Container Transport Tracking System (CTTS)
Software Design Document (SDD)
For Shipping X Corporation (SX)
Version 1.12

11/02/2022

for

CS 325

Professor Czernik

by

Team #1

Evan Gardner

Premkumar Patel

Robert Reichard

Table of Contents

1 Introduction and Purpose (2 pts)

The purpose of this document is to define the Software Design of the CTTS system that is being created for Shipping X Company. The Container Transport Tracking System will be designed throughout this document outlining the specifics within the high-level design as well as the low-level design. The Shipping X (SX) shipping company runs a high-tech container shipping business. SX owns ships, smart containers, and trucks. SX ships containers between Japan, Los Angeles, CA, and Baltimore MD. SX needs a new ordering and tracking system. The new ordering and tracking system will be named Container Transport Tracking System (CTTS). CTTS maintains ship, container, truck, loading/unloading crew, container, ordering, ship tracking, weather, customs, and special alert data. CTTS capabilities include the ability to reserve, schedule, track, and monitor ships, trucks, containers, loading/unloading crews, and events that may impact the shipping or contents of a container.

2 Product Requirements - Epics and User Stories (given)

For Shipping X's company, the Product Requirements for the CTTS system are:

All of the user stories can be found within the requirements document of the project

Epic 1: Order Processing and Scheduling: SX needs to be able to accept and process shipping container orders, reserve and schedule containers, space on the ship, trucks, and loading/unloading crews to ship containers from an origin to a destination. SX also needs to be able to update shipping requests and schedules as events mandate. Account creation needs to be addressed as either another epic or under this epic

Acceptance Criteria: The CTTS system will be used to create an account in which the order from the customer will be processed/ delivered so that billing can follow.

Epic 2: Billing and Payments: SX needs to be able to ensure shipping container customer bills are calculated by the billing system and are paid before reserving and scheduling the container shipment.

Acceptance Criteria: Confirm the order details by calculating a price for the service being requested by the user in which they will provide a payment method that will be sent to the AMBS for verification until it is approved for production.

Epic 3: Ship and Container Tracking: SX needs to be able to track the location of the ship, each container, and trucks.

Acceptance Criteria: The Tracking Specialist should receive the order details in which throughout the shipment all aspects can be monitored such as location, temperature, humidity, and battery life so there can be a response if any of these systems fail

Epic 4: User Interface: SX needs to be able to view and access CTTS capabilities through a web browser. SX needs a smartphone application to scan container bar codes and Q-codes to identify a container and access CTTS capabilities using a smartphone application.

Acceptance Criteria: Each of the positions within the company will monitor and report each of the containers to the CTTS so that throughout shipment the customer's orders are never lost or mishandled

Epic 5: Customs: SX needs to report to customs ship and container manifest, and request permissions to leave/enter the port. Ships will be notified (via radio) of any request approvals or denials.

Acceptance Criteria: For international shipments to Japan and back requests must be made through the SCCS to be approved or Denied Access to enter and leave a port for delivery

Epic 6: Weather Alerts: SX needs current weather information so it may warn its ship crew (via radio) of upcoming weather events.

Acceptance Criteria: The Tracking Specialist must determine whether the weather is safe to travel in and then notify each of the different positions and the customer that there will be a delay in the shipment

Epic 7: Homeland Security: SX needs current threat and terrorism information so it may warn its ship crew (via radio) of possible attacks.

Acceptance Criteria: The Tracking Specialist would be the first to notify all of the employees of Shipping X to stop what they are doing and evacuate to a safe area while also contacting the customer of the potential threat

Epic 8: Dashboard: SX needs dashboards at SX head to be able to quickly view the status of ships and containers.

Acceptance Criteria: This must describe the processes that will be carried about by each of the different personas through the CTTS dashboard

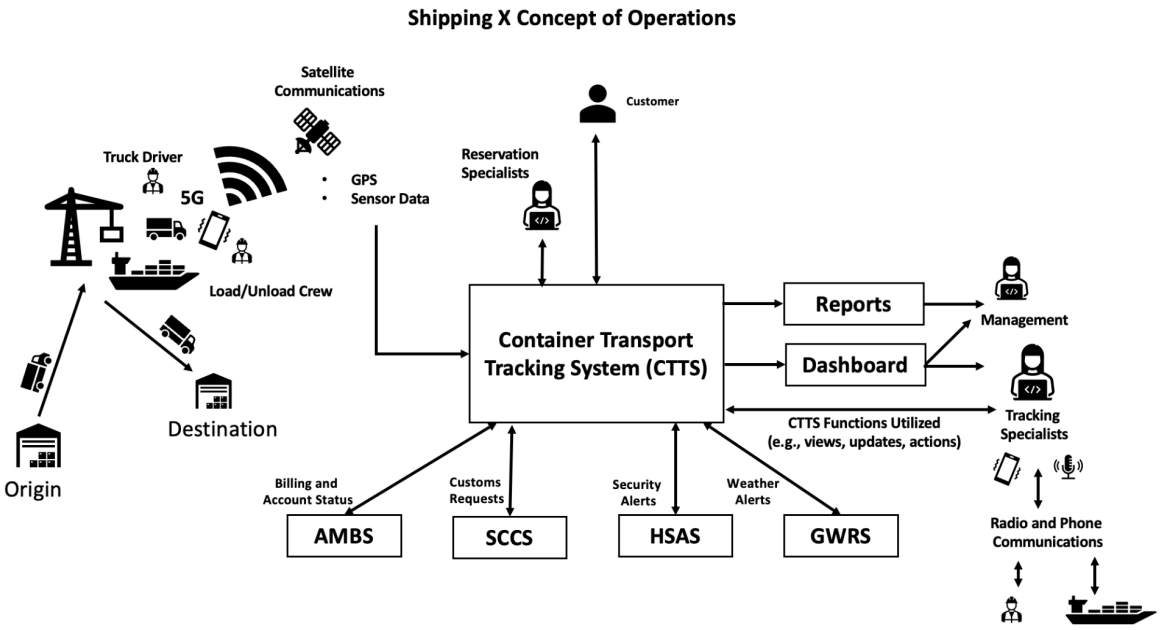
Epic 9: Authentication and Authorization: SX needs to have various roles, users, and interfaces authenticated and authorized.

Acceptance Criteria: Each employee within the Shipping X company must check in to complete their daily tasks within the CTTS system and the SX management must authorize those who have access to the system

Epic 10: Reporting: SX needs to produce monthly reports of shipping activity.

Acceptance Criteria: The SX Manager must create a monthly report for the data that has been entered into the CTTS system to determine how well the company is doing and how it can be improved

3 Product Description and Scope (2 pts)



The Shipping X (SX) shipping company runs a high-tech container shipping business. SX owns ships, smart containers, and trucks. SX ships containers between Japan, Los Angeles, CA, and Baltimore MD. SX needs a new ordering and tracking system. The new ordering and tracking system will be named Container Transport Tracking System (CTTS). CTTS maintains ship, container, truck, loading/unloading crew, container, ordering, ship tracking, weather, customs, and special alert data. CTTS capabilities include the ability to reserve, schedule, track, and monitor ships, trucks, containers, loading/unloading crews, and events that may impact the shipping or contents of a container. For example, if a container temperature must be maintained at 60 degrees or less and the temperature goes above the threshold, the CTTS will create an alert of possible content spoilage so the ship crew can respond and attempt to correct the problem.

Each ship is equipped with the ability to report its GPS location via satellite link. Each smart container contains a small device that reports its GPS location, internal temperature, humidity, and battery life via a satellite link. An assumption that can be made is that the container device is already installed, and the solution will require each device to be configured to communicate to the CTTS. The configuration information includes the RESTful API URLs, container ID, credentials, and reporting time interval. Each shipping container also has a bar code and/or Q-code that can be scanned to identify the container and to view container information such as

container number, contents (manifest), size, weight limits, and the contents owner. The CTTS also has interfaces with the USA Ship and Container Customs System (SCCS), the SX company billing system called Account Management and Billing System (AMBS), the Global Weather Reporting System (GWRS), and the Homeland Security Alerting System (HSAS).

4 Architecture and Design Goals (3 pts)

For Shipping X's company, the architectural and design goals for the CTTS system are:

- The CTTS system will rely on Google Maps or some other mapping application to show the location of the shipped items within its container to the customer at any point 24/7/365.

- For the creation of the CTTS there will be a creation of road maps and pipelines that will involve the integration of technology and processes into the system

- While building the architecture and design goals the standards and policies of the CTTS system must be implemented for different portions of the project such as the Homeland Security System, global weather reporting system, account management system, and shipping container customs.

- The risk management of the project must be created to protect all of the systems in case problems are to ensue during the complete lifecycle of the CTTS system

- The scaleability of the CTTS system depends on the number of customers, the amount of inventory, and the number of orders that can be processed at a single period of time. The system needs to rely on the amount of inventory of ships, containers, and staff to scale the system up while also relying on how much the online interface can handle. Maintenance and monitoring are key to ensure that the system is not overhandling what the company can handle

- The customer's needs for the project are also severely important since the tracking and order handling must be done correctly to ensure the customer is happy with the services to keep them coming back to Shipping X.

- The performance of the CTTS system will be based on the technologies working to manage the system and run the application ensuring that the system is able to handle all the tasks at hand with little to no delay if possible

- The adaptability of the CTTS system relies on the proper maintenance being performed when needed so that the goal is in mind when performing tasks to ensure the GPS and system can handle all operations being performed

- The Performance of the CTTS was used to Localise critical operations and minimize communications. Use large rather than fine-grain components.

- The Security of the CTTS Uses a layered architecture with critical assets in the inner layers.

- The Safety of the CTTS was used to Localise safety-critical features in a small number of sub-systems.

- The Availability of the CTTS Includes redundant components and mechanisms for fault tolerance.

- The Maintainability of the CTTS Uses fine-grain, replaceable components

5 Assumptions and Constraints (3 pts)

For Shipping X's company, the assumptions, and constraints for the CTTS system are:

- The CTTS system will rely on Google Maps or some other mapping application to show the location of the shipped items within its container to the customer at any point 24/7/365.

- Weather affecting ETA status needs to increase the time to arrive.

- The system depends on a reliable network in order to function properly.

- The shipping container sensors are assumed to be in working order.

- All external systems are assumed to be functional.

- The system depends on the ship crews' ability to transport the containers efficiently between land and the ship, as well as pilot the ship safely to its destination.

- The system needs to be active 24/7 to ensure the reports and communication between workers and customers stay active.

- Additionally, the CTTS System is hosted in a cloud-based system, so it depends on the cloud's functionality to execute its processes and deliver value to customers.

- The system also depends on the containers passing customs inspections when shipping internationally, as the container will not arrive if it fails inspection.

- Only one content type will be supported, JSON.

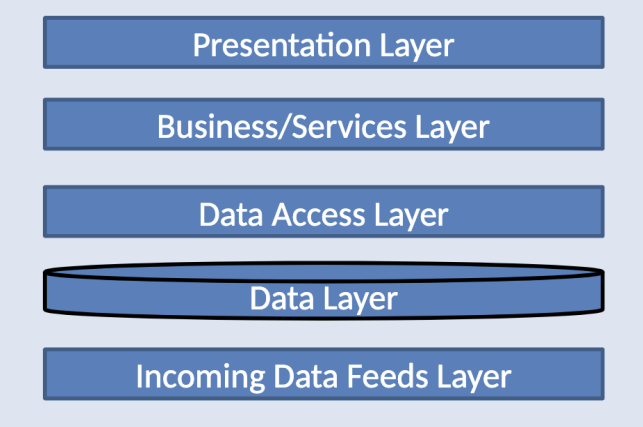
- Versioning will be accomplished by placing the version number in the URI.

- The PATCH and OPTIONS HTTP methods will not be supported.

- The services layer application server provides API routing.

6 High-Level Design

6.1 Software Architecture Layers (5 pts)



- Presentation Layer - responsible for handling all user interface and browser communication logic

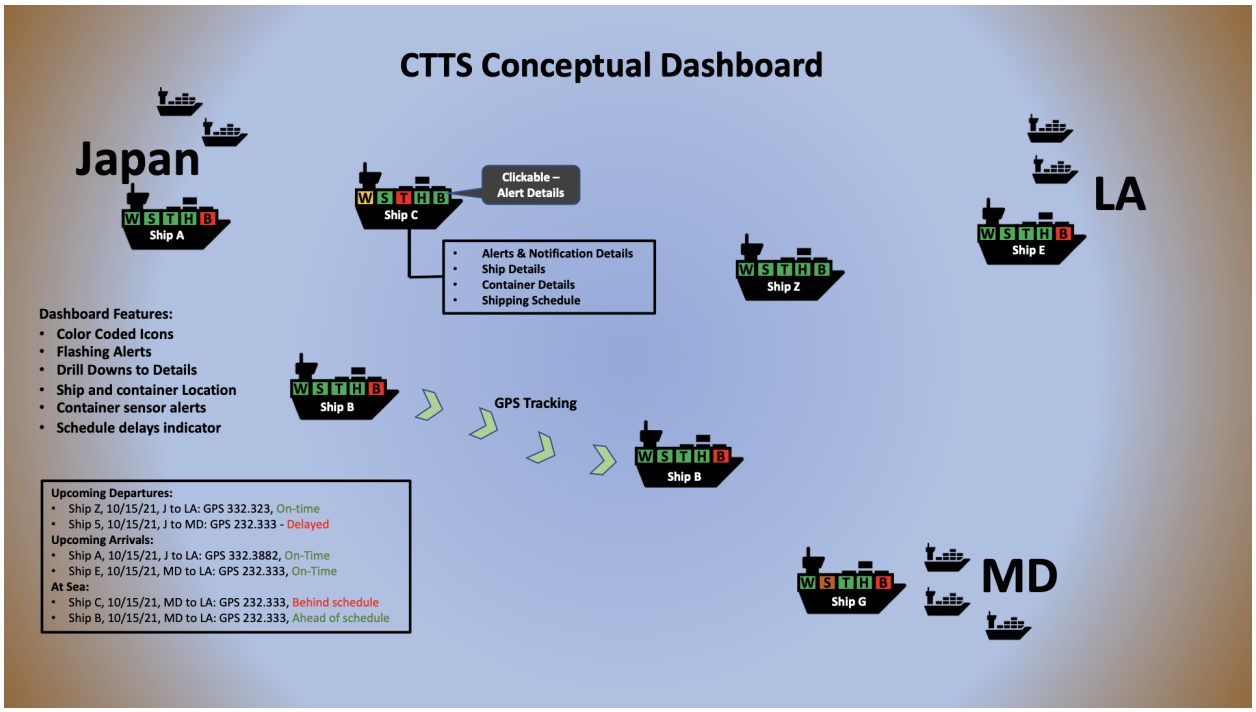
- Business/Services Layer - responsible for executing specific business rules associated with the request

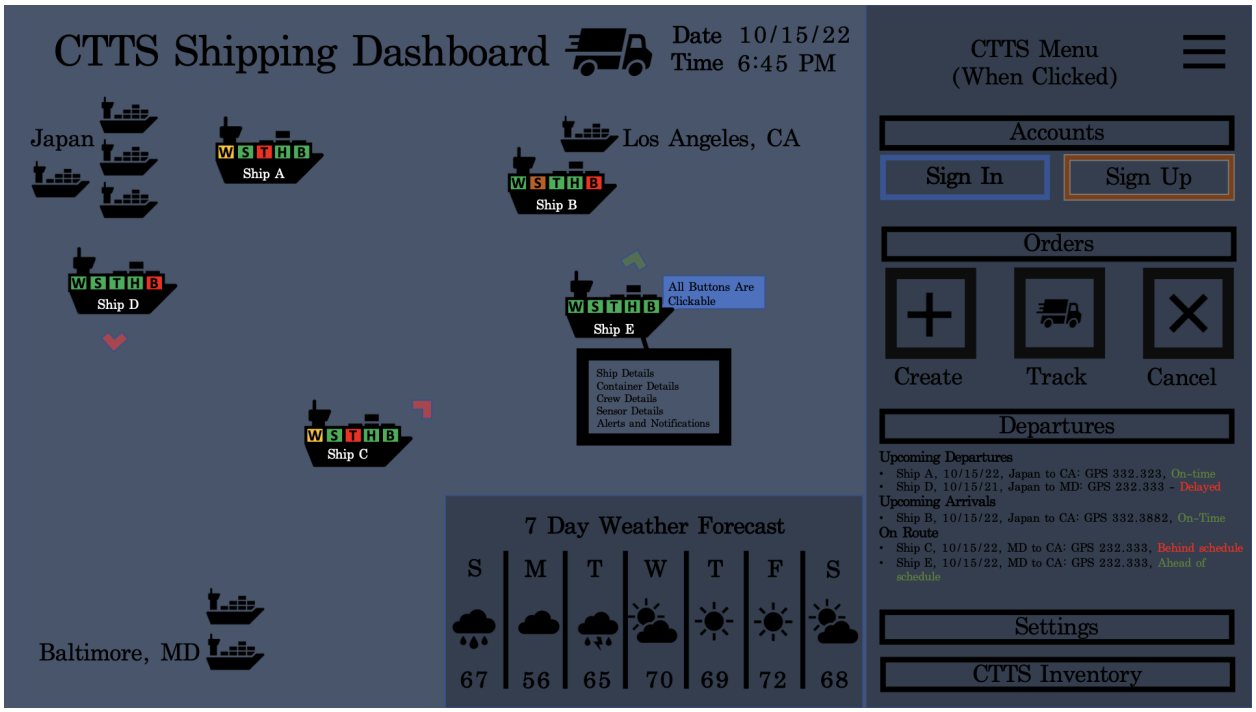
- Data Access Layer – CRUD operations, abstracts physical data store to Business/Servicers Layer

- Data Layer – data relationship model

- Incoming Data Layer – Receives incoming data on a regular basis or stream

6.2 Presentation Layer – User Interface (10 pts)





The major areas of the User Interface includes the dashboard and the CTTS Menu with these two sections broken even further down into more specific user interface options. To begin with the dashboard you can begin by looking at the top of the screen which displays to the user the name of the dashboard along with a shipping logo, date, and time. Below this, you can then see the three delivery locations that Shipping X has ports for at the moment which are Japan, Los Angeles California, and Baltimore Maryland. These three locations are named on the dashboard and show the dock where ships belonging to Shipping X may be located at the time of the dashboard being opened. This will also show the ships that are en route at the

moment of the CTTS dashboard being opened to allow the user to understand the route that the ships are taking and if the ships are on schedule for delivery. The ships will appear with an arrow in front of each if they are ahead of schedule or behind schedule with the green indicating they are ahead of schedule and red indicating behind schedule. The buttons on each of the boats are also clickable indicating the specific details of the alarms on the boats so that it is understood if there is a problem with one of the sensors that need to be taken care of by being shown with a colored visual. Below this, there is also a weather forecast for the next week that is shown which will allow the user to determine if their shipment might be delayed because of bad weather within the coming days. On the right side of the screen, there is a drop-down menu that is indicated by three lines. This will open up a menu condensing the left-hand side of the screen which is the dashboard and show the full extent of the CTTS Menu. Starting with the Customer Accounts, it shows the option to either sign in or sign up for an account with the CTTS dashboard service. Once done with either of these options the system will allow the user to view their details or other volatile information. After this section, there is an orders section with three buttons that may be pressed that will allow the user to create an order, track an order, or cancel an order if possible so that the user may complete tasks that they wish to have serviced on their own. Following this, there is a departures list where it will show a list of the ships that are en route at any given time indicating where they are going and the status of how the ship is doing on its delivery for example ship E is ahead of schedule. Finally, there is a settings option for users to be able to configure anything that they may need to be changed for their settings and the CTTS inventory which is mostly meant for management so that they can keep track of the number of containers, staff, ships, and etc. ensuring they have enough materials to complete any given order given to them at one time.

The screen framework works in a very simple fashion with the main screen taking up the entire screen when opening the application moving into a ⅔ screen when the CTTS menu pops up on the right side of the screen when the three lines are pressed condensing the left side of the screen. The sign-in and sign-up buttons will lead to a follow-up screen that will log the user in so that when they look at this screen again they can then access their account details within an account button rather than the sign or sign-up. This is followed up by the create, track, and cancel buttons which for creating it will bring you to a questionnaire about the order being created so that all the info can be taken down for the order to be placed in a sequence of screens such as order information to payment methods. Track order works by bringing up any orders placed and showing any delivery updates similar to UPS or FedEx while also showing where the order is throughout transit. The cancel order will bring you to a similar screen to the track order only this will have the list of orders and show the available orders that can be canceled if they are not too far along within transit. The settings menu will bring up a screen displaying all the settings that can be changed within the CTTS menu for the user to mess around with. The final button is for the management staff to use which allows them to measure the amount of inventory being used such as containers, ships, and crew so that they can manage their staff wisely and address any situations where there isn't enough inventory for orders being made. The buttons on the ships are also clickable displaying the status of the sensor for weather, sensor, temperature, humidity, and battery. The refresh rate of the system will be around 1 minute making it so the system is not overclocking with too much feedback being received but so it can also stay up to date at all times with the time and how the status of all the alarms.

The general style guide of the whole system is blue with a darker shade being used to distinguish the menu/weather forecast from the dashboard. The buttons on the ships are the most notable colors within the system and are used to show the status of the different sensors throughout shipment.

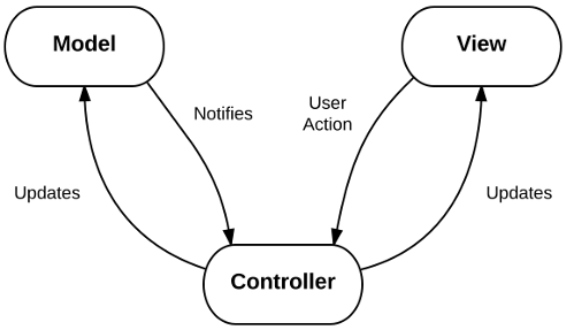
Green Meaning: All Good

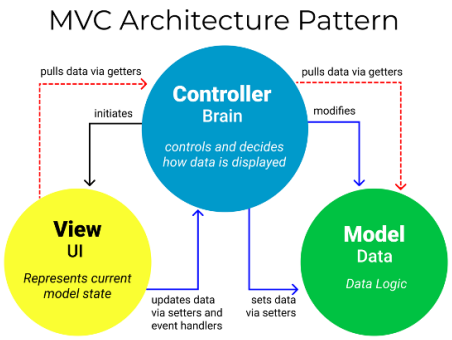
Yellow Meaning: One of the systems is beginning to have an issue

Orange Meaning: The issue on the system is beginning to become serious and needs attention
Red Meaning: Emergency this needs to be addressed now with an alarm

The colors on the arrows on the ships are also important seeing as how a green arrow shows if it is ahead of schedule and a red arrow shows if it is behind schedule. This can also be seen similarly displayed only in writing within the CTTS Menu where the On Time and etc are colored to show how the delivery is doing throughout transit. There is also a distinct orange distinguishing the sign-up from the sign-in so that they do not get mixed up and orange is the complementary color of blue. The font that was used throughout the whole thig was times new roman to make it clean and readable while also looking professional.

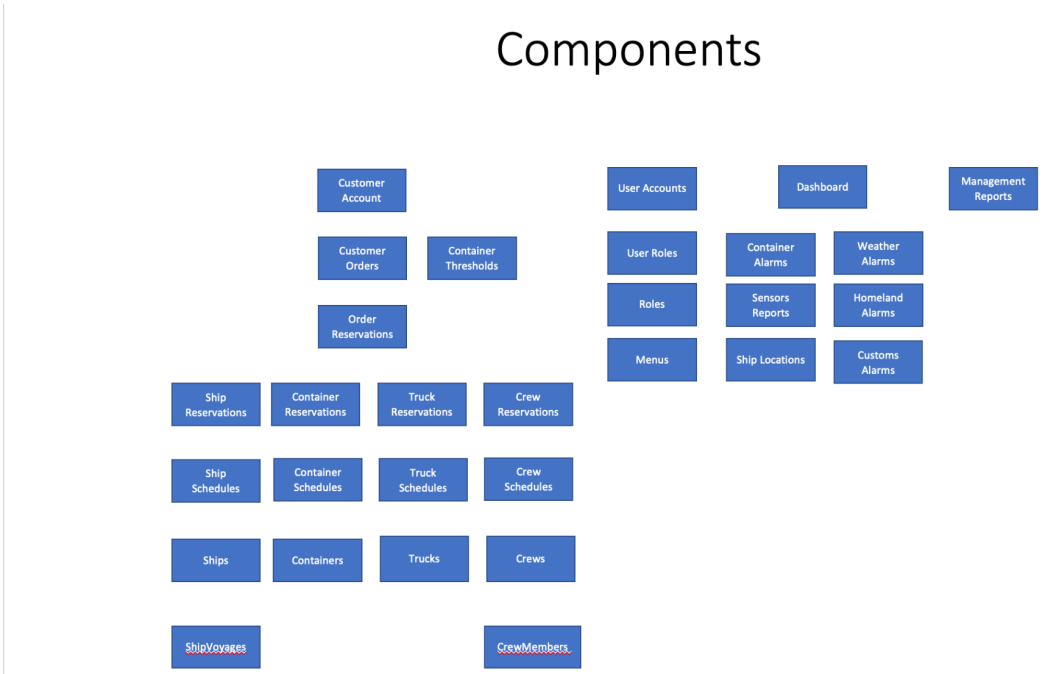
To describe the MVC or the Model View Controller I first need to explain what exactly this is. The MVC can be generally described as the framework of the User interface which can be broken down into The model, the view, and the controller, The model refers to the data access or the data model. The view refers to the user presentation layer and UI logic. The Controller refers to the services and business logic between the model and the view. The Model View Controller is followed within this system by first ensuring that the controller is the head of operations within the CTTS architecture. The view of the UI will then pull data via getters that will be sent to the controller so that it can be sent to the Model to store the data logically. The view at this time will also be updated via setters similar to the getters which will set the data being sent to the model layer of the data logic. The controller essentially initiates the process of the view which will then modify the data logic after the user enters info into the UI.





6.3 Services Layer (Given)

The service layer will be provided by a number of components. Each component will offer a set of services through defined endpoints (e.g., interface) The following is the high-level Component Diagram for the Services Layer.

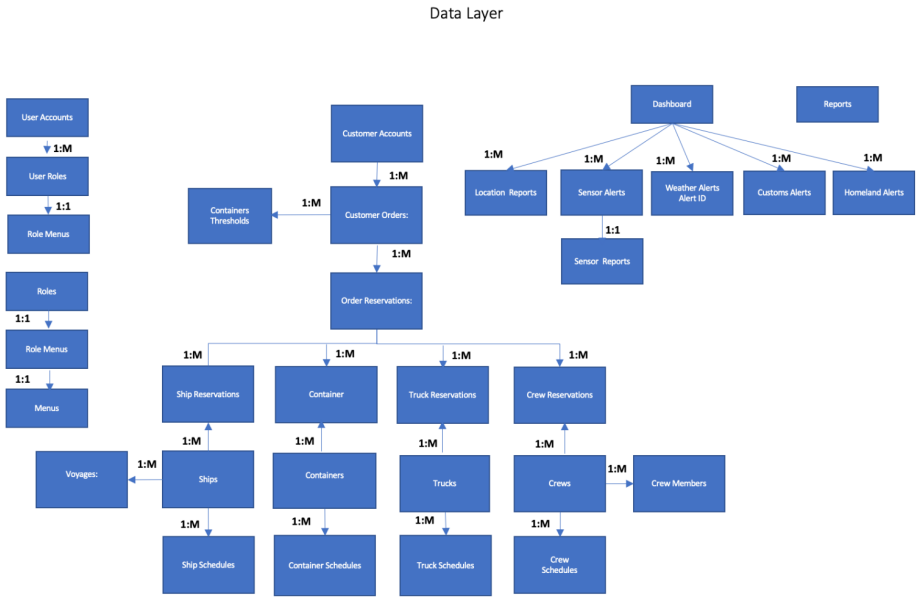


6.4 Data Access Layer (Given)

- The data access layer objects will follow a naming scheme of *object_name*DAO, for example, CustomerAccountsDAO. Each DAO will provide CRUD operations along with DOAs providing more complex data access operations.

6.5 Data Layer (Given)

The following diagrams describe the CTSS data layer.



6.6 Resource Naming Scheme (5 pts)

- The following CTTS resource naming schemes will act as a development guideline when defining resource endpoints.

Customer_Accounts
Parameters:
https://www.CTTS.com/customer_accounts/{AccountNumber}
Query Strings: https://www.CTTS.com/customer_accounts/653?CustName=Will_David

Orders
Parameters:
https://www.CTTS.com/orders/{CustomerID}/{OrderNumber}
Query Strings:
https://www.CTTS.com/orders//6532/7658?CustomerID=12976

Container_Thresholds
Parameters:
https://www.CTTS.com/container_thresholds/{OrderNumber}/{ContainerNumber}/{ContainerID}
Query Strings:
https://www.CTTS.com/container_thresholds/7659/643/58?OrderNumber=7658

Order_Reservations
Parameters:
https://www.CTTS.com/order_reservations{OrderNumber}/{ReservatioNumber}
Query Strings:
https://www.CTTS.com/order_reservations/5439/984?ReservationStatus=En_Route

Ship_Reservations
Parameters:
https://www.CTTS.com/ship_reservations/{ReservatioNumber}/{ShipVoyageID}
Query Strings:
https://www.CTTS.com/ship_reservations/873/865?ReservationNumber=127

Container_Reservations
Parameters:
https://www.CTTS.com/container_reservations/{ContainerID}/{ReservatioNumber}
Query Strings:
https://www.CTTS.com/container_reservations/763/384?PickUpTime=16:30

Truck_Reservations
Parameters:
https://www.CTTS.com/truck_reservations/{TruckID}/{ContainerID}/{ReservatioNumber}
Query Strings:
https://www.CTTS.com/truck_reservations/175/234/136?ReservationStatus=Delivered

Crew_Reservations
Parameters:

https://www.CTTS.com/crew_reservations/{CrewID}/{ContainerID}/{ReservatioNumber}
Query Strings:
https://www.CTTS.com/crew_reservations/328/234/123?StartTime=9:00

Ships
Parameters:
https://www.CTTS.com/ships/{ShipID}
Query Strings:
https://www.CTTS.com/ships/34?MaxContainers=5000

Containers
Parameters:
https://www.CTTS.com/containers/{ContainerID}
Query Strings:
https://www.CTTS.com/containers/1265?ContainerSize=Large

Trucks
Parameters:
https://www.CTTS.com/trucks/{TruckID}
Query Strings:
https://www.CTTS.com/trucks/421?DriverName=Tom_Drew

Crews
Parameters:
https://www.CTTS.com/crews{CrewID}
Query Strings:
https://www.CTTS.com/crews/45?CrewName=Alpha

CrewMembers
Parameters:
https://www.CTTS.com/CrewMembers/{CrewID}
Query Strings:
https://www.CTTS.com/CrewMembers/701?MemberName=James_Roe

Ship_Schedules
Parameters:
https://www.CTTS.com/ship_schedules/{ShipID}
Query Strings:
https://www.CTTS.com/ship_schedules/43?Status=On_Dock

ShipVoyages
Parameters:
https://www.CTTS.com/ShipVoyages{ShipVoyageID}
Query Strings:
https://www.CTTS.com/ShipVoyages/145?ShipID=34

Container_Schedules

Parameters:
https://www.CTTS.com/container_schedules/{ContainerID}
Query Strings:
https://www.CTTS.com/container_schedules/1265?Status=Reserved

Truck_Schedules
Parameters:
https://www.CTTS.com/truck_schedules/{TruckID}
Query Strings:
https://www.CTTS.com/truck_schedules164?Status=Available

Crew_Schedules
Parameters: https://www.CTTS.com/crew_schedules/{CrewID}
Query Strings:
https://www.CTTS.com/crew_schedules/2847?Status=Reserved

Dashboard
Parameters:
https://www.CTTS.com/dashboard/{DashboardViewID}
Query Strings:
https://www.CTTS.com/dashboard/2?ShowLastUpdateTime=6:30

ShipLocations
Parameters:
https://www.CTTS.com/ShipLocations{ShipID}
Query Strings:
https://www.CTTS.com/ShipLocations/54?TimeStamp=4:30

ContainerAlarms
Parameters:
https://www.CTTS.com/ContainerAlarms/{ShipID}/{ContainerID}/{AlarmID}
Query Strings:
https://www.CTTS.com/ContainerAlarms/12/4998/4?AlarmType=Container

WeatherAlarms
Parameters:
https://www.CTTS.com/WeatherAlarms{ShipID}/{AlarmID}
Query Strings:
https://www.CTTS.com/WeatherAlarms/23/6?AlarmTime=4:32

CustomsAlarms
Parameters:
https://www.CTTS.com/CustomsAlarms{ShipID}/{AlarmID}
Query Strings:
https://www.CTTS.com/CustomsAlarms/17/7?AlarmType=Customs

HomelandAlarms

Parameters:
https://www.CTTS.com/HomelandAlarms{ShipID}/{AlarmID}
Query Strings:
https://www.CTTS.com/HomelandAlarms/16/8?AlarmType=Homeland

UserAccounts
Parameters:
https://www.CTTS.com/UserAccounts{Username}
Query Strings:
https://www.CTTS.com/UserAccounts/JoeDorsey34?User_Name=Joe_Dorsey

Roles
Parameters:
https://www.CTTS.com/Roles{RoleID}
Query Strings:
https://www.CTTS.com/Roles/536?RoleName=Reservation_Specialist

UserRoles
Parameters:
https://www.CTTS.com/UserRoles{UserName}
Query Strings:
https://www.CTTS.com/UserRoles/Joe_Dornel?=RoleID=478

Menus
Parameters:
https://www.CTTS.com/Menus{MenuID}
Query Strings:
https://www.CTTS.com/Menus/25?MenuItem=Create_Order

RoleMenus
Parameters:
https://www.CTTS.com/RoleMenus{RoleID}
Query Strings:
https://www.CTTS.com/RoleMenus/437?Tempurature=68

SensorReports
Parameters:
https://www.CTTS.com/SensorReports{ShipID}/{ContainerID}
Query Strings:
https://www.CTTS.com/SensorReports/59/654?MenuID=29

7 Detailed Design

7.1 Database Tables (given)

The following are the CTTS tables.

Customer_Accounts

- **AccountNumber**
- CustName
- CustomerID
- CustAddress
- CustPhone
- AccountStatus

Orders

- **CustomerID**
- **OrderNumber**
- Orderstatus
- ShipByDate
- ArriveByDate
- From
- To
- Number_Containers

Container_Thresholds

- **OrderNumber**
- **ContainerNumber**
- **ContinerID**
- MinTemp
- MaxTemp
- MinHumidity
- MaxHumidity

Order_Reservations

- **OrderNumber**
- **ReversationNumber**
- ReservationStatus

Ship_Reservations

- **ReservationNumber**
- **ShipVoyageID**
- ReservationStatus

Container_Reservations

- **ContainerID**
- **ReservationNumber**
- PickUpLocation
- PickupTime
- DropOffLocation
- DropOffTime
- ReservationSatus

Truck_Reservations

- **TruckID**
- **ContainerID**
- **ReservationNumber**
- StartTime
- EndTime
- ReservationStatus

Crew_Reservations

- **CrewID**
- **ContainerID**
- **ReservationNumber**
- StartTime
- EndTime
- ReservationStatus

Ships

- **ShipID**
- ShipInfo – could be several fields such as type, model
- MaxContainers
- Contacts
- Status – active, inactive

Containers

- **ContainerID**
- ContainerType
- Containersize
- InServiceDate

Trucks

- **TruckID**
- DriverName
- DriverPhone
- DriverLicenseNumber
- LicensePlate
- ContainerSize

Crews

- **CrewID**
- CrewName
- CrewType

CrewMembers

- **CrewID**
- **MemberName**
- MemberPhone

Ship_Schedules

- **ShipID**
- **StartTime**
- **EndTime**
- Status – voyage, at dock, under maintenance

ShipVoyages

- **ShipVoageID**
- ShipID
- From
- To
- LoadingTime
- DepartureTime
- ArrivalTime
- UnloadingTime
- MaxContainers
- NumbContainers

Container_Schedules

- **ContainerID**
- **StartTime**
- **EndTime**
- Status – available, reserved, under maintenance

Truck_Schedules

- **TruckID**
- **StartTime**
- **EndTime**
- Status – available, reserved, under maintenance

Crew_Schedules

- **CrewID**
- **StartTime**
- **EndTime**
- Status– available, reserved, not available

DashBoard

- **DashBoardViewID**
- ShowShipSpeed
- ShowLastUpdateTime
- ShowPortStatus
- RefreshRate

ShipsLocations

- **ShipID**
- **TimeStamp**
- GPS

ContainerAlarms

- **ShipID**
- **ContainerID**
- **AlarmID**
- AlarmType
- AlarmServerity
- AlarmText
- AlarmTime

WeatherAlarms

- **ShipID**
- **AlarmID**
- AlarmType
- AlarmServerity
- AlarmText
- AlarmTime

CustomsAlarms

- **ShipID**
- **AlarmID**
- AlarmType
- AlarmServerity
- AlarmText
- AlarmTime

HomeLandAlarms

- **ShipID**
- **AlarmID**
- AlarmType
- AlarmServerity
- AlarmText
- AlarmTime

UserAccounts

- **UserName**
- Password
- User Name
- User Contact_Phone

Roles

- **RoleID**
- RoleName
- RoleDescription

UserRoles

- **UserName**
- **RoleID**

Menus

- **MenuID**
- MenuItem

RoleMenus

- **RoleID**
- **MenuID**

SensorReports

- **TimeStamp**
- **ShipID**
- **ContainerID**
- Temperature
- Humidity
- Battery

7.2 Data Access Objects (10 pts)

Each database table has a corresponding data access object (DAO) that provides CRUD operations as well as more complex capabilities. CRUD operations will follow the naming format:

- *object_name*DAO.create() – data in the body of the message
- *object_name*DAO.get(*ID*)
- *object_name*DAO.update(*ID*) – data in body of message
- *object_name*DAO.delete(*ID*)

The following DOAs provide additional capabilities beyond the CRUD operations.

7.2.1 CustomerAccounts

- find_customer_accounts_search(name/value search criteria) returns a list of customer accounts using a specific search criteria

- find_customer_accounts_number(AccountNumber) - used to return the list of accounts associated with their AccountNumbers
- find_customer_accounts_name(CustName) - used to return the customer names associated with accounts
- find_customer_accounts_address(CustAddress) - used to return the addresses associated with accounts
- find_customer_accounts_ID(CustomerID) - used to return the CustomerID associated with accounts
- find_customer_accounts_phone(CustPhone) - used to return the customer phone numbers associated with accounts
- find_customer_accounts_status(AccountStatus) - used to return the account status associated with each account

7.2.2 Orders

- find_orders_number(OrderNumber) - used to return the list of orders associated with their OrderNumbers
- find_orders_customer(CustomerID) - used to return the list orders associated with their CustomerID
- find_orders_containers(NumberContainers) - used to return the amount of containers needed for the orders that are processed
- find_orders_arrival(ArrivalByDate) - used to return the arrival date of each of the orders that have been processed
- find_orders_ship(ShipByDate) - used to return the ship date of each of the orders that have been processed
- find_orders_status(OrderStatus) - used to return all of the orders and their order statuses so that delivery status can be determined for the orders
- find_orders_search(search criteria) - used to return the orders using a specific search criteria
- find_orders_start(From) - used to return the orders that are coming from a specific location such as Japan
- find_orders_destination(To) - used to return the orders that are heading to a specific location such as Baltimore

7.2.3 Containers_Thresholds

- find_container_thresholds_order_number(OrderNumber) - used to return the list of container thresholds associated with their OrderNumbers

- find_container_thresholds_container_number(ContainerNumber) - used to return the list container thresholds associated with their container number

- find_container_thresholds_ID(ContainerID) - used to return the container thresholds associated with container IDs

- find_container_thresholds_minTemp(MinTemp) - used to return the minimum temperature associated with different container thresholds

- find_container_thresholds_maxTemp(MaxTemp) - used to return the maximum tempurature associated with different container thresholds

- find_container_thresholds_minHumidity(MinHumidity) - used to return the minimum humidity associated with different container thresholds

- find_container_thresholds_maxHumidity(MaxHumidity) - used to return the maximum humidity associated with different container thresholds

- find_container_thresholds_search(search criteria) - used to return the container thresholds using a specific search criteria

7.2.4 Order reservations

- find_order_reservations_number(OrderNumber) - used to return the list of order reservations associated with their OrderNumbers

- find_order_reservations_reservation_number(ReservationNumber) - used to return the list of ship reservations associated with their ReservationNumbers

- find_ship_reservations_reservation_status(ReservationStatus) - used to return the list of ship reservations associated with their ReservationStatus

- find_order_reservations_search(search criteria) - used to return the order reservations using a specific search criteria

7.2.5 Ship Reservations

- find_ship_reservations_ID(ShipVoyageID) - used to return the list of ship reservations associated with their ShipVoyageID

- find_ship_reservations_reservation_number(ReservationNumber) - used to return the list of order reservations associated with their ReservationNumbers

- find_order_reservations_reservation_status(ReservationStatus) - used to return the list of order reservations associated with their ReservationStatus

- find_ship_reservations_search(search criteria) - used to return the ship reservations using a specific search criteria

7.2.6 Container reservations

- find_container_reservations_ID(ContainerID) - used to return the list of container reservations associated with their Container ID
- find_container_reservations_reservation_number(ReservationNumber) - used to return the list of container reservations associated with their ReservationNumbers
- find_container_reservations_reservation_status(ReservationStatus) - used to return the list of container reservations associated with their ReservationStatus
- find_container_reservations_pickUp_location(PickUpLocation) - used to return the pick up location associated with container reservations
- find_container_reservations_pickUp_time(PickUpTime) - used to return the pick up time associated with container reservations
- find_container_reservations_dropOff_location(DropOffLocation) - used to return the drop off location associated with container reservations
- find_container_reservations_dropOff_time(DropOffTime) - used to return the drop off time associated with container reservations
- find_container_reservations_search(search criteria) - used to return the container reservations using a specific search criteria

7.2.7 truck reservations

- find_truck_reservations_truck_ID(TruckID) - used to return the list of truck reservations associated with their TruckID
- find_truck_reservations_container_ID(ContainerID) - used to return the list of truck reservations associated with their Container ID
- find_truck_reservations_reservation_number(ReservationNumber) - used to return the list of truck reservations associated with their ReservationNumbers
- find_truck_reservations_reservation_status(ReservationStatus) - used to return the list of truck reservations associated with their ReservationStatus
- find_container_reservations_start(StartTime) - used to return the start time associated with truck reservations
- find_truck_reservations_end(EndTime) - used to return the end time associated with truck reservations
- find_truck_reservations_search(search criteria) - used to return the truck reservations using a specific search criteria

7.2.8 crew reservations

- find_crew_reservations_crew_ID(CrewID) - used to return the list of Crew reservations associated with their CrewID
- find_crew_reservations_container_ID(ContainerID) - used to return the list of crew reservations associated with their Container ID
- find_crew_reservations_reservation_number(ReservationNumber) - used to return the list of crew reservations associated with their ReservationNumbers
- find_crew_reservations_reservation_status(ReservationStatus) - used to return the list of crew reservations associated with their ReservationStatus
- find_crew_reservations_start(StartTime) - used to return the start time associated with crew reservations
- find_crew_reservations_end(EndTime) - used to return the end time associated with crew reservations
- find_crew_reservations_search(search criteria) - used to return the crew reservations using a specific search criteria

7.2.9 ships

- find_ships_ID(ShipID) - used to return the list of ships associated with their ShipID
- find_ships_info(ShipsInfo) - used to return the info associated with a ship such as model or type
- find_ships_max(MaxContainers) - used to return the ships max number of containers that can be stored onboard the ship
- find_ships_contacts(Contacts) - used to return the ships contacts that are associated with it
- find_ships_status(Status) - used to return the status associated with their ship
- find_ships_search(search criteria) - used to return the ships using a specific search criteria

7.2.10 containers

- find_containers_ID(ContainerID) - used to return the list of containers associated with their ContainerID
- find_containers_type(ContainerType) - used to return the container types associated with their container
- find_containers_size(ContainerSize) - used to return the container sizes that are associated with the containers in inventory
- find_containers_service_date(InServiceDate) - used to return the containers in service date that are associated with it

- find_containers_search(search criteria) - used to return the containers using a specific search criteria

7.2.11 trucks

- find_trucks_ID(TruckID) - used to return the list of trucks associated with their TruckID
- find_trucks_name(DriverName) - used to return the driver name associated with their truck
- find_trucks_phone(DriverPhone) - used to return the driver phone number associated with their truck
- find_trucks_license(DriverLicenseNumber) - used to return the driver license number associated with their truck
- find_trucks_licensePlate(LicensePlate) - used to return the license plate associated with their truck
- find_trucks_container(ContainerSize) - used to return the container size that will be used with their truck
- find_trucks_status(search criteria) - used to return the trucks using a specific search criteria

7.2.12 crews

- find_crew_ID(CrewID) - used to return the list of crews associated with their CrewID
- find_crew_name(CrewName) - used to return the nane of a crew associated with their crew such as Alpha Team
- find_crew_type(CrewType) - used to return the type of crew that is associated with their crew such as loading crew
- find_crew_search(search criteria) - used to return the crews using a specific search criteria

7.2.13 crewMembers

- find_crew_members_ID(CrewID) - used to return the list of crew members associated with their CrewID
- find_crew_members_name(MemberName) - used to return the nane of a crew member associated with their crew
- find_crew_members_phone(MemberPhone) - used to return the crew members phone number that is associated with their crew member
- find_crew_member_search(search criteria) - used to return the crew members using a specific search criteria

7.2.14 ship_schedules

- find_ship_schedules_ID(ShipID) - used to return the list of ship schedules associated with their ShipID
- find_ship_schedules_start(StartTime) - used to return the list of start times associated with their ship schedules
- find_ship_schedules_end(EndTime) - used to return the list of end times associated with their ship schedules
- find_ship_schedules_status(Status) - used to return the status associated with a ship schedule such as at dock
- find_ship_schedules_search(search criteria) - used to return the ship schedules using a specific search criteria

7.2.15 ship_voyages

- find_available_ships(*from_local, to_local, from_date,to_date,numb_containers_requrested*) – returns a list of Ships matching the search criteria
- find_ship_voyages_ID(ShipVoyagesID) - used to return the list of ship voyages associated with their ShipVoyageID
- find_ship_voyages_ship_ID(ShipID) - used to return the list of ShipIDs associated with ship voyages
- find_ship_voyages_from(From) - used to return the list from locations associated with ship voyages
- find_ship_voyages_to(To) -used to return the list to locations associated with ship voyages
- find_ship_voyages_loading(LoadingTime) - used to return the loading times associated with their ship voyages
- find_ship_voyages_departure(DepartureTime) - used to return the list of departure times associated with ship voyages
- find_ship_voyages_arrival(ArrivalTime) - used to return the list of arrival times associated with their ship voyages
- find_ship_voyages_unloading(UnloadingTime) - used to return the list of unloading times associated with their ship voyages
- find_ship_voyages_max(MaxContainers) - used to return the max number of containers associated with a ship voyage

- find_ship_voyages_number(NumbContainers) - used to return the number of containers on a ship associated with a ship voyage

- find_ship_voyages_search(search criteria) - used to return the ship voyages using a specific search criteria

7.2.16 container_schedules

- find_container_schedules_ID(ContainerID) - used to return the list of container schedules associated with their ContainerID

- find_container_schedules_start(StartTime) - used to return the list of start times associated with their container schedules

- find_container_schedules_end(EndTime) - used to return the list of end times associated with their container schedules

- find_container_schedules_status(Status) - used to return the status associated with a container schedule such as reserved

- find_container_schedules_search(search criteria) - used to return the container schedules using a specific search criteria

7.2.17 truck_schedules

- find_truck_schedules_ID(TruckID) - used to return the list of truck schedules associated with their TruckID

- find_truck_schedules_start(StartTime) - used to return the list of start times associated with their truck schedules

- find_truck_schedules_end(EndTime) - used to return the list of end times associated with their truck schedules

- find_truck_schedules_status(Status) - used to return the status associated with a truck schedule such as reserved

- find_truck_schedules_search(search criteria) - used to return the truck schedules using a specific search criteria

7.2.18 crew_schedules

- find_crew_schedules_ID(CrewID) - used to return the list of crew schedules associated with their CrewID

- find_crew_schedules_start(StartTime) - used to return the list of start times associated with their crew schedules

- find_crew_schedules_end(EndTime) - used to return the list of end times associated with their crew schedules

- find_crew_schedules_status(Status) - used to return the status associated with a crew schedule such as reserved

- find_crew_schedules_search(search criteria) - used to return the crew schedules using a specific search criteria

7.2.19 dashboard

- find_dashboard_ID(DashboardViewID) - used to return the list of dashboards associated with their DashboardViewID

- find_dashboard_speed(ShowShipSpeed) - used to return the list shipping speed associated with the dashboard

- find_dashboard_lastUpdate(ShowLastUpdateTime) - used to return the list of the last update time associated with the dashboard

- find_dashboard_port(ShowPortStatus) - used to return the port status associated with the dashboard

- find_dashboard_refresh(RefreshRate) - used to return the refresh rate associated with the dashboard

- find_dashboard_search(search criteria) - used to return the dasboard using a specific search criteria

7.2.20 Shiplocations

- find_ship_locations_ID(ShipID) - used to return the list of ship locations associated with their ShipID

- find_ship_locations_time(TimeStamp) - used to return the list of time stamps associated with their ship locations

- find_ship_locations_GPS(GPS) - used to return the list of GPSs associated with their ship locations

- find_ship_locations_search(search criteria) - used to return the ship locations using a specific search criteria

7.2.21 CcontainersAlarms

- find_container_alarms_ship_ID(ShipID) - used to return the list of container alarms associated with their ShipIDs

- find_container_alarms_container_ID(ContainerID) - used to return the list Container IDs associated with the container alarms

- find_container_alarms_alarn_ID(AlarmID) - used to return the list of the alarm ids associated with the container alarms

- find_container_alarms_alarm_type(AlarmType) - used to return the alarm type associated with the container alarms

- find_container_alarms_alarm_severity(AlarmSeverity) - used to return the alarm severity associated with the container alarms

- find_container_alarms_alarm_text(AlarmText) - used to return the alarm text associated with the container alarms

- find_container_alarms_alarm_time(AlarmTime) - used to return the alarm time associated with the container alarms

- find_container_alarms_search(search criteria) - used to return the containerAlarms using a specific search criteria

7.2.22 weatherAlarms

- find_weather_alarms_ship_ID(ShipID) - used to return the list of weather alarms associated with their ShipIDs

- find_weather_alarms_alarm_ID(AlarmID) - used to return the list of the alarm ids associated with the weather alarms

- find_weather_alarms_alarm_type(AlarmType) - used to return the alarm type associated with the weather alarms

- find_weather_alarms_alarm_severity(AlarmSeverity) - used to return the alarm severity associated with the weather alarms

- find_weather_alarms_alarm_text(AlarmText) - used to return the alarm text associated with the weather alarms

- find_weather_alarms_alarm_time(AlarmTime) - used to return the alarm time associated with the weather alarms

- find_weather_alarms_search(search criteria) - used to return the weather Alarms using a specific search criteria

7.2.23 CustomsAlarms

- find_customs_alarms_ship_ID(ShipID) - used to return the list of customs alarms associated with their ShipIDs

- find_customs_alarms_alarn_ID(AlarmID) - used to return the list of the alarm ids associated with the customs alarms

- find_customs_alarms_alarm_type(AlarmType) - used to return the alarm type associated with the customs alarms

- find_customs_alarms_alarm_severity(AlarmSeverity) - used to return the alarm severity associated with the customs alarms

- find_customs_alarms_alarm_text(AlarmText) - used to return the alarm text associated with the customs alarms

- find_customs_alarms_alarm_time(AlarmTime) - used to return the alarm time associated with the customs alarms

- find_customs_alarms_search(search criteria) - used to return the customs Alarms using a specific search criteria

7.2.24 HomelandAlarms

- find_homeland_alarms_ship_ID(ShipID) - used to return the list of homeland security alarms associated with their ShipIDs

- find_homeland_alarms_alarn_ID(AlarmID) - used to return the list of the alarm ids associated with the homeland security alarms

- find_homeland_alarms_alarm_type(AlarmType) - used to return the alarm type associated with the homeland security alarms

- find_homeland_alarms_alarm_severity(AlarmSeverity) - used to return the alarm severity associated with the homeland security alarms

- find_homeland_alarms_alarm_text(AlarmText) - used to return the alarm text associated with the homeland security alarms

- find_homeland_alarms_alarm_time(AlarmTime) - used to return the alarm time associated with the homeland security alarms

- find_homeland_alarms_search(search criteria) - used to return the homeland security Alarms using a specific search criteria

7.2.25 userAccounts

- find_userAccounts_username(UserName) - used to return the list of userAccounts associated with their UserName

- find_userAccounts_password(Password) - used to return the list Passwords associated with the userAccounts

- find_userAccounts_name(User_names) - used to return the list of the User_names associated with the userAccounts

- find_userAccounts_phone(User_contact_phone) - used to return the User_contact_phone associated with the userAccounts

- find_userAccounts_search(search criteria) - used to return the userAccounts using a specific search criteria

7.2.26 Roles

- find_roles_ID(RolesID) - used to return the list of roles associated with their RolesID
- find_roles_name(RolesName) - used to return the list role names associated with the roles
- find_roles_description(RolesDescription) - used to return the list of the role description associated with the roles
- find_roles_search(search criteria) - used to return the roles using a specific search criteria

7.2.27 userRoles

- find_userRoles_username(UserName) - used to return the list of userRoles associated with their UserName
- find_userRoles_ID(RoleID) - used to return the list RolesID associated with the userRoles
- find_userRoles_search(search criteria) - used to return the userRoles using a specific search criteria

7.2.28 menus

- find_menus_ID(MenuID) - used to return the list of dashboards associated with their MenuID
- find_menus_items(menuItems) - used to return the list menuItems associated with the Menus
- find_menus_search(search criteria) - used to return the Menus using a specific search criteria

7.2.29 RoleMenus

- find_roleMenus_role_ID(RoleID) - used to return the list of roleMenus associated with their RoleID
- find_roleMenus_menu_ID(menuID) - used to return the list menuIDs associated with the roleMenus
- find_roleMenus_search(search criteria) - used to return the roleMenus using a specific search criteria

7.2.30 SensorReports

- find_sensorReports_ship_ID(ShipID) - used to return the list of sensorReports associated with their ShipID
- find_sensorReports_container_ID(ContainerID) - used to return the list of sensorReports associated with their ContainerID
- find_sensorReports_ID(TimeStamp) - used to return the list of the time stamps associated with the sensorReports

- find_sensorReports_temp(Tempurature) - used to return the temperatures associated with the sensorReports

- find_sensorReports_humidity(Humidity) - used to return the humidities associated with the sensorReports

- find_sensorReports_battery(Battery) - used to return the battery levels associated with the sensorReports

- find_sensorReports_search(search criteria) - used to return the sensorReports using a specific search criteria

7.3 Resources/Components Descriptions

Describe the resources and data access using the table format used in CustomerAccounts section below.

7.3.1 CustomerAccounts

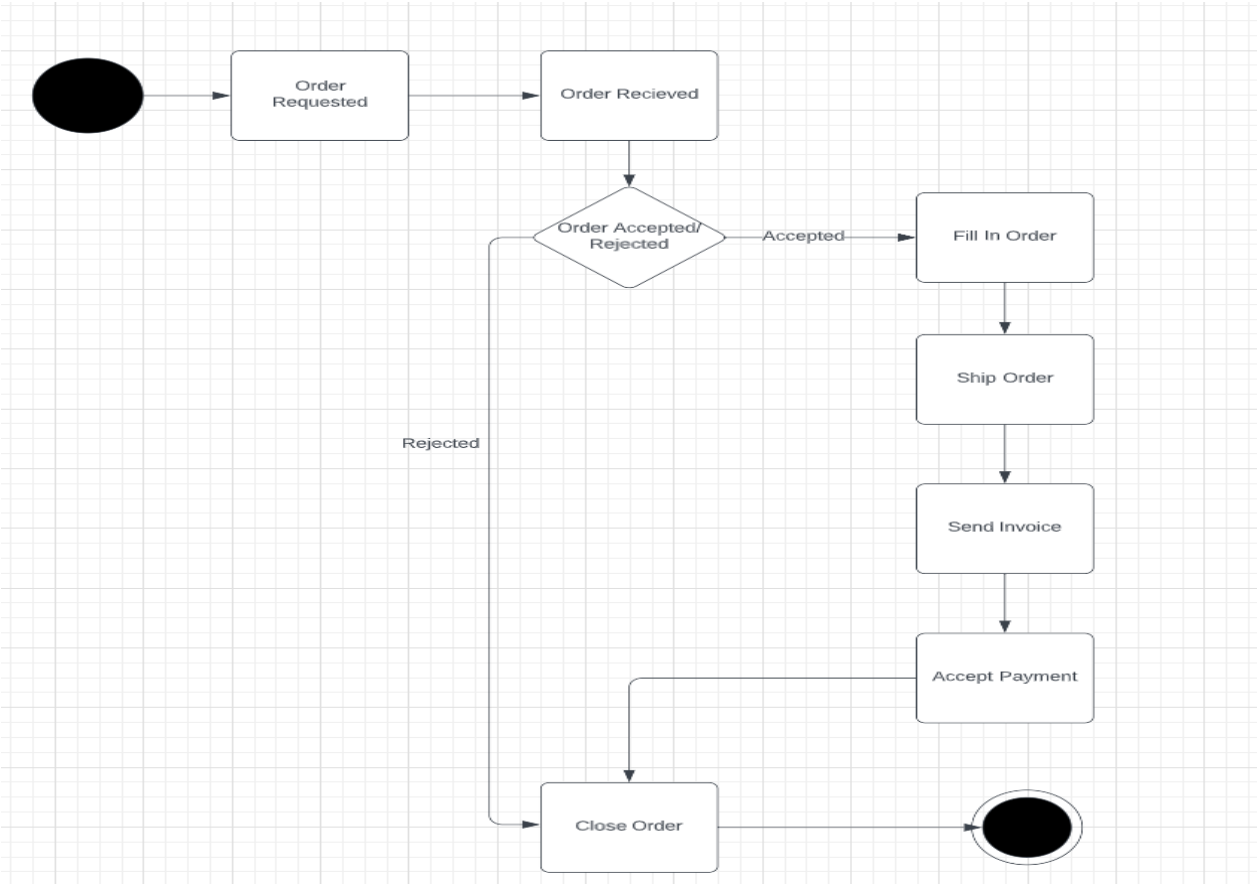
Description: Resource-providing capabilities to maintain customer account information.

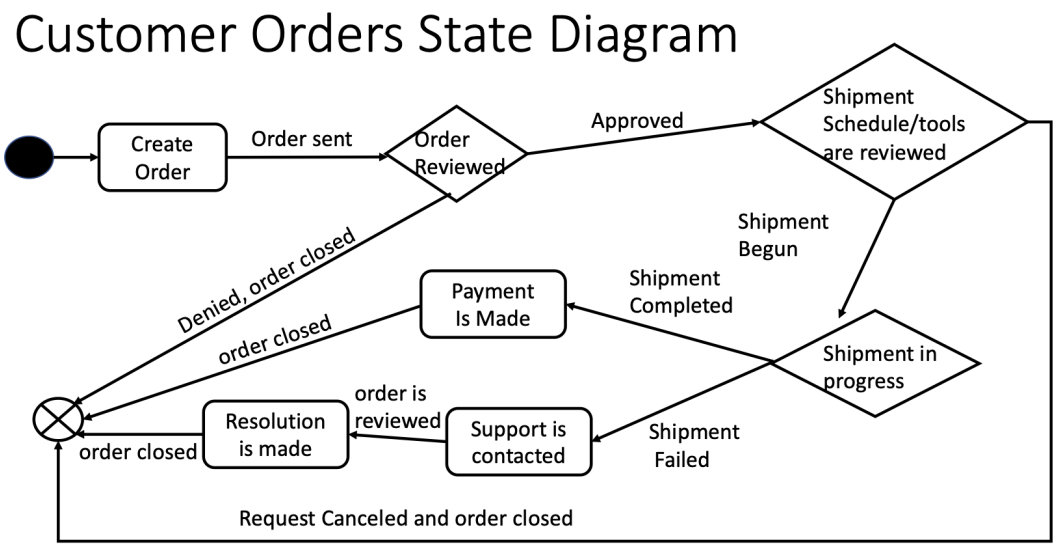
Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/CustomerAccounts	create_ customer_account	CustomerAccountsDAO create_customer_account (Account Details)	Account Number
GET	/CustomerAccoutns/*{AccountNumber}*	get_ customer_account	CustomerAccountsDAO read_customer_account (AccountNumber)	Account Details
PUT	/CustomerAccoutns/*{AccountNumber}*	update_ customer_account	CustomerAccountsDAO update_customer_account (Account Details)	Return code only
DELETE	/CustomerAccoutns/*{AccountNumber}*	delete_ customer_account	CustomerAccountsDAO delete_customer_account (AccountNumber)	Return code only
GET	/CustomerAccounts?{Attribute}={Value}	find_ customer_accounts	CustomerAccountsDAO find_customer_accounts (*name/value search criteria)*	Array of Account Details

7.3.2 Customer Orders – Provide a UML activity diagram for taking an order. Provide a UML state diagram showing the states and transitions of order (10 points - 5 points per diagram)

Description: Resource-providing capabilities to maintain customer order information. Orders have a defined set of states. The following state diagram describes the state and state transitions.





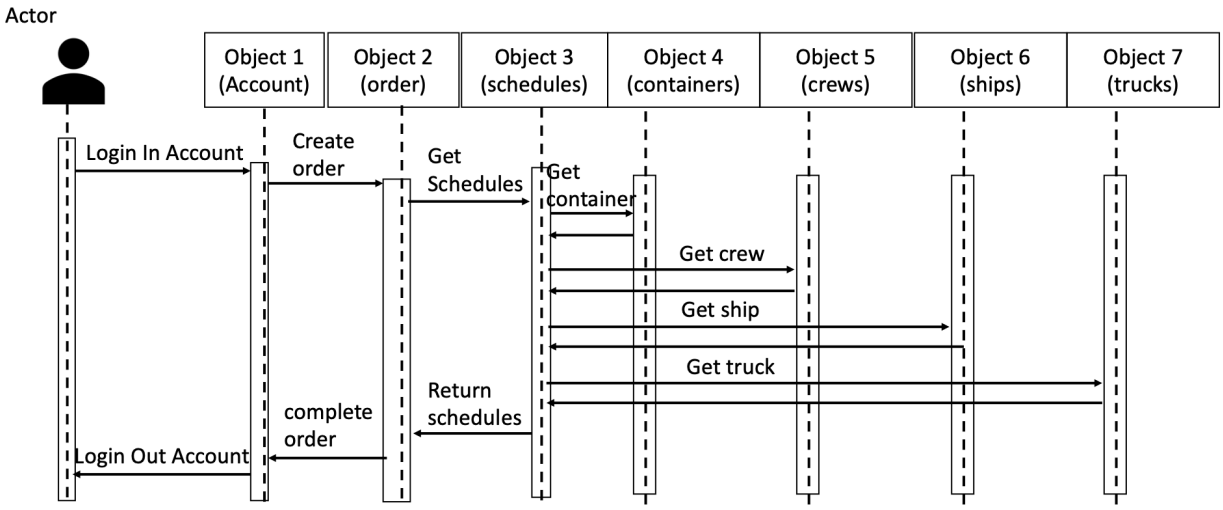
Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/orders	create_order	ordersDAOcreate_order (Order Details)	OrderNumber
GET	/orders/*{OrderNumber}*	get_order	ordersDAO read_orders (OrderNumber)	Order Details
PUT	/orders/*{OrderNumber}*	update_orders	ordersDAO update_orders (Order Details)	Return code only
DELETE	/orders/*{OrderNumber}*	delete_orders	ordersDAO delete_orders (OrderNumber)	Return code only
GET	/orders?{Attribute}={Value}	find_orders	ordersDAO find_orders(*name/value search criteria)*	Array of Order Details

7.3.3 Order Reservations - Provide a UML a sequence diagram for reserving a ship, containers, trucks, and crews for an order (5 points for diagram)

Description: Resource-providing capabilities to maintain customer making a reservation for a customers order. Orders have a defined set of states. The following sequence diagram describes the state and state transitions.

Order Reservation Sequence Diagram



Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/order_reservations	create_order_reservations	order_reservationsDAOcreate_order_reservations (ReservationNumber)	OrderNumber
GET	/order_reservations/*{OrderNumber}*	get_order_reservations	order_reservationsDAO read_order_reservations (OrderNumber)	ReservationNumber

PUT	/order_reservations/*{OrderNumber}*	update_order_reservations	order_reservationsDAO update_order_reservations (OrderNumber)	Return code only
DELETE	/order_reservations/*{OrderNumber}*	delete_order_reservations	order_reservationsDAO delete_order_reservations (OrderNumber)	Return code only
GET	/order_reservations?{Attribute}={Value}	find_order_reservations	order_reservationsDAO find_order_reservations(*name/value search criteria)*	Array of Reservation Details

7.3.4 Customer Bill

Description: Resource-providing capabilities to maintain the ability to send customers their bill prior to the service being provided.

Resource End Points and Data Access Objects:

Method	**End Point**	**API**	**Data Access Objects: Methods**	**Outputs**
POST	/customer_bill	create_customer_bill	customer_billDAOcreate_customer_bill (Bill Details)	OrderNumber
GET	/customer_bill/*{OrderNumber}*	get_customer_bill	customer_billDAO read_customer_bills (OrderNumber)	Bill Details
PUT	/customer_bill/*{OrderNumber}*	update_customer_bill	customer_billDAO update_customer_bills (Bill Details)	Return code only
DELETE	/customer_bill/*{OrderNumber}*	delete_customer_bill	customer_billDAO delete_customer_bill (OrderNumber)	Return code only
GET	/customer_bill?{Attribute}={Value}	find_customer_bill	customer_billDAO find_customer_bill(*name/value search criteria)*	Array of Bill Details

7.3.5 Ships

Description: Resource-providing capabilities to maintain the ability to obtain the information regarding any ship that is registered under the Shipping X company

Resource End Points and Data Access Objects:

Method	**End Point**	**API**	**Data Access Objects: Methods**	**Outputs**
POST	/ships	create_ships	shipsDAOcreate_ships (shipInfos)	ShipID
GET	/ships/*{ShipID}*	get_ships	shipsDAO read_ships (ShipID)	ShipInfo
PUT	/ships/*{ShipID}*	update_ships	shipsDAO update_ships (ShipIndo)	Return code only
DELETE	/ships/*{ShipID}*	delete_ships	shipsDAO delete_ships (ShipID)	Return code only
GET	/ships?{Attribute}={Value}	find_ships	shipsDAO find_shipss(*name/value search criteria)*	Array of Ship Details

7.3.6 Containers

Description: Resource-providing capabilities to maintain the ability to obtain the information regarding any container that is registered under the Shipping X company

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/containers	create_containers	containersDAOcreate_containers (container Details)	containersID
GET	/containers/*{containersID}*	get_containers	containersDAO read_containers (containersID)	containers Details
PUT	/orders/*{containersID}*	update_containers	containersDAO update_container (containers Details)	Return code only
DELETE	/orders/*{containersID}*	delete_containers	containersDAO delete_containers (containersID)	Return code only
GET	/containers?{Attribute}={Value}	find_containers	containersDAO find_containers(*name/value search criteria)*	Array of containers Details

7.3.7 Trucks

Description: Resource-providing capabilities to maintain the ability to obtain the information regarding any truck that is registered under the Shipping X company

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/trucks	create_trucks	trucksDAOcreate_trucks (truck Details)	trucksID
GET	/trucks/*{truckID}*	get_trucks	trucksDAO read_trucks (truckID)	truck Details
PUT	/trucks/*{truckID}*	update_trucks	trucksDAO update_trucks (truck Details)	Return code only
DELETE	/trucks/*{truckID}*	delete_trucks	trucksDAO delete_trucks (trucksID)	Return code only
GET	/trucks?{Attribute}={Value}	find_trucks	trucksDAO find_trucks(*name/value search criteria)*	Array of truck Details

7.3.8 Crews

Description: Resource-providing capabilities to maintain the ability to obtain the information regarding any crew that is registered under the Shipping X company

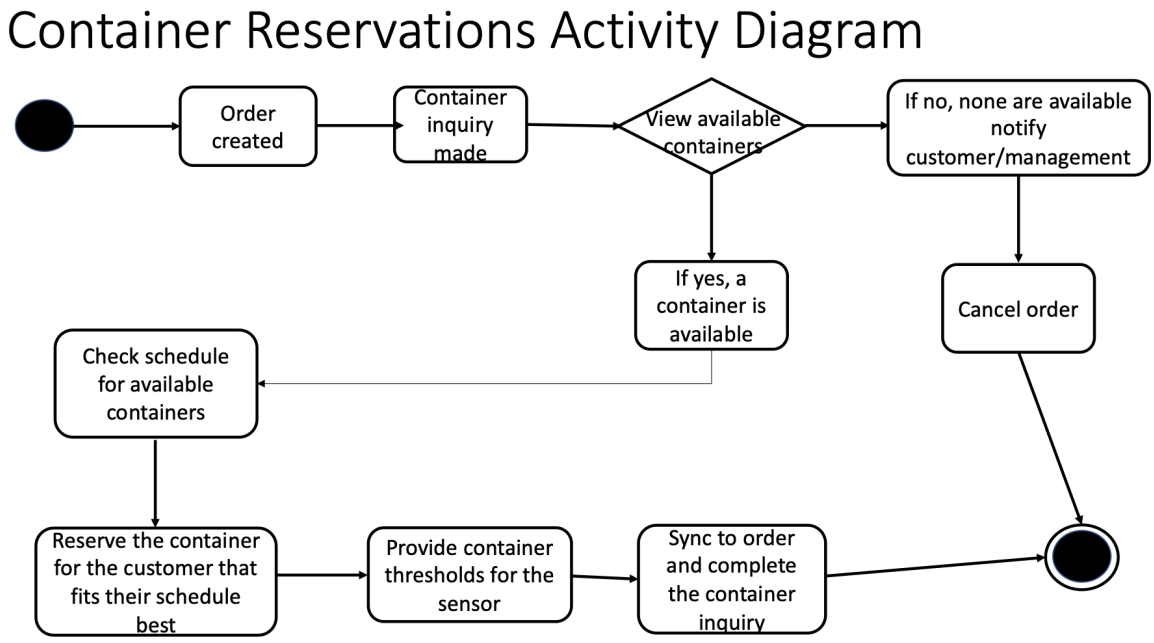
Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/crews	create_crews	crewsDAOcreate_crews (crews Details)	crewsID
GET	/crews/*{crewsID}*	get_crews	crewsDAO read_crews	crews details

			(crewsID)	
PUT	/crews/{crewsID}	update_crews	crewsDAO update_crews (crews Details)	Return code only
DELETE	/crews/{crewsID}	delete_crews	crewsDAO delete_crews (crewsID)	Return code only
GET	/crews?{Attribute}={Value}	find_crews	crewsDAO find_crews(*name/value search criteria)*	Array of crew Details

7.3.9 Container Reservations – Provide a UML activity diagram that includes: (5 points)

Description: Resource-providing capabilities to maintain customer making a reservation for a customers container that is needed for an order. Containers have a defined set of states. The following activity diagram describes the state and state transitions.



Resource End Points and Data Access Objects:

Method	**End Point**	**API**	**Data Access Objects: Methods**	**Outputs**
POST	/container_reservations	create_container_reservations	container_reservationsDAOcreate_container_reservations (ReservationNumber)	containerID
GET	/container_reservations/{ContainerID}	get_container_reservations	container_reservationsDAO read_container_reservations (containerID)	ReservationNumber
PUT	/container_reservations/{ContainerID}	update_container_reservations	container_reservationsDAO update_container_reservations	Return code only

			(ReservationNumber)	
DELETE	/container_reservations/{ ContainerID}	delete_container_reservations	container_reservationsDAO delete_container_reservations (containerID)	Return code only
GET	/container_reservations?{ Attribute}={Value}	find_container_reservations	container_reservationsDAO find_container_reservations(*name/value search criteria)*	Array of Reservation Details

7.3.10 Ship Reservations

Description: Resource-providing capabilities to maintain a customer's reservation is made for an order to ensure there is space on a ship to deliver the package. ships have a defined set of states.

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/ship_reservations	create_ship_reservations	ship_reservationsDAOcreate_ship_reservations {ReservationNumber)	ShipVoyageID
GET	/ship_reservations/{Ship VoyageID}	get_ship_reservations	ship_reservationsDAO read_ship_reservations (ShipVoyageID)	ReservationNumber
PUT	/ship_reservations/{Ship VoyageID}	update_ship_reservations	ship_reservationsDAO update_ship_reservations (ReservationNumber)	Return code only
DELETE	/ship_reservations/{Ship VoyageID}	delete_ship_reservations	ship_reservationsDAO delete_ship_reservations (ShipVoyageID)	Return code only
GET	/ship_reservations?{Attribute}={Value}	find_cship_reservations	ship_reservationsDAO find_ship_reservations(*name/value search criteria)*	Array of Reservation Details

7.3.11 Truck Reservations

Description: Resource-providing capabilities to maintain a customer's reservation being made for an order to ensure there is space on a truck to deliver the package. Trucks have a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/truck_reservations	create_truck_reservations	truck_reservationssDAOcreate_truck_reservations {ReservationNumber)	TruckID
GET	/truck_reservations/{TruckID}	get_truck_reservations	truck_reservationsDAO read_truck_reservations (TruckID)	ReservationNumber
PUT	/truck_reservations/{TruckID}	update_truck_reservations	truck_reservationsDAO update_truck_reservations (ReservatioNumber)	Return code only
DELETE	/truck_reservations/{TruckID}	delete_truck_reservations	truck_reservationsDAO delete_truck_reservations (TruckID)	Return code only
GET	/truck_reservations?{Attribute}={Value}	find_truck_reservations	truck_reservationsDAO find_truck_reservations(*name/value search criteria)*	Array of Reservation Details

7.3.12 Crew Reservations

Description: Resource-providing capabilities to maintain a customer's reservation being made for an order to ensure there is a crew that will be able to load and unload the packages from the ships onto the trucks .crews have a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/crew_reservations	create_crew_reservations	crew_reservationsDAOcreate_crew_reservations{ ReservationNumber)	crewID
GET	/crew_reservations/{crew ID}	get_crew_reservations	crew_reservationsDAO read_crew_reservations (crewID)	ReservationNumber
PUT	/crew_reservations/{crew ID}	update_crew_reservations	crew_reservationsDAO update_crew_reservations (ReservatioNumber)	Return code only
DELETE	/crew_reservations/{crew ID}	delete_crew_reservations	crew_reservationsDAO delete_crew_reservations (crewID)	Return code only
GET	/crew_reservations?{Attribute}={Value}	find_crew_reservations	crew_reservationsDAO find_crew_reservations(*name/value search criteria)*	Array of Reservation Details

7.3.13 Ship Schedules

Description: Resource-providing capabilities to maintain a customer's order being processed while ensuring that the ship's schedule lines up with the customer order to make space on a ship. Ships have a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/ship_schedules	create_ship_schedules	ship_schedulesDAOcreate_ship_schedules (schedule Details)	ShipID
GET	/ship_schedules/{ShipID}	get_ship_schedules	ship_schedulesDAO read_ship_schedules (ShipID)	schedule Details
PUT	/ship_schedules/{ShipID}	update_ship_schedules	ship_schedulesDAO update_ship_schedules (schedule Details)	Return code only
DELETE	/ship_schedules/{ShipID}	delete_ship_schedules	ship_schedulesDAO delete_ship_schedules (ShipID)	Return code only
GET	/ship_schedules?{Attribute}={Value}	find_ship_schedules	ship_schedulesDAO find_ship_schedules(*name/value search criteria)*	Array of schedule Details

7.3.14 Container Schedules

Description: Resource-providing capabilities to maintain a customer's order being processed while ensuring that the containers schedule lines up with the customer order to so that there is an open container. Containers have a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/container_schedules	create_container_schedules	container_schedulesDAO create_container_schedules (schedule Details)	ContainerID
GET	/container_schedules/{ContainerID}	get_container_schedules	container_schedulesDAO read_container_schedules (ContainerID)	schedule Details
PUT	/container_schedules/{ContainerID}	update_container_schedules	container_schedulesDAO update_container_schedules (schedule Details)	Return code only
DELETE	/container_schedules/{ContainerID}	delete_container_schedules	container_schedulesDAO delete_container_schedules (ContainerID)	Return code only
GET	/container_schedules?{Attribute}={Value}	find_container_schedules	container_schedulesDAO find_container_schedules (*name/value search criteria)*	Array of schedule Details

7.3.15 Truck Schedules

Description: Resource-providing capabilities to maintain a customer's order being processed while ensuring that the truck's schedule lines up with the customer order to so that there is an open truck to deliver the order. Truck have a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/truck_schedules	create_truck_schedules\	truck_schedulesDAOcreate_crew_schedules(schedule Details)	TruckID
GET	/truck_schedules/{TruckID}	get_truck_schedules	truck_schedulesDAO read_truck_schedules (TruckID)	schedule Details
PUT	/truck_schedules/{TruckID}	update_truck_schedules	truck_schedulesDAO update_truck_schedules (schedule Details)	Return code only
DELETE	/truck_schedules/{TruckID}	delete_truck_schedules	truck_schedulesDAO delete_truck_schedules(TruckID)	Return code only
GET	/truck_schedules?{Attribute}={Value}	find_truck_schedules	container_schedulesDAO find_container_schedules (*name/value search criteria)*	Array of schedule Details

7.3.16 Crew Schedules

Description: Resource-providing capabilities to maintain a customer's order being processed while ensuring that the crews schedule lines up with the customer order to so that there is an open crew to load or unload the order. The crew has a defined set of states

Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs

POST	/crew_schedules	create_crew_schedules	crew_schedulesDAOcreate_truck_schedules(schedule Details)	CrewID
GET	/crew_schedules/{CrewID}	get_crew_schedules	crew_schedulesDAO read_crew_schedules (CrewID)	schedule Details
PUT	/crew_schedules/{CrewID}	update_crew_schedules	crew_schedulesDAO update_crew_schedules (schedule Details)	Return code only
DELETE	/crew_schedules/{CrewID}	delete_crew_schedules	crew_schedulesDAO delete_crew_schedules(CrewID)	Return code only
GET	/crew_schedules?{Attribute}={Value}	find_crew_schedules	crew_schedulesDAO find_crew_schedules(*name/value search criteria)*	Array of schedule Details

7.3.17 Weather Interface

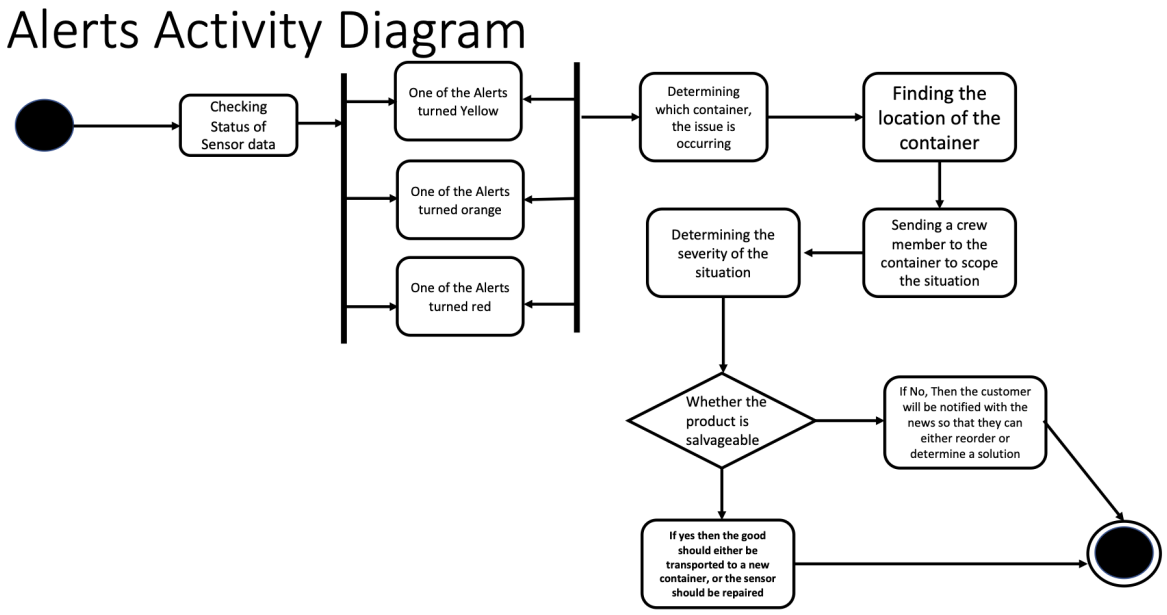
Description: Resource-providing capabilities to maintain a status of the weather on the routes that ships will be traveling so that no ships are traveling in any unnecessary risks. The weather alarm has a defined set of states

Resource End Points and Data Access Objects:

Method	**End Point**	**API**	**Data Access Objects: Methods**	**Outputs**
POST	/weather_interface	create_weather_interface	weather_interfaceDAOcreate_weather_interface (weather details)	ShipID
GET	/weather_interface/*{OrderNumber}*	get_weather_interface	weather_interfaceDAO read_weather_interface (ShipID)	Weather Details
PUT	/weather_interface/*{OrderNumber}*	update_weather_interface	weather_interfaceDAO update_weather_interface (Weather Details)	Return code only
DELETE	/weather_interface/*{OrderNumber}*	delete_weather_interface	weather_interfaceDAO delete_weather_interface (ShipID)	Return code only
GET	/weather_interface?{Attribute}={Value}	find_weather_interface	weather_interfaceDAO find_weather_interfacEs(*name/value search criteria)*	Array of Weather Details

7.3.18 Alerts – Provide a UML activity diagram for receiving contain location and sensor information and creating an alert. (10 points for diagram)

Description: Resource-providing capabilities to maintain status on any alarms so that they can be responded to in any emergency cases. The alarms have a defined set of states



Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/alerts	create_alerts	alertsDAOcreate_alerts (alarm Details)	AlarmID
GET	/alerts/{AlarmID}	get_alerts	alertsDAO read_alerts (AlarmID)	alarm Details
PUT	/alerts/{AlarmID}	update_alerts	alertsDAO update_alerts (alarm Details)	Return code only
DELETE	/alerts/{AlarmID}	delete_alerts	alertsDAO delete_alerts (AlarmID)	Return code only
GET	/alerts?{Attribute}={Value}	find_alerts	alertsDAO find_alertss(*name/value search criteria)*	Array of alarm Details

7.3.19 Monthly Reports

Description: A monthly report will be generated by using the resources and capabilities at hand to show the number of orders that have been taken and completed since the beginning of the month. This will be done by showing all of the statistics from the company including when alarms are triggered or how much money has been made by the company or issues that have been occurring through the Shipping X company. This will show all of the statistics from the dashboard also including how many customers enter the site or what could be configured to get more used within the site and so on and so forth.

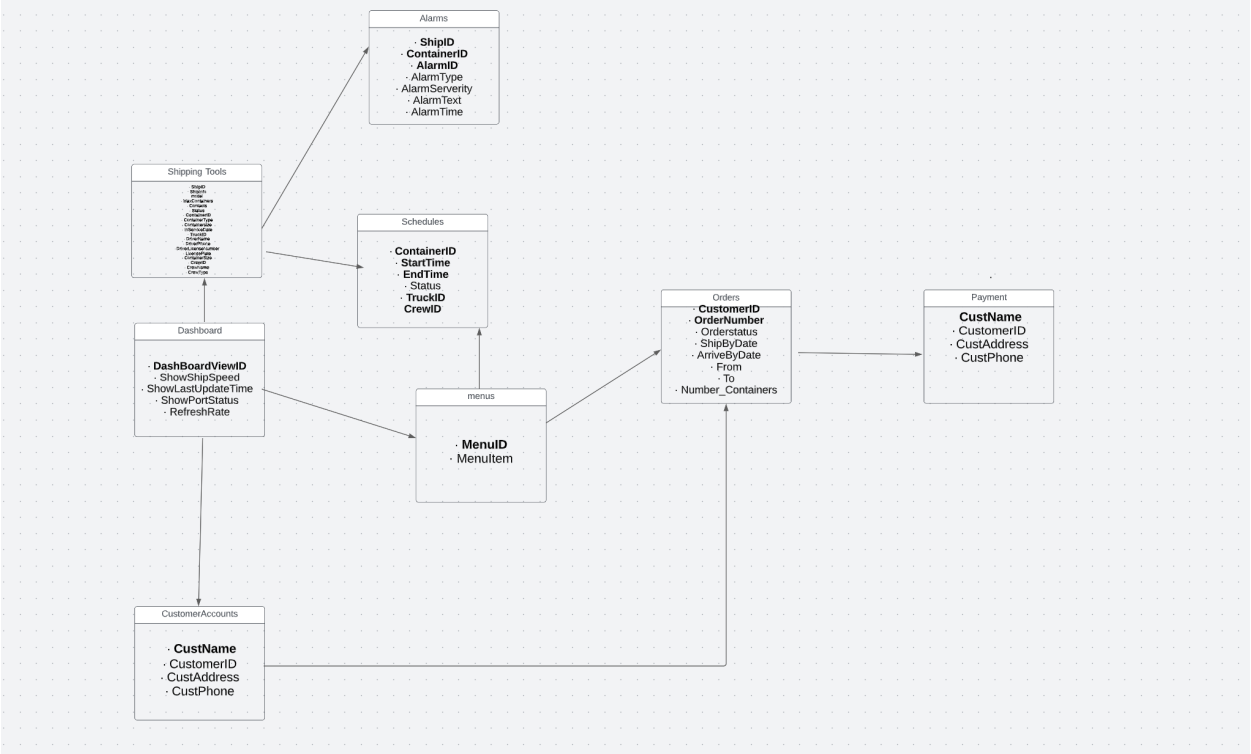
Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/monthly_reports	create_monthly_reports	ordersDAOcreate_order (monthly_reports details)	monthlyReportID

GET	/monthly_reports/{monthlyReportID}	get_monthly_reports	ordersDAO read_orders (monthlyReportID)	monthly_reports Details
PUT	/monthly_reports/{monthlyReportID}	update_monthly_reports	ordersDAO update_orders (monthly_reports details)	Return code only
DELETE	/monthly_reports/{monthlyReportID}	delete_monthly_reports	ordersDAO delete_orders (monthlyReportID)	Return code only
GET	/monthly_reports?{Attribute}={Value}	find_monthly_reports	ordersDAO find_orders(*name/value search criteria)*	Array of month report Details

7.3.20 Dashboard – Provide a UML class diagram for managing all the information on the dashboard (ships, containers, alarms, DAOs, etc.) (10 points for diagram)

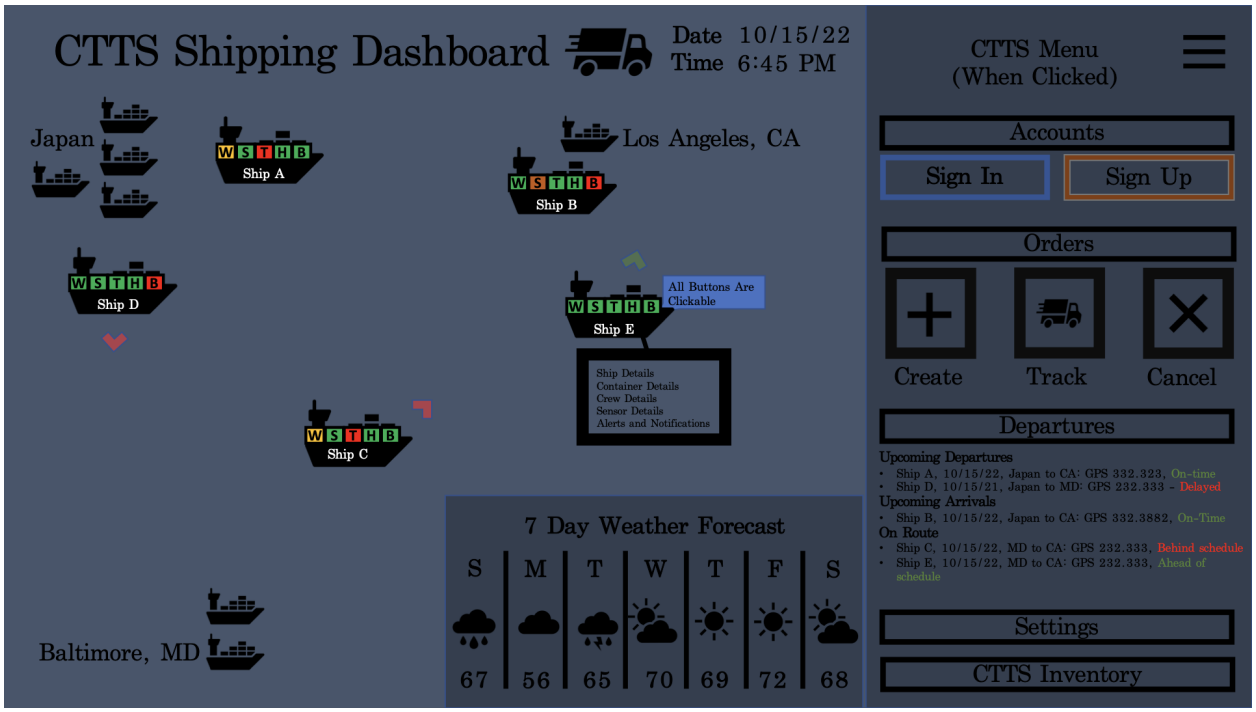
Description: The major areas of the User Interface includes the dashboard and the CTTS Menu with these two sections broken even further down into more specific user interface options. To begin with the dashboard you can begin by looking at the top of the screen which displays to the user the name of the dashboard along with a shipping logo, date, and time. Below this, you can then see the three delivery locations that Shipping X has ports for at the moment which are Japan, Los Angeles California, and Baltimore Maryland. These three locations are named on the dashboard and show the dock where ships belonging to Shipping X may be located at the time of the dashboard being opened. This will also show the ships that are en route at the moment of the CTTS dashboard being opened to allow the user to understand the route that the ships are taking and if the ships are on schedule for delivery. The ships will appear with an arrow in front of each if they are ahead of schedule or behind schedule with the green indicating they are ahead of schedule and red indicating behind schedule. The buttons on each of the boats are also clickable indicating the specific details of the alarms on the boats so that it is understood if there is a problem with one of the sensors that need to be taken care of by being shown with a colored visual. Below this, there is also a weather forecast for the next week that is shown which will allow the user to determine if their shipment might be delayed because of bad weather within the coming days. On the right side of the screen, there is a drop-down menu that is indicated by three lines. This will open up a menu condensing the left-hand side of the screen which is the dashboard and show the full extent of the CTTS Menu. Starting with the Customer Accounts, it shows the option to either sign in or sign up for an account with the CTTS dashboard service. Once done with either of these options the system will allow the user to view their details or other volatile information. After this section, there is an orders section with three buttons that may be pressed that will allow the user to create an order, track an order, or cancel an order if possible so that the user may complete tasks that they wish to have serviced on their own. Following this, there is a departures list where it will show a list of the ships that are en route at any given time indicating where they are going and the status of how the ship is doing on its delivery for example ship E is ahead of schedule. Finally, there is a settings option for users to be able to configure anything that they may need to be changed for their settings and the CTTS inventory which is mostly meant for management so that they can keep track of the number of containers, staff, ships, and etc. ensuring they have enough materials to complete any given order given to them at one time.



Resource End Points and Data Access Objects:

Method	End Point	API	Data Access Objects: Methods	Outputs
POST	/dashboard	create_dashboard	ordersDAOcreate_order (dashboard_Details)	DashboardViewID
GET	/dashboard/{DashboardViewID}	get_dashboard	ordersDAO read_orders (DashboardViewID)	dashboard Details
PUT	/dashboard/{DashboardViewID}	update_dashboard	ordersDAO update_orders (dashboard Details)	Return code only
DELETE	/dashboard/{DashboardViewID}	delete_dashboard	ordersDAO delete_orders (DashboardViewID)	Return code only
GET	/dashboard?{Attribute}={Value}	find_dashboard	ordersDAO find_orders(*name/value search criteria)*	Array of dashboard Details

7.4 User Interface (10 pts)



7.4.1 Menus and Visualizations

- Navigation
    - The main CTTS Dashboard
    - The side panel CTTS menu that can be opened or closed using the three lines
    - The clickable buttons that navigate the user to different portions of the system
        - The Sign In that brings the user to the sign in page
        - The Sign Up that brings the user to the page to sign up entering the details necessary to create an account
        - The Create Order button that will navigate the user to the page to create an order using a questionnaire/survey type response answering all the questions necessary with the information needed to create an order
        - The Track Order button which will navigate the user to a page to track an order by using their account info to find the order that they have placed displayinbg the option to click anyone which will display the status of the ship, the status of arrival such as on-time, the estimated time of delivery, the contents of the order, and the visual display of all the elements on a map
        - The Cancel order button which will navigate the user to a cancel order page will display once again the orders that are in progress similarly to the track order screen only this will show the orders that are able to be canceled and the order that are too far into transit to cancel without approval by management
        - The settings navigation button is useful for changing any settings of visuals on the page of the system such as text size and so forth.
        - The CTTS Inventory navigation button will bring the user to a new page which is useful for the manager to ensure that the supplies are enough to suffice the number of orders being received and if more needs to be added to the inventory
- Menus

- The two main screens within the CTTS
  - The CTTS Shipping Dashboard
  - The CTTS Menu

- Clickable Items
  - The specific ships when they are transported that will display all of the information on the ship
    - The buttons on the ship will also be able to be clicked which are color coded for each of the specific alarms on the sensors with color-coded distinctions of all good all the way to emergency statuses.
      - Weather Alarm
      - Sensor Alarm
      - Temperature Alarm
      - Humidity Alarm
      - Battery Alarm
        - Green Meaning: All Good
        - Yellow Meaning: One of the systems is beginning to have an issue
        - Orange Meaning: The issue on the system is beginning to become serious and needs attention
        - Red Meaning: Emergency this needs to be addressed now with an alarm
    - Then there are the buttons on the CTTS menu that have been described previously but can be explained again
      - The Sign-In Button
        - Used to navigate the user to a sign-in page
      - The Sign-Up Button
        - Used to navigate the user to a sign-up page
      - The Create Order Button
        - Used to navigate the user to a create order page
      - The Track Order Button
        - Used to navigate the user to a track order page
      - The Cancel Order Button
        - Used to navigate the user to a cancel order page
      - The Settings Button
        - Used to navigate the user to a settings page
      - The CTTS Inventory Button
        - Used to navigate the user to a CTTS Inventory page

- Images/Icons
  - Some of the images/ icons that are used throughout the page include many to display functionality, as well as label tasks ▪ Ship Icons The ship icons, are used to display to the user where the ships are throughout transport or if there are on the port, etc. These icons and images are also clickable and make it easy to interact with the ship details and sensors aboard the ship
    - Weather Icons
      - The weather sensor is located on all of the ships but the weather forecast includes using icons to demonstrate the weather for a specific day within the seven-day forecast. These will update with the weather so that the

user can determine if their delivery might be delayed because of the weather or if the management decides not to transport on a specific day

- Create Order Icon
  - The create an order icon is a plus sign indicating to the user that this is how you create an order which is a clickable icon
- Track Order Icon
  - The Track an order is an icon with a delivery truck indicating to the user that this is how you track your shipment through delivery which is also a clickable icon
- Cancel Order Icon
  - The cancel an order is an icon with an X indicating to the user that this is how you can cancel an order if it is not too far through shipment which is also a clickable icon
- Dashboard Logo Icon
  - The dashboard logo icon is located right by the name of the dashboard and is just to indicate to the user the purpose of the shipping X company which is deliveries indicated by delivery trucks

7.4.2 Dashboard

- The main CTTS Dashboard
  - The main dashboard has the blue color style with the black icons and text. There is a header text labeling it as the Shipping dashboard. Below this, there are the three ports that the Shipping X company delivers to which include Japan, Los Angeles California, and Baltimore Maryland. Within this dashboard, there is also the date and time in the top right corner and the 7-day forecast for the weather. The buttons on the ship will also be able to be clicked which are color coded for each of the specific alarms on the sensors with color-coded distinctions of all good all the way to emergency statuses.
    - Weather Alarm
    - Sensor Alarm
    - Temperature Alarm
    - Humidity Alarm
    - Battery Alarm
      - Green Meaning: All Good
      - Yellow Meaning: One of the systems is beginning to have an issue
      - Orange Meaning: The issue on the system is beginning to become serious and needs attention
      - Red Meaning: Emergency this needs to be addressed now with an alarm
- The side panel CTTS menu that can be opened or closed using the three lines
- The clickable buttons that navigate the user to different portions of the system
  - The Sign In that brings the user to the sign in page
  - The Sign Up that brings the user to the page to sign up entering the details necessary to create an account
  - The Create Order button that will navigate the user to the page to create an order using a questionnaire/survey type response answering all the questions necessary with the information needed to create an order
  - The Track Order button which will navigate the user to a page to track an order by using their account info to find the order that they have placed displayinbg the option to click anyone which will display the status of the ship, the status of arrival such as on-time, the

estimated time of delivery, the contents of the order, and the visual display of all the elements on a map

- The Cancel order button which will navigate the user to a cancel order page will display once again the orders that are in progress similarly to the track order screen only this will show the orders that are able to be canceled and the order that are too far into transit to cancel without approval by management
- The settings navigation button is useful for changing any settings of visuals on the page of the system such as text size and so forth.
- The CTTS Inventory navigation button will bring the user to a new page which is useful for the manager to ensure that the supplies are enough to suffice the number of orders being received and if more needs to be added to the inventory
- The refresh rate of the CTTS system will be completed every minute to ensure that all of the information is up to date for everyone looking at it. The staff and management need to be ankle to address any alarms that need to be taken care of and the inventory is crucial to ensure that there are not too many orders over the capacity

7.4.3 UI/Service Mapping

- Using the following table format, for each item listed in 7.4.1 and 7.4.2 specify the resource endpoints utilized. You may want to use more than one table, such as one per main menu item.

Create Account

Web Page/Menu Item/Action/Event	**HTTP Methods: Resources End Points**
Create Customer Account	POST: /CustomerAccounts
Update a Customer Account	PUT: /CustomerAccounts/*{AccountNumber}*
Find Customer Account by search attributes: Name, Phone	GET: /CustomerAccounts?{Attribute}={Value}&…

Sign In To Account

Web Page/Menu Item/Action/Event	**HTTP Methods: Resources End Points**
Sign In To Customer Account	POST: /CustomerAccounts/*{AccountNumber}*
Update a Customer Account	PUT: /CustomerAccounts/*{AccountNumber}*
Find Customer Account by search attributes: Name, Phone	GET: /CustomerAccounts?{Attribute}={Value}&…

Create Order

Web Page/Menu Item/Action/Event	**HTTP Methods: Resources End Points**
Create Order	POST: /Order
Update an Order	PUT: /Order/*{OrderNumber}*
Find Order by search attributes: OrderNumber	GET: /Order?{Attribute}={Value}&…

Track Order

Web Page/Menu Item/Action/Event	**HTTP Methods: Resources End Points**
Read An Order	GET: /Orders/{OrderNumber}
Update an Order	PUT: /Order/*{OrderNumber}*
Find Order by search attr\iibutes: OrderNumber	GET: /Order?{Attribute}={Value}&…

Cancel Order

Web Page/Menu Item/Action/Event	**HTTP Methods: Resources End Points**
Delete An Order	Delete: /Orders/{OrderNumber}
Find Order by search attr\ibutes: OrderNumber	GET: /Order?{Attribute}={Value}&…

Settings

Web Page/Menu Item/Action/Event	HTTP Methods: Resources End Points
Update a Settings	PUT: /Settings/*{AccountNumber}*
Find settingsby search attributes:	GET: /Settings?{Attribute}={Value}&…

CTTS Inventory

Web Page/Menu Item/Action/Event	HTTP Methods: Resources End Points
read CTTS Inventory	READ: /CTTS_Inventory
Update a CTTS Inventory	PUT: /CTTS_Inventory/*{AccountNumber}*
Find CTTS Inventoryt by search attributes: Name, Phone	GET: /CTTS_Inventory?{Attribute}={Value}&…

Ship Details

Web Page/Menu Item/Action/Event	HTTP Methods: Resources End Points
Create Ship Details	POST: /Ships
Update a Ship Details	PUT: /Ships/*{ShipsID}*
Read Ship Details	READ: /Ships/*{ShipsID}*
Find Ships by search attributes: Name, Phone	GET: /Ships?{Attribute}={Value}&…

Ship Icon Alarms

Web Page/Menu Item/Action/Event	HTTP Methods: Resources End Points
Create Ship Details	POST: /Ships
Update a Ship Details	PUT: /Ships/*{ShipsID}*
Read Ship Details	READ: /Ships/*{ShipsID}*
Find Ships by search attributes: Name, Phone	GET: /Ships?{Attribute}={Value}&…