Megan Koczur
Final Project - SWENG 837

Non-functional Requirements:

a. Define performance requirements like scalability, response time, and throughput
b. Specify security requirements like authentication, authorization, and data encryption.
c. Outline maintainability requirements like code modularity, documentation, and testing strategies.
d. Indicate any other non-functional requirements relevant to the system's success.

Problem Statement and Requirements

Business Requirements

Problem Statement:

A cell-phone repair company collects phones that need to be fixed. Once they repair them, they need an easy and reliable way to sort them so they can be sent out.

Functionalities:

1. Scan a QR code on the phone
2. Trigger light for the specific box the phone needs to be placed in
2. Place the phone in the box
3. Press the light to turn it off
4. System clears the screen and gets ready for the next scan
5. System needs to keep track of the number of phones in each box and start a new one when the box is full
6. When starting a new box, print a label for it with the name of the phones that will be going into it and the company they will be going to
7. When a box is full, print a shipping label for the box to be returned to the company

Target Users and Their Needs:

Type	**Actors**	**Goal Description**
Primary	Engineer	The engineer needs to develop a reliable sorting system based on cell phone type and company.

	Operator	Operators need a quick, easy, and reliable way to see where to place the phone.
	Outside Company	Companies requesting the system need a quick and reliable way for their phones to be fixed and sorted.
Supporting	Framework Manufacturer	This manufacturer needs to build a sturdy and reliable framework to hold all of the boxes filled with phones.
	PCB Manufacturer	This manufacturer needs to build a PCB to connect all of the different electrical pieces in the system
	LED set-up Manufacturer	This manufacturer needs to create a structure to securely hold the LED and connect the correct wires to it.
	Software System	Track the count of phones. Print labels for starting a new box and for shipping. Store information about the phones.
Offstage	Phone Users	Eventually use the phones when they are sold as refurbished.
	Resellers	Buy the phones from the above companies and resell them for a higher cost.

Business Goals:

1. Decrease the amount of time it takes to get these fixed phones out of the warehouse and back to the companies.
2. Increase revenue by returning the phones quicker.
3. Decrease the amount of mishaps that occur such as placing the phone in the wrong company's box or placing the phone in a box that holds a different type of phone.

Non-functional Requirements

Performance Requirements:

Security Requirements:

Maintainability Requirements:

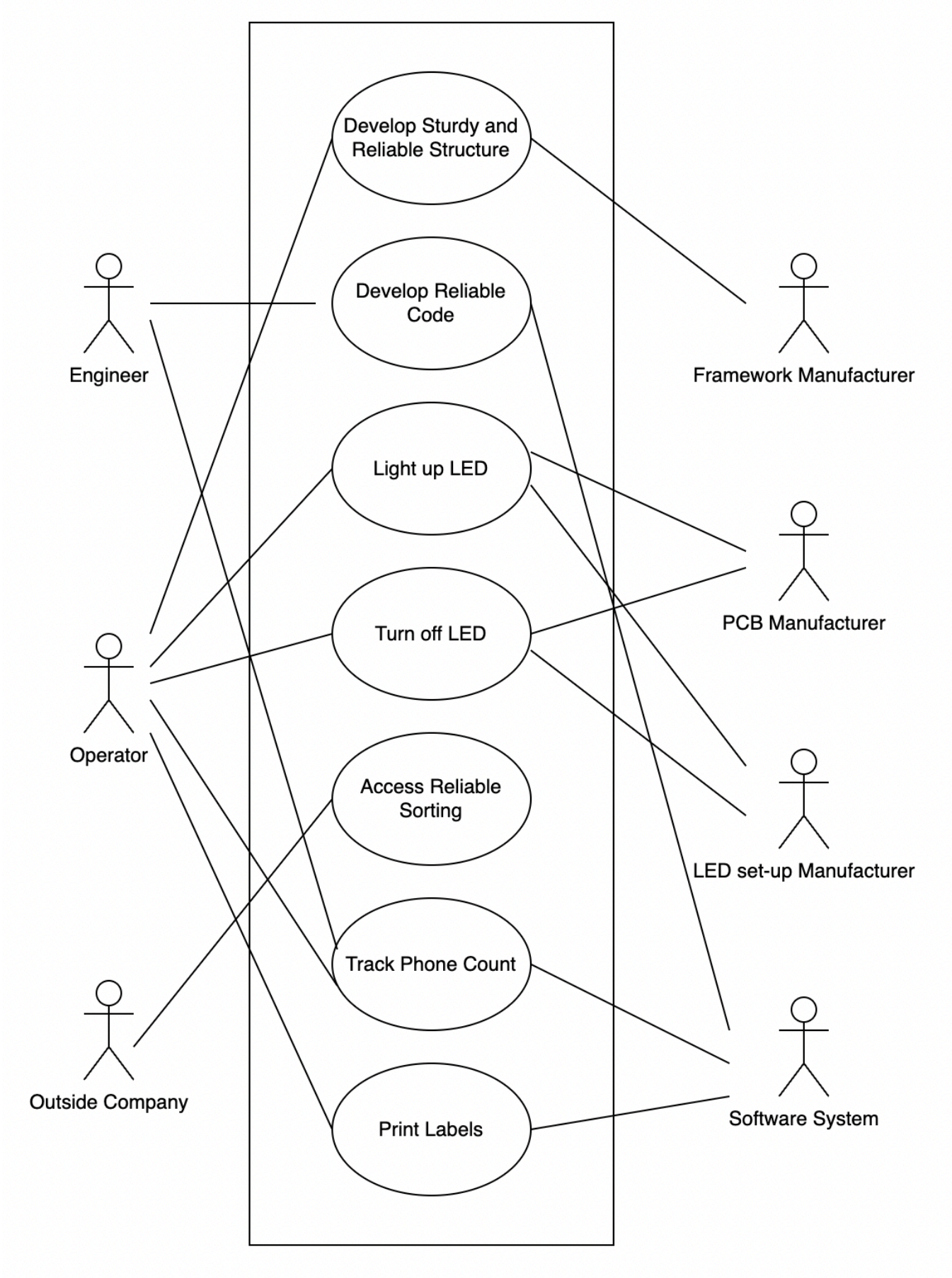
Other Non-functional Requirements:

System Design Using Domain Modeling

UML Use Case Diagram

Use Cases:

1. Develop Reliable Code (Engineer, software system)
2. Light up LED (operator, PCB, LED)
3. Access Reliable Sorting (Outside Company)
4. Turn off LED (Operator, LED, PCB)
5. Track Phone Count (Operator, Engineer, software system)
6. Print labels (operator, software system)
7. Create sturdy and reliable structure (operator, framework)


Develop Sturdy and Reliable Structure
Develop Reliable Code
Light up LED
Turn off LED
Access Reliable Sorting
Track Phone Count
Print Labels
Engineer
Operator
Outside Company
Framework Manufacturer
PCB Manufacturer
LED set-up Manufacturer
Software System

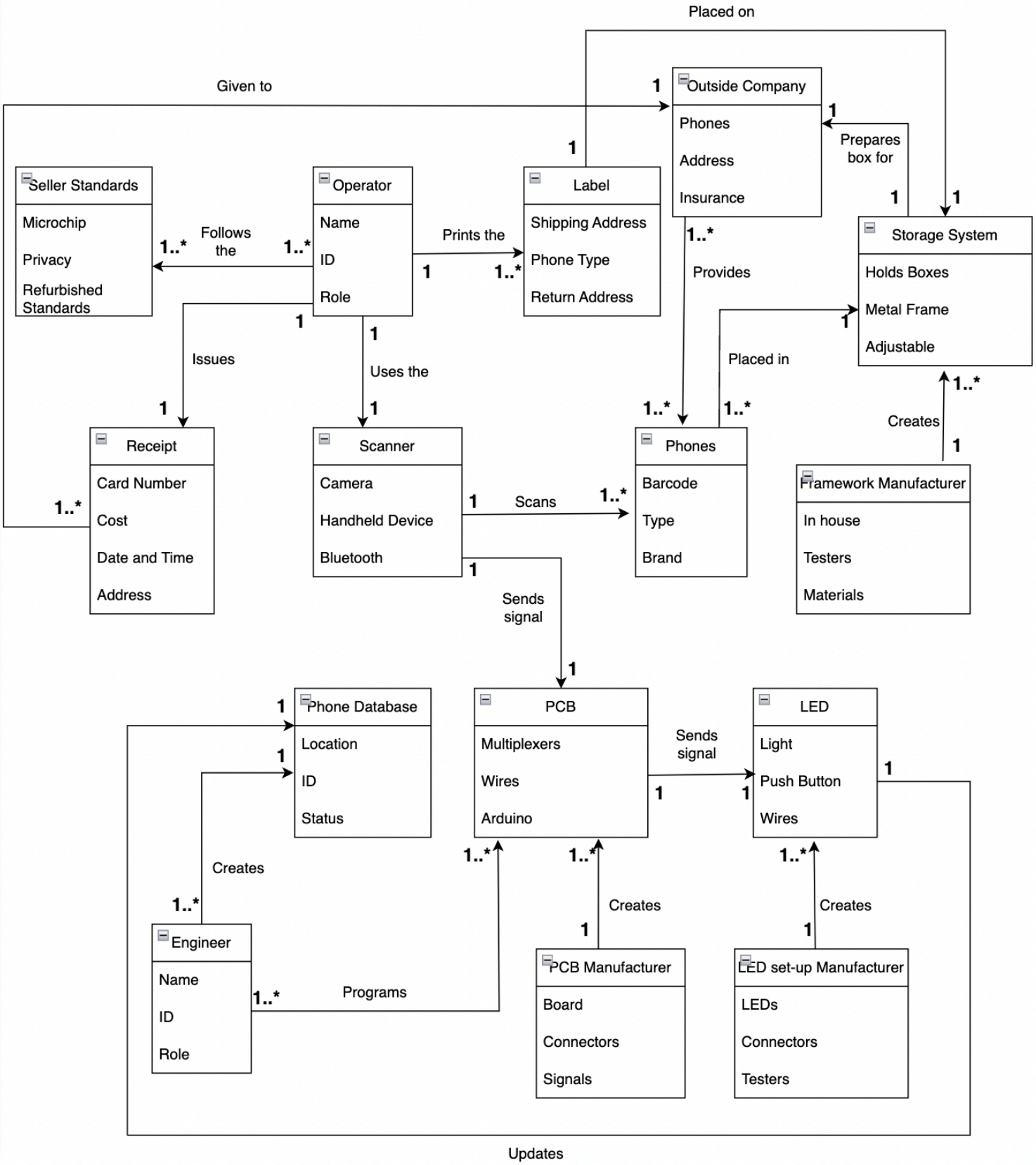
UML Domain Model

Conceptual Class Category	**Example**
Physical or Tangible Objects	Scanner, Barcode, Label, LED, Phone, PCB, Storage System
Specifications, Designs or Descriptions of Things	Phone Specifications
Places	Warehouse
Transactions	Sell Phones
Transaction Line Items	Phones
Roles of People	Framework Manufacturer, PCB Manufacturer, LED set-up Manufacturer, Operator, Engineer
Containers of Other Things	Phone Database
Things in a Container	Phone Information
Other Computers/Systems (External)	Database, Manufacturing Companies
Abstract Noun Concepts	Skillfulness
Organizations	Outside Company
Events	Scan barcode, Scan shipping label
Processes	Repair
Rules and Policies	Seller Standards, Privacy (in regards to the chip)
Catalogs	Electronics
Records of Finance, Work, Contracts, Legal Matters, etc.	Receipt
Financial Instruments and Services	Insurance Company
Manuals, Books, Documents, Reference Papers	Research Articles

Pruning Classes

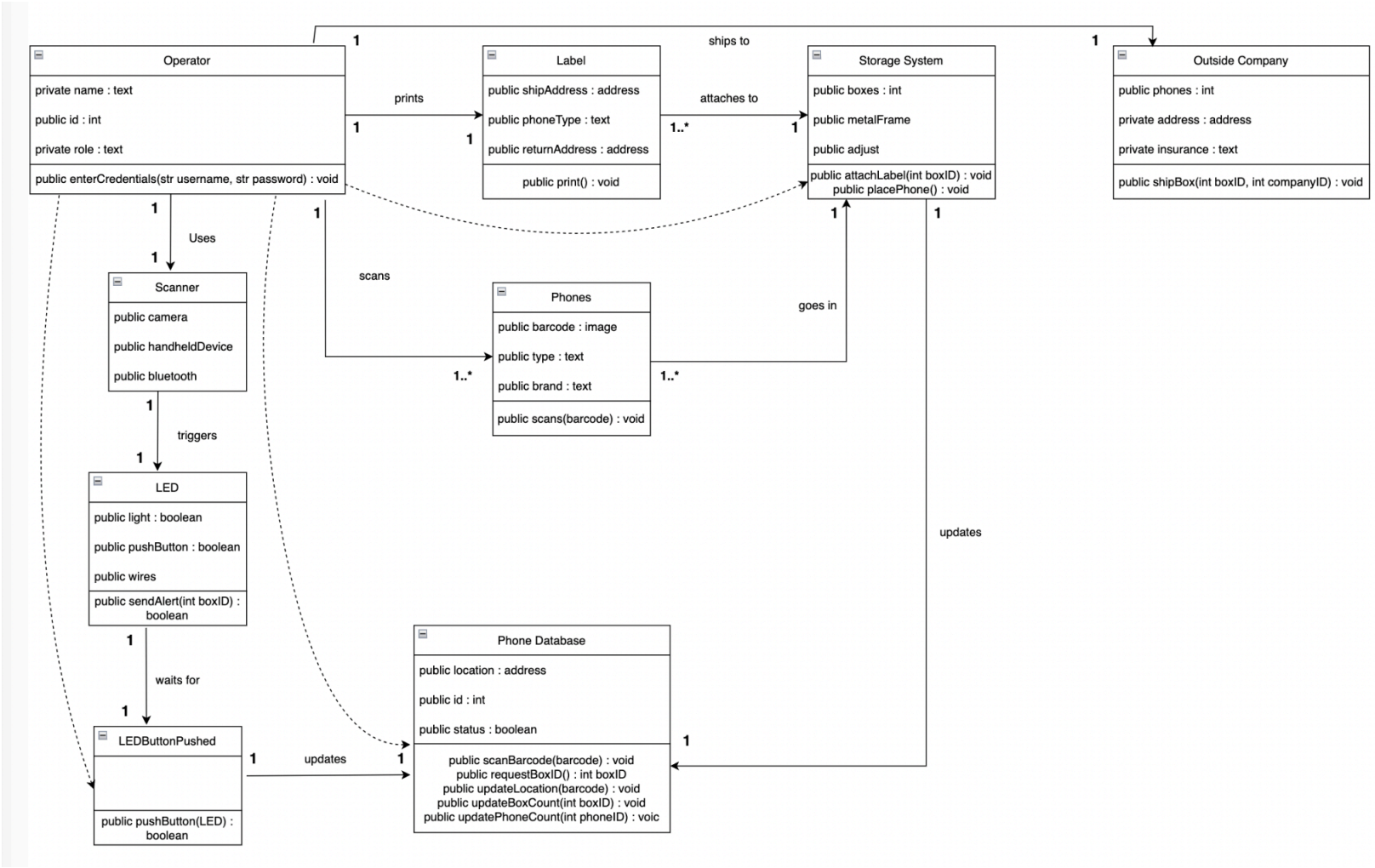
Good Classes (Retained)	**Bad Classes (Pruned)**

Scanner	Phone
Label	Phone Specifications
Storage system	Warehouse
LED	Sell Phones
Phones	Phone Information
Framework Manufacturer	Database
PCB Manufacturer	Manufacturing Companies
LED set-up Manufacturer	Skillfulness
Operator	Scan barcode
Engineer	Scan shipping label
Phone Database	Repair
Outside Company	Privacy
Seller Standards	Electronics
Receipt	Insurance Company
PCB	Research Articles
	Barcode


Placed on
Given to
Outside Company
Phones
Address
Insurance
Prepares box for
Seller Standards
Microchip
Privacy
Refurbished Standards
Follows the
Operator
Name
ID
Role
Prints the
Label
Shipping Address
Phone Type
Return Address
Provides
Storage System
Holds Boxes
Metal Frame
Adjustable
Placed in
Issues
Uses the
Receipt
Card Number
Cost
Date and Time
Address
Scanner
Camera
Handheld Device
Bluetooth
Scans
Phones
Barcode
Type
Brand
Creates
Framework Manufacturer
In house
Testers
Materials
Sends signal
Phone Database
Location
ID
Status
PCB
Multiplexers
Wires
Arduino
Sends signal
LED
Light
Push Button
Wires
Creates
Engineer
Name
ID
Role
Programs
Creates
PCB Manufacturer
Board
Connectors
Signals
Creates
LED set-up Manufacturer
LEDs
Connectors
Testers
Updates

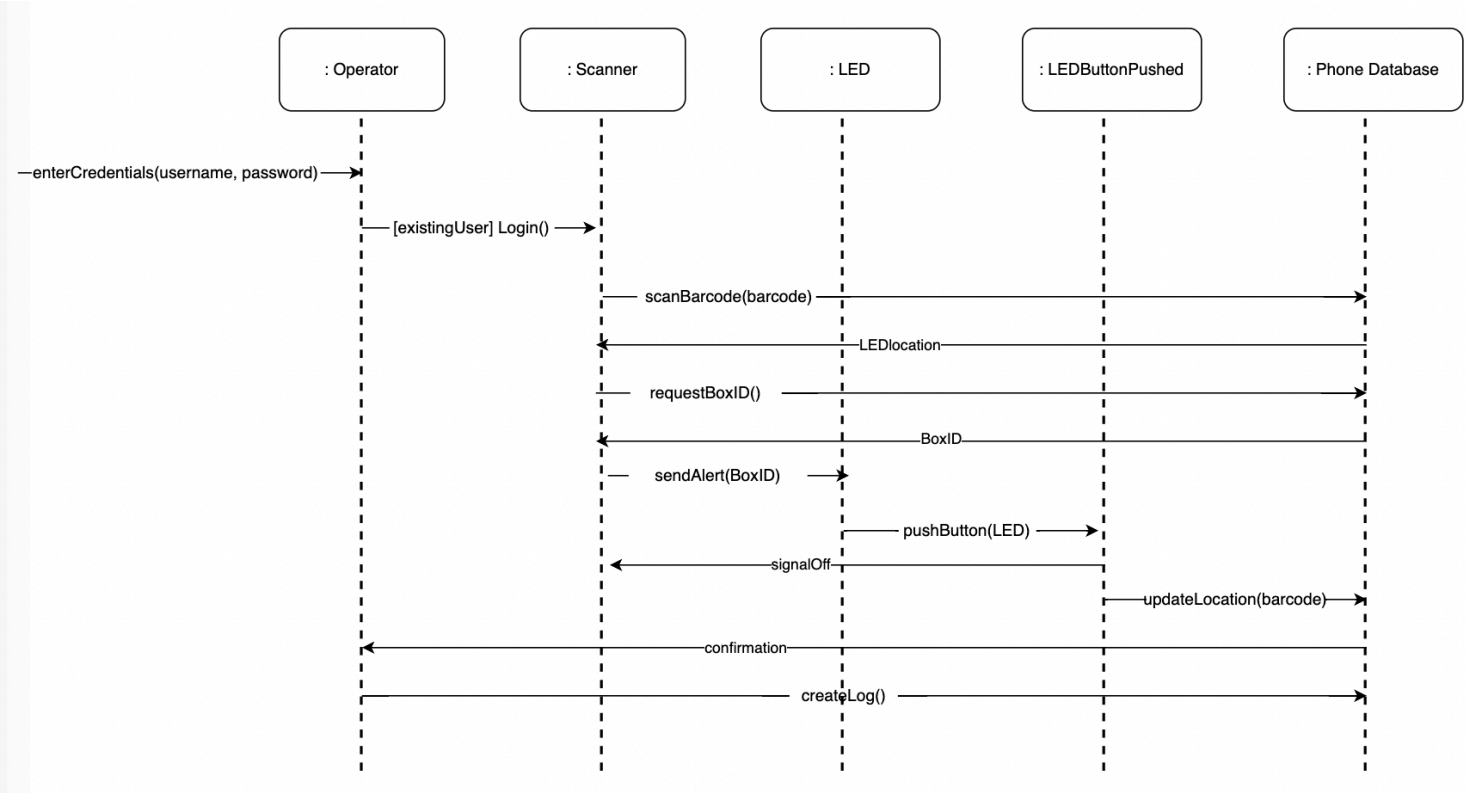
UML Class Diagram

Pruned:
Seller Standards (not relevant to the sorting system), Receipt (not relevant), Framework Manufacturer (implementation), Engineer (has to do with implementation), PCB (implementation), PCB Manufacturer (implementation), LED set-up Manufacturer (implementation)
Kept:
Operator, Label, Outside Company, Storage System, Scanner, Phones, Phone Database, LED

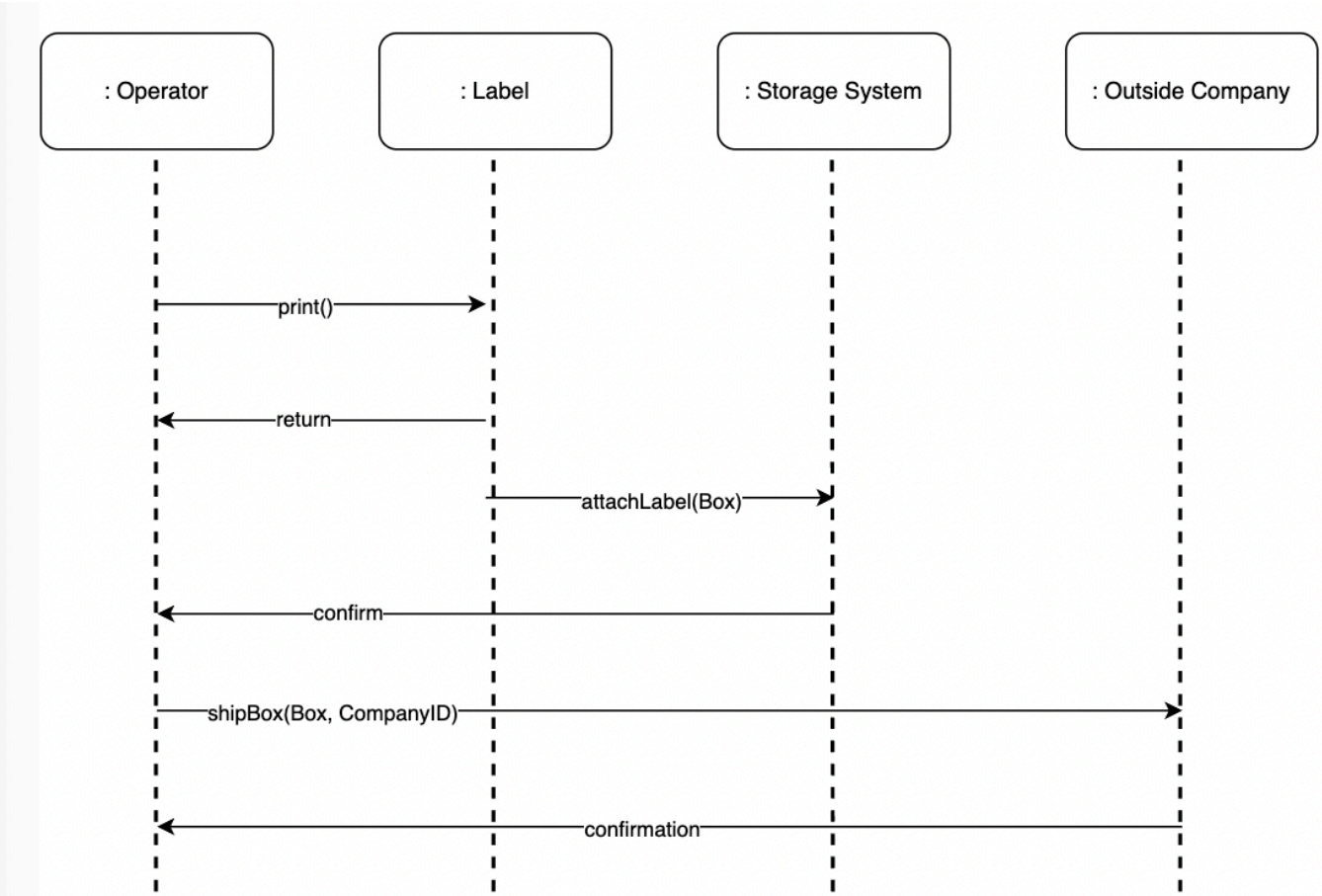


UML Sequence Diagrams

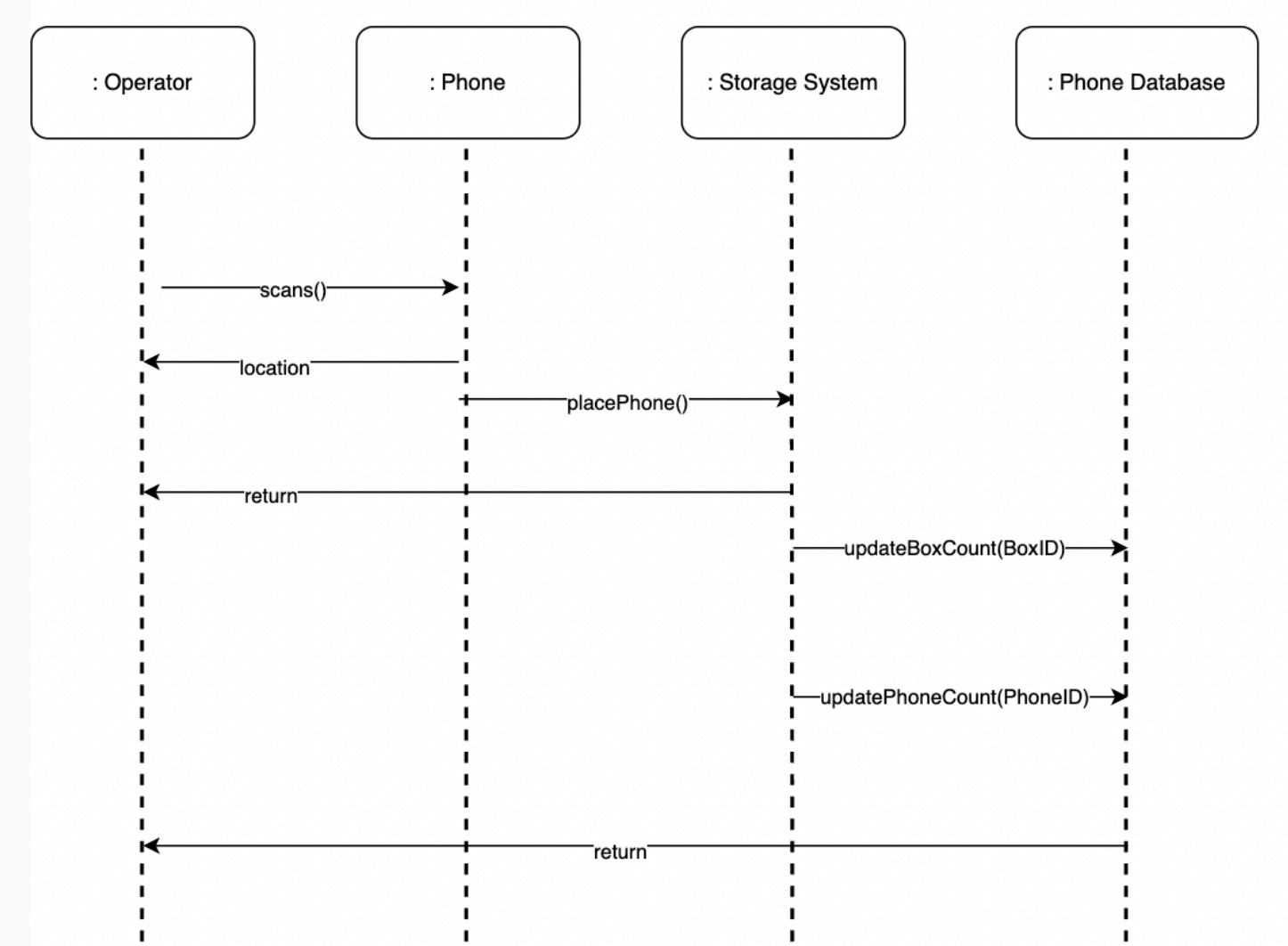
1. Light Up LED



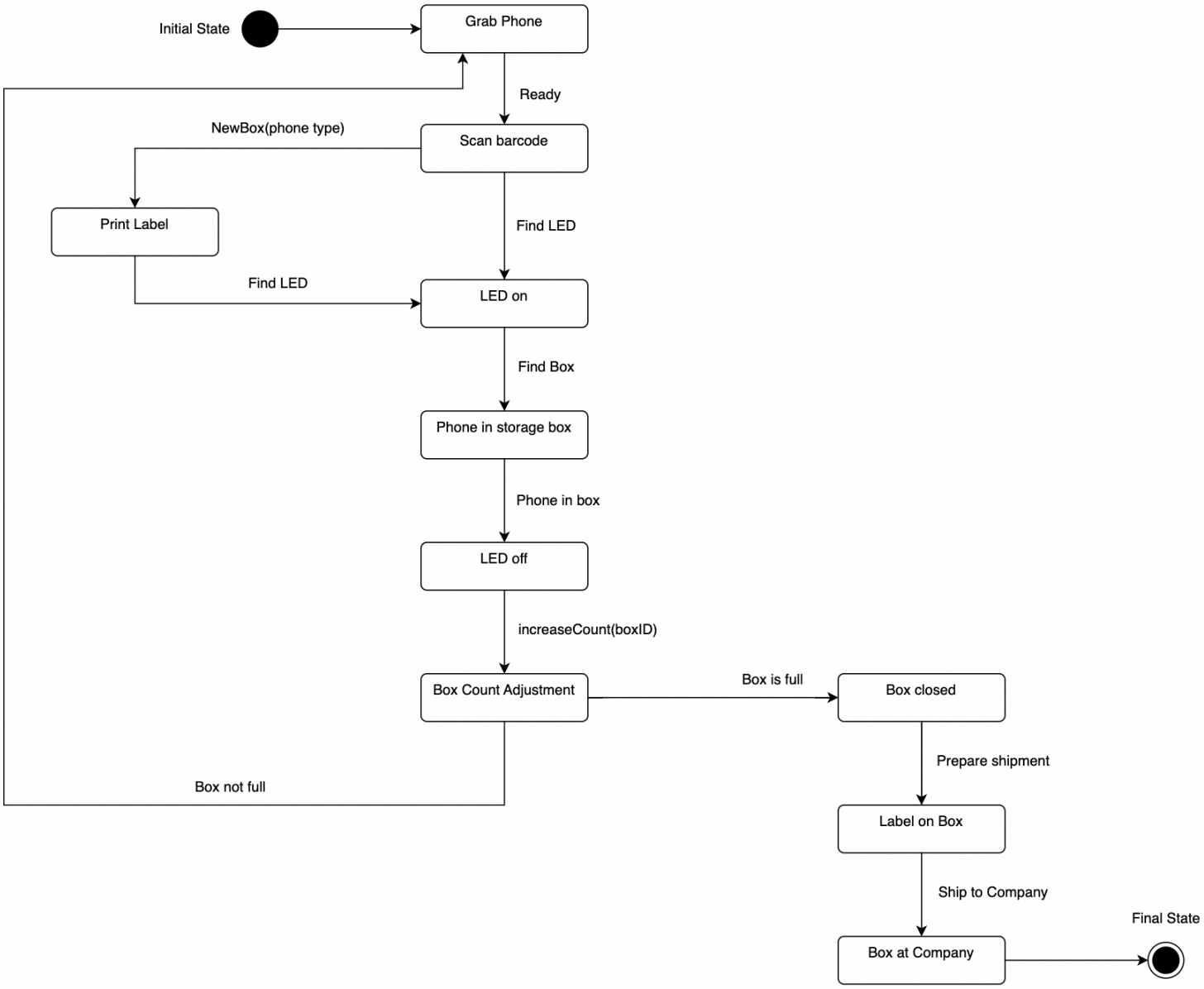
2. Print Labels



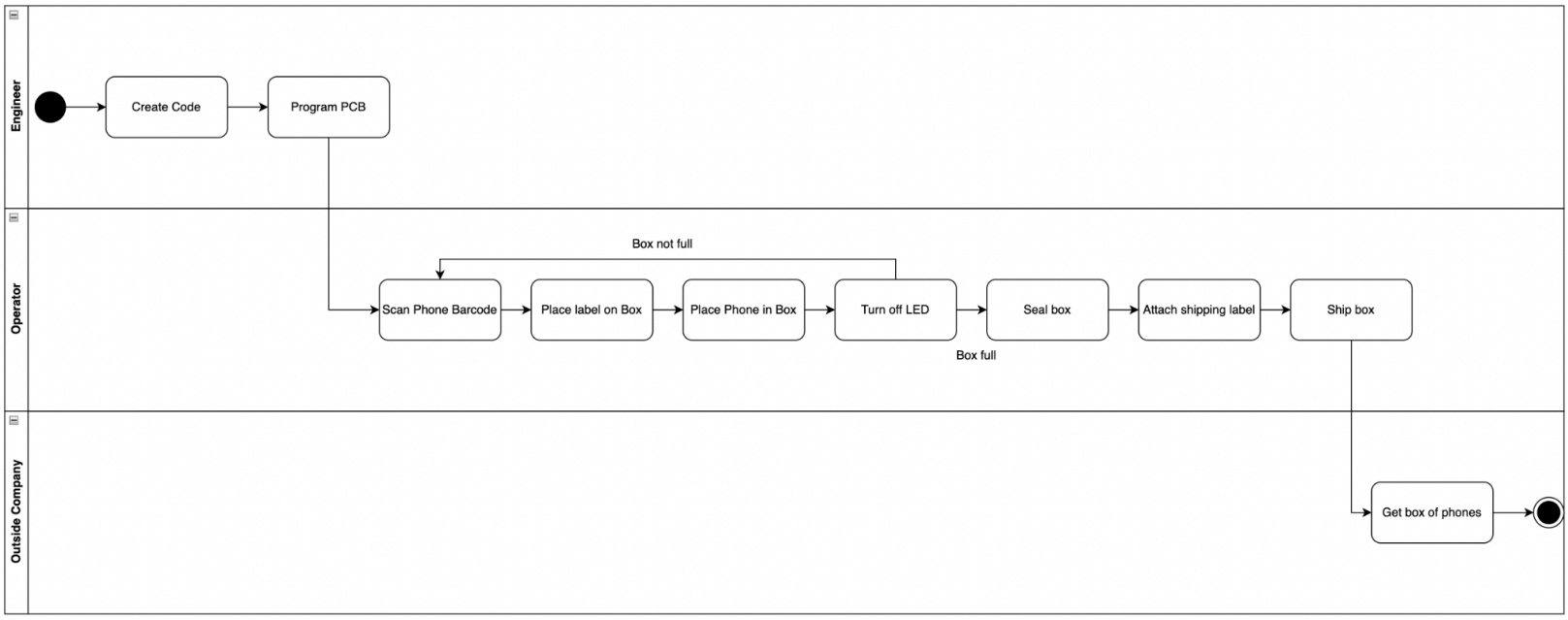
3. Track Phone Count



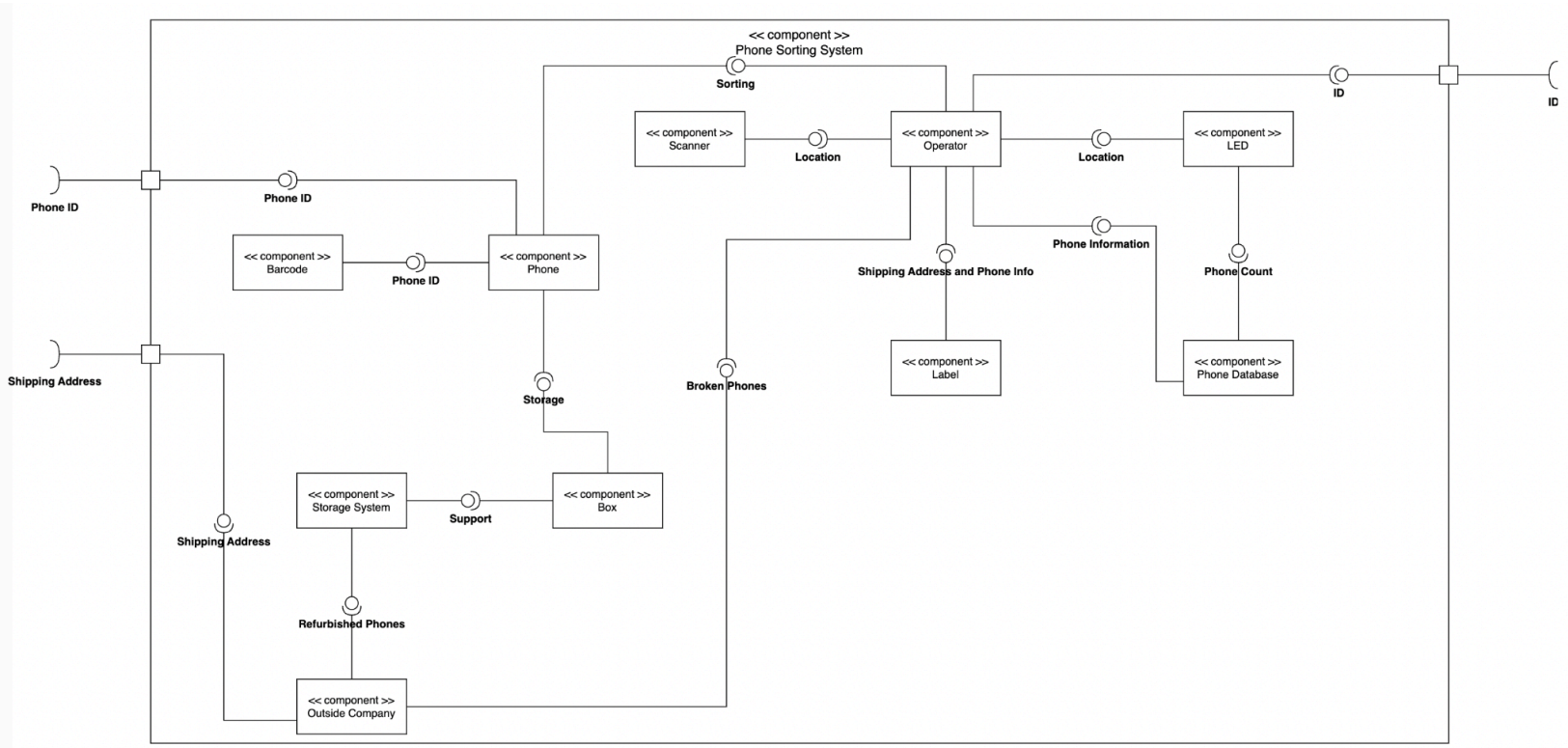
UML State Diagram



UML Activity Diagram

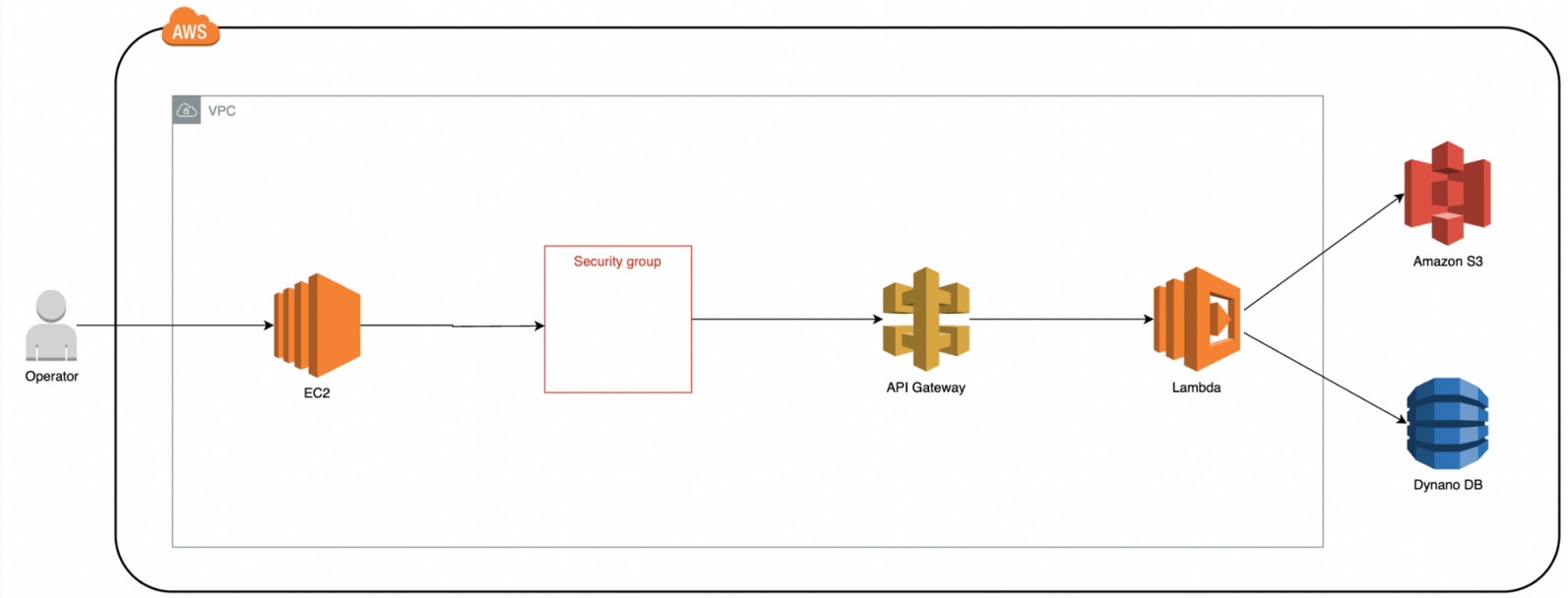


UML Component Diagram



Cloud Deployment Diagram

S3: used to store the phone database information
API: used to access the sorting system
DynamoDB: NoSQL database used for the sorting app
Lambda: used to run the code after the phone is scanned
Security group: controls what comes in and out of the ec2 instance
EC2 instance: used to run things virtually



Skeleton Classes and Tables Definition

Operator

```
class Operator {
    String name;
    String role;
    int id;

    public void enterCredentials(String username, String, password) {
    }

    public static void main(String[] args) {
    }
}
```

Label

```
class Label {
    String shipAddress;
    String phoneType;
    String returnAddress;

    public void print() {
    }

    public static void main(String[] args) {
    }
}
```

Storage System

```
class StorageSystem {
    int boxes;
    metalFrame;
    adjust;

    public void attachLabel(int boxID) {
    }

    public void placePhone() {

    }

    public static void main(String[] args) {
    }
}
```

Outside Company

```
class OutsideCompany {
    int phones;
    String address;
    String insurance;

    public void shipBox(int boxID, int companyID) {
    }

    public static void main(String[] args) {
    }
}
```

LED

```
class LED {
    boolean light;
    boolean pushButton;
    wires;

    public boolean sendAlert(int boxID) {
          return boolean;
    }

    public static void main(String[] args) {
    }
}
```

Phones

```
class Phones {
    image;
    String type;
    String brand;

    public void scans(int barcode) {
    }

    public static void main(String[] args) {
    }
}
```

Phone Database

```
class PhoneDatabase {
    String location;
    String type;
    int id;
    boolean status;

    public void scanBarcode(int barcode) {
    }

    public int requestBoxID() {
        return boxID;
    }

    public void updateLocation() {
    }

    public void updateBoxCount(int boxID) {
    }
```

```
    public void updatePhoneCount(int phoneID) {
    }

    public static void main(String[] args) {
    }
}
```

Table Definition:

The phone database includes the location of the phone in the warehouse as a string, the type of phone it is as a string, the id of the phone as an integer, and the status of it as a boolean. True means the phone has been fixed and is ready to be sent out while false means the phone is still broken. The information is stored in a table similar to an Excel sheet.

Design Patterns

GRASP:

I implemented low coupling into my system as there are not too many connections between the classes.

I incorporated high cohesion into the system as most of the methods and attributes in a specific class are related to one another.

There needs to be logs for when the phones are refurbished and moved around. Instead of placing the responsibility of storing this information on the storage system itself, another class is created to focus solely on these logs.

SOLID:

Single Responsibility Principle is seen as most of the classes are responsible for one task.

Open Closed Principle is incorporated as attributes can easily be added to the classes without making large modifications.

Dependency Inversion Principle is used as the high-level classes are not affected by changes in the low-level ones.

GOF:

The Operator Class is an example of a composite role as it is holding many components.

The Storage System class is an example of the Facade role as it encompasses a large number of subsystems.

The LEDButtonPushed class is an example of the Command role as this class is used solely to show the execution of the operator pushing the LED.

Microservices:

API Gateway is used for accessing the phone sorting service.

Authentication is used when scanning the phone barcode and allowing access to the system for the operator.