*"""  
Name: ReRead.py  
Authors: Michael Coughlin, Leah Mattingly, Aubrie McIntyre, Perrin Brumfield, Gautam Mehla  
Date Last Updated: May 9th, 2024  
Description: Tkinter application that facilitates the purchase and selling of used books.  
Users can register, login, browse books, add books to cart, and purchase books.  
"""*import tkinter as tk  
from tkinter import ttk  
from tkinter import messagebox  
import sqlite3  
from user import User  
from book import Book  
from PIL import Image, ImageTk  
  
global logged\_in  
logged\_in = False  
global userID  
userID = None  
global cart\_empty  
  
  
class MainPage:  
 def \_\_init\_\_(self, master, db\_connection, inventory\_db\_connection, cart=None):  
 *"""  
 Initialize the main page.  
  
 Args:  
 master (tk.Tk): The master Tkinter window.  
 db\_connection: SQLite database connection for user data.  
 inventory\_db\_connection: SQLite database connection for inventory data.  
 """* self.master = master  
 self.db\_connection = db\_connection  
 self.inventory\_db\_connection = inventory\_db\_connection *# Store inventory\_db\_connection* self.master.title("ReRead - Main Page")  
 self.master.configure(background='#F7F7F7') *# Light grey background color* self.master.protocol("WM\_DELETE\_WINDOW", self.on\_closing)  
  
 if cart is None:  
 self.cart = [] *# Initialize cart as an empty list if not provided* else:  
 self.cart = cart *# Don't recreate the cart if reopening this page* image\_path = "books.jpg" *# Adjust the path accordingly* image = Image.open(image\_path)  
  
 *# Resize the image if needed* image = image.resize((500, 300)) *# Adjust width and height as needed  
  
 # Convert the image to a format compatible with Tkinter* self.photo = ImageTk.PhotoImage(image)  
  
 *# Create a Label widget to display the image* image\_label = tk.Label(self.master, image=self.photo, bg='#F7F7F7')  
 image\_label.image = self.photo *# Keep a reference to prevent garbage collection* image\_label.pack()  
  
 welcomea\_label = tk.Label(self.master, text="Welcome to ReRead!", font=("Arial", 16), bg='#F7F7F7')  
 welcomea\_label.pack(pady=10)  
 *# Introduction* intro\_text = """  
 Where the love for books meets the joy of recycling!   
 Dive into our virtual bookstore, where every page holds a story and every purchase breathes new life into pre-loved books.  
 Search for your next literary adventure, add favorites to your cart, and embark on a journey through the endless shelves of knowledge.  
 Let's rediscover the magic of reading while also caring for our planet. Happy browsing!  
 """  
 intro\_label = tk.Label(master, text=intro\_text, wraplength=400, justify="center", font=("Arial", 12),  
 bg='#F7F7F7') *# Set background color* intro\_label.pack()  
  
 inventory\_button = tk.Button(master, text="Inventory", command=self.open\_inventory\_window, font=("Arial", 12),  
 bg='#007BFF', fg='white') *# Set button color* inventory\_button.pack(pady=5)  
  
 if not logged\_in:  
 *# Login and Register buttons* login\_button = tk.Button(master, text="Login", command=self.open\_login\_window, font=("Arial", 12),  
 bg='#28A745', fg='white')  
 login\_button.pack(pady=5)  
  
 register\_button = tk.Button(master, text="Register", command=self.open\_register\_window, font=("Arial", 12),  
 bg='#28A745', fg='white')  
 register\_button.pack(pady=5)  
  
 if logged\_in:   
 view\_cart\_button = tk.Button(master, text="View Cart", command=self.open\_cart\_window, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 view\_cart\_button.pack(pady=5)  
  
 logout\_button = tk.Button(master, text="Log Out", command=self.logout, font=("Arial", 12),  
 bg='#DC3545', fg='white')  
 logout\_button.pack(pady=5)  
  
 def logout(self):  
 *"""  
 Logs the user out  
 """* global logged\_in   
 logged\_in = False   
 self.master.destroy()  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
  
 def clear\_cart(self):   
 *"""  
 Clears the cart after checkout   
 """* self.cart = []  
  
 def open\_inventory\_window(self):  
 *"""  
 Open the inventory window.  
 """* self.master.withdraw() *# Hide the main window* inventory\_window = tk.Toplevel(self.master) *# Create a new window* inventory\_window.title("ReRead - Inventory")   
 inventory\_window.configure(bg='#F7F7F7')   
 inventory\_window.protocol("WM\_DELETE\_WINDOW", self.on\_inventory\_window\_close)   
 InventoryPage(inventory\_window, self.db\_connection, self.inventory\_db\_connection, self.cart, self.open\_cart\_window)   
  
 def open\_cart\_window(self):  
 *"""  
 Open the cart window.  
 """* self.master.withdraw() *# Hide the main window* cart\_window = tk.Toplevel(self.master) *# Create a new window* cart\_window.protocol("WM\_DELETE\_WINDOW", self.on\_cart\_window\_close) *# Set the close window callback* cart\_window.title("ReRead - View Cart") *# Set the window title* cart\_window.configure(bg='#F7F7F7')   
 CartPage(cart\_window, self.cart, self.db\_connection, self.inventory\_db\_connection, self.clear\_cart)   
  
 def open\_login\_window(self):  
 *"""  
 Open the login window.  
 """* self.master.withdraw() *# Hide the main window* login\_window = tk.Toplevel(self.master) *# Create a new window* login\_window.protocol("WM\_DELETE\_WINDOW", self.on\_login\_window\_close) *# Set the close window callback* login\_window.title("ReRead - Login") *# Set the window title* login\_window.configure(bg='#F7F7F7')   
 LoginPage(login\_window, self.db\_connection, self.inventory\_db\_connection)   
  
  
 def open\_register\_window(self):  
 *"""  
 Open the registration window.  
 """* self.master.withdraw() *# Hide the main window* register\_window = tk.Toplevel(self.master) *# Create a new window* register\_window.protocol("WM\_DELETE\_WINDOW", self.on\_register\_window\_close) *# Set the close window callback* register\_window.title("ReRead - Registration")  
 register\_window.configure(bg='#F7F7F7')  
 RegistrationPage(register\_window, self.db\_connection, self.inventory\_db\_connection)  
  
 *# Show the main window when window is closed* def on\_inventory\_window\_close(self):  
 *"""  
 Callback when the inventory window is closed.  
 """* self.master.destroy()   
 if logged\_in:   
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection, self.cart)   
 else:  
 main()  
  
 def on\_cart\_window\_close(self):  
 *"""  
 Callback when the cart window is closed.  
 """* self.master.destroy()   
 if logged\_in:  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
 else:  
 main()  
  
 def on\_register\_window\_close(self):  
 *"""  
 Callback when the register window is closed.  
 """* self.master.destroy()  
 if logged\_in:  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
 else:  
 main()  
  
 def on\_login\_window\_close(self):  
 *"""  
 Callback when the login window is closed.  
 """* self.master.destroy()  
 self.photo = None  
 if logged\_in:  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
 else:  
 main()  
  
 def on\_closing(self):  
 *"""  
 Callback when the main window is closed.  
 """  
 # Close database connections* self.db\_connection.close()  
 self.inventory\_db\_connection.close()  
 global logged\_in  
 logged\_in = False  
 self.master.quit()  
 self.master.destroy()  
  
class RegistrationPage:  
 *"""  
 Class representing the registration page.  
 """* def \_\_init\_\_(self, master, db\_connection, inventory\_db\_connection):  
 *"""  
 Initialize the registration page.  
  
 Args:  
 master (tk.Tk): The master Tkinter window.  
 db\_connection: SQLite database connection for user data.  
 """* self.master = master  
 self.db\_connection = db\_connection  
 self.inventory\_db\_connection = inventory\_db\_connection  
 self.master.title("ReRead - Registration")  
 self.master.configure(background='#F7F7F7')  
  
 join\_label = tk.Label(self.master, text="Join Us!", font=("Arial", 16), bg='#F7F7F7')  
 join\_label.pack(pady=10)  
 *# Load and display the login image* login\_image = Image.open("login.png")  
 resized\_login\_image = login\_image.resize((200, 200))  
 login\_photo = ImageTk.PhotoImage(resized\_login\_image)  
  
 login\_label = tk.Label(master, image=login\_photo, bg='#F7F7F7')  
 login\_label.image = login\_photo  
 login\_label.pack()  
  
 *# Labels and Entry Widgets* tk.Label(master, text="Username:", font=("Arial", 12), bg='#F7F7F7').pack()  
 self.username\_entry = tk.Entry(master, font=("Arial", 12))  
 self.username\_entry.pack()  
  
 tk.Label(master, text="Password:", font=("Arial", 12), bg='#F7F7F7').pack()  
 self.password\_entry = tk.Entry(master, show="\*", font=("Arial", 12))  
 self.password\_entry.pack()  
  
 *# Register Button* register\_button = tk.Button(master, text="Register", command=self.register\_user, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 register\_button.pack(pady=10)  
  
 def register\_user(self):  
 username = self.username\_entry.get()  
 password = self.password\_entry.get()  
 global logged\_in  
  
 if not username or not password:  
 tk.messagebox.showerror("Error", "Please enter both username and password.")  
 return  
  
 *# Check if username already exists* cursor = self.db\_connection.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=?", (username,))  
 existing\_user = cursor.fetchone()  
  
 if existing\_user:  
 tk.messagebox.showerror("Error", "Username already exists. Please choose a different username.")  
 return  
 else:  
 *# Insert new user into the database* cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, password))  
 self.db\_connection.commit()  
  
 *# Get the last inserted row ID (auto-incremented)* user\_id = cursor.lastrowid  
  
 *# Format the user ID to a four-digit number* user\_id\_four\_digits = '{:04d}'.format(user\_id)  
  
 *# Create a new user object* new\_user = User(user\_id\_four\_digits, username, password)  
  
 tk.messagebox.showinfo("Success", "Registration successful!")  
 logged\_in = True  
  
 *# Clear entry fields after registration* self.username\_entry.delete(0, tk.END)  
 self.password\_entry.delete(0, tk.END)  
  
 *# Withdraw and destroy window after registering* self.master.withdraw()  
 self.master.destroy()  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
   
  
class LoginPage:  
 *"""  
 Class representing the login page.  
 """* def \_\_init\_\_(self, master, db\_connection, inventory\_db\_connection):  
 *"""  
 Initialize the login page.  
  
 Args:  
 master (tk.Tk): The master Tkinter window.  
 db\_connection: SQLite database connection for user data.  
 inventory\_db\_connection: SQLite database connection for inventory data.  
 """* self.master = master  
 self.db\_connection = db\_connection  
 self.inventory\_db\_connection = inventory\_db\_connection  
 self.master.title("ReRead - Login")  
 self.master.configure(background='#F7F7F7')  
  
 welcomeb\_label = tk.Label(self.master, text="Welcome Back!", font=("Arial", 16), bg='#F7F7F7')  
 welcomeb\_label.pack(pady=10)  
  
 *# Load and display the login image* login\_image = Image.open("login.png")  
 resized\_login\_image = login\_image.resize((200, 200))  
 login\_photo = ImageTk.PhotoImage(resized\_login\_image)  
  
 login\_label = tk.Label(master, image=login\_photo, bg='#F7F7F7')  
 login\_label.image = login\_photo  
 login\_label.pack()  
  
 *# Labels and Entry Widgets* tk.Label(master, text="Username:", font=("Arial", 12), bg='#F7F7F7').pack()  
 self.username\_entry = tk.Entry(master, font=("Arial", 12))  
 self.username\_entry.pack()  
  
 tk.Label(master, text="Password:", font=("Arial", 12), bg='#F7F7F7').pack()  
 self.password\_entry = tk.Entry(master, show="\*", font=("Arial", 12))  
 self.password\_entry.pack()  
  
 *# Login Button* login\_button = tk.Button(master, text="Login", command=self.login\_user, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 login\_button.pack(pady=10)  
  
 def login\_user(self):  
 *"""  
 Log in the user.  
 """* username = self.username\_entry.get()   
 password = self.password\_entry.get()   
 global logged\_in  
 global userID  
  
 if not username or not password:  
 messagebox.showerror("Error", "Please enter both username and password.")  
 return  
  
 *# Check if username and password match* cursor = self.db\_connection.cursor()  
 cursor.execute("SELECT \* FROM users WHERE username=? AND password=?", (username, password))   
 user = cursor.fetchone()   
  
 if user:  
 """  
 Log in check  
 """  
 userID = user[0]  
 messagebox.showinfo("Success", "Login successful!")  
 logged\_in = True  
 self.master.withdraw()   
 self.master.destroy()  
 main\_page = MainPage(tk.Toplevel(), self.db\_connection, self.inventory\_db\_connection)  
  
 else:  
 messagebox.showerror("Error", "Invalid username or password.")  
  
class InventoryPage:  
 *"""  
 Class representing the inventory page.  
 """* def \_\_init\_\_(self, master, db\_connection, inventory\_db\_connection, cart, open\_cart\_window):  
 *"""  
 Initialize the inventory page.  
  
 Args:  
 master (tk.Tk): The master Tkinter window.  
 db\_connection: SQLite database connection for user data.  
 inventory\_db\_connection: SQLite database connection for inventory data.  
 """* self.master = master   
 self.db\_connection = db\_connection   
 self.inventory\_db\_connection = inventory\_db\_connection   
 self.cart = cart  
 self.open\_cart\_window2 = open\_cart\_window   
 self.master.configure(bg='#F7F7F7')   
  
 self.inventory\_tree = ttk.Treeview(master)  
 self.inventory\_tree["columns"] = ("Title", "Author", "Price", "Quantity")  
 self.inventory\_tree.heading("#0", text="ID")  
 self.inventory\_tree.column("#0", width=50)  
 self.inventory\_tree.heading("Title", text="Title")  
 self.inventory\_tree.heading("Author", text="Author")  
 self.inventory\_tree.heading("Price", text="Price")  
 self.inventory\_tree.heading("Quantity", text="Quantity")  
 self.inventory\_tree.pack(padx=10, pady=10)  
 self.populate\_inventory()  
  
 refresh\_button = tk.Button(master, text="Refresh", command=self.refresh\_inventory, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 refresh\_button.pack(pady=10)  
  
 if logged\_in:  
 sell\_button = tk.Button(master, text="Sell", command=self.open\_sell\_page, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 sell\_button.pack(pady=10)  
  
 add\_to\_cart\_button = tk.Button(master, text="Add to Cart", command=self.add\_to\_cart, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 add\_to\_cart\_button.pack(pady=10)  
  
 view\_cart\_button = tk.Button(master, text="View Cart", command=self.open\_cart\_window2, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 view\_cart\_button.pack(pady=5)  
  
 else:  
 tk.Label(master, text="Return to Main Page to Login", font=("Arial", 12), bg='#D0E7F9').pack()  
  
 def populate\_inventory(self):  
 *"""  
 Populate the inventory list.  
 """* cursor = self.inventory\_db\_connection.cursor()  
 cursor.execute("SELECT \* FROM inventory WHERE quantity > 0")  
 books = cursor.fetchall()  
  
 for book in books:  
 *# Format the price with two decimal places and a dollar sign* formatted\_price = "${:.2f}".format(book[5])  
 self.inventory\_tree.insert("", "end", text=book[0], values=(book[1], book[2], formatted\_price, book[6]))  
  
 def refresh\_inventory(self):  
 *# Clear existing items in the inventory treeview* for item in self.inventory\_tree.get\_children():  
 self.inventory\_tree.delete(item)  
  
 *# Repopulate the inventory treeview with updated data* self.populate\_inventory()  
  
 def open\_sell\_page(self):  
 *"""  
 Open the sell page.  
 """* sell\_window = tk.Toplevel(self.master)  
 sell\_window.title("ReRead - Sell Book")   
 sell\_window.configure(bg='#F7F7F7')  
 SellPage(sell\_window, self.inventory\_db\_connection)  
   
 def add\_to\_cart(self):  
 *"""  
 Get the selected item from the inventory treeview  
 """* selected\_item = self.inventory\_tree.selection()  
 if not selected\_item:  
 messagebox.showerror("Error", "Please select a book to add to cart.")  
 return  
  
 *# Extract book details from the selected item* book\_id = self.inventory\_tree.item(selected\_item, "text")  
 book\_title = self.inventory\_tree.item(selected\_item, "values")[0]  
 book\_author = self.inventory\_tree.item(selected\_item, "values")[1]   
 book\_price = self.inventory\_tree.item(selected\_item, "values")[2]  
 book\_quantity = int(self.inventory\_tree.item(selected\_item, "values")[3])  
   
 if book\_quantity <= 0:  
 messagebox.showerror("Error", "This book is out of stock.")  
 return  
  
 *# Update the inventory (subtract 1 from quantity)* new\_quantity = book\_quantity - 1  
 cursor = self.inventory\_db\_connection.cursor()  
 cursor.execute("UPDATE inventory SET quantity = ? WHERE id = ?", (new\_quantity, book\_id))  
 self.inventory\_db\_connection.commit()  
  
 *# Add the book to the cart list* self.cart.append({  
 "id": book\_id,  
 "title": book\_title,  
 "author": book\_author,  
 "price": book\_price  
 })  
  
 messagebox.showinfo("Success", f"Book '{book\_title}' added to cart.")  
  
  
class SellPage:  
 *"""  
 Class representing the page for selling a book.  
 """* def \_\_init\_\_(self, master, inventory\_db\_connection):  
 *"""  
 Initialize the sell page.  
  
 Args:  
 master (tk.Tk): The master Tkinter window.  
 inventory\_db\_connection: SQLite database connection for inventory data.  
 """* self.master = master   
 self.inventory\_db\_connection = inventory\_db\_connection *# Store inventory\_db\_connection* self.master.title("ReRead - Sell Book") *# Set window title* self.master.configure(background='#F7F7F7')   
  
 *# Labels and Entry Widgets for the book details form* tk.Label(master, text="Title:", font=("Arial", 12), bg='#F7F7F7').pack()   
 self.title\_entry = tk.Entry(master, font=("Arial", 12))   
 self.title\_entry.pack()  
  
 tk.Label(master, text="Author:", font=("Arial", 12), bg='#F7F7F7').pack()   
 self.author\_entry = tk.Entry(master, font=("Arial", 12))  
 self.author\_entry.pack()  
  
 tk.Label(master, text="Price:", font=("Arial", 12), bg='#F7F7F7').pack()  
 self.price\_entry = tk.Entry(master, font=("Arial", 12))   
 self.price\_entry.pack()  
  
 tk.Label(master, text="Quantity:", font=("Arial", 12), bg='#F7F7F7').pack()   
 self.quantity\_entry = tk.Entry(master, font=("Arial", 12))  
 self.quantity\_entry.pack()  
  
 sell\_button = tk.Button(master, text="Sell", command=self.sell\_book, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 sell\_button.pack(pady=10)  
  
 def sell\_book(self):  
 *# Get book details from the form* title = self.title\_entry.get()   
 author = self.author\_entry.get()  
 price = self.price\_entry.get()  
 quantity = self.quantity\_entry.get()  
  
 *# Validate quantity and price.* try:  
 quantity\_test = int(quantity)  
 price\_test = float(price)  
 if quantity\_test < 1 or price\_test < 0:  
 raise ValueError  
 except ValueError:  
 messagebox.showerror("Error", "Please enter a valid quantity or price.")  
 return  
  
 *# Validate input* if not title or not author or not price or not quantity:  
 messagebox.showerror("Error", "Please fill in all fields.")  
 return  
  
 *# Insert book into the inventory database* cursor = self.inventory\_db\_connection.cursor()  
 cursor.execute("SELECT \* FROM inventory WHERE title=? AND author=?", (title, author))  
 repeat = cursor.fetchone()  
 if repeat:  
 new\_quantity = int(quantity) + int(repeat[6]) *# Ensure consistency by converting repeat[6] to int* cursor.execute("UPDATE inventory SET quantity=? WHERE title=? AND author=?", (new\_quantity, title, author))  
 else:  
 cursor.execute("INSERT INTO inventory (title, author, price, quantity) VALUES (?, ?, ?, ?)",  
 (title, author, price, quantity))  
  
 self.inventory\_db\_connection.commit()  
 messagebox.showinfo("Success", "Book added to inventory successfully!")  
 self.master.destroy()  
  
  
class CartPage:  
 *"""  
 Cart page with treeview to see items in cart  
 """* def \_\_init\_\_(self, master, cart, db\_connection, inventory\_db\_connection, clear\_cart2):  
 self.master = master  
 self.cart = cart  
 self.db\_connection = db\_connection  
 self.inventory\_db\_connection = inventory\_db\_connection  
 self.clear\_cart2 = clear\_cart2  
  
 *# Create a Treeview to display cart items* self.cart\_tree = ttk.Treeview(master, columns=("Title", "Author", "Price"), show="headings")  
 self.cart\_tree.heading("Title", text="Title")  
 self.cart\_tree.heading("Author", text="Author")  
 self.cart\_tree.heading("Price", text="Price")  
 self.cart\_tree.pack(padx=10, pady=10)  
  
 *# Display cart items* self.display\_cart\_items()  
  
 *# Checkout button* checkout\_button = tk.Button(master, text="Checkout", command=self.checkout, font=("Arial", 12),  
 bg='#007BFF', fg='white')  
 checkout\_button.pack(pady=10)  
  
  
 def display\_cart\_items(self):  
 for item in self.cart:  
 self.cart\_tree.insert("", "end", values=(item['title'], item['author'], item['price']))  
  
 def checkout(self):  
 *# Insert cart items into user\_purchases table* cursor = self.db\_connection.cursor()  
 for item in self.cart:  
 cursor.execute("INSERT INTO user\_purchases (user\_id, book\_title, author, price, quantity) "  
 "VALUES (?, ?, ?, ?, ?)", (userID, item['title'], item['author'], item['price'], 1))  
 self.db\_connection.commit()  
 self.clear\_cart2()  
 messagebox.showinfo("Success", "Checkout successful!")  
 self.master.destroy()  
  
  
def main():  
 *# Create a SQLite database connection* db\_connection = sqlite3.connect("user\_database.db")  
 cursor = db\_connection.cursor()  
  
 *# Create users table if not exists* cursor.execute('''CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 username TEXT UNIQUE,  
 password TEXT)''')  
  
 *# Create users table if not exist* cursor.execute('''CREATE TABLE IF NOT EXISTS user\_purchases (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 user\_id INTEGER,  
 item\_no INTEGER,  
 book\_title TEXT,  
 author TEXT,  
 price REAL,  
 quantity INTEGER,  
 FOREIGN KEY (user\_id) REFERENCES users(id))''')  
  
 inventory\_db\_connection = sqlite3.connect("inventory\_database.db")  
 cursor\_inventory = inventory\_db\_connection.cursor()  
  
 *# Create inventory table if not exists* cursor\_inventory.execute('''CREATE TABLE IF NOT EXISTS inventory (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 title TEXT,  
 author TEXT,  
 ISBN TEXT,  
 condition TEXT,  
 price REAL,  
 quantity INTEGER)''')  
  
 root = tk.Tk()  
 main\_page = MainPage(root, db\_connection, inventory\_db\_connection)  
 root.mainloop()  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

user.py

*"""  
Name: user.py  
Authors: Michael Coughlin, Leah Mattingly, Aubrie McIntyre, Perrin Brumfield, Gautam Mehla  
Date Last Updated: May 9th, 2024  
Description: Class file for the User  
"""*class User:  
 *"""  
 User class when registering a new user  
 """* def \_\_init\_\_(self, userID, username, password):  
 self.userID = userID  
 self.username = username  
 self.password = password

book.py

*"""  
Name: book.py  
Authors: Michael Coughlin, Leah Mattingly, Aubrie McIntyre, Perrin Brumfield, Gautam Mehla  
Date Last Updated: May 9th, 2024  
Description: Class file for books  
"""*class Book:  
 *"""  
 Class for Books to be sold  
 """* def \_\_init\_\_(self, title, author, price):  
 self.title = title  
 self.author = author  
 self.price = price