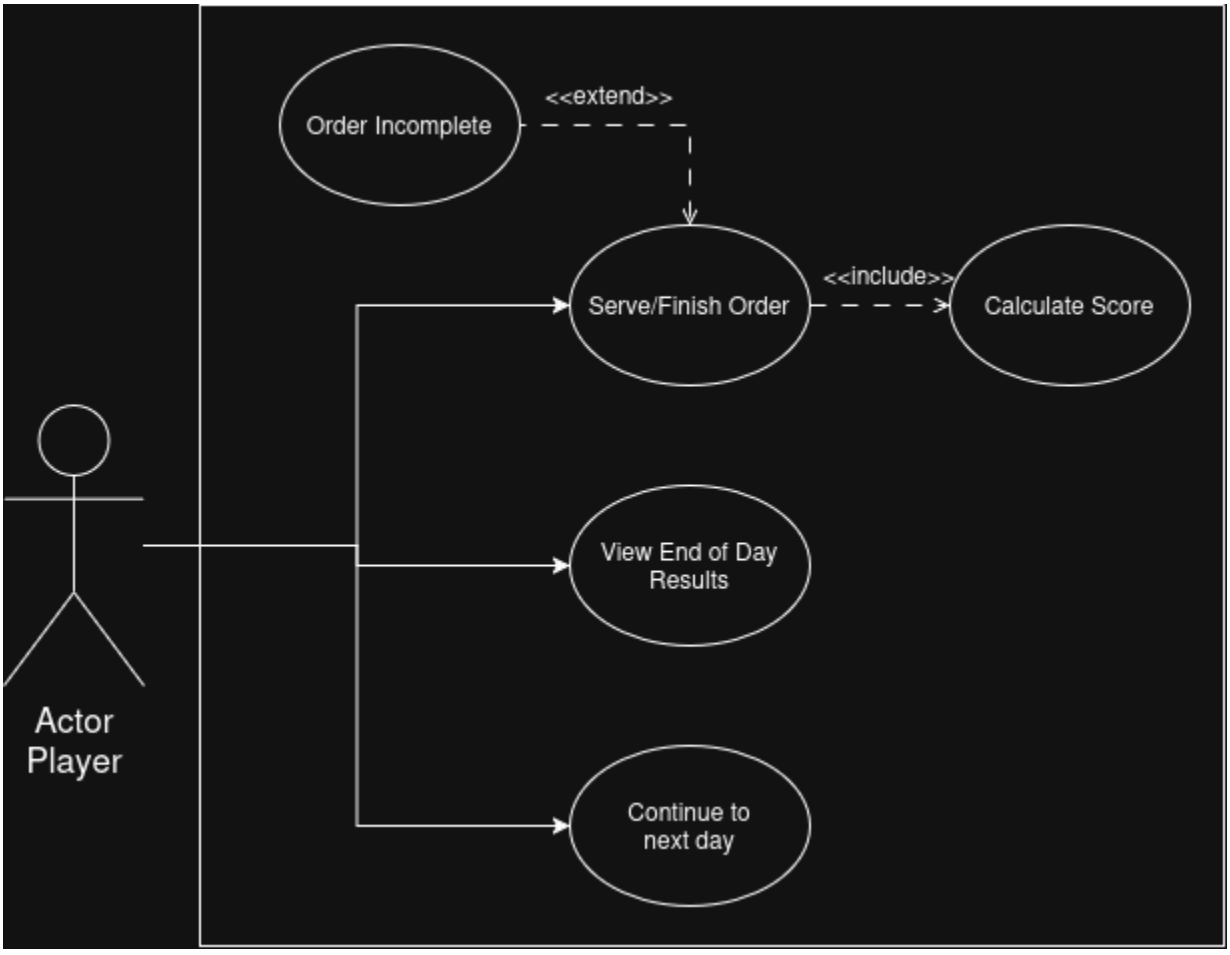
Name: Noah Turner Mark _____/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

1. Brief introduction __/3

ET-Pizzeria is a 2D mobile time-management game where the player completes pizza orders through multiple steps (ordering → toppings → baking → cutting → finish). This feature covers the final "delivery" portion of the gameplay loop: boxing/serving an order, validating that the pizza meets the customer request, calculating a score/tip, updating day totals, and displaying an end-of-day results screen. The feature also includes the scene transitions between the final gameplay step and the results screen.

2. Use case diagram with scenario __14



Scenarios

Name: Serve/Finish Order

Summary: The player serves the completed pizza order; the system validates completion, calculates score/tip, updates day totals, and transitions to results.

Actors: Player

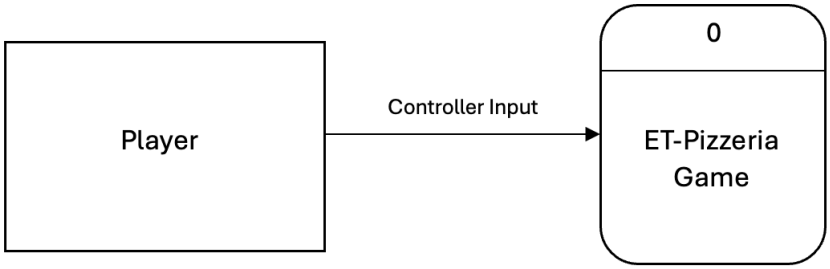
Preconditions: An order exists and is currently active; player is on the Finish/Serve screen; pizza state from previous steps is available (toppings, bake state, cut state).
Basic sequence:

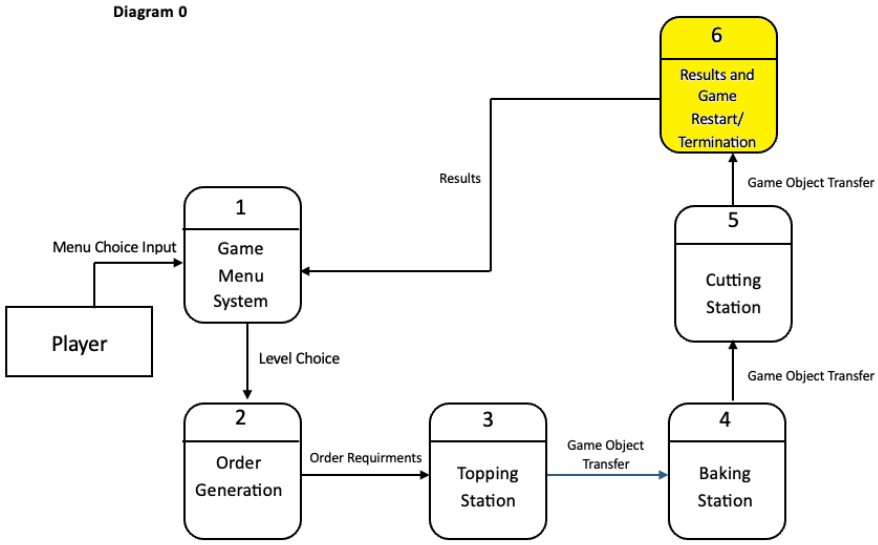
1. Player taps Serve/Finish.
2. System verifies required completion flags are present (order exists, pizza created, bake state recorded, cut state recorded, order is boxed/ready).
3. System calls [Calculate Score] (<<include>> UC-I01).
4. System updates day totals (money earned, accuracy %, tips, number of orders served).
5. System displays Serve Summary feedback (e.g., accuracy %, tip amount).
6. System transitions to End-of-Day Results when day/order count is complete (or transitions back to the next order if day is not complete).

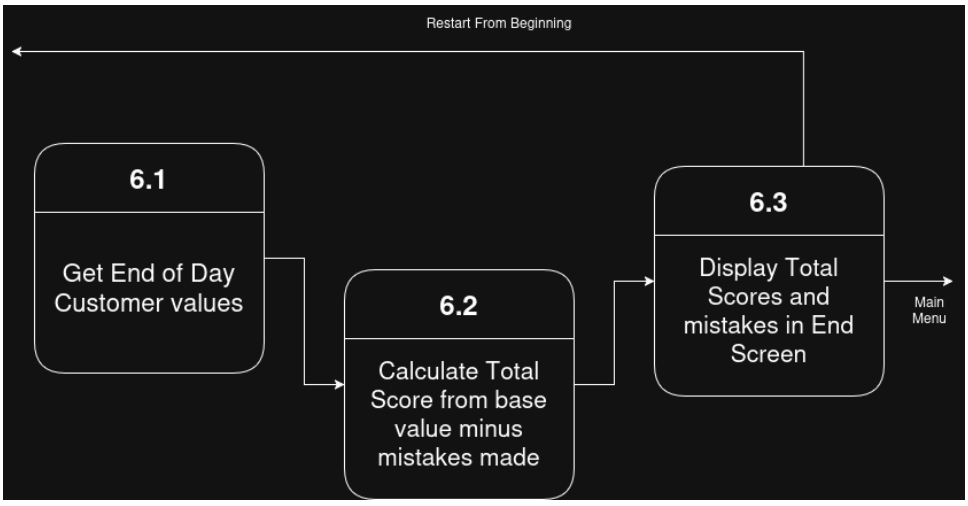
Exceptions:
Step 2 Exception: Serve tapped but pizza is missing a required state (not baked / not cut / not boxed / no order loaded).
System blocks serve, displays message like “Order not ready—complete all steps,” and remains on the Finish screen.
Step 3 Exception: Score calculation fails due to missing comparison data (order requirements not loaded).
System returns score = 0, logs error (debug), and displays minimal feedback.
Postconditions: Day totals reflect the served order; results/next-state screen is shown; current order is marked complete.
Priority: 1 (must have)
ID: F01

3. Data Flow diagram(s) from Level 0 to process description for your feature _____14

Data Flow Diagrams







Process Descriptions

// ---------- 6.1 Get End of Day Customer values ---------- function GetEndOfDayValues(): // pull whatever your other scenes saved (order results / mistakes) baseTotal = sum(allOrders.basePay) mToppings = sum(allOrders.mistakesToppings) mBake = sum(allOrders.mistakesBake) mCut = sum(allOrders.mistakesCut) mFinish = sum(allOrders.mistakesFinish) return { baseTotal, mToppings, mBake, mCut, mFinish }

// ---------- 6.2 Calculate Total Score from base value minus mistakes made ---------- function CalculateTotalScore(values): penalty = 10*values.mToppings + 15*values.mBake + 10*values.mCut + 10*values.mFinish totalScore = max(0, values.baseTotal - penalty) return { totalScore, penalty }

// ---------- 6.3 Display Total Scores and mistakes in End Screen ---------- function ShowEndScreen(): values = GetEndOfDayValues() // 6.1 calc = CalculateTotalScore(values) // 6.2

UI.SetText("Base Total", values.baseTotal)
UI.SetText("Topping Mistakes", values.mToppings)
UI.SetText("Baking Mistakes", values.mBake)
UI.SetText("Cutting Mistakes", values.mCut)
UI.SetText("Finish Mistakes", values.mFinish)
UI.SetText("Penalty", calc.penalty)
UI.SetText("Final Score", calc.totalScore)

// button outputs shown in your diagram
UI.OnClick("Main Menu", LoadMainMenu)
UI.OnClick("Restart From Beginning", RestartGame)

4. Acceptance Tests _____9

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

Example for random number generator feature

Run feature 1000 times sending output to a file.

The output file will have the following characteristics:

- Max number: 9
- Min number: 0
- Each digit between 0 and 9 appears at least 50 times
- No digit between 0 and 9 appears more than 300 times
- Consider each set of 10 consecutive outputs as a substring of the entire output. No substring may appear more than 3 times.

Example for divide feature

Output	Numerator (int)	Denominator (int)	Notes
0.5	1	2	
0.5	2	3	We only have 1 bit precision for outputs. Round all values to the nearest .5
0.0	1	4	At the 0.25 mark always round to the nearest whole integer
1.0	3	4	At the 0.75 mark always round to the nearest whole integer
255.5	5	0	On divide by 0, do not flag an error. Simply return our MAX_VAL which is 255.5.

5. Timeline _____/10

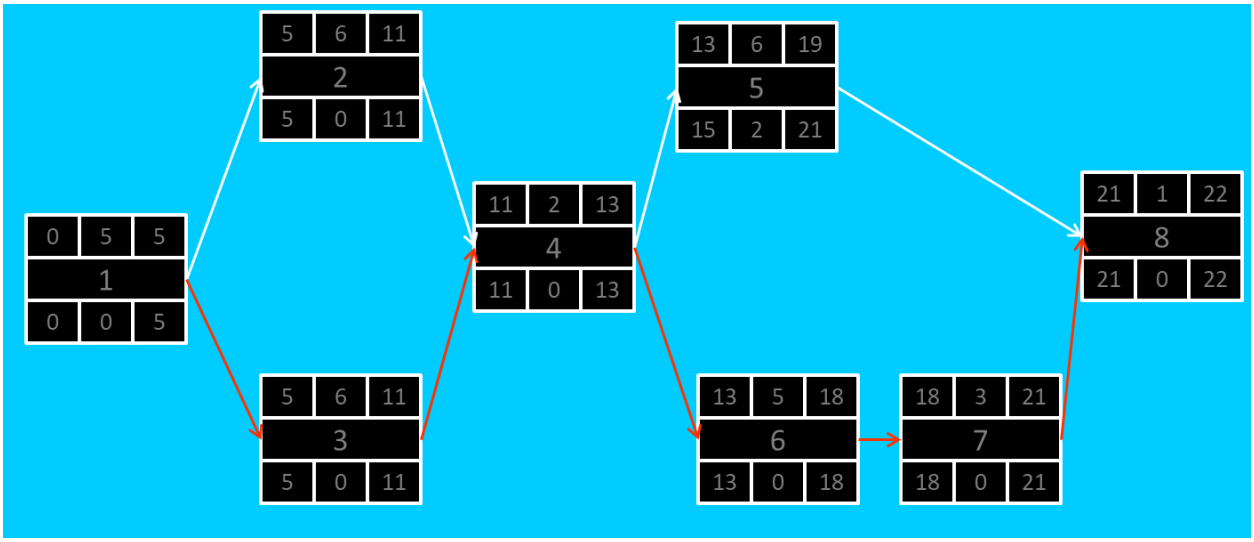
[Figure out the tasks required to complete your feature]

Example:

Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Requirements Collection	1	-
2. Finishing screen	1	1
3. Animation Design	2	2
4. score calculation	1	2, 3
5. Animation Integration	1	3, 4
6. Programming	2	5
7. Testing	1	6
8. development	1	7

Pert diagram



Gantt timeline