

Name Emily\_Thomas

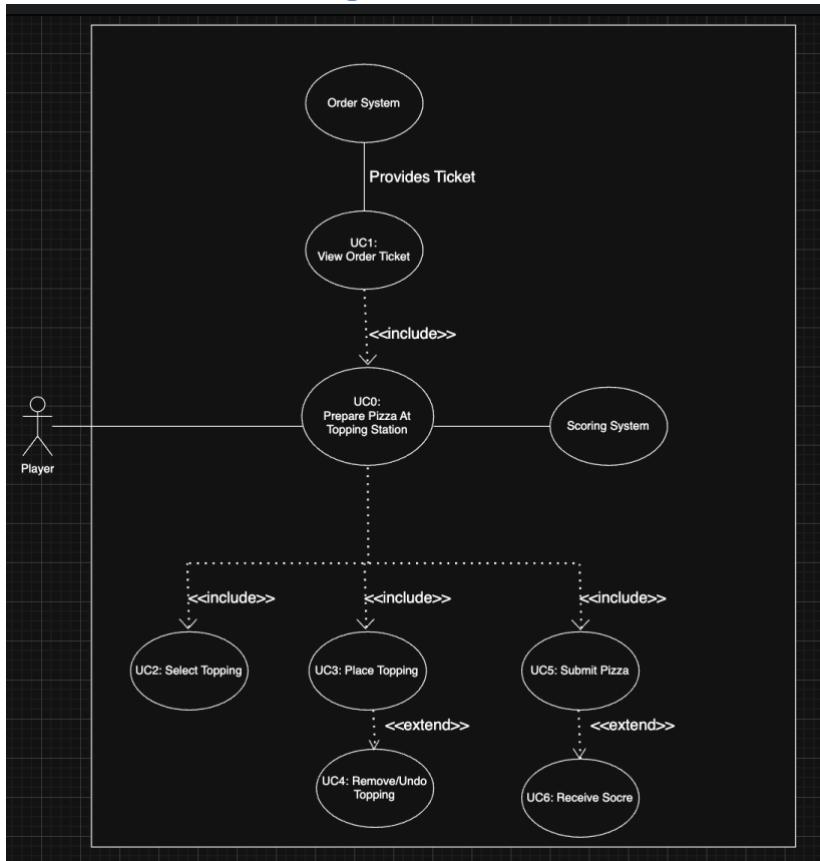
Mark \_\_\_\_\_/50

## Brief introduction \_\_/3

The Topping Station will its own scene in our game where the player will put the ordered toppings on base pizza. The Topping Station will be the scene right after the player takes the order. The player will be given a base pizza, which will be a crust with sauce and cheese, as soon as they start making the order. The order will have a range of toppings that can be chosen and a percentage of the pizza they will be on. After the player has deemed the pizza toppings to be correct to the order, they will click a button to go to the Baking Station.

## Use case diagram with scenario \_\_14

Use Case Diagram



### Scenario

**Name:** Prepare Pizza at Topping Station

**Summary:** The player will interact with the screen to drag toppings on and off of the pizza then submit it to receive a score.

**Actors:** Player

Supporting Actors: Order System, Scoring System

**Preconditions:** Order has been placed and scoring system is in place.

**Basic sequence:**

1. The use case begins when the Player enters the Topping Station.
2. The system automatically includes View Order Ticket to display the customer's topping requirements.
3. The Player reviews the order ticket.
4. The system includes Select Topping as the Player chooses a topping from the topping menu.
5. The system includes Place Topping as the Player places the selected topping onto the pizza.
6. Steps 4–5 repeat until the Player believes the pizza matches the order.
7. The system includes Submit Pizza when the Player sends the pizza to the oven.
8. The Receive Score use case is triggered as an extension, and the Scoring System evaluates topping accuracy.
9. The use case ends. Exceptions:

#### **E1 – Incorrect Topping Placement**

- The Player places a topping in the wrong location.
- The system accepts the placement.
- The scoring system applies a penalty during evaluation.

#### **E2 – Excess Toppings Added**

- The Player adds more toppings than required.
- The system accepts the placement.
- The scoring system applies a penalty during evaluation.

#### **E3 – Missing Required Toppings**

- The Player submits the pizza without all required toppings.
- The system accepts the submission.
- The scoring system applies a penalty during evaluation.

#### **E4 – Undo/Remove Topping**

- The Player chooses to undo or remove a topping.
- The system removes the selected topping and updates the pizza.

#### **E5 – Order Ticket Updated Mid-Process**

- The order system updates the customer's order.
- The system refreshes the order ticket.
- The Player continues topping placement using the updated requirements.

**Post conditions:** Player receives topping score

**Priority:** 1

Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_ 14

### Data Flow Diagrams

#### Context Diagram

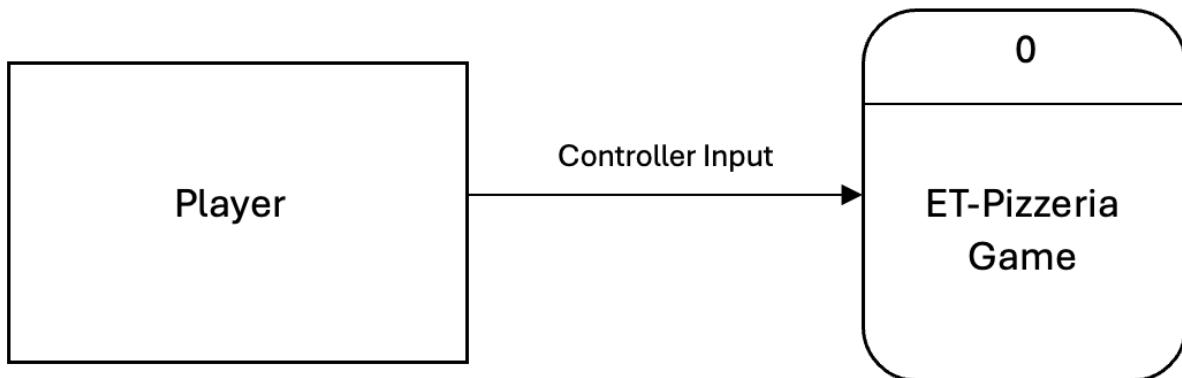


Diagram 0

Diagram 0

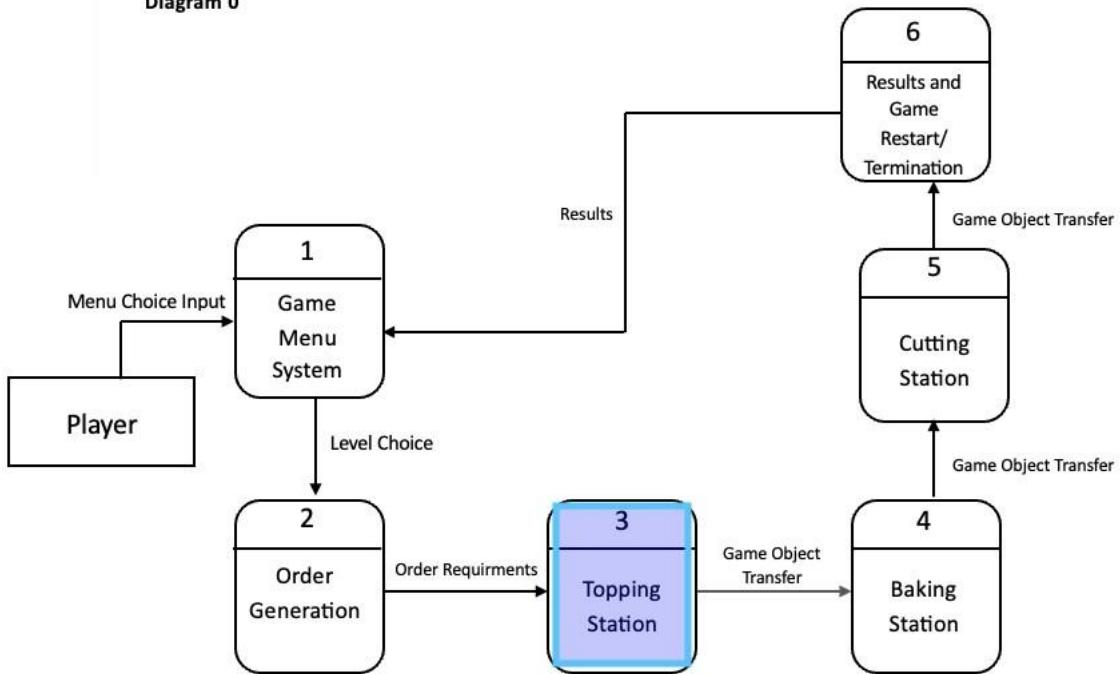
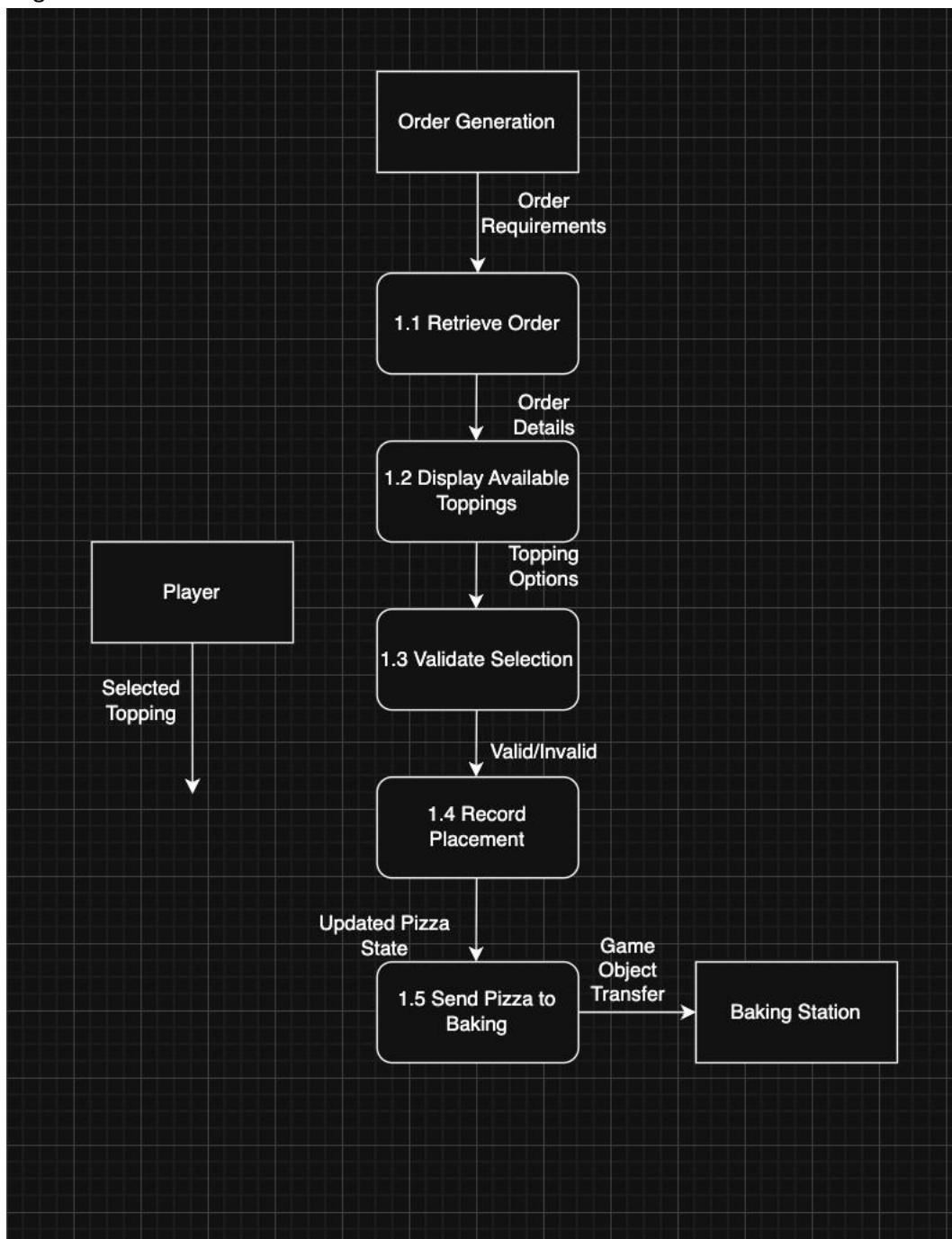


Diagram One



### Process Descriptions Inputs

#### Player Input (from Player)

Actions such as selecting toppings, placing toppings, undoing actions, and submitting the pizza.

#### Order Information (from Order Data)

The required toppings and configuration for the current pizza order.

#### Pizza Data (from Pizza Data Store)

Current pizza configuration, topping placements, and any previously stored pizza state.

#### Score Information (from Score Data Store)

Scoring rules, penalty values, and evaluation criteria.

### ***Outputs***

#### Game Output (to Player)

Updated visuals, feedback, topping placement results, and final score display.

#### Updated Pizza Data (to Pizza Data Store)

Revised pizza configuration after each topping placement or removal.

#### Updated Game State (to Game State Data Store)

Current progress, order status, and gameplay variables.

#### 0.1 Process Player Input

- Reads player actions (select topping, place topping, undo, submit).
- Sends topping placement requests to the pizza update logic.
- Sends submission requests to scoring logic.

#### 0.2 Manage Order Requirements

- Retrieves order information from Order Data.
- Displays order ticket to the player.
- Ensures the system always uses the correct order requirements.

#### 0.3 Update Pizza Configuration

- Adds or removes toppings based on player actions.
- Validates topping placement format (location, count).
- Stores updated pizza configuration in Pizza Data.

#### 0.4 Update Game State

- Tracks progress toward completing the order.
- Updates internal variables such as time, steps taken, or topping count.
- Stores updated state in Game State Data.

#### 0.5 Evaluate Pizza and Score

- Compares final pizza to order requirements.
- Applies penalties for missing, incorrect, or excess toppings.
- Stores scoring results in Score Data.
- Sends final score to the player.

## Acceptance Tests \_\_\_\_\_ 9

### Test Group 1 — Topping Selection and Placement

#### **Test 1.1 — Valid Topping Placement**

Input:

Player selects a topping listed on the order ticket

Player places topping on a valid pizza zone

Expected Output:

Topping appears on the pizza

Pizza Data updates with new topping

No penalties applied

#### **Test 1.2 — Incorrect Topping Placement (Boundary Case)**

Input:

Player selects a correct topping

Player places it in the wrong location (e.g., outside the intended region)

Expected Output:

Topping still appears on the pizza

System records placement

Scoring System applies penalty during evaluation **Test**

#### **1.3 — Excess Toppings Added (Boundary Case)**

Input:

Player adds more toppings than required (e.g., 5 pepperonis when order requires 3)

Expected Output:

All toppings appear on the pizza

Pizza Data stores all placements

Scoring System applies penalty for excess

#### **Test 1.4 — Undo/Remove Topping Input:**

Player selects “Undo” or “Remove” Player

chooses a topping to remove

Expected Output:

Selected topping is removed

Pizza Data updates No

penalty applied

### **Test Group 2 — Order Ticket Handling Test**

#### **2.1 — Order Ticket Displays Correctly**

Input:

New order arrives from Order Data

Expected Output:

Order ticket appears with correct toppings and quantities **Test**

#### **2.2 — Order Ticket Updates Mid-Process (Boundary Case)**

Input:

Order Data sends an updated order while player is placing toppings

Expected Output:

Order ticket refreshes

Player sees updated requirements

Game State updates accordingly

### **Test Group 3 — Pizza Submission and Scoring**

#### **Test 3.1 — Correct Pizza Submitted**

Input:

Player submits pizza with all required toppings correctly placed

Expected Output:

System evaluates pizza

Score reflects full accuracy

Game Output displays success

**Test 3.2 — Missing Required Toppings (Boundary Case)** Input:

Player submits pizza missing one or more required toppings

Expected Output:

System evaluates pizza

Penalty applied for missing toppings

Score reflects deductions

**Test 3.3 — All Toppings Incorrect (Extreme Boundary Case)**

Input:

Player submits pizza with no correct toppings

Expected Output:

System evaluates pizza

Maximum penalty applied

Score reflects failure

**Test 3.4 — Submit With No Toppings (Edge Case)**

Input:

Player submits an empty pizza

Expected Output:

System evaluates pizza

Score = minimum possible

Game Output displays failure

**Test Group 4 — Game State and Data Storage Test**

**4.1 — Pizza Data Updates After Each Action**

Input:

Player places or removes toppings

Expected Output:

Pizza Data store updates after each action

Game State reflects current progress **Test**

#### **4.2 — Score Data Accessed Correctly**

	Task	Duration (PWks)	Predecessor Task(s)
T1	Define feature requirements	2 Hours	—
T2	Create use case diagram	1 Hour	T1
T3	Write use case scenario	1 Hour	T2
T4	Build Data Flow Diagram (DFD 0)	1 Hour	T1

T5	Create Level-1 DFD	1 Hour	T4
T6	Write process descriptions	1 Hour	T4
T7	Design topping placement logic	5 Hours	T1
T8	Implement topping selection UI	5 Hours	T7
T9	Implement topping placement mechanics	10 Hours	T7
T10	Implement undo/remove topping	2 Hours	T9
T11	Implement order ticket display	1 Hour	T1
T12	Implement scoring logic	1 Hour	T9
T13	Integrate scoring with game output	1 Hour	T12
T14	Internal testing of topping station	5 Hours	T8, T9, T10, T11, T13
T15	Write acceptance tests	2 Hours	T3, T6
T16	Final polish & bug fixes	5 Hours	T14

Input: Player submits pizza

Expected Output:

Scoring System retrieves scoring rules

Score is calculated and stored

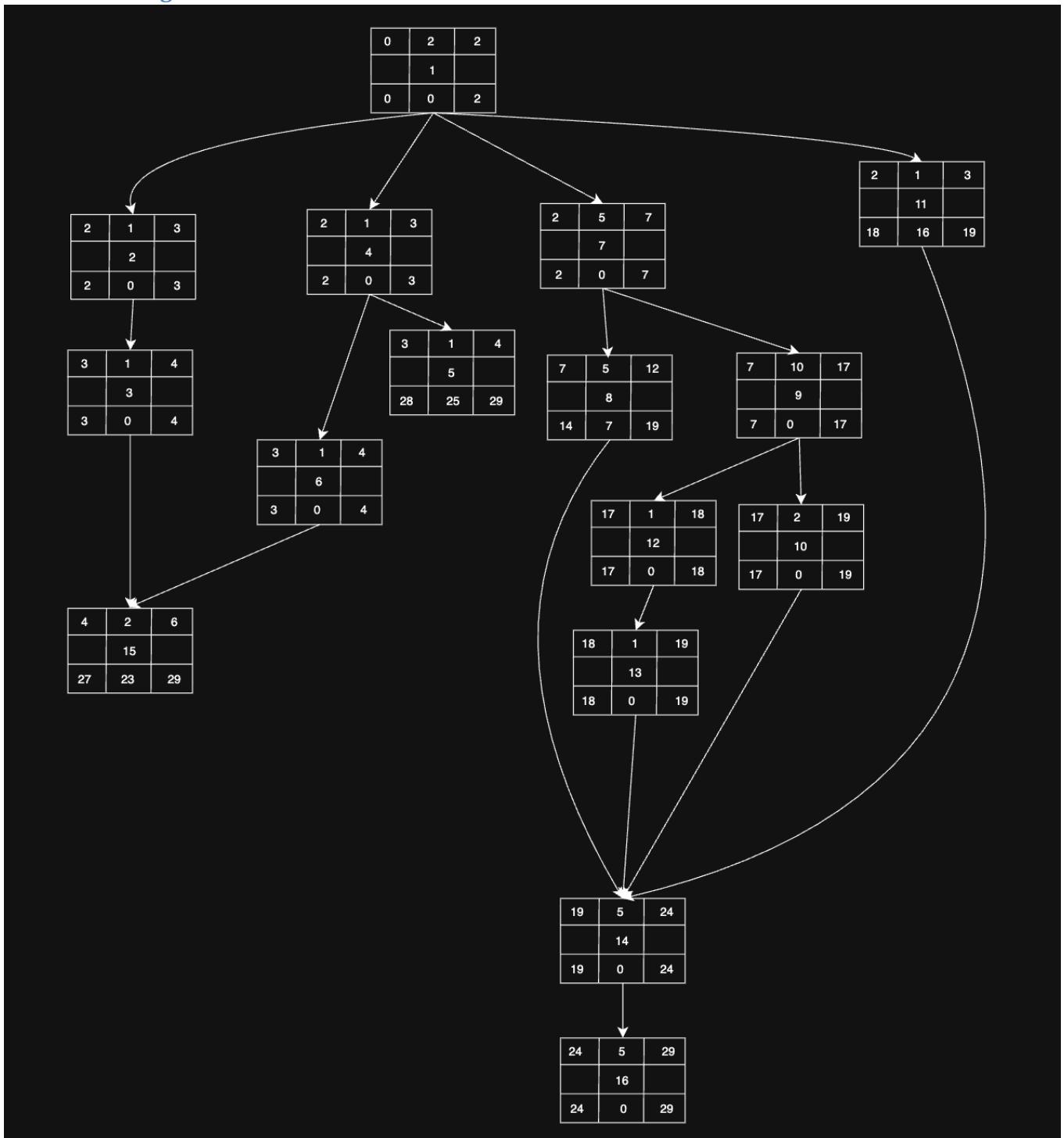
Game Output displays final score

**Timeline \_\_\_\_\_/10**

Example:

**Work items**

## Pert diagram



## Gantt timeline