CONES Design

Authors: Daniel Carlson, Daniel Pickard, and Sidi Diaby
Group: 17

Document Revision History

Date	Version	Description	Author
3/2	0.0	Initial Draft	All
4/10	0.1	Updated Class	All
4/17	0.2	Updated System Architecture Details	All
5/1	1.0	Final Release	All

Table Of Contents

1. Introduction

1.1. System Overview

The design document as present outlines the Clinical Orders Notification System designed for use within a single hospital system. The design of the system is to interact with Hospital Databases to obtain orders and represent them in a medium that is easy to interact with and store orders so that they are available for later use or analysis.

1.2. Design Objective

The objective of the document is as follows

- Provide a clearly outline on what needs to be programmed for CONES and provided for CONES to use
- Provide an idea for what classes hold what responsibilities
- Provide enough responsibilities for each class so that test cases can easily be written for expected outputs.

1.3. References

The CONES Requirements document available on the CSCI5801 moodle site.

2. Glossary

CONES: Clinical Orders Notification Entry System.

Order: An order is an electronic form that a request has been made for a certain drug by a user for a specific patient. Orders persist throughout the system of CONES.

Doctor: A user who is a registered doctor. This user can create an order and automatically have it sent to the pharmacy for approval.

Nurse: A user who is a registered nurse. This user needs an
Nurse Practitioner: Similar to a doctor in the sense that orders created are automatically sent to the pharmacy.

Pharmacist: A pharmacy employee who has doctor-level credentials. Can approve orders sent to the pharmacy, but cannot create orders.

Pharmacy Technician: A pharmacy employee who has medical training, but not at the same level of a doctor. They cannot approve orders, but they can physically prepare the orders once approved. Not within the system scope.

Anesthesiologists: Doctors who specialize in drug interactions, often using The Clinical Monitoring System in their line of work.

Clinical Monitoring System: A Third Party Software designed to test drug-to-drug interactions.
DBS: database system
CMS: Clinical Monitoring System

3. Use Cases

3.1. Normal Interaction

3.1.1. Use Case Name: Place An Order

Iteration: Filled.

Summary: The user make an order

Basic Course of Events:

1. The user selects the drug and place the order

2. The software will order the drug

3. The software update the status of the order to “ordered”.

4. The software will record the user who ordered the drug, and the user who provided the permission.

Alternative Paths: None

Exception Paths: None

Extension Points: None

Trigger: the user wants to place an order.

Assumptions: The user can find the drug in the database.

Precondition: the user has the permission required to make an order

Postcondition: the order will be placed.

Author: Sidi Diaby

Date: 5/1/17

3.1.2. Use Case Name: Update Order Status

Iteration: Filled

Summary: CONES will allow nurses, nurse practitioners, doctors, and pharmacists to update drug lifecycle status to eventual confirmation.

Basic Course of Events:

1. Order is overviewed by a doctor and approved
2. Order is sent to pharmacy and is overviewed by a pharmacist
3. Pharmacist then fills the order
4. Order is officially confirmed and recorded with CONES database
5. Medical Administrator who placed the order originally is notified that order is confirmed

Alternative Paths:

If the person who placed the order is a Nurse Practitioner or Doctor then the order does not need approval from another doctor and can bypass step 1 and sent off to the pharmacy for confirmation.

Exception Paths: Pharmacist can reject the order and have it sent back the doctor.

Extension points: None

Trigger: Order is placed by medical administrator.

Assumptions: Doctors, Nurses, Nurse Practitioners, and Pharmacists all are authorized to use The CONES System.

Precondition: Order has been placed and is within the system.

Postcondition: Order has been confirmed and medicine can be picked up and/or administered to the patient

Author: Daniel Carlson

Date: 5/1/17

3.1.3. Record Patient Compliance

Iteration: Filled

Summary: CONES will allow anesthesiologists to confirm and monitor patient compliance.

Basic Course of Events:

1. Medical Administrator, usually a nurse, will verbally administer a questionnaire to the patient, as well as a physical copy of the questionnaire.
2. Patient answers questions and their answers are recorded electronically by the nurse
3. Patient Compliance is stored in medical history

Alternative Paths: None

Exception Paths: None

Extension Points: None

Trigger: Nurse meets with patient face-to-face for first time

Assumptions: Medically administrator is authorized to access the questionnaire within CONES system.

Precondition: Patient is in the hospital and is able to answer question in some manner of communication.

Postcondition: Patient is now ready to be given/administered medication prescribed within an order.

Author: Daniel Carlson

Date: 5/1/17

3.1.4. Discontinue Order

Iteration: Filled.

Summary: The user discontinues his/her order.

Basic Course of Events:

1. The user accesses the placed orders

2. The user will select his/her order to discontinue.

3. The user will request a discontinuation of his order.

4. The software will discontinue the order.

5. The software will record the order information

6. The software will update the status of the order to “discontinued”.

7. The software will record the user who discontinued the order, and the user who provided the permission.

Alternative Paths: None

Exception Paths: None

Extension Points: None

Trigger: the user wants to discontinue an order.

Assumptions: An item has already been ordered.

Precondition: the user needs to have a permission and the order hasn’t been discontinued previously.

Postcondition: the order will be discontinued.

Author: Sidi Diaby

Date: 5/1/17

3.1.5. Sorting/Viewing Order

Iteration: Filled

Summary: CONES will allow medicinal administrators to analyze and sort orders.

Basic Course of Events:

1. User accesses orders as stored in some sort of database TBD.
2. Orders can be sorted by various features including, but not exclusive to, date, lifecycle, and status.

Alternative Paths: None

Exception Paths: None

Extension points: None

Trigger: User accesses order database.

Assumptions: User is authorized to view orders and their respective lists of data.

Precondition: Orders have been placed and recorded within the CONES system.

Postcondition: Orders can be viewed.

Author Daniel Carlson

Date: 5/1/17

3.1.6. Clinical Monitoring System Use

Iteration: Filled

Summary: CONES will allow The Clinical Monitoring System to interface with itself to identify drug-to-drug interactions.

Basic Course of Events:

1. User Executes Clinical Monitoring System Software through its respective portal
2. Clinical Monitoring System requests the data necessary from CONES databases
3. CONES returns the requested data to The Clinical Monitoring System
4. Clinical Monitoring System executes and returns the results to the user

Alternative Paths: None

Exception Paths: None

Extension points: None

Trigger: User runs the program.

Assumptions: User has authorized credentials to access the information needed to run The Clinical Monitoring System and view any resulting info.

Precondition: Data that is required for The Clinical Monitoring System to operate has been entered into the system and is able to be accessed.

Postcondition: Clinical Monitoring System is active and can be or has been run, depending on the specifications of The Clinical Monitoring System.

Author: Daniel Carlson

Date: 5/1/17

4. Design Overview

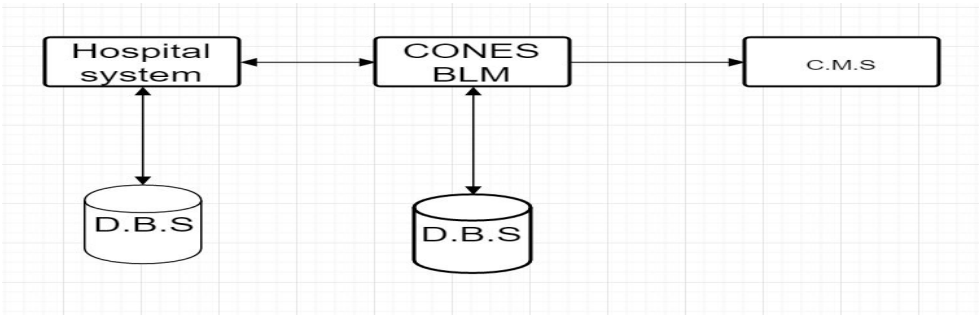
4.1. Introduction

The system of CONES is designed to tie into the Hospital Database, meaning a user would log onto the Hospital System and access CONES as a program. From there a user's information (ID, Medical Title, Name) would persist into the system. From a CONES interface, users can create, edit, approve, or cancel orders. They can also access The Clinical Monitoring System or Database.

4.2. System Architecture :

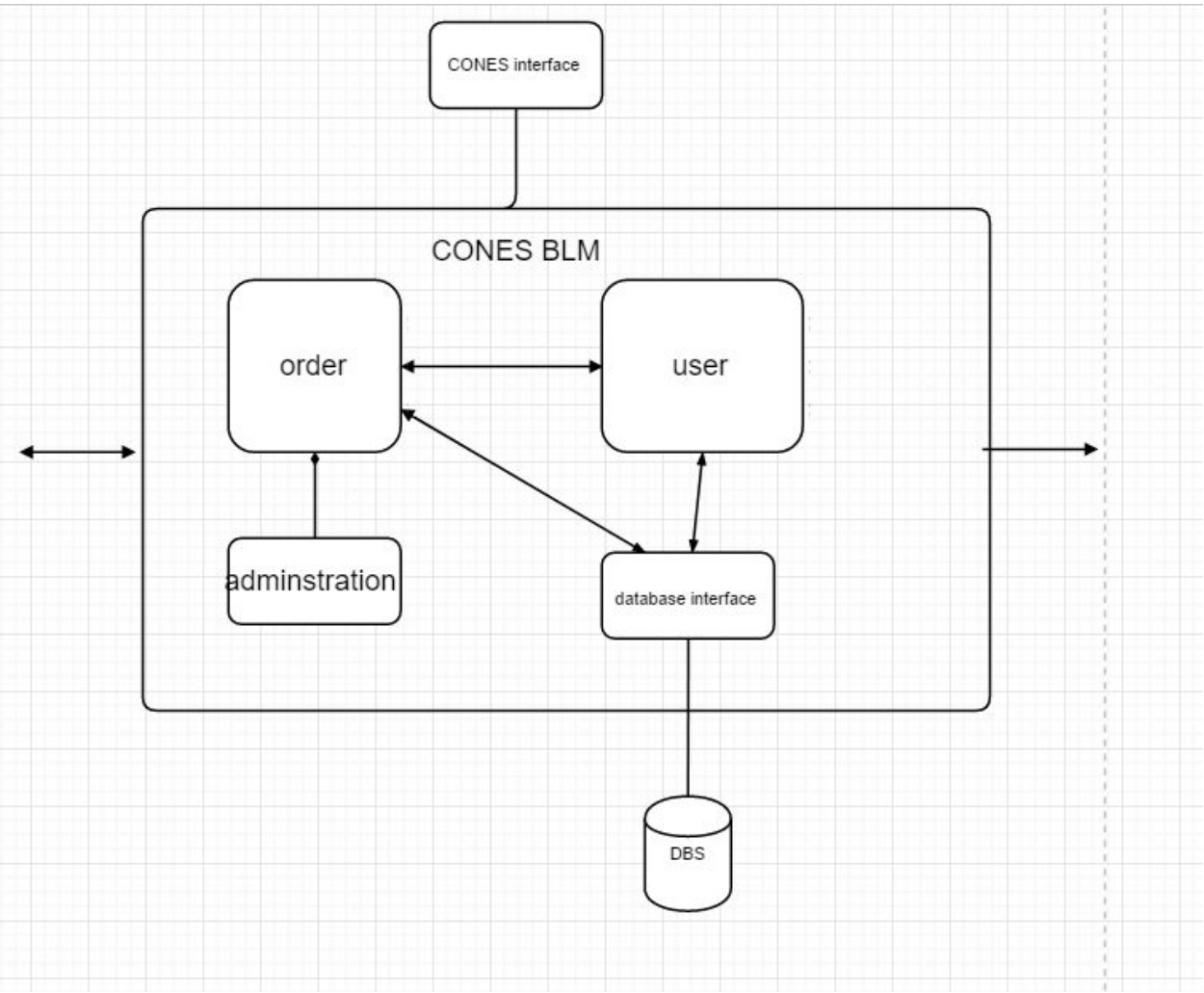
4.2.1 System overview:

The Users of CONES will access the BLM through their hospital system which will be able to interact with CONES BLM to performs the desired actions. The Clinical Monitoring system also interact with CONES.



4.2.2: CONES Architecture:

CONES will have a user class which will allow the appropriate user to perform actions, and interact with CONES Order class which is a representation of medical orders. CONES also possesses its own database interface which permits it to store, retrieve, or modify orders in the database. Finally, CONES has its own interface which permits the software to interact with hospital systems and clinical monitoring systems.



4.3. System Interfaces

4.3.1. **Hospital Interface:** Interface designed by the hospital. It handles most of the front end interaction with the user such as login and access to the permission to access the system. Design of the interface is left to the hospital and their own design plans.

4.3.2. **CONES Interface:** Interface that is being implemented to allow the user to access the functionality of CONES. Will be responsible for all order capabilities,

notifications, approvals, and administrations done in the hospital. The interface will also allow access to the Clinical Monitoring System and database of all orders in the system bases on user's access level.

4.3.3. **Observer Interface:** Interface responsible for handling the necessary parties for notifications. It will send notifications to the pharmacy staff when an order has been placed, and after filled will send a notification to only the staff needed for the administration of the order.

4.4. Constraints and Assumptions

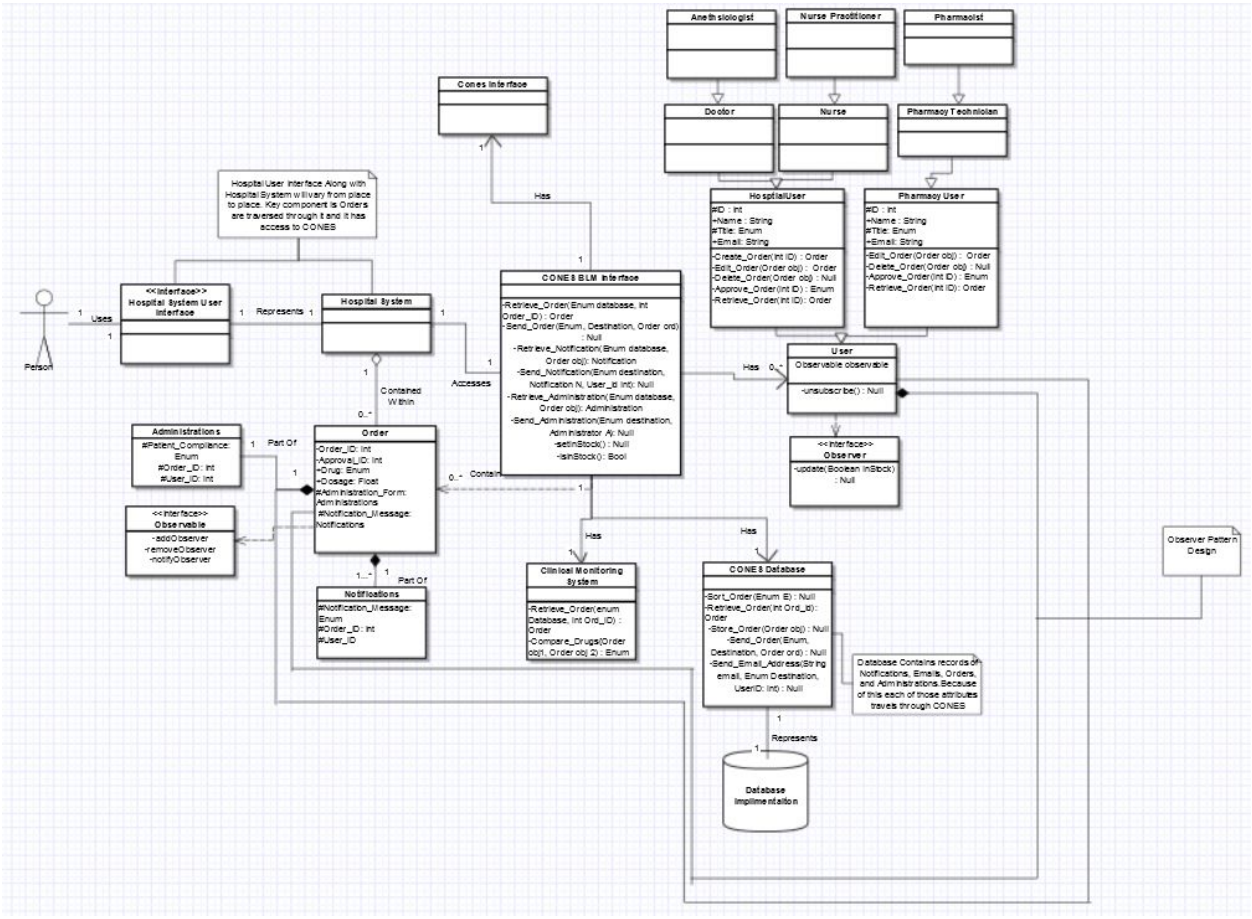
Constraints relating to CONES are mostly outside of the system scope, but need to be handled by the hospital system such as security, authentication, and an ability to have orders created by CONES to persist in it. There are also physical aspects such as making sure the hospital's pharmacy stations have access to the same software the hospital staff stations do as well as making sure CONES has enough storage for the high volume of orders that need to be stored.

5. Design

5.1. Introduction

The design portion is split into two sections. The UML Diagram to show how data flows and is represented by parts of the system and The Dynamic Models to specifically show the routes of data and actions.

5.2. UML Diagram

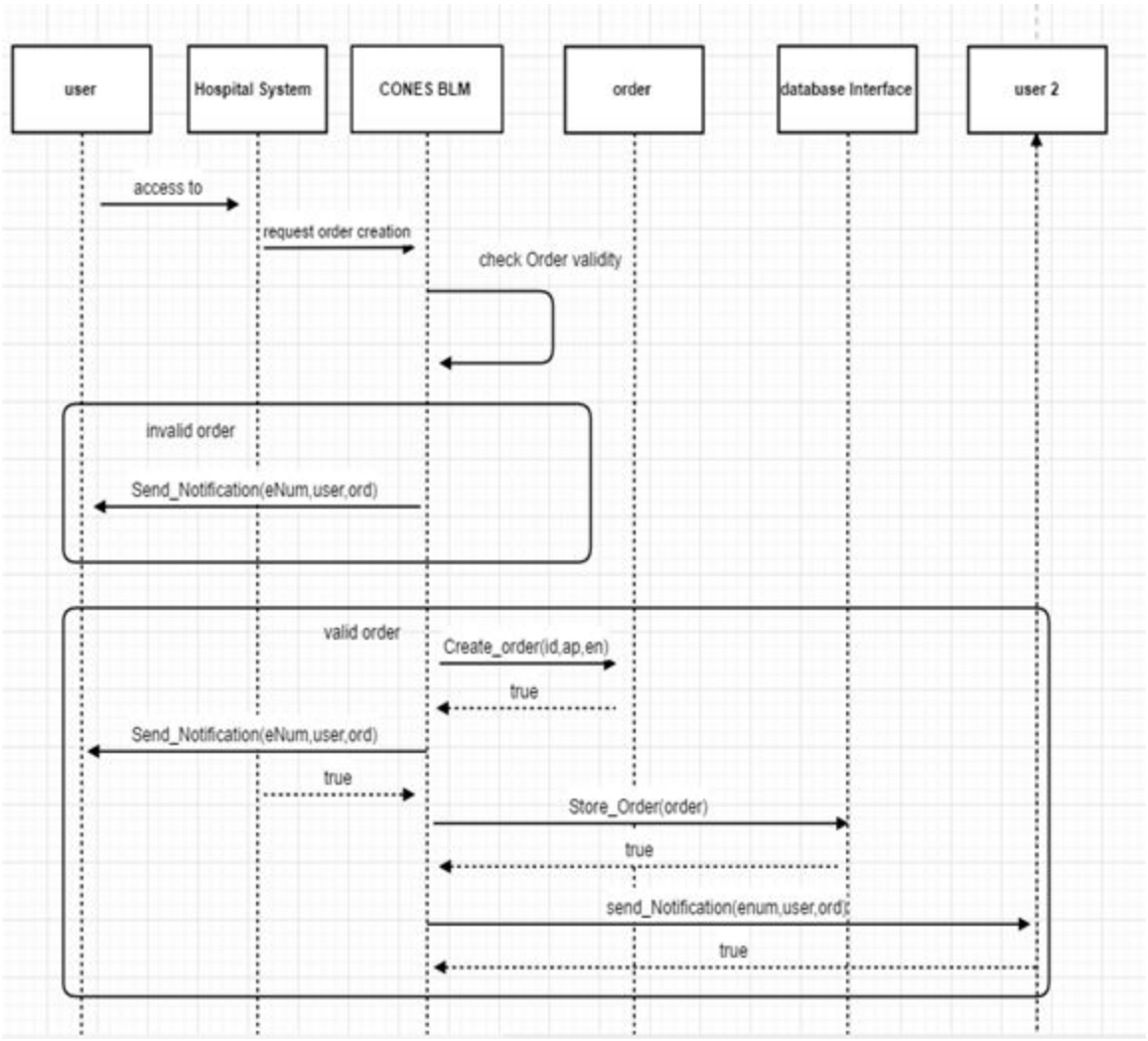


5.3. Dynamic Model

Scenario Name: create an order

Scenario Description: user use CONES to create an order, CONES will check if the order is valid. Then will process the order and notify the users

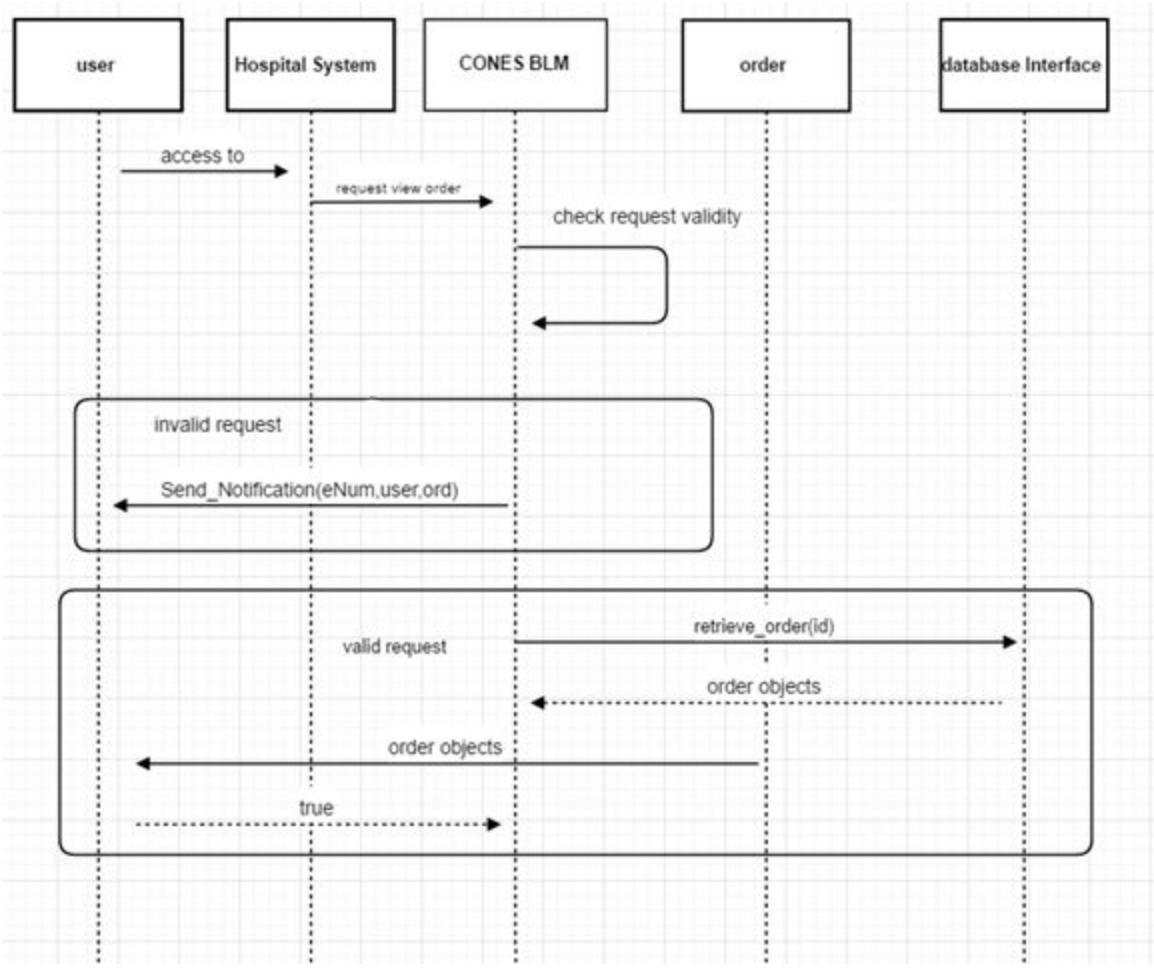
Sequence Diagram:


user
Hospital System
CONES BLM
order
database Interface
user 2
access to
request order creation
check Order validity
invalid order
Send_Notification(eNum,user,ord)
valid order
Create_order(id,ap,en)
true
Send_Notification(eNum,user,ord)
true
Store_Order(order)
true
send_Notification(enum,user,ord)
true

Scenario Name: view order

Description: the user will be able to view an order in the database via CONES

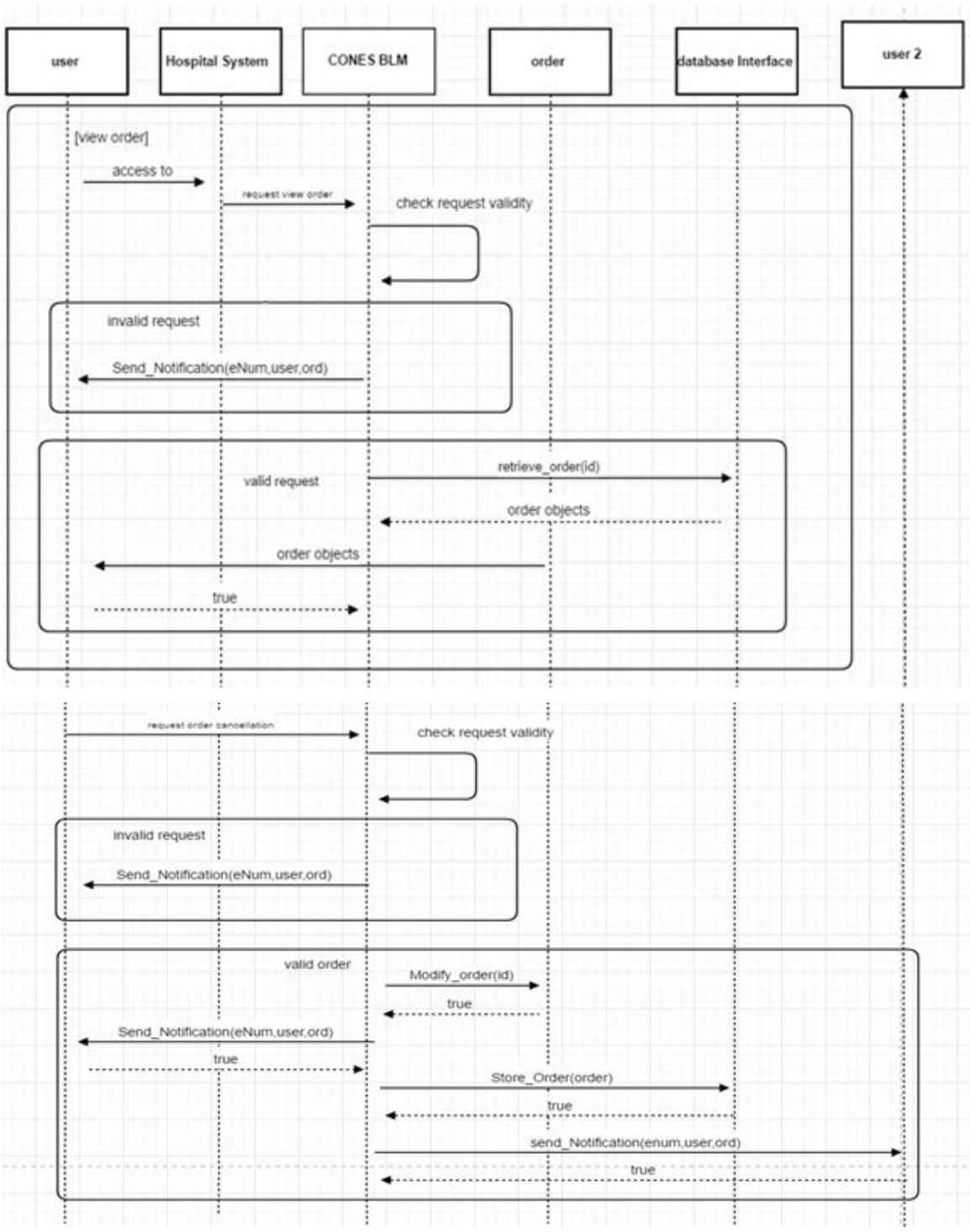
Diagram:



Scenario Name: Modify order

Scenario Description: The user will look for the order in the database, use CONES to Modify the order, the order will be Modified, and the notification will be sent.

Scenario Diagram:



6. Detailed Function and Class Definitions

6.1. Class Definitions

CONES

Attributes:None

Methods:

1. Method: *Retrieve_Order(Enum Database, int ID)*
Return Type: *Object:Order*
Parameters: *authorization- user can view orders*
Return value: *order object*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *UserID, PatientID, Drug Name, Dosage*
Methods called: *None*

Processing logic:

CONES accesses the database of orders and selects the order they want to view. Order is accessed and displayed for the user to see, edit, approve, or cancel.

Send_Order(Enum destination, Order ord) : Null

2. Method: *Send_Order(Enum destination, Order ord)*
Return Type: *Null*
Parameters: *authorization- user can view orders*
Return value: *Null*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is sent.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:

CONES sends the order and its information to all the necessary recipients.

3. Method: *Retrieve_Notification(Order obj)*
Return Type: *Notification*
Parameters: *authorization- user can view orders*
Return value: *Notification message*

Pre-condition: *order is in the system actively or within the database*
Post-condition: *Notification is received.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES retrieves the notifications and their information for the user.

4. Method: *Send_Notification(Order obj)*
Return Type: *Enum*
Parameters: *authorization- user can view orders*
Return value: *Success or failure*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *Notification is sent.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES sends the notifications and their information to all the necessary recipients.

Retrieve_Administration(Order obj): Administration

5. Method: *Retrieve_Administration(Order obj)*
Return Type: *Administration*
Parameters: *authorization- user can view orders*
Return value: *Success or failure*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *Administration information is retrieved and displayed.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES retrieves the Administration object and its information for the user.

6. Method: *Send_Administration(Administrator A)*
Return Type: *Null*
Parameters: *authorization- user can view orders*
Return value: *Null*

Pre-condition: *order is in the system actively or within the database, and contains an Administration object*
Post-condition: *Administration information is sent.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES sends the Administration object and its information to all the necessary recipients.

CONES/Hospital Interface
Attributes:TBD

Methods:TBD

Hospital System User Interface
Attributes:TBD

Methods:TBD

Hospital System
Attributes:TBD

Methods:TBD

Hospital User
Attributes:

1. Attribute: *ID*
Type: *Int*
Description: *ID of user in the system*
Constraints: *should be unique to an individual user*

2. Attribute: *Name*
Type: *String*
Description: *Name of the user*
Constraints: *None*

3. Attribute: *Title*
Type: *Enum*
Description: *The professional title of an individual, Nurse, Nurse Practitioner, Doctor, or orderly*
Constraints: *None*

4. Attribute: *Email*
Type: *String*
Description: *The professional email of the user.*
Constraints: *None*

Methods:

1. Method: *Create_Order()*
Return Type: *Object:Order*
Parameters: *authorization- user can cancel orders*
Return value: *Order Object or fail state message*
Pre-condition: *None*
Post-condition: *None*
Attributes read/used: UserID, PatientID, Drug Name, Dosage
Methods called: *None*

Processing logic:
A user creates an order. They enter the necessary information into the blank order and order is officially created and sent for approval to either a doctor or the pharmacy.

2. Method: *Edit_Order(int ID)*
Return Type: *Object:Order*
Parameters: *order- order has been created. authorization- user can modify orders*
Return value: *a new order object*
Pre-condition: *None*
Post-condition: None
Attributes read/used: UserID, PatientID, Drug Name, Dosage
Methods called: *Retrieve_Order()*

Processing logic:

The user attempts to edit the order. Once order is edited, a new order object is created and the old one is recorded into the database and deleted.

3. Method: *Delete_Order(int ID)*
Return Type: *Enum*
Parameters: *order- order has been created. authorization- user can cancel orders*
Return value: *success or failure*
Pre-condition: *Order has been created.*
Post-condition: *Order is recorded into database.*
Attributes read/used: *None*
Methods called: *Retrieve_Order()*

Processing logic:

User submits a cancellation request. Order is deleted and recorded into database.

4. Method: *Approve_Order(Int ID)*
Return Type: *Enum*
Parameters: *authorization- user can approve orders*
Return value: *success or failure*
Pre-condition: *Order is active in the system*
Post-condition: *None*
Attributes read/used: UserID
Methods called: *Retrieve_Order()*

Processing logic:

The order is accessed then approved by an authorized user. The order is then either thrown to the pharmacy for another approval ID or is actively put into the database.

5. Method: *Retrieve_Order(int ID)*
Return Type: *Object:Order*
Parameters: *authorization- user can view orders*
Return value: *order object*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *UserID, PatientID, Drug Name, Dosage*
Methods called: *None*

Processing logic:

Cones accesses the database of orders and selects the order based on the ID. Order is accessed and displayed for the user to see, edit, approve, or cancel.

Pharmacy User

Attributes:

1. Attribute: *ID*
Type: *Int*
Description: *ID of user in the system*
Constraints: *should be unique to an individual user*

2. Attribute: *Name*
Type: *String*
Description: *Name of the user*
Constraints: *None*

3. Attribute: *Title*
Type: *Enum*
Description: *The professional title of an individual Pharmacist or Pharmacist technician.*
Constraints: *None*

4. Attribute: *Email*
Type: *String*
Description: *The professional email of the user.*
Constraints: *None*

Methods:

1. Method: *Edit_Order(int ID)*
Return Type: *Object:Order*
Parameters: *order- order has been created. authorization- user can modify orders*
Return value: *a new order object*
Pre-condition: *None*

Post-condition: None
Attributes read/used: UserID, PatientID, Drug Name, Dosage
Methods called: *Retrieve_Order()*

Processing logic:

The user attempts to edit the order. Once order is edited, a new order object is created and the old one is recorded into the database and deleted.

2. Method: *Delete_Order(int ID)*
Return Type: *Enum*
Parameters: *order- order has been created. authorization- user can cancel orders*
Return value: *success or failure*
Pre-condition: *Order has been created.*
Post-condition: *Order is recorded into database.*
Attributes read/used: *None*
Methods called: *Retrieve_Order()*

Processing logic:

User submits a cancellation request. Order is deleted and recorded into database.

3. Method: *Approve_Order(Int ID)*
Return Type: *Enum*
Parameters: *authorization- user can approve orders*
Return value: *success or failure*
Pre-condition: *Order is active in the system*
Post-condition: *None*
Attributes read/used: UserID
Methods called: *Retrieve_Order()*

Processing logic:

The order is accessed then approved by an authorized user. The order is then either thrown to the pharmacy for another approval ID or is actively put into the database.

4. Method: *Retrieve_Order(int ID)*
Return Type: *Object:Order*
Parameters: *authorization- user can view orders*
Return value: *order object*

Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *UserID, PatientID, Drug Name, Dosage*
Methods called: *None*

Processing logic:

Cones accesses the database of orders and selects the order based on the ID. Order is accessed and displayed for the user to see, edit, approve, or cancel.

CLINICAL MONITOR SYSTEM:

Attributes:None

Methods:

1. Method: *Retrieve_Order(Enum Database, int ID)*

Return Type: *Object:Order*
Parameters: *authorization- user can view orders*
Return value: *order object*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *UserID, PatientID, Drug Name, Dosage*
Methods called: *None*

Processing logic:

CONES accesses the database of orders and selects the order they want to view. Order is accessed and displayed for the user to see, edit, approve, or cancel.

2. Method: *CompareDrugs(Order obj1, Order obj2)*

Return Type: *Enum*
Parameters: *authorization- user can view orders*
Return value: *Complication when combined or no complication*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *Drug Name*
Methods called: *Retrieve_Order()*

Processing logic:

Database checks the names of drugs from two orders against each other, then returns whether or not their is a known complication when they interact.

CONES DATABASE:
Attributes:None
Methods:

1. Method: *Sort_Order(Enum E)*
Return Type: *Null*
Parameters: *authorization- user can view database*
Return value: *Null*
Pre-condition: *database exists*
Post-condition: *Database is sorted in new way by selected Enum.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES sorts the database of orders based on the selected Enum; order ID, Drug Name, Patient ID, Date, Order Status.

2. Method: *Retrieve_Order(Enum Database, int ID)*
Return Type: *Object:Order*
Parameters: *authorization- user can view orders*
Return value: *order object*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*
Attributes read/used: *UserID, PatientID, Drug Name, Dosage*
Methods called: *None*

Processing logic:
CONES accesses the database of orders and selects the order they want to view. Order is accessed and displayed for the user to see, edit, approve, or cancel.

3. Method: *Store_Order(Order obj)*
Return Type: *Null*
Parameters: *authorization- user can create orders*
Return value: *Null*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is accessed.*

Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES stores the order in the database of orders.
Send_Order(Enum Destination, Order ord) : Null
4. Method: *Send_Order(Enum Destination*, *Order obj)*
Return Type: *Null*
Parameters: *authorization- user can view orders*
Return value: *Null*
Pre-condition: *order is in the system actively or within the database*
Post-condition: *order is sent.*
Attributes read/used: *None*
Methods called: *None*

Processing logic:
CONES sends the order object and all of its information to all necessary recipients.

6.2. Object Definitions

Administrations:

1. Attribute: *Patient_Compliance*
Type: *Enum*
Description: *Whether or not the patient has filled out their compliance forms.*
Constraints: *None.*

2. Attribute: *Order_ID*
Type: *Int*
Description: *ID of the order the object is connected to.*
Constraints: *Should be a positive integer.*

2. Attribute: *User_ID*
Type: *Int*
Description: *ID of the User who created the object.*
Constraints: *Should be a positive integer.*

Orders:

1. Attribute: *Order_ID*
Type: *Int*
Description: *ID of each individual order created.*
Constraints: *Should be a positive integer.*

2. Attribute: *Approval_ID*
Type: *Int*
Description: *ID of Pharmacist who approved the order.*
Constraints: *Should be a positive integer assigned to an employee.*

3. Attribute: *Drug_Name*
Type: *Enum*
Description: *Name of prescribed drug, selected from the list of all drugs in the pharmacy.*
Constraints: *Drug is in Pharmacy and name is in the system*

4. Attribute: *Dosage*
Type: *Float*
Description: *Numerical float value of the dosage requested for the patient*
Constraints: *should be a valid dosage level (non-negative)*

5. Attribute: *Administration*
Type: *Object: Administration*
Description: *Forms filled out by the patient.*
Constraints: *Filled out by a valid patient or guardian with ID*

5. Attribute: *Notification_Message*
Type: *Object: Notification*
Description: *Notification object created with message to be sent to the pharmacy to await approval.*
Constraints: *Order must be created with valid recipients.*

Notifications:

Attribute:

1. Attribute: *Notification_Message*
Type: *Enum*
Description: *Notification object with selected message*

Constraints: *Order must be created with valid recipients for notification message.*

7. Test Cases

Test Case ID: Make an Order

Description	Test if the order has been placed.
Test Inputs:	Drug name and dosage
Expected Results:	Received a notification confirming that the order has been placed.
Dependencies:	None
Initialization:	The user should have the permission to make an order. The drug should be in the Database.
Test Steps:	1. Test if the drug is a valid 2. Test if the order has been transmitted 3. Test if the status of the order has been updated to "ordered" 4. Test if the user name and information have been recorded
Owner:	Sidi Diaby

Test Case ID: Discontinue Order

Description	Test if the order has been discontinued.
Test Inputs:	Discontinue trigger
Expected Results:	Received a notification confirming that the order has been discontinued.
Dependencies:	User discontinuing the order has the proper authorization to discontinue an order
Initialization:	The user should have the permission to discontinue the order. The order should be placed.
Test Steps:	1. Test if the order recipient has been notified. 2. Test if the status of the order has been updated to "Discontinued" 3. Test if the user name and information have been recorded.
Owner:	Sidi Diaby

Test Case ID: Update Order

Description:	Test if the order has been updated.
Test Inputs:	update trigger, data to be updated
Expected Results:	Received a notification confirming that the order has been updated.
Dependencies:	Cancel order
Initialization:	The user should have the permission to update the order. The order should be placed.
Test Steps:	1. Test if the order is different from the previous order. 2. Test if the recipient of the order has been notified. 3. Test if the previous order has been cancel 4. Test if the new order has been placed 5. Test if the user name and information of the user have been recorded.
Owner:	Sidi Diaby

Test Case ID **Recording Patient Compliance**

Description:	The test case shall create multiple records of patient compliance, testing to see if all options within the digital patient compliance can be correctly entered and recorded.
Test Inputs:	Whatever answers in whatever form (write-in, multiple choice, etc) is required
Expected Results:	All the orders are correctly recorded with the system and when accessed all info was correctly stored.
Dependencies:	Create An Order Test. Sorting Orders.Viewing Orders
Initialization:	Patient compliance is recorded when a drug is prescribed, therefore an order needs to be properly created and properly recorded.
Test Steps:	1. Various template orders are created and confirmed into the system 2. The patient compliance is filled with whatever is being tested for that particular iteration of the test 3. The order is officially confirmed and the record 4. Order is then accessed in whatever database it is stored and the record is compared to the inputs.
Owner:	Daniel Carlson

Test Case ID **Viewing Orders**

Description:	The test case will create multiple orders, all of which are confirmed, then test to see if those orders can be accessed from the database.
Test Inputs:	Complete Orders
Expected Results:	Order is recorded properly into the database and can be accessed
Dependencies:	Create An Order
Initialization:	Order is properly placed and confirmed by users with the proper authorization.
Test Steps:	1. User creates a series of orders 2. Orders are confirmed within the system and recorded 3. User accesses the database to see if orders are properly recorded
Owner:	Daniel Carlson

Test Case ID **Sorting Orders**

Description:	The test case will sort the orders in various (Alphabetically by name, drug type, dosage, doctors/nurses/NP involved in order, etc).
Test Inputs:	Complete Orders
Expected Results:	Orders are displayed in correct order as specified
Dependencies:	Create and Order and Viewing Orders
Initialization:	Order is properly placed and confirmed by users with the proper authorization.
Test Steps:	1. User creates a series of orders 2. Orders are confirmed within the system and recorded 3. User accesses the database to see if orders are properly recorded 4. User tests to see if sorting orders work properly and are still accessible
Owner:	Daniel Carlson

Test Case ID **Clinical Monitoring System Use**

Description:	Test shall use Clinical Monitoring System to detect drug-drug conflicts
Test Inputs:	Complete order along with all records from patient in question.

Expected Results:	Clinical Monitoring System will check for potential conflicts in an order. If there is one an alert will be sent to necessary parties. Otherwise order will proceed on as normal.
Dependencies:	Create an Order. Viewing Order. Sort Order. Editing Order.
Initialization:	Order is properly placed and confirmed by users with the proper authorization. Clinical Monitoring System also has access to database of CONES.
Test Steps:	1. Users create a series of orders to test. Some with obvious conflicts, some with no conflicts. 2. User runs Clinical Monitoring System orders 3. User sees if alerts are sent when needed and not when they are not needed. Also confirms they are sent to the proper place.
Owner:	Daniel Carlson

8. Appendix

A.1 Traceability Matrix

	Use Cases	Place an Order	Update Order Status	Record Patient Compliance	Discontinue Order	Sorting/Viewing Order	Third Party Software Use
Test Cases							
Create an Order		X		X		X	X
Discontinue Order					X		
Update Order			X				
Recording Patient Compliance				X			
Viewing Orders		X	X	X	X	X	
Sorting Orders						X	
Third-Party Software Use							X