The goal of the restaurant system simulation is to create an intelligent multi-agent system that manages the ordering, food preparation, and dining experience in a restaurant setting. The critical events include customers placing orders, the chef preparing the food, and the waiter serving the food to customers. The focus is on simulating the interactions and coordination between the customer agent, waiter agent, and chef agent to provide a seamless dining experience. The ability to set the food preparation time allows for experimentation and analysis related to the efficiency of food preparation processes. The project can be considered moderately challenging, as it involves implementing the agents' behaviors, communication protocols, and managing the simulation dynamics.

Development summary:

Jason version 2.0 will be used for this implementation.

The multi-agent system will be developed at the ASL level.

The BDI model will be used to program the agents' behaviors.

The entire program will be implemented as a multi-agent system.

The user can set the time taken for food preparation using an input field.

GUI plan:

1. Parameter Input Section:
   * Retain the parameter input section at the top or in a separate panel to allow users to set essential simulation parameters, including the food preparation time using an input field or a slider.
2. Visual Representation of Dining Area:
   * Simplify the visual representation of the dining area by using basic shapes to represent tables only (without customers).
   * Consider using numbered or labeled tables to make them distinguishable, but avoid adding additional visual elements for occupied or highlighted tables.
3. Order Queue Status:
   * Instead of a section displaying order cards, integrate the order queue status directly into the visual representation of the dining area.
   * Use a small icon or indicator next to each table to represent the number of pending orders for that table.
4. Message Log:
   * Include a compact and scrollable message log section that displays communication and important events within the system.
   * Opt for a simplified format that includes the sender, receiver, and a concise message/event description for each log entry.
5. Controls and Buttons:
   * Retain essential controls and buttons at the bottom of the GUI, such as buttons for starting, pausing, or resetting the simulation.
   * Consider removing the button for generating random events or triggering specific scenarios unless it is crucial for the assignment requirements.
6. Real-Time Updates:
   * Ensure that the GUI updates in real-time to reflect changes in the system, such as the status of tables (occupied or available) and the order queue.

Remember to keep the GUI clean and uncluttered, focusing on the essential elements described above. The goal is to provide valuable information and control to the user without overwhelming them with unnecessary details.

Intelligent restaurant system requires the implementation of three agents:

1. Customer Agent:
   * Responsible for placing orders and communicating with the waiter agent.
   * Uses BDI beliefs about menu items, desires, and preferences.
   * Sends messages to the waiter agent to place orders.
2. Waiter Agent:
   * Manages the dining area, takes orders, and communicates with the chef agent.
   * Uses BDI beliefs about customer orders, dining area status, and kitchen status.
   * Receives orders from the customer agent, communicates with the chef agent to prepare orders, and serves food to customers.
3. Chef Agent:
   * Manages the kitchen, prepares orders, and communicates with the waiter agent.
   * Uses BDI beliefs about the order queue, kitchen equipment, and ingredients.
   * Receives orders from the waiter agent, prepares the orders, and sends a message to the waiter agent when the order is ready.

These three agents together form the core components of the intelligent restaurant system. The customer agent initiates the conversation by placing an orderchef

and the chef agent handles the food preparation in the kitchen. Their interactions enable the system to handle customer orders, manage the kitchen activities, and provide a seamless dining experience.

To represent customers and food in the simulation with a minimal implementation, you can use simple data structures and attributes. Here's a suggestion for representing customers and food:

1. Food Items:
   * Food items can be represented using a list or an array of strings.
   * Each string can correspond to a specific menu item or dish.

By using these minimal attributes, you can track the status of customers (e.g., if they have placed an order, received their food) in the simulation.

Additionally incorporate a queue or list data structure to manage the order queue, where each entry represents an order placed by a customer. This queue can be updated as customers place orders and the chef agent prepares the food.