

A
PROJECT REPORT
ON

Sizzle N Serve

PC GAME DESIGNED AND DEVELOPED

BY

MR.ALIRAZA SOHEL KHAN

TYBSC CS 30

2022-2023

UNDER

THE GUIDANCE OF

PROF. ANUSHKA PADHYE

SUBMITTED IN PARTIAL FULFILLMENT OF
ACADEMIC PROJECT

[BACHELOR OF SCIENCE COMPUTER SCIENCE]

[UNIVERSITY OF MUMBAI]

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It tells us how much we rely on the efforts and goodwill of others. It gives me immense pleasure to present this report to fulfill my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who helped me complete this project successfully.

I take this opportunity to express my profound gratitude to management of Royal College Of Arts, Commerce & Science for giving me this opportunity to accomplish this project work.

A special vote of thanks to **Prof. Anushka Padhye**, our professor & project guide, for their most sincere, useful and encouraging contribution throughout the project span.

Finally, I would like to thank the entire Computer Science department who directly or indirectly helped me in the completion of this project & my family without their support, motivation & encouragement this would not have been possible.

ALIRAZA SOHEL KHAN

DECLARATION

DECLARATION

I “**Mr. ALIRAZA SOHEL KHAN** ” hereby declare that I myself have completed the project under the guidance of **Prof. Anushka Padhye** I on my own have designed the system and have done all the programming required.

It may require some modifications in the future as per the user’s requirements. From practical implementation point of view flexibility in the changes have incorporated in the package.

I am sure that I can do any kind of modification suggested while practical implementation by modifying file design or the program code if necessary.

ALIRAZA SOHEL KHAN

CERTIFICATE



ROYAL COLLEGE

OF ARTS, SCIENCE & COMMERCE

Empowerment through Value Education

DEPARTMENT OF COMPUTER SCIENCE

Class: **TYBSc**

Roll. No. **30**

Seat No: _____

Certificate

Certified that **Mr. ALIRAZA SOHEL KHAN** of T.Y.BSc -CS Semester-VI has successfully completed the project as prescribed by the University of Mumbai on **SizzleNServe PC Game** as partial fulfilment of requirement for completing Bachelor's Degree in Computer Science during the academic year 2022-2023.

Signature of Project Guide

Date: _____

Signature of Examiner

Date: _____

H.O.D

Dept. of Computer Science

ABOUT PROJECT

OBJECTIVES

Sizzle N Serve is a cooking game that challenges players to cook and deliver as many randomly generated recipes as possible within a given time frame. The game is designed to be engaging and challenging for players of all ages and skill levels.

Features:

1. **Recipe Generation:** The game automatically generates new recipes at regular intervals, keeping the game fresh and challenging.
2. **Ingredient and Cookware Selection:** Players must choose the correct ingredients and cookware for each recipe in order to prepare it correctly and quickly.
3. **Preparation and Cooking:** Players must follow the recipe instructions to prepare and cook the ingredients in the correct order and for the correct amount of time.
4. **Plating and Serving:** Once the dish is cooked, players must plate it and serve it to the customer in a timely and presentable manner.
5. **Scoring System:** The game keeps track of the speed and accuracy of the player's preparation and delivery of each dish, awarding points accordingly.

Objectives:

1. To provide an engaging and entertaining gaming experience for players of all ages and skill levels.
2. To challenge players to cook and deliver a variety of randomly generated recipes within a given time frame.
3. To promote the development of culinary skills and knowledge among players by exposing them to different types of ingredients and cooking techniques.
4. To encourage players to develop problem-solving skills by requiring them to choose the correct ingredients and cookware for each recipe and to adapt to changing game conditions.
5. To foster a sense of competition and achievement among players by awarding points and rewards based on the speed and accuracy of their performance.
6. To provide a learning experience that is both fun and educational, helping players to acquire new knowledge and skills related to cooking and food preparation.

INDEX

Sr.No	Topic	Page No.
I	PreliminaryInvestigation	12
1.	Present System and its advantages	13
2.	System Requirements	14
3.	Phase Title	15
4.	Gantt Chart	17
II	System Analysis	18
1.	Event Table	19
2.	Use Case Diagram	20
3.	ERD	21
4.	Class Diagram	22
III	System Design	23
1.	Sequence Diagram	24
IV	System Coding	25
1	System Coding	25
2	Test Cases	45
3	Form Layouts	48
V	Future Enhancements	52
VI	Plagiarism Report	53
VII	References and Bibliography	56

PRELIMINARY INVESTIGATION

PRESENT SYSTEM AND ITS ADVANTAGES

Present System:

Currently, there are several cooking games available on various platforms, including mobile devices and gaming consoles. These games typically involve players selecting and preparing ingredients according to recipe instructions, cooking those using virtual appliances, and serving them to virtual customers. Some of the popular cooking games include Cooking Dash, Cooking Craze, and Cooking Fever.

Advantages:

1. Entertainment: Cooking games provide an entertaining and engaging way for players to learn about different types of ingredients and cooking techniques while having fun.
2. Skill development: Cooking games can help players develop their culinary skills and knowledge by exposing them to new recipes and cooking methods.
3. Problem-solving: Cooking games require players to think critically and solve problems by choosing the correct ingredients and cookware, managing their time effectively, and adapting to changing game conditions.
4. Accessibility: Cooking games are widely available on various platforms, including mobile devices and gaming consoles, making them easily accessible to a large audience.
5. Community building: Cooking games can help build a strong and loyal community of players who share a common interest in cooking and food preparation.
6. Revenue generation: Cooking games can generate revenue through in-game advertising and microtransactions, allowing game developers to continue improving and expanding the game over time.

SYSTEM REQUIREMENTS

Technology Used:

Platform :	Windows
Game Engine :	Unity Engine
The Language Used :	C#
Code Editor :	Visual Studio

System Requirement:

OS :	Windows 11
RAM :	4 GB
PROCESSOR :	Intel i5
GPU :	Geforce GTX 750 Ti

PHASE TITLE

TYBSC Computer Science Semester 6
2022-2023

Phase Title	Expected Date of Completion	Actual Time of Completion with Guide's Signature	Remarks
I. Preliminary Investigation			
(i) Organizational Overview			
(ii) Present System and its advantages			
(iii) System Requirements			
(iv) Feasibility Study			
(v) Fact Finding Methods			
(vi) Phase Title			
(vii) Gantt Chart			
II. System Analysis			
(i) Event Table			
(ii) Use Case Diagram			
(iii) ERD			
(iv) Class Diagram			
III. System Design			
(i) Sequence Diagram			
IV. System Coding			
(i) System Coding			
(ii) Test Cases			



(iii) Form Layouts	}		
(iv) Report Layouts			
V. Future Enhancements			
VI. Reference and Bibliography			

Student Name: ALIRAZA SOHEL KHAN

Roll No: 30

GANTT CHART

TYBSc Computer Science Semester 6 Project Gantt Chart		Time Requirement	Year 2022-23 Weeks												
			December				January				February				
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	
I	Preliminary Investigation	Estimated													
		Actual													
II	Requirement Gathering	Estimated													
		Actual													
III	System Analysis	Estimated													
		Actual													
IV	System Design	Estimated													
		Actual													
V	System Coding	Estimated													
		Actual													
VI	Testing	Estimated													
		Actual													
VII	Implementation	Estimated													
		Actual													
VIII	Deployment	Estimated													
		Actual													

Estimated	
Actual	

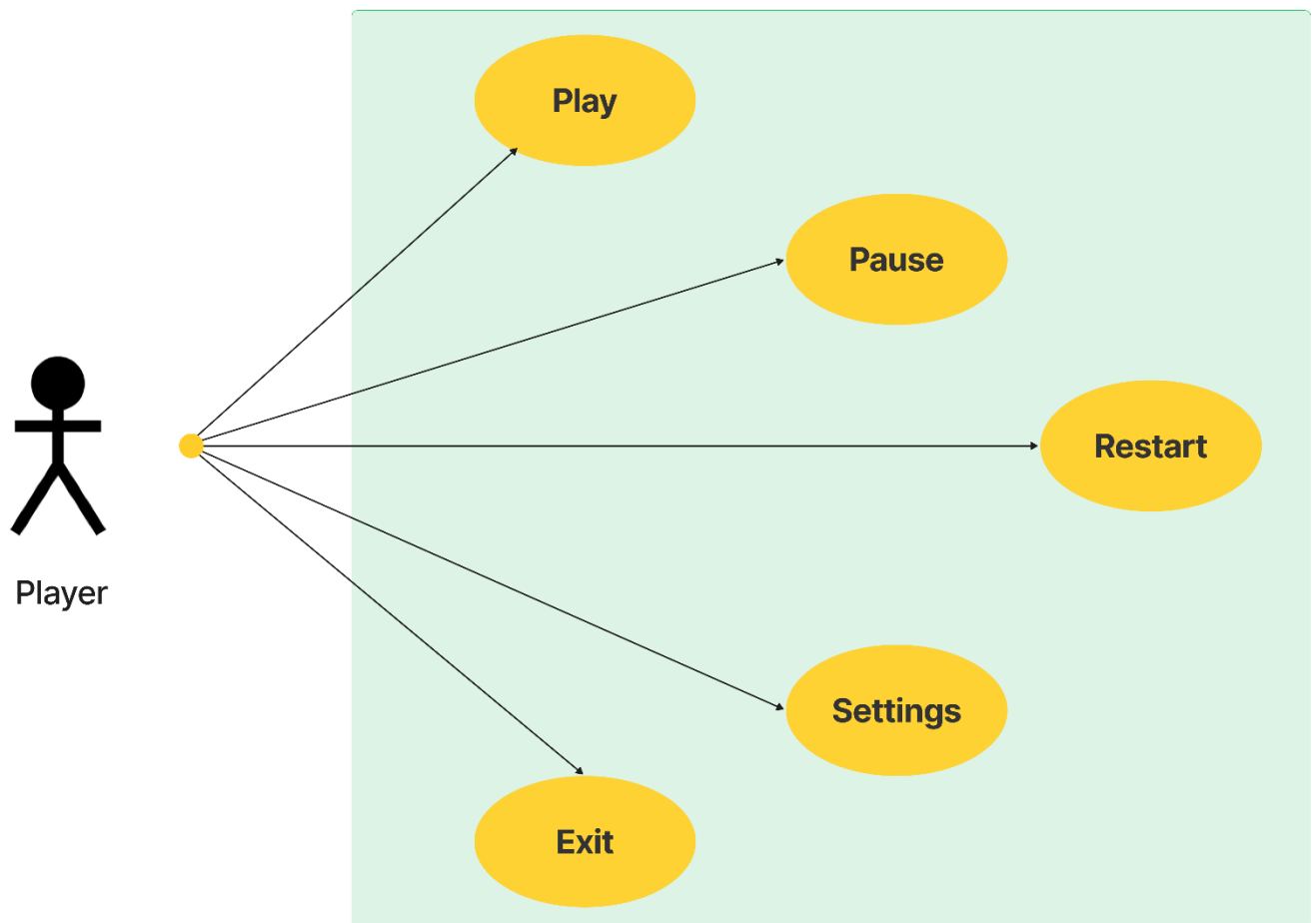
SYSTEM ANALYSIS

EVENT TABLE

<u>Event</u>	<u>Description</u>	<u>Parameters</u>
Start Game	Begins the game and sets the timer	Time Limit
Start Cooking	Begins the cooking process for a recipe	Recipe ID
Start Timer	Starts a timer for a cooking step	Timer ID, Duration
Add Ingredient	Adds an ingredient to the dish being cooked	Ingredient ID, Amount
Serve Dish	Finishes cooking and serves the dish	N/A
Fail Cooking	Ends cooking process prematurely due to overcooking, burning, etc	Reason (e.g. burned, overcooked)
Deliver Recipe	Delivers a completed recipe	Recipe ID
Time's Up	Ends the game when the time limit is reached	N/A
Record Results	Records the total number of recipes delivered by the player	Number of Recipes Delivered

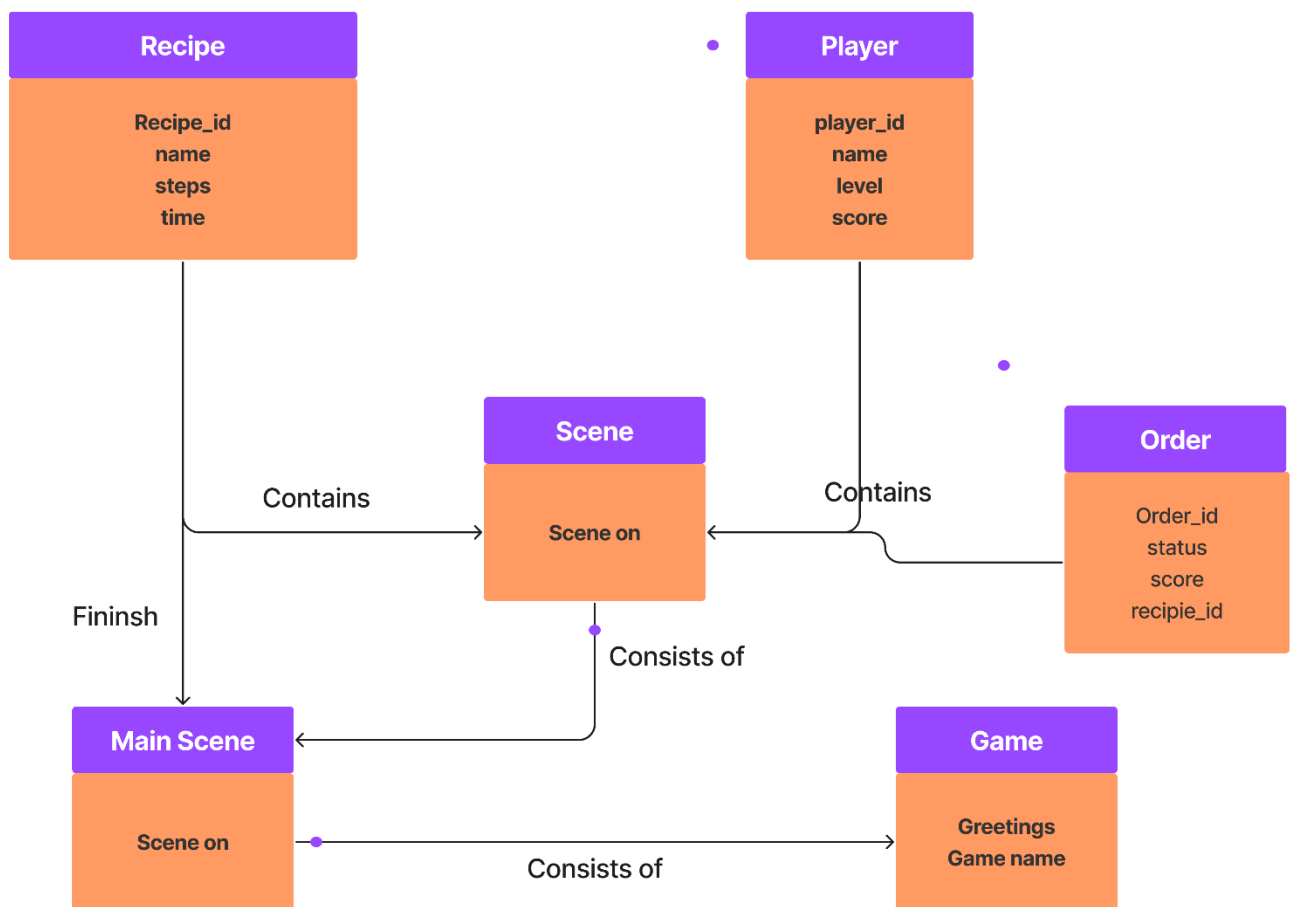
USE CASE DIAGRAM

Use case Diagram



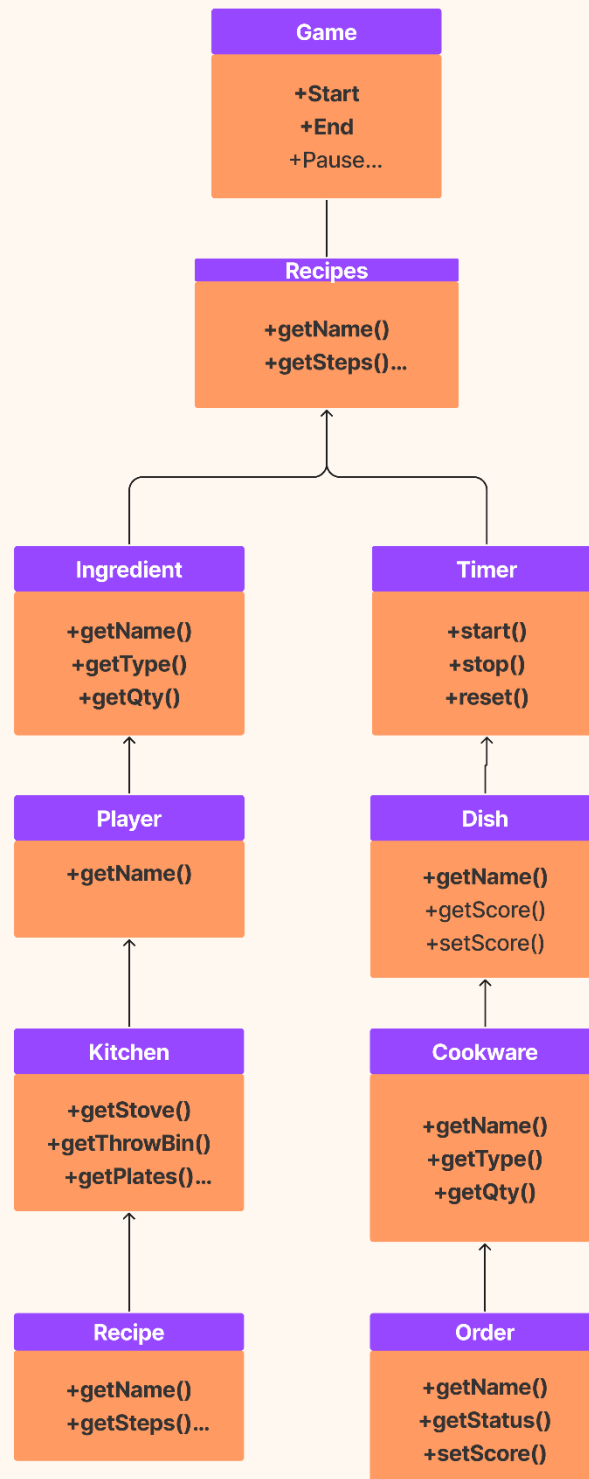
ENTITY RELATIONSHIP DIAGRAM

ER Diagram



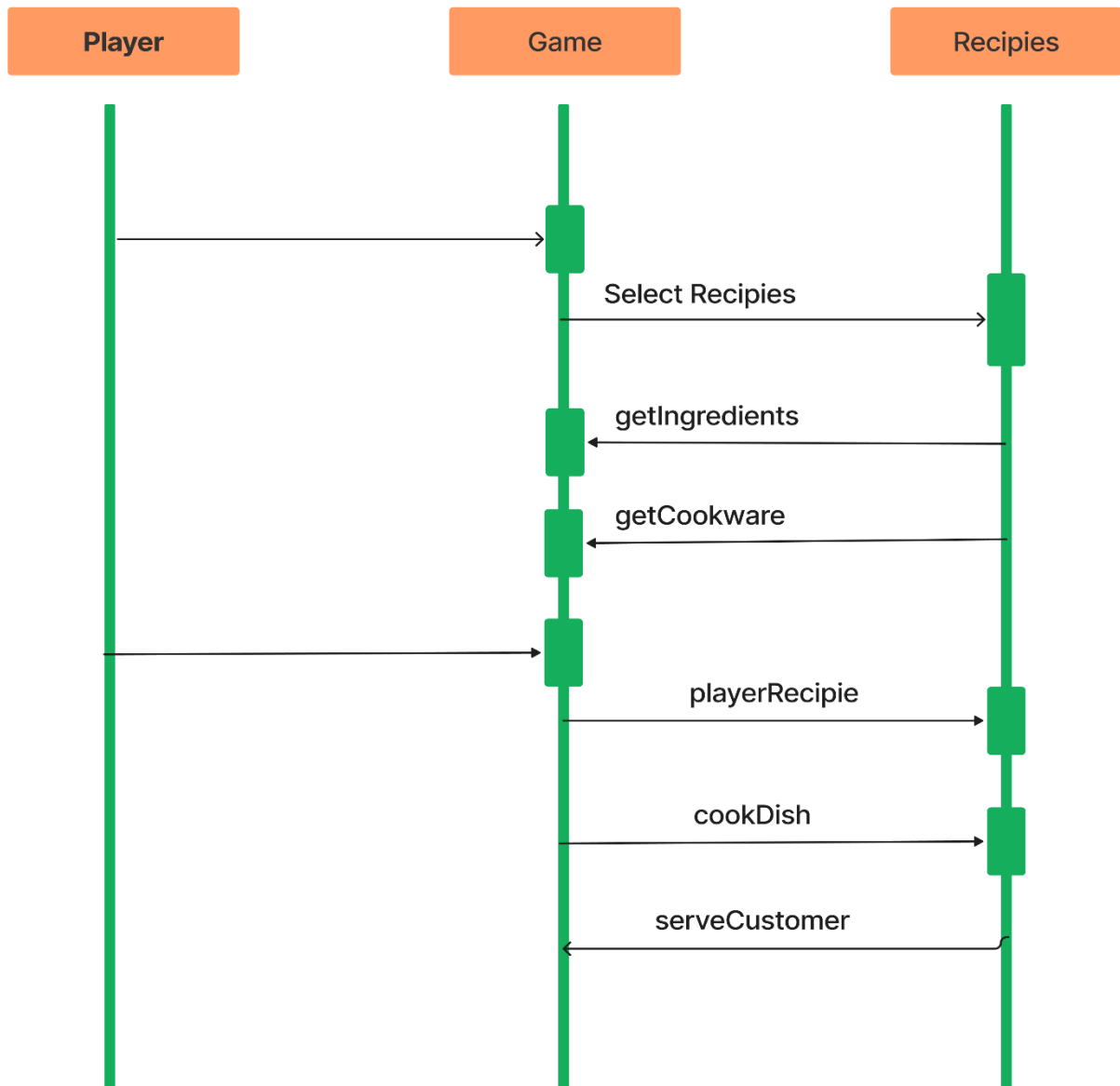
CLASS DIAGRAM

Class Diagram



SYSTEM DESIGN

SEQUENCE DIAGRAM



SYSTEM CODING

Player.cs

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

public class Player : MonoBehaviour, IKitchenObjectParent {
    public static Player Instance { get; private set; }
    public event EventHandler OnPickedSomething;
    public event EventHandler<OnSelectedCounterChangedEventArgs>
OnSelectedCounterChanged;
    public class OnSelectedCounterChangedEventArgs : EventArgs {
        public BaseCounter selectedCounter;
    }
    [SerializeField] private float moveSpeed = 7f;
    [SerializeField] private GameInput gameInput;
    [SerializeField] private LayerMask countersLayerMask;
    [SerializeField] private Transform kitchenObjectHoldPoint;

    private bool isWalking;
    private Vector3 lastInteractDir;
    private BaseCounter selectedCounter;
    private KitchenObject kitchenObject;

    private void Awake() {
        if (Instance != null) {
```

```

        Debug.LogError("There is more than one Player instance");
    }
    Instance = this;
}

private void Start() {
    gamelInput.OnInteractAction += GamelInput_OnInteractAction;
    gamelInput.OnInteractAlternateAction +=
GamelInput_OnInteractAlternateAction;
}

private void GamelInput_OnInteractAlternateAction(object sender,
EventArgs e) {
    if (!KitchenGameManager.Instance.IsGamePlaying()) return;
    if (selectedCounter != null) {
        selectedCounter.InteractAlternate(this);
    }
}

private void GamelInput_OnInteractAction(object sender, System.EventArgs
e) {
    if (!KitchenGameManager.Instance.IsGamePlaying()) return;
    if (selectedCounter != null) {
        selectedCounter.Interact(this);
    }
}

private void Update() {
    HandleMovement();
    HandleInteractions();
}

```

```

public bool IsWalking() {
    return isWalking;
}

private void HandleInteractions() {
    Vector2 inputVector = gameInput.GetMovementVectorNormalized();
    Vector3 moveDir = new Vector3(inputVector.x, 0f, inputVector.y);
    if (moveDir != Vector3.zero) {
        lastInteractDir = moveDir;
    }

    float interactDistance = 2f;

    if (Physics.Raycast(transform.position, lastInteractDir, out RaycastHit
raycastHit, interactDistance, countersLayerMask)) {
        if (raycastHit.transform.TryGetComponent(out BaseCounter
baseCounter)) {
            // Has ClearCounter
            if (baseCounter != selectedCounter) {
                SetSelectedCounter(baseCounter);
            }
        } else {
            SetSelectedCounter(null);
        }
    } else {
        SetSelectedCounter(null);
    }
}

private void HandleMovement() {
    Vector2 inputVector = gameInput.GetMovementVectorNormalized();

```

```

Vector3 moveDir = new Vector3(inputVector.x, 0f, inputVector.y);
float moveDistance = moveSpeed * Time.deltaTime;
float playerRadius = .7f;
float playerHeight = 2f;

bool canMove = !Physics.CapsuleCast(transform.position,
transform.position + Vector3.up * playerHeight, playerRadius, moveDir,
moveDistance);

if (!canMove) {
    // Cannot move towards moveDir

    // Attempt only X movement
    Vector3 moveDirX = new Vector3(moveDir.x, 0, 0).normalized;

    canMove = (moveDir.x < -.5f || moveDir.x > +.5f) &&
!Physics.CapsuleCast(transform.position, transform.position + Vector3.up *
playerHeight, playerRadius, moveDirX, moveDistance);

    if (canMove) {
        // Can move only on the X
        moveDir = moveDirX;
    } else {
        // Cannot move only on the X
        // Attempt only Z movement
        Vector3 moveDirZ = new Vector3(0, 0, moveDir.z).normalized;

        canMove = (moveDir.z < -.5f || moveDir.z > +.5f) &&
!Physics.CapsuleCast(transform.position, transform.position + Vector3.up *
playerHeight, playerRadius, moveDirZ, moveDistance);

        if (canMove) {
            // Can move only on the Z
            moveDir = moveDirZ;
        }
    }
}

```

```

        } else {
            // Cannot move in any direction
        }
    }
}

if (canMove) {
    transform.position += moveDir * moveDistance;
}

isWalking = moveDir != Vector3.zero;

float rotateSpeed = 10f;

transform.forward = Vector3.Slerp(transform.forward, moveDir,
Time.deltaTime * rotateSpeed);
}

private void SetSelectedCounter(BaseCounter selectedCounter) {
    this.selectedCounter = selectedCounter;

    OnSelectedCounterChanged?.Invoke(this, new
OnSelectedCounterChangedEventArgs {
        selectedCounter = selectedCounter
    });
}

public Transform GetKitchenObjectFollowTransform() {
    return kitchenObjectHoldPoint;
}

public void SetKitchenObject(KitchenObject kitchenObject) {
    this.kitchenObject = kitchenObject;

    if (kitchenObject != null) {

```

```

        OnPickedSomething?.Invoke(this, EventArgs.Empty);
    }
}

public KitchenObject GetKitchenObject() {
    return kitchenObject;
}

public void ClearKitchenObject() {
    kitchenObject = null;
}

public bool HasKitchenObject() {
    return kitchenObject != null;
}

}

```

DeliveryManager.cs

```

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

public class DeliveryManager : MonoBehaviour
{
    public event EventHandler OnRecipeSpawned;
    public event EventHandler OnRecipeCompleted;
    public event EventHandler OnRecipeSuccess;

```

```

public event EventHandler OnRecipeFailed;
public static DeliveryManager Instance { get; private set; }
[SerializeField] private RecipeListSO recipeListSO;
private List<RecipeSO> waitingRecipeSOList;
private float spawnRecipeTimer;
private float spawnRecipeTimerMax = 4f;
private int waitingRecipesMax = 4;
private int successfulRecipesAmount;
private void Awake()
{
    Instance = this;

    waitingRecipeSOList = new List<RecipeSO>();
}
private void Update()
{
    spawnRecipeTimer -= Time.deltaTime;
    if (spawnRecipeTimer <= 0f)
    {
        spawnRecipeTimer = spawnRecipeTimerMax;

        if (KitchenGameManager.Instance.IsGamePlaying() &&
            waitingRecipeSOList.Count < waitingRecipesMax)
        {
            RecipeSO waitingRecipeSO =
            recipeListSO.recipeSOList[UnityEngine.Random.Range(0,
            recipeListSO.recipeSOList.Count)];

```



```

        waitingRecipeSOList.Add(waitingRecipeSO);

        OnRecipeSpawned?.Invoke(this, EventArgs.Empty);
    }
}

public void DeliverRecipe(PlateKitchenObject plateKitchenObject)
{
    for (int i = 0; i < waitingRecipeSOList.Count; i++)
    {
        RecipeSO waitingRecipeSO = waitingRecipeSOList[i];

        if (waitingRecipeSO.kitchenObjectSOList.Count ==
            plateKitchenObject.GetKitchenObjectSOList().Count)
        {
            // Has the same number of ingredients

            bool plateContentsMatchesRecipe = true;

            foreach (KitchenObjectSO recipeKitchenObjectSO in
                waitingRecipeSO.kitchenObjectSOList)
            {
                // Cycling through all ingredients in the Recipe

                bool ingredientFound = false;

                foreach (KitchenObjectSO plateKitchenObjectSO in
                    plateKitchenObject.GetKitchenObjectSOList())
                {
                    // Cycling through all ingredients in the Plate

```

```

        if (plateKitchenObjectSO == recipeKitchenObjectSO)
        {
            // Ingredient matches!
            ingredientFound = true;
            break;
        }
    }
    if (!ingredientFound)
    {
        // This Recipe ingredient was not found on the Plate
        plateContentsMatchesRecipe = false;
    }
}
if (plateContentsMatchesRecipe)
{
    // Player delivered the correct recipe!
    successfulRecipesAmount++;
    waitingRecipeSOList.RemoveAt(i);
    OnRecipeCompleted?.Invoke(this, EventArgs.Empty);
    OnRecipeSuccess?.Invoke(this, EventArgs.Empty);
    return;
}
}

// No matches found!
// Player did not deliver a correct recipe

```

```

        OnRecipeFailed?.Invoke(this, EventArgs.Empty);
    }
    public List<RecipeSO> GetWaitingRecipeSOList()
    {
        return waitingRecipeSOList;
    }
    public int GetSuccessfulRecipesAmount()
    {
        return successfulRecipesAmount;
    }
}

```

KitchenGameManager.cs

```

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

public class KitchenGameManager : MonoBehaviour {
    public static KitchenGameManager Instance { get; private set; }
    public event EventHandler OnStateChanged;
    public event EventHandler OnGamePaused;
    public event EventHandler OnGameUnpaused;

    private enum State {

```

```

    WaitingToStart,
    CountdownToStart,
    GamePlaying,
    GameOver,
}

private State state;

private float countdownToStartTimer = 3f;

private float gamePlayingTimer;

[SerializeField]private float gamePlayingTimerMax = 60f;

private bool isGamePaused = false;

private void Awake() {
    Instance = this;
    state = State.WaitingToStart;
}

private void Start() {
    GamelInput.Instance.OnPauseAction += GamelInput_OnPauseAction;
    GamelInput.Instance.OnInteractAction += GamelInput_OnInteractAction;
}

private void GamelInput_OnInteractAction(object sender, EventArgs e) {
    if (state == State.WaitingToStart) {
        state = State.CountdownToStart;
        OnStateChanged?.Invoke(this, EventArgs.Empty);
    }
}

private void GamelInput_OnPauseAction(object sender, EventArgs e) {
    TogglePauseGame();
}

```

```

    }
    private void Update() {
        switch (state) {
            case State.WaitingToStart:
                break;
            case State.CountdownToStart:
                countdownToStartTimer -= Time.deltaTime;
                if (countdownToStartTimer < 0f) {
                    state = State.GamePlaying;
                    gamePlayingTimer = gamePlayingTimerMax;
                    OnStateChanged?.Invoke(this, EventArgs.Empty);
                }
                break;
            case State.GamePlaying:
                gamePlayingTimer -= Time.deltaTime;
                if (gamePlayingTimer < 0f) {
                    state = State.GameOver;
                    OnStateChanged?.Invoke(this, EventArgs.Empty);
                }
                break;
            case State.GameOver:
                break;
        }
    }

    public bool IsGamePlaying() {
        return state == State.GamePlaying;
    }

```

```

    }

    public bool IsCountdownToStartActive() {
        return state == State.CountdownToStart;
    }

    public float GetCountdownToStartTimer() {
        return countdownToStartTimer;
    }

    public bool IsGameOver() {
        return state == State.GameOver;
    }

    public float GetGamePlayingTimerNormalized() {
        return 1 - (gamePlayingTimer / gamePlayingTimerMax);
    }

    public void TogglePauseGame() {
        isGamePaused = !isGamePaused;
        if (isGamePaused) {
            Time.timeScale = 0f;
            OnGamePaused?.Invoke(this, EventArgs.Empty);
        } else {
            Time.timeScale = 1f;
            OnGameUnpaused?.Invoke(this, EventArgs.Empty);
        }
    }
}

```

GameInput.cs

```
using System;

using UnityEngine;
using UnityEngine.InputSystem;

public class GameInput : MonoBehaviour {

    private const string PLAYER_PREFS_BINDINGS = "InputBindings";

    public static GameInput Instance { get; private set; }

    public event EventHandler OnInteractAction;
    public event EventHandler OnInteractAlternateAction;
    public event EventHandler OnPauseAction;
    public event EventHandler OnBindingRebind;

    public enum Binding {
        Move_Up,
        Move_Down,
        Move_Left,
        Move_Right,
        Interact,
        InteractAlternate,
        Pause,
        Gamepad_Interact,
        Gamepad_InteractAlternate,
```

```

        Gamepad_Pause
    }

    private PlayerInputActions playerInputActions;

    private void Awake() {
        Instance = this;
        playerInputActions = new PlayerInputActions();
        if (PlayerPrefs.HasKey(PPLAYER_PREFS_BINDINGS)) {
            playerInputActions.LoadBindingOverridesFromJson(PlayerPrefs.GetString(PPLAYER_PREFS_BINDINGS));
        }
        playerInputActions.Player.Enable();
        playerInputActions.Player.Interact.performed += Interact_performed;
        playerInputActions.Player.InteractAlternate.performed +=
InteractAlternate_performed;
        playerInputActions.Player.Pause.performed += Pause_performed;
    }

    private void OnDestroy() {
        playerInputActions.Player.Interact.performed -= Interact_performed;
        playerInputActions.Player.InteractAlternate.performed -=
InteractAlternate_performed;
        playerInputActions.Player.Pause.performed -= Pause_performed;
        playerInputActions.Dispose();
    }

    private void
Pause_performed(UnityEngine.InputSystem.InputAction.CallbackContext obj) {
        OnPauseAction?.Invoke(this, EventArgs.Empty);
    }

```



```

    private void
InteractAlternate_performed(UnityEngine.InputSystem.InputAction.CallbackCo
ntext obj) {

    OnInteractAlternateAction?.Invoke(this, EventArgs.Empty);

}

    private void
Interact_performed(UnityEngine.InputSystem.InputAction.CallbackContext
obj) {

    OnInteractAction?.Invoke(this, EventArgs.Empty);

}

    public Vector2 GetMovementVectorNormalized() {

        Vector2 inputVector =
playerInputActions.Player.Move.ReadValue<Vector2>();

        inputVector = inputVector.normalized;

        return inputVector;

    }

    public string GetBindingText(Binding binding) {

        switch (binding) {

            default:

            case Binding.Move_Up:

                return playerInputActions.Player.Move.bindings[1].ToDisplayString();

            case Binding.Move_Down:

                return playerInputActions.Player.Move.bindings[2].ToDisplayString();

            case Binding.Move_Left:

                return playerInputActions.Player.Move.bindings[3].ToDisplayString();

            case Binding.Move_Right:

                return playerInputActions.Player.Move.bindings[4].ToDisplayString();

        }

    }

```

```

        case Binding.Interact:
            return
playerInputActions.Player.Interact.bindings[0].ToString();

        case Binding.InteractAlternate:
            return
playerInputActions.Player.InteractAlternate.bindings[0].ToString();

        case Binding.Pause:
            return playerInputActions.Player.Pause.bindings[0].ToString();

        case Binding.Gamepad_Interact:
            return
playerInputActions.Player.Interact.bindings[1].ToString();

        case Binding.Gamepad_InteractAlternate:
            return
playerInputActions.Player.InteractAlternate.bindings[1].ToString();

        case Binding.Gamepad_Pause:
            return playerInputActions.Player.Pause.bindings[1].ToString();
    }
}

public void RebindBinding(Binding binding, Action onActionRebound) {
    playerInputActions.Player.Disable();

    InputAction inputAction;
    int bindingIndex;

    switch (binding) {
        default:
        case Binding.Move_Up:
    }
}

```

```
        inputAction = playerInputActions.Player.Move;
        bindingIndex = 1;
        break;
    case Binding.Move_Down:
        inputAction = playerInputActions.Player.Move;
        bindingIndex = 2;
        break;
    case Binding.Move_Left:
        inputAction = playerInputActions.Player.Move;
        bindingIndex = 3;
        break;
    case Binding.Move_Right:
        inputAction = playerInputActions.Player.Move;
        bindingIndex = 4;
        break;
    case Binding.Interact:
        inputAction = playerInputActions.Player.Interact;
        bindingIndex = 0;
        break;
    case Binding.InteractAlternate:
        inputAction = playerInputActions.Player.InteractAlternate;
        bindingIndex = 0;
        break;
    case Binding.Pause:
        inputAction = playerInputActions.Player.Pause;
        bindingIndex = 0;
```

```

        break;
    case Binding.Gamepad_Interact:
        inputAction = playerInputActions.Player.Interact;
        bindingIndex = 1;
        break;
    case Binding.Gamepad_InteractAlternate:
        inputAction = playerInputActions.Player.InteractAlternate;
        bindingIndex = 1;
        break;
    case Binding.Gamepad_Pause:
        inputAction = playerInputActions.Player.Pause;
        bindingIndex = 1;
        break;
}

inputAction.PerformInteractiveRebinding(bindingIndex)
    .OnComplete(callback => {
        callback.Dispose();
        playerInputActions.Player.Enable();
        onActionRebound();

        PlayerPrefs.SetString(PPLAYER_PREFS_BINDINGS,
playerInputActions.SaveBindingOverridesAsJson());

        PlayerPrefs.Save();

        OnBindingRebind?.Invoke(this, EventArgs.Empty);
    })
    .Start();
}
}

```

TEST CASES

Test Case	Expected Result	Actual Result	Status
Player clicks on "Start" button	Game starts and timer starts counting down from given time limit	Game starts and timer starts counting down from given time limit	Pass
Verify each recipe is unique	Each recipe should be different from previous recipes	Each recipe is different from previous recipes	Pass
Verify recipe is randomly generated	Recipe should not be predictable or repeated	Recipe is not predictable or repeated	Pass
Verify recipe is complete only if all required ingredients used in correct order and quantity	Player should only receive points for complete and correct recipes	Player does not receive points for incomplete or incorrect recipes	Pass
Verify game ends when timer reaches zero	Game should end and total score should be calculated based on number of completed recipes	Game ends and total score is calculated correctly	Pass
Verify total score is displayed correctly at end of game	Player should see their total score at the end of the game	Player can see their total score at the end of the game	Pass
Verify player can pause and resume the game	Player should be able to pause and resume the game without any issues	Player can pause and resume the game without any issues	Pass

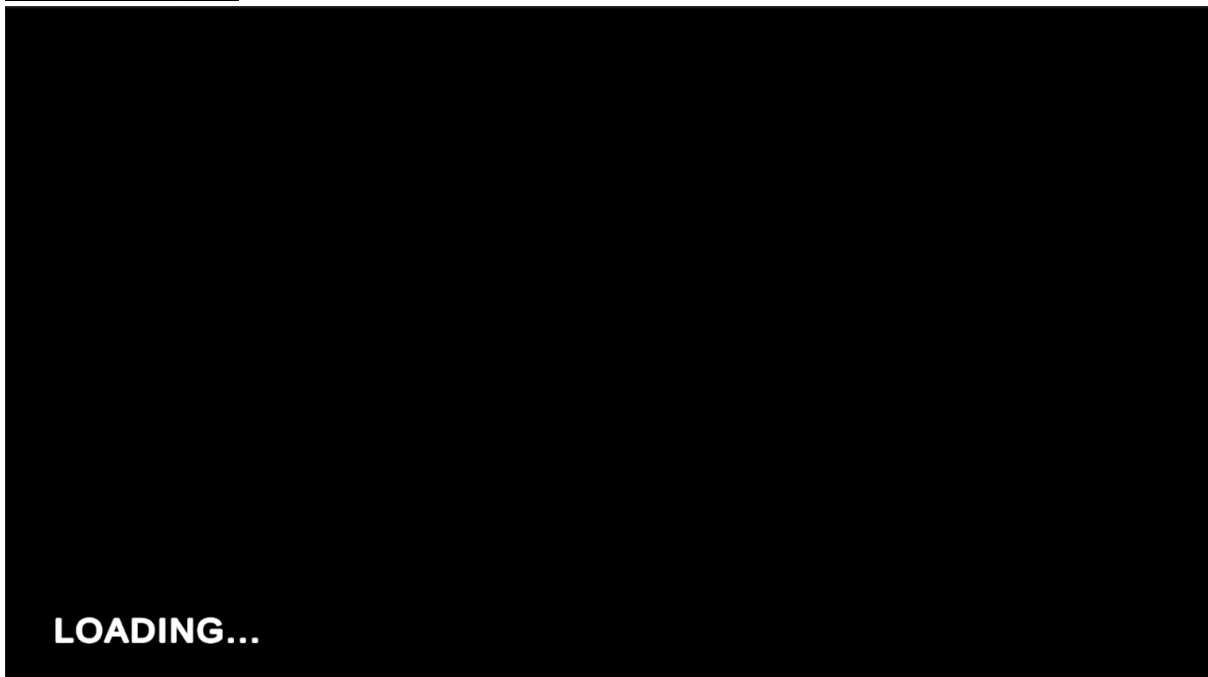
Verify player can restart the game from beginning	Player should be able to restart the game from beginning without any issues	Player can restart the game from beginning without any issues	Pass
Verify game adapts to player's skill level	Game should offer more challenging recipes as player progresses	Game offers more challenging recipes as player progresses	Pass

FORMAL LAYOUT

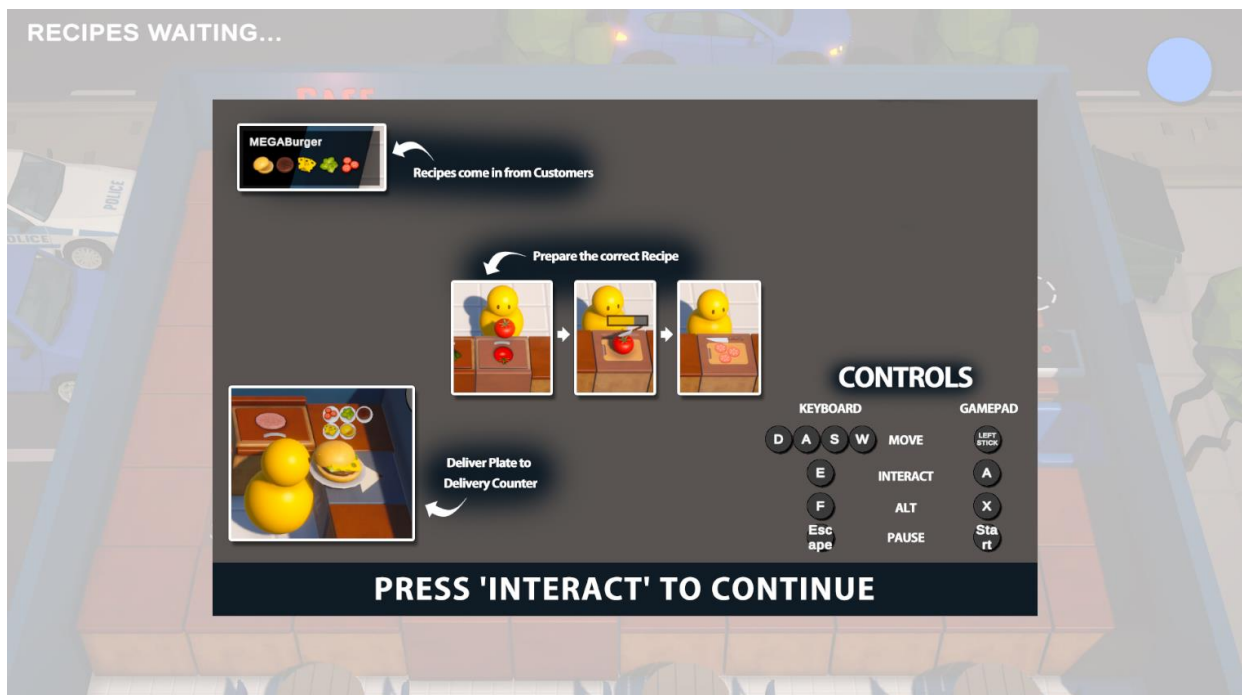
MainMenu



Loading Screen



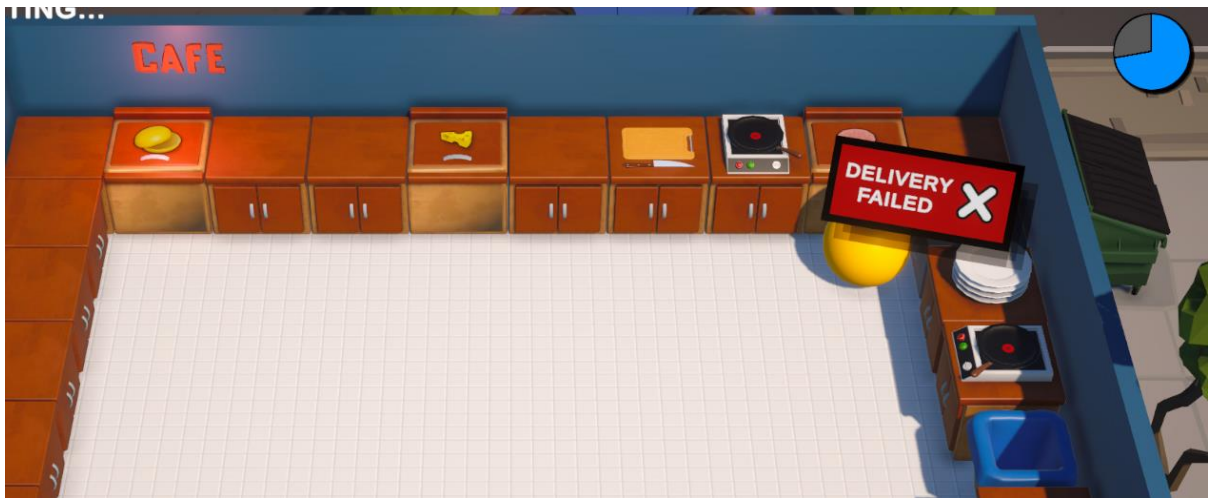
Tutorial Screen



Game Screen



Delivery failed



Delivery Successful



Credits UI



Future Enhancement

1. **Multiplayer Mode:** Add a multiplayer mode where players can compete against each other in real-time. This can add a new level of excitement and make the game more engaging.
2. **Customizable Recipes:** Allow players to create their own recipes or modify existing ones. This can add more variety to the game and give players a sense of ownership.
3. **Virtual Currency:** Add a virtual currency that players can earn by completing recipes and use to buy new ingredients, unlock new recipes or even purchase power-ups that can help them during the game.
4. **Power-ups:** Add power-ups that players can use during the game to gain an advantage. For example, a power-up that slows down the timer, or one that provides an extra hint for a recipe.
5. **Achievements and Leaderboards:** Add achievements that players can earn by completing specific challenges or reaching milestones. Also, add leaderboards that display the top players in the game, which can encourage healthy competition.
6. **In-game Store:** Add an in-game store where players can purchase new recipes, ingredients, or other game items using virtual currency earned during gameplay.
7. **Social Sharing:** Allow players to share their progress, recipes, and scores on social media platforms, such as Facebook or Twitter. This can increase the game's visibility and encourage more players to join in.



PLAGIARISM SCAN REPORT

Date March 21, 2023

Exclude URL: NO



Unique Content **98**

Plagiarized Content **2**

Word Count 993

Records Found 0

CONTENT CHECKED FOR PLAGIARISM:

A
PROJECT REPORT
ON
Sizzle N Serve
PC GAME DESIGNED AND DEVELOPED
BY
MR.ALIRAZA SOHEL KHAN
TYBSC CS 30
2022-2023
UNDER
THE GUIDANCE OF
PROF. ANUSHKA PADHYE
SUBMITTED IN PARTIAL FULFILLMENT OF
ACADEMIC PROJECT
[BACHELOR OF SCIENCE COMPUTER SCIENCE]
[UNIVERSITY OF MUMBAI]
ACKNOWLEDGEMENT
ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are

your capabilities and how well you can present yourself or other things. It tells us how much we rely on the efforts and goodwill of others. It gives me immense pleasure to present this report to fulfill my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who helped me complete this project successfully.

I take this opportunity to express my profound gratitude to management of Royal College Of Arts, Commerce & Science for giving me this opportunity to accomplish this project work.

A special vote of thanks to Prof. Anushka Padhye, our professor & project guide, for their most sincere, useful and encouraging contribution throughout the project span.

Finally, I would like to thank the entire Computer Science department who directly or indirectly helped me in the completion of this project & my family without their support, motivation & encouragement this would not have been possible.

ALIRAZA SOHEL KHAN

DECLARATION

DECLARATION

I "Mr. ALIRAZA SOHEL KHAN " hereby declare that I myself have completed the project under the guidance of Prof. Anushka Padhye I on my own have designed the system and have done all the programming required.

It may require some modifications in the future as per the user's requirements. From practical implementation point of view flexibility in the changes have incorporated in the package.

I am sure that I can do any kind of modification suggested while practical implementation by modifying file design or the program code if necessary.

ALIRAZA SOHEL KHAN

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE

Class: TYBScRoll. No. 30

Seat No: _____

Certificate

Certified that Mr. ALIRAZA SOHEL KHAN of T.Y.BSc -CS Semester-VI has successfully completed the project as prescribed by the University of Mumbai on SizzleNServe PC Game as partial fulfilment of requirement for completing Bachelor's Degree in Computer Science during the academic year 2022-2023.

Problem-solving: Cooking games require players to think critically and solve problems by choosing the correct ingredients and cookware, managing their time effectively, and adapting to changing game conditions.

Accessibility: Cooking games are widely available on various platforms, including mobile devices and gaming consoles, making them easily accessible to a large audience.

Community building: Cooking games can help build a strong and loyal community of players who share a common interest in cooking and food preparation.

Revenue generation: Cooking games can generate revenue through in-game advertising and microtransactions, allowing game developers to continue improving and expanding the game over time.

MATCHED SOURCES:

Blood Bank Managementee | PDF - Scribd

<https://www.scribd.com/document/441404120/BLOOD-BANK-MANAGEM....>

Report Generated on **March 21, 2023** by check-plagiarism.com

Reference and Bibliography

- Youtube.com
- Unity3d.com
- Figma.com
- Stackoverflow
- assetstore.unity.com