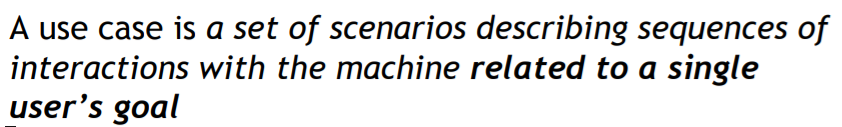
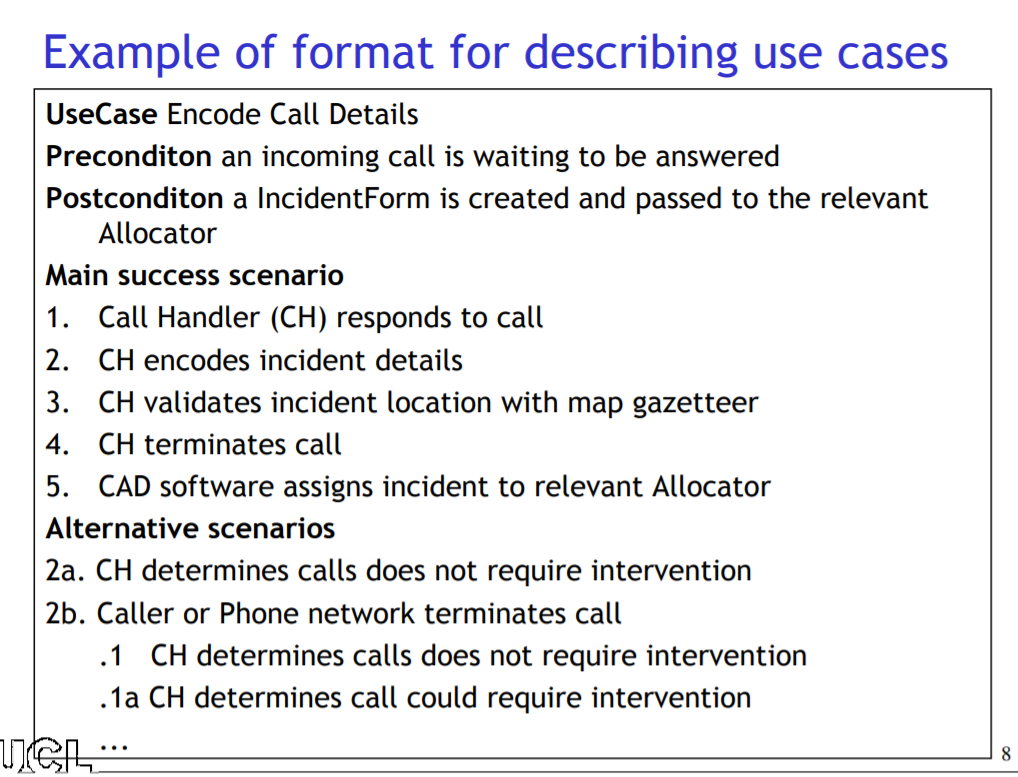
**References**





Customer Use Cases:

* Login / Register for Account
* Order Food
* Rate Cook, Rate Deliverer
  + affects Cook warning, then Status
  + affects Deliverer warning, then status
* Register to restaurant
* Search (for food type, restaurant)
* View Ratings

Manager Use Cases:

* Approve Customer Registration
* Hire Employee
* Start Delivery Bid, Choose Delivery
* Fire Staff
* Remove Staff Warning
* Pay staff
* Handle customer complaints

Cook Use Cases:

* View Order
* Finish Order
* Rate Salespeople
* Add menu items
* Request Supplies

Deliverer Use Cases:

* Bid on delivery
* Pickup order
* Determine Route
* Deliver Order
* Rate Customer

Salespeople Use cases:

* Order Supplies

**UC Template**

Name:

Actors:

*Optional: Group of Use-cases/General Methods*:

Pre:

Post:

Success:

Exception:

**UC 1**

Name: User Login/Signup

Actors: All Users, System

*Group of Use-cases/General Methods:* Login, Register for Account

Success:

1. User enters email and password
2. User successfully logs in
3. Based on what user type they are, they are redirected to their appropriate page

Alternative:

1a. User does not put in login credentials (leaves blank), becomes Guest Customer

User is asked if want to register for account (maybe a popup?)

if yes, send to signup page

User signs up for account (firstName, lastName, email, password, userType)

if no, send to List of Restaurants page

Exception:

1a. User puts invalid (wrong) login credentials.

1b. error msg of invalid credentials

**UC 2**

Name: Restaurant Registration

Actors: Customer, Manager

*Group of Use-cases/General Methods*: Subscribe, log in, search

Pre: not black listed

Post: Customer is registered into Restaurant Registration Database

Success:

1. Customer logs in
2. Customer searches for restaurant
3. Customer clicks to register
4. Manager approves of registration

Exception:

4a. Customer registration is not approved

4b. Customer is blacklisted and registration is automatically rejected

**UC 3**

Name: Process Food Order

Actors: Customer, Cook, Deliverer, Manager

*Group of Use-cases/General Methods:* Customer Order Food, Cook View Order, Manager Start/Choose Bid, Deliverer Bid/Pickup/Find Best Route/Deliver

Pre: Customer is registered in restaurant. (logged in, not black listed). Customer orders.

Post: Customer receives order

Success:

1. Customer orders from restaurant (payment is successful). Order status == requested/pending
2. Cook sees Order request. Cook prepares foods and submits once done (order status == prepared)
3. Manager starts a bid for the Deliverer
4. Deliverer bids the order
5. Manager chooses the Deliverer, lowest bid
6. Deliverer picks up order (order status == in transit), chooses the best route with findPath(), and delivers order successfully.
7. Order status == delivered/complete

Exception:

4a. No deliverer bids

6a. Delivery is unsuccessful

can’t get in contact with Customer.

delivery to wrong address/person

**UC 4**

Name: Customer Sends Order Request to Restaurant (Details and Processing)

Actors: Customer, System/Restaurant

Pre: Customer Logged in, at restaurant page (not blacklisted)

Post: Order is sent to restaurant to be processed

Success:

1. Customer views menu of restaurant,
2. Customer selects food and reads reviews, adds food to order (cart)

//continue ordering (loop back to view menu)

1. Customer clicks to view Checkout and Complete/submit order
2. Customer chooses payment method
3. Customer fills in payment form (with option to save)
4. Customer completes order, order is sent to system/restaurant

Alternative:

1a. i. Customer status, isRegistered == true.

Gets discounted price.

View menu differently by personal order history! food items previously ordered (assuming with highest frequency) are at top

ii. Customer status, isVIP == true.

Gets higher discount and free food item (random or chosen? maybe up to us to decide)

iii. Customer status, isRegistered == false. Views menu by top 3 most ordered

**UC 5** //must use tokens for petri-net for rating and Staff warnings!

Name: Customer Rate Order

Actors: Customer, Deliverer, Cook

Pre: Customer is registered. Order status is complete.

Post: Ratings saved to Order, Staff warning may be affected, Staff may be fired.

Success:

1. Customer rates food (Cook) 1-5, delivery (Deliverer) 1-5.
2. Customer Ratings saved into Order
3. Rating updates Food rating history. Rating updates Deliverer rating history.

Alternative:

1a. For each, if rating <=2, must write a complaint sentence

3a. FoodRating == 1. Food has 2 previous rating == 1. Food is dropped. Cook foodDropCount++.

if foodDropCount%2 == 0, warningCount++

if warningCount == 4, Cook fired

3b. DeliveryRating == 1. Deliverer has 2 previous rating == 1. Deliverer warningCount++.

if warningCount == 4, Deliverer fired

Exception:

3a. Food item deleted before Rating is saved into Food rating history

**UC 6** //must use tokens for petri-net for rating!!

Name: Deliverer Rate Customer

Actors: Deliverer, Customer

Pre: Order status is complete.

Post: Rating saved to Order, Customer status may be affected

Success:

1. Deliverer rates Customer 1-5
2. Deliverer Rating saved into Order
3. Rating updates Customer rating history and avgRating.

Alternative:

3a. Customer OrderCount >= 4. AvgRating = x:

x > 4 => Customer is VIP

1 < x < 2 => Customer loses registration, is Visitor

x == 1 => Customer is Blacklist

**UC 7 //needs cleanup**

Name: Manager Handles Customer Support/Management

Actors: Manager, Customer

Pre: Manager views customer questions/concern

Use-cases:

1. Manager can see the order history of the customer
2. Manager can see the complaints made by the customer
3. Manager can manually promote/demote the customer

**UC 8 //very simple**

Name: Employment Registration

Actors: Cook, Deliverer, Salesperson, Manager

Pre: Manager needs new employee, accepting applications to hire

Post: Manager hires new employee

Success:

1. Manager posted request for new employee
2. New employee chooses a restaurant to register for employment
3. Manager reviews application

Alternative:

3a. if reject, loop back to reviewing applications

3b. if accept, User is added to Restaurant Staff,

**UC 9 //very simple**

Name: Remove Warning

Actors: Manager, Staff (Cook, Deliverer, Salesperson)

Pre: Staff has warningCount >= 1, <=3 (Cook/Deliverer), <= 2 (Salesperson)

Post: Staff warningCount--

Success:

1. Manager removes 1 warning

Exception:

1a. Staff receives another warning during removal process,

Deliverer/Cook: warningCount == 4

Salesperson: warningCount == 3

**UC 10 //very simple**

Name: Pay Salary

Actors: Manager, Staff (Cook, Deliverer, Salesperson)

Success:

1. Manager pays the employee their salary. Record in payment history
2. Employee receives their salary

**UC 11**

UseCase: Add, Change, Delete (CRUD) Food

Actors: Cook

*Group of Use-cases/General Methods*:

1. Cook can change the price, name, & description of a food item
2. Cook can add a food item from menu
3. Cook can delete a food item from menu

Success:

1. Cook selects existing or new food item
2. Cook inputs/changes food feature (price, name, description), or Cook deletes food item.
3. Cook submits changes, food item is updated in database.

Exception:

1a. Cook is fired in the middle of viewing

2a. Food is dropped in the middle of viewing

2b. Sends error, food not found

**UC 12**

Name: Order Supplies

Actors: Cook, Salesperson

Success:

1. Cook requests supplier to order supplies
2. Salesperson orders supplies

**UC 13**

Name: Rate Supplies

1. Cook rates supplies 1-5
2. Salesperson rating history updated. May affect Salesperson commission/staff status.

Alternative:

4a. Supplies Rating == 5. Salesperson previous 3 ratings == 5. Salesperson commission 10% increase.

4b. Supplies Rating <= 2 (complaint). Salesperson has history of 2 complaints.

3 complaints total == 1 warning, commission 10% reduction

3 warnings = Salesperson fired

If number of salesperson = 1, Need to find new salesperson (need at least 2 in restaurant), manager must hire someone (maybe within time frame?)