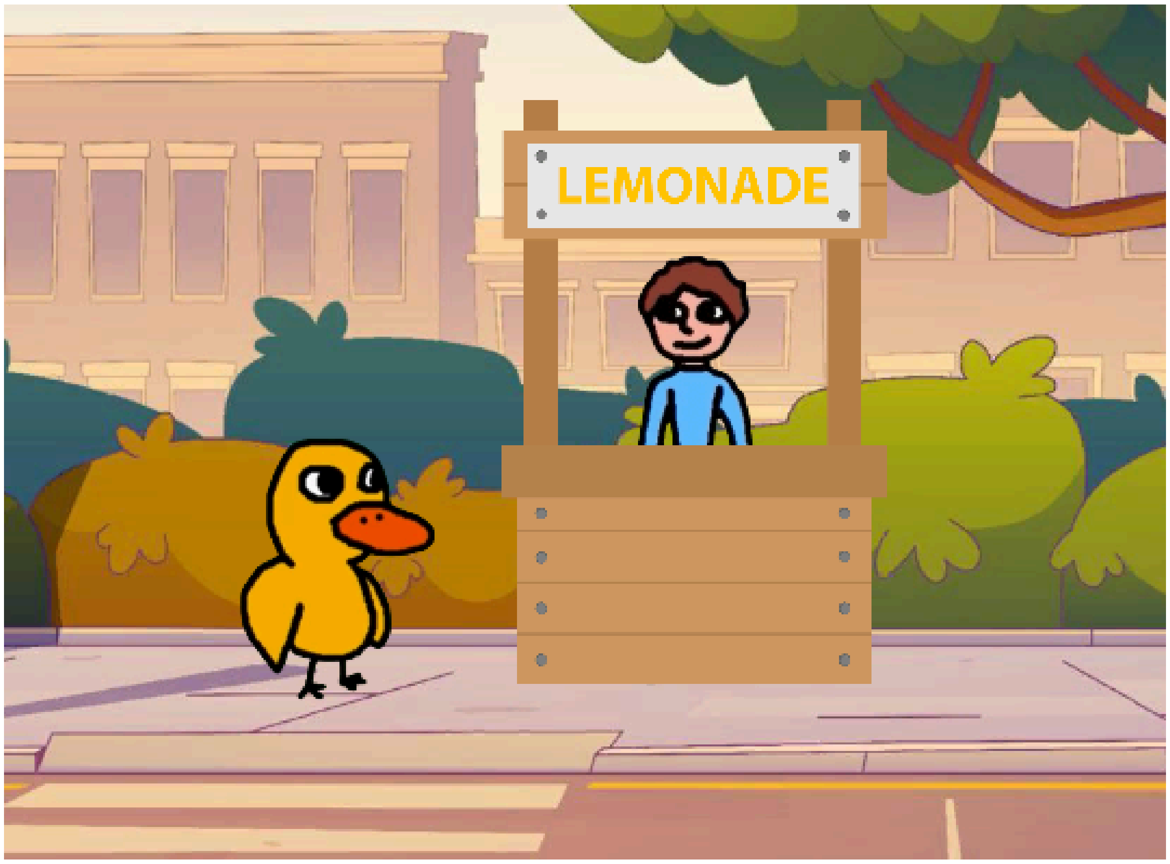
SQUEEZE! Coding Final Report



A game about running a lemonade stand and learning financial values.

Group 2 - Jose Bolanos, Alejandro Bravo, Diego Bravo, Nick Filipov

CS 442
University of Illinois at Chicago

Spring 2025

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

SQUEEZE! is an engaging and educational 2D game designed to teach financial literacy and business acumen through the dynamic experience of running a lemonade stand. From understanding basic financial principles to making strategic decisions, the game aims to provide a fun and interactive way to develop real-world skills.

The original overview details can be found in the original design document of SQUEEZE! [5]

2 Project Domain

This project produces an educational-level 2D game that aims to teach young people financial literacy by practicing business in a fun and colorful simulation environment. The goal is to produce a simple game that just about anyone can pick up and learn in a matter of minutes, which allows the focus of the game to be learning basic financial skills.

3 Relationship to Other Documents

Group 16's "SQUEEZE! Final Development Report" from Spring 2025 [5]

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Probability: The game system will take into consideration every interaction between the player and customers and business endeavors to change the probability of success of the player's business. This probability controls how many customers come in. If the customers agree with the pricing, the probability of successful sales increases. This would be built using a set of mathematical rules that attempt to model real-world economics, integrate players' business and finance decisions, and affect sales in a dynamic and econometric way.

Customers: Characters in the game, the ones who approach the lemonade stand during the day to order a product from the lemonade stand. They are randomized characters.

Products: This is what is sold at the player's lemonade stand. We have implemented a handful of products, such as Lemonade (and variations of Lemonade), berries (such as strawberries, raspberries, and grapes), and Tea.

Ingredients: These are things that make products in the game. Some ingredients can be sold as products as is, but ingredients like sugar and lemons cannot be sold and can only be made into lemonade.

Crafting: This is the system the player will use to create products out of ingredients. Players must buy a sufficient amount of ingredients in order to craft products.

Pricing: The sale price of products can be changed by the player so they can aim to make more profit per sale. Prices that differ from the default can affect probabilities such as the success rate of sales.

Timer: Customers each have a timer that will start when their order is taken and end within a set amount of time if they are not served their order before then. If the timer runs out before the customer is served, the rating will decrease. Serving a customer successfully before the timer ends will increase the rating.

Rating: This 5-star based system is affected by the success of selling products to customers and serving customers before the timer ends. This system affects the customer amount in a day, which could result in a loss state if the rating reaches 0.

Day: When the game starts, it is day 1, and a day ends when there are no customers left. The player must start the next day through the pricing screen. Every new day that starts will add a bonus customer to the daily amount. This bonus can be dynamically affected throughout the day depending on the rating. Some days unlock new ingredients and products.

Graphs: When the day ends, the customer can review their financial statistics through graphs that display each transaction in a day and daily balances. This allows the player to assess their financial situation and determine if their pricing is profitable enough.

Feedback: The player will receive feedback on their financial and gameplay performance, which will let them know about what areas they need to improve on and what they can change, such as pricing.

Graphical User Interface: The interface where the player may serve customers, buy and sell products, change prices, and view financial graphs. This is the graphical element of the game.

4b UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [4]. Any exceptions are noted where used.

4c Data Dictionary for Any Included Models

Customer Amount = 3 + (Day * Rating/100)

Rating Change = + or - 10

Rating Range = 0-100 (converted to display 5-star rating)

II Project Deliverables

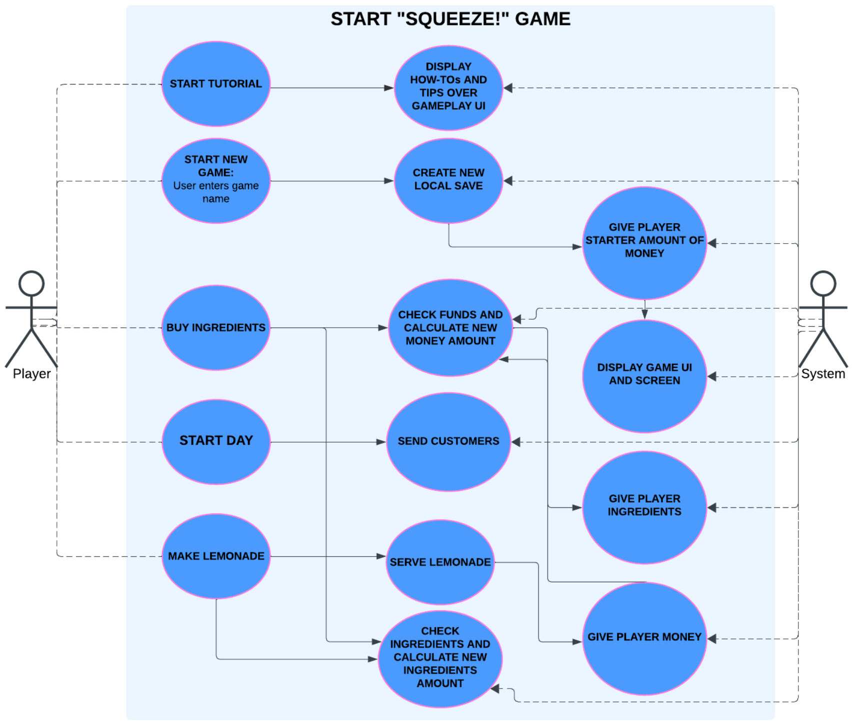
We have produced a simple 2D educational game that explores basic financial practices that may be encountered in running a small business like a lemonade stand. Players can buy ingredients, craft products, and sell products to multiple customers in a day while dealing with challenging factors such as probability and timers, which represent customer satisfaction. Keeping customers satisfied will maintain a high rating, which affects the number of customers that are seen in a day. When the day ends, the player can review their financial situation through graphs and feedback. This will allow them to assess if prices need to be changed and if they need to improve their business practice to maintain a better rating. Players will also be able to change prices of their products to aim for higher profits, but at the cost of customers being dissatisfied with said prices. After changing prices, the player may start a new day to continue playing.

This describes the typical gameplay loop of our final SQUEEZE! deliverable.

1 First Release

February 28, 2025: In this release, the player could start the game, buy ingredients, craft lemonade, and sell products to one customer who would remain in front of the stand. This allowed developers to test and implement an instance of the typical gameplay loop. It also allowed developers to learn the basics of developing a 2D game on Unity. Besides that, the main focus was to create the gameplay GUI that suits SQUEEZE!

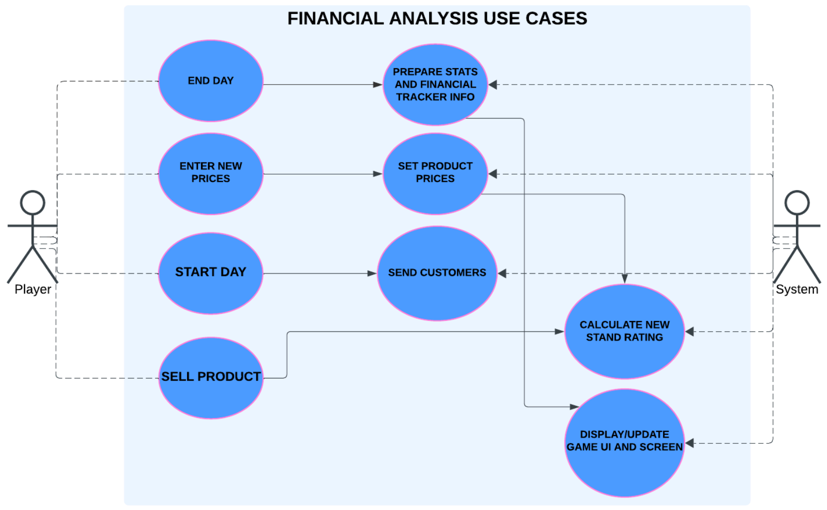
Figure 1 - Use Cases: Starting the game and playing the first day.


START "SQUEEZE!" GAME
START TUTORIAL
DISPLAY HOW-TOs AND TIPS OVER GAMEPLAY UI
START NEW GAME: User enters game name
CREATE NEW LOCAL SAVE
GIVE PLAYER STARTER AMOUNT OF MONEY
Player
System
BUY INGREDIENTS
CHECK FUNDS AND CALCULATE NEW MONEY AMOUNT
DISPLAY GAME UI AND SCREEN
START DAY
SEND CUSTOMERS
GIVE PLAYER INGREDIENTS
MAKE LEMONADE
SERVE LEMONADE
GIVE PLAYER MONEY
CHECK INGREDIENTS AND CALCULATE NEW INGREDIENTS AMOUNT

2 Second Release

April 4, 2025: In this release, the player would be welcomed with a play screen and have the option to modify audio in the settings, and be shown instructions on how to play the game. You were now able to play for multiple days and were able to unlock new products. Graphs were introduced to give the users insight into what to do to earn more money and how they did in a day. Overall, the main focus was on making the gameplay a lot better and providing the educational aspect of the game.

Figure 2 - Financial Analysis Use Case Diagram


FINANCIAL ANALYSIS USE CASES
END DAY
PREPARE STATS AND FINANCIAL TRACKER INFO
ENTER NEW PRICES
SET PRODUCT PRICES
START DAY
SEND CUSTOMERS
CALCULATE NEW STAND RATING
SELL PRODUCT
DISPLAY/UPDATE GAME UI AND SCREEN
Player
System

3 Comparison with Original Project Design Document

Differences:

- In section I.1 (Project Overview) of the original design document [5], SQUEEZE! is described as a first-person experience, presumably in 3D. The game we developed is not in first person nor in 3D. We made a 2D simulation-style game.
- In the same section specified above, a feature that would resemble an Excel spreadsheet would have been included in the game for most financial analysis and decision-making activities. We did not implement any spreadsheet feature since we believed it would have made the player experience more overwhelming than fun. Instead, the pricing entry is simply typed in for each product, and some graphs do the financial calculations for the player so that they can analyze their performance with ease.
- The overview also specified that the product is meant to be sold to colleges and high schools for advanced financial education. We scaled down the product to resemble a game that is aimed toward younger students who could use introductory-level education in financial endeavors.

Similarities:

- Our game gives the player full control over a simulated lemonade stand environment by letting them change prices, buy and sell products, and do financial analyses as the original design intended.
- The game does offer some form of a financial advisor in the form of feedback that is dynamically given to the player at the end of each day.
- The game strives to be engaging while creating an educational experience for the player that will build on their financial skills.

III Testing

1 Items to be Tested

Virtually every component of the program was tested by all team members. Each of us worked with a synchronized copy of the application, pulled from a shared collaborative repository. The items tested included, but were not limited to:

- The welcome screen's play button initializes the game
- The welcome screen's play button initializes the game
- Inventory updates corresponding to purchases
- Players get money from successful sales
- Players get a positive star rating from successful and fast sales
- Players get no money from unsuccessful and slow sales
- The timer above the player counts time accurately.
- Players get a negative start rating from unsuccessful and slow sales
- Players are able to craft items successfully
- Players receive accurate data on how sales went after a day has over
- Players receive good and accurate financial advice at the end of each day.

2 Test Specifications

0 - Star Screen and Buttons

Description: Users should be able to either start the game or go to the screen that displays instructions on how to play the game.

Items covered by this test: Play game, and how to play buttons

Requirements addressed by this test: RQ-01, RQ-02

Environmental needs: Unity and Visual Studio Code.

Intercase Dependencies: N/A

Test Procedures:

1) Launch the application.
2) Wait for the welcome screen to load.
3) Click the "Play" button.

4) Observe the transition from the welcome screen to the game screen.
5) Verify that the game timer starts and player controls become active.

Input Specification: User clicks the "Play" button using a mouse.

Output Specifications:

- Screen transitions to main gameplay interface
- Timer bar begins counting from 10 seconds
- Player can take customer order and interact with items

Pass/Fail Criteria:

Pass

- The screen successfully changes from the welcome screen to the game screen
- The game timer bar starts at 10
- Player control is responsive immediately after the transition

Fail

- The screen does not change
- The timer does not start or starts incorrectly
- Player control is non-responsive or delayed

<u>1 - Crafting System</u>

Description: Players should be able to craft items successfully using available resources.

Items covered by this test: Crafting mechanism, Inventory update, Resource deduction

Requirements addressed by this test: R4.3, R5.1

Environmental needs: Unity and Visual Studio Code.

Intercase Dependencies: N/A

Test Procedures:

1) Start the game and reach a point where crafting is available.
2) Ensure the player has the required resources in inventory.
3) Open crafting UI and select a craftable item.
4) Click the "Craft" button.
5) Check inventory for the new item and reduced resources.

Input Specification: Enough in-game currency and resources (lemons, sugar, etc.)

Output Specifications: Item crafted successfully, resources deducted, UI feedback (text, updated menu)

Pass/Fail Criteria:

Pass

- Crafted item is added to inventory.
- Used resources are subtracted correctly.
- Visual/auditory feedback confirms successful crafting.

Fail

- Item is not added to inventory.
- Resources are not deducted or deducted incorrectly.
- No feedback or error shown.

<u>2 - Day-End Sales Report</u>

Description: Players should receive an accurate summary of their sales, star rating, and earnings at the end of each game day.

Items covered by this test: Sales tracking, Rating system, Financial summary

Requirements addressed by this test: R6.1, R6.2

Environmental needs: Unity and Visual Studio Code.

Intercase Dependencies: Must complete at least one full game day (#1 can be helpful)

Test Procedures:

- Sell at least one item during the game day.
- Let the timer expire and trigger the end-of-day summary
- Review summary for accuracy.

Input Specification: Item sales (both successful and failed)

Output Specifications: End-of-day screen showing number of sales, total money earned, and star rating.

Pass/Fail Criteria:

Pass

- All successful and failed sales are shown.

- Star rating matches performance.
- Money total is correct.

Fail

- Summary is missing sales data.
- Star rating is inconsistent with performance.
- Earnings are incorrect.

3 - Financial Advice

Description: After each game day, the player receives tailored financial advice based on their sales performance.

Items covered by this test: Financial advisor AI logic, UI feedback

Requirements addressed by this test: R6.3

Environmental needs: Unity and Visual Studio Code.

Intercase Dependencies: #2

Test Procedures:

- Complete a full day of selling items.
- On the day-end screen, observe the financial advice displayed.
- Compare the advice with performance data.

Input Specification: Player's performance metrics (speed of sales, number of failed sales, etc.)

Output Specifications: Textual advice given on performance and strategies for improvement

Pass/Fail Criteria:

Pass

- Advice corresponds logically with player's performance.
- Advice is constructive and relevant.
- The advice text is visible and grammatically correct.

Fail

- Advice is missing or irrelevant.
- Incorrect analysis of performance.
- Text is broken or unclear.

3 Test Results

0 - Star Screen and Buttons

Date(s) of Execution: Feb. 3- Feb. 12

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- Game starts correctly after clicking "Play"
- Timer bar begins at 10
- Player movement and control are immediately available

Actual Results:

- Game transitioned to gameplay screen immediately
- Timer bar began counting from 10
- Player was able to interact and move as expected

Test Status: Pass

1- Crafting System

Date(s) of Execution: Feb 17 2025- Feb 26 2025

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- Player is able to craft an item when resources are available
- Crafted item appears in inventory
- Resources used in crafting are properly deducted
- Feedback (e.g., sound or message) confirms success

Actual Results:

- Player successfully crafted selected item
- Crafted item appeared in inventory
- Resources were deducted correctly
- Audio and message feedback confirmed successful crafting

Test Status: Pass

2 - Day-End Sales Report

Date(s) of Execution: Mar. 3 2025 - Mar. 14 2025

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- End-of-day screen appears when time expires
- All sales (successful and failed) are shown
- Earnings and star rating reflect player's performance

Actual Results:

- End-of-day summary displayed correctly after timer ended
- All sales data was present and accurate
- Star rating and total earnings matched performance

Test Status: Pass

<u>3 - Financial Advice</u>

Date(s) of Execution: Mar. 20 2025 - Mar. 28 2025

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- Advice is shown at the end of the day
- Feedback is relevant to player performance
- Text is grammatically correct and visible

Actual Results:

- Financial advice appeared after the day ended
- Advice accurately reflected low sales performance and suggested better stock choices
- Text was clear, readable, and grammatically correct

Test Status: Pass

4 Regression Testing

N/A

IV Inspection

1 Items to be Inspected

Each group member reviewed every piece of code pushed to ensure it functioned correctly across all of our devices. Additionally, we wanted to ensure that all of the newly implemented functions worked without disturbing the other features that were previously implemented in previous versions.

2 Inspection Procedures

To manage our development process for Squeeze, our team used a custom checklist created in Jira to track tasks, features, and bug fixes. This checklist was collaboratively developed by team members and is included in the appendix for reference. We held weekly meetings that typically lasted about an hour. While most of the actual game development took place outside of these meetings, our sessions were primarily used for brainstorming and refining ideas to improve the game. Progress updates and discussions of results were conducted mostly through online platforms, which allowed us to stay aligned between meetings and efficiently manage our workflow.

3 Inspection Results

Throughout the development of Squeeze, inspections were regularly conducted to ensure the stability and functionality of key components, particularly the UI layout developed in Unity and the associated C# scripts. All team members were responsible for inspecting implemented features. The developer of a given feature would stress test it initially, while the rest of the team would perform follow-up checks to validate its performance and integration. These inspections occurred every other day during the development process. Common issues discovered included the UI not updating correctly, scripts modifying incorrect values, and unexpected runtime errors in the code. To resolve these problems, the team collaborated closely, sharing ideas and debugging together to identify root causes and implement effective fixes. After adjustments were made, components were re-inspected to verify that the issues had been successfully resolved. This cycle of inspection and re-inspection was carried out collectively by the entire team to maintain high quality throughout the project.

V Recommendations and Conclusions

All inspected components of Squeeze, including the UI layout and C# scripts, successfully passed their testing and inspection processes. After resolving the identified issues through collaborative debugging and re-inspection, no major flaws remained. As a result, no further testing or inspection is currently required. The next steps for the team involve finalizing the implementation and preparing the game for deployment and user feedback, ensuring a smooth transition from development to release.

VI Project Issues

1 Open Issues

Some open-ended issues that happened while working on SQUEEZE was that we were not able to add as much financial statistics as we wanted too because it was created using Unity in 2D rather than a website using other tools. Despite this we still added multiple ways for the user to learn about managing a business. Due to the scope of the project we focused on not handling personal accounts or not adding a database to keep track of players data.

2 Waiting Room

If we were given the opportunity to work on this project more we would have derived away from the singleton pattern and take another approach. We would have also added a way to store information within a database so the user can come back and play the game. We would have also included many other financial information to help the user be guided in the game and how to improve. We would be able to include multiple other types of graphs and a record log that shows every day's transactions.

3 Ideas for Solutions

Some solutions to implement some of the features we wanted would be by getting more time to do them. Due to time constraints we weren't able to implement some features we wanted. Using a database to be able to save users data and let them continue playing the game moving forward. Additionally, adding more products into the game to make it more fun and informative.

4 Project Retrospective

What worked well:

- Pair Programming
- Everyone in group contributed
- Version control
- All deadlines were met
- Adapting to 2D unity despite it being the first time for most
- Weekly meetings

What didn't work well:

- Saving users data
- Replacing financial core features in game
- Scheduling when to work on the game

VII Glossary

Probability: The game system will take into consideration every interaction between the player and customers and business endeavors to change the probability of success of the player's business. This probability controls how many customers come in. If the customers agree with the pricing, the probability of successful sales increases. This would be built using a set of mathematical rules that attempt to model real-world economics, integrate players' business and finance decisions, and affect sales in a dynamic and econometric way.

Customers: Characters in the game, the ones who approach the lemonade stand during the day to order a product from the lemonade stand. They are randomized characters.

Products: This is what is sold at the player's lemonade stand. We have implemented a handful of products, such as Lemonade (and variations of Lemonade), berries (such as strawberries, raspberries, and grapes), and Tea.

Ingredients: These are things that make products in the game. Some ingredients can be sold as products as is, but ingredients like sugar and lemons cannot be sold and can only be made into lemonade.

Crafting: This is the system the player will use to create products out of ingredients. Players must buy a sufficient amount of ingredients in order to craft products.

Pricing: The sale price of products can be changed by the player so they can aim to make more profit per sale. Prices that differ from the default can affect probabilities such as the success rate of sales.

Timer: Customers each have a timer that will start when their order is taken and end within a set amount of time if they are not served their order before then. If the timer runs out before the customer is served, the rating will decrease. Serving a customer successfully before the timer ends will increase the rating.

Rating: This 5-star based system is affected by the success of selling products to customers and serving customers before the timer ends. This system affects the customer amount in a day, which could result in a loss state if the rating reaches 0.

Day: When the game starts, it is day 1, and a day ends when there are no customers left. The player must start the next day through the pricing screen. Every new day that starts will add a bonus customer to the daily amount. This bonus can be dynamically affected throughout the day depending on the rating. Some days unlock new ingredients and products.

Graphs: When the day ends, the customer can review their financial statistics through graphs that display each transaction in a day and daily balances. This allows the player to assess their financial situation and determine if their pricing is profitable enough.

Feedback: The player will receive feedback on their financial and gameplay performance, which will let them know about what areas they need to improve on and what they can change, such as pricing.

Graphical User Interface: The interface where the player may serve customers, buy and sell products, change prices, and view financial graphs. This is the graphical element of the game.

Unity: Game engine that was used to create the game.

Pair programming: Agile software development technique where two people work together to design and test code.

2D game: A game that is created using only UI and 2D meshes and sprites.

Sprites: Character models in 2D usually made with images.

Version control: Software development practice that helps to track and manage code.

Database: A collection of data.

Jira: Project management tool used for collaboration and bug tracking.

C#: Programming language that was used for scripting.

Singleton pattern: Design pattern where one only one instance of the class is created.

VIII References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] Group 16, “SQUEEZE! Final Development Report”, Spring 2025

IX Index