Apocalypse Cooker

Project Overview

A cooking game with an apocalypse theme, where the chef opens the restaurant to send food to the survivors of a zombie horde in the quarantine zone around the city. The chef, who is a player, must avoid zombies who burst into the restaurants as possible while he has to cook food for the survivors.

Project Review

Overcooked is a cooking game against time and each map there will be obstacles in the levels (floors disappearing, players misstep and fall from buildings), but we will add walking zombies that will spawn random places and chase players around.

Programming Development

Game Concept

The purpose of the game is to serve an ordered menu as many dishes as you can to reach the highest score and the game will stop when the timer runs out or you are attacked by zombies until your health runs out.

- *Cooking Mechanics*
    - *Players take the role of chef who must prepare dishes based on customer orders.*
    - *Players will have the main 4 minutes to manage for cooking and serving*
    - *Each dish requires a sequence of actions: chopping, grilling, boiling, frying, etc.*
    - *Customers have patience meters, taking too long makes them leave.*

    - *Players must manage time efficiently to keep customers happy and earn more points.*
    - *The food may run out by rot. Player has to go order more ingredients*
- *Zombie Threat*
    - *A while after the player cooks dishes, Zombies will burst in and roam around the restaurant, blocking movement and disrupting cooking.*
    - *If a zombie touches the chef, they lose health.*
    - *Zombies will keep following the player and the player can't attack zombies back.*
- *Customer System*
    - *Customers arrive in waves, requesting meals.*
    - *Serving customers correctly earns money and combo points.*

Object-Oriented Programming Implementation

Class Descriptions for Cooking Game:

GameApp (Main Game Controller)

- Manages the overall gameplay, including game initialization, event handling and game loop execution
- Listens for player input and updates game objects accordingly.
- Handles interactions between the chef, kitchen tools, ingredients and enemies

Chef (Player Character)

- Represents the player in the game, allowing movement and interaction with kitchen elements
- Can pick up, carry and place ingredients

- Uses kitchen tools (e.g. pan, pot, cutting board) to prepare food.
- Must avoid the zombie while cooking

Ingredient (Food Items)

- Represents food items that the chef can pick up, cook, or use in recipes.
- The food can be raw, cooked, or chopped, depending on interactions with kitchen tools.
- Can be stored in the fridge or placed on a plate for serving.

Menu (Customer Orders)

- Displays incoming customer orders dynamically on the side of the screen.
- Orders appear randomly over time, reflecting real-time customer requests.
- Provides a complete menu that players should follow, not random ingredients.
- Updates in real time as new orders arrive and completed dishes are served.

Zombie (Enemy)

- The main antagonist of the game, constantly chasing the chef and will attack the chef to reduce the chef's health.
- Forces the player to balance cooking and escaping to avoid being caught.

Equipment (Kitchen Equipment)

- A parent class for various kitchen tools, including Pan and Cutting Board.
- Defines common behaviors for interacting with ingredients.

Pan (Frying Tool)

- Used for frying ingredients such as meat, eggs, and vegetables.

- Can hold multiple ingredients and cook them over time.
- When cooking is complete, the chef can transfer the food to a plate.

CuttingBoard (Chopping Tool)

- Used for chopping ingredients before cooking.
- Transforms raw ingredients into their chopped versions.
- Essential for preparing certain dishes that require diced ingredients.

TrashBin (Disposal Tool)

- Used for disposing of waste materials such as ingredients that players don’t want or the wrong finished order.

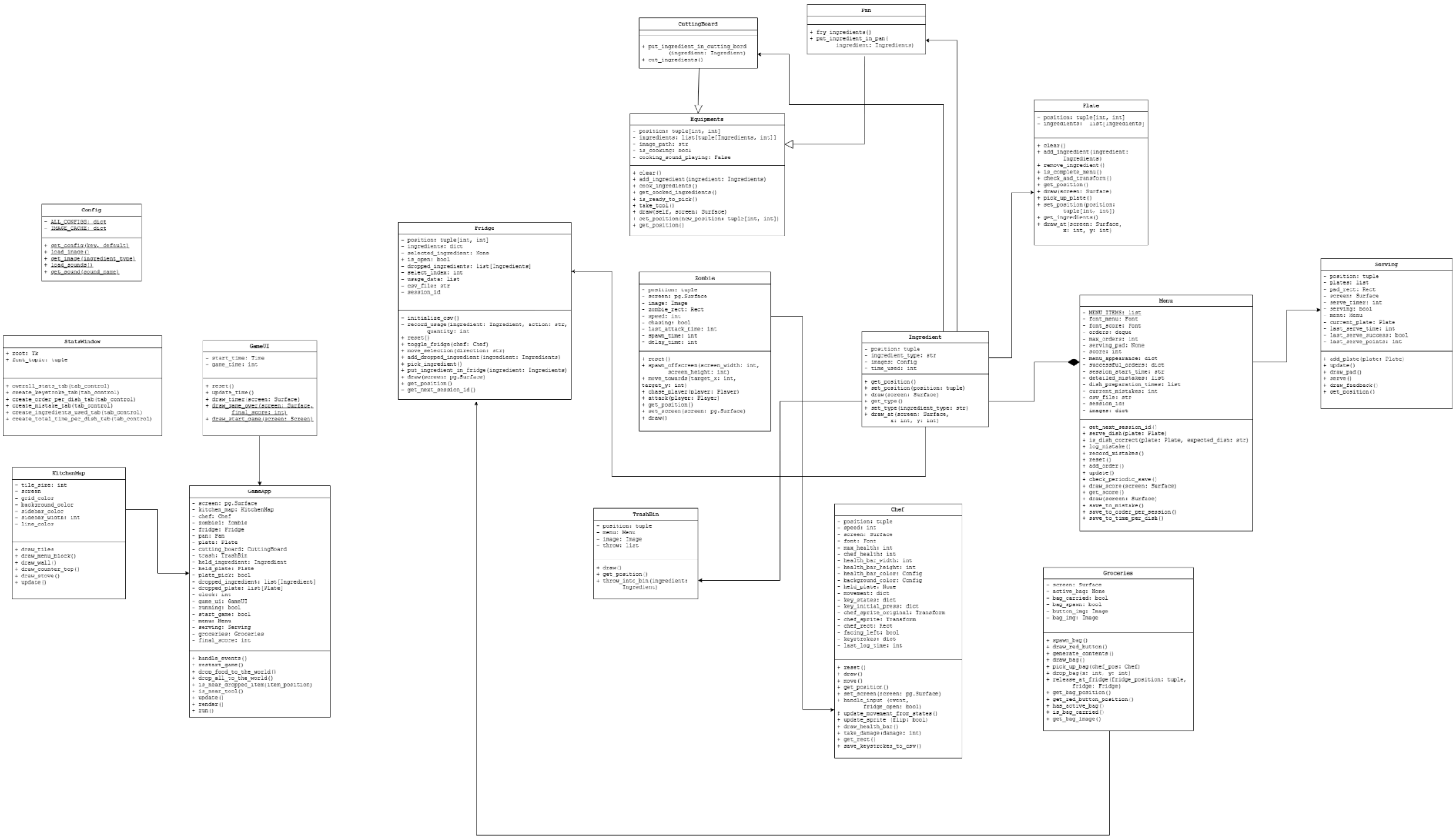
Groceries (Restock food)

- Used for restocking food if it runs out of stock from the fridge.
- Players can request for ingredients by pressing the red button.
- The groceries will be in a paper bag for the player to put it in the fridge.

Serving (Serving station)

- Used for sending the finished order to the customer
- Acts as the final checkpoint where quality and completeness of orders are verified
- Coordinates with waitstaff or delivery personnel to ensure timely and accurate delivery to customers

The Game UML Diagram



Algorithms Involved

Order Generation Algorithm (Queue System)

- *Randomly generates a customer's menu with different ingredients and complexity such as chicken fried, sandwich or just fried egg.*

Ingredients Matching Algorithm

- *Checks if the player has selected the correct ingredients for a recipe*

Timer Algorithm

- *Tracks how long an ingredient or dish is being cooked and determines if it is perfectly cooked, uncooked, or burnt*

Statistical Data (Prop Stats)

Data Features

	Why is it good to have this data? what can it be used for	**How to obtain 50 values of this feature data**	**Which variable and class will be collect this from**	**How will I display this feature data?**
Time taken per dish	Helps analyze player efficiency and difficulty of recipes.	Log time taken for each dish is over 50 instances.	class **Menu** will contain a score total method and it will track the time since each menu appears.	The display will be boxplot, it will allow us to compare the preparation times of different dishes
Number of ingredient used	Useful to check if players preferred which menu they cooked the most.	Record the number of ingredients required per dish over 50 orders.	class **Fridge** contain an ingredients so it will track when the player choose to pick ingredients out	A pie chart can be useful because it will show the proportion of ingredient-heavy vs. simple dishes.
Mistakes per order	Measures how accurately players complete orders. Helps identify if the game is too easy/hard.	record the wrong ingredient added, Trashing ingredients, Serving an incomplete dish.	Increment a counter every time a mistake occurs in class **Menu** since it tracks orders	The display will be held in a bar graph to see which mistakes player occur the most
Keystrokes Per dish	Can help identify if players are struggling or using extra steps.	Count keystrokes needed for 50 different dish preparations.	class **Chef** will track the movement of the player.	The display will be a line chart, it will show the frequency distribution of keystrokes per dish.
Number of orders complete per game session	Measures if players complete few or too many orders to see if the games might be too hard or too easy	count when the orders are successfully ordered per game session for 50 times	Increment when an order is successfully served in class **Menu**	The display will be held in a scatter plot since we want to see which orders players occur the most.

Table Presentation

Feature	Category	Graph Types
Total time taken per dish	Distribution	Histogram
Number of ingredient used	Proportion	Pie graph
Mistakes per order	Time-series	Line graph
Keystrokes per dish	Distribution	Histogram
Orders completed per game session	Relation	Scatter Plot (vs Time)

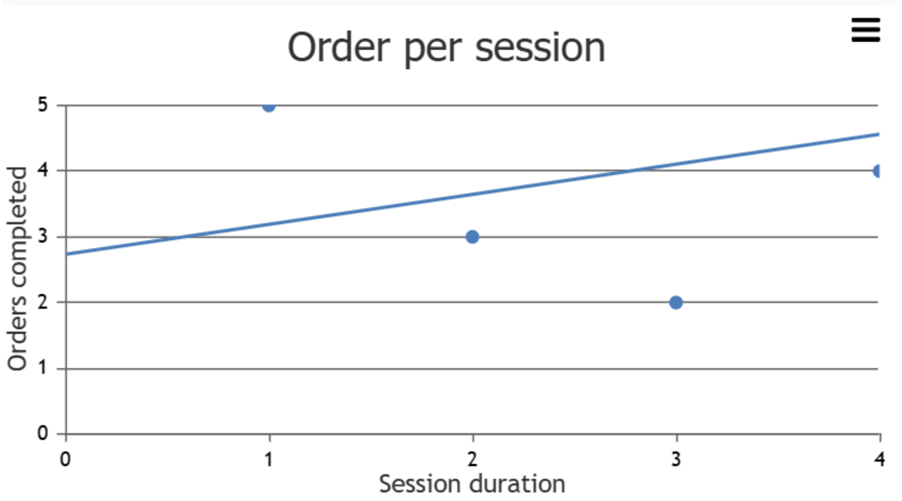
Example of Total time taken per dish table: Statistical Summary

Dish	Total Time (sum)	Mean Time	Median Time	Std Dev
Lamb fried	2456s	122.8s	118s	15.2
Chicken fried	1678s	94.6s	92s	12.7
Sandwich	324s	83.6s	80s	10.4
Egg fried	4532s	67.8s	65s	8.3

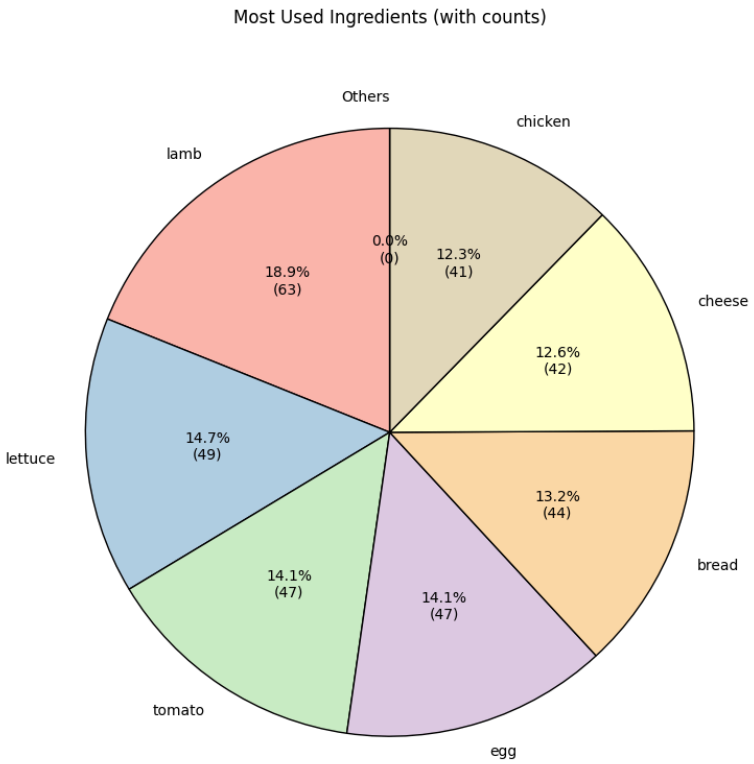
Graph Representation

Feature Name	Graph Objective	Graph type	X-axis	Y-axis
Total time per dish	Compare cooking efficiency across dishes	Boxplot	Dish Types (sandwich, chicken fried, etc.)	Average Time
Ingredients used	Show the proportional distribution of ingredients across all dishes.	Pie Chart	Ingredient Names	Percentage
Mistakes per order	Measures how accurately players complete orders.	Bar graph	The type of accident the player made.	Mistake count
Keystrokes per dish	Analyze input efficiency distribution	Line Chart	Keystroke bins	Frequency
Orders per session	Measure engagement vs. session duration	Scatter plot	Session duration (min)	Orders completed

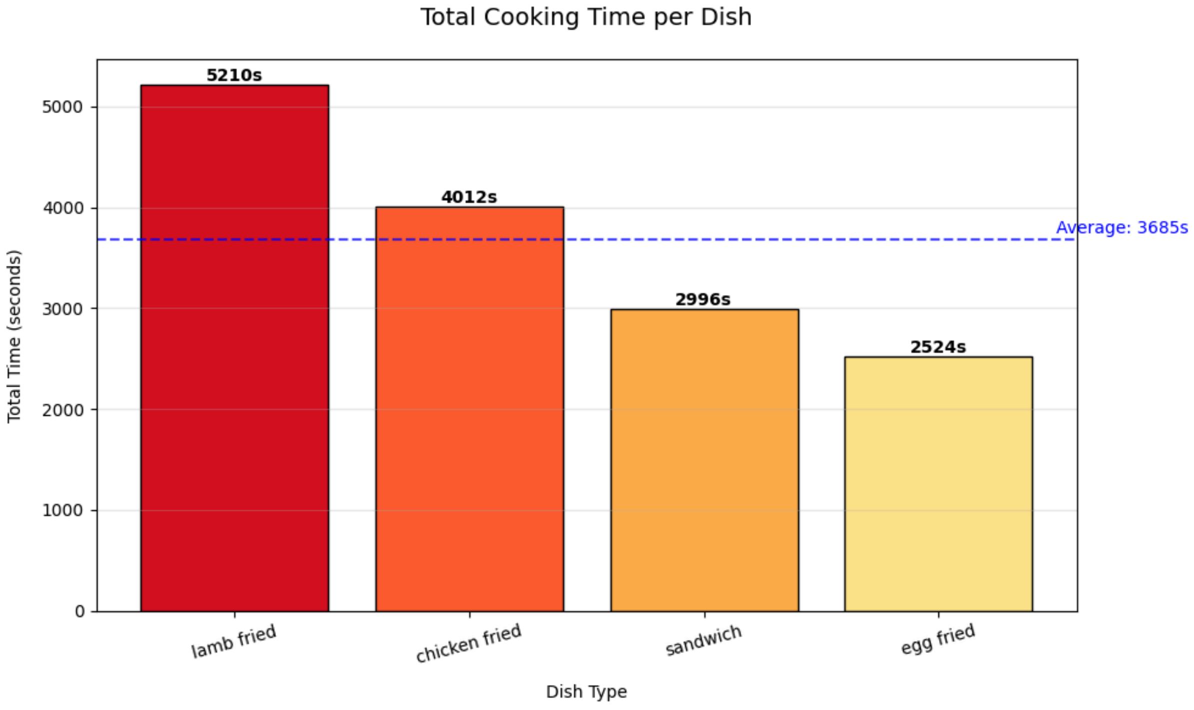
Example of scatter plot Orders per session



Example of pie chart Ingredients used



Example of bar chart Total Cooking Time per Dish



Data Recording Method

- *We use the CSV file to store total score, time spend and other stuffs*

Overview of Data Analysis

The collected data will be analyzed to evaluate players behavior, game difficulty and engagement. The goal is to gain insights into how players interact with the game mechanics, how well the game balances challenges and how to improve the overall experience

Statistical Measures Used

Descriptive Statistics

- Mean, Median, Mode used to analyze average player score, time spent in each level, and zombie encounters per session.
- Standard Deviation measures variance in player performance, e.g., if players struggle with specific recipes.

Correlation Analysis

- **Time spent cooking vs. success rate** → Do players who take longer perform better?
- **Zombie encounters vs. failure rate** → Are zombies making the game too difficult?
- **Customer patience vs. dish complexity** → Does a harder recipe cause more customers to leave?

Data Visualization & Presentation

- **Bar Charts** → Display **most popular recipes**, **zombie encounter rates**, and **common causes of failure**.
- **Line Graphs** → Show **player progress over time**, **average time spent per level**, and **completion rates**.

- **Pie Charts** → Represent **percentage of orders completed vs. failed**, **distribution of zombie types**, and **customer patience levels**.

Project Timeline

Week	**Task**
26 March-2 April	Complete main game
3 April-9 April	Complete Statistic part (can collect data)
10 April-16 April	List 50% - Complete details of game (fixed bug, decorate UI) - Can be able to play games in flow and can collect data
17 April-23 April	List 75% - Complete UI part and fixed bug if necessary
24 April-11 May	List 100% - Can play game in flow with least bug as possible - beautiful UI completed

Document version

Version: *5.0*

Date: *11 May 2025*