



NUST-CEME

## **Hospital Emergency Department System Simulation**

**OBJECT ORIENTED  
PROGRAMMING**

# **Project Report**

### **Authors:**

**Muhammad Hisham Bin Nauman**

**Rohan Arshad**

**Talha Anwar Saeed Qureshi**

## Table Of Contents

Introduction.....	3
Problem Statement.....	4
Objectives: .....	5
System Design: .....	6
UML Diagram: .....	6
System Level Diagram: .....	14
Implementation Details:.....	15
Key Features .....	17
Results and Outputs: .....	19
Challenges:.....	21
Conclusion: .....	22
References:.....	23

# Introduction

## Overview

In this project, a simulation model is developed to investigate potential impacts of changing the following aspects of ED:

1. number of beds
2. number and rate of patient arrivals
3. acuity of illness or injury of patients
4. hospital staffing arrangements
5. access to inpatient beds

A Hospital Emergency Department (ED) System manages patient care in an emergency setting. It involves triage, treatment, and resource allocation to address urgent medical needs effectively. Key components include patient registration, triage categorization, medical care, and admission/discharge.

## Importance of a Hospital ED System

1. **Patient Prioritization:** Ensures critically ill patients receive immediate care.
2. **Resource Management:** Allocates staff, beds, and equipment efficiently.
3. **Data Tracking:** Maintains records for informed decision-making and audits.
4. **Emergency Preparedness:** Handles surges during crises like pandemics or natural disasters.

## Role of Simulations in Healthcare Systems

Simulations are virtual models used to analyze, test, and optimize healthcare processes without disrupting real-world operations.

1. **Training:** Prepares staff for handling emergencies.
2. **Process Optimization:** Identifies bottlenecks in patient flow and resource usage.

3. **Scenario Testing:** Evaluates strategies for various situations, such as mass casualties.
4. **Cost-Effectiveness:** Tests system improvements before investing in real changes.

## Problem Statement

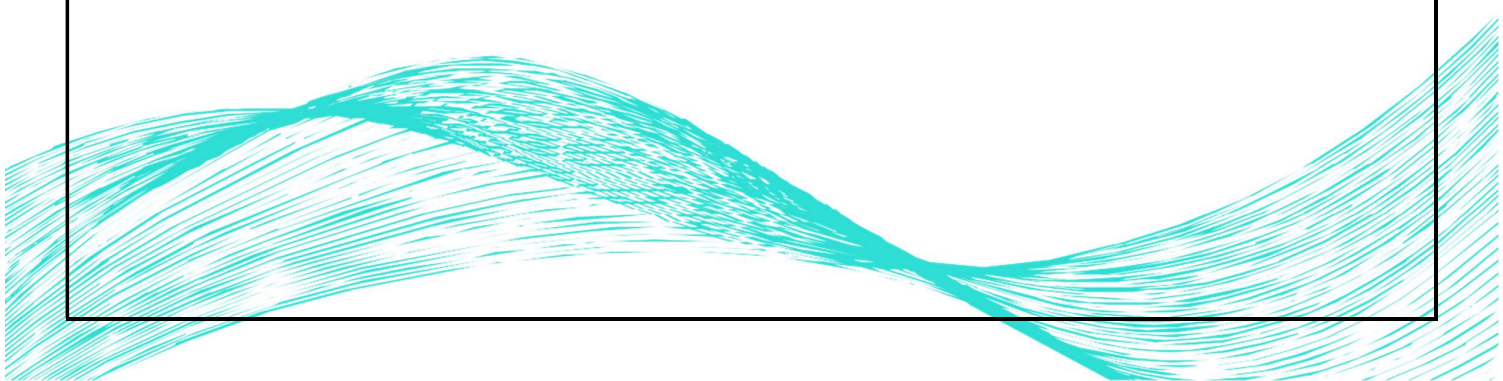
### Problem Definition

The project addresses the challenge of **simulating patient flow in a hospital Emergency Department (ED)**. This includes:

1. Assigning diseases and conditions to patients based on **triage categories**, which prioritize care according to urgency.
2. Managing limited resources such as staff, beds, and medical equipment to handle varying patient demands.
3. Optimizing decision-making in real-time to ensure efficient and equitable care delivery.

### Use of Probability Distribution Functions (PDFs) in Decision-Making

Probability distribution functions play a crucial role in the simulation by modeling uncertainties and variations in real-world ED scenarios:

1. **Arrival Times:** Simulating random patient arrivals using distributions like Poisson or exponential.
  2. **Triage Categories:** Assigning patients to triage levels based on weighted probabilities reflecting real-world statistics.
  3. **Treatment Durations:** Modeling the time required for different procedures using normal or gamma distributions.
  4. **Disease Assignment:** Randomly associating conditions with triage categories to reflect variability in patient presentations.
  5. **Resource Availability:** Simulating randomness in staff availability and equipment usage.
- 

# Objectives:

## Goals of the Project

### 1. Simulate Disease Assignment in an Emergency Department

Accurately assign diseases and conditions to patients based on their triage categories and severity levels.

### 2. Implement an OOP-Based Approach

Design a modular, scalable, and maintainable system using Object-Oriented Programming principles to represent real-world entities such as patients, diseases, and ED operations.

### 3. Incorporate Probability Distributions

Use various probability distribution functions (e.g. normal, exponential) to model:

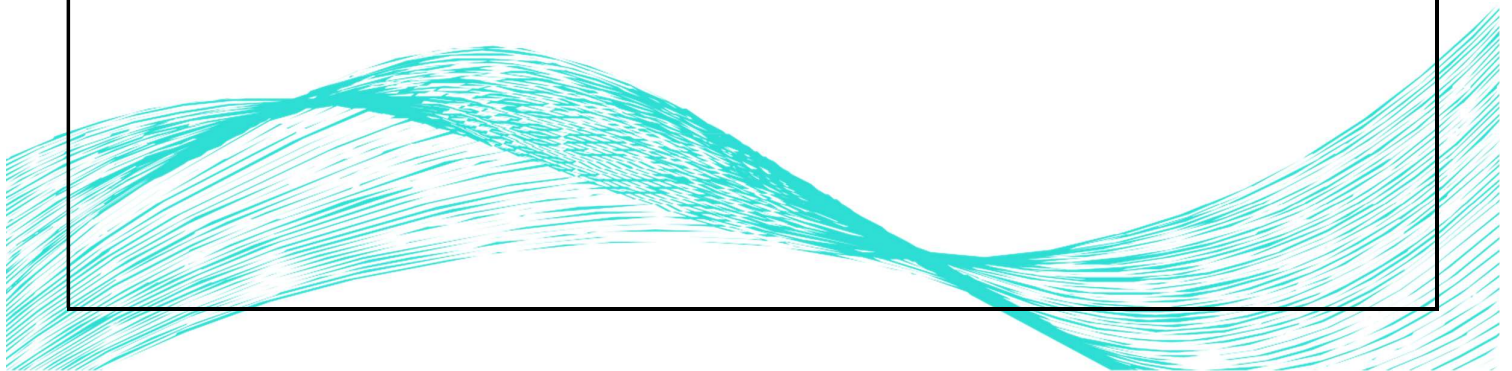
- i. Patient arrival times.
- ii. Disease severity assignments.
- iii. Treatment durations and resource allocation.

### 4. Optimize Emergency Department Operations

Simulate and analyze patient flow to identify bottlenecks and test strategies for efficient resource utilization.

### 5. Provide a Realistic and Flexible Simulation Environment

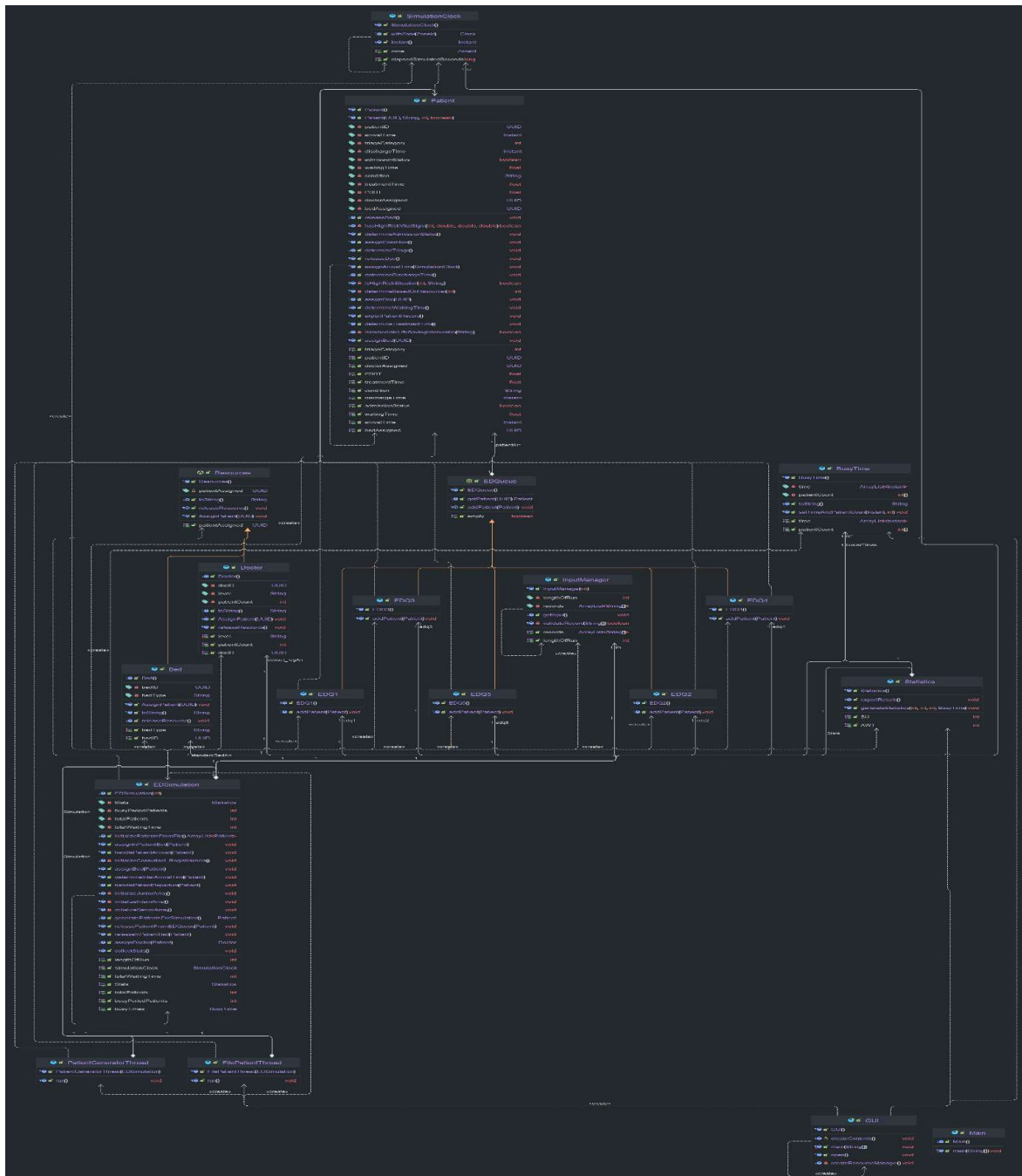
Allow customization of parameters (e.g., number of patients, triage levels, simulation duration) for different scenarios and emergency preparedness.





## System Design:

### UML Diagram:



## **The key classes in the system include:**

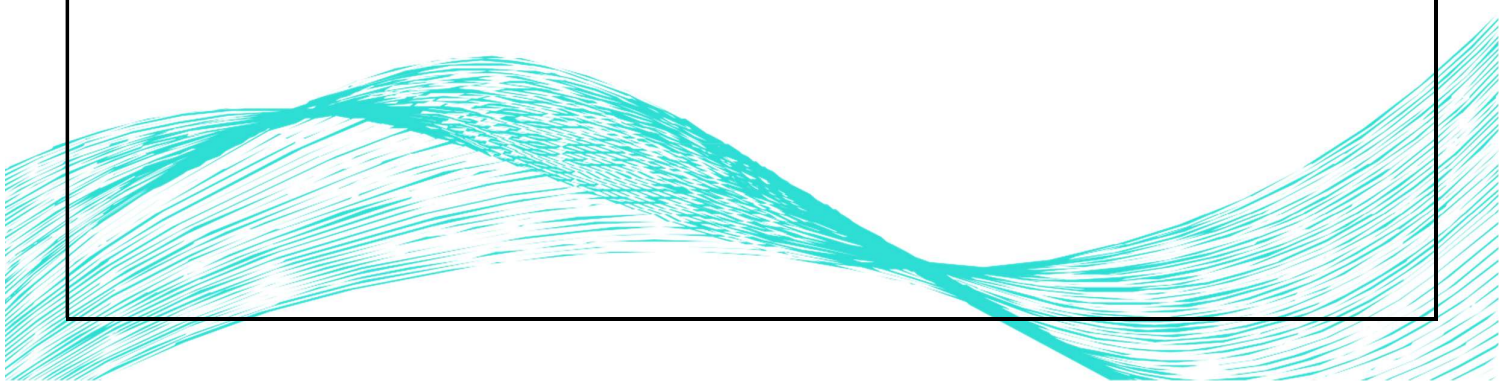
### **1. Patient**

#### **Attributes:**

- private UUID patientID;
- private String condition;
- private int triageCategory;
- private Instant arrivalTime;
- private float waitingTime;
- private float treatmentTime;
- private UUID bedAssigned;
- private UUID doctorAssigned;
- private float PDDT;
- private Instant dischargeTime;
- private boolean admissionStatus;

#### **Methods:**

- public Patient(UUID patientID, String condition, int triageCategory, boolean admissionStatus)
- public Patient()
- public final UUID getPatientID()
- public final String getCondition()
- public final int getTriageCategory()
- public final Instant getArrivalTime()
- public final float getWaitingTime()
- public final float getTreatmentTime()
- public final UUID getBedAssigned()
- public final UUID getDoctorAssigned()
- public final float getPDDT()
- public final Instant getDischargeTime()
- public final boolean getAdmissionStatus()
- public void determineTriage()
- public void determineWaitingTime()
- private boolean isImmediateLifeSavingIntervention(String condition)
- private boolean isHighRiskSituation(int painLevel, String condition)



- private boolean hasHighRiskVitalSigns(int HR, double RR, double SpO2, double BP)
- private int determineBasedOnResources(int resourcesNo)
- public void assignArrivalTime(SimulationClock sc)
- public void determineAdmissionStatus()
- public void determineTreatmentTime()
- public void determineDischargeTime()
- public void assignCondition()
- public void assignBed(UUID BedID)
- public void assignDoc(UUID DocID)
- public void releaseBed()
- public void releaseDoc()
- public void exportPatientRecord()

## **2. Resources (Abstract Class)**

### **Attributes:**

- patientAssigned: UUID

### **Methods:**

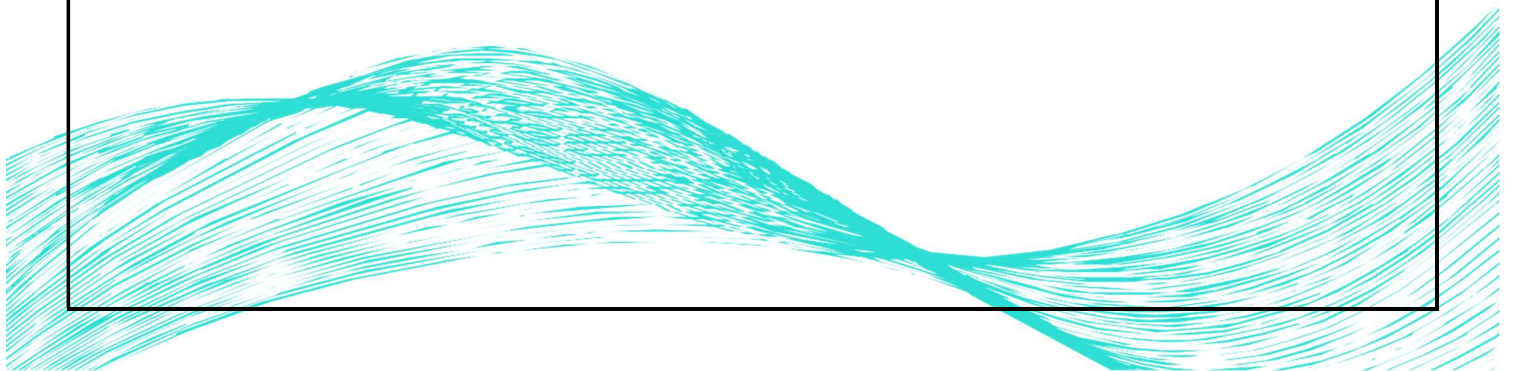
- public Resources()
- public final UUID getPatientAssigned()
- public abstract String toString()
- public abstract void releaseResource()
- public abstract void AssignPatient(UUID patientID)

## **3. Bed (inherits from Resources)**

The Bed class represents hospital beds. Each bed is uniquely identified by a UUID and categorized by type. The class manages assignment and release of resources.

### **Attributes:**

- bedID: UUID
- bedType: String





**Methods:**

- public Bed()
- public final UUID getBedID()
- public void setBedType(String bType)
- public String getBedType()
- public void releaseResource()
- public String toString()
- public void AssignPatient(UUID patientID)

**4. Doctor (inherits from Resources)**

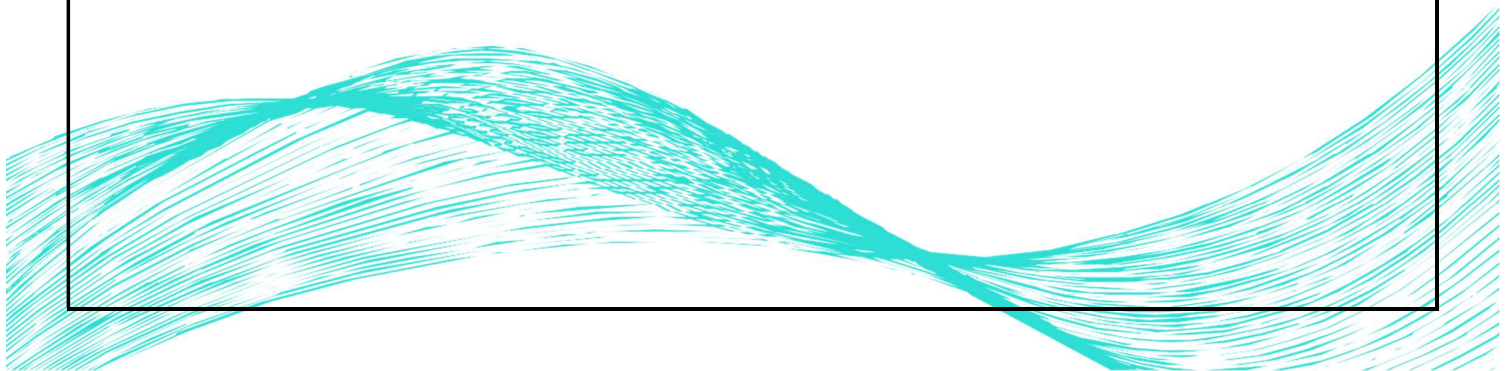
The Doctor class represents hospital doctors. Each doctor has a unique ID, a level, and the ability to treat multiple patients simultaneously based on their level.

**Attributes:**

- ext\_patientAssigned: ArrayList<UUID>
- patientCount: int
- docID: UUID
- level: String

**Methods:**

- public Doctor()
- public final UUID getDocID()
- public String getLevel()
- public final int getPatientCount()
- public void setLevel(String newLevel)
- public void releaseResource()
- public String toString()
- public void AssignPatient(UUID patientID)

**5. Abstract Class: EDQueue**

**Attributes:**

- protected ArrayList<Patient> patientArr;

**Methods:**

- public EDQueue()
- public abstract void addPatient(Patient patient)
- public Patient getPatient(UUID patientID)
- public boolean isEmpty()

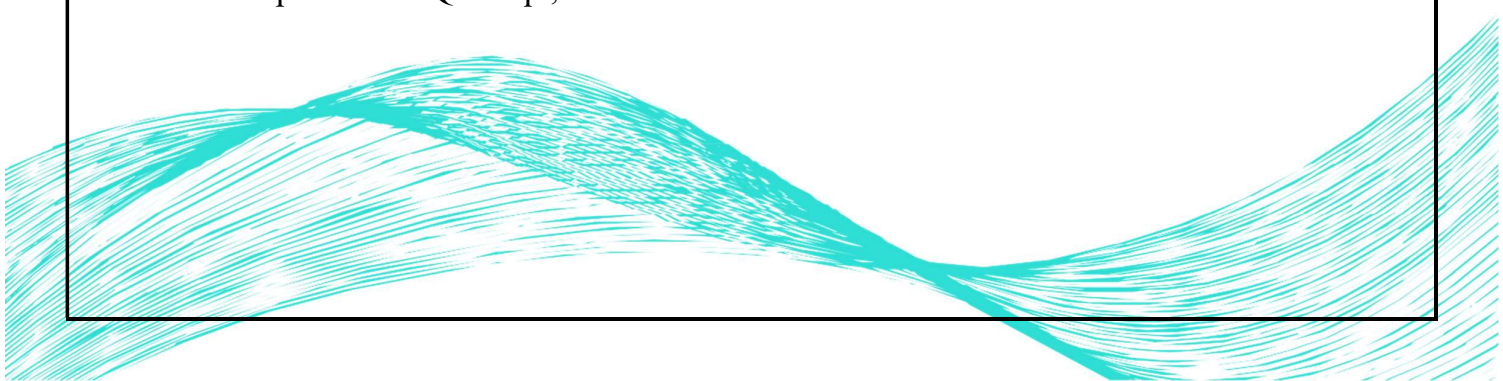
**Class: EDQ1, EDQ2, EDQ3, EDQ4, EDQ5 (Subclass of EDQueue)****Attributes:**

- patientArr (Inherited from EDQueue class)

**Methods:**

- Default Constructor (e.g public EDQ1())
- public void addPatient(Patient patient)

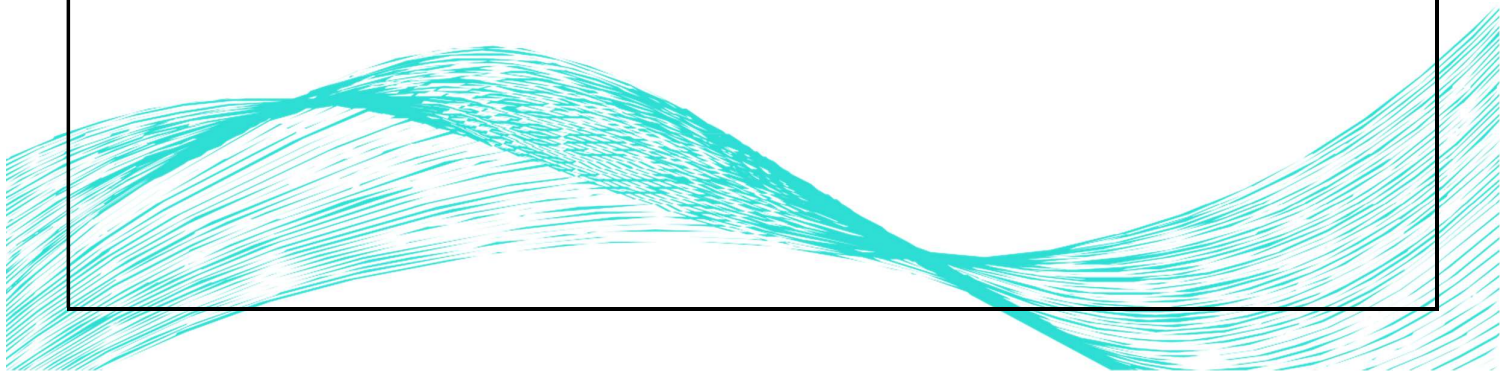
**6. EDSimulation****Attributes:**

- private double interArrivalTime;
  - private ArrayList<Doctor> consul\_regArr;
  - private ArrayList<Doctor> seniorArr;
  - private ArrayList<Doctor> juniorArr;
  - private ArrayList<Doctor> internArr;
  - private ArrayList<Bed> standardBedArr;
  - private ArrayList<Bed> corridorBedArr;
  - private ArrayList<Bed> reclinerChair;
  - private ArrayList<Bed> inPatientBeds;
  - private EDQ1 edq1;
  - private EDQ2 edq2;
  - private EDQ3 edq3;
- 

- private EDQ4 edq4;
- private EDQ5 edq5;
- private Statistics Stats;
- private InputManager im;
- private SimulationClock sc;
- private int totalPatients;
- private int busyPeriodPatients;
- private int totalWaitingTime;
- private BusyTime BT;

### **Methods:**

- public EDSimulation(int hours)
- public Statistics getStats()
- public int getTotalPatients()
- public int getBusyPeriodPatients()
- public int getTotalWaitingTime()
- public BusyTime getBusyTimes()
- public synchronized int getLengthOfRun()
- public synchronized SimulationClock getSimulationClock()
- public synchronized Patient generatePatientsForSimulation()
- public synchronized ArrayList<Patient> initializePatientsFromFile()
- public synchronized void handlePatientArrivals(Patient patient)
- public synchronized void releasePatientFromEDQueue(Patient patient)
- public synchronized void handlePatientDeparture(Patient patient)
- public synchronized void assignBed(Patient patient)
- public synchronized Doctor assignDoctor(Patient patient)
- public synchronized void assignInPatientBed(Patient patient)
- public synchronized void releaseInPatientBed(Patient patient)
- public synchronized void determineInterArrivalTime(Patient patient)
- private synchronized void initializeConsultant\_RegistrarArray()
- private synchronized void initializeSeniorArray()
- private synchronized void initializeJuniorArray()
- private synchronized void initializeInternArray()
- public synchronized void collectStats()



## 7. BusyTime

The BusyTime class tracks periods of high activity in the ED, recording timestamps and patient counts.

### Attributes:

- private ArrayList<Instant> time;
- private int[] patientCount;

### Methods:

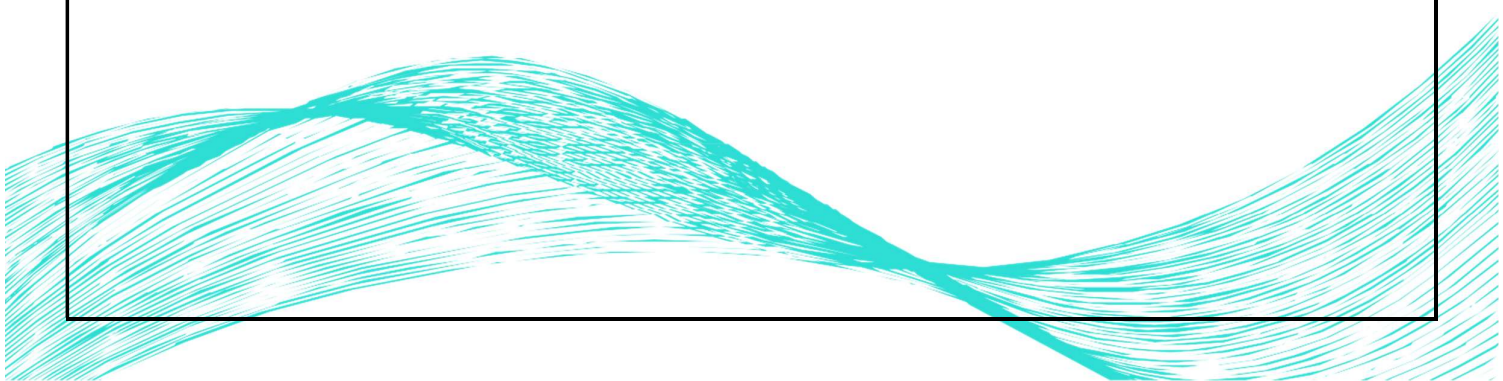
- public BusyTime()
- public ArrayList<Instant> getTime()
- public int[] getPatientCount()
- public void setTimeAndPatientCount(Instant t, int pc)
- public String toString()

## 8. Statistics

### Attributes:

- private int avgWaitingTime;
- private int bedUtilization;
- private BusyTime busierTimes;

### Methods:

- public Statistics()
  - public int getAWT()
  - public int getBU()
  - public void generateStatistics(int totalPatients, int totalWaitingTime, int busyPeriodPatients, BusyTime BT)
  - public void exportResults()
- 



## 9. InputManager

### Attributes:

- lengthOfRun: int
- records: ArrayList<String[]>

### Methods:

- InputManager(int hours)
- getRecords()
- getLengthOfRun()
- getInput()
- validateRecord(String[] data)

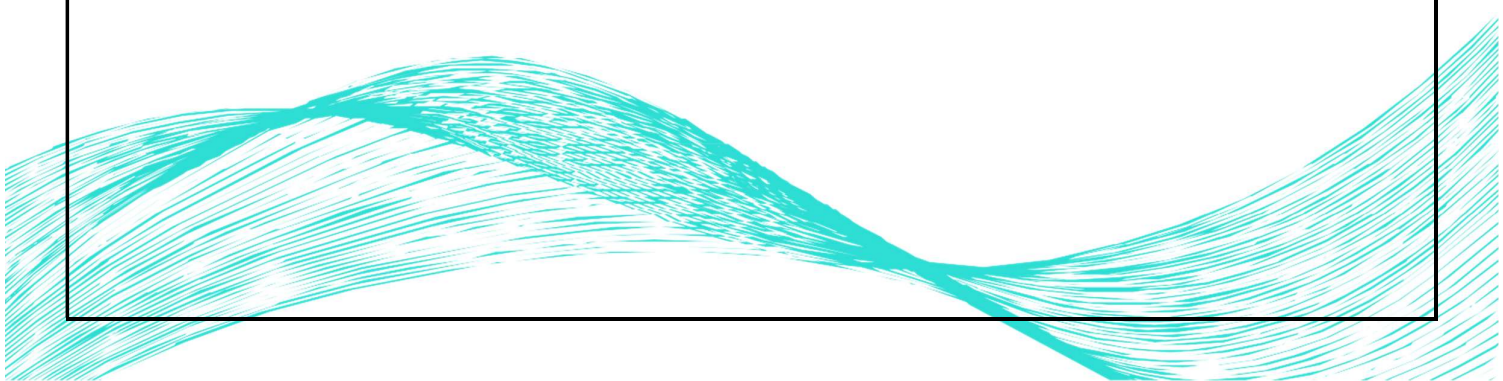
## 10. SimulationClock

### Attributes:

- real\_world\_Start\_Time\_ms: long
- simulationTime: Instant

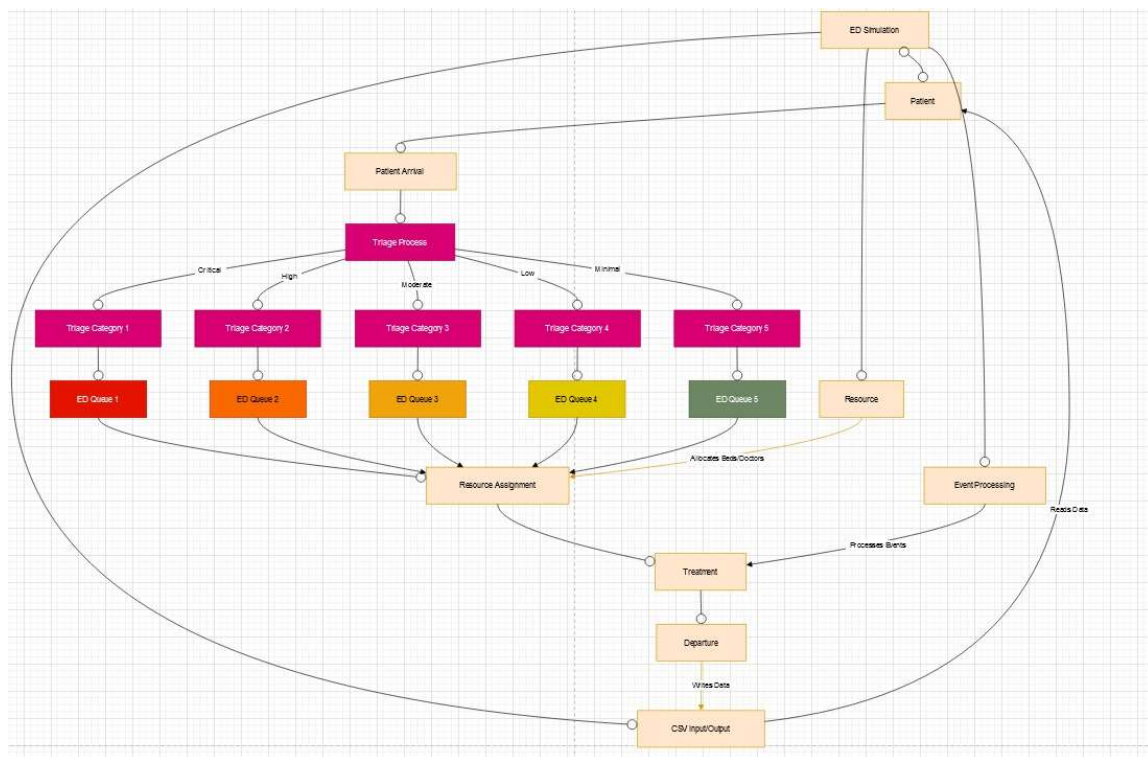
### Methods:

- public SimulationClock()
- public ZoneId getZone()
- public Clock withZone(ZoneId zone)
- public Instant instant()
- public long getElapsedSimulatedSeconds()



## System Level Diagram:

The workflow begins with patient arrival at the ED. Each patient is assigned a triage category and added to the corresponding queue. The system matches available doctors and beds to patients based on priority. Simulation results are logged, including waiting times, bed utilization, and discharge details



# Implementation Details:

## Patient Class:

The Patient class encapsulates the attributes and behaviors of individual patients. Each patient has a unique ID, medical condition, triage category, waiting time, treatment time, and discharge status. Key methods include:

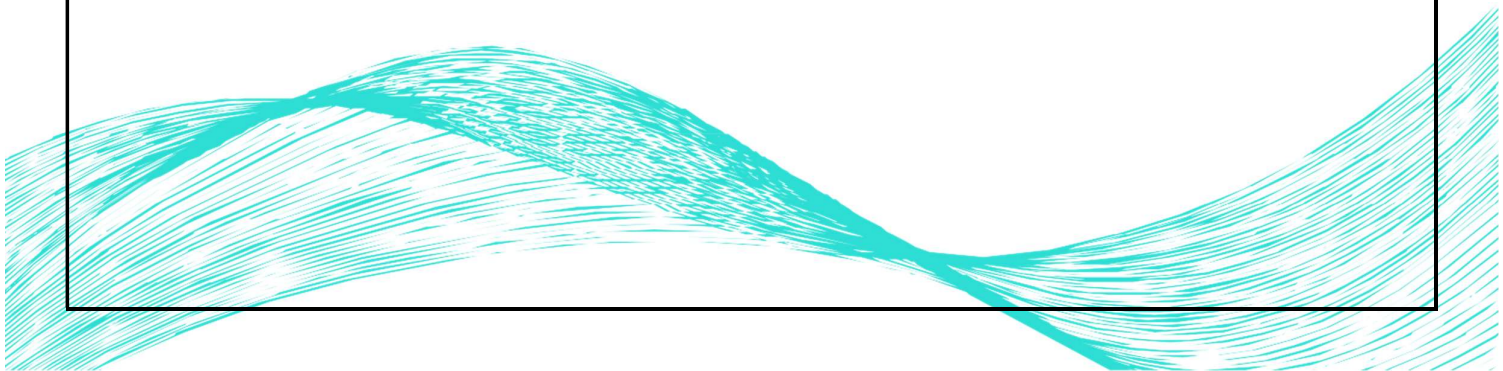
- **determineTriage():** Assigns a triage category based on vitals, severity, and decision points.
- **determineTreatmentTime():** Calculates treatment duration using Pearson VI distribution.
- **determineDischargeTime():** Simulates post-discharge decision time (PDDT) using an exponential distribution.
- **exportPatientRecord():** Exports patient data to a CSV file.

## Resource Management:

The Resources abstract class represents hospital resources such as beds and doctors. It is inherited by the Bed and Doctor classes, which implement resource-specific functionalities. Key attributes and methods include:

- **Bed:**
  - Tracks the patient assigned and the bed type (e.g., Resuscitation, Acute, Sub-acute).
  - Methods to assign and release beds.
- **Doctor:**
  - Categorized into Interns, Junior Residents, Senior Residents, and Registrars/Consultants.
  - Tracks the number of patients assigned and supervises lower-level staff.

## EDQueue Management:



The EDQueue class organizes patients into queues based on their triage levels. Derived classes (EDQ1, EDQ2, etc.) manage specific triage categories, ensuring prioritization of critical cases. Key methods include:

- **addPatient():** Adds a patient to the queue.
- **removePatient():** Removes a patient upon assignment to resources.

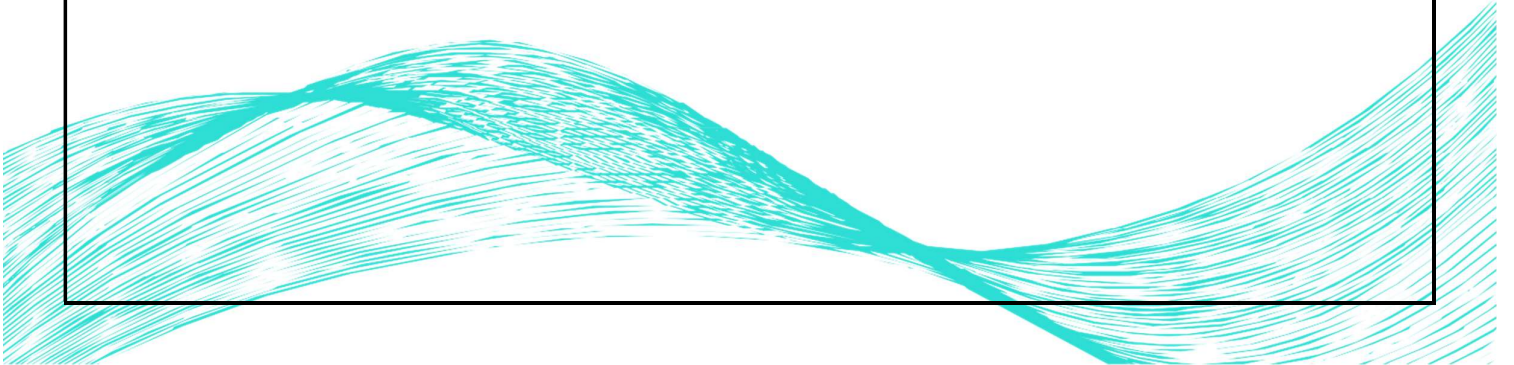
## **EDSimulation Class:**

The EDSimulation class serves as the core of the simulation, orchestrating patient arrivals, resource allocation, and treatment processes. Key functionalities include:

- **Patient Flow:**
  - **generatePatientsForSimulation():** Dynamically generates new patients using Weibull distribution for interarrival times.
  - **handlePatientArrivals():** Processes patient arrivals and assigns them to appropriate queues.
- **Resource Assignment:**
  - **assignBed():** Allocates beds based on availability and triage priority.
  - **assignDoctor():** Assigns doctors while adhering to their capacity limits.
- **Statistics Collection:**
  - **collectStats():** Gathers performance metrics, including average waiting times and bed utilization.

## **Statistics and Reporting:**

The Statistics class compiles simulation results and exports them to a CSV file. It calculates:

- Average waiting time.
  - Bed utilization rates.
- 



- Busy periods using the BusyTime class, which tracks timestamps and patient counts during peak activity.

## **Simulation Clock:**

The SimulationClock class synchronizes real-world and simulated time. It enables precise control over the simulation timeline, ensuring accurate event sequencing. The clock tracks:

- Real-world start time.
- Simulated time progression, where 1 real-world second equals 1 simulated minute.

## **Key Features**

- **Randomized Patient Management:**

Patients are assigned attributes like arrival time, triage categories, and treatment durations using advanced probability distributions (e.g. Weibull, Pearson VI, and exponential).

These methods ensure that patients' flow mimics the real-world Emergency Department (ED) conditions.

- **Object-Oriented Programming Design**

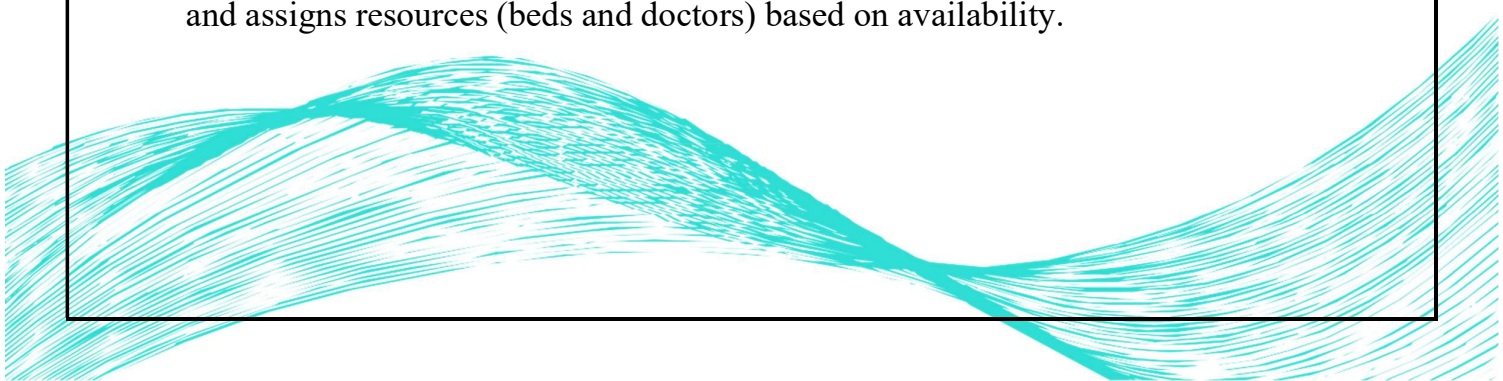
Utilizes a modular OOP approach for scalability and maintainability.

Key classes such as Patient, Doctor, Bed, and EDQueue are interconnected, enabling realistic representation of hospital entities.

Abstract classes like Resources and EDQueue allow for polymorphism, with specific implementations tailored for different hospital resources and patient queues.

- **Simulation Capabilities:**

Handles dynamic patient arrivals, prioritizes critical cases using triage logic, and assigns resources (beds and doctors) based on availability.



Simulates realistic bottlenecks and peak activity periods using the BusyTime and Statistics classes.

- **Comprehensive Reporting**

Exports simulation results, including patient records and performance metrics, to files (e.g., CSV format).

Generates detailed statistics, such as:

Average patient waiting time.

Resource utilization rates.

Identification of busier periods.

- **Hierarchical Resource Management**

Implements a hierarchy for doctors:

Interns, Junior Residents, Senior Residents, Registrars, and Consultants.

Each type of doctor has defined responsibilities and patient load capacities, ensuring realistic supervision and treatment distribution.

- **Advanced Resource Allocation:**

Beds categorized by type (e.g., resuscitation, sub-acute, recliners for overflow scenarios).

Automated reassignment of doctors and beds to handle critical patients during resource shortages.

- **Probability-Based Decision Making:**

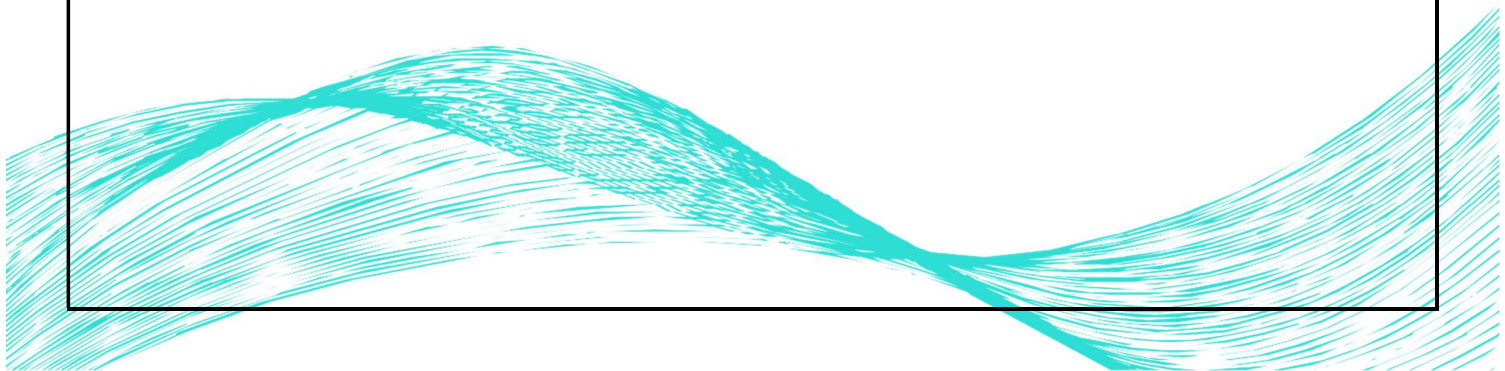
Uses cumulative and truncated probability distributions for specific parameters such as heart rates and treatment durations.

Simulates post-discharge decision times (PDDT) to understand follow-up care needs.

- **Graphical User Interface (GUI):**

Provides a user-friendly interface for visualizing patient flows, resource usage, and real-time simulation progress.

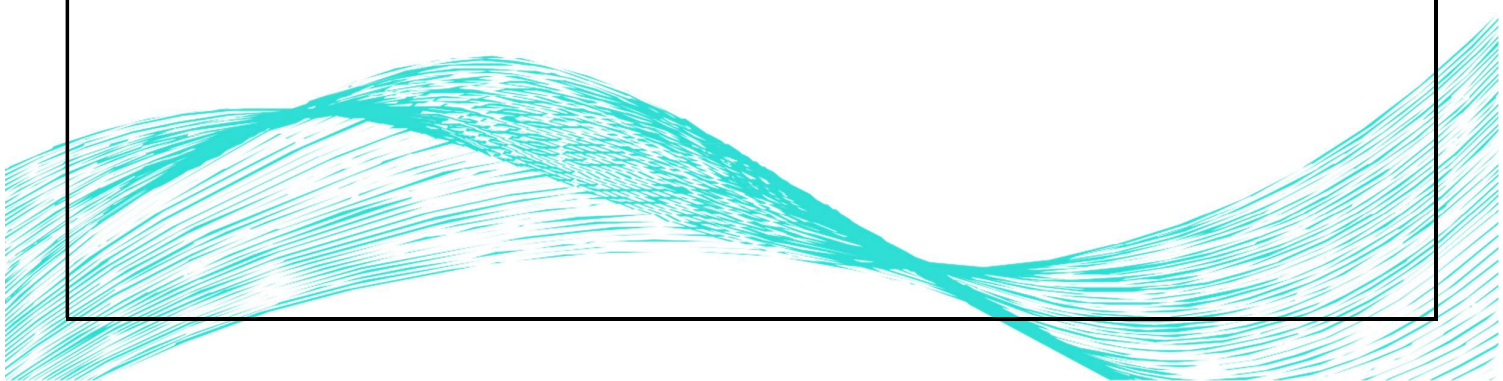
Developed to enhance user interaction and presentation of complex data.



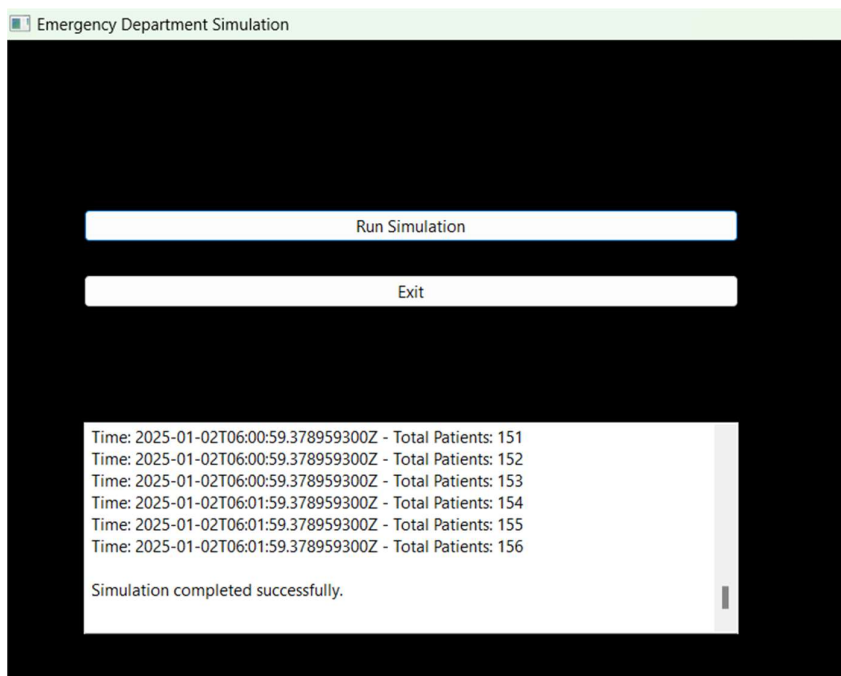
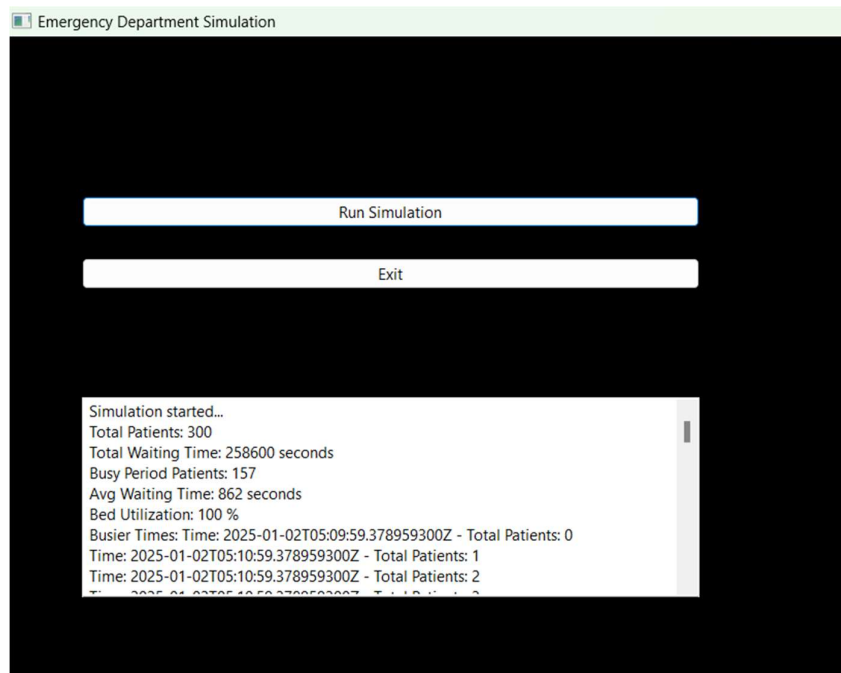
## Results and Outputs:

```
=== Patient Details ===
Patient ID: 52aaae2b-213b-414e-aac4-fca0950bc3b9
Condition: Neurological
Triage Category: 1
Waiting Time: 60.0 seconds
Assigned Consultant/Registrar to patient: 52aaae2b-213b-414e-aac4-fca0950bc3b9
Patient 52aaae2b-213b-414e-aac4-fca0950bc3b9 released from EDQueue for triage category: 1
Treatment Time: 8695.63 seconds
Admission Status: true
Assigned Bed: 93cc52f6-c398-4e92-961f-6c85147e1584
Assigned Doctor: bae4d81c-c89f-443c-92e9-7ebda6b19e22
Post-Discharge Decision Time: 247.92532 seconds
Discharge Time: 2025-01-02T07:33:02.378959300Z
=====
=== Patient Details ===
Patient ID: b956af44-ca1c-4b02-b23f-d9470fb3e3e2
Condition: Paediatric
Triage Category: 3
Waiting Time: 1800.0 seconds
Assigned Intern to patient: b956af44-ca1c-4b02-b23f-d9470fb3e3e2
Patient b956af44-ca1c-4b02-b23f-d9470fb3e3e2 released from EDQueue for triage category: 3
Treatment Time: 819.9231 seconds
Admission Status: true
Assigned Bed: 36281a38-9d1f-4f1a-9188-7d41a1c4d137
Assigned Doctor: 834071a4-7d20-447b-8768-a8b5b08d66bb
Post-Discharge Decision Time: 223.04466 seconds
Discharge Time: 2025-01-02T05:50:21.378959300Z
=====
Data successfully fetched from CSV file.
=== Patient Details ===
Patient ID: 12b37a0f-de83-4721-9126-c654ca45a198
Condition: Respiratory
Triage Category: 3
Waiting Time: 1800.0 seconds
Assigned Intern to patient: 12b37a0f-de83-4721-9126-c654ca45a198
Patient 12b37a0f-de83-4721-9126-c654ca45a198 released from EDQueue for triage category: 3
Treatment Time: 1460.0913 seconds
Admission Status: true
Assigned Bed: 2c6c444c-3af5-4b40-b0b9-a971cef80592
Assigned Doctor: 834071a4-7d20-447b-8768-a8b5b08d66bb
Post-Discharge Decision Time: 299.53876 seconds
Discharge Time: 2025-01-02T06:02:18.378959300Z
```

**Figure of Console and Hospital Management**







**Figure of GUI and Statistics**



# Challenges:

## Challenges Faced During Development

### 1. Implementing Probability Distributions

1. **Challenge:** Mapping real-world ED scenarios to mathematical probability distributions (e.g., matching disease severity to triage levels).
2. **Resolution:** Conducted research on statistical models commonly used in healthcare. Utilized libraries like Java's `java.util.Random` or external libraries (e.g., Apache Commons Math) to generate distributions that align with realistic data.

### 2. Debugging and Data Validation

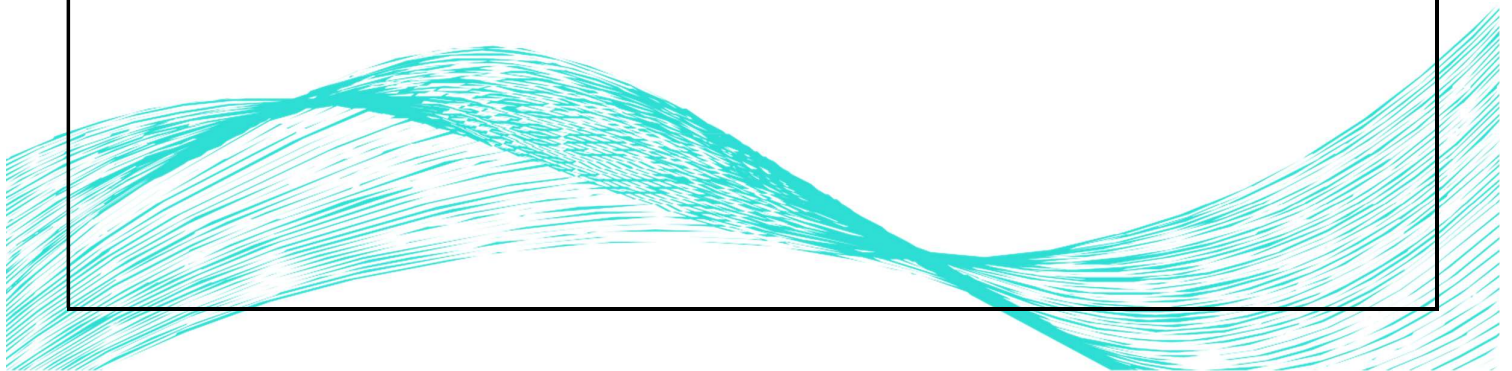
1. **Challenge:** Ensuring data integrity, such as validating records from the CSV file, and debugging issues with invalid inputs or edge cases.
2. **Resolution:** Implemented robust validation functions (`validateRecord`) and extensive error handling to catch and address issues early. Used test cases with varied data to ensure reliability.

### 3. Scalability

1. **Challenge:** Designing a system capable of scaling as more entities (e.g., patients, diseases) and features are added.
2. **Resolution:** Used an Object-Oriented Programming (OOP) approach to create modular and reusable components, allowing for easy expansion.

### 4. Simulation Accuracy

1. **Challenge:** Achieving realistic simulation results that match real-world ED operations.
2. **Resolution:** Conducted iterative testing, adjusting parameters like arrival rates and triage weights until the simulation closely resembled observed trends.



# **Conclusion:**

## **Achievements**

1. Successfully developed a modular and scalable simulation system for an Emergency Department using OOP principles.
2. Implemented realistic patient flow by utilizing probability distributions for arrival times, disease severity, and resource allocation.
3. Validated the integrity of input data, ensuring that only accurate and meaningful records were processed.
4. Created a flexible platform for testing ED operations under various scenarios.

## **Potential for Further Development**

### **Integration of Real-Time Patient Data**

1. Incorporate live data from hospital information systems for dynamic simulations.

### **Machine Learning Integration**

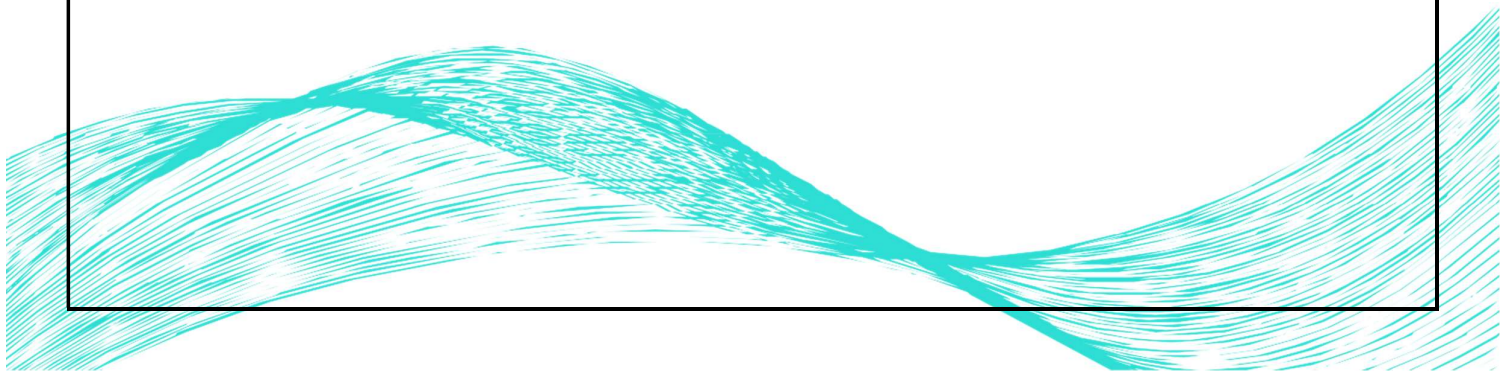
1. Use machine learning models to predict patient outcomes and resource needs based on historical data.

### **Enhanced Visualization**

1. Add graphical interfaces to visualize patient flow, resource allocation, and bottlenecks in real time.

### **Multi-Department Simulation**

1. Extend the system to include other hospital departments, such as ICUs or operating rooms, for a holistic view of hospital operations.



## References:

We conducted thorough research on triage categories by studying various standards such as ATS and ESI e.t.c, reviewing informative YouTube video and analyzing relevant documents. We have also attached a document as a reference from which we extracted critical insight with regards to the triage system.

1. **Emergency Severity Index Handbook.pdf**
2. **Project Proposal Document**
3. **[Java Oracle](#)**

