

Assignment 1

BN232



Gagan Shrestha

MIT235219

31/07/2025

Table of Contents

[1. Design Explanation 2](#_Toc204896919)

[1.1 Workflow 3](#_Toc204896920)

[2. OS concept used 4](#_Toc204896921)

[3. Python Concepts Used 4](#_Toc204896922)

[4. Socket programming 5](#_Toc204896923)

[5. Challenges and Reflection 5](#_Toc204896924)

[References 6](#_Toc204896925)

# Design Explanation

A screenshot of a computer screen

AI-generated content may be incorrect.

The SmartCafe ordering system is a simulation of the digital order processing in a cafe where the site and the client communicate through the client-server socket in Python [1]. The systems are made up of three modules:

* Meny.py:

stores the menu (dictionary) in one location and displays it in another location (in this case, a separate files).

* Customer.py:

The structure of the program customer.py is that it takes the user input and places the order in the kitchen by means of the sockets.

* kitchen.py:

Receive, process, and acknowledgement of orders with a simulated delay.

## Workflow

A screenshot of a computer program

AI-generated content may be incorrect.

Figure1 of the work flow

A screenshot of a computer

AI-generated content may be incorrect.

Figure2 of the work flow

Customer Interaction:

* Prints the menu (display\_menu()).
* Takes user input (case insensitive, quantity validates).
* Transmits the order as a JSON through the sockets.

Kitchen Processing:

* Listens to the listening port 65432 to receive new orders.
* Computes the total price (computes.py).
* With time.sleep(2), the food is simulated to be prepared.

Communication:

* TCP sockets offer reliability with respect to order transmission.
* The result of JSON serialization is a structured data exchange.

# OS concept used

|  |  |
| --- | --- |
| Concept | Implementation |
| Inter-process Communication (IPC) | TCP sockets (localhost:65432) |
| Process Scheduling | Simulated delay (time.sleep(2)) |
| Resource Management [1] | with statements for auto-closing sockets |

# Python Concepts Used

Key Implementations

* 1. Data Structures (menu.py):

python

menu = {

"Coffee": 5.0, # Dictionary for O(1) price lookup

"Burger": 8.0 # Ensures consistency

}

* 1. Error Handling (customer.py):

python

try:

quantity = int(input("How many? ")) # Prevents crashes

except ValueError:

print("Invalid input!")

* 1. JSON Serialization (customer.py & kitchen.py):

python

# Sending: customer.py (Line 26)

s.sendall(json.dumps(order).encode())

# Receiving: kitchen.py (Line 15)

order = json.loads(data.decode())

Socket Programming (socket module):

Customer: s.connect(('localhost', 65432))

Kitchen: s.bind(('localhost', 65432))

# Socket programming

A diagram of a system

AI-generated content may be incorrect.

Key Features:

Port 65432: Unprivileged port for local testing.

JSON Format: Ensures structured data transfer.

Reliability: TCP guarantees order delivery [2].

# Challenges and Reflection

* 1. Challenges faced

1. Input validation like the case mismatch coffee vs Coffee

Solution Fix .title() conversion in the customer.py code in the line 10

1. Price Synchronization

Solution: Centralized menu.py ensured consistent pricing.

1. Socket Conflicts

Solution: Used SO\_REUSEADDR in final implementation

Lessons Learned:

"Centralized configuration files reduce maintenance overhead"

# References

|  |  |
| --- | --- |
| [1] | R. N.Cardinal, “Whisker: A client—server high-performance multimedia research control system,” *Behavior Research Methods,* vol. 42, no. 4, pp. 1059-1071, NOvember 2010. |
| [2] | D. Huber, “Towards Dynamic Resource Management with MPI Sessions and PMIx,” *HAL (Le Centre pour la Communication Scientifique Directe),* 2022. |
| [3] | M. Donahoo, TCP/IP Socket in Python, 2016. |