

# Report: Pide Shop Project

ÇAĞRI YILDIZ 1901042630

CSE\_344\_FINAL

## Introduction

The Pide Shop project simulates a food production and delivery system using a multi-threaded server architecture. The system efficiently handles orders from clients, processes them through various stages, and ensures timely delivery by managing multiple threads and synchronization mechanisms.

## System Design

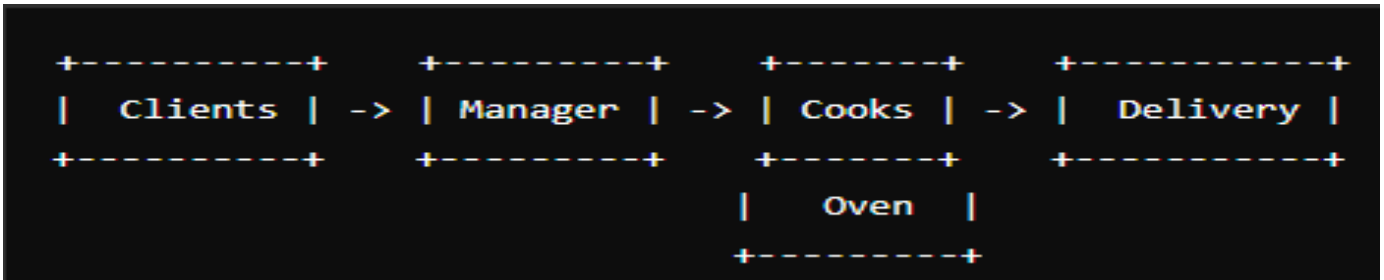
### Components

1. **Manager:** Manages orders received via phone, assigns them to available cooks.
2. **Cooks:** Prepare the meal, simulate the preparation time by calculating the pseudo-inverse of a 30x40 complex matrix, then cook the meal for half the preparation time.
3. **Oven:** Holds up to 6 meals, has 2 openings for placing and removing meals.
4. **Delivery Personnel:** Deliver meals and return to the shop. Each delivery person can carry up to 3 meals.
5. **Clients:** Place orders to the server.

## Design Decisions

1. **Multithreading:** Separate thread pools for cooks and delivery personnel to ensure efficient processing.
2. **Synchronization:** Mutexes and condition variables to manage shared resources.
3. **Signal Handling:** Graceful shutdown and order cancellation using signal handlers.
4. **Logging:** Maintain logs for tracking activities and debugging.

## Diagram



## Implementation Details

### Manager

The manager receives orders from clients, assigns them to available cooks, and signals when orders are ready for delivery.

```
32 void start_manager(void) {
33     printf("Initializing manager...\n");
34     pthread_t thread;
35     if (pthread_create(&thread, NULL, manager_thread, NULL) != 0) {
36         perror("Failed to create manager thread");
37         exit(EXIT_FAILURE);
38     }
39     pthread_detach(thread);
40     printf("Manager initialized...\n");
41 }
```

```
81 void manager_receive_order() {
82     pthread_mutex_lock(&manager_mutex);
83     order_received++;
84     pthread_cond_signal(&manager_cond);
85     pthread_mutex_unlock(&manager_mutex);
86
87     char message[256];
88     snprintf(message, sizeof(message), "Manager received an order");
89     log_message(message);
90 }
```

### Cooks

Cooks prepare meals by computing the pseudo-inverse of a 30x40 complex matrix, then cook the meal for half the preparation time.

```
57 // Thread function for each cook
58 void *cook_thread(void *arg) {
59     Cook *cook = (Cook *)arg;
60 }
```

```
111 // Function to compute the pseudo-inverse of a 30x40 matrix with complex elements
112 void compute_pseudo_inverse(void) {
113     int m = 30, n = 40;
114     double complex A[m][n];
115     double complex B[n][m];
```

## Oven

The oven holds up to 6 meals and has 2 openings for placing and removing meals.

```
16  typedef struct {
17      int num_aparatus; // Number of available oven aparatus
18      int num_meals;    // Number of meals currently in the oven
19      pthread_mutex_t mutex;
20      pthread_cond_t cond_aparatus;
21      pthread_cond_t cond_meals;
22  } Oven;
23
24  #define MAX_APARATUS 3
25  #define MAX_MEALS 6
```

## Delivery Personnel

Delivery personnel receive prepared meals from the manager, deliver them based on customer addresses, and return to the shop.

```
85  void *delivery_thread(void *arg) {
86      DeliveryPerson *person = (DeliveryPerson *)arg;
87  }
```

```
120  /*
121   * Simulate the delivery process by sleeping for the calculated time based on distance
122   * Side effects:
123   * - Uses sleep to simulate delivery time.
124   */
125  void simulate_delivery(float x, float y, float velocity) {
126      float distance = sqrt(x * x + y * y);
127      int sleepTime = (int)(distance / velocity);
128      sleep(sleepTime);
129  }
```

## Multithreaded Internet Server

The server implements thread pools for cooks and delivery personnel.

```
54  void start_server(const char *ip_address, int port) {
55      int socket_desc, client_sock, c;
56      struct sockaddr_in server, client;
```

## Signal Handling

The server handles signals for order cancellations and shop closure.

```
184 void signal_handler(int sig) {
185     log_message("Signal received, shutting down...");
186     cancel_all_orders();
187     write_log_file();
188     log_message("Log file written");
189     exit(0);
190 }
```

## Logging

Logs are created to record shop activities and order states.

```
10 // Logging function
11 void log_message(const char *message) {
12     time_t now;
13     time(&now);
14     char timestamp[20];
15     strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", localtime(&now));
16     printf("[%s] %s\n", timestamp, message);
17     FILE *log_file = fopen("pide_shop_log.txt", "a");
18     if (log_file) {
19         fprintf(log_file, "[%s] %s\n", timestamp, message);
20         fclose(log_file);
21     } else {
22         perror("Failed to open log file");
23     }
24 }
```

## Server and Client Interaction

### Server Command

The server is started with the following command:

```
31 # Run server with specified arguments
32 run_server: server
33     ./server 127.0.0.1 6 6 1000
34
```

### Client Command

Clients generate orders with the following command:

```
27 # Run client with specified arguments
28 v run_client: client
29     ./client 127.0.0.1 600 6 8
```

## Additional Details

### Customer Location

Customers are located relative to the mayor's office south entrance (0,0).

### Order Cancellation

Orders can be canceled at any stage of preparation, cooking, or delivery.

```
98 void cancel_order() {
99     pthread_mutex_lock(&manager_mutex);
100     order_received = 0;
101     order_ready = 0;
102     pthread_cond_broadcast(&manager_cond);
103     pthread_mutex_unlock(&manager_mutex);
104
105     log_message("All orders cancelled");
106 }
```

### Server Shutdown

The server handles ^C and ^D signals for orderly shutdown.

```
226 signal(SIGINT, signal_handler);
227 signal(SIGTERM, signal_handler);
228 signal(SIGPIPE, SIG_IGN); // Ignore SIGPIPE signal to prevent crashes on broken pipe
229
```

## Implementation Steps

### Step-by-Step Execution

#### 1. Server Initialization:

- The server initializes by parsing command line arguments to get the IP address, port number, cook thread pool size, delivery thread pool size, and delivery speed.
- Signal handlers are set up to ensure graceful shutdown.

#### 2. Starting Thread Pools:

- The server starts the thread pools for cooks and delivery personnel.

#### 3. Accepting Client Connections:

- The server listens for incoming client connections and creates a new thread to handle each client.

4. Client Handler:

- Each client handler thread receives orders from the client, processes them, and sends responses back to the client.

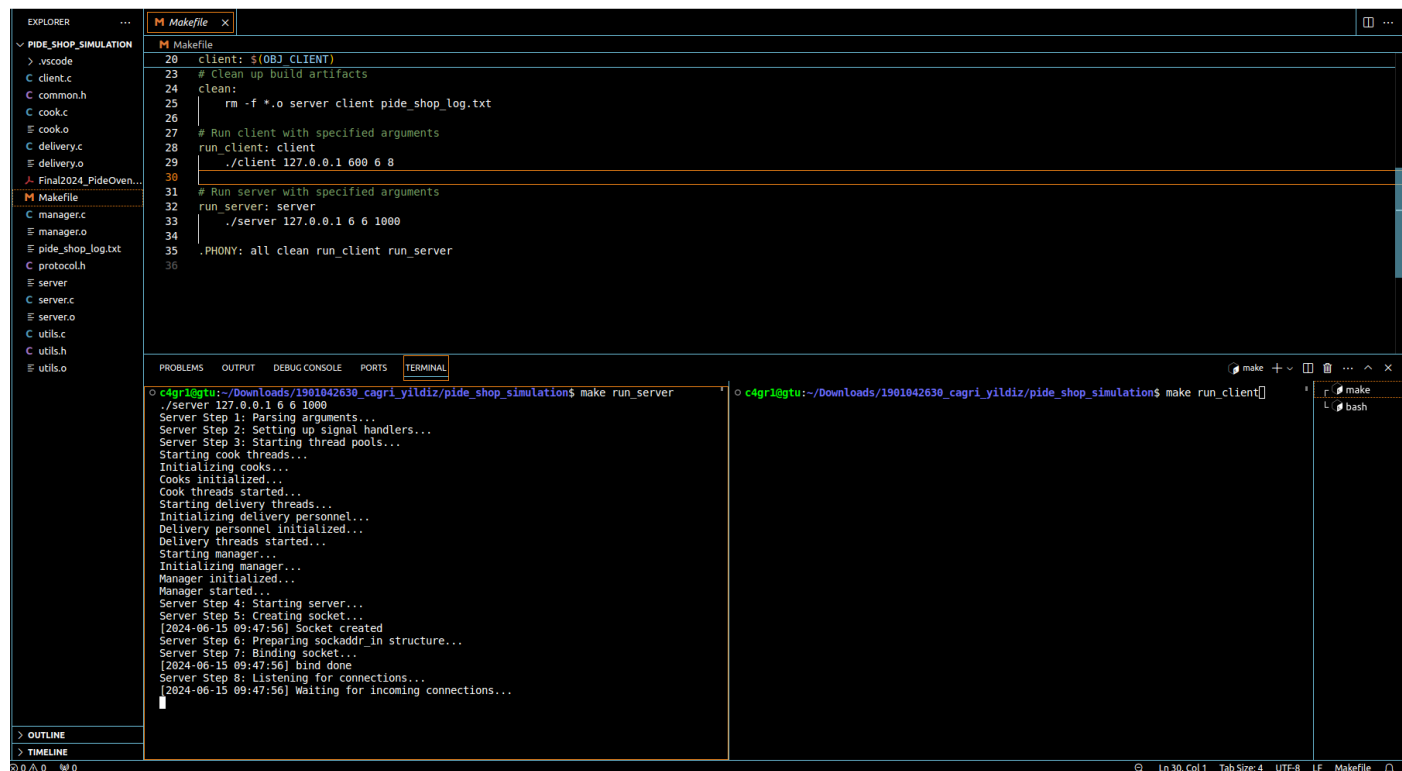
5. Order Processing by Cooks:

- Cooks process the orders by simulating meal preparation and cooking.

6. Meal Delivery:

- Delivery personnel deliver the meals to the customers and return to the shop.

# Screenshots



The screenshot shows the VS Code interface with the Makefile editor open. The Makefile contains rules for building and running the client and server. The terminal window shows the output of the 'make run\_client' command, which includes the compilation of client.o and the execution of the client program. The client program sends messages to the server, and the server responds with estimated delivery times.

```
client: $(OBJ_CLIENT)
# clean up build artifacts
clean:
    rm -f *.o server client pide_shop_log.txt

# Run client with specified arguments
run_client: client
    ./client 127.0.0.1 600 6 8

# Run server with specified arguments
run_server: server
    ./server 127.0.0.1 6 6 1000

.PHONY: all clean run_client run_server
```

```
[2024-06-15 09:48:24] Manager: Order is ready for delivery
[2024-06-15 09:48:24] Order 1 is delivering by deliver 0 to address (1.00, 4.00)
[2024-06-15 09:48:24] Order 1 is delivered by deliver 0 to address (1.00, 4.00)
Delivery person 0 completed delivery.
[2024-06-15 09:48:25] Order 2 is being cooked by cooker 3
[2024-06-15 09:48:25] Received order from client
[2024-06-15 09:48:25] Order processed and response sent to client
[2024-06-15 09:48:25] Manager received an order
[2024-06-15 09:48:26] Preparing order 3 by cooker 4
[2024-06-15 09:48:26] Order 2 is cooked by cooker 3
[2024-06-15 09:48:26] Manager: Order is ready for delivery
[2024-06-15 09:48:26] Order 2 is delivering by deliver 1 to address (2.00, 5.00)
[2024-06-15 09:48:26] Order 2 is delivered by deliver 1 to address (2.00, 5.00)
Delivery person 1 completed delivery.
[2024-06-15 09:48:27] Order 3 is being cooked by cooker 4
[2024-06-15 09:48:28] Received order from client
[2024-06-15 09:48:28] Order processed and response sent to client
[2024-06-15 09:48:28] Manager received an order
[2024-06-15 09:48:28] Preparing order 4 by cooker 5
[2024-06-15 09:48:28] Order 3 is cooked by cooker 4
[2024-06-15 09:48:28] Manager: Order is ready for delivery
[2024-06-15 09:48:28] Order 3 is delivering by deliver 2 to address (1.00, 3.00)
[2024-06-15 09:48:28] Order 3 is delivered by deliver 2 to address (1.00, 3.00)
Delivery person 2 completed delivery.
[2024-06-15 09:48:30] Order 4 is being cooked by cooker 5
[2024-06-15 09:48:30] Received order from client
[2024-06-15 09:48:30] Order processed and response sent to client
[2024-06-15 09:48:30] Manager received an order
[2024-06-15 09:48:30] Preparing order 5 by cooker 1
```

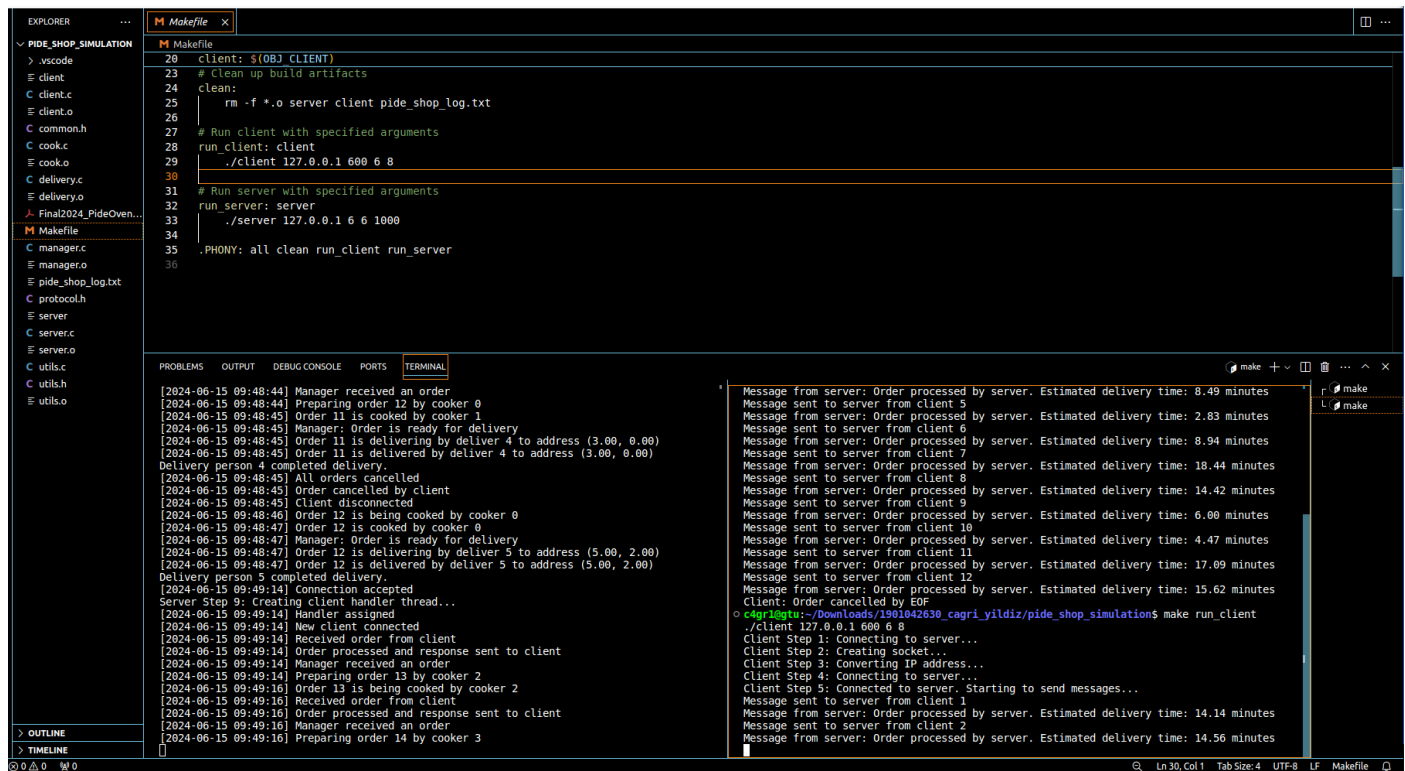
## After ctrl+d in client side

The screenshot shows the VS Code interface with the Makefile editor open. The terminal window shows the output of the 'make run\_client' command, which includes the compilation of client.o and the execution of the client program. The client program sends messages to the server, and the server responds with estimated delivery times. The output continues from the previous screenshot, showing the completion of order 5 and the cancellation of all orders.

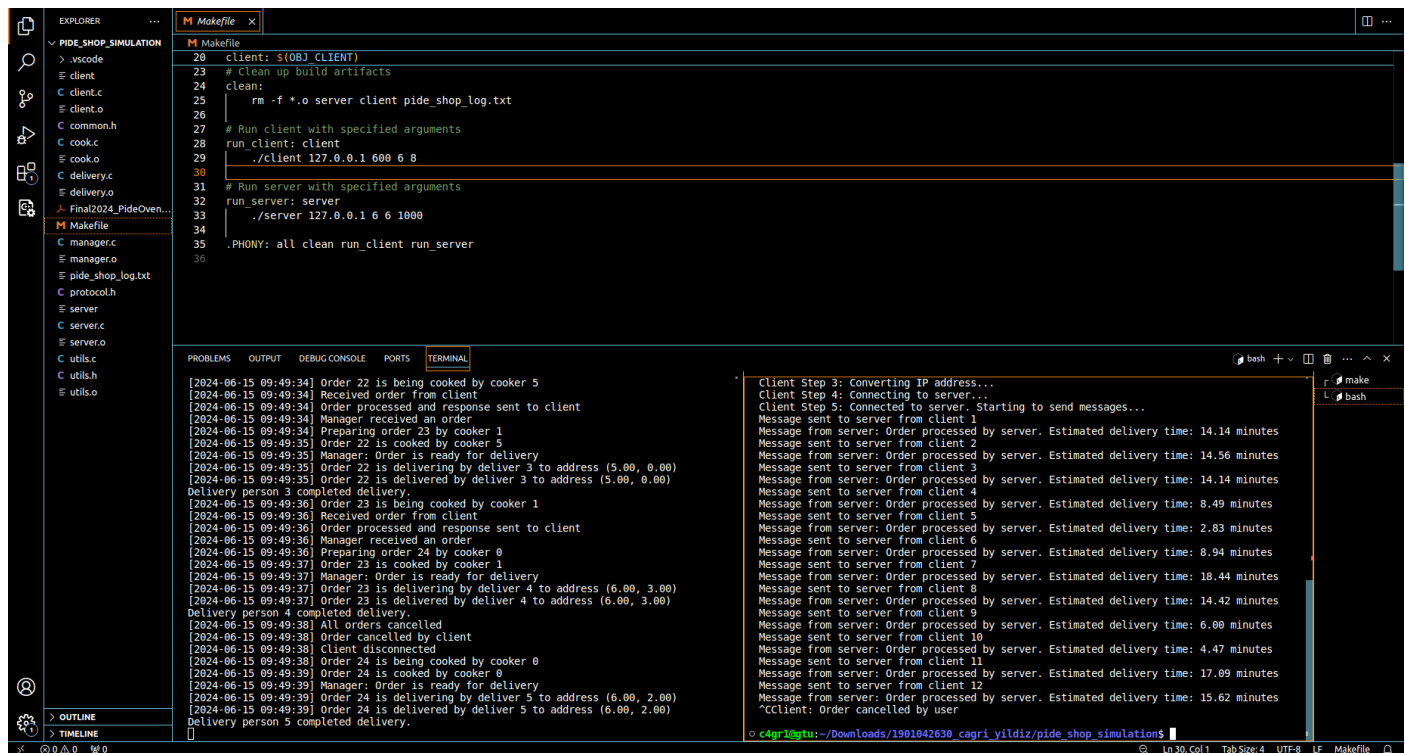
```
Client Step 2: Creating socket...
Client Step 3: Converting IP address...
Client Step 4: Connecting to server...
Client Step 5: Connected to server. Starting to send messages...
Message sent to server from client 1
Message from server: Order processed by server. Estimated delivery time: 14.14 minutes
Message sent to server from client 2
Message from server: Order processed by server. Estimated delivery time: 14.56 minutes
Message sent to server from client 3
Message from server: Order processed by server. Estimated delivery time: 14.14 minutes
Message sent to server from client 4
Message from server: Order processed by server. Estimated delivery time: 8.49 minutes
Message sent to server from client 5
Message from server: Order processed by server. Estimated delivery time: 2.83 minutes
Message sent to server from client 6
Message from server: Order processed by server. Estimated delivery time: 8.94 minutes
Message sent to server from client 7
Message from server: Order processed by server. Estimated delivery time: 18.44 minutes
Message sent to server from client 8
Message from server: Order processed by server. Estimated delivery time: 14.42 minutes
Message sent to server from client 9
Message from server: Order processed by server. Estimated delivery time: 6.00 minutes
Message sent to server from client 10
Message from server: Order processed by server. Estimated delivery time: 4.47 minutes
Message sent to server from client 11
Message from server: Order processed by server. Estimated delivery time: 17.09 minutes
Message sent to server from client 12
Message from server: Order processed by server. Estimated delivery time: 15.62 minutes
Client: Order cancelled by EOF
```



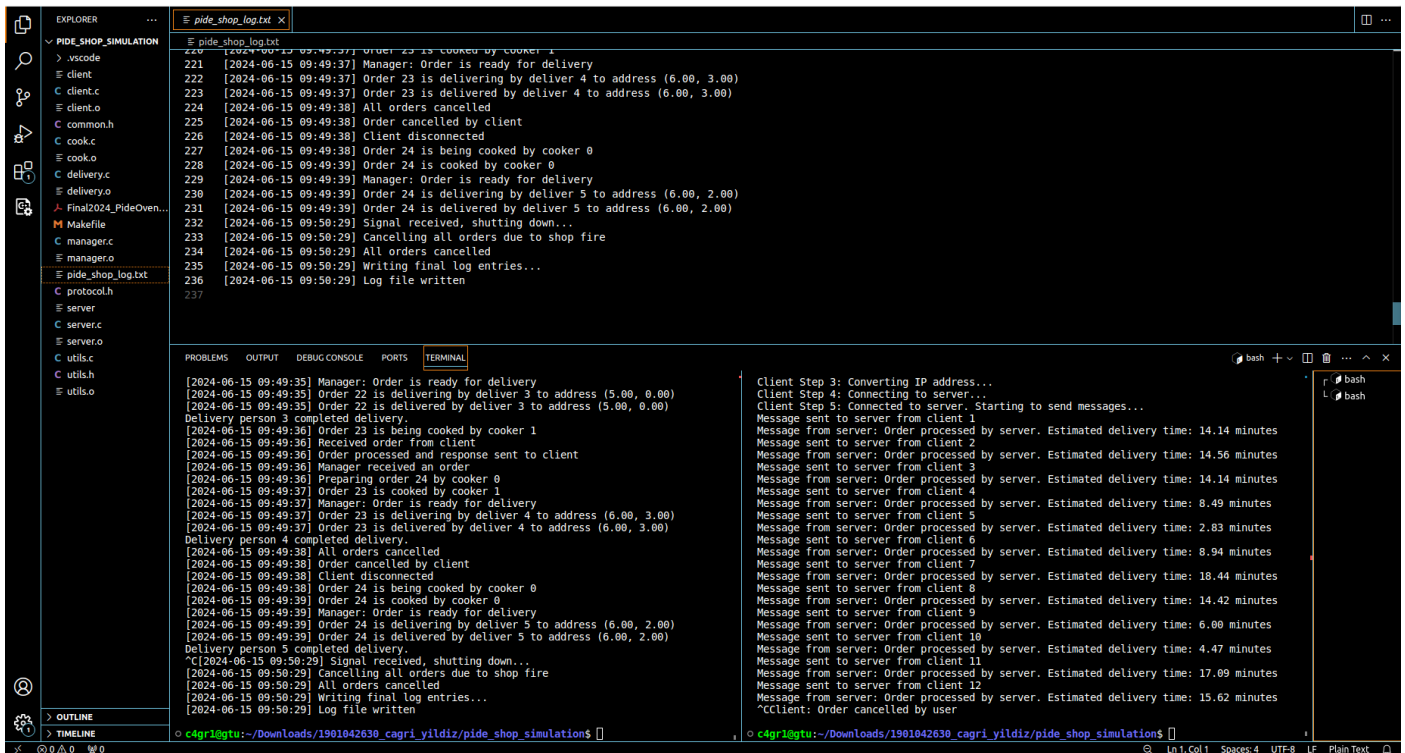
# Adding new orders



# After ctrl+c in client side



## After ctrl + c in server side and log.txt



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of the 'PIDE\_SHOP\_SIMULATION' project. The 'pide\_shop\_log.txt' file is selected and its contents are displayed in the main editor. The log file contains a detailed record of the simulation, including order processing, delivery status, and cancellation events. The terminal at the bottom shows the output of the simulation, which ends with a signal received and the simulation shutting down. The terminal also displays the estimated delivery times for various orders.

```
# pide_shop_log.txt
220 [2024-06-15 09:49:37] Order 23 is cooked by cooker 1
221 [2024-06-15 09:49:37] Manager: Order is ready for delivery
222 [2024-06-15 09:49:37] Order 23 is delivered by deliver 4 to address (6.00, 3.00)
223 [2024-06-15 09:49:37] Order 23 is delivered by deliver 4 to address (6.00, 3.00)
224 [2024-06-15 09:49:38] All orders cancelled
225 [2024-06-15 09:49:38] Order cancelled by client
226 [2024-06-15 09:49:38] Client disconnected
227 [2024-06-15 09:49:38] Order 24 is being cooked by cooker 0
228 [2024-06-15 09:49:39] Order 24 is cooked by cooker 0
229 [2024-06-15 09:49:39] Manager: Order is ready for delivery
230 [2024-06-15 09:49:39] Order 24 is delivered by deliver 5 to address (6.00, 2.00)
231 [2024-06-15 09:49:39] Order 24 is delivered by deliver 5 to address (6.00, 2.00)
232 [2024-06-15 09:50:29] Signal received, shutting down...
233 [2024-06-15 09:50:29] Cancelling all orders due to shop fire
234 [2024-06-15 09:50:29] All orders cancelled
235 [2024-06-15 09:50:29] Writing final log entries...
236 [2024-06-15 09:50:29] Log file written
237
```

```
[2024-06-15 09:49:35] Manager: Order is ready for delivery
[2024-06-15 09:49:35] Order 22 is delivering by deliver 3 to address (5.00, 0.00)
[2024-06-15 09:49:35] Order 22 is delivered by deliver 3 to address (5.00, 0.00)
Delivery person 1 completed delivery.
[2024-06-15 09:49:36] Order 23 is being cooked by cooker 1
[2024-06-15 09:49:36] Received order from client
[2024-06-15 09:49:36] Order processed and response sent to client
[2024-06-15 09:49:36] Manager received an order
[2024-06-15 09:49:36] Preparing order 24 by cooker 0
[2024-06-15 09:49:37] Order 23 is cooked by cooker 1
[2024-06-15 09:49:37] Manager: Order is ready for delivery
[2024-06-15 09:49:37] Order 23 is delivering by deliver 4 to address (6.00, 3.00)
[2024-06-15 09:49:37] Order 23 is delivered by deliver 4 to address (6.00, 3.00)
Delivery person 4 completed delivery.
[2024-06-15 09:49:38] All orders cancelled
[2024-06-15 09:49:38] Order cancelled by client
[2024-06-15 09:49:38] Client disconnected
[2024-06-15 09:49:38] Order 24 is being cooked by cooker 0
[2024-06-15 09:49:39] Order 24 is cooked by cooker 0
[2024-06-15 09:49:39] Manager: Order is ready for delivery
[2024-06-15 09:49:39] Order 24 is delivering by deliver 5 to address (6.00, 2.00)
[2024-06-15 09:49:39] Order 24 is delivered by deliver 5 to address (6.00, 2.00)
Delivery person 5 completed delivery.
^C[2024-06-15 09:50:29] Signal received, shutting down...
[2024-06-15 09:50:29] Cancelling all orders due to shop fire
[2024-06-15 09:50:29] All orders cancelled
[2024-06-15 09:50:29] Writing final log entries...
[2024-06-15 09:50:29] Log file written

Client Step 3: Converting IP address...
Client Step 4: Connecting to server...
Client Step 5: Connected to server. Starting to send messages...
Message sent to server from client 1
Message from server: Order processed by server. Estimated delivery time: 14.14 minutes
Message sent to server from client 2
Message from server: Order processed by server. Estimated delivery time: 14.56 minutes
Message sent to server from client 3
Message from server: Order processed by server. Estimated delivery time: 14.14 minutes
Message sent to server from client 4
Message from server: Order processed by server. Estimated delivery time: 8.49 minutes
Message sent to server from client 5
Message from server: Order processed by server. Estimated delivery time: 2.83 minutes
Message sent to server from client 6
Message from server: Order processed by server. Estimated delivery time: 8.94 minutes
Message sent to server from client 7
Message from server: Order processed by server. Estimated delivery time: 14.42 minutes
Message sent to server from client 8
Message from server: Order processed by server. Estimated delivery time: 18.44 minutes
Message sent to server from client 9
Message from server: Order processed by server. Estimated delivery time: 6.00 minutes
Message sent to server from client 10
Message from server: Order processed by server. Estimated delivery time: 4.47 minutes
Message sent to server from client 11
Message from server: Order processed by server. Estimated delivery time: 17.09 minutes
Message sent to server from client 12
Message from server: Order processed by server. Estimated delivery time: 15.62 minutes
^CClient: Order cancelled by user
```

## I update the makefile with lower values



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of the 'PIDE\_SHOP\_SIMULATION' project. The 'Makefile' file is selected and its contents are displayed in the main editor. The Makefile contains rules for building and running the simulation. The terminal at the bottom shows the output of the 'make' command, which successfully builds and runs the simulation.

```
20 client: $(OBJ) CLIENT)
23 # Clean up build artifacts
24 clean:
25     rm -f *.o server client pide_shop_log.txt
26
27 # Run client with specified arguments
28 run_client: client
29     ./client 127.0.0.1 6 6 8
30
31 # Run server with specified arguments
32 run_server: server
33     ./server 127.0.0.1 3 3 10
34
35 .PHONY: all clean run_client run_server
36
```

```
c4gri@gtu:~/Downloads/1901042630_cagri_yildiz/pide_shop_simulation$ make
rm -f *.o server client pide_shop_log.txt
./client 127.0.0.1 6 6 8
./server 127.0.0.1 3 3 10
```

The screenshot shows the VS Code interface with the `Makefile` open in the editor. The `Makefile` contains rules for building and running the simulation. The terminal output shows the execution of `make run_client`, which runs the `client` program. The output displays the client's steps: connecting to the server, creating a socket, converting IP address, connecting to the server, and sending messages. It also shows the server's response: "Message from server: Order processed by server. Estimated delivery time: 14.56 minutes".

```
20 client: $(OBJ_CLIENT)
21
22 # Clean up build artifacts
23 clean:
24     rm -f *.o server client pidi_shop_log.txt
25
26 # Run client with specified arguments
27 run_client: client
28     ./client 127.0.0.1 6 6 8
29
30 # Run server with specified arguments
31 run_server: server
32     ./server 127.0.0.1 3 3 10
33
34 .PHONY: all clean run_client run_server
```

```
Delivery threads started...
Starting manager...
Initializing manager...
Manager initialized...
Manager started...
Server Step 4: Starting server...
Server Step 5: Creating socket...
[2024-06-15 09:52:00] Socket created
Server Step 6: Preparing sockaddr_in structure...
Server Step 7: Binding socket...
[2024-06-15 09:52:00] bind done
Server Step 8: Listening for connections...
[2024-06-15 09:52:00] Waiting for incoming connections...
[2024-06-15 09:52:04] Connection accepted
Server Step 9: Creating client handler thread...
[2024-06-15 09:52:04] Handler assigned
[2024-06-15 09:52:04] New client connected
[2024-06-15 09:52:04] Received order from client
[2024-06-15 09:52:04] Order processed and response sent to client
[2024-06-15 09:52:04] Manager received an order
[2024-06-15 09:52:04] Preparing order 1 by cooker 2
[2024-06-15 09:52:06] Order 1 is being cooked by cooker 2
[2024-06-15 09:52:06] Received order from client
[2024-06-15 09:52:06] Order processed and response sent to client
[2024-06-15 09:52:06] Manager received an order
[2024-06-15 09:52:06] Preparing order 2 by cooker 1
[2024-06-15 09:52:07] Order 1 is cooked by cooker 2
[2024-06-15 09:52:07] Manager: Order is ready for delivery
[2024-06-15 09:52:07] Order 1 is delivering by deliver 0 to address (3.00, 2.00)
```

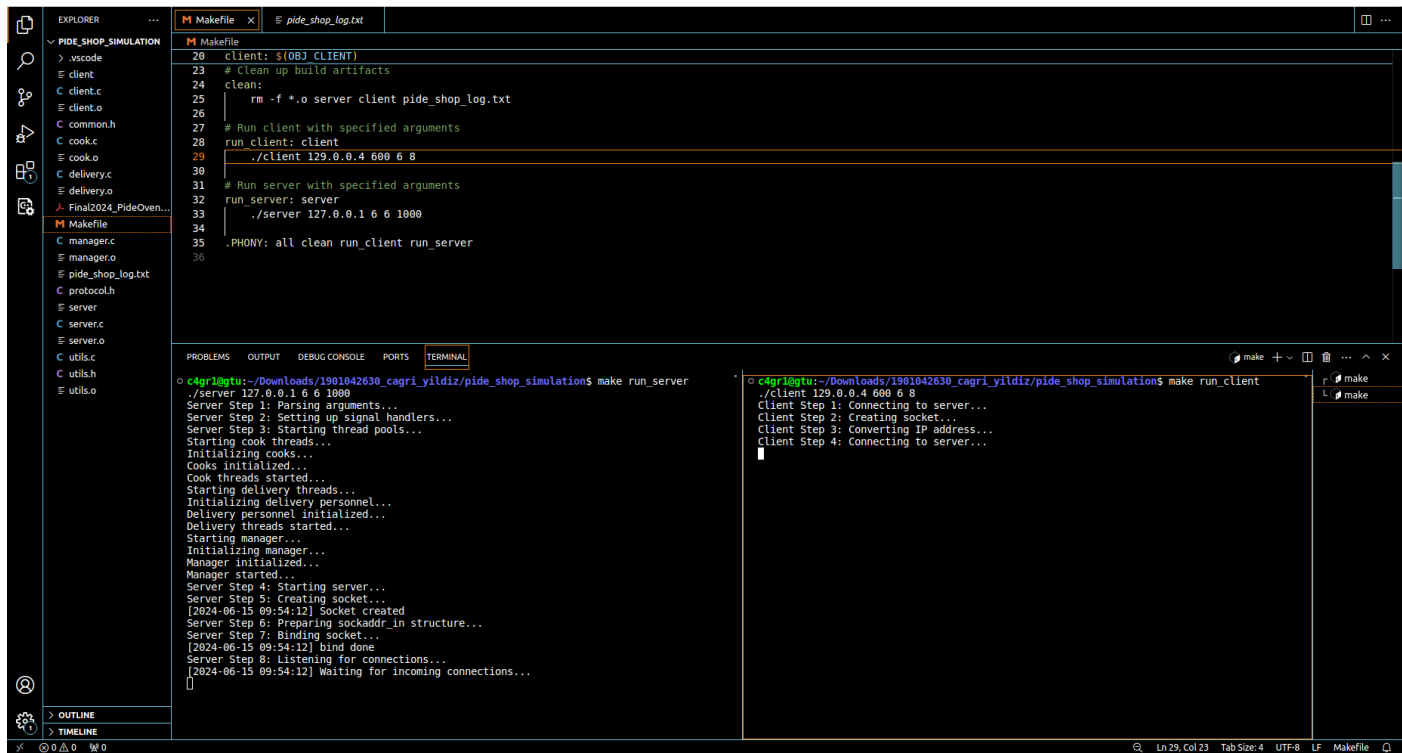
After the all process.

The screenshot shows the VS Code interface with the `pidi_shop_log.txt` file open in the editor. The log file contains a detailed record of the simulation's execution, including the server's status, the client's actions, and the delivery process. The terminal output shows the execution of `make run_client`, which runs the `client` program. The output displays the client's steps: connecting to the server, creating a socket, converting IP address, connecting to the server, and sending messages. It also shows the server's response: "Message from server: Order processed by server. Estimated delivery time: 14.56 minutes".

```
286 [2024-06-15 09:52:10] Client disconnected
287 [2024-06-15 09:52:17] Order 6 is cooked by cooker 0
288 [2024-06-15 09:52:17] Manager: Order is ready for delivery
289 [2024-06-15 09:52:27] Order 2 is delivered by deliver 1 to address (1.00, 3.00)
290 [2024-06-15 09:52:27] Order 4 is delivering by deliver 1 to address (2.00, 0.00)
291 [2024-06-15 09:52:28] Order 1 is delivered by deliver 0 to address (3.00, 2.00)
292 [2024-06-15 09:52:28] Order 5 is delivering by deliver 0 to address (1.00, 1.00)
293 [2024-06-15 09:52:29] Order 3 is delivered by deliver 2 to address (1.00, 3.00)
294 [2024-06-15 09:52:29] Order 6 is delivering by deliver 2 to address (2.00, 3.00)
295 [2024-06-15 09:52:36] Order 5 is delivered by deliver 0 to address (1.00, 1.00)
296 [2024-06-15 09:52:39] Order 4 is delivered by deliver 1 to address (2.00, 0.00)
297
```

```
[2024-06-15 09:52:12] Order processed and response sent to client
[2024-06-15 09:52:12] Manager received an order
[2024-06-15 09:52:12] Preparing order 5 by cooker 1
[2024-06-15 09:52:13] Order 4 is cooked by cooker 2
[2024-06-15 09:52:13] Manager: Order is ready for delivery
[2024-06-15 09:52:14] Order 5 is being cooked by cooker 1
[2024-06-15 09:52:14] Received order from client
[2024-06-15 09:52:14] Order processed and response sent to client
[2024-06-15 09:52:14] Manager received an order
[2024-06-15 09:52:14] Preparing order 6 by cooker 0
[2024-06-15 09:52:15] Order 5 is cooked by cooker 1
[2024-06-15 09:52:15] Manager: Order is ready for delivery
[2024-06-15 09:52:16] Order 6 is being cooked by cooker 0
[2024-06-15 09:52:16] Client disconnected
[2024-06-15 09:52:17] Order 6 is cooked by cooker 0
[2024-06-15 09:52:17] Manager: Order is ready for delivery
[2024-06-15 09:52:27] Order 2 is delivered by deliver 1 to address (1.00, 3.00)
Delivery person 1 completed delivery.
[2024-06-15 09:52:27] Order 4 is delivering by deliver 1 to address (2.00, 0.00)
[2024-06-15 09:52:28] Order 1 is delivered by deliver 0 to address (3.00, 2.00)
Delivery person 0 completed delivery.
[2024-06-15 09:52:28] Order 5 is delivering by deliver 0 to address (1.00, 1.00)
[2024-06-15 09:52:29] Order 3 is delivered by deliver 2 to address (1.00, 3.00)
Delivery person 2 completed delivery.
[2024-06-15 09:52:29] Order 6 is delivering by deliver 2 to address (2.00, 3.00)
[2024-06-15 09:52:36] Order 5 is delivered by deliver 0 to address (1.00, 1.00)
Delivery person 0 completed delivery.
[2024-06-15 09:52:39] Order 4 is delivered by deliver 1 to address (2.00, 0.00)
Delivery person 1 completed delivery.
```

# If the client IP is entered incorrectly (can't connect)



## Another test case

