**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**-----🙣 🕮 🙡-----**

**Sinh viên thực hiện:**

**Mã lớp: 151965**

**BÁO CÁO BÀI TẬP**

**HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG – IT3103**

**CHỦ ĐỀ 10: CPU SCHEDULER**

**Hà Nội, tháng 12 năm 2024**

|  |  |
| --- | --- |
| **Lê Nhật Sơn** | **20225918** |
| **Phạm Mạnh Quyết** | **20225663** |
| **Nguyễn Đức Quang** | **20225913** |
| **Nguyễn Nhuận Quang** | **20225914** |
| **Nguyễn Khắc Quang** | **20225760** |



**0. Phân công công việc** **3**

1. **Mô tả bài toán** **3**

1.1. Yêu cầu **3**

1.2. Giao diện **3**

1.2.1. Giao diện trang chủ **3**

1.2.2. Giao diện mô phỏng **4**

1.2.3. Giao diện hướng dẫn **4**

1. **Thiết kế** **5**

2.1. Use Case Diagram **5**

2.2. Class Diagram **6**

1. **Các kỹ thuật OOP trong thiết kế** **9**

3.1. Kế thừa **13**

3.2. Đa hình **13**

3.3. Đóng gói **13**

3.4. Trừu tượng **14**

3.5. Các mối quan hệ OOP **14**

3.5.1. Association (Liên kết) **14**

3.5.2. Aggregation (Kết tập) **14**

3.5.3. Composition (Hợp thành) **14**

**4. Demo** **15**

|  |  |  |
| --- | --- | --- |
| **Họ tên** | **MSSV** | **Công việc** |
| Lê Nhật Sơn | 20225918 | Tham gia xây dựng giao diện, liên kết view với controller |
| Phạm Mạnh Quyết | 20225663 | Xây dựng usecase, class diagram, viết mã nguồn 3 thuật toán, class, Process, sửa GUI, viết mã nguồn main menu, sửa logic bảng process, live output, schedule metric |
| Nguyễn Đức Quang | 20225913 | sửa class Process, slide, viết báo cáo |
| Nguyễn Nhuận Quang | 20225914 | slide |
| Nguyễn Khắc Quang | 20225760 | X |

**0. Phân công công việc**

**1. Mô tả bài toán**

**1.1. Yêu cầu**

- Dự án mô phỏng các thuật toán lập lịch CPU phổ biến, bao gồm:

+ FCFS (First-Come-First-Serve): Thuật toán thực hiện các tiến trình theo thứ tự đến.

+ SJN (Shortest Job Next)/ SJF: Ưu tiên các tiến trình có thời gian xử lý ngắn nhất, giúp tối ưu thời gian chờ trung bình nhưng yêu cầu biết trước thời gian xử lý của tiến trình.

+ RR (Round Robin): Phân bổ thời gian cố định cho từng tiến trình theo vòng tròn, đảm bảo công bằng nhưng tăng chi phí chuyển đổi ngữ cảnh.

- Ứng dụng sẽ minh họa cách từng thuật toán xử lý tiến trình và ảnh hưởng của chúng đến các chỉ số hệ thống, chẳng hạn như:

+ Thời gian chờ trung bình (Waiting Time)

+ Thời gian hoàn thành (Turnaround Time)

**1.2. Giao diện**

**1.2.1. Giao diện trang chủ (Main Menu):**

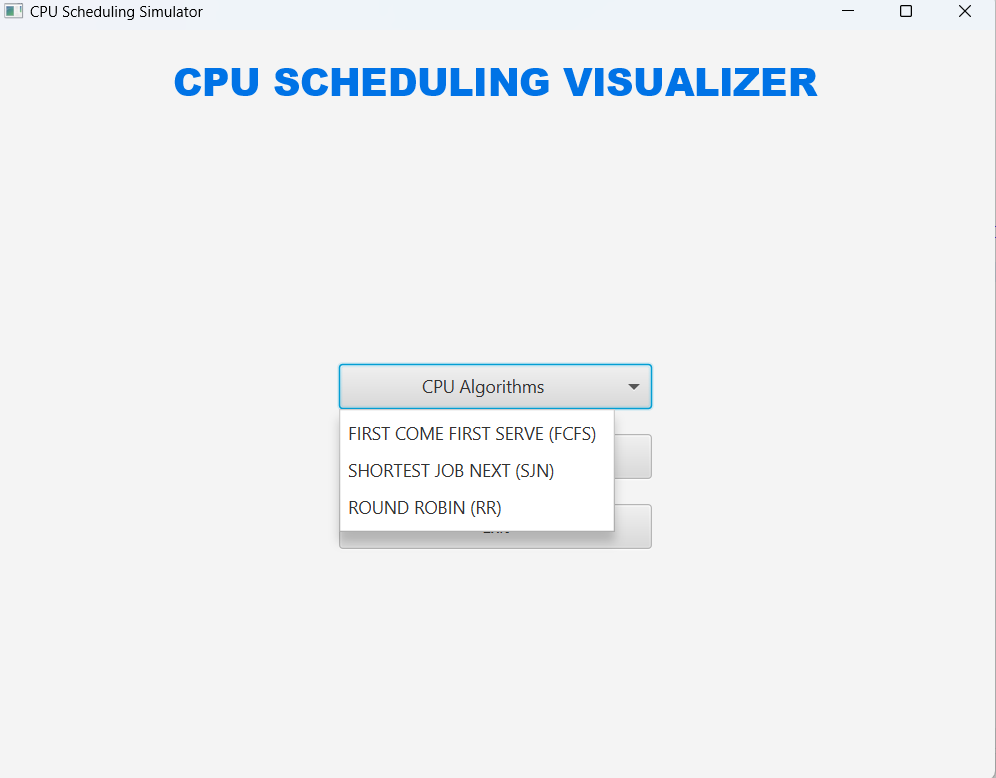
- Cho phép người dùng lựa chọn thuật toán lập lịch.

- Nút "Trợ giúp" (Help) cung cấp hướng dẫn về các thuật toán.

- Nút "Thoát" (Exit) thoát ứng dụng.

A screenshot of a computer

Description automatically generated

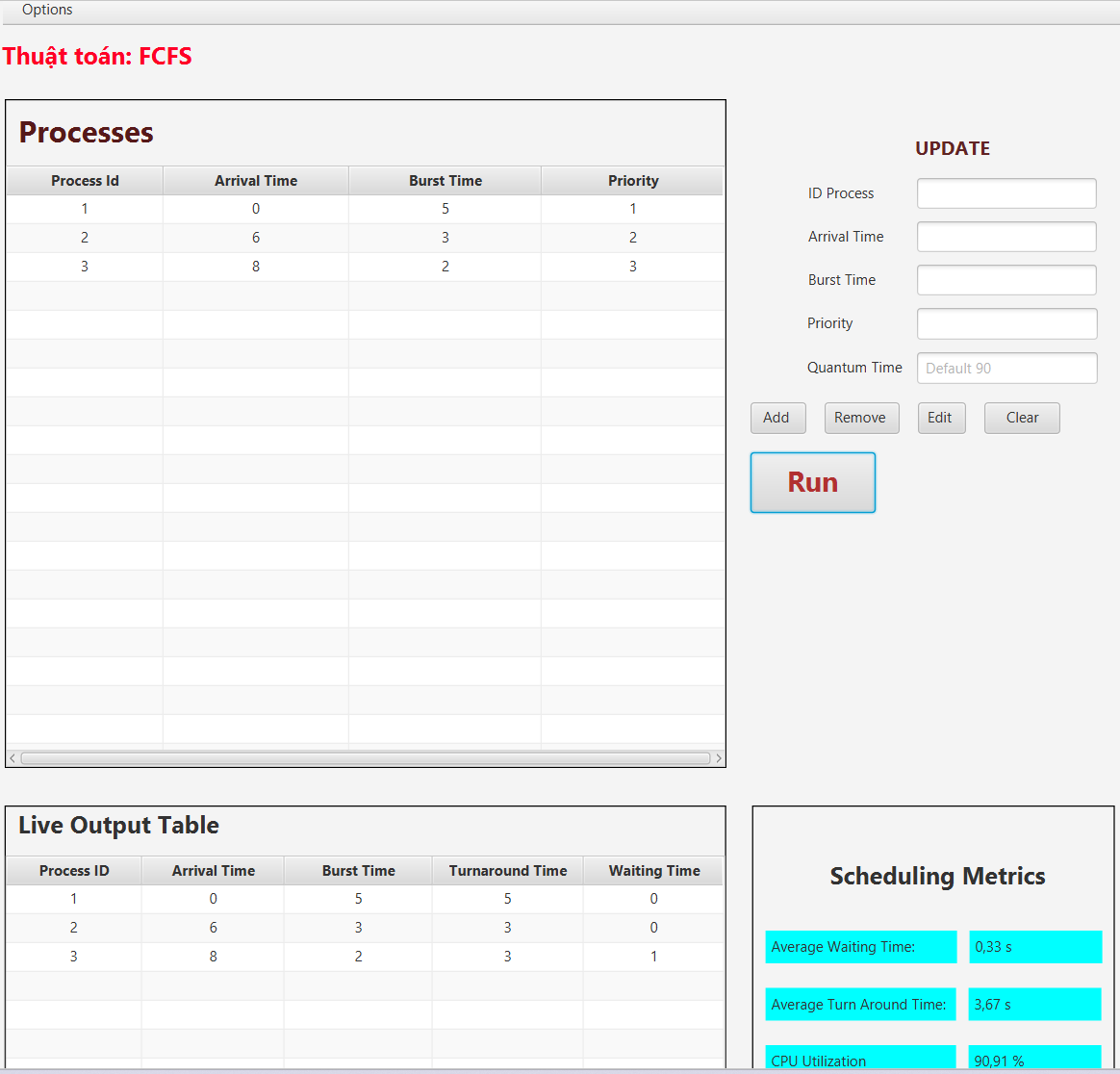


**1.2.2. Giao diện mô phỏng (Simulation Screen):**

- Hiển thị danh sách các tiến trình với thông tin như: ID, thời gian đến, thời gian xử lý, và mức độ ưu tiên.

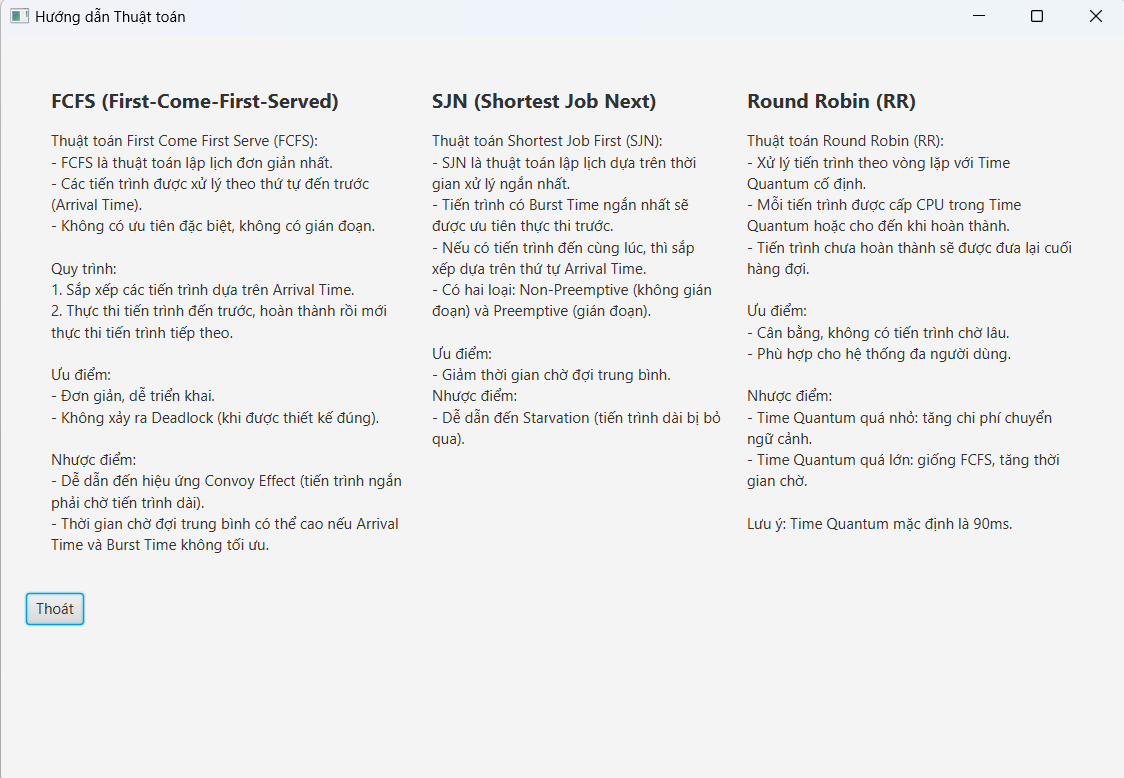
- Minh họa trực quan thứ tự thực hiện tiến trình theo biểu đồ Gantt động.

- Tính toán và hiển thị các chỉ số hiệu suất.

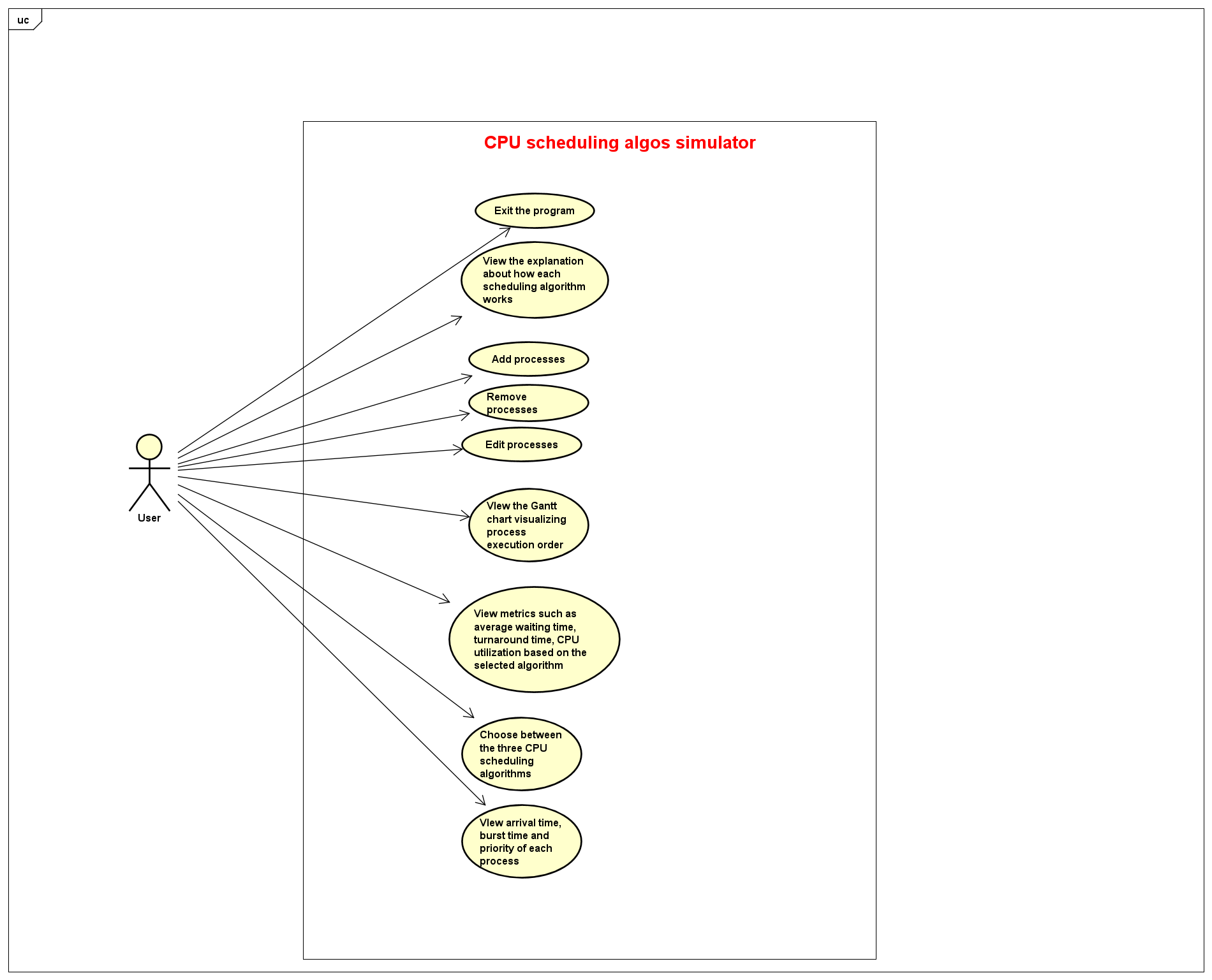


**1.2.3. Hướng dẫn (Help Menu):**

- Mô tả ngắn gọn cách hoạt động của từng thuật toán.



**2. Thiết kế**

**2.1. Usecase Diagram**

- Người dùng có thể chọn thuật toán để thực hiện lập lịch

- Người dùng có thể mở cửa số help để xem giải thích về các hoạt động của thuật toán

- Người dùng có thể thoát chương trình

- Người dùng có thể thêm, xóa, sửa process

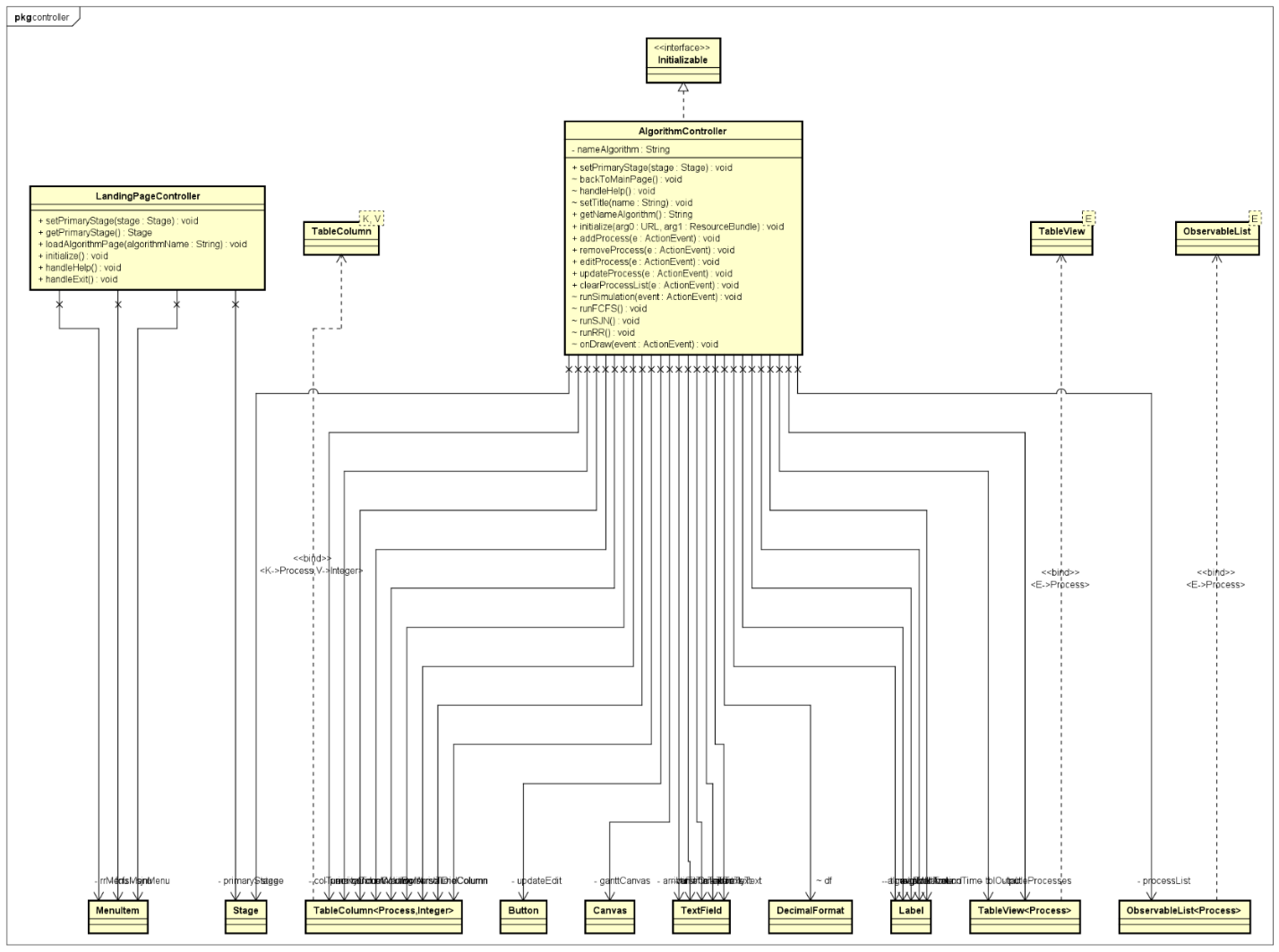
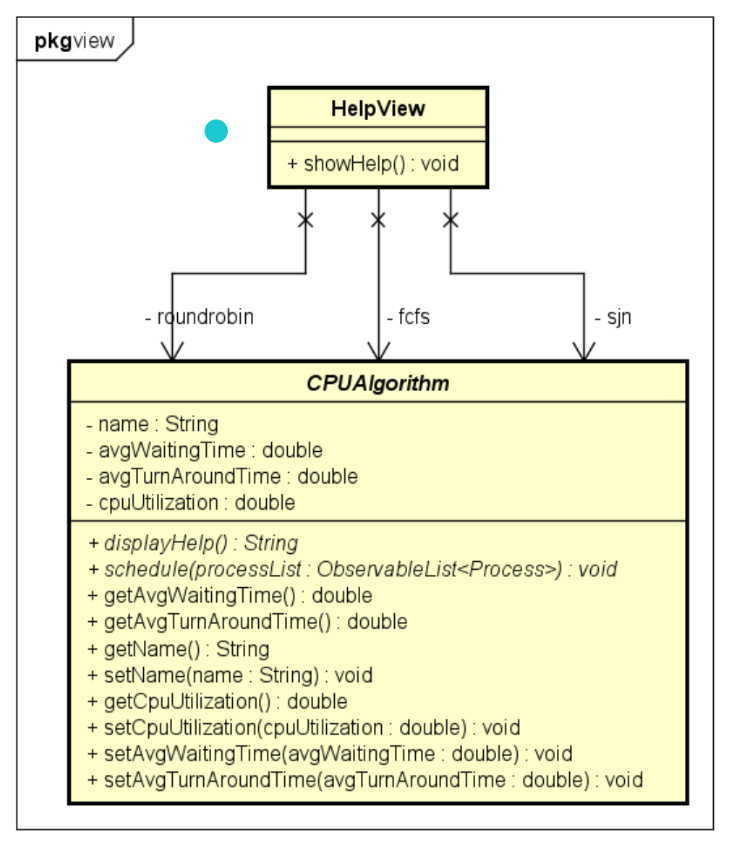
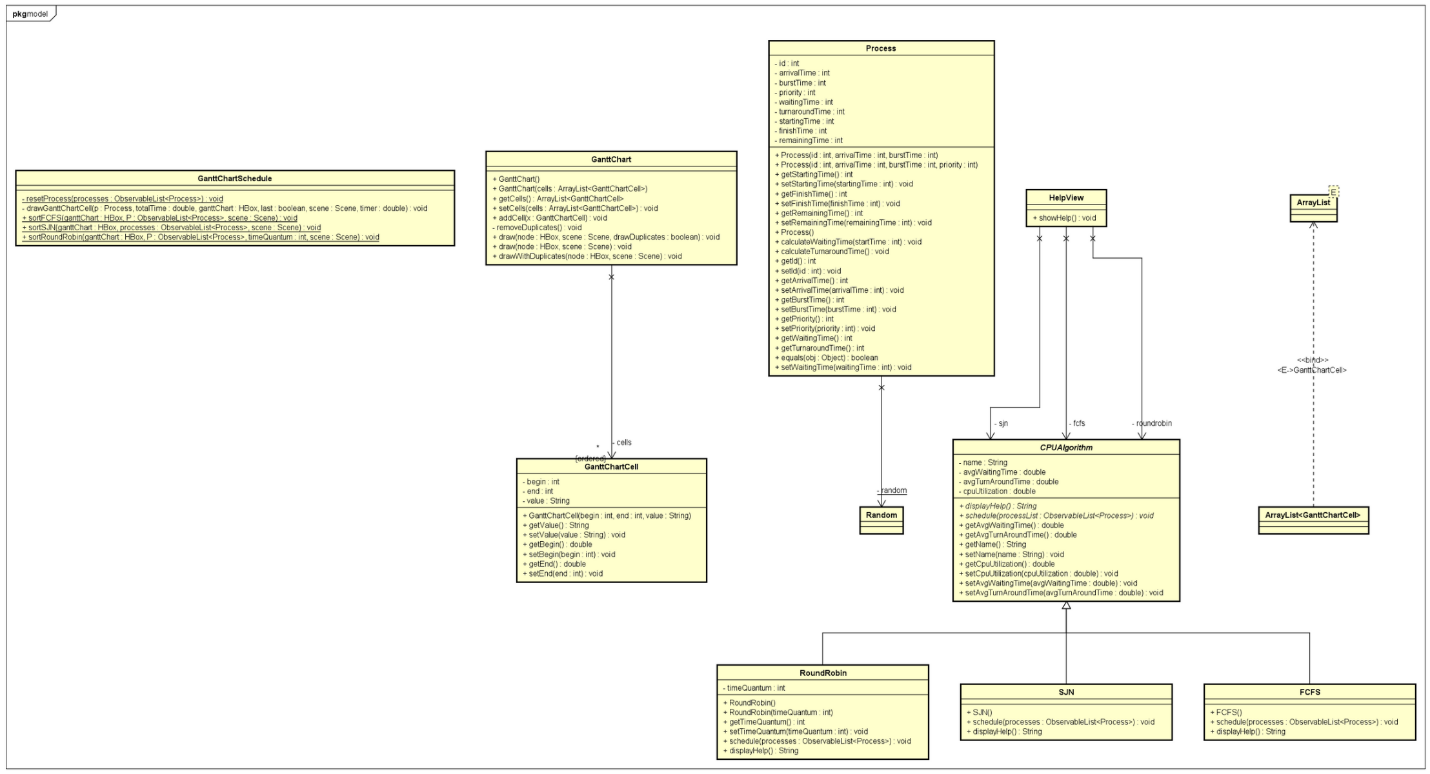
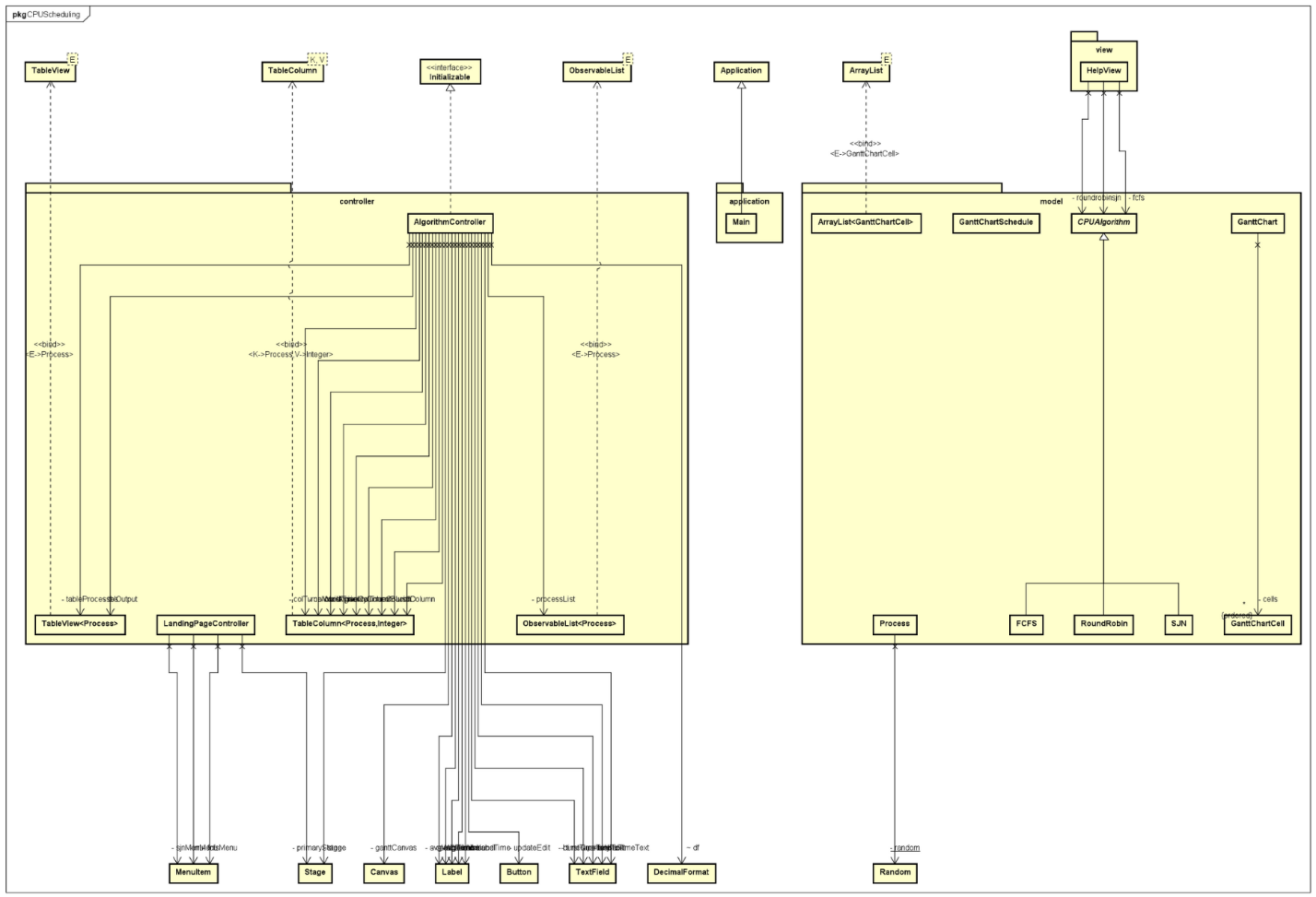
- Người dùng có thể xem danh sách process hiện tại đang có

- Người dùng có thể xem bảng thời gian chờ, thời gian quay vòng sau khi chạy thuật toán

- Người dùng có thể xem thời gian chờ trung bình, thời gian quay vòng trung bình, % sử dụng CPU sau khi chạy thuật toán

- Người dùng có thể xem biểu đồ gantt chart sau khi chạy thuật toán

**2.2. Class Diagram**

****

**a) Model package**

**- Class Process:** Đại diện cho process trong hệ thống, lưu trữ thông tin cần thiết và hỗ trợ tính toán các chỉ số liên quan.

|  |  |
| --- | --- |
| **Thuộc tính** | |
| id: int | Mã định danh cho mỗi process. |
| arrivalTime: int | Thời gian mà process xuất hiện trong hệ thống. |
| burstTime: int | Thời gian cần thiết để process thực thi. |
| priority: int | Độ ưu tiên của process (trong thuật toán ưu tiên). |
| waitingTime: int | Thời gian chờ của process trước khi được thực thi. |
| turnaroundTime: int | Thời gian hoàn thành của process kể từ khi nó đến hệ thống. |

|  |  |
| --- | --- |
| **Phương thức** | |
| calculateWaitingTime(startTime: int): void | Tính toán thời gian chờ của process dựa trên thời điểm bắt đầu. |
| calculateTurnaroundTime(): void | Tính toán thời gian hoàn thành của process. |

**- Class CPUAlgorithm:**

|  |  |
| --- | --- |
| **Thuộc tính** | |
| name: char | Tên của thuật toán lập lịch (ví dụ: FCFS, SJF, ...). |
| avgWaitingTime: double | Thời gian tính toán trung bình của thuật toán |
| avgTurnAroundTime: double | Thời gian quay vòng trung bình của thuật toán |

|  |  |
| --- | --- |
| **Phương thức** | |
| getName(): char | Lấy tên của thuật toán. |
| setName(name: char): void | Đặt tên cho thuật toán. |
| displayHelp(): abstract void | Hiển thị hướng dẫn hoặc mô tả về thuật toán. |
| schedule(ObservableList<Process> processList): abstract void | Cách tính toán của thuật toán |

**- Class FCFS (Kế thừa từ CPUAlgorithm):**

|  |  |
| --- | --- |
| **Phương thức** | |
| schedule(ObservableList<Process> processes): void | Triển khai thuật toán lập lịch First Come First Serve (FCFS). |
| displayHelp(): String | Hiển thị mô tả chi tiết về thuật toán FCFS |

|  |  |
| --- | --- |
| **Phương thức** | |
| schedule(ObservableList<Process> processes): void | Triển khai thuật toán lập lịch Shortest Job First (SJN). |
| displayHelp(): String | Hiển thị mô tả chi tiết về thuật toán SJN |

**- Class SJN (Kế thừa từ CPUAlgorithm):**

**- Class RoundRobin (Kế thừa từ CPUAlgorithm):**

|  |  |
| --- | --- |
| **Thuộc tính** | |
| timeQuantum: int | Khoảng thời gian tối đa mà một process có thể chạy trong một lượt. |

|  |  |
| --- | --- |
| **Phương thức** | |
| schedule(ObservableList<Process> processes): void | Triển khai thuật toán lập lịch Round Robin |
| displayHelp(): String | Hiển thị mô tả chi tiết về thuật toán RR |

**b) View package**

**- Class HelpView:**

|  |  |
| --- | --- |
| **Thuộc tính** | |
| CPUAlgorithm fcfs = **new** FCFS(); | Thuộc tính này là một thể hiện của lớp FCFS: mục đích lấy ra phương thức displayHelp() của thuật toán |
| CPUAlgorithm sjn = **new** SJN(); | Thuộc tính này là một thể hiện của lớp SJN: mục đích lấy ra phương thức displayHelp() của thuật toán |
| CPUAlgorithm roundrobin = **new** RoundRobin(); | Thuộc tính này là một thể hiện của lớp RoundRobin: mục đích lấy ra phương thức displayHelp() của thuật toán |

|  |  |
| --- | --- |
| **Phương thức** | |
| showHelp(): void | Phương thức này tạo ra một cửa sổ để hiển thị hướng dẫn cho các thuật toán lập lịch CPU (FCFS, SJN, và Round Robin). |

**- LandingPage.fxml:**

|  |  |
| --- | --- |
| **Giao diện** | |
| Option: *CPU Algorithms* | Chọn thuật toán muốn thực hiện |
| Help | Mở cửa sổ help |
| Exit | Thoát chương trình |

**- sample.fxml:**

|  |  |
| --- | --- |
| **Giao diện** | |
| Option back và help | Back :trở lại của sổ main và Help: hiện thị cửa sổ help |
| Table Process và Update | Hiện thị danh sách process và cập nhật bảng |
| Live output và Scheduling Metrics | Hiện thị kết quả tĩnh khi thực hiện chạy thuật toán |
| Gannt Chart | Hiện thị kết quả động theo thời gian thực khi chạy thuật toán |

**b) Controller package**

**- Class AlgorithmController**

|  |  |
| --- | --- |
| **Thuộc tính** | |
| String nameAlgorithm | Tên của thuật toán |
| Label algorithmLabel | Hiện thị tên thuật toán |
| TableView<Process> tableProcesses; | Bảng các process gồm id, arrivalTime, burstTime, priority |
| TableColumn<Process, Integer> idColumn | Cột id trong bảng process |
| TableColumn<Process, Integer> arrivalTimeColumn | Cột arrivalTime trong bảng process |
| TableColumn<Process, Integer> burstTimeColumn | Cột burstTime trong bảng process |
| TableColumn<Process, Integer> priorityColumn | Cột priority trong bảng process |
| TextField idText | Id trường text nhập id nếu muốn add, remove, edit |
| TextField arrivalTimeText | Id trường text nhập arrivalTime nếu muốn add, remove, edit process |
| TextField burstTimeText | Id trường text nhập burstTimenếu muốn add, remove, edit |
| TextField priorityText | Id trường text nhập prioritynếu muốn add, remove, edit |
| TextField timeQuantumText | Id trường text nhập Quantum nếu chạy RoundRobin |
| Button runButton; | Id của nút Run |
| ObservableList<Process> processList | Khởi tạo danh sách process |
| TableView<Process> tblOutput | Bảng live output |
| TableColumn<Process, Integer> colID | Cột id của process trong bảng live output |
| TableColumn<Process, Integer> colArrival | Cột arrivalTime của process trong bảng live output |
| TableColumn<Process, Integer> colBurst | Cột burstTime của process trong bảng live output |
| TableColumn<Process, Integer> colTurnaround | Cột turnAround của process trong bảng live output |
| TableColumn<Process, Integer> colWaiting | Cột waitingTime của process trong bảng live output |
| Label avgWaitTime | Id label avgWaitTime trong  *Scheduling Metrics* |
| Label avgTurnAroundTime | Id label avgTurnAroundTime trong  *Scheduling Metrics* |
| Label cpuUtilization | Id label cpuUtilization trong  *Scheduling Metrics* |
| Stage stage; | Stage để set stage về main menu |

|  |  |
| --- | --- |
| **Phương thức** | |
| **void** backToMainPage() | Chuyển về main menu |
| **void** handleHelp() | Hiển thị cửa sổ help |
| **void** setTitle(String name) | Set tên của thuật toán |
| **void** initialize(URL arg0, ResourceBundle arg1) | Set init các cột của bảng table view, live output, scheduling metrics |
| **void** addProcess(ActionEvent e) | Thêm process |
| **void** removeProcess(ActionEvent e) | Xóa process |
| **void** editProcess(ActionEvent e) | Edit process |
| **void** updateProcess(ActionEvent e) | Update process sau khi edit |
| **void** clearProcessList(ActionEvent e) | Xóa toàn bộ danh sách process |
| **void** runSimulation(ActionEvent event) | Chạy thuật toán |
| **void** runFCFS() | Chạy thuật toán FCFS |
| **void** runSJN() | Chạy thuật toán SJN |
| **void** runRR() | Chạy thuật toán RR |

**- Class LandingPageController**

|  |  |
| --- | --- |
| **Thuộc tính** | |
| Stage primaryStage | Stage chính |
| MenuItem fcfsMenu | Id của menu item fcfsMenu |
| MenuItem sjnMenu | Id của menu item sjnMenu |
| MenuItem rrMenu | Id của menu item rrMenu |

|  |  |
| --- | --- |
| **Phương thức** | |
| **void** initialize() | Set init cho các menu item |
| **void** handleHelp() | Hiển thị cửa số help |
| **void** handleExit() | Thoát main menu |
| **void** loadAlgorithmPage(String algorithmName) | Load trang thuật toán |

**3. Các kỹ thuật OOP trong thiết kế**

**3.1. Kế thừa**

- Kế thừa được sử dụng để chia sẻ các thuộc tính và phương thức từ lớp cha đến các lớp con. Trong dự án này ta có lớp cha trừu tượng CPUAlgorithm và các lớp con FCFS, RoundRobin, SJN.

- Lớp CPUAlgorithm cung cấp các thuộc tính và phương thức cơ bản như avgWaitingTime, avgTurnAroundTime và schedule().

**3.2. Đa hình**

- Đa hình cho phép các lớp con triển khai các phương thức theo cách riêng của mình, mặc dù các phương thức này được định nghĩa chung trong lớp cha.

- Trong dự án này ta có phương thức schedule() trong CPUAlgorithm được định nghĩa dưới dạng abstract trong CPUAlgorithm và được triển khai khác nhau trong các lớp FCFS, RoundRobin, và SJN.

**3.3. Đóng gói.**

- Đóng gói được sử dụng để bảo vệ dữ liệu và kiểm soát truy cập thông qua các phương thức.

+ Các thuộc tính của lớp Process như arrivalTime, burstTime được khai báo private và chỉ có thể truy cập thông qua các phương thức getter và setter.

**3.4. Trừu tượng**

- Trừu tượng giúp ẩn đi các chi tiết triển khai và chỉ tập trung vào các tính năng chính.

+ Lớp CPUAlgorithm được khai báo là abstract, chỉ cung cấp cấu trúc chung như phương thức schedule() mà không quan tâm đến cách cài đặt cụ thể. Các lớp con FCFS, SJN, RoundRobin triển khai chi tiết phương thức này theo thuật toán của riêng mình.

**3.3. Các mối quan hệ OOP**

**3.3.1. Association (Liên kết)**

- Một lớp sử dụng hoặc tương tác với một lớp khác mà không thuộc về nó.

+ Lớp AlgorithmController sử dụng lớp CPUAlgorithm để thực hiện các thuật toán.

+ Lớp Process được sử dụng bởi CPUAlgorithm để lưu trữ thông tin tiến trình.

**3.3.2. Aggregation (Kết tập)**

- Một lớp chứa tham chiếu đến các đối tượng của lớp khác, nhưng các đối tượng này không bị hủy khi lớp chứa bị hủy.

+ Lớp LandingPageController chứa các thuật toán (FCFS, SJN, RoundRobin) như các thành phần riêng biệt.

**3.3.3. Composition (Hợp thành)**

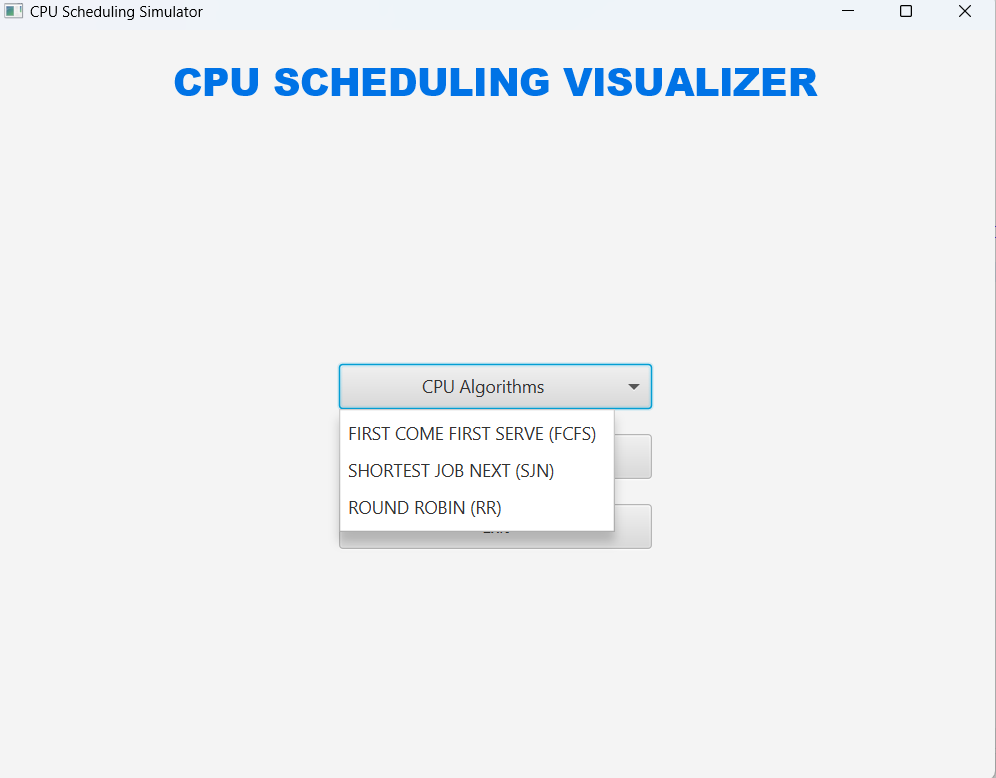
- Một lớp chứa các đối tượng của lớp khác và kiểm soát hoàn toàn vòng đời của chúng.

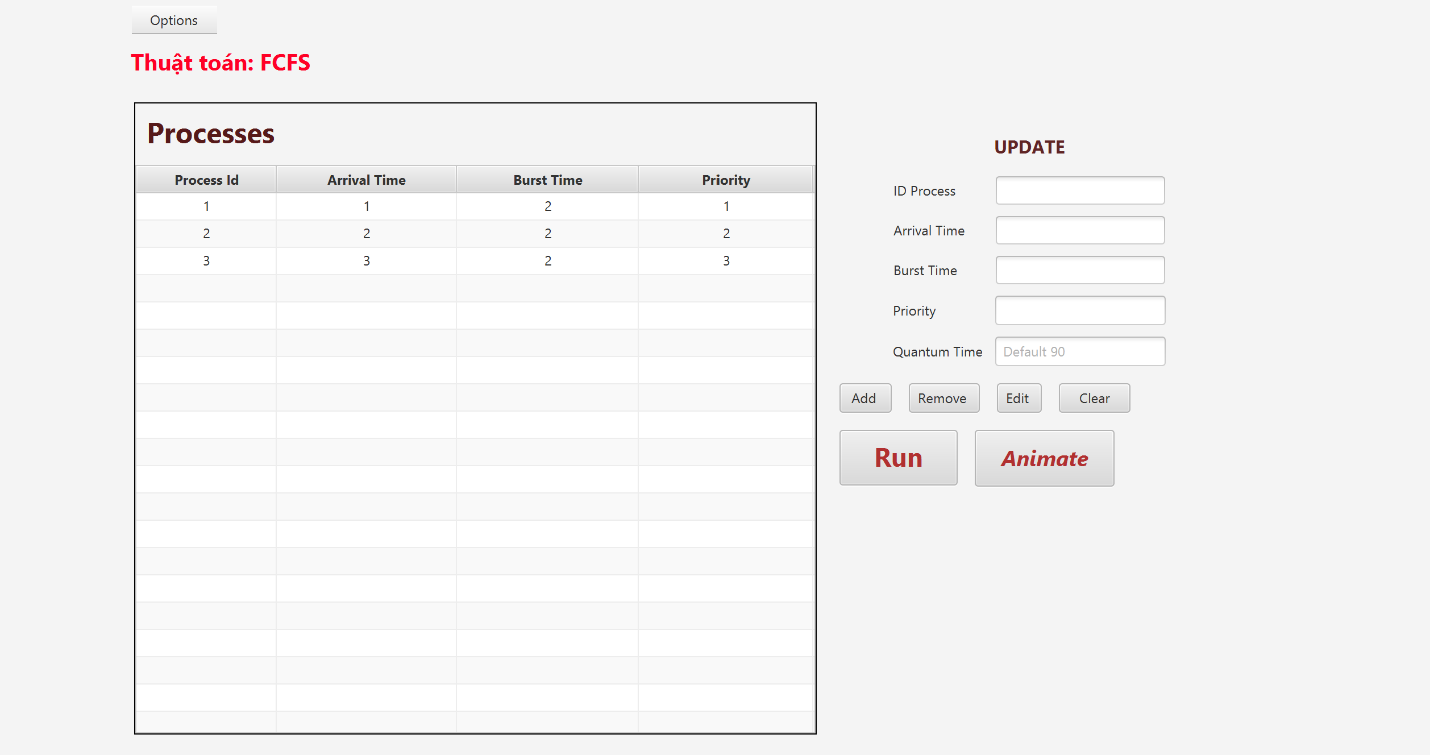
+ Lớp AlgorithmController chứa danh sách tiến trình (ObservableList<Process>) và điều khiển chúng.

**4. Demo**

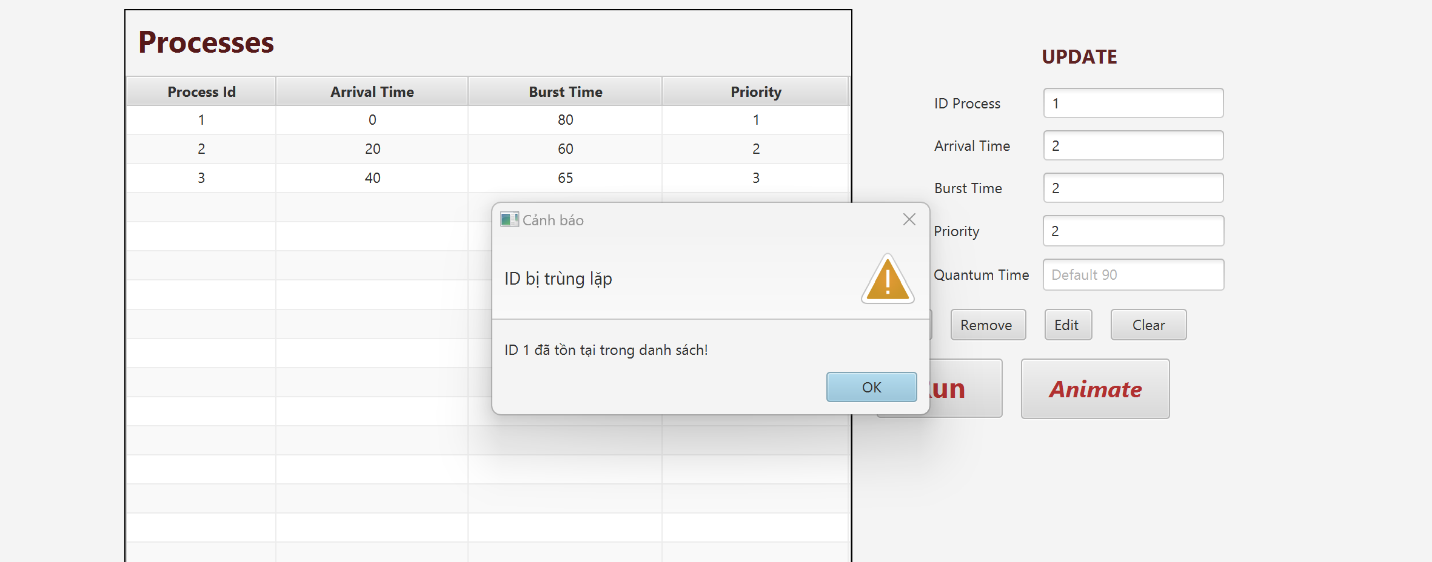
**- Lựa chọn thuật toán**

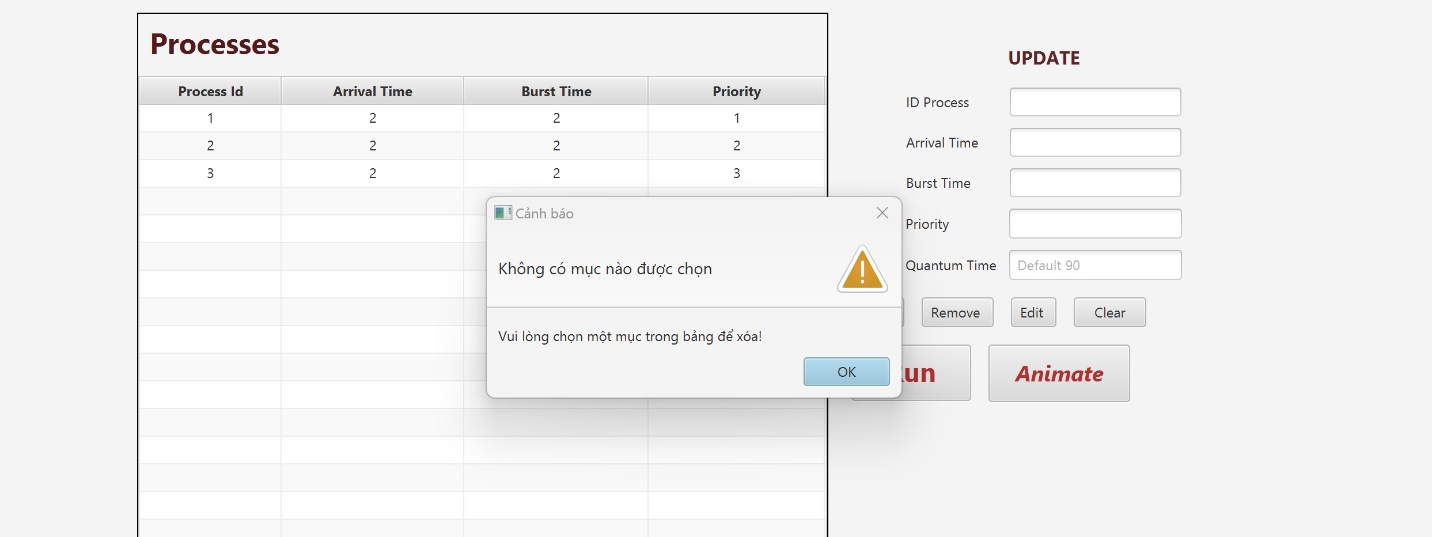
A screenshot of a computer

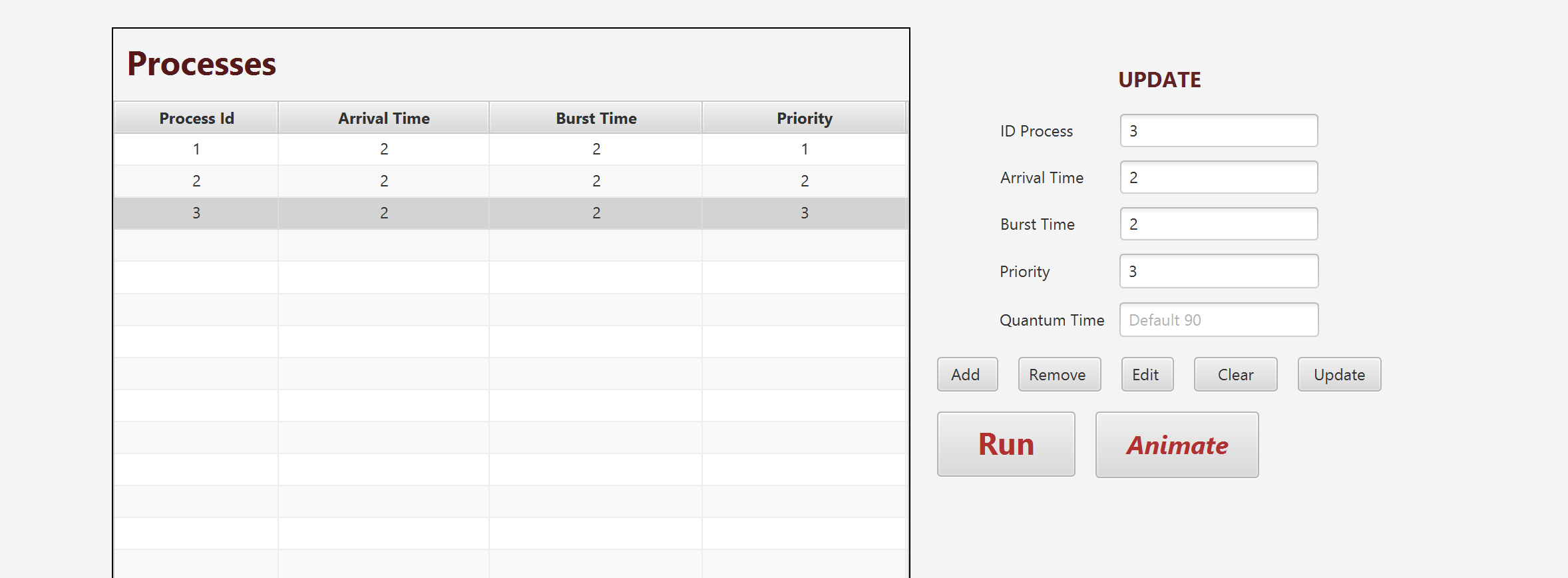
Description automatically generated

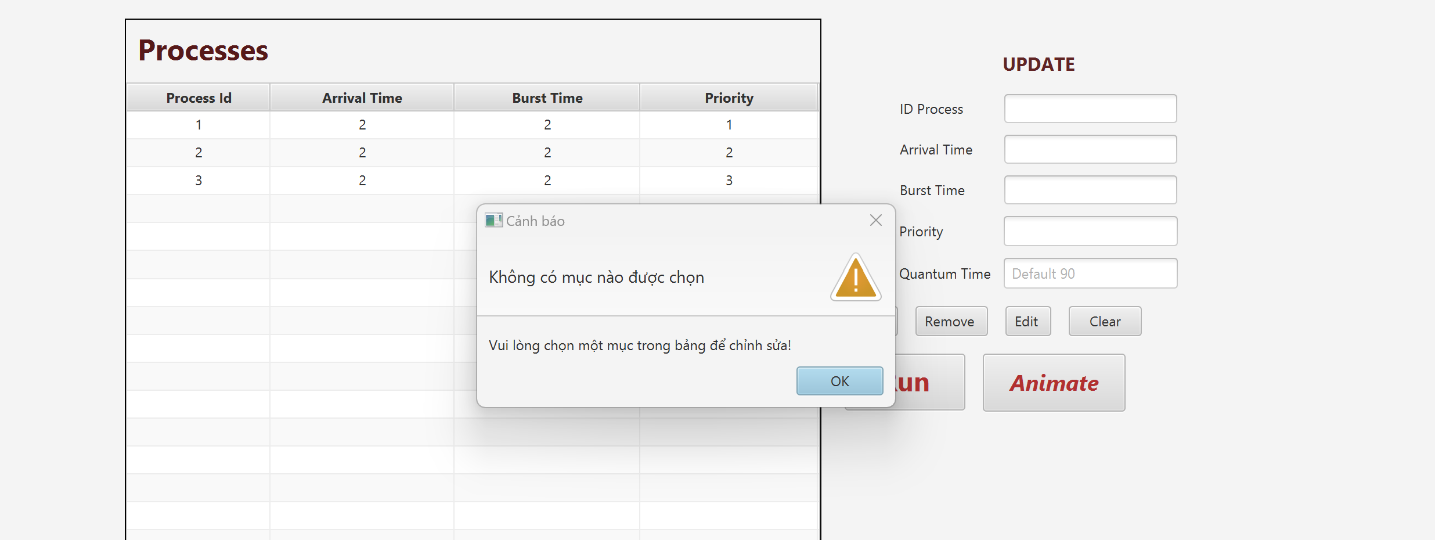
**- Thêm: Điền thông tin (ID, arrival time, burst time, …) và chọn ‘Add’ để thêm các tiến trình.**

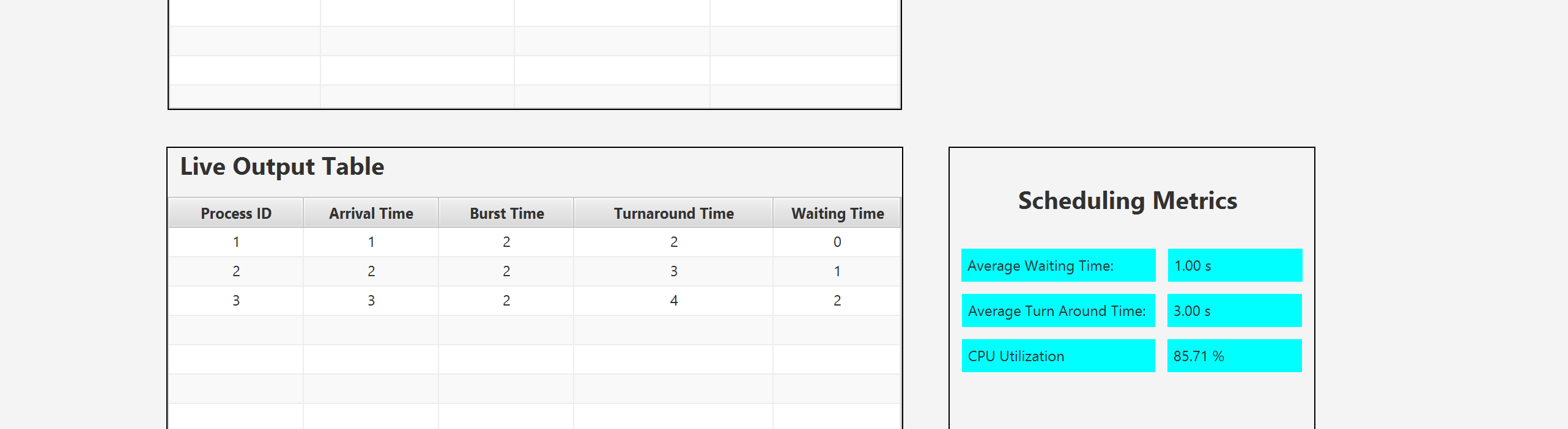
**Nếu ID đã tồn tại, chương trình sẽ hiển thị thông báo:**

****

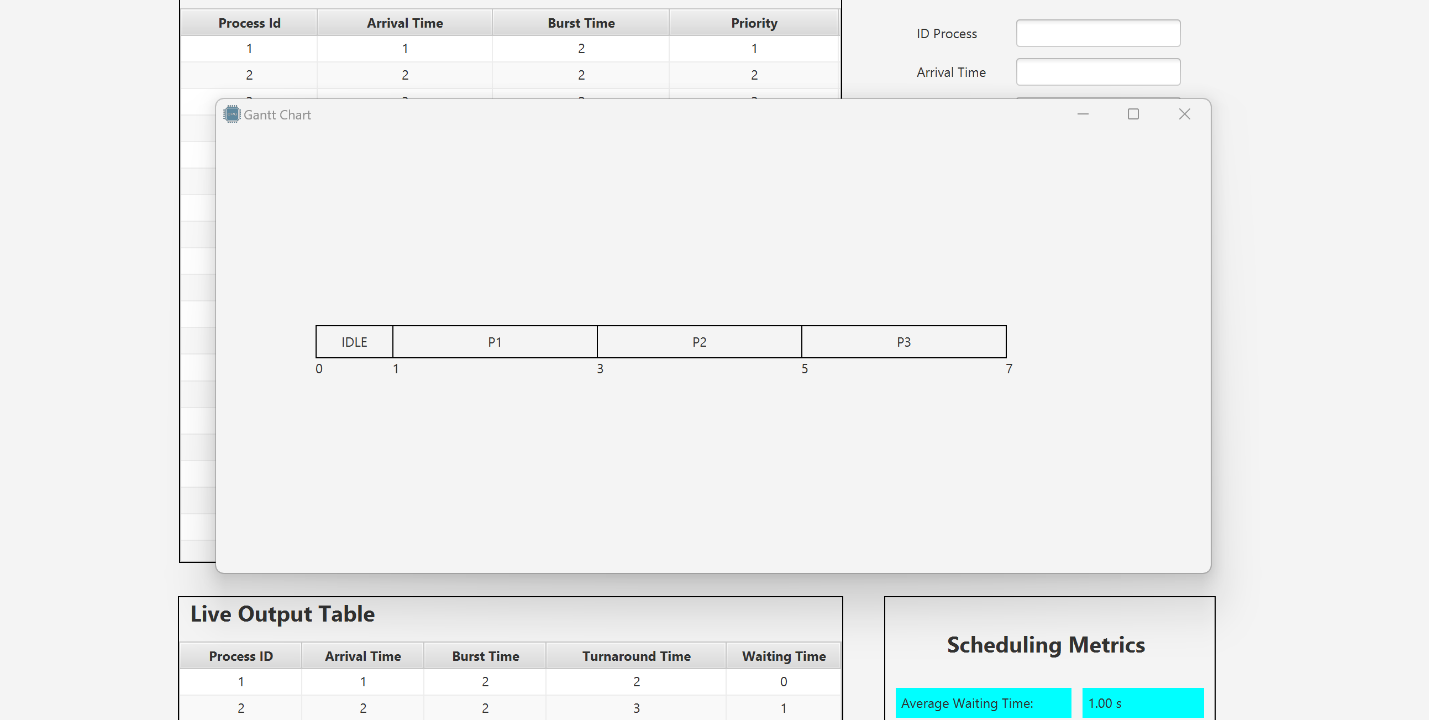
**- Xóa: Chọn một tiến trình đã có trong bảng và chọn ‘Remove’, nếu không chọn tiến trình cần xóa, chương trình sẽ hiển thị thông báo**

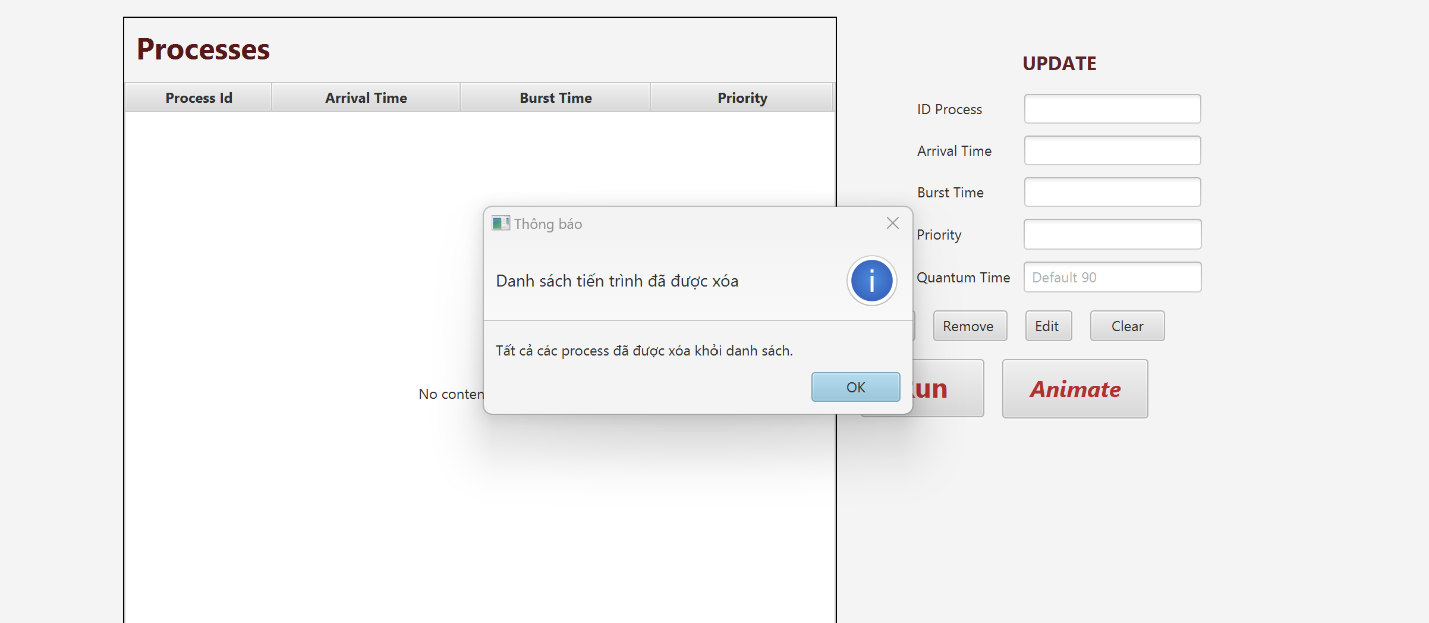
**- Sửa: Chọn 1 tiến trình trong bảng và chọn ‘Edit’, sửa thông tin và chọn ‘Update’.**

**Nếu không chọn tiến trình trong bảng và chọn ‘Edit’ chương trình sẽ hiển thị thông báo**

**- Chọn ‘Run’, chương trình sẽ hiển thị các thông số**

**- Chọn Animate, chương trình sẽ hiển thị Gantt Chart**

****

**- Clear để xóa tất cả tiến trình**