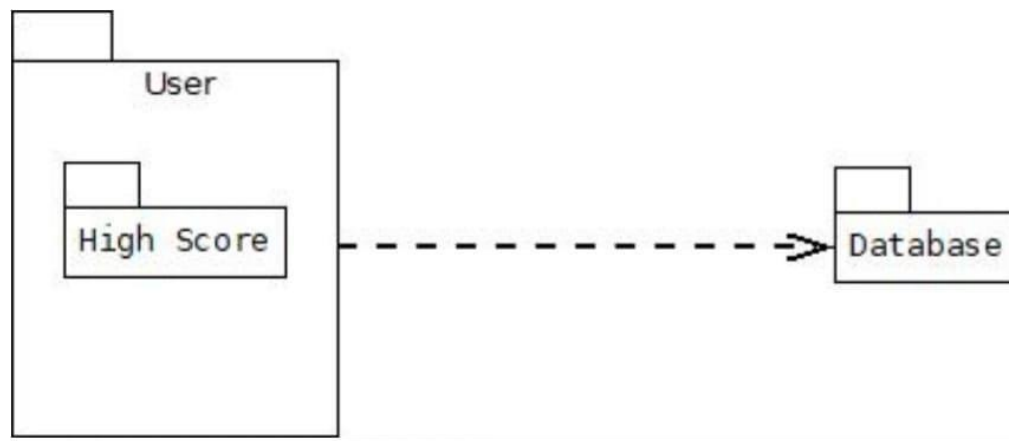
## 4.6 State Diagram:



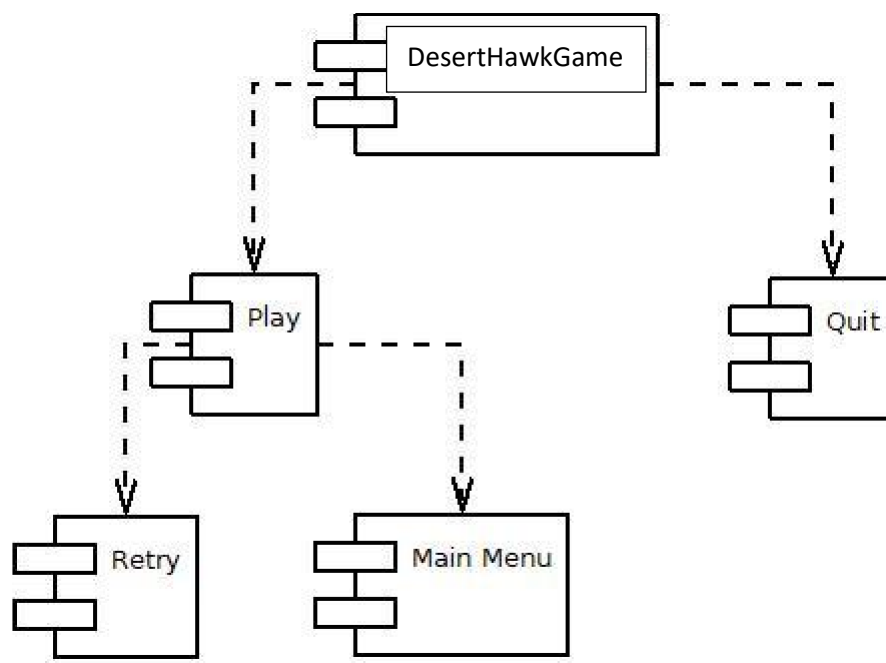
State diagram (Admin Side)



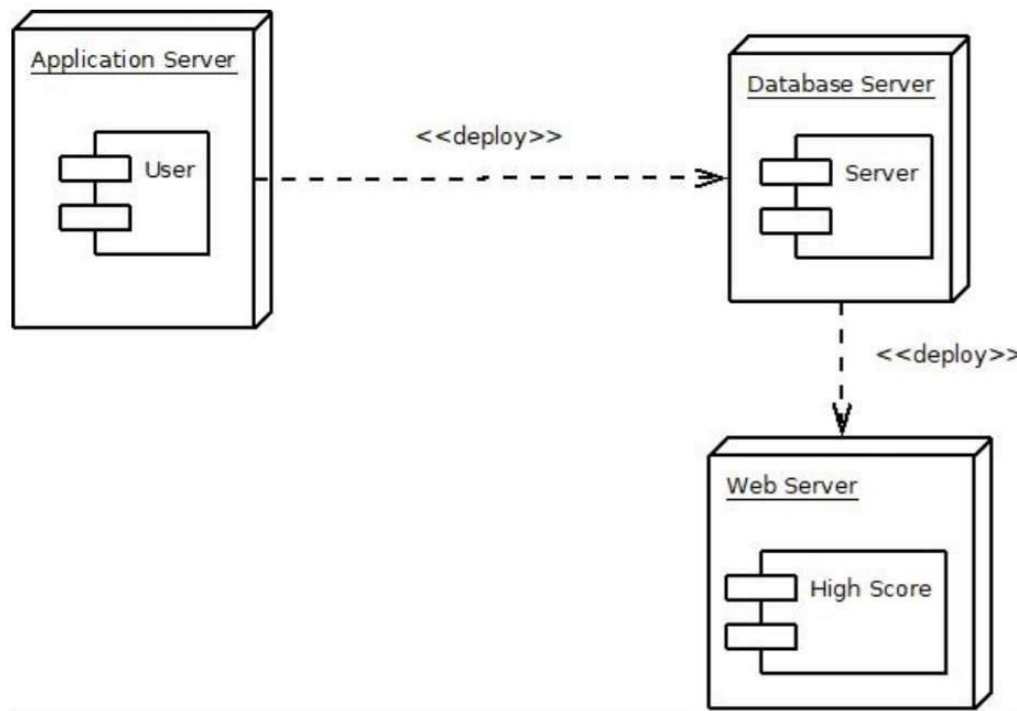
## 4.7 Package Diagram:



## 4.8 Component Diagram:





## 4.9 Deployment Diagram:



## 4.10 Database Diagram:

### 4.10.1 Players Information:

Players					
<div> <div>Players</div> <div>Segments</div> <div>Shared Group Data</div> </div> <div> <div>Players</div> <div>2 total players</div> <div>New player</div> <div>Search tips</div> </div>					
<div> <div>Most recent logins</div> <div>Search players</div> <div>Search</div> <div>Clear</div> </div>					
Type	Name	Last login ↓	Created	Country/region	VTD
	CAD5A99A6C17A776	Feb 22, 2023 4:30 AM	38 days ago	India	\$0.00
	9136887AE0E3991B	Jan 17, 2023 9:00 AM	53 days ago	India	\$0.00

### 4.10.2 Players Highscore:

Leaderboard			History (resets manually)			
Rank	Name	Value	Version	Start	End	Archive log
 0	9136887AE0E3991B	21	0	Jan 1, 2023 7:24 PM		
 1	CAD5A99A6C17A776	8				

## 5. IMPLEMENTATION

1)BossShoot.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BossShoot : MonoBehaviour {
    public GameObject bullet;
    public bool isReadyToShoot, fireBullet;
    private float fireRate, firstDelay, secondDelay;

    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
        if(isReadyToShoot){
            fireRate = Random.Range(4, 6);
            firstDelay += Time.deltaTime;
            if (firstDelay >= fireRate) {
                if (!fireBullet) {
                    fireBullet = true;
                    if (GameObject.FindGameObjectWithTag ("Player") != null) {
                        GameObject newBullet = Instantiate (bullet, new Vector3
(transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;
                        Transform target = GameObject.FindGameObjectWithTag
("Player").transform;
                        newBullet.GetComponent<Rigidbody2D> ().velocity =
(target.position - transform.position).normalized * 5f;
                    } else {
                        GameObject newBullet = Instantiate (bullet, new Vector3
(transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;
                        newBullet.GetComponent<Rigidbody2D> ().velocity = new
Vector2 (Random.Range(-1f, 1f), -2f);
                    }
                }
            }
            if (fireBullet) {
                secondDelay += Time.deltaTime;
                if (secondDelay >= 0.5f) {
                    secondDelay = 0;
                }
            }
        }
    }
}
```

```

        firstDelay = 0;
        if (GameObject.FindGameObjectWithTag ("Player") != null)
        {
            GameObject newBullet = Instantiate (bullet, new
Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;
            Transform target =
GameObject.FindGameObjectWithTag ("Player").transform;
            newBullet.GetComponent<Rigidbody2D> ().velocity
= (target.position - transform.position).normalized * 5f;
        } else {
            GameObject newBullet = Instantiate (bullet, new
Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;
            newBullet.GetComponent<Rigidbody2D> ().velocity
= new Vector2 (Random.Range(-0.1f, 0.1f), -2f);
        }
        fireBullet = false;
    }
}
}
}
}

```

```
2)BossHealth.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class BossHealth : MonoBehaviour {
    public int maxHealth;
    public Slider health;
    public bool invulnerable;
    public GameObject explode, coin;
    void Awake(){
        InitializeBossVariables ();
    }
    // Use this for initialization
    void Start () {
    }
    void InitializeBossVariables(){
        health.maxValue = maxHealth;
        health.value = maxHealth;
        invulnerable = true;
    }
}
```

```

    }
    public void Health(int damage){
    if (!invulnerable) {
        if (!health.gameObject.activeInHierarchy) {
            health.gameObject.SetActive (true);
        }if (health.value > 0) {
            health.value -= damage;
        }
        if (health.value == 0) {
            BossDestroyed ();
        }
    }
}

void BossDestroyed(){
    Destroy (gameObject);
    GameObject.FindGameObjectWithTag
("Spawner").transform.GetComponent<EnemySpawner> ().active = true;
    GameObject.FindGameObjectWithTag
("Spawner").transform.GetComponent<EnemySpawner> ().isBossReady = false;
    if(GameController.instance != null && MusicController.instance != null){
        if(GameController.instance.isMusicOn){
            MusicController.instance.audioSource.PlayOneShot
(MusicController.instance.bossExplode);
        }
    }
    Instantiate (explode, transform.position, Quaternion.identity);
    for (int i = 0; i < 5; i++) {
        Instantiate (coin, transform.position, Quaternion.identity);
    }
}
}

```

### 3)EnemyHealth.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class EnemyHealth : MonoBehaviour {
    public Slider health;
    public int maxHealth;
    public GameObject explosion, coin;
}

```

```

private bool spawn;
void Awake(){
    health.maxValue = maxHealth;
    health.value = maxHealth;
}
// Use this for initialization
void Start () {
}
public void Health(int damage){
    if(!health.gameObject.activeInHierarchy){
        health.gameObject.SetActive (true);
    }
    if(health.value > 0){
        health.value -= damage;
    }
    if(health.value == 0){
        Destroy (gameObject);
        if(!spawn){
            spawn = true;
            Instantiate (explosion, transform.position, Quaternion.identity);
            Instantiate (coin, transform.position, Quaternion.identity);
        }
    }
}
}

```

#### 4)EnemySpawner.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemySpawner : MonoBehaviour {
    public GameObject[] enemies;
    public GameObject[] boss;
    public int count, maxCount;
    public bool active, isBossReady;
    public float time;
    private float delay;
    private float maxTop;

    void Awake(){
    }
}

```

```

// Use this for initialization
void Start () {
    InitializeVariables ();
    LimitBounds ();
}
// Update is called once per frame
void Update () {
    if(GameplayController.instance.gameInProgress){
        LimitBounds ();
        SpawnEnemies ();
        CheckedEnemies ();
    }
}
void SpawnEnemies(){
    if (active) {
        if (count != 0) {
            delay = 3;
            Vector3[] position = new[] {
                new Vector3 (-2, maxTop, 0),
                new Vector3 (-1, maxTop, 0),
                new Vector3 (0, maxTop, 0),
                new Vector3 (1, maxTop, 0),
                new Vector3 (2, maxTop, 0)
            };
            time += Time.deltaTime;
            if (time > delay) {
                delay = Random.Range (3, 8);
                int randomEnemy = Random.Range (0, position.Length);
                time = 0;
                for (int i = 0; i < position.Length; i++) {
                    if (i != randomEnemy) {
                        Instantiate (enemies [0], position [i], Quaternion.Euler (0, 0, 180));
                    } else {
                        Instantiate (enemies [1], position [i], Quaternion.Euler (0, 0, 180));
                    }
                }
                count--;
            }
        } else {
            active = false;
        }
    }
}

```



```

    } else {
        count = maxCount;
    }
}
void LimitBounds(){
    Vector3 topBoundary = Camera.main.ViewportToWorldPoint (new Vector3 (0, 1, 0));
    maxTop = topBoundary.y;
}
void CheckedEnemies(){
    if(!active){
        if (GameObject.FindGameObjectWithTag("Enemy") == null) {
            if(!isBossReady){
                isBossReady = true;
                GameplayController.instance.ShowWarning();
            }
        }
    }
}
}
public void SpawnBoss(){
    Instantiate (boss[0], new Vector3 (0, maxTop + 3, 0), Quaternion.Euler(0, 0, 180));
}
public void InitializeVariables(){
    count = maxCount;
    time = 0f;
    active = true;
    isBossReady = false;
}
}

```

##### 5)GameController.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
public class GameController : MonoBehaviour {
    public static GameController instance;
    public bool isGameStartedFirstTime;
    public bool isMusicOn;
    public int coins;

```

```

public int weaponLevel;
private GameData data;
void Awake(){
    CreateInstance ();
    InitializeGameVariables ();
}
// Use this for initialization
void Start () {
}
// Update is called once per frame
void Update () {
}
void CreateInstance(){
    if (instance != null) {
        Destroy (gameObject);
    } else {
        instance = this;
        DontDestroyOnLoad (gameObject);
    }
}
void InitializeGameVariables(){
    Load ();
    if (data != null) {
        isGameStartedFirstTime = data.GetIsGameStartedFirstTime ();
    } else {
        isGameStartedFirstTime = true;
    }

    if (isGameStartedFirstTime) {
        isGameStartedFirstTime = false;
        isMusicOn = true;
        coins = 0;
        weaponLevel = 1;
        data = new GameData ();
        data.SetIsGameStartedFirstTime (isGameStartedFirstTime);
        data.SetIsMusicOn (isMusicOn);
        data.SetCoins (coins);
        data.SetWeaponLevel (weaponLevel);
        Save ();

        Load ();
    }
}

```

```

    } else {
        isGameStartedFirstTime = data.GetIsGameStartedFirstTime ();
        isMusicOn = data.GetIsMusicOn ();
        coins = data.GetCoins ();
        weaponLevel = data.GetWeaponLevel ();
    }
}

public void Save(){
    FileStream file = null;
    try{
        BinaryFormatter bf = new BinaryFormatter();
        file = File.Create(Application.persistentDataPath + "/data.dat");

        if(file != null){
            data.SetIsGameStartedFirstTime(isGameStartedFirstTime);
            data.SetIsMusicOn(isMusicOn);
            data.SetCoins (coins);
            data.SetWeaponLevel (weaponLevel);
            bf.Serialize(file, data);
        }
    }catch(Exception e){
        Debug.LogException (e, this);
    }finally{
        if(file != null){
            file.Close ();
        }
    }
}

public void Load(){
    FileStream file = null;
    try{
        BinaryFormatter bf = new BinaryFormatter();
        file = File.Open(Application.persistentDataPath + "/data.dat", FileMode.Open);
        data = bf.Deserialize(file) as GameData;
    }catch(Exception e){
        Debug.LogException (e, this);
    }finally{
        if(file != null){
            file.Close ();
        }
    }
}

```

```

    }
}
[Serializable]
class GameData{
    private bool isGameStartedFirstTime;
    private bool isMusicOn;
    private int coins;
    private int weaponLevel;
    public void SetIsGameStartedFirstTime(bool isGameStartedFirstTime){
        this.isGameStartedFirstTime = isGameStartedFirstTime;
    }
    public bool GetIsGameStartedFirstTime(){
        return this.isGameStartedFirstTime;
    }
    public void SetIsMusicOn(bool isMusicOn){
        this.isMusicOn = isMusicOn;
    }
    public bool GetIsMusicOn(){
        return this.isMusicOn;
    }
    public void SetCoins(int coins){
        this.coins = coins;
    }
    public int GetCoins(){
        return this.coins;
    }
    public void SetWeaponLevel(int weaponLevel){
        this.weaponLevel = weaponLevel;
    }
    public int GetWeaponLevel(){
        return this.weaponLevel;
    }
}

```

6)GameplayControler.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

```

```

public class GameplayController : MonoBehaviour {

```

```

    public static GameplayController instance;
    public playfabManager playfabmanager;

    public GameObject player, statusTab, deployButton, upgradeButton, coinTab,
    upgradMenuPanel, arrowIcon, gameoverPanel, pausePanel, cloud, pauseMenu, pauseButton,
    hsPanel;

    public bool gameInProgress, startMoving;

    public Text warningText, coinText, currentCoinText, currentWeaponLevel,
    nextWeaponLevel, maxLevelText, activeWeaponLevelText, priceText, scoreText,
    gameoverCoinText;

    public int coins, price;
    public Animator upgrade;
void Awake(){
    CreateInstance ();
}
// Use this for initialization
void Start () {
    if(MusicController.instance != null && GameController.instance != null){
        if (GameController.instance.isMusicOn) {
            MusicController.instance.PlayGameplaySound ();
        } else {
            MusicController.instance.StopAllSound ();
        }
    }
    pauseMenu.SetActive(false);
    hsPanel.SetActive(false);
    InitializeGameplayVariables ();
}
// Update is called once per frame
void Update () {
    if(Input.GetKeyDown(KeyCode.Escape)){
        PausePanel ();
    }
}

public void HighScore()
{
    hsPanel.SetActive(true);
    gameoverPanel.SetActive(false);
}

    public void PauseGame()
    {
        pauseMenu.SetActive(true);

```

```

Time.timeScale = 0f;
}

void InitializeGameplayVariables(){
    if(GameController.instance != null){
        InitializeCoins ();
        InitializeWeaponLevel ();
        InitializeUpgradePrice ();
    }
    InitializePlayer ();
    statusTab.SetActive (true);
    deployButton.SetActive (true);
    upgradeButton.SetActive (true);
    coinTab.SetActive (false);
    gameoverPanel.SetActive (false);
    startMoving = false;
}

void InitializeCoins(){
    coins = 0;
    coinText.text = coins.ToString ();
    currentCoinText.text = GameController.instance.coins.ToString("N0");
}

void InitializeWeaponLevel(){
    if (GameController.instance.weaponLevel != 5) {
        currentWeaponLevel.text = "LVL " +
GameController.instance.weaponLevel.ToString ();
        int newLevel = GameController.instance.weaponLevel + 1;
        nextWeaponLevel.text = "LVL " + newLevel.ToString ();
        maxLevelText.gameObject.SetActive (false);
        activeWeaponLevelText.text = GameController.instance.weaponLevel.ToString();
    } else {
        arrowIcon.SetActive (false);
        currentWeaponLevel.gameObject.SetActive (false);
        nextWeaponLevel.gameObject.SetActive (false);
        maxLevelText.gameObject.SetActive (true);
        activeWeaponLevelText.text = "MAX";
    }
}

void InitializeUpgradePrice(){
    if (GameController.instance.weaponLevel != 5) {
        price = GetPrice (GameController.instance.weaponLevel);
        priceText.text = price.ToString ();
    }
}

```

```
} else {  
    priceText.transform.GetChild (0).gameObject.SetActive (false);  
    priceText.text = "MAX";  
}  
  
}  
private int GetPrice(int level){  
    int nextPrice = 0;  
  
    switch(level){  
  
        case 1:  
            nextPrice = 500;  
            break;  
  
        case 2:  
            nextPrice = 1200;  
            break;  
  
        case 3:  
            nextPrice = 2500;  
            break;  
  
        case 4:  
            nextPrice = 5000;  
            break;  
    }  
  
    return nextPrice;  
}  
  
void InitializePlayer(){  
    Instantiate (player, new Vector3 (0, -3.5f, 0), Quaternion.identity);  
}  
  
void CreateInstance(){  
    if(instance == null){  
        instance = this;  
    }  
}
```

```

void ClearAllEnemies(){
    GameObject[] enemies = GameObject.FindGameObjectsWithTag ("Enemy");
    GameObject[] bossbullets = GameObject.FindGameObjectsWithTag ("Boss Bullet");
    GameObject[] coins = GameObject.FindGameObjectsWithTag ("Coin");
    GameObject boss = GameObject.FindGameObjectWithTag ("Boss");

    if(boss != null){
        Destroy (boss);
    }

    if(enemies != null){
        foreach (GameObject enemy in enemies) {
            Destroy (enemy);
        }
    }

    if(bossbullets != null){
        foreach (GameObject bossBullet in bossbullets) {
            Destroy (bossBullet);
        }
    }

    if(coins != null){
        foreach (GameObject coin in coins) {
            Destroy (coin);
        }
    }

}

public void ShowWarning(){
    StartCoroutine (SpawnBossTime());
}

IEnumerator SpawnBossTime(){
    warningText.gameObject.SetActive (true);
    warningText.transform.GetComponent<WarningText> ().PlayWarning ();
    yield return new WaitForSeconds (5f);
    warningText.gameObject.SetActive (false);
    warningText.transform.GetComponent<WarningText> ().StopWarning ();
    GameObject.FindGameObjectWithTag
("Spawner").transform.GetComponent<EnemySpawner> ().SpawnBoss ();
}

```



```

    }
    public void DeployButton(){
        statusTab.SetActive (false);
        deployButton.SetActive (false);
        upgradeButton.SetActive (false);
        coinTab.SetActive (true);
        Instantiate (cloud, new Vector3(0, 0, 0), Quaternion.identity);
        startMoving = true;
    }
    public void UpgradeButton(){
        upgradMenuPanel.SetActive (true);
    }
    public void CloseButton(){
        upgradMenuPanel.SetActive (false);
    }
    public void PurchaseUpgrade(){
        if(GameController.instance != null){
            if(GameController.instance.weaponLevel != 5){
                if (GameController.instance.coins >= price) {
                    GameController.instance.coins -= price;
                    GameController.instance.weaponLevel += 1;

                    GameController.instance.Save ();

                    InitializeCoins ();
                    InitializeUpgradePrice ();
                    InitializeWeaponLevel ();
                } else {
                    upgrade.Play ("Upgrade Button Anim");
                }
            }
        }
    }
    public void UpdateCoins(){
        coins += 1;
        coinText.text = coins.ToString ()
    }

    public void GameOver(){
        gameoverPanel.SetActive (true);
        gameoverCoinText.text = coins.ToString ("N0");
        scoreText.text = coins.ToString ("N0");
    }

```

```

//For LeaderBoard
    string gameOverCoin = gameOverCoinText.text;
int gc = int.Parse(gameOverCoin);
playfabmanager.SendLeaderboard(gc);
    //playfabmanager.GetLeaderboard();
    //
if (GameController.instance.coins != null){
    GameController.instance.coins += coins;
    GameController.instance.Save ();
}
}

public void RetryButton(){
    InitializeGameplayVariables ();
    ClearAllEnemies ();
    hsPanel.SetActive(false);
    gameInProgress = false;
    GameObject.FindGameObjectWithTag
("Spawner").transform.GetComponent<EnemySpawner> ().active = false;
}
public void InitializeSpawners(){
    GameObject.FindGameObjectWithTag
("Spawner").transform.GetComponent<EnemySpawner> ().InitializeVariables ();
    gameInProgress = true;
}
public void ResumeButton(){
    pausePanel.SetActive (false);
    Time.timeScale = 1;
    gameInProgress = true;
}
public void QuitButton(){
    SceneManager.LoadScene ("Main Menu");
    Time.timeScale = 1;
}
void PausePanel(){
    if (gameInProgress && !pausePanel.activeInHierarchy) {
        pausePanel.SetActive (true);
        Time.timeScale = 0;
        gameInProgress = false;
    } else if(!gameInProgress && !pausePanel.activeInHierarchy) {
        SceneManager.LoadScene ("Main Menu");
    }
}

```

```

    }
}

```

#### 7)MainMenuController.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class MainMenuController : MonoBehaviour {
    public GameObject quitPanel;
    // Use this for initialization
    void Start () {
        if(GameController.instance && MusicController.instance){
            if (GameController.instance.isMusicOn) {
                MusicController.instance.PlayBackgroundSound ();
            } else {
                MusicController.instance.StopAllSound ();
            }
        }
    }
    void Update(){
        if(Input.GetKeyDown(KeyCode.Escape)){
            if (!quitPanel.activeInHierarchy) {
                quitPanel.SetActive (true);
            } else {
                quitPanel.SetActive (false);
            }
        }
    }
    public void StartButton(){
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
    public void YesButton(){
        Application.Quit ();
    }
    public void NoButton(){
        if (quitPanel.activeInHierarchy) {
            quitPanel.SetActive (false);
        }
    }
}

```

```

}
8)MusicController.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MusicController : MonoBehaviour {
    public static MusicController instance;
    [HideInInspector]
    public AudioSource audioSource;
    public AudioClip background, gameplay, explode, coin, bossExplode
    void Awake(){
        audioSource = GetComponent<AudioSource> ();
        CreateInstance ();
    }
    // Use this for initialization
    void Start () {
    }
    void CreateInstance(){
        if (instance != null) {
            Destroy (gameObject);
        } else {
            instance = this;
            DontDestroyOnLoad (gameObject);
        }
    }
    public void PlayBackgroundSound(){
        if(background){
            audioSource.clip = background;
            audioSource.volume = 0.5f;
            audioSource.loop = true;
            audioSource.Play ();
        }
    }

    public void PlayGameplaySound(){
        if(gameplay){
            audioSource.clip = gameplay;
            audioSource.volume = 0.5f;
            audioSource.loop = true;
            audioSource.Play ();
        }
    }
}

```

```

    }
}
public void StopAllSound(){
    if(audioSource.isPlaying){
        audioSource.Stop ();
    }
}
public void PlayerDeath(){
    if(explode){
        audioSource.PlayOneShot (explode);
    }
}
}
}

```

#### 9)PlayerBullet.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class PlayerBullet : MonoBehaviour {
    public int damage;
    public GameObject hit;
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
    }
    void OnTriggerEnter2D(Collider2D collider){
        if(collider.CompareTag("Enemy")){
            Instantiate (hit, transform.position, Quaternion.identity);
            collider.transform.GetComponent<EnemyHealth> ().Health (damage);
            Destroy (gameObject);
        }

        if(collider.CompareTag("Boss")){
            Instantiate (hit, transform.position, Quaternion.identity);
            collider.transform.GetComponent<BossHealth> ().Health (damage);
            Destroy (gameObject);
        }
    }
}
}

```

```

10)PlayfabManager.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using PlayFab;
using PlayFab.ClientModels;
using UnityEngine.UI;
using Newtonsoft.Json;

public class playfabManager : MonoBehaviour
{
    public GameObject rowPrefab;
    public Transform rowsParent;
    // Start is called before the first frame update
    void Start()
    {
        Login();
    }

    void Login()
    {
        var request = new LoginWithCustomIDRequest
        {
            CustomId = SystemInfo.deviceUniqueIdentifier,
            CreateAccount = true
        };
        PlayFabClientAPI.LoginWithCustomID(request, OnSuccess, OnError);
    }
    void OnSuccess(LoginResult result)
    {
        Debug.Log("Successful login/account create!");
    }
    void OnError(PlayFabError error)
    {
        Debug.Log("Error while logging in/creating account!");
        Debug.Log(error.GenerateErrorReport());
    }
    public void SendLeaderboard(int score)
    {
        var request = new UpdatePlayerStatisticsRequest
        {

```

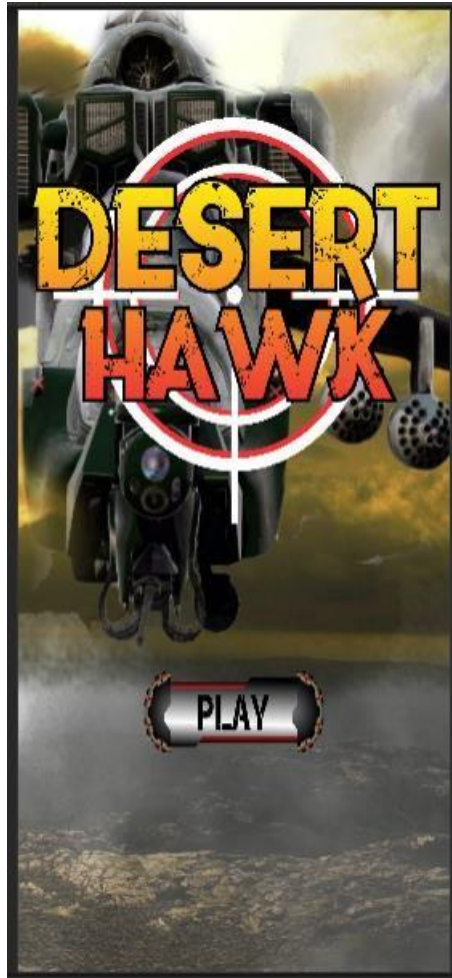
```

        Statistics = new List<StatisticUpdate>
        {
            new StatisticUpdate
            {
                StatisticName = "PlatformScore",
                Value = score
            }
        }
    };
    PlayFabClientAPI.UpdatePlayerStatistics(request, OnLeaderboardUpdate, OnError);
}
void OnLeaderboardUpdate(UpdatePlayerStatisticsResult result)
{
    Debug.Log("Successfull leaderboard sent");
}
//onleaderBoard Method
public void GetLeaderboard()
{
    var request = new GetLeaderboardRequest
    {
        StatisticName = "PlatformScore",
        StartPosition = 0,
        MaxResultsCount = 10
    };
    PlayFabClientAPI.GetLeaderboard(request, OnLeaderboardGet, OnError);
}
void OnLeaderboardGet(GetLeaderboardResult result)
{
    foreach (var item in result.Leaderboard)
    {
        GameObject newGo = Instantiate(rowPrefab, rowsParent);
        Text[] texts = newGo.GetComponentsInChildren<Text>();
        texts[0].text = item.StatValue.ToString();
        Debug.Log(item.StatValue);
    }
}
}

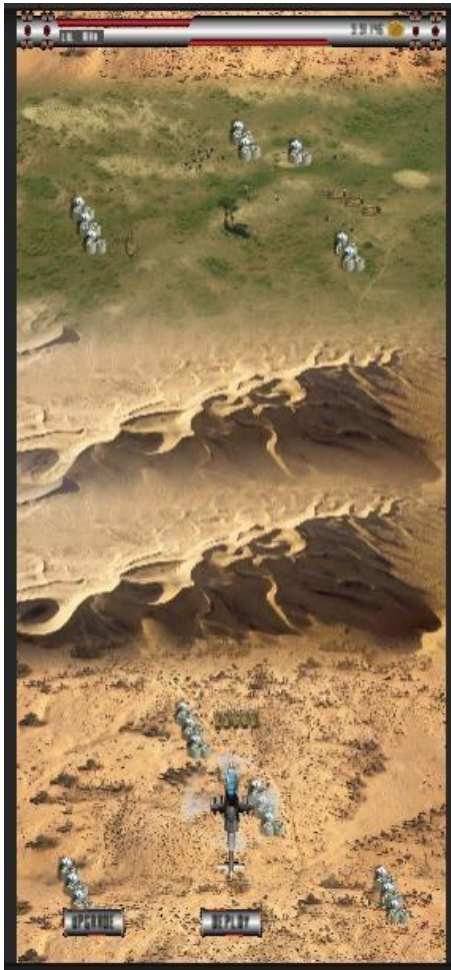
```

## 6. Result:

Screenshots:











## 7. Conclusion and future scope

### A. Future enhancement:

- Adding a better UI.
- Adding new features.
- Changing the background and adding levels.
- User can change the helicopter.
- Increasing Difficulty level.

### B. Conclusion

The black book has been created with the objective of providing a detail information about the game its features on what system it works and how it was developed. Also it provides the information regarding the specification required to run the game on android or desktop. It also provides us with information regarding how the bird is controlled and what future enhancements can be made.

## 8. Reference

- [www.google.com](http://www.google.com)
- [www.youtube.com](http://www.youtube.com)
- [www.wikipedia.com](http://www.wikipedia.com)

## 9. Annexure

Serial Number	Name of figure	Page Number
4.2	Class Diagram	12
4.3	Use Case Diagram	13
4.4	Sequence Diagram	14
4.5	Activity Diagram for user	15
4.6	State Diagram for user	16
4.7	Package Diagram	17
4.8	Component Diagram	17
4.9	Deployment Diagram	18

## 10. Table List

Table number	Name of Table	Page Number
4.1	Event table	11
4.10.1	Database design	19
4.10.2	Database design for user's high score	19