A Minor Project (AI3270) **PROJECT REPORT** on

# Marytini

Submitted to Manipal University Jaipur

Towards the partial fulfillment for the Award of the Degree of

**B. Tech Computer Science and Engineering (Artificial Intelligence and Machine Learning)**

2025

By

Saransh Sondhi

229310333

Arnav Agrawal

229310426

Under the guidance of

**Dr. Shail Saharan**

**Department of Artificial Intelligence and Machine Learning**

**Manipal University Jaipur**

**Jaipur, Rajasthan**

Date

# CERTIFICATE

This is to certify that the project entitled “Marytini” is a Bonafide work carried out as part of the course AI3270, under my guidance from Jan 2025 to May 2025 by **Saransh Sondhi (229310333)**, student of B. Tech (hons.) Computer Science and Engineering (AIML), 6th Semester at the Department of Artificial Intelligence and Machine Learning, Manipal University Jaipur, during the academic semester 6th in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AIML), at MUJ, Jaipur.

**Dr. Shail Saharan**
*Project Guide, Dept. of AIML*
*Manipal University Jaipur*

**Dr. Deepak Panwar**
*HOD, Dept. of AIML*
*Manipal University Jaipur*

Date

# CERTIFICATE

This is to certify that the project entitled “Marytini” is a Bonafide work carried out as part of the course AI3270, under my guidance from Jan 2025 to May 2025 by **Arnav Agrawal (229310426)**, student of B. Tech (hons.) Computer Science and Engineering (AIML), 6th Semester at the Department of Artificial Intelligence and Machine Learning, Manipal University Jaipur, during the academic semester 6th in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering (AIML), at MUJ, Jaipur.

**Dr. Shail Saharan**
*Project Guide, Dept. of AIML*
*Manipal University Jaipur*

**Dr. Deepak Panwar**
*HOD, Dept. of AIML*
*Manipal University Jaipur*

## ACKNOWLEDGMENTS

We would like to express our sincere gratitude to **Dr. Shail Saharan**, our Minor Guide and Mentor, for her invaluable guidance, encouragement, and support throughout the course of this project. Her insightful suggestions and constant motivation have been instrumental in the successful completion of this work.

We are also deeply thankful to **Dr. Deepak Panwar**, Head of Department of AI & ML, for providing a stimulating academic environment and for his continuous support and encouragement.

Their expertise and guidance have greatly contributed to our learning and the overall quality of this project.

## ABSTRACT

In today's world of interactive entertainment, deckbuilding roguelikes have experienced a resurgence in popularity with their blend of strategic complexity and replayability. The combination of bartending simulation together with deckbuilding roguelikes creates the opportunity for a unique gameplay experience. The player must interpret vague or limited customer requests and then create drinks in order to satisfy them. "Marytini" is intended to be a fun bartending game where players maximize their profits when preparing drinks based on customer feedback.

The approach to this project will be to simulate the bartending process through combination of class-based parameterized and cards representing ingredients chosen by the player. Each customer will present a request for drink characterized by parameters such as sweetness, saltiness, bitterness, and sourness. Each of those will be peg, along with qualitative states corresponding to the levels of Not, Little, Medium, Very, and Extremely. The player will receive a deck of ingredient cards and then select a drink base of either vodka, rum, or soda. The player will then create a drink by combining drink ingredients from cards in hand. The mixing process will occur in two tiers: at certain combinations of ingredients, new ingredients will be created.

To increase player engagement in the experience, the game features a dialogue system that determines customer requests based on the underlying drink parameters providing an element of narrative and challenge. In addition, each customer may also have hidden drink preferences, some player hints are provided. When successfully uncovering and constructing these secret drinks, players will also earn bonus revenue; which will lead to exploration and experimentation. The score system evaluates the organization of the drinks made, compared to what the customer requested. The orderly further with how accurately the two are aligned, the more monetary returns received.

In "Marytini" the ultimate aim is to earn as much for the players by the end of the in-game day. The experience illustrates how when game mechanics are considered, such as two-tier mixing with ingredients and hidden inquiries, a very engaging and replayable experience can occur. A game that balances strategic decision-making with creative problem-solving and allows for a new spin on both the deckbuilding and simulation game genre.

# LIST OF FIGURES

# Contents

# Chapter 1: Introduction

## 1.1 Background and Motivation

The gaming industry has seen a surge in innovative mechanics that blend strategy, chance, and creativity. Deckbuilding roguelike games, in particular, have captivated players by offering replayability and strategic depth. Inspired by the challenge of interpreting ambiguous customer requests and the creativity of mixology, "Marytini" was conceived to merge these elements into a unique bartending experience.

## 1.2 Overview of Deckbuilding and Roguelike Genres

Deckbuilding games revolve around constructing a personalized set of cards or actions, with each playthrough offering new combinations and strategies. Roguelike games are characterized by procedural generation, permadeath, and high replay value. Combining these genres allows for dynamic gameplay, where each session presents fresh challenges and opportunities for mastery.

## 1.3 Unique Aspects of Marytini

Marytini stands out by placing the player in the role of a bartender, tasked with crafting drinks based on vague, parameter-driven customer requests. The game introduces a two-tier ingredient mixing system, hidden drink objectives, and a scoring mechanism that rewards both accuracy and discovery. This blend of strategic card play, creative problem-solving, and narrative-driven interaction offers a fresh take on both genres.

## 1.4 Scope of the Work

This project encompasses the design and implementation of the Marytini game using the LOVE2D framework and Lua scripting. It covers the development of core gameplay mechanics, customer dialogue systems, scoring algorithms, and user interface elements. The work also includes balancing game parameters and integrating feedback systems to enhance player engagement.

## 1.5 Product Scenarios

Marytini is designed for players seeking a strategic and creative challenge. Scenarios include interpreting customer hints, selecting and combining ingredients from a card-based deck, and maximizing earnings through skillful drink preparation. The game supports replayability through randomized customer requests, hidden objectives, and evolving player strategies, making each play session unique and rewarding.

# Chapter 2: Literature Review

## 2.1 Related Games and Mechanics

Marytini is influenced by a number of exciting indie titles that mix innovation along with strategy and narrative involvement.

The first major inspiration is Balatro [1], a deckbuilding roguelike with deck a poker hands mechanic. Balatro is driven through a process of deck building through incremental discoveries of synergy and unpredictable outcomes, emphasizing decision making and the use case of our ingredient system and evolving drink concoction outline in Marytini's ingredient system.

Another large influence is Katana ZERO [2], which has less to do with mechanics and more to do with the narrative interaction and a hidden layer of decisions. Katana ZERO uses player choices that affect the narrative experience, as well as player circumstances and interpretations of dialog throughout each stage of the game varying subtly and at times not so subtly, additional empowerment. Similarly, Marytini draws on some vague customer dialogue, and hidden drink objectives to provide moments of emergent decision making, with players needing to rely on hidden information to anticipate what were the components of an optimal outcome.

Alchemy Dungeon [3] introduced a means of crafting as a roguelike where you could combine elements together to make new and stronger items. Marytini is borrowing two-tiered ingredient mixing. It allows players to take creative and experimental measures by identifying strategic combinations.

Papers, Please [4] connected our game, since it involved satisfying daily demands, that were specific, but in some cases ambiguous to the customer. The structure of a daily cycle in pursuit of accuracy versus efficiency to reward are demonstrated in Marytini's bartender play loop games.

Together, these games illustrate how combining simple mechanical layers (deckbuilding, narrative decisions, crafting, daily progression) can result in deep, strategic, and emotionally engaging gameplay—an approach that Marytini aims to recreate within the bartending theme.

## 2.2 Existing Bartending and Deckbuilding Games

In terms of the bartending simulation genre, VA-11 HALL-A: Cyberpunk Bartender Action [5] makes a fair standard (for example, it centers on crafting drinks to choose which narrative outcomes the player takes; these player choices will lead to different dialogue and endings). However, it is still not a true deckbuilding or roguelike structure.

Marytini hopes to take things a step further by merging the interpretations of customers from VA-11 HALL-A with a more dynamic deckbuilding and procedural system to increase replayability.

Deckbuilding mechanics have become progressively popular, e.g., with games like Slay the Spire [6], that build decks over the course of the run and have a risk/reward balance. Marytini incorporates this mechanics over across the customers and manages the ingredient decks with said orders while attempting a more strategic play method for deck performance with ingredient crafting and management.

Simple mobile games like Drink Master [7] have explored very basic drink-mixing, though these types of games primarily focus on a visual aspect and accuracy in recreating drinks rather than a taste profile or something that could entail strategy.

Marytini, in this way, is explicitly different than other mobile game titles due to a parameterized taste system (i.e., sweetness, bitterness, sourness, saltiness) and a scoring algorithm that considered more quantifiable outcomes, forcing players to think about taste outcomes and not simply visual aspects of drinks.

Marytini posits itself as a unique opportunity to bridge bartending simulation, deckbuilding strategy, and strategy for oneself and one's customers through deck management and crafting.

# Chapter 3: System Design

### 3.1 Game Architecture Overview

The architecture of Marytini is modular, separating core gameplay logic, user interface, and data management. The main game loop manages the flow of each day, handling customer generation, deck management, drink mixing, and scoring. Key modules include the customer request system, card deck and hand management, ingredient mixing logic, and the scoring and feedback system. The use of the following technological stack ensures efficient handling of game states, rendering, and user input.

1.1 Game Engine: LÖVE2D

1.2 Programming Languages: Lua (for game logic, audio/graphics implementation, scripting), Python (for Object Oriented Programming, score calculation, future scopes: NLP, API calls)

1.3 Asset Production: Asperite, Adobe Photoshop

1.4 Hosting Services : PC Web Application on itch.io using a .love file, GameJolt for demo testing

### 3.2 Class Diagrams

The system models drink parameters—sweet, salty, bitter, and sour—as classes or structured data types, each with qualitative levels: Less, Medium, and Extreme. These classes encapsulate the logic for comparing player-made drinks to customer requests and for generating descriptive feedback.

- DrinkParameter: Represents a single taste attribute (e.g., sweet) with a value and qualitative label.
- Drink: Aggregates all parameters and holds ingredient composition.
- Customer: Stores the requested drink profile and dialogue options.

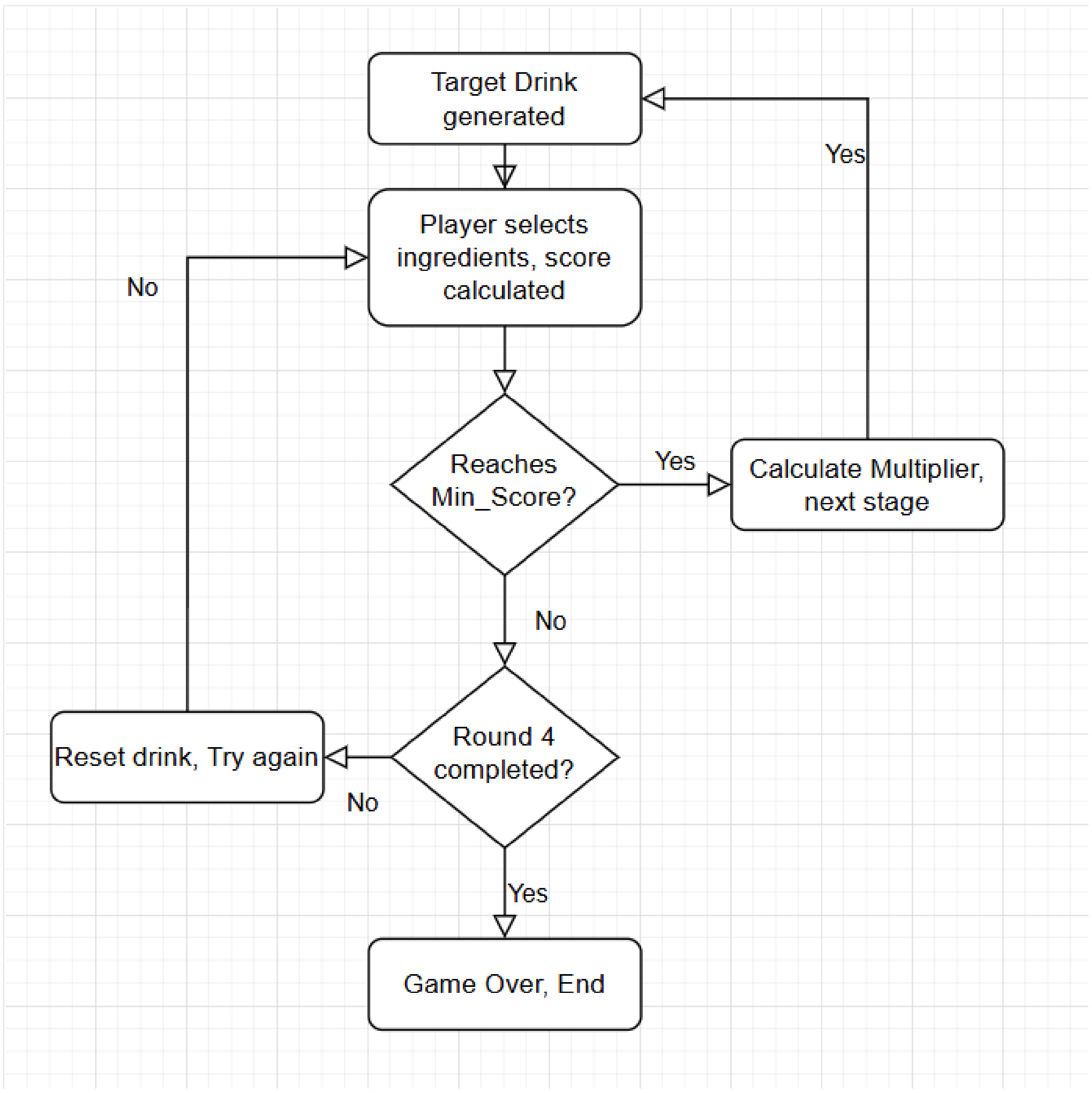


*Fig 1 Flow chart of crude gameplay*

### 3.3 Description of Ingredient and Base Classes

Ingredients and drink bases are represented as classes or objects, each with defined effects on the drink parameters.

- Ingredients: Contains properties such as name (e.g., lemon, coffee), and the quantitative effect on each parameter (e.g., lemon increases sourness).

- Base: Represents the foundational drink (vodka, rum, soda), each providing a unique starting profile for the drink parameters.

The system supports two-tier mixing, where combining two ingredients can yield a new, more complex ingredient, which is then added to the drink.

### 3.4 Dialogue System for Customer Requests

The dialogue system dynamically generates customer requests based on randomly assigned drink parameter profiles. For each customer, the system selects or constructs dialogue that hints at the desired taste profile using qualitative descriptors (e.g., "I want something a little sweet and very sour"). The dialogue system also provides hints for hidden drink objectives, encouraging players to experiment and discover special recipes. This system enhances immersion and ensures that each customer interaction feels unique and contextually relevant.

# Chapter 4: Implementation

## 4.1 Technologies Used

Marytini was implemented using the LOVE2D framework, a common engine for 2D game development, and Lua, a lightweight scripting language that happens to be much faster and flexible than used commonly. These technologies are both useful for developing Marytini at its rapid prototyping speed, handling a range of game states efficiently, and rendering graphics and user interfaces with ease.

## 4.2 Game Flow

The game starts each new day with a number of customers. Players assume the role of the bartender, and each customer has a drink order that will be given in qualitatively defined parameters like sweet, salty, bitter, and sour etc. The player chooses a drink base (which also depends on a number of other parameters) and then selects ingredient cards from their hand to mix that drink. After mixing is done, the game automatically compares the prepared drink to the customer order, and calculates the score and monetary reward. The day ends after all customers are served, and the player attempts to maximize their monetary reward.

## 4.3 Deck and Card Mechanics

Players are given a deck of ingredient cards. Each ingredient card is an individual ingredient, such as lemon, coffee, salt, or sugar. There will be a hand of cards drawn from the ingredient deck at the beginning of each round and then the player will choose which cards to play to reflect the customer's taste profile. The deckbuilding aspect adds strategic variety and re-playability because players can adapt ideas using the cards available to them and based on the requests.

## 4.4 Two-Tier Mixing Logic

A unique feature of Marytini is its two-tier mixing system, inspired by the logic implemented in scorecheck.py. In the first tier, players can combine two ingredient cards to create a new, more complex ingredient. This new ingredient can then be added to the main drink in the second tier, allowing for deeper strategy and more nuanced flavor profiles. This system encourages experimentation and rewards players for discovering effective ingredient combinations.

## 4.5 Hidden Drink System and Hints

To increase the challenge and reward, each customer can also have one or more hidden drinks that also meet their preferences. The game gives clues, either verbally or through visuals, as to what those hidden drinks could be. When the player is successful at making a hidden drink, they receive bonus money, which creates an incentive to explore options and listen carefully. This feature allows for greater replayability and helps shape creative thinking around mix orders.

Ultimately, the execution of Marytini combines solid game mechanics, strategic decision-making, and engaging feedback systems to provide an exciting bartending experience.

# Chapter 5: Gameplay and User Experience

## 5.1 Player Actions and Decision-Making

- At the start of each day, the player is presented with a series of customers, each requesting a drink with specific taste parameters (sweet, salty, bitter, sour).
- The player selects one of three available drink bases: vodka, rum, or soda.
- The player draws a hand of ingredient cards from their deck (e.g., lemon, coffee, salt, sugar).
- The player chooses which ingredient cards to play, either adding them directly to the drink or combining two cards to create a new ingredient (two-tier mixing).
- The player can interpret customer dialogue and hints to deduce both the explicit and hidden drink preferences.
- Once satisfied, the player serves the drink to the customer.

## 5.2 Scoring and In-game Feedback

- After serving, the game compares the prepared drink's parameters to the customer's request.
- Points are awarded based on how closely the drink matches the requested taste profile.
- Additional points and rewards are given if the player successfully creates a hidden drink, as indicated by hints.
- The game provides immediate feedback, including customer reactions and a breakdown of the score.
- Constructive feedback helps the player understand where they succeeded or missed the mark, encouraging learning and improvement.

## 5.3 Money and Progression System

- Each successful drink earns the player money, with higher accuracy and hidden drinks yielding greater rewards.
- Poorly matched drinks result in lower earnings.
- The player's total money is tracked throughout the day and across multiple days.
- The primary goal is to maximize earnings by the end of each day.
- Accumulated money can be used for progression elements, such as unlocking new ingredients, upgrading the deck, or accessing special scenarios (if implemented).
- The game's progression system motivates players to refine their strategies and experiment with different ingredient combinations for optimal results.

# Chapter 6: Testing and Feedback

## 6.1 Playtesting

Playtesting was done during the creation of "Marytini," to make sure the game mechanics were fun and easy to use. To get a range of opinions, playtesting sessions included both external and internal team members. The main objective was to assess the user interface, general player experience, and drink mixing mechanisms. Testers were urged to experiment with various drink-mixing techniques in order to get goal scores, which gave them important information on the game's difficulty and degree of enjoyment.

## 6.2 Balancing

A crucial part of game creation was maintaining the game's balance to guarantee fun and fairness. The score structure was designed to preserve a manageable difficulty curve while rewarding players for deft mixing. Feedback from playtesting was used to modify the multipliers for stages and rounds. This made it possible for players to go through the game without running into insurmountable obstacles and still feel proud of themselves for learning the concepts.

## 6.3 Adjustments

Based on the feedback from playtesting. The initial ingredient bank was expanded to offer more strategic choices, and the scoring system was refined to better reflect the complexity of the drinks created. Additionally, the user interface was improved to provide clearer feedback on player actions and scores. These adjustments were aimed at creating a more engaging and rewarding gameplay experience, encouraging players to experiment with different ingredient combinations and strategies.

# Chapter 7: Conclusion and Future Work

## 7.1 Summary of Achievements

- Successfully designed and implemented "Marytini," a deckbuilding roguelike bartending game using the LOVE2D framework and Lua scripting.
- Developed a modular game architecture with clear separation of gameplay logic, user interface, and data management.
- Implemented core mechanics including customer request generation, card-based ingredient selection, two-tier mixing logic, and a dynamic scoring system.
- Created a dialogue system that provides players with hints and feedback, enhancing immersion and replayability.
- Incorporated hidden drink objectives and a progression system based on player earnings, encouraging strategic experimentation.

## 7.2 Potential Improvements and Future Work

### 7.2.1 Dialogue Option Expansion Using Language Models

- Integrate advanced language models to generate more varied, random, and intuitive customer dialogues.
- Enhance player immersion by making customer requests feel more natural and unpredictable.
- Allow for dynamic hint generation and richer narrative interactions.

### 7.2.2 Implementing Other Mechanics Such as Story-Based Progression

- Introduce a story mode with branching narratives and character development.
- Add unique events, challenges, and milestones that influence gameplay and player choices.
- Provide long-term goals and unlockable content to further motivate player engagement.

### 7.2.3 Additional Enhancements

- Expand the ingredient pool and introduce new card types for greater strategic depth.
- Develop multiplayer modes for cooperative or competitive play.
- Refine the user interface and visual feedback for a more polished player experience.
- Incorporate player feedback and analytics to continuously balance and improve game mechanics.

These future directions aim to make Marytini even more engaging, replayable, and enjoyable for a wide range of players.

## References

[1] LocalThunk, "Balatro", Playstack, 2024.
[2] Askiisoft, "Katana ZERO", Devolver Digital, 2019.
[3] Nukearts, "Alchemy Dungeon", Nukearts, 2020.
[4] Lucas Pope, "Papers, Please", 3909 LLC, 2013.
[5] Sukeban Games, "VA-11 HALL-A: Cyberpunk Bartender Action", Ysbryd Games, 2016.
[6] MegaCrit, "Slay the Spire", Humble Games, 2019.
[7] SayGames Ltd., "Drink Master", SayGames, 2021.

## Annexures

- UI mockups

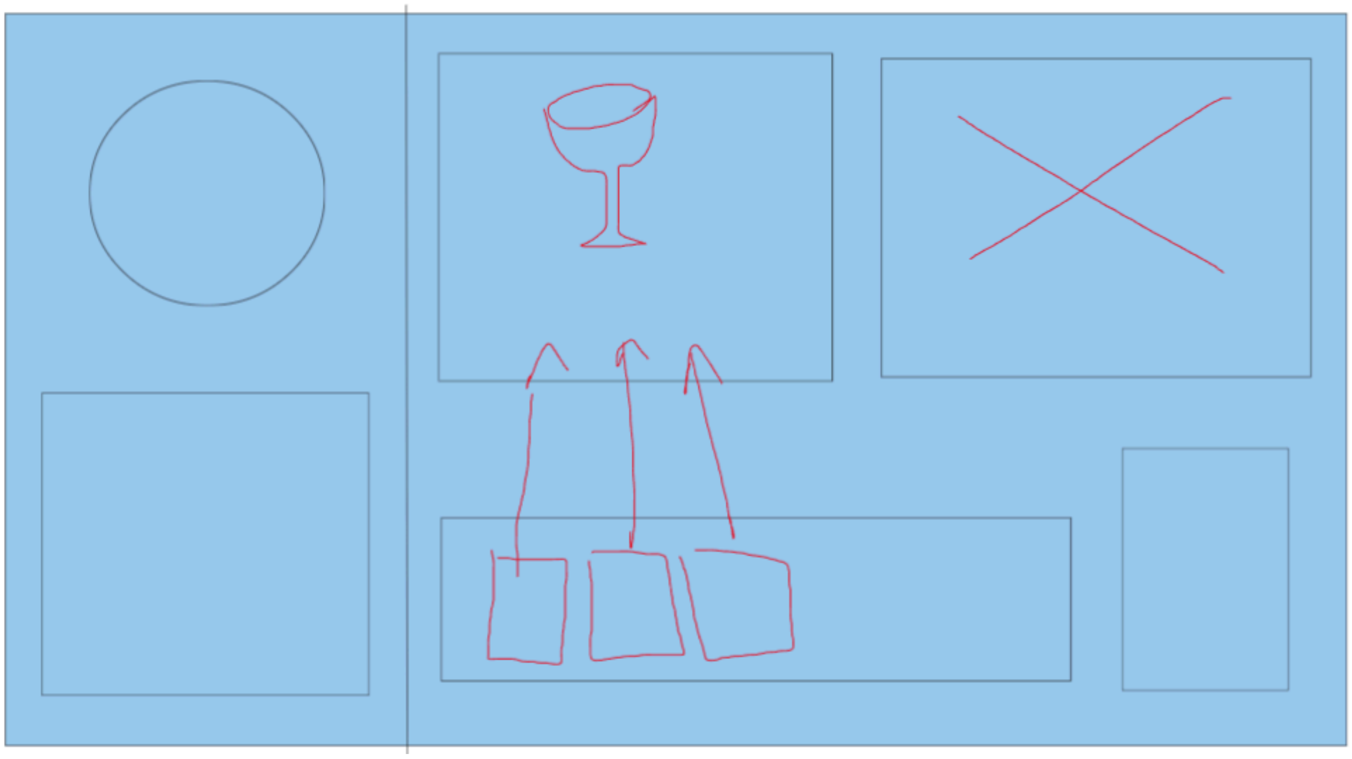


*Fig 2 Original Mockup for UI*

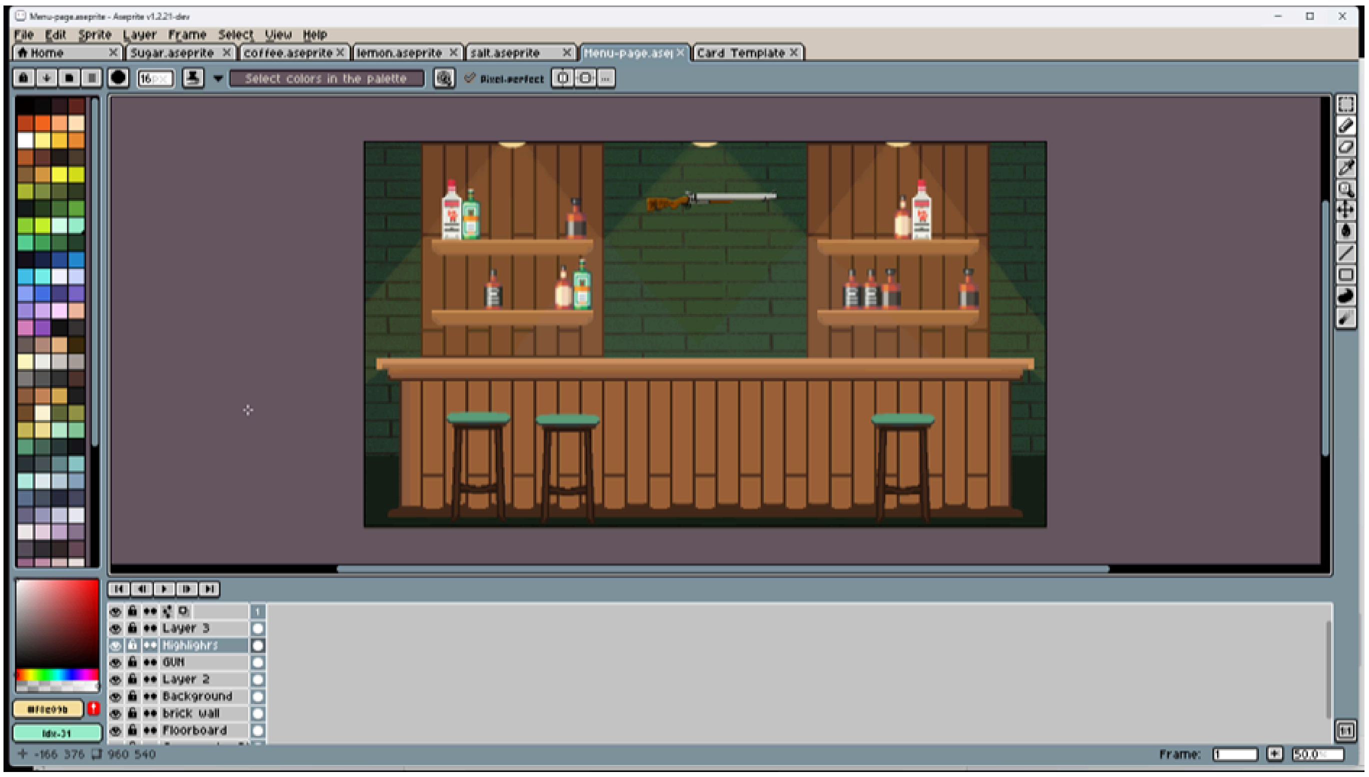
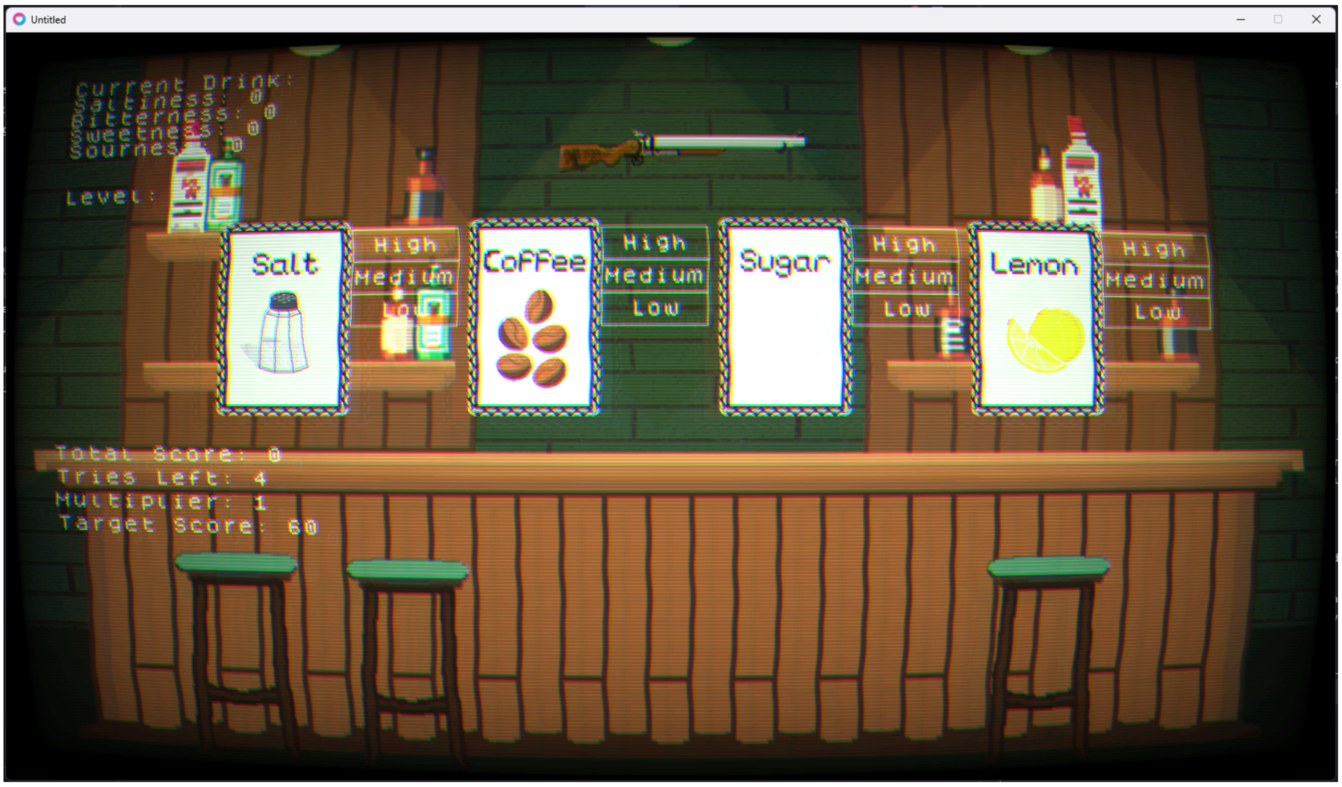


*Fig 3 Menu Background*



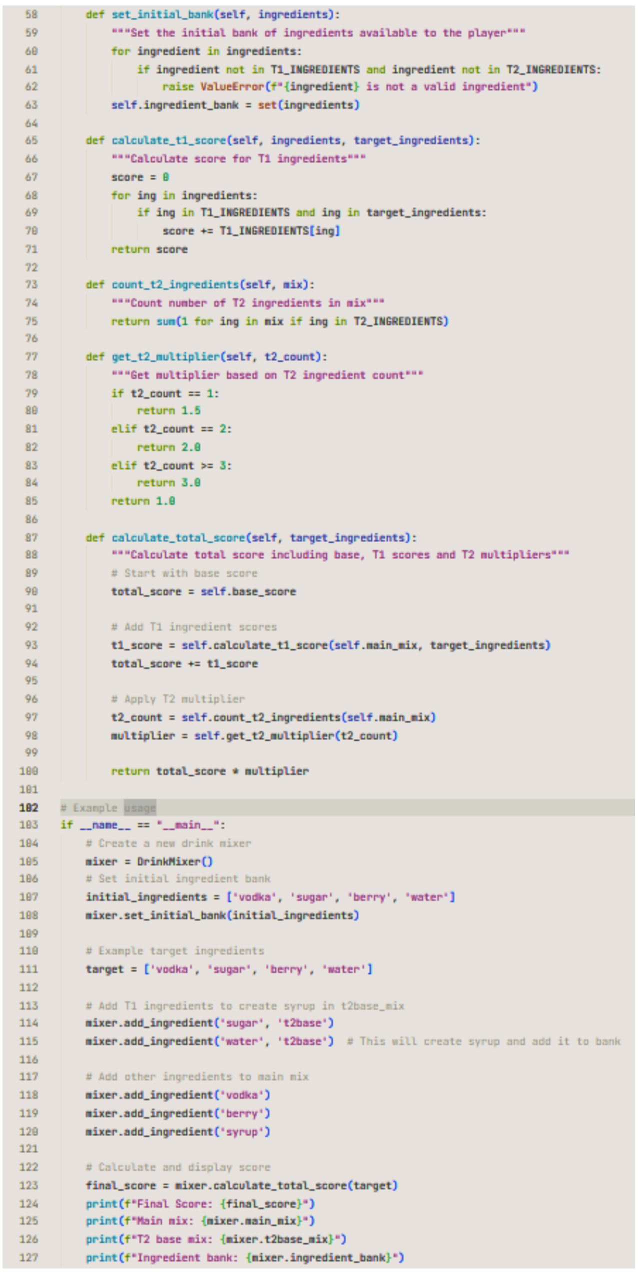
Current Drink:
Saltiness: 0
Bitterness: 0
Sweetness: 0
Sourness: 0

Level:

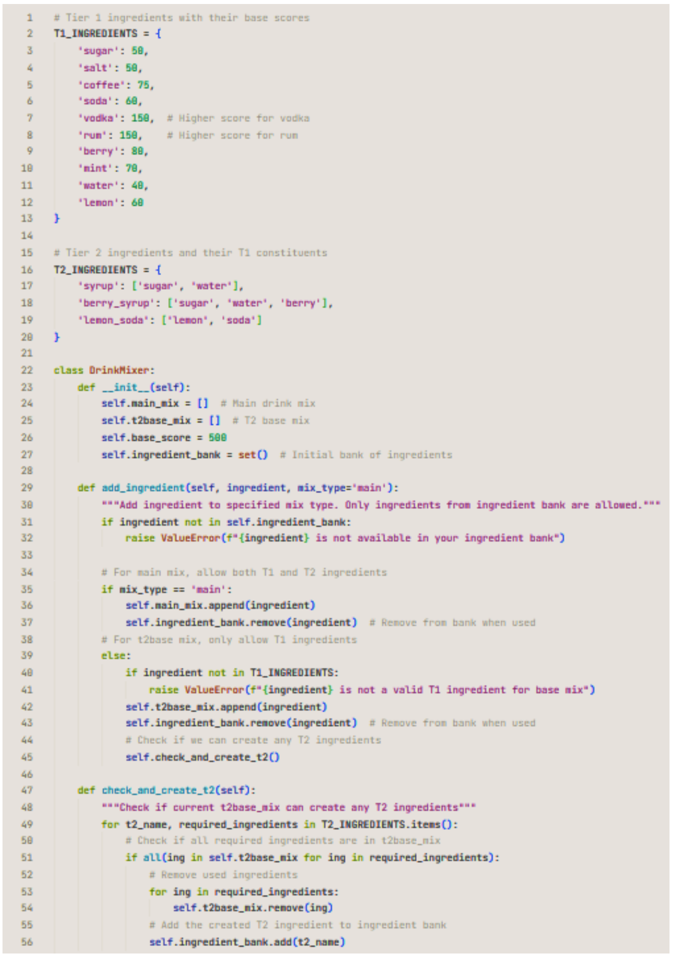
Total Score: 0
Tries Left: 4
Multiplier: 1
Target Score: 60

*Fig 4 Interactive UI for Game and selection*

- Sample code snippets



```python
    def set_initial_bank(self, ingredients):
        """Set the initial bank of ingredients available to the player"""
        for ingredient in ingredients:
            if ingredient not in T1_INGREDIENTS and ingredient not in T2_INGREDIENTS:
                raise ValueError(f"{ingredient} is not a valid ingredient")
        self.ingredient_bank = set(ingredients)

    def calculate_t1_score(self, ingredients, target_ingredients):
        """Calculate score for T1 ingredients"""
        score = 0
        for ing in ingredients:
            if ing in T1_INGREDIENTS and ing in target_ingredients:
                score += T1_INGREDIENTS[ing]
        return score

    def count_t2_ingredients(self, mix):
        """Count number of T2 ingredients in mix"""
        return sum(1 for ing in mix if ing in T2_INGREDIENTS)

    def get_t2_multiplier(self, t2_count):
        """Get multiplier based on T2 ingredient count"""
        if t2_count == 1:
            return 1.5
        elif t2_count == 2:
            return 2.0
        elif t2_count >= 3:
            return 3.0
        return 1.0

    def calculate_total_score(self, target_ingredients):
        """Calculate total score including base, T1 scores and T2 multipliers"""
        # Start with base score
        total_score = self.base_score

        # Add T1 ingredient scores
        t1_score = self.calculate_t1_score(self.main_mix, target_ingredients)
        total_score += t1_score

        # Apply T2 multiplier
        t2_count = self.count_t2_ingredients(self.main_mix)
        multiplier = self.get_t2_multiplier(t2_count)

        return total_score * multiplier

# Example usage
if __name__ == "__main__":
    # Create a new drink mixer
    mixer = DrinkMixer()
    # Set initial ingredient bank
    initial_ingredients = ['vodka', 'sugar', 'berry', 'water']
    mixer.set_initial_bank(initial_ingredients)

    # Example target ingredients
    target = ['vodka', 'sugar', 'berry', 'water']

    # Add T1 ingredients to create syrup in t2base_mix
    mixer.add_ingredient('sugar', 't2base')
    mixer.add_ingredient('water', 't2base')  # This will create syrup and add it to bank

    # Add other ingredients to main mix
    mixer.add_ingredient('vodka')
    mixer.add_ingredient('berry')
    mixer.add_ingredient('syrup')

    # Calculate and display score
    final_score = mixer.calculate_total_score(target)
    print(f"Final Score: {final_score}")
    print(f"Main mix: {mixer.main_mix}")
    print(f"T2 base mix: {mixer.t2base_mix}")
    print(f"Ingredient bank: {mixer.ingredient_bank}")
```



```python
# Tier 1 ingredients with their base scores
T1_INGREDIENTS = {
    'sugar': 50,
    'salt': 50,
    'coffee': 75,
    'soda': 60,
    'vodka': 150,  # Higher score for vodka
    'rum': 150,    # Higher score for rum
    'berry': 80,
    'mint': 70,
    'water': 40,
    'lemon': 60
}

# Tier 2 ingredients and their T1 constituents
T2_INGREDIENTS = {
    'syrup': ['sugar', 'water'],
    'berry_syrup': ['sugar', 'water', 'berry'],
    'lemon_soda': ['lemon', 'soda']
}

class DrinkMixer:
    def __init__(self):
        self.main_mix = []  # Main drink mix
        self.t2base_mix = []  # T2 base mix
        self.base_score = 500
        self.ingredient_bank = set()  # Initial bank of ingredients

    def add_ingredient(self, ingredient, mix_type='main'):
        """Add ingredient to specified mix type. Only ingredients from ingredient bank are allowed."""
        if ingredient not in self.ingredient_bank:
            raise ValueError(f"{ingredient} is not available in your ingredient bank")

        # For main mix, allow both T1 and T2 ingredients
        if mix_type == 'main':
            self.main_mix.append(ingredient)
            self.ingredient_bank.remove(ingredient)  # Remove from bank when used
        # For t2base mix, only allow T1 ingredients
        else:
            if ingredient not in T1_INGREDIENTS:
                raise ValueError(f"{ingredient} is not a valid T1 ingredient for base mix")
            self.t2base_mix.append(ingredient)
            self.ingredient_bank.remove(ingredient)  # Remove from bank when used
            # Check if we can create any T2 ingredients
            self.check_and_create_t2()

    def check_and_create_t2(self):
        """Check if current t2base_mix can create any T2 ingredients"""
        for t2_name, required_ingredients in T2_INGREDIENTS.items():
            # Check if all required ingredients are in t2base_mix
            if all(ing in self.t2base_mix for ing in required_ingredients):
                # Remove used ingredients
                for ing in required_ingredients:
                    self.t2base_mix.remove(ing)
                # Add the created T2 ingredient to ingredient bank
                self.ingredient_bank.add(t2_name)
```

*Fig 5 & 6 Code Snippet for Complex Score Calculation*

- Additional diagrams

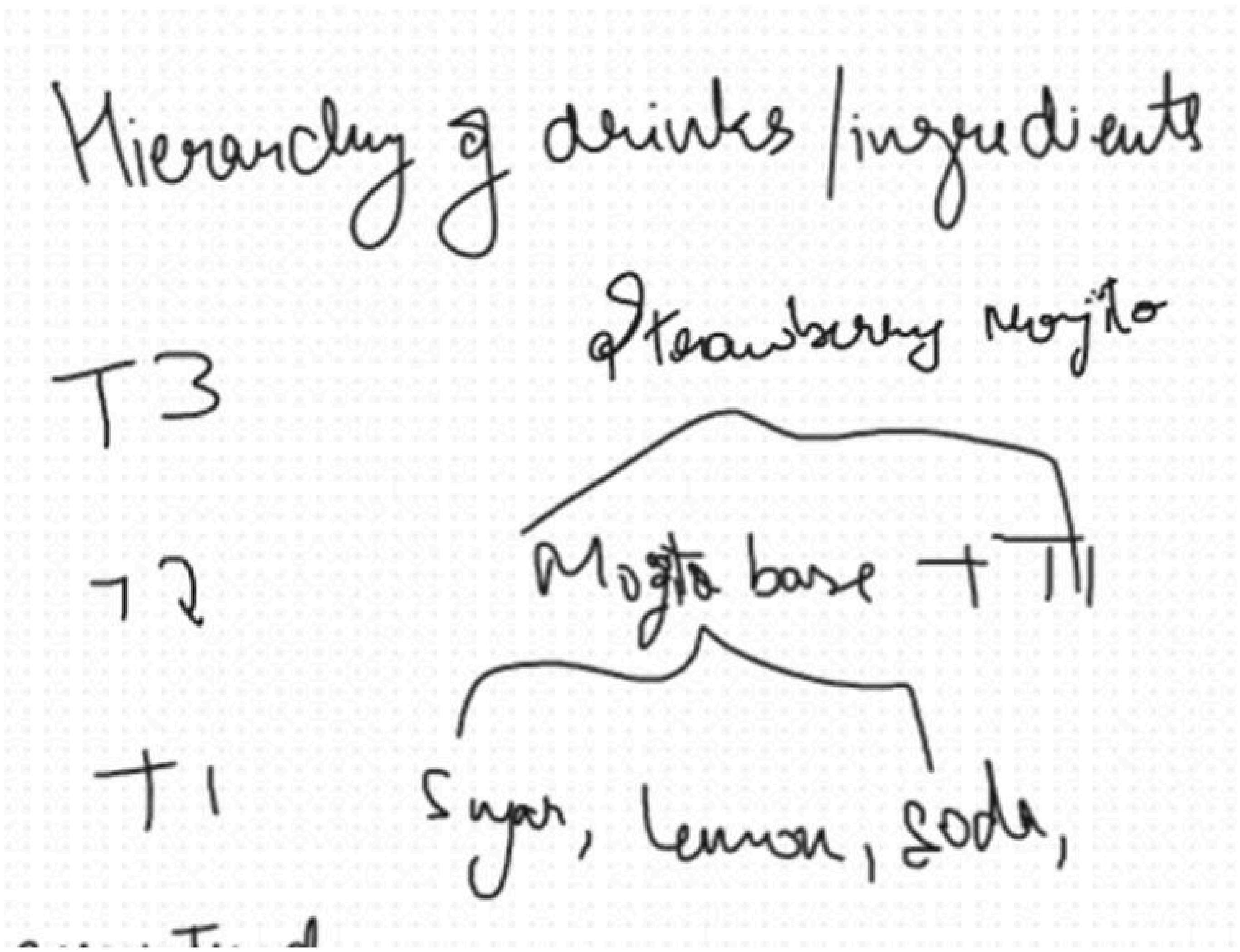
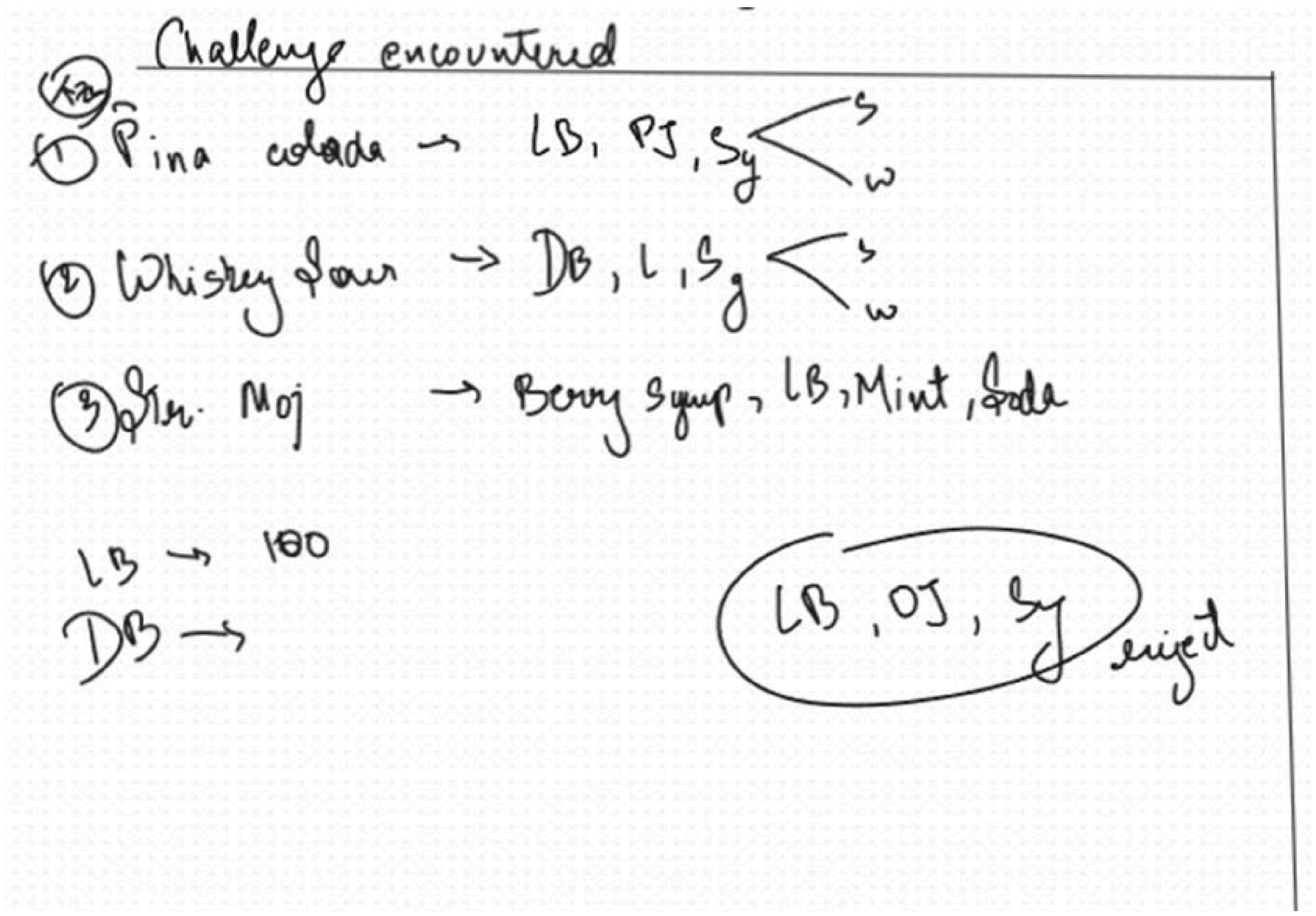


*Fig 7 Game Design thought process*



Challenge encountered

(Ex)

① Pina colada → LB, PJ, Sy < s / w

② Whiskey Sour → DB, L, Sg < s / w

③ Str. Moj → Berry Syrup, LB, Mint, Soda

LB → 100

DB →

LB, OJ, Sy Rejected

*Fig 8 Game Design thought process*

-Inspired from ‘Balatro’, ‘Katana ZERO’, ‘Alchemy Dungeon’ and ‘Papers Please’.

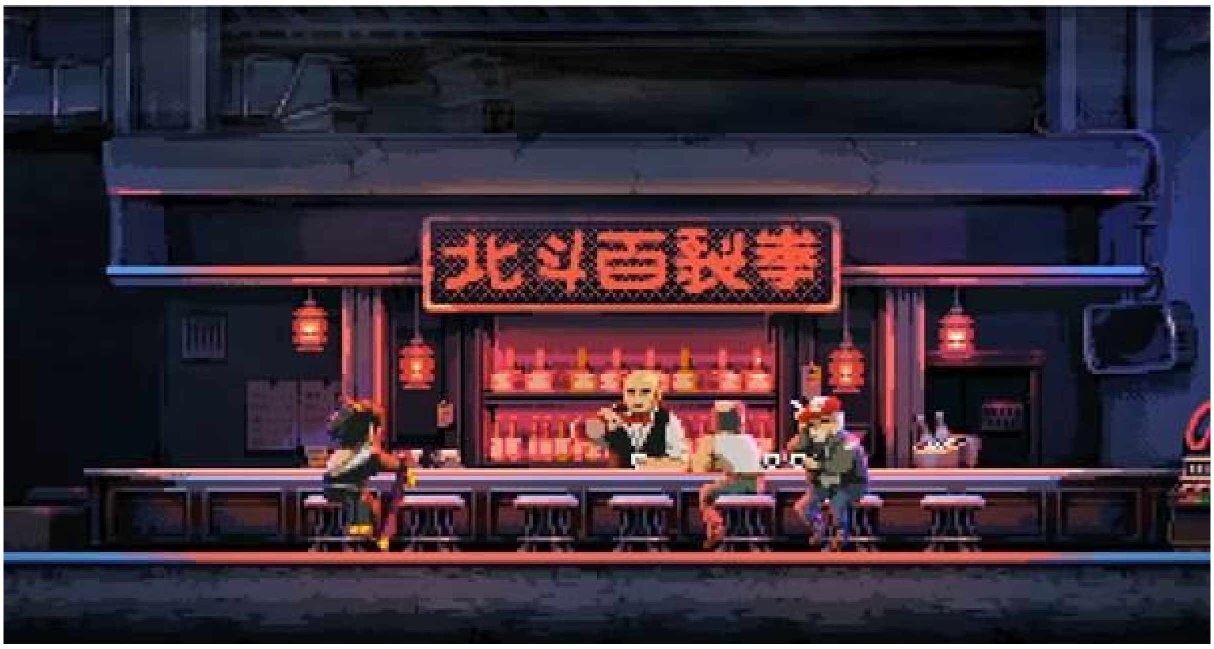


*Fig 9 A temporary bar setting seen in Katana ZERO*

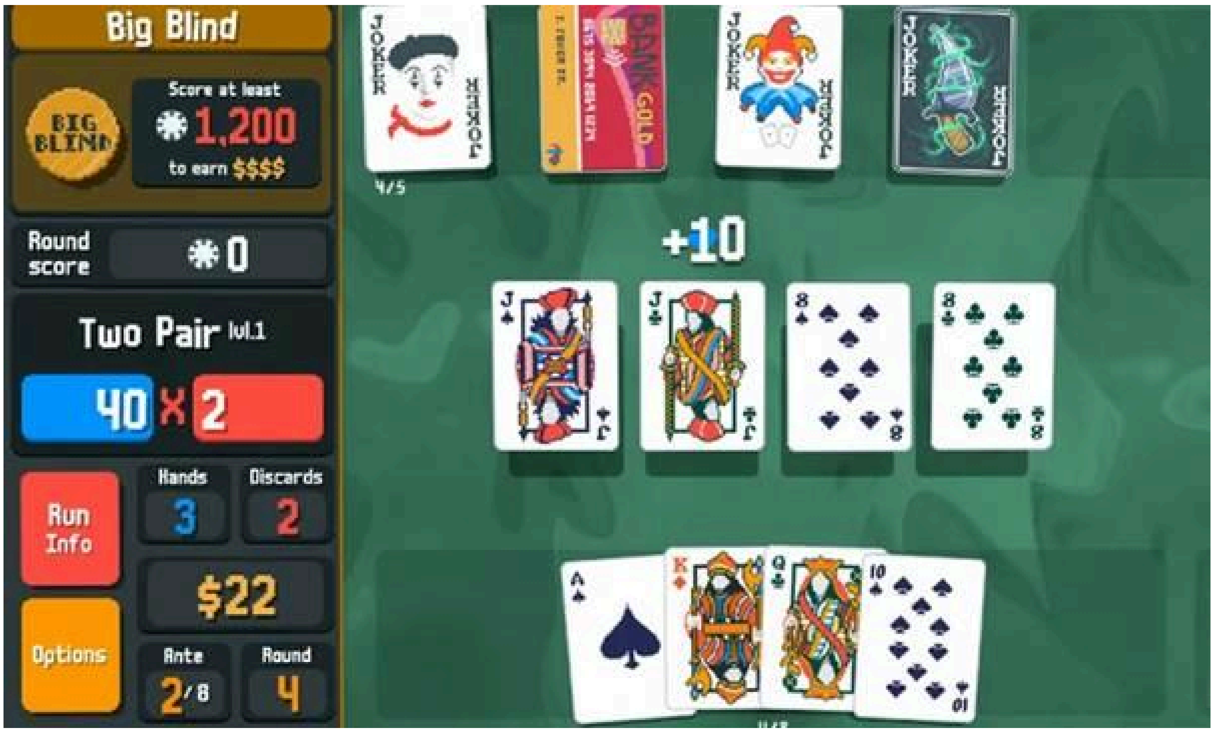


*Fig 10 Balatro (genre defining card deck-building roguelike made by indie developer LocalThunk)*

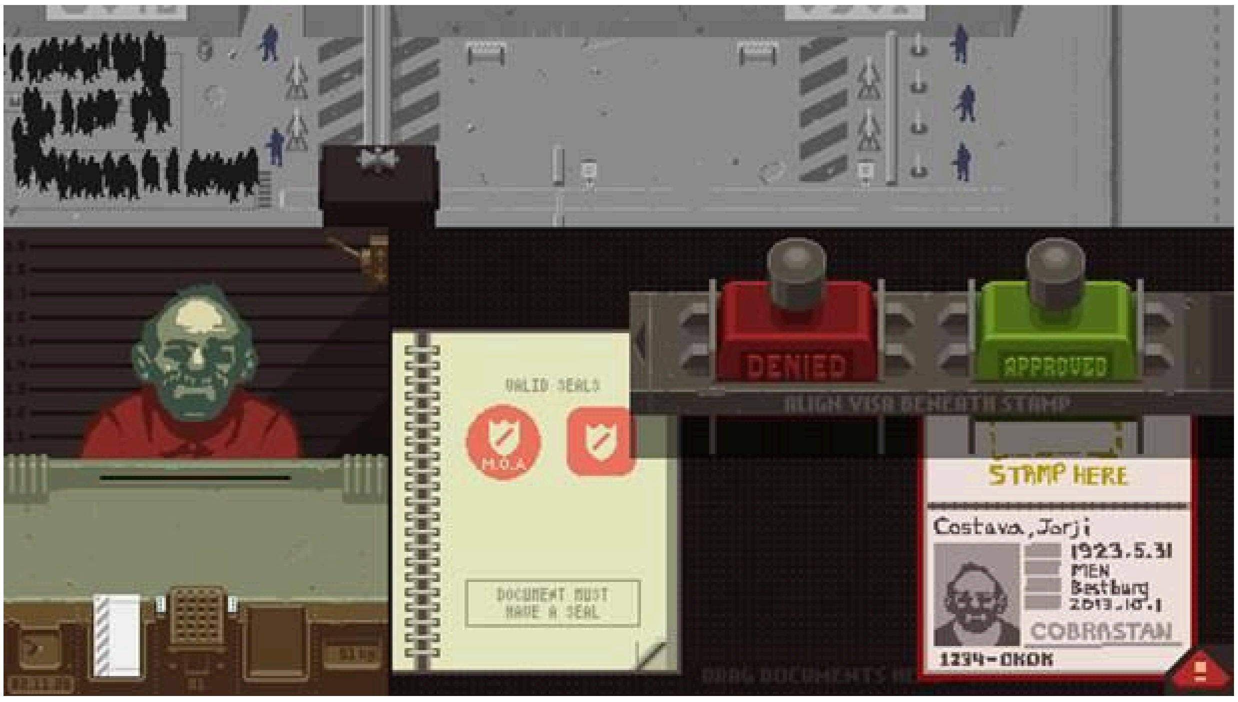


*Fig 11 The multi-screen UI from Paper's Please (made by solo indie developer Lucas Pope)*