UML

Use Case: Place an order

Primary Actor: Registered user

Scope: Server/Database

Level: User Goal

Brief: A registered user places a new food order using the online form, which connects to and updates the database accordingly.

Stakeholders: Food pantry management, DBA

Post conditions: Food is received by user (leaves pantry). Database is updated.

Preconditions: User has an account. Food ordered has a quantity >0 in database system.

Triggers: User successfully fills in & submits form to server.

Basic Flow:

1. User log in.
2. Fill out required fields in form.
3. Press the submit button.
4. Database updated.
5. Pantry staff prepares food for pickup.
6. User picks up order.

Extensions:

Cancel – The user cancels their order. The database is updated accordingly (quantity+)

Timeout – The user's order will be automatically canceled is form takes X time to complete.

API

1. **<u>Log in process;</u>**

{

“action”: “log in”

“username”: “User1”

“password”: “***”

}

=>

{

“success”: “true”

“message”: “Welcome username!”

}

OR

{

“success”: “false”

“message”: “Incorrect username and/or password. Please try again.”

}

2. **Fill in Form (Place Order);**

{

“action”: “Fill required fields”

“field x (Item Name)”: “some data”

“field Y (Item Quantity)”: “some quantity”

}

=>

{

“success”: “true”

“action”: “Order is sent to database”

}

OR

{

“success”: “false”

“message”: “Invalid name/quantity of item.

}

3. **Order Sent to Database**

{

“action”: “successfully filled in form data is sent to database”

“field x (Item Name)”: “compared to database name field”

“field Y (Item Quantity)”: “compared to database quantity field”

}

=>

{

“success”: “true”

“Form action”: “Order placed!”

“Database action”: “item quantity updated (subtracted)”

}

OR

{

“success”: “false”

“message”: “Item not in stock.”

}