DOCUMENTATION

ASSIGNMENT *Queue Management System*

STUDENT NAME: Gozman-Pop Maria-Eliza
GROUP: 30424

CONTENTS

1. Assignment Objective

The main objective is to simulate a series of N clients arriving for service, entering Q queues, waiting, being served, and finally leaving the queues; and to compute the average waiting time, average service time, and peak hour..

Sub-objectives:

- **Problem Analysis and Requirements Identification:** Analyze requirements for polynomial calculator, including supported operations, input formats, error handling, and GUI specifications.
- **Design Phase:** Create detailed specifications for application architecture, GUI layout, parsing algorithms, mathematical operations, and error handling mechanisms.
- **Implementation:** Translate design specifications into executable code, developing GUI using Java Swing, backend logic for parsing and operations, and integrating components.
- **Testing:** Execute comprehensive testing strategy, including unit tests for individual components, integration tests for system functionality, and user acceptance testing for usability validation.

2. Problem Analysis, Modeling, Scenarios, Use Cases

2.1.

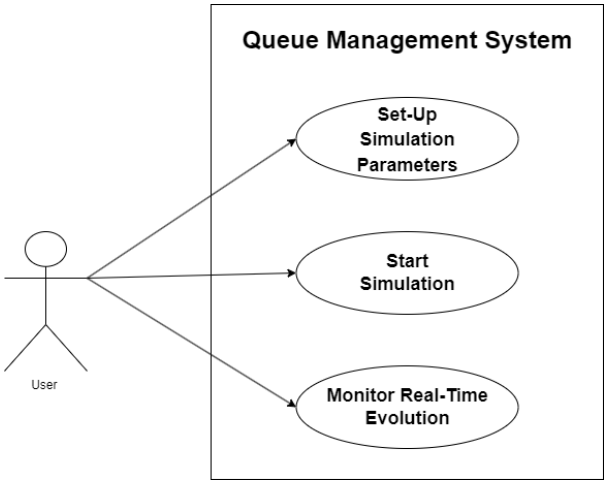
Functional Requirements:

- The Queue Management System should simulate arrival, queuing, service, and departure of clients.
- The Queue Management System should allow users to input simulation parameters.
- The Queue Management System should dynamically manage queues based on selected strategies.
- The Queue Management System should log events with timestamps for analysis.
- The Queue Management System should log events with timestamps for analysis.

Non-functional Requirements:

- The Queue Management System should provide an intuitive user interface.
- The Queue Management System should handle large loads efficiently.
- The Queue Management System should ensure robustness and data integrity.
- The Queue Management System should allow easy adjustment and expansion.
- The Queue Management System should produce precise simulation results.
- The Queue Management System should ensure confidentiality and integrity of data.

2.2. Use Cases:



- User to Set Up Simulation Parameters:
  - the user interacts with input fields to specify parameters such as the number of clients, number of queues, simulation time, arrival time distribution, and service time distribution.
  - the input fields allow the user to input numerical values for each parameter.
- User to Start Simulation:
  - the system provides a button labeled "Start Simulation" for initiating the simulation.
  - the user clicks on the "Start Simulation" button to begin the simulation process.
- User to Monitor Real-Time Queue Evolution:
  - during the simulation, the system continuously updates and displays the status of each queue in real-time.
  - the user observes the changes in queue status, including the number of clients in each queue and their respective wait times.
  - the queue evolution is presented visually in a designated area of the interface, providing a clear representation of the simulation progress.

3. Design

3.1.

The Queue Management System exemplifies various principles of object-oriented programming (OOP), fostering a structured and adaptable design:

- **Abstraction:** Abstraction is evident in the definition of classes such as Task, Server, and Strategy. These classes abstract real-world entities and concepts related to queuing systems, providing clear interfaces for interaction while hiding internal complexities. For example, the

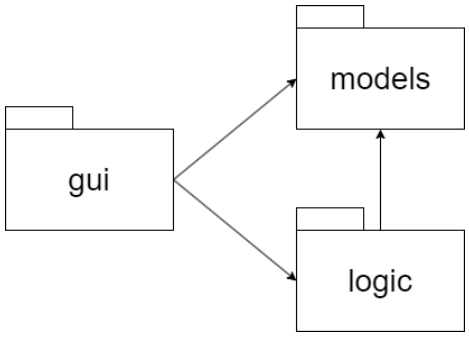
Task class encapsulates attributes like arrival time and service time, abstracting the concept of a task in the queueing system.

- **Encapsulation:** Encapsulation is fundamental to the design of the system, with each class encapsulating its data and behavior within well-defined boundaries. For instance, the Task class encapsulates attributes and methods related to tasks in the queue, ensuring data integrity and promoting modular design. Encapsulation also facilitates information hiding, allowing the internal implementation details of classes to remain hidden from external components.

- **Polymorphism**: Polymorphism is achieved through method overriding and interface implementation. The Strategy interface and its concrete implementations, such as ShortestQueueStrategy and ShortestTimeStrategy, exemplify polymorphic behavior. This allows different queueing strategies to be used interchangeably within the SimulationManager, enhancing flexibility and promoting code extensibility.

- **Modularity:** The codebase is organized into modular packages, each containing related classes and functionalities. For example, the logic package encapsulates core simulation logic, while the gui package handles graphical user interface components. Modularity promotes code organization, readability, and maintainability by separating distinct concerns and reducing dependencies between modules.

3.2. Division into packages

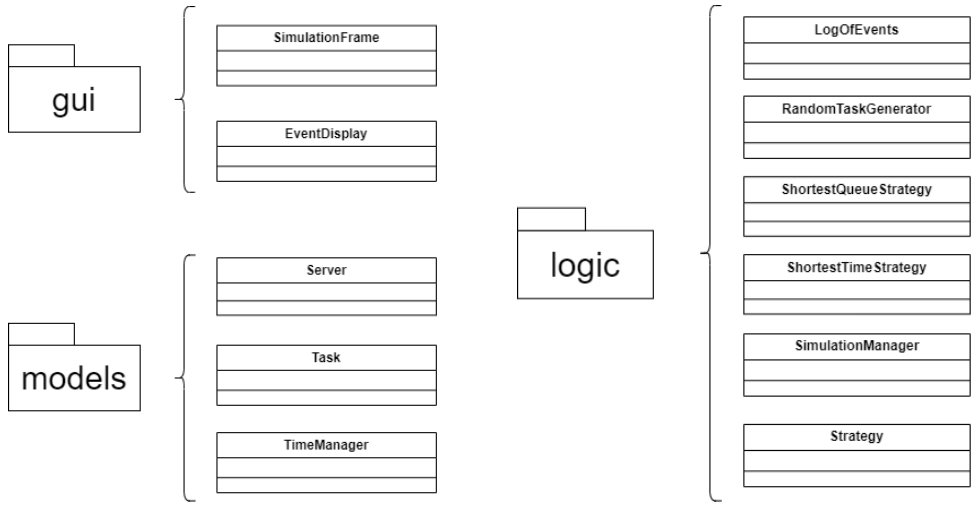


- The **gui** package depends on the logic package because the GUI classes (e.g., SimulationFrame) utilize classes from the **logic** package (e.g., SimulationManager) to orchestrate the simulation process, handle user input, and initiate simulation events.

- Additionally, the **gui** package depends on the **models** package because GUI components interact with the data model classes (e.g., Task, Server) to display simulation-related information, such as client tasks, server status, and queue evolution.

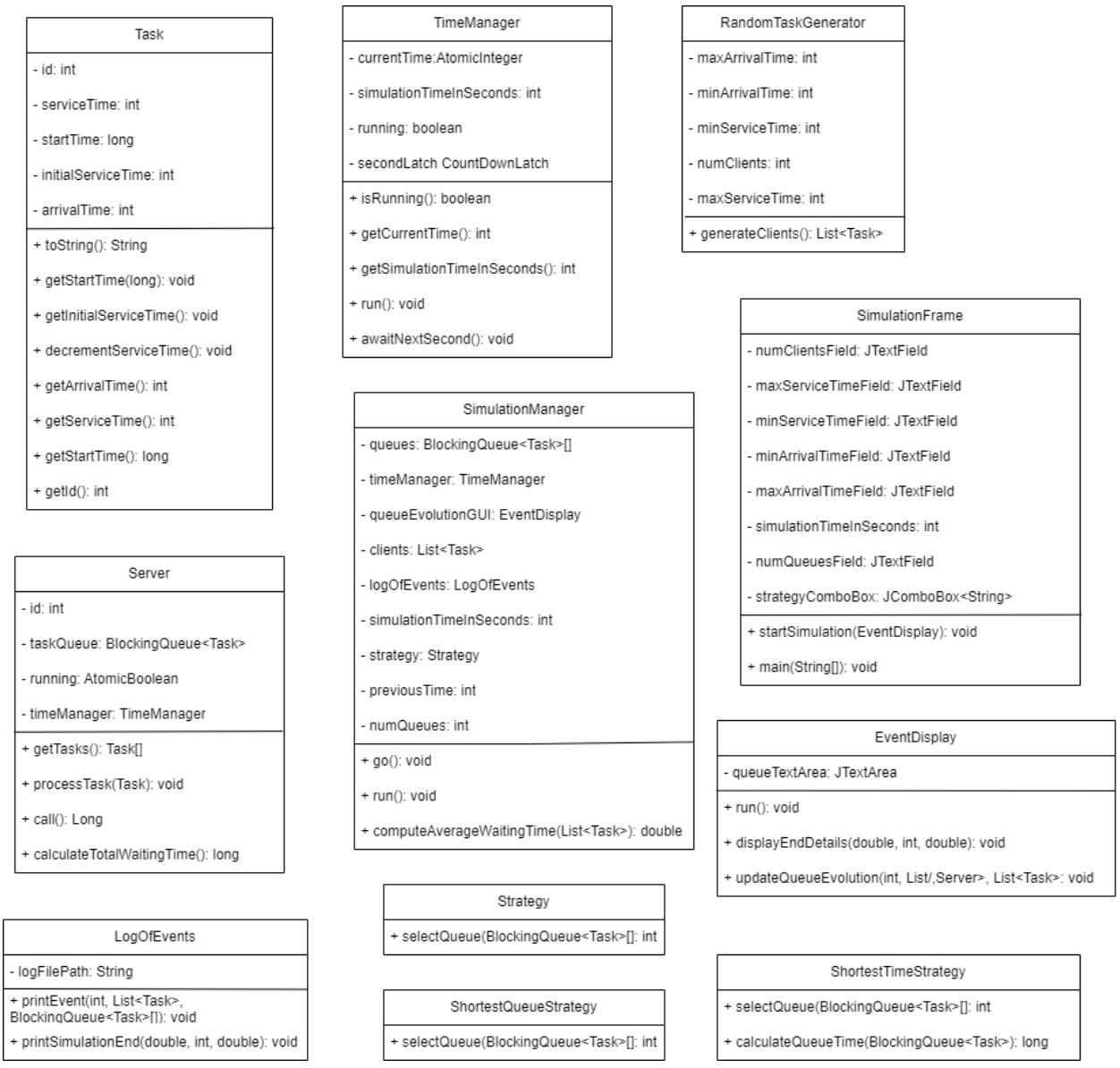
- The **logic** package depends on the **models** package to access and manipulate the data model objects (e.g., Task, Server) during the simulation process, performing operations such as task assignment, server processing, and queue management.

3.3. Division into classes



- **gui Package:**
  - SimulationFrame: Interacts with classes from the logic package to start and manage simulations. It also utilizes classes from the models package to display simulation parameters and results.
  - EventDisplay: Represents the graphical display of simulation events. It interacts with classes from the models package to update the display based on simulation data.
- **logic Package:**
  - SimulationManager: Orchestrates the simulation process and interacts with classes from the models package to handle tasks and queues.
  - ShortestQueueStrategy and ShortestTimeStrategy: Implement different strategies for queue selection and work independently of the GUI. They interact with classes from the models package to access task and queue information.
  - RandomTaskGenerator: Generates random tasks for the simulation. It interacts with classes from the models package to create task objects.
  - LogOfEvents: Manages logging events during the simulation. It interacts with classes from the models package to access task and queue information.
  - Strategy: Defines the interface for different queue selection strategies.
- **models Package:**
  - Task: Represents individual tasks in the simulation. It is used by the SimulationManager to create and manage tasks during the simulation.
  - Server: Represents server instances in the simulation. It interacts with classes from the logic package to process tasks from queues.
  - TimeManager: Manages the simulation time and is utilized by the SimulationManager to synchronize simulation events.

3.4. Division into routines



4. Implementation

4.1. Class Description

- **Task:**

This class represents a task within a simulation, encapsulating details such as its unique identifier, arrival time, service time, and start time. It is an essential component used by the simulation manager to model tasks and track their progress throughout the simulation. This class is integral to modeling tasks within the simulation, allowing for the tracking and processing of individual units of work.

The methods used are:

- getId(): Returns the unique identifier of the task.
- getArrivalTime(): Returns the arrival time of the task.
- getServiceTime(): Returns the remaining service time of the task.
- getInitialServiceTime(): Returns the initial service time of the task.
- toString(): Provides a string representation of the task, including its ID, arrival time, and remaining service time.
- decrementServiceTime(): Decrements the remaining service time of the task by one.
- getStartTime(): Returns the start time of the task.
- setStartTime(long time): Sets the start time of the task to the specified value.

- **Server:**

This class represents a server within the simulation, responsible for processing tasks from a queue. It encapsulates details such as the server's ID, the task queue it operates on, and the current time manager. Servers play a crucial role in the simulation by simulating the processing of tasks and tracking waiting times. This class is essential for simulating task processing within the simulation, providing functionality to process tasks, manage waiting times, and interact with the task queue.

The methods used are:

- call(): Implements the Callable interface method, representing the server's main processing logic. It continuously retrieves tasks from the queue and processes them until interrupted or instructed to stop.
- processTask(Task task): Simulates the processing of a task by the server, updating its start time and decrementing the task's service time until completion.
- getTasks(): Returns an array of tasks currently held in the server's task queue.
- calculateTotalWaitingTime(): Calculates the total waiting time of tasks in the server's queue.

```
@Override
public Long call() {
    while (running.get() && !Thread.currentThread().isInterrupted()) {
        Task task;
        try {
            timeManager.awaitNextSecond();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            break;
        }

        synchronized (taskQueue) {
            task = taskQueue.peek();
        }
        if (task != null) {
            processTask(task);
            synchronized (taskQueue) {
                taskQueue.remove();
            }
        }
        if (!running.get()) {
            break;
        }
    }
    return calculateTotalWaitingTime();
}
```

(the call() method from Server)

- **TimeManager:**

This class manages simulation time, controlling the progression of time within the simulation. It tracks the current simulation time, handles time increments, and provides synchronization mechanisms for time-related operations. This class serves as a fundamental component for managing time within the simulation, ensuring accurate time progression and synchronization of simulation events.

The methods used are:

- getCurrentTime(): Returns the current simulation time.
- isRunning(): Checks if the simulation is currently running.
- run(): Implements the Runnable interface method, representing the main logic for time progression in the simulation.
- getSimulationTimeInSeconds(): Returns the total duration of the simulation.
- awaitNextSecond(): Blocks until the next simulation second, using the secondLatch for synchronization.

```
@Override
public void run() {
    while (currentTime.get() < simulationTimeInSeconds && running) {
        try {
            TimeUnit.SECONDS.sleep( timeout: 2);
            currentTime.incrementAndGet();
            secondLatch.countDown();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
    running = false;
}
```

(the run() method of TimeManager)

- **SimulationFrame:**

This class represents the main graphical user interface (GUI) frame for configuring and starting the simulation. It provides input fields for setting simulation parameters such as the number of clients, queues, simulation time, arrival time range, service time range, and selection policy. Users interact with this frame to initialize and start the simulation. This class serves as the user interface for interacting with the simulation, allowing users to input simulation parameters and initiate the simulation process.

The methods used are:

- startSimulation(EventDisplay queueEvolutionGUI): Starts the simulation with the parameters specified by the user.
- main(String[] args): The entry point of the application, creating an instance of SimulationFrame and EventDisplay.

```
int simulationTimeInSeconds = Integer.parseInt(simulationTimeField.getText());
int numQueues = Integer.parseInt(numQueuesField.getText());
int numClients = Integer.parseInt(numClientsField.getText());
int minArrivalTime = Integer.parseInt(minArrivalTimeField.getText());
int maxArrivalTime = Integer.parseInt(maxArrivalTimeField.getText());
int minServiceTime = Integer.parseInt(maxServiceTimeField.getText());
int maxServiceTime = Integer.parseInt(minServiceTimeField.getText());
String selectedStrategy = (String) strategyComboBox.getSelectedItem();
Strategy strategy;
assert selectedStrategy != null;
if (selectedStrategy.equals("Shortest Queue Strategy")) {
    strategy = new ShortestQueueStrategy();
} else {
    strategy = new ShortestTimeStrategy();
}

RandomTaskGenerator clientGenerator = new RandomTaskGenerator(
        numClients, minArrivalTime, maxArrivalTime, minServiceTime, maxServiceTime);
List<Task> clients = clientGenerator.generateClients();
clients.sort(Comparator.comparingInt(Task::getArrivalTime));

BlockingQueue<Task>[] queues = new LinkedBlockingQueue[numQueues];
for (int i = 0; i < numQueues; i++) {
    queues[i] = new LinkedBlockingQueue<>();
}

SimulationManager simulationManager = new SimulationManager(numQueues, simulationTimeInSeconds, queues, clien
Thread simulationManagerThread = new Thread(simulationManager);
simulationManagerThread.start();
```

(the main logic from the SimulationFrame class)

- **EventDisplay:**

This class represents a graphical display for visualizing the evolution of queues during the simulation. It provides a visual representation of how queues change over time as tasks are processed, allowing users to observe queue dynamics and performance metrics during the simulation. This class enhances the simulation experience by providing a visual representation of queue dynamics, enabling users to analyze and interpret simulation results more effectively.
The methods used are:

- updateQueueEvolution(int currentTime, List<Server> servers, List<Task> clients): Updates the display with the current state of queues and tasks at the specified simulation time.
- displayEndDetails(double averageWaitingTime, int peakSecond, double averageServiceTime): Displays end-of-simulation details such as average waiting time, peak hour, and average service time.

```
public EventDisplay() {
    setTitle("Events");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout());

    JLabel titleLabel = new JLabel( text: "Real Time Display of Events");
    titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
    titleLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 24)); // Updated font size to 24
    add(titleLabel, BorderLayout.NORTH);

    queueTextArea = new JTextArea();
    queueTextArea.setEditable(false);
    queueTextArea.setFont(new Font( name: "Arial", Font.PLAIN,  size: 16)); // Updated font size to 16
    JScrollPane scrollPane = new JScrollPane(queueTextArea);
    add(scrollPane, BorderLayout.CENTER);

    setSize( width: 500,  height: 400); // Adjusted size to accommodate larger text
    setLocationRelativeTo(null);
}
```

(the DisplayEvent constructor)

- **LogOfEvents:**

This class handles logging events and simulation statistics to a file during the simulation. It provides methods for printing events such as the current state of waiting clients and queues at each time step, as well as printing simulation summary statistics at the end of the simulation. This class facilitates logging simulation events and statistics, providing a record of simulation behavior and performance for later analysis.

The methods used are:

- printEvent(int currentTime, List<Task> waitingClients, BlockingQueue<Task>[] queues): Prints the current state of waiting clients and queues at the specified simulation time.
- printSimulationEnd(double averageWaitingTime, int peakHour, double averageServiceTime): Prints summary statistics at the end of the simulation, including average waiting time, average service time, and peak hour.

```
public void printEvent(int currentTime, List<Task> waitingClients, BlockingQueue<Task>[] queues) {
    try (FileWriter writer = new FileWriter(logFilePath, append: true)) {
        writer.write( str: "Time " + currentTime + "\n");

        writer.write( str: "Waiting clients: ");
        for (Task task : waitingClients) {
            if (task.getArrivalTime() > currentTime && task.getServiceTime() != 0) { // Include only clients with
                writer.write( str: "(" + task.getId() + "," + task.getArrivalTime() + "," + task.getServiceTime() +
            }
        }
        writer.write( str: "\n");

        for (int i = 0; i < queues.length; i++) {
            BlockingQueue<Task> queue = queues[i];
            writer.write( str: "Queue " + (i + 1) + ": ");
            if (queue.isEmpty()) {
                writer.write( str: "closed\n");
            } else {
                synchronized (queue) {
                    for (Task task : queue) {
                        writer.write( str: "(" + task.getId() + "," + task.getArrivalTime() + "," + task.getService
                    }
                }
                writer.write( str: "\n");
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

(the logic behind the printEvent() method)

- **RandomTaskGenerator:**

This class generates random tasks (clients) for the simulation. It allows specifying the number of clients, arrival time range, and service time range. Random tasks are generated based on these parameters. This class facilitates the creation of random tasks, allowing for variability in simulation inputs and scenarios.

The methods used are:

- generateClients(): Generates a list of random tasks (clients) based on the specified parameters.

```java
public List<Task> generateClients() {
    List<Task> clients = new ArrayList<>();
    Random random = new Random();

    for (int id = 1; id <= numClients; id++) {
        int arrivalTime = minArrivalTime + random.nextInt( bound: maxArrivalTime - minArrivalTime + 1);
        int serviceTime = minServiceTime + random.nextInt( bound: maxServiceTime - minServiceTime + 1);

        Task client = new Task(id, arrivalTime, serviceTime);
        clients.add(client);
    }

    return clients;
}
```

(the client generating logic)

- **SimulationManager:**

This class manages the simulation process. It coordinates the interaction between clients, queues, servers, and the time manager. It orchestrates the flow of tasks through the system according to the selected strategy and logs events during the simulation. This class serves as the core component of the simulation system, orchestrating the flow of tasks and managing simulation events and statistics.

The methods used are:

- run(): Implements the Runnable interface, defining the main logic of the simulation.
- go(): Manages the simulation process, including task distribution, queue processing, and logging.
- computeAverageWaitingTime(List<Task> completedTasks): Computes the average waiting time of completed tasks.

```java
while (timeManager.isRunning()) {
    int currentTime = timeManager.getCurrentTime();

    if (currentTime >= timeManager.getSimulationTimeInSeconds()) {
        break;
    }
    if (currentTime == previousTime) {
        continue;
    }
    previousTime = currentTime;

    for (Task client : clients) {
        if (client.getArrivalTime() == currentTime) {
            int selectedQueue = strategy.selectQueue(queues);
            try {
                queues[selectedQueue].put(client);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    Server[] servers = new Server[numQueues];
    for (int i = 0; i < numQueues; i++) {
        servers[i] = new Server( id: i + 1, queues[i], running, timeManager);
    }
```

(the main logic of the go() method)

- **Strategy:**

This interface defines the contract for different queue selection strategies used in the simulation. It requires implementing classes to provide a method for selecting the appropriate queue based on the current state of the queues.
The methods used are:

- int selectQueue(BlockingQueue<Task>[] queues): Selects the queue according to the specific strategy implemented.

- **ShortestQueueStrategy:** this class implements the shortest queue strategy, selecting the queue with the fewest tasks in it, using the method: selectQueue(BlockingQueue<Task>[] queues) that selects the queue with the shortest length from the array of queues.

- **ShortestTimeStrategy:** this class implements the shortest time strategy, selecting the queue with the shortest total service time, using the methods: selectQueue(BlockingQueue<Task>[] queues) that elects the queue with the shortest total service time from the array of queues and calculateQueueTime(BlockingQueue<Task> queue) that calculates the total service time of tasks in a queue.

5. Conclusions

In conclusion, the simulation project has provided valuable insights into the dynamics of queueing systems and the challenges of managing server resources efficiently. Through the simulation, we were able to observe how different strategies for queue management and server allocation can impact system performance, including average waiting times, peak queue sizes, and overall service efficiency.

Throughout the assignment, I gained a deeper understanding of various concepts related to queueing theory, concurrency, and simulation modeling. I learned how to design and implement a simulation framework using Java, including the use of multithreading for concurrent processing of tasks and the visualization of system dynamics. Additionally, I enhanced my problem-solving skills by addressing challenges such as client-server interactions, event-driven simulation, and performance optimization.

Looking ahead, there are several areas where the simulation project could be extended or improved. One possibility is to explore more advanced queueing models, such as priority queues or queues with dynamic server allocation policies. Additionally, expanding the scope of the simulation to encompass broader scenarios, such as network traffic management or supply chain logistics, could offer valuable insights into diverse real-world applications of queueing theory and simulation modeling.

6. Bibliography

The following references were consulted during the implementation of the project:
1. https://dsrl.eu/courses/pt/materials/PT_2024_A2_S1.pdf
2. https://dsrl.eu/courses/pt/materials/PT_2024_A2_S2.pdf
3. https://www.geeksforgeeks.org/java-threads/
4. https://www.freecodecamp.org/news/what-are-threads-in-java-how-to-create-one/
5. https://www.javatpoint.com/synchronization-in-java
6. https://www.geeksforgeeks.org/synchronization-in-java/

7. Results

These are the results from running the provided examples in the Queue Simulation System:

- **Example 1**

Time 0
Waiting clients: (4,3,2); (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 1
Waiting clients: (4,3,2); (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 2
Waiting clients: (4,3,2); (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 3
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: (4,3,2);
Queue 2: closed
Time 4
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: (4,3,1);
Queue 2: closed
Time 5
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 6
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed

Time 7
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 8
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 9
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 10
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 11
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 12
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 13
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 14
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 15
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 16
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 17
Waiting clients: (3,18,4); (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed

Time 18
Waiting clients: (1,28,3); (2,30,4);
Queue 1: (3,18,4);
Queue 2: closed
Time 19
Waiting clients: (1,28,3); (2,30,4);
Queue 1: (3,18,3);
Queue 2: closed
Time 20
Waiting clients: (1,28,3); (2,30,4);
Queue 1: (3,18,2);
Queue 2: closed
Time 21
Waiting clients: (1,28,3); (2,30,4);
Queue 1: (3,18,1);
Queue 2: closed
Time 22
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 23
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 24
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 25
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 26
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 27
Waiting clients: (1,28,3); (2,30,4);
Queue 1: closed
Queue 2: closed
Time 28
Waiting clients: (2,30,4);
Queue 1: (1,28,3);
Queue 2: closed

Time 29
Waiting clients: (2,30,4);
Queue 1: (1,28,2);
Queue 2: closed
Time 30
Waiting clients:
Queue 1: (1,28,1);
Queue 2: (2,30,4);
Time 31
Waiting clients:
Queue 1: closed
Queue 2: (2,30,3);
Time 32
Waiting clients:
Queue 1: closed
Queue 2: (2,30,2);
Time 33
Waiting clients:
Queue 1: closed
Queue 2: (2,30,1);
Time 34
Waiting clients:
Queue 1: closed
Queue 2: closed
Average waiting time: 0.0
Average service time: 3.25
Peak hour: 30
Simulation ended.

- **Example 2**

Time 0
Waiting clients: (10,2,3); (22,2,4); (32,2,1); (33,3,5); (11,4,5); (38,4,5); (43,4,5); (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed
Time 1

Waiting clients: (10,2,3); (22,2,4); (32,2,1); (33,3,5); (11,4,5); (38,4,5); (43,4,5); (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed
Time 2
Waiting clients: (33,3,5); (11,4,5); (38,4,5); (43,4,5); (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (10,2,3);
Queue 2: (22,2,4);
Queue 3: (32,2,1);
Queue 4: closed
Queue 5: closed
Time 3
Waiting clients: (11,4,5); (38,4,5); (43,4,5); (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (10,2,2);
Queue 2: (22,2,3);
Queue 3: closed
Queue 4: (33,3,5);
Queue 5: closed
Time 4
Waiting clients: (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (10,2,1); (43,4,5);
Queue 2: (22,2,2);
Queue 3: (11,4,5);
Queue 4: (33,3,4);
Queue 5: (38,4,5);

Time 5
Waiting clients: (8,6,3); (39,6,3); (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (43,4,5);
Queue 2: (22,2,1);
Queue 3: (11,4,4);
Queue 4: (33,3,3);
Queue 5: (38,4,4);
Time 6
Waiting clients: (15,7,1); (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (43,4,4); (8,6,3);
Queue 2: (39,6,3);
Queue 3: (11,4,3);
Queue 4: (33,3,2);
Queue 5: (38,4,3);
Time 7
Waiting clients: (21,8,2); (27,8,4); (42,8,4); (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (43,4,3); (8,6,3);
Queue 2: (39,6,2); (15,7,1);
Queue 3: (11,4,2);
Queue 4: (33,3,1);
Queue 5: (38,4,2);
Time 8
Waiting clients: (1,9,6); (35,9,2); (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (43,4,2); (8,6,3);
Queue 2: (39,6,1); (15,7,1);
Queue 3: (11,4,1); (21,8,2);
Queue 4: (27,8,4);
Queue 5: (38,4,1); (42,8,4);
Time 9

Waiting clients: (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (43,4,1); (8,6,3); (35,9,2);
Queue 2: (15,7,1);
Queue 3: (21,8,2);
Queue 4: (27,8,3); (1,9,6);
Queue 5: (42,8,4);
Time 10
Waiting clients: (16,11,3); (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (8,6,3); (35,9,2);
Queue 2: closed
Queue 3: (21,8,1);
Queue 4: (27,8,2); (1,9,6);
Queue 5: (42,8,3);
Time 11
Waiting clients: (12,12,4); (34,12,6); (50,12,4); (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (8,6,2); (35,9,2);
Queue 2: (16,11,3);
Queue 3: closed
Queue 4: (27,8,1); (1,9,6);
Queue 5: (42,8,2);
Time 12
Waiting clients: (6,13,1); (40,13,6); (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (8,6,1); (35,9,2);
Queue 2: (16,11,2); (34,12,6);
Queue 3: (12,12,4); (50,12,4);
Queue 4: (1,9,6);
Queue 5: (42,8,1);
Time 13
Waiting clients: (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);

Queue 1: (35,9,2);
Queue 2: (16,11,1); (34,12,6);
Queue 3: (12,12,3); (50,12,4);
Queue 4: (1,9,5); (6,13,1);
Queue 5: (40,13,6);
Time 14
Waiting clients: (2,15,6); (23,15,7); (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (35,9,1);
Queue 2: (34,12,6);
Queue 3: (12,12,2); (50,12,4);
Queue 4: (1,9,4); (6,13,1);
Queue 5: (40,13,5);
Time 15
Waiting clients: (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,6);
Queue 2: (34,12,5); (23,15,7);
Queue 3: (12,12,1); (50,12,4);
Queue 4: (1,9,3); (6,13,1);
Queue 5: (40,13,4);
Time 16
Waiting clients: (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,5);
Queue 2: (34,12,4); (23,15,7);
Queue 3: (50,12,4);
Queue 4: (1,9,2); (6,13,1);
Queue 5: (40,13,3);
Time 17
Waiting clients: (29,18,1); (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,4);
Queue 2: (34,12,3); (23,15,7);
Queue 3: (50,12,3);
Queue 4: (1,9,1); (6,13,1);
Queue 5: (40,13,2);
Time 18

Waiting clients: (36,19,7); (37,19,5); (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,3); (29,18,1);
Queue 2: (34,12,2); (23,15,7);
Queue 3: (50,12,2);
Queue 4: (6,13,1);
Queue 5: (40,13,1);
Time 19
Waiting clients: (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,2); (29,18,1);
Queue 2: (34,12,1); (23,15,7);
Queue 3: (50,12,1); (36,19,7);
Queue 4: (37,19,5);
Queue 5: closed
Time 20
Waiting clients: (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (2,15,1); (29,18,1);
Queue 2: (23,15,7);
Queue 3: (36,19,7);
Queue 4: (37,19,4);
Queue 5: closed
Time 21
Waiting clients: (46,22,7); (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (29,18,1);
Queue 2: (23,15,6);
Queue 3: (36,19,6);
Queue 4: (37,19,3);
Queue 5: closed
Time 22
Waiting clients: (3,23,5); (9,23,4); (24,23,6); (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: closed
Queue 2: (23,15,5);
Queue 3: (36,19,5);
Queue 4: (37,19,2);
Queue 5: (46,22,7);

Time 23
Waiting clients: (31,24,2); (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (3,23,5); (9,23,4);
Queue 2: (23,15,4); (24,23,6);
Queue 3: (36,19,4);
Queue 4: (37,19,1);
Queue 5: (46,22,6);
Time 24
Waiting clients: (30,25,1); (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (3,23,4); (9,23,4);
Queue 2: (23,15,3); (24,23,6);
Queue 3: (36,19,3); (31,24,2);
Queue 4: closed
Queue 5: (46,22,5);
Time 25
Waiting clients: (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (3,23,3); (9,23,4);
Queue 2: (23,15,2); (24,23,6);
Queue 3: (36,19,2); (31,24,2);
Queue 4: (30,25,1);
Queue 5: (46,22,4);
Time 26
Waiting clients: (20,27,1); (26,27,6); (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (3,23,2); (9,23,4);
Queue 2: (23,15,1); (24,23,6);
Queue 3: (36,19,1); (31,24,2);
Queue 4: closed
Queue 5: (46,22,3);
Time 27
Waiting clients: (4,28,1); (5,28,5); (25,28,5); (47,28,3); (49,28,1); (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (3,23,1); (9,23,4);
Queue 2: (24,23,6);
Queue 3: (31,24,2);
Queue 4: (20,27,1); (26,27,6);
Queue 5: (46,22,2);
Time 28

Waiting clients: (13,29,2); (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (9,23,4); (47,28,3);
Queue 2: (24,23,5); (4,28,1); (49,28,1);
Queue 3: (31,24,1); (5,28,5);
Queue 4: (26,27,6);
Queue 5: (46,22,1); (25,28,5);
Time 29
Waiting clients: (17,30,2); (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (9,23,3); (47,28,3);
Queue 2: (24,23,4); (4,28,1); (49,28,1);
Queue 3: (5,28,5);
Queue 4: (26,27,5); (13,29,2);
Queue 5: (25,28,5);
Time 30
Waiting clients: (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (9,23,2); (47,28,3);
Queue 2: (24,23,3); (4,28,1); (49,28,1);
Queue 3: (5,28,4); (17,30,2);
Queue 4: (26,27,4); (13,29,2);
Queue 5: (25,28,4);
Time 31
Waiting clients: (18,32,5); (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (9,23,1); (47,28,3);
Queue 2: (24,23,2); (4,28,1); (49,28,1);
Queue 3: (5,28,3); (17,30,2);
Queue 4: (26,27,3); (13,29,2);
Queue 5: (25,28,3);
Time 32
Waiting clients: (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (47,28,3);
Queue 2: (24,23,1); (4,28,1); (49,28,1);
Queue 3: (5,28,2); (17,30,2);
Queue 4: (26,27,2); (13,29,2);
Queue 5: (25,28,2); (18,32,5);
Time 33
Waiting clients: (45,34,6); (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (47,28,2);
Queue 2: (4,28,1); (49,28,1);
Queue 3: (5,28,1); (17,30,2);
Queue 4: (26,27,1); (13,29,2);

Queue 5: (25,28,1); (18,32,5);
Time 34
Waiting clients: (28,35,5); (41,35,7); (44,35,7); (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (47,28,1); (45,34,6);
Queue 2: (49,28,1);
Queue 3: (17,30,2);
Queue 4: (13,29,2);
Queue 5: (18,32,5);
Time 35
Waiting clients: (14,36,3); (48,38,2); (7,40,6); (19,40,4);
Queue 1: (45,34,6);
Queue 2: (28,35,5);
Queue 3: (17,30,1); (41,35,7);
Queue 4: (13,29,1); (44,35,7);
Queue 5: (18,32,4);
Time 36
Waiting clients: (48,38,2); (7,40,6); (19,40,4);
Queue 1: (45,34,5); (14,36,3);
Queue 2: (28,35,4);
Queue 3: (41,35,7);
Queue 4: (44,35,7);
Queue 5: (18,32,3);
Time 37
Waiting clients: (48,38,2); (7,40,6); (19,40,4);
Queue 1: (45,34,4); (14,36,3);
Queue 2: (28,35,3);
Queue 3: (41,35,6);
Queue 4: (44,35,6);
Queue 5: (18,32,2);
Time 38
Waiting clients: (7,40,6); (19,40,4);
Queue 1: (45,34,3); (14,36,3);
Queue 2: (28,35,2); (48,38,2);
Queue 3: (41,35,5);
Queue 4: (44,35,5);
Queue 5: (18,32,1);
Time 39
Waiting clients: (7,40,6); (19,40,4);
Queue 1: (45,34,2); (14,36,3);
Queue 2: (28,35,1); (48,38,2);
Queue 3: (41,35,4);
Queue 4: (44,35,4);
Queue 5: closed
Time 40

Waiting clients:
Queue 1: (45,34,1); (14,36,3);
Queue 2: (48,38,2);
Queue 3: (41,35,3); (19,40,4);
Queue 4: (44,35,3);
Queue 5: (7,40,6);
Time 41
Waiting clients:
Queue 1: (14,36,3);
Queue 2: (48,38,1);
Queue 3: (41,35,2); (19,40,4);
Queue 4: (44,35,2);
Queue 5: (7,40,5);
Time 42
Waiting clients:
Queue 1: (14,36,2);
Queue 2: closed
Queue 3: (41,35,1); (19,40,4);
Queue 4: (44,35,1);
Queue 5: (7,40,4);
Time 43
Waiting clients:
Queue 1: (14,36,1);
Queue 2: closed
Queue 3: (19,40,4);
Queue 4: closed
Queue 5: (7,40,3);
Time 44
Waiting clients:
Queue 1: closed
Queue 2: closed
Queue 3: (19,40,3);
Queue 4: closed
Queue 5: (7,40,2);
Time 45
Waiting clients:
Queue 1: closed
Queue 2: closed
Queue 3: (19,40,2);
Queue 4: closed
Queue 5: (7,40,1);
Time 46
Waiting clients:
Queue 1: closed

Queue 2: closed
Queue 3: (19,40,1);
Queue 4: closed
Queue 5: closed
Time 47
Waiting clients:
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed
Average waiting time: 1.8
Average service time: 3.98
Peak hour: 28
Simulation ended.

- **Example 3**

(attached to the project)