[Phân Tích Giao Diện Người Dùng 1](#_Toc199160677)

[Phân Tích Cấu Trúc. 1](#_Toc199160678)

## Phân Tích Giao Diện Người Dùng.

Phần này trình bày cấu trúc và hành vi chức năng của giao diện người dùng (UI) của mô phỏng D.S.A.S. Thiết kế hướng đến sự rõ ràng, hỗ trợ tương tác và kiểm tra hợp lệ theo thời gian thực (real-time validation) nhằm giúp người dùng mô phỏng thuật toán lập lịch đĩa hiệu quả.

1. **Giao diện ban đầu (Initial View)**

Khi trang được tải, giao diện hiển thị một khối trung tâm với tiêu đề mô phỏng:

* **Main Title**: D.S.A.S
* **Subtitle**: Disk Scheduling Algorithms Simulator
* **Instructional Message**:  
  Một đoạn thông báo hướng dẫn người dùng nhập các thông tin cần thiết:  
  *"Please: prepare below fields for simulation!"*

1. **Input Section**

Giao diện gồm ba ô nhập liệu chính:

| **Field Label** | **Placeholder** | **Chức năng** |
| --- | --- | --- |
| Disk's Track Capacity | "Disk's track capacity" | Xác định tổng số track của đĩa. |
| Head Position | "Head position" | Vị trí ban đầu của đầu đọc đĩa. |
| Request Queue | "Request Queue" | Danh sách các yêu cầu truy xuất track. |

**Hành vi của vùng nhập liệu như sau**:

* **Real-Time Validation**:  
  Nếu người dùng để trống hoặc nhập dữ liệu không hợp lệ, thông báo cảnh báo sẽ hiển thị ngay bên dưới ô tương ứng (ví dụ: *"capacity warning message"*).
* **Visual Feedback**:  
  Các input không hợp lệ sẽ được áp dụng class CSS (invalid), có thể làm đổi màu viền hoặc nền của ô để cảnh báo lỗi.
* **Queue Assistance**:  
  Nếu Request Queue không bao gồm Head Position, hệ thống sẽ tự động thêm giá trị của Head vào queue để đảm bảo tính đầy đủ cho mô phỏng.

1. **Algorithm Selector (Radio Group)**

Người dùng có thể chọn một trong sáu thuật toán lập lịch bằng các nút radio. Mỗi lựa chọn được gắn nhãn rõ ràng như thẻ:

| **Radio ID** | **Label** | **Giải thích** |
| --- | --- | --- |
| FCFS | [FCFS] | First-Come, First-Served |
| SSTF | [SSTF] | Shortest Seek Time First |
| ScanU | [ScanU] | SCAN (hướng lên) |
| ScanD | [ScanD] | SCAN (hướng xuống) |
| CScanU | [CScanU] | Circular SCAN (lên) |
| CScanD | [CScanD] | Circular SCAN (xuống) |

* **Hành vi**:
  + Chỉ một lựa chọn được bật tại một thời điểm.
  + Khi được chọn, label tương ứng có thể đổi màu hoặc nổi bật hơn.
  + Một thuật toán mặc định (thường là FCFS) có thể đã được chọn sẵn khi giao diện tải.

1. **RUN Simulation Button**

Nút RUN Simulation nằm bên dưới phần chọn thuật toán, có chức năng khởi chạy mô phỏng.

* **Click Behavior**:
  + Gọi hàm runSimulation().
  + Kiểm tra lại tất cả ô nhập liệu.
  + Nếu còn lỗi, sẽ hiển thị các thông báo cảnh báo tương ứng.
  + Nếu hợp lệ, mô phỏng sẽ được thực hiện và hiển thị kết quả.

1. **Output Section – Simulation Results**

Khi mô phỏng thành công, giao diện hiển thị:

* **Seek Time Display**  
  Một đoạn văn với nội dung *"Seek Time Accumulated:"*, tiếp theo là tổng số bước di chuyển đầu đọc (seek).
* **Chart/Table Area** (#sim-table)
  + Khu vực này được tạo động (dynamic) sau khi mô phỏng.
  + Hiển thị chuỗi di chuyển của đầu đọc đĩa theo thứ tự yêu cầu.
  + Có thể bao gồm nhãn track và hướng đi theo trình tự mô phỏng.

## Phân Tích Cấu Trúc Script.

##### Page Setup & Interface Initialization

Các hàm và khối này được thiết kế để chạy ngay page initial load hoặc trong runtime để thiết lập giao diện đầu vào và hành vi:

1. **Interface Setup**
   * Xác nhận tham chiếu tập lệnh chính xác bằng */// <reference path="algorithm.js" /> và thử gọi hàm ngoại checkScriptLink();*
   * Nhận biến globle chỉ về các trường input.
   * Mọi initial html element được css với opacity = 1, Tiến hành dọn dẹp mọi warning message trên trang bằng các điều chỉnh biến opacity = 0 khi page load/run script.
   * Điều chỉnh Radio box “FCFS” làm option mặc định.
2. **Event Listener Setup**
   * Tập trung con trỏ vào trường đầu tiên (Track Capacity) khi trang được tải.
   * Thực hiện update warning message liên tục cho từng input được thực hiện trong các trường.
   * Thực hiện validate input liên tục, xóa mọi ký tự khác ký số và khác kí tự chia cách được nhập.
3. **Autostretch Input Setup**
   * Điều chỉnh động độ rộng của trường yêu cầu I/O để phù hợp với nội dung văn bản bằng cách sử dụng phần tử "mirorer" ẩn, với mọi input trường I/O nhận, mirorer copy và điều chỉnh độ rộng trường để cải thiện khả năng chứa chuỗi dài và hiển thị thông tin input tối đa cho người nhập.

##### Simulation Workflow

Đây là các hàm *cốt lõi* xử lý dữ liệu đầu vào và mô phỏng kết quả lập lịch đĩa thông qua việc inject html element linh hoạt.

1. **Main Entry Point**

Hàm runSimulation(), được link với html button “Run Simulation” thực hiện một chuỗi nhiệm vụ sử dụng các hàm khác nhau để tiến hành quá trình mô phỏng.

* Validate inputs.
* Đọc I/O requests list bằng hàm numberMapping() trả về array dạng ký số
* Sử dụng hàm *touchUp(), removeDuplicate() và sort()* để tạo queue với cấu trúc đầy đủ cho header của bảng mô phỏng.
* Chuẩn bị queue kết quả simulatedQueue bằng

1. **Algorithym Selection**

Hàm *algorithmDriver(queue),* được link với html button “Run Simulation” thực hiện một chuỗi nhiệm vụ sử dụng các hàm khác nhau để tiến hành quá trình mô phỏng.

* Switchcase Đọc Radio boxes để chọn giải thuật
* Nhận request queue từ form và trả về queue mới đã được xử lý

1. **Input & Queue Handling**

Nhóm hàm xử lý các chuỗi input:

*numberMapping(inputStr), sanitizeToNumberArray(arr)*

* Dọn dẹp ký tự thừa trong “messy string”

*touchUp(queue) + addHead() / addMin() / addMax() + removeDuplicates(queue)*

* Đảm bảo queue trả về có
  + Current Head
  + Min (0)
  + Max (Disk Capacity)
* Deduplicates queue
* Sanitize phần tử

1. **Table Construction**  (Visualization)

Nhóm hàm thực hiện DOM manipulation để khởi tạo simulation visual

*constructSimTable(headQueue, simulatedQueue, simTable) + addRow() / addCell() + clearSimTable()*

* Xây dựng hình ảnh trực quan 2D về hoạt động của đĩa
  + Top row = danh sách được sắp xếp của tất cả các vị trí yêu cầu *(tiêu đề)*
  + Each row = một bước hoạt động *(với live Cell hoặc chỉ báo lỗi)*

1. **SVG Drawing (Visualization)**

Sử dụng Scalable Vector Graphics (SVG) để tạo vector kết nối cho bảng hiển thị

* Hàm *injectSVG() Tạo một lớp svg bao phủ toàn bộ trang web*
* Hàm *connectCell(svg)* kết nối live Cells với các dòng SVG, cho phép theo dõi trực quan quá trình tìm kiếm và xử lý của đĩa.

1. **Utility Getters**

DOM access với *getCapacity() / getHead() / getQueue() / getCapacityMsg() / getHeadMsg() / getQueueMsg()*

*END*