Final Project: Point of Sale System

Use Cases Diagram:

Use Case Elaborations:

Use Case #1

Use Case Name:	Add Clerk
Id:	1
Scenario:	New user is added to system as a clerk
Triggering Event:	New manager is hired
Brief Description:	A new manager is added to the system
Actors:	Administrator
Assumptions:	User is not already in the system
Frequency of Use:	Monthly
Related Use Cases:	Deactivate User, Promote User
Stakeholders:	Administrator, Clerk
Preconditions:	Clerk is not in system. Clerk details are known.
Postconditions:	Clerk has been added to the system with a PIN assigned.
Main Course:	1. Administrator selects Add Clerk from Employee Management menu

```
                    2. Administrator enters clerk details
                    3. System prompts for confirmation of details
                    4. System offers test of assigned PIN
                    5. System adds clerk
Alternate Course:   1. Employee number already in system
                       a. Offer to change employee number for new employee
                       b. Offer to reactivate deactivated employee
                    2. PIN doesn't match
                       a. Re-enter PIN
```

Use Case #2

```
Use Case Name:      Add Manager
Id:                 2
Scenario:           New user is added to system as a manager
Triggering Event:   New manager is hired
Brief Description:  A new manager is added to the system
Actors:             Administrator
Assumptions:        Manager is not already in the system (as Clerk)
Frequency of Use:   Monthly
Related Use Cases:  Deactivate User, Promote User
Stakeholders:       Administrator, Manager
Preconditions:      Manager is not in system.  Manager details are known.
Postconditions:     Manager has been added to the system with a PIN assigned.
Main Course:        1. Administrator selects Add Manager from Employee Management menu
                    2. Administrator enters manager details
                    3. System prompts for confirmation of details
                    4. System offers test of assigned PIN
                    5. System adds manager
Alternate Course:   1. Employee number already in system
                       a. Offer to change employee number for new employee
                       b. Offer to promote clerk to manager
                       c. Offer to reactivate deactivated employee
                    2. PIN doesn't match
                       a. Re-enter PIN
```

Use Case #3

```
Use Case Name:      Deactivate User
Id:                 3
Scenario:           Current user is deactivated in system.
Triggering Event:   User's employment is terminated
Brief Description:  A user is deactivated within the system
Actors:             Administrator
Assumptions:        User is active in the system
Frequency of Use:   Monthly
Related Use Cases:  Add Clerk, Add Manager
Stakeholders:       Administrator, Manager, Clerk
Preconditions:      Employee is in active in the system.
Postconditions:     Employee has been deactivated.
Main Course:        1. Administrator selects Deactivate User from Employee Management menu
                    2. Administrator specifies employee to be deactivated
                    3. System prompts for confirmation of deacivation
                    4. System deactivates employee
```

```
Alternate Course:   1. Employee already deactivated
                       a. System notifies administrator that user is not currently active
```

Use Case #4

```
Use Case Name:      Add Item to System
Id:                 4
Scenario:           New item for sale gets added to system
Triggering Event:   New item becomes available
Brief Description:  A new item is added to the sale system.
Actors:             Manager
Assumptions:        Item is not already in the system.
                    Item details (description, codes, price) are known.
Frequency of Use:   Weekly
Related Use Cases:  Deactivate Item
Stakeholders:       Manager, Clerk
Preconditions:      Item is not in system.  Item details are known.
Postconditions:     Item is available to be sold through the system.
Main Course:        1. Manager selects Add Item from Item Management menu
                    2. Manager enters item details
                    3. System prompts for confirmation of details
                    4. System offers test scan of item (to verify barcode)
                    5. System adds item
Alternate Course:   1. Item code already in system
                       a. Offer to change details of existing item
                    2. Barcode scan doesn't match code entered
                       a. Offer to scan item again to try for match (Wrong item was scanned)
                       b. Offer to update item code to match what was scanned
```

Use Case #5

```
Use Case Name:      Deactivate Item
Id:                 5
Scenario:           Existing item for sale gets deactivated in the system
Triggering Event:   Item is no longer available for sale
Brief Description:  An existing item is deactivated in the sale system.
Actors:             Manager
Assumptions:        Item is active in the system.
Frequency of Use:   Weekly
Related Use Cases:  Add Item to System
Stakeholders:       Manager, Clerk
Preconditions:      Item is available in system.
Postconditions:     Item is not available to be sold through the system.
Main Course:        1. Manager selects Deactivate Item from Item Management menu
                    2. Manager specficies the item
                    3. System prompts for confirmation of deactivation
                    4. System deactivates item
Alternate Course:   1. Item has already been deactivated
                       a. System alerts manager that item has already been deactivated
                    2. Item is not in the system
                       a.  System alerts manager that item is not in the system
```

Use Case #6

```
Use Case Name:      Update Item Details
```

```
Id:                 6
Scenario:           Item for sale gets updated
Triggering Event:   Item details change
Brief Description:  Item details are updated in the sale system.
Actors:             Manager
Assumptions:        Item is already in the system.
                    Item details (description, codes, price) are known.
Frequency of Use:   Weekly
Related Use Cases:  Add Item to System
Stakeholders:       Manager, Clerk
Preconditions:      Item is in system.  Updated details are known.
Postconditions:     Item is available to be sold through the system with updated details.
Main Course:        1. Manager selects Update Item Details from Item Management menu
                    2. Manager selects which detail to update
                    3. Manager inputs updated detail information
                    4. System prompts for confirmation of details
                    5. System updates item details
Alternate Course:   1. Item code not already in system
                       a. Offer to add new item
```

Use Case #7

```
Use Case Name:      Approve Item Return
Id:                 7
Scenario:           Item returned is being approved and processed
Triggering Event:   Clerk or manager has initiated an item return
Brief Description:  Item return is approved
Actors:             Manager
Assumptions:        Item is eligible for return, return has been initiated
Frequency of Use:   Daily
Related Use Cases:  Initiate Item Return
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Item return has been initiated
Postconditions:     Item return is processed
Main Course:        1. Summary of returned items is presented
                    2. Manager logs in
                    3. Manager approves return
                    4. Refund is processed
Alternate Course:   1. Manager initiated returna and summary can be approved as displayed
```

Use Case #8

```
Use Case Name:      Ring Up Item
Id:                 8
Scenario:           Item to be purchased is entered for the sale
Triggering Event:   Customer wishes to purchase item
Brief Description:  Employee rings up an item
Actors:             Clerk or Manager, Customer
Assumptions:        Item has already been entered into the system
Frequency of Use:   Many times each hour
Related Use Cases:  Add Item to Sysem, Update Item Details, Remove Item from Sale
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Item has been entered into system
Postconditions:     Item has been added to the sale transaction
```

```
Main Course:        1. Item is scanned by clerk or manager
                    2. Item is added to sale
                    3  System displays item details
Alternate Course:   1. Employee can specify quantity of items after scanning
                    2. System updates quantity on display
```

Use Case #9

```
Use Case Name:      Remove Item from Sale
Id:                 9
Scenario:           Item that has been rung up needs to be removed from the transaction
Triggering Event:   Customer changes mind, Item is unavailable, Mistaken item scan
Brief Description:  A new manager is added to the system
Actors:             Clerk or Manager, Customer
Assumptions:        Item has been rung up, item should not be processed with sale
Frequency of Use:   Daily or Hourly
Related Use Cases:  Ring Up Item
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Item has been rung up
Postconditions:     Item is not included on sale
Main Course:        1. Employee selects item from current sale
                    2. Employee either:
                       a. Specifices new (reduced) quanity
                       b. Selects remove to eliminate the item
                    3. System shows updated quantity (possibly zero)
Alternate Course:   1. Employee sets system to item removal mode
                    2. Employee scans item to indicate which item to remove
                    3. Employee specificies quantity to remove (or all)
                    4. System displays updated quantity (zero)
```

Use Case #10

```
Use Case Name:      Compute Total and Process Payment
Id:                 10
Scenario:           Item(s) have been rung up and are ready for the sale to be completed
Triggering Event:   Customer has completed purchase and all items have been entered
Brief Description:  Total is calculated and payment made.
Actors:             Clerk or Manager, Customer
Assumptions:        Item(s) have been rung up, sale can proceed
Frequency of Use:   Hourly
Related Use Cases:  Ring Up Item, Print Invoice/Receipt
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Item(s) have been rung up
Postconditions:     Sale is recorded and payment made
Main Course:        1. Employee selects Start Payment from sale screen
                    2. System displays total
                    3. Employee specifies payment method
                    4. System contacts payment device
                    5. System records sale and prints reciept
Alternate Course:   1. Payment is denied
                       a. System offers to cancel transaction
```

Use Case #11

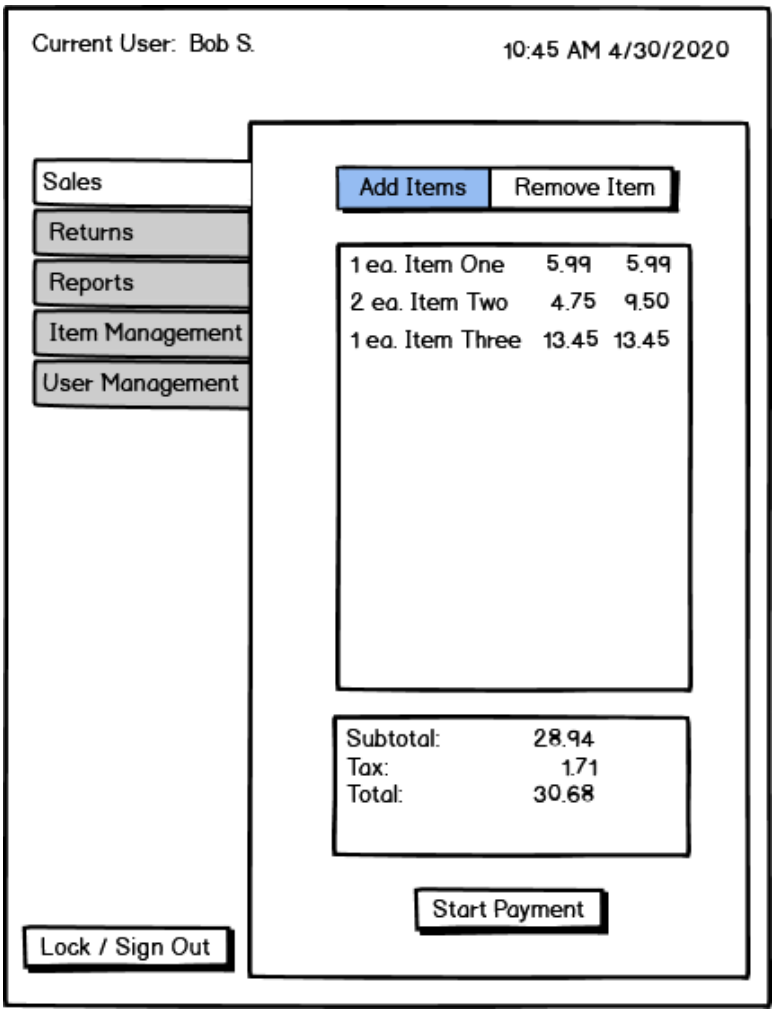
```
Use Case Name:      Print Invoice/Receipt
```

```
Id:                 11
Scenario:           Item is being returned by a customer
Triggering Event:   Customer brings previously purchased item back for return
Brief Description:  Item is processed for return by clerk
Actors:             Clerk or Manager, Customer
Assumptions:        Sale has been completed
Frequency of Use:   Hourly
Related Use Cases:  Compute Total and Process Payment
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Sale has been processed
Postconditions:     Receipt has been printed
Main Course:        1. Sale is completed
                    2. Clerk or Manager selects Print Receipt
                    3. Receipt is printed
Alternate Course:   1. Printer Malfunction
                       a. Alert Clerk that printing has failed
                       b. Provide options to cancel or retry
```

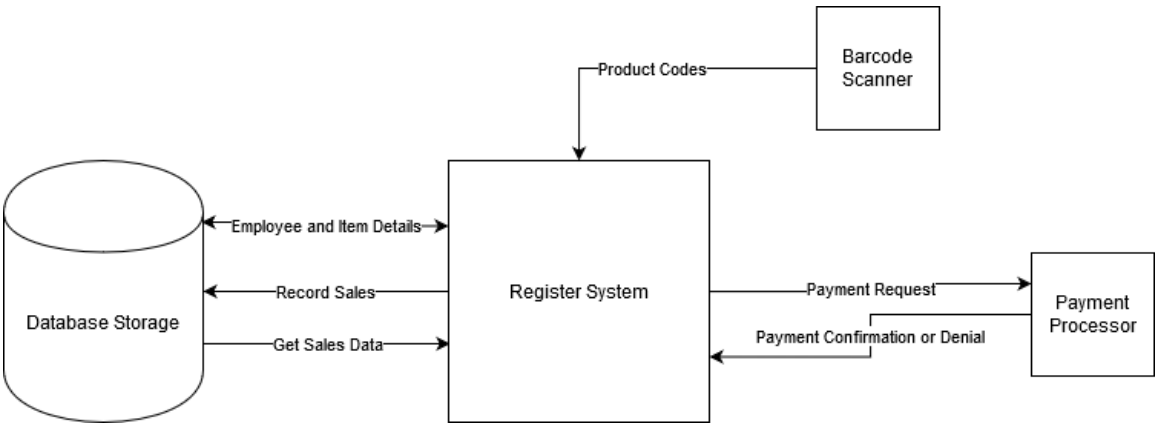
Use Case #12

```
Use Case Name:      Initiate Item Return
Id:                 12
Scenario:           Item is being returned by a customer
Triggering Event:   Customer brings previously purchased item back for return
Brief Description:  Item is processed for return by clerk
Actors:             Clerk or Manager, Customer
Assumptions:        Item is eligible for return
Frequency of Use:   Daily
Related Use Cases:  Approve Item Return, Ring Up Item
Stakeholders:       Manager, Clerk, Customer
Preconditions:      Item is being returned
Postconditions:     Item return is pending approval
Main Course:        1. Item is scanned by clerk or manager
                    2. Item is added to return set
                    3. Clerk or Manager specifies original sale price for refund amount
                    4. Clerk initiates approval request
Alternate Course:   1. Manager initiates return and automatically proceeds with approval
```

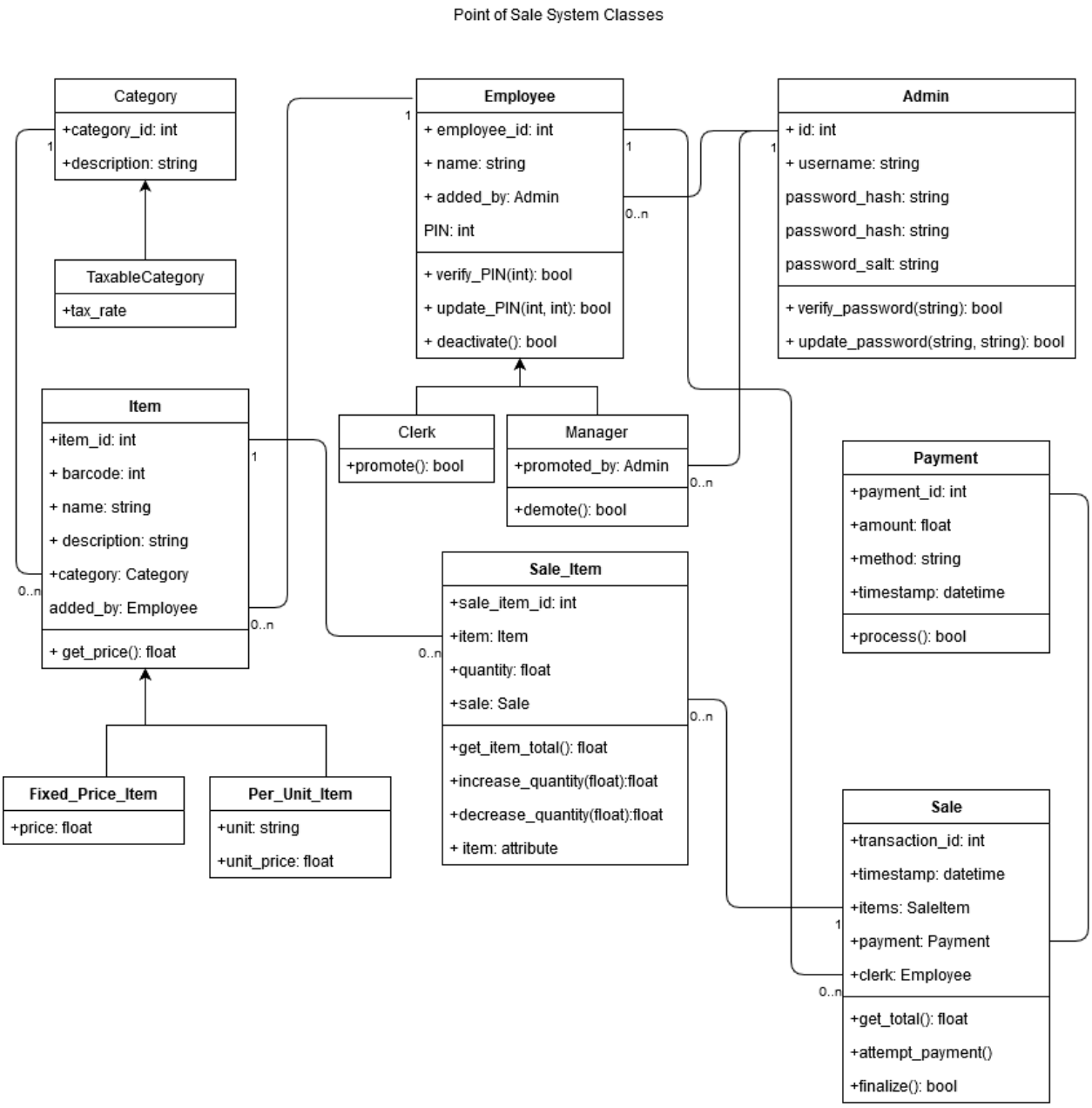
UI Sketch:



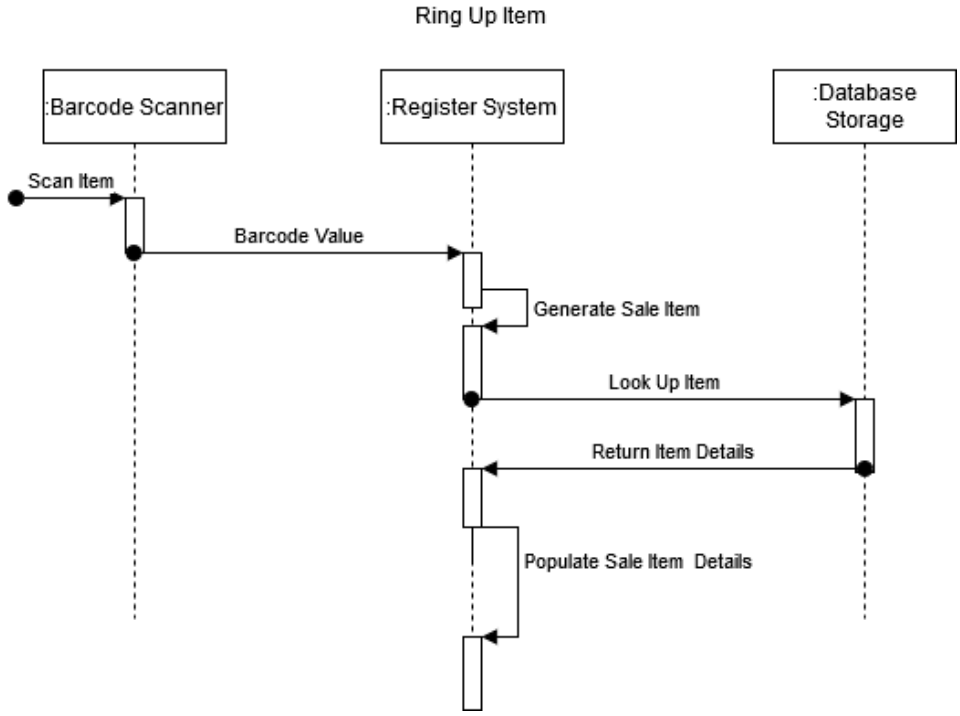
Architecture Diagram:



Class Diagram:



Sequence Diagrams:



Application:

```
# Patrick Johnson           5/1/2020 #
# SWDV 630 3W 20/SP2  Final Project #
#####################################
#Point of Sale System – Partial Implementation
from datetime import datetime
import sqlalchemy as sa
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

# Classes
class Category(Base):
    def __init__(self, description):
        self.description = description

    def __str__(self):
        return self.description

    __tablename__ = "Categories"

    category_id = sa.Column(sa.Integer, primary_key = True)
    description = sa.Column(sa.String)

class Employee(Base):
    def __init__(self, name, PIN):
        self.name = name
        self.__PIN = PIN

```

```
30      def __repr__(self):
31          return "Employee - ID: {}; Name: {}".format(self.employee_id, self.name)
32
33      __tablename__ = "Employees"
34      employee_id = sa.Column(sa.Integer, primary_key = True)
35      name = sa.Column(sa.String(24))
36      __PIN = sa.Column(sa.String(8))
37      employee_type = sa.Column(sa.String(7))
38
39      __mapper_args__ = {
40          'polymorphic_on':employee_type,
41          'polymorphic_identity':'Employee'
42      }
43
44      def verify_PIN(self, PIN):
45          return self.__PIN == PIN
46
47      def update_PIN(self, old_PIN, new_PIN):
48          if self.verify_PIN(old_PIN):
49              self.__PIN = new_PIN
50              return True
51          else:
52              return False
53
54  class Manager(Employee):
55      def __repr__(self):
56          return "Manager  - ID: {}; Name: {}".format(self.employee_id, self.name)
57
58      __mapper_args__ = {
59          'polymorphic_identity':'Manager'
60      }
61
62  class Clerk(Employee):
63      def __repr__(self):
64          return "Clerk    - ID: {}; Name: {}".format(self.employee_id, self.name)
65
66      __mapper_args__ = {
67          'polymorphic_identity':'Clerk'
68      }
69
70  class Item(Base):
71      def __init__(self, barcode, description, category, employee):
72          self.barcode = barcode
73          self.description = description
74          self.category = category
75          self.added_by = employee
76
77      def __repr__(self):
78          return "{} Item - ID: {} Barcode: {}; Category: {}; Description: {}".format(
79              self.type, self.item_id, self.barcode, self.category, self.description)
80
81      __tablename__ = "Items"
82      item_id = sa.Column(sa.Integer, primary_key = True)
83      barcode = sa.Column(sa.Integer)
84      description = sa.Column(sa.String(32))
85      category_id = sa.Column(sa.Integer, sa.ForeignKey('Categories.category_id'))
86      category = relationship("Category")
87      added_by_employee_id = sa.Column(sa.Integer, sa.ForeignKey('Employees.employee_id'))
88      add_by = relationship("Employee")
```

```
89      type = sa.Column(sa.String(10))
90
91      __mapper_args__ = {
92          'polymorphic_on':type,
93          'polymorphic_identity':'BaseItem'
94      }
95
96      def get_price(self):
97          pass
98
99      price = property(get_price, None, None, "Item Price")
100
101     @classmethod
102     def get_item_from_barcode(cls, barcode, session):
103         return session.query(Item).filter(Item.barcode == barcode).one()
104
105 class Fixed_Price_Item(Item):
106     def __init__(self, barcode, description, category, employee, price):
107         Item.__init__(self, barcode, description, category, employee)
108         self._price = price
109
110     _price = sa.Column(sa.Float)
111
112     __mapper_args__ = {
113         'polymorphic_identity':'FixedPrice'
114     }
115
116     def get_price(self):
117         return self._price
118
119     price = property(get_price, None, None, "Price")
120
121 class Per_Unit_Item(Item):
122     def __init__(self, barcode, description, category, employee, price, unit):
123         Item.__init__(self, barcode, description, category, employee)
124         self.unit = unit
125         self._unit_price = price
126
127     _unit_price = sa.Column(sa.Float)
128     unit = sa.Column(sa.String(12))
129     __mapper_args__ = {
130         'polymorphic_identity':'PerUnit'
131     }
132
133     def get_price(self):
134         return self._unit_price
135
136     price = property(get_price, None, None, "Unit Price")
137
138 class SaleItem(Base):
139     def __init__(self, item, quantity, sale):
140         self.item = item
141         self.quantity = quantity
142         self.sale = sale
143
144     def __str__(self):
145         return "{} - {} @ ${}".format(self.item.description,
146                                       self.quantity,
147                                       self.item.price)
```

```
148
149     __tablename__ = "SaleItems"
150     sale_item_id = sa.Column(sa.Integer, primary_key = True)
151     quantity = sa.Column(sa.Float)
152     item_id = sa.Column(sa.Integer, sa.ForeignKey('Items.item_id'))
153     item = relationship("Item")
154     sale_id = sa.Column(sa.Integer, sa.ForeignKey('Sales.transaction_id'))
155     sale = relationship("Sale", back_populates="items")
156
157     def get_item_total(self):
158         return self.item.price * self.quantity
159
160     price = property(get_item_total)
161
162 class Sale(Base):
163     def __init__(self, employee):
164         self.clerk = employee
165
166     __tablename__ = "Sales"
167     transaction_id = sa.Column(sa.Integer, primary_key = True)
168     timestamp = sa.Column(sa.DateTime)
169
170     items = relationship("SaleItem", back_populates="sale")
171     clerk_id = sa.Column(sa.Integer, sa.ForeignKey('Employees.employee_id'))
172     clerk = relationship("Employee")
173
174     def add_item(self, barcode, session, quantity = 1):
175         item = Item.get_item_from_barcode(barcode, session)
176         saleitem = SaleItem(item, quantity, self)
177
178     def get_total(self):
179         total = 0
180         for i in self.items:
181             total += i.price
182         return total
183
184     def finalize(self, session):
185         self.timestamp = datetime.now()
186         session.commit()
187
188 def populate_database(session):
189     bob = Manager("Bob S.", "1234")
190     session.add(bob)
191     session.add(Clerk("Ann D.", "0000"))
192     session.commit()
193
194     # Add a category
195     paper_category = Category("Paper Products")
196     session.add(paper_category)
197     session.commit()
198
199     # Create Some Items
200     session.add(Fixed_Price_Item(2812, "Notebook", paper_category, bob, 5.99))
201     session.add(Per_Unit_Item(4353, "Paper", paper_category, bob, 0.25, "Sheet"))
202     session.commit()
203
204 def main():
205     engine = sa.create_engine("sqlite://", echo = False)
206     Base.metadata.create_all(engine)
```

```
207     session = sa.orm.sessionmaker(bind=engine)()
208
209     populate_database(session)  # Adds sample items and users
210
211     # Check PINs:
212     bob = session.query(Employee).first()
213     print("Check incorrect PIN for Bob:", bob.verify_PIN("1111"))
214     print("Check correct PIN for Bob:  ", bob.verify_PIN("1234"))
215
216     # Update PIN for Bob
217     print("Updating Bob's PIN to '5309':", bob.update_PIN("1234", "5309"))
218     print("Check old PIN for Bob:", bob.verify_PIN("1234"))
219     print("Check new PIN for Bob:", bob.verify_PIN("5309"))
220
221     # Print Employee List
222     print("Employees:")
223     for employee in session.query(Employee).all():
224         print(employee)
225
226     # Create a  Sale
227     sale = Sale(bob)                    # Generate new sale
228     session.add(sale)                   # Add to session
229     sale.add_item(2812, session)        # Add items
230     sale.add_item(4353, session, 5)
231     sale.finalize(session)              # timestamp and commit to database
232
233     print("Sale Items:")
234     for si in sale.items:
235         print(si)
236     print("Sale Total:", sale.get_total())
237     print("Sale Clerk:", sale.clerk)
238     print("Sale Time: ", sale.timestamp)
239
240
241 if __name__ == "__main__":
242     main()
```

Output:

```
Check incorrect PIN for Bob: False
Check correct PIN for Bob:   True
Updating Bob's PIN to '5309': True
Check old PIN for Bob: False
Check new PIN for Bob: True
Employees:
Manager  - ID: 1; Name: Bob S.
Clerk    - ID: 2; Name: Ann D.
Sale Items:
Notebook - 1.0 @ $5.99
Paper - 5.0 @ $0.25
Sale Total: 7.24
Sale Clerk: Manager  - ID: 1; Name: Bob S.
Sale Time:  2020-05-02 16:56:08.008369
```

GitHub Link:

https://github.com/PJohnson9/SWDV630_POS