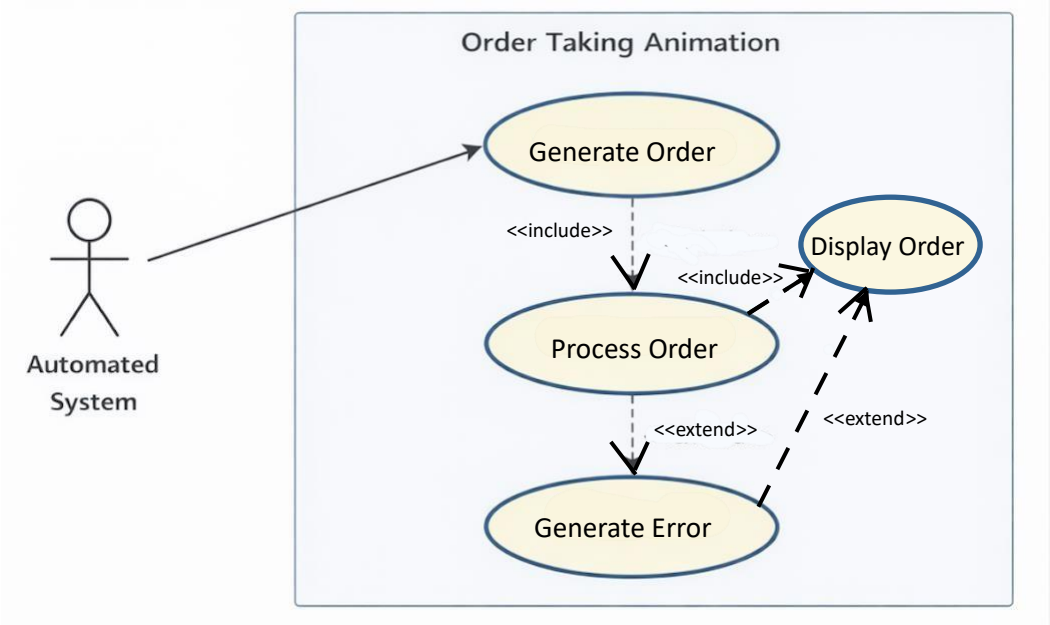
Name: Zuzanna Whitman Mark _____/50

1. Brief introduction __/3

This feature consists of an automated order-taking animation. Orders are randomly generated by the system and displayed sequentially through an animation that simulates an active ordering process. There is no user interaction involved; the feature is intended purely as a visual demonstration of system behavior and order flow.

2. Use case diagram with scenario __14

Use Case Diagrams



Scenarios

Name: Order Taking Animation

Summary: The system automatically generates and displays orders through an animation.

Actors: System (Game/Application Engine).

Preconditions: Game is running

Basic sequence:

Step 1: The system starts order animation.

Step 2: The system generates a random order.

Step 3: The system displays the order through animation.

Step 4: The system waits for a predefined time interval.

Step 5: The system clears the order and generates the next one after the previous one has been finished.

Exceptions:

Step 1: Random order generation fails.

Step 2: The system generates a default order instead.

Post conditions: An order has been displayed and cleared. • System is ready to generate the next order.

Priority: 1*

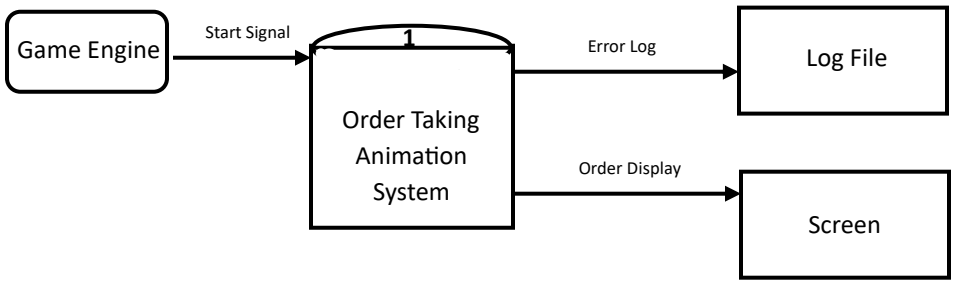
ID: OTA1

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

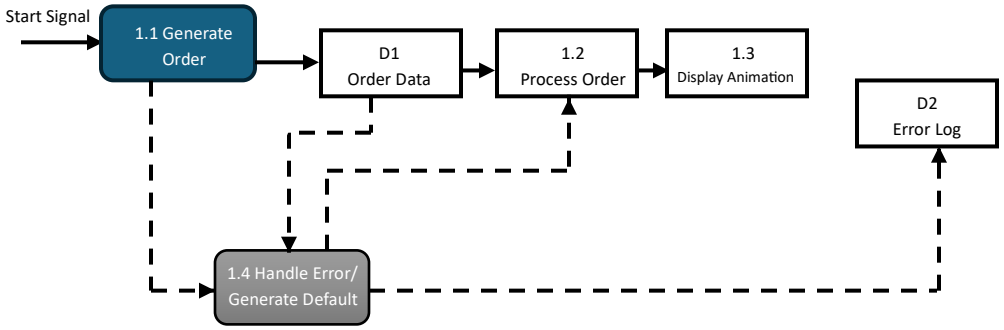
3. Data Flow diagram(s) from Level 0 to process description for your feature _____14

Data Flow Diagrams

Level 0



Level 1



The Level 0 Data Flow Diagram represents the Order Taking Animation System as a single high-level process that interacts with external entities. The System Clock / Game Engine provides the start signal that initiates the animation cycle. The system produces two possible outputs: visual order display data sent to the Screen and error data written to the Log File. This diagram provides a contextual overview of the system without showing internal processes.

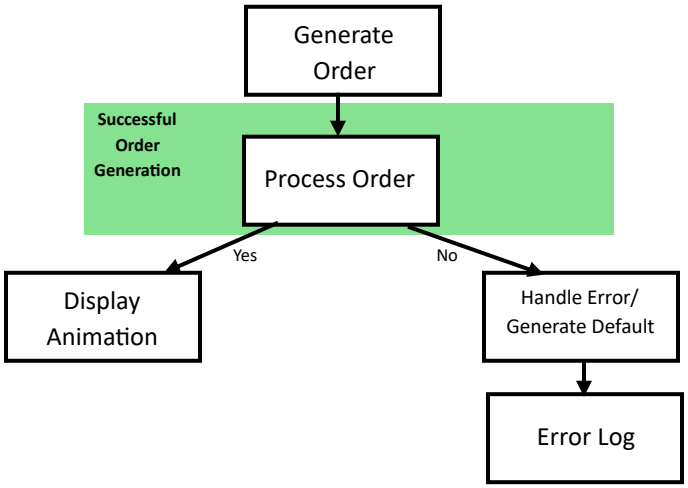
The Level 1 Data Flow Diagram decomposes Process 1 into four internal processes: Generate Order (1.1), Process Order (1.2), Display Order (1.3), and Handle Error (1.4). The Generate Order process creates a randomized order

and stores it temporarily in D1 (Order Data). The Process Order step formats and validates the order before passing it to the Display Order process, which visually renders the order on the Screen.

If an error occurs during order generation, control is transferred to the Handle Error process. This process logs the failure in D2 (Error Log) and generates a predefined default order, which is then sent back to the Process Order stage to resume normal system execution.

The decomposition from Level 0 to Level 1 ensures that the system's behavior is represented clearly and hierarchically, while maintaining proper data flow structure and separation of processes, data stores, and external entities.

Process Descriptions



4. Acceptance Tests _____9

This feature contains both deterministic elements (animation timing, order structure, process sequence) and random elements (order content generation). Therefore, acceptance testing will focus on validating bounded randomness, structural correctness, process sequencing, timing constraints, and system stability under extended execution. The acceptance tests described below validate the feature at three levels: Order Generator, Animation Processor, and Full System Integration.

Order Generator Testing

The Order Generator will be executed automatically 200 times to test the limits of randomness and structural validity. Each order generated will be written to an output file containing:

- Cycle number
- Order ID
- Order contents
- Generation timestamp
- Generation status (Success/Failure)

The output file will be analyzed for the following measurable conditions:

Random Distribution Constraints:

- At least 65% of orders generated must be unique within 200 cycles.
- No individual order configuration may appear, more than 25% of the total executions.
- If order categories exist (e.g., single item, combo, special), each category must appear at least once within 50 consecutive cycles.

Structural Validation:

- Each order must contain all required fields.
- No null or undefined values are permitted.
- Order formatting must conform exactly to the predefined data structure.
- Generated data must remain within predefined logical bounds (e.g., valid item names only).

Failure Handling Validation:

The order generator will be intentionally forced to fail (simulated exception or null return). When failure occurs:

- The system must generate a predefined default order.
- The failure must be recorded in the log file.
- The system must continue execution without terminating.
- The default order must only appear when a generation failure occurs.

Animation Processor Testing

The animation process will be tested for correct sequencing, timing consistency, and visual clearing behavior. The animation loop will run for 100 continuous cycles. Timestamps will be recorded for:

- Animation start time
- Animation end time
- Next cycle start time

The following measurable constraints must be satisfied:

Timing Constraints:

- Each order must be displayed for the predefined duration (±0.5 seconds tolerance).
- The delay between clearing one order and generating the next must not exceed 1 second.
- No order may remain visible beyond its assigned duration.

Sequencing Constraints:

The process must strictly follow this sequence:
Generate Order → Process Order → Display Order → Clear Order → Generate Next Order
If the process deviates from this sequence at any point, the test is considered a failure.

Rendering Failure Simulation:
If animation rendering fails:

- The system must log the error event.
- The animation cycle must automatically restart.
- No system crash, freeze, or infinite loop may occur.

Integration and Stability Testing
The complete feature (Order Generator + Animation Processor combined) will be executed for 500 total cycles to test long-term system behavior.

The following conditions must be verified:

- No runtime exceptions occur.
- Memory usage remains stable (no accumulation of undeleted objects).
- CPU usage remains within acceptable operational bounds.
- The system runs continuously without user input.
- Orders continue generating sequentially without interruption.

Any crash, freeze, unhandled exception, or improper termination constitutes a failure of the acceptance test.

Example Output Test Analysis for Success/Failure

Cycle	Order ID	Unique	Display Time	Generation Success	Animation Success	Default Used	Success	Fail
1	#102	Yes	3.01s	Yes	Yes	No	Yes	No
48	#102	No	3.00s	Yes	Yes	No	Yes	No
89	Default	—	3.02s	No	Yes	Yes	Yes	No
212	#310	Yes	4.20s	Yes	No	No	No	Yes

Final Acceptance Conditions
The feature will be accepted if:

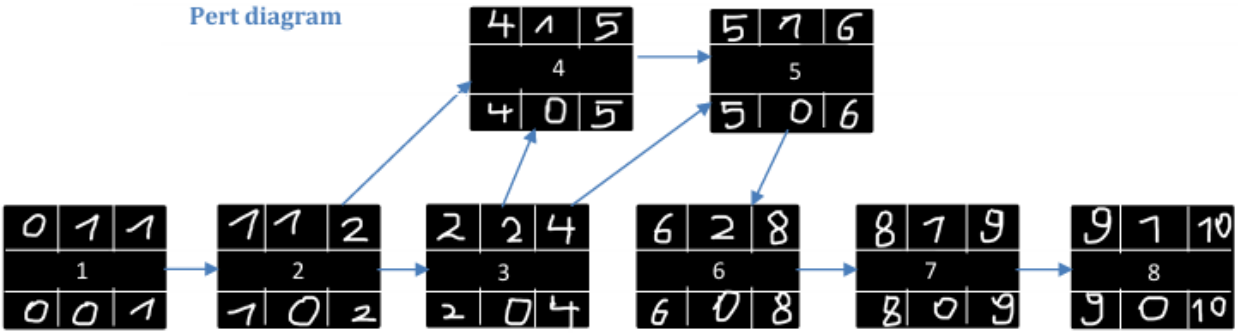
- Random order generation remains within defined statistical limits.
- Orders are structurally valid in 100% of executions.
- Animation timing remains within tolerance bounds.
- The system recovers from simulated failures.
- No crashes or major interruptions occur during extended execution.
- The feature operates fully autonomously without user interaction.

5. Timeline _____/10

Work items

Task	Duration (PWks)	Predecessor Task(s)
1. Requirements Collection	1	-
2. Order Logic Design	1	1
3. Animation Design	2	2
4. Random Order Generation	1	2, 3
5. Animation Integration	1	3, 4
6. Programming	2	5
7. Testing	1	6
8. Deployment	1	7

Pert diagram



Gantt timeline

Key:

Work hours