```python
import bcrypt
import mysql.connector

class Auth:
    def __init__(self, db):
        self.db = db
        self.cursor = db.cursor()

    def register(self, username, password, role):
        # Hash the password using bcrypt
        password_hash = bcrypt.hashpw(password.encode('utf-8'),
bcrypt.gensalt())
        sql = "INSERT INTO users (username, password_hash, role) VALUES
(%s, %s, %s)"
        self.cursor.execute(sql, (username, password_hash, role))
        self.db.commit()
        print(f"User {username} registered successfully as {role}.")

    def login(self, username, password):
        # Query the database to check if the user exists
        sql = "SELECT password_hash, role FROM users WHERE username = %s"
        self.cursor.execute(sql, (username,))
        result = self.cursor.fetchone()

        if result:
            stored_password_hash, role = result
            """Check if the entered password matches the hashed
password"""
            if bcrypt.checkpw(password.encode('utf-8'),
stored_password_hash.encode('utf-8')):
                print(f"Login successful! Welcome, {username} ({role}).")
                return role
            else:
                print("Invalid password.")
                return None
        else:
            print("User not found.")
            return None
```

```
class BakeryItem:
    def __init__(self, db):
        self.db = db
        self.cursor = db.cursor()

    def add_to_db(self, name, price, stock):
        # Fixed the missing line break
        sql = "INSERT INTO inventory (item_name, price, stock) VALUES (%s,
%s, %s)"
        self.cursor.execute(sql, (name, price, stock))
        self.db.commit()
        print(f"Item '{name}' added to inventory.")

    def update_stock(self, item_name, quantity_sold):
        # Check if the item exists
        sql_check = "SELECT stock, price FROM inventory WHERE item_name =
%s"
        self.cursor.execute(sql_check, (item_name,))
        item = self.cursor.fetchone()

        if item:
            current_stock, price = item
            if current_stock >= quantity_sold:
                # Update stock in inventory
                sql_update = "UPDATE inventory SET stock = stock - %s
WHERE item_name = %s"
                self.cursor.execute(sql_update, (quantity_sold,
item_name))

                # Record the sale in the sales table
                sql_sales = """
                INSERT INTO sales (item_id, quantity_sold, total_price)
                VALUES ((SELECT item_id FROM inventory WHERE item_name =
%s), %s, %s)
                """
                total_price = price * quantity_sold
                self.cursor.execute(sql_sales, (item_name, quantity_sold,
total_price))
                self.db.commit()
                print(f"Stock updated for {item_name}.")
            else:
                print(f"Not enough stock for {item_name}. Available stock:
{current_stock}")
        else:
            print(f"Item '{item_name}' not found.")

    def display_inventory(self):
        sql = "SELECT * FROM inventory"
        self.cursor.execute(sql)
        items = self.cursor.fetchall()
        print("--- Inventory ---")
        for item in items:
            print(f"Item ID: {item[0]}, Name: {item[1]}, Price:
${item[2]:.2f}, Stock: {item[3]}")
```

```
import mysql.connector
from auth import Auth
from items import BakeryItem
from reports import Reports

def main():
    # Database Connection
    db = mysql.connector.connect(
        host="localhost",
        user="root",
        password="shreya93@",
        database="bakery_management"
    )

    auth = Auth(db)
    items = BakeryItem(db)
    reports = Reports(db)

    user_role = None
    while True:
        print("\n--- Bakery Management ---")
        print("1. Register")
        print("2. Login")
        print("3. Add Item (Manager only)")
        print("4. Update Stock")
        print("5. Generate Sales Report")
        print("6. Display Inventory")
        print("7. Exit")

        choice = input("Select an option: ")

        if choice == '1':
            username = input("Username: ")
            password = input("Password: ")
            role = input("Role (cashier/manager): ").lower()
            auth.register(username, password, role)

        elif choice == '2':
            username = input("Username: ")
            password = input("Password: ")
            user_role = auth.login(username, password)

        elif choice == '3':
            if user_role == 'manager':
                name = input("Item Name: ")
                price = float(input("Item Price: "))
                stock = int(input("Item Stock: "))
                items.add_to_db(name, price, stock)
            else:
                print("Access Denied: Only managers can add items.")

        elif choice == '4':
            item_name = input("Item Name: ")
            quantity_sold = int(input("Quantity Sold: "))
            items.update_stock(item_name, quantity_sold)

        elif choice == '5':
            period = input("Enter report period (daily/weekly/monthly):
").lower()
            reports.generate_report(period)

        elif choice == '6':
            items.display_inventory()
        elif choice == '7':
            print("Exiting system. Goodbye!")
            db.close()
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

```
class Reports:
    def __init__(self, db):
        self.db = db
        self.cursor = db.cursor()

    def add_sale(self, item_name, quantity, price):
        sql = "INSERT INTO sales (item_id, quantity_sold, total_price)
VALUES ((SELECT item_id FROM inventory WHERE item_name = %s), %s, %s)"
        self.cursor.execute(sql, (item_name, quantity, price * quantity))
        self.db.commit()
        print(f"Sale recorded: {quantity} of {item_name} sold.")

    def generate_report(self, period):
        if period == 'daily':
            sql = "SELECT item_name, quantity_sold, total_price FROM sales
JOIN inventory ON sales.item_id = inventory.item_id WHERE DATE(sale_date)
= CURDATE()"
        elif period == 'weekly':
            sql = "SELECT item_name, quantity_sold, total_price FROM sales
JOIN inventory ON sales.item_id = inventory.item_id WHERE
YEARWEEK(sale_date, 1) = YEARWEEK(CURDATE(), 1)"
        elif period == 'monthly':
            sql = "SELECT item_name, quantity_sold, total_price FROM sales
JOIN inventory ON sales.item_id = inventory.item_id WHERE MONTH(sale_date)
= MONTH(CURDATE())"
        else:
            print("Invalid period. Choose daily, weekly, or monthly.")
            return

        self.cursor.execute(sql)
        report = self.cursor.fetchall()
        print(f"--- {period.capitalize()} Sales Report ---")
        for row in report:
            print(f"{row[0]}: Sold {row[1]} @ ${row[2] / row[1]:.2f}
each")
        print(f"Total Sales: ${sum(row[2] for row in report):.2f}")
```