UML Use Case Diagram:

There are 7 different use cases specified for this system. They include: Develop Reliable Code, Light up LED, Access Reliable Sorting, Turn Off LED, Track Phone Count, Print Labels, and Develop Sturdy/Reliable Structure. The left side of the diagram contains the primary actors which are the engineer, operator, and the outside company which is the one providing and receiving the phones. On the right side of the diagram are the supporting actors which include the framework, PCB, and LED set-up manufacturers. The other supporting actor is the software system. The engineer develops reliable code and incorporates phone count tracking into the system which affects the software system. The operator is involved in most of the use cases. They rely on the reliable structure, need the correct LED to light up to notify them of where to place the phone, turn off the LED to alert the system that the phone was placed in said box, use the push of the LED to increase the phone count of the box, and need to print shipping/ID labels for the boxes. The outside company relies on a reliable sorting method so they get back all of the devices that they provided. The framework manufacturer creates the sturdy structure. The PCB manufacturer deals with programming the LEDs as does the LED set-up manufacturer. The software system is used for developing reliable code, tracking phone count, and sending signals to print labels.

UML Domain Model:

After listing out all classes for my system, there were several I decided to keep and many others I chose to prune. After pruning, the classes left were scanner, label, storage system, LED, phones, framework manufacturer, PCB manufacturer, LED set-up manufacturer, operator, engineer, phone database, outside company, seller standards, receipt, and PCB. There are many connections in the domain model among classes such as the operator follows the seller standards, the operator issues receipts, the operator uses the scanner, and the operator prints the labels. Each class's connections involve multiplicity as 1, 1..*, or * as well as verb/verb-phrase association names. Every class also has 3 or more attributes.

UML Class Diagram:

I pruned the classes more and decided to get rid of the Seller Standards and receipt classes as they are not relevant to the storage system. I removed engineer, PCB, and all 3 manufacturer classes because they have to do with the implementation. The classes I chose to keep are operator, label, outside company, storage system, scanner, phones, phone database, and LED. I felt these were the most important classes in the system. Most classes contain attributes and methods. Multiplicity is shown between classes as is the verb/verb-phrase association names. Dependencies are shown as dotted lines and are seen between the operator class and LED button pushed, phone database, and storage system classes.

UML Sequence Diagrams:

Sequence Diagram 1 represents the Light Up LED use case.The diagram contains the operator, scanner, LED, LED button pushed, and phone database. The operator enters their credentials and is authenticated. They login and are able to use the scanner. The scanner scans the barcode and in return receives the LED location. The scanner requests the box ID and receives it from the phone database. The scanner sends alerts to the LED which then sends a

button pushed signal to the LED button pushed class. This then updates the phone's location in the phone database. Logs are created to track the movement.

Sequence Diagram 2 represents the Print Labels use case. It involves the operator, label, storage system, and outside company. The operator sends print signals to print the label. The label is attached to the box on the storage system. The operator ships the box to the outside company by using the shipping label.

Sequence Diagram 3 represents the Track Phone Count use case. It involves the operator, phone, storage system, and phone database. The operator scans the phone and receives the location it needs to go to. The phone is placed in the storage system. The box count and phone count are updated in the phone database.

UML State Diagram

The first state after the initial one is to grab a phone. After grabbing the phone, the barcode is scanned. From there, a label is printed if a new box is being started, or an LED is turned on for an existing box. If a label is printed, the next state is to turn on the LED for that label's box. Next, the phone is put in the storage box and the LED is pushed off. The box count is then adjusted. From here, if said box is not full, the state is set back to the first state of grabbing a phone. If the box is full, the box is closed then packaged with a label then shipped to the outside company. This is the final state of the system.

UML Activity Diagram

A swimlane diagram was used to create this activity diagram. The 3 lanes consist of the engineer, operator, and outside company. The engineer creates code, then programs the PCB. The operator scans the phone barcode, places a label on the box, places the phone in the box, and turns off the LED. If the box is not full, we return back to scanning a phone, but if it is full, the box gets sealed. The operator attaches the shipping label to the box then ships the box. The outside company receives the box full of refurbished phones.

UML Component Diagram

The components included in my phone sorting system are barcode, phone, scanner, operator, LED, label, phone database, storage system, box, and outside company. There are many interfaces such as: phone ID, shipping address, refurbished phones, support, storage, sorting, location, broken phones, shipping address and phone info, phone information, phone count and ID. Most components are providing the interface while another component is receiving it. For example, the phone database provides phone information for the operator and the operator provides sorting for the phones.

Cloud Deployment Diagram

I chose to use an AWS Deployment Diagram set up that includes a Virtual Private Cloud. Within the VPC, there are EC2 instances, a security group, API gateway, and lambda. Outside of the VPC, there is a Dynamo DB and an S3 bucket. The operator is symbolized by the user. The EC2 instance is used to run things virtually. The security group controls what goes in and out of the EC2 instance. The API gateway is used to access the sorting system. Lambda is in

place to run the code after the phone is scanned. Dynamo DB is a NoSQL database needed for the sorting app. The S3 bucket is used to store the phone database information.

UI:

The scanner contains a UI with phone database information. Once a phone is scanned, the location of the box is displayed on the UI.