

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



BÀI TẬP LỚN
Lập trình nâng cao (CO2029)

Xây dựng ứng dụng tra cứu dữ liệu sao kê
của Mặt trận Tổ quốc Việt Nam

Giảng viên hướng dẫn:	Lê Đình Thuận	
SV thực hiện:	Đặng Nguyên Bảo	2310211
	Bùi Trần Duy Khang	1234567
	Nguyễn Tuấn Cường	1234567
	Trần Đức Tiến	1234567
	Đào Hữu Gia Huy	1234567
	Nguyễn Như Xuân	1234567
	Nguyễn Ngô Hoàng Long	1234567

Thành phố Hồ Chí Minh, Tháng 12 năm 2024

Mục lục

1	Tổng quan đề tài	2
2	Phân tích và thiết kế hệ thống	3
2.1	Yêu cầu chức năng và phi chức năng	3
2.1.1	Yêu cầu chức năng	3
2.1.2	Yêu cầu phi chức năng	3
2.2	Kiến trúc hệ thống	3
2.3	Luồng hoạt động	4
2.4	Class Diagram cho hệ thống	5
2.5	Công cụ sử dụng	5
3	Mô tả giải pháp	6
3.1	Cấu trúc dữ liệu được sử dụng	6
3.2	Thuật toán tìm kiếm được áp dụng	6
3.3	Quy trình xử lý truy vấn:	7
3.4	Ưu điểm:	7
3.5	Nhược điểm:	7
4	Cấu trúc mã nguồn và hiện thực thuật toán	8
4.1	Tổng quan	8
4.2	Backend	9
4.3	Front-end	11
5	Kiểm thử và đánh giá hệ thống	13
5.1	Khởi động chương trình	13
5.2	Kiểm thử	13
5.3	Kết luận	16
6	Tài liệu tham khảo	18

1 Tổng quan đề tài

Ngày nay, việc minh bạch thông tin tài chính đóng vai trò quan trọng trong xây dựng lòng tin và sự tin cậy từ cộng đồng. Đặc biệt đối với các tổ chức như Mặt trận Tổ quốc Việt Nam (MTTQ VN), việc công khai và dễ dàng tra cứu các giao dịch tài chính giúp tăng cường tính minh bạch, đồng thời tạo điều kiện thuận lợi cho việc giám sát và kiểm tra.

Dự án "**Xây dựng ứng dụng tra cứu dữ liệu sao kê**" được thực hiện với mục tiêu xây dựng một hệ thống giúp tra cứu thông tin sao kê từ file dữ liệu công khai dưới định dạng CSV. Người dùng có thể dễ dàng tìm kiếm thông tin theo các tiêu chí như số tiền, tên người gửi, nội dung chuyển khoản hoặc thời gian giao dịch.

Ngoài việc cung cấp giao diện thân thiện cho người dùng, đề tài còn tập trung vào việc áp dụng các cấu trúc dữ liệu và thuật toán tối ưu như Boyer-Moore và Binary Search để đảm bảo hiệu năng cao ngay cả với tập dữ liệu lớn.

2 Phân tích và thiết kế hệ thống

2.1 Yêu cầu chức năng và phi chức năng

2.1.1 Yêu cầu chức năng

1. Tra cứu dữ liệu từ file CSV cố định:
 - File dữ liệu sao kê (`chuyen_khoan.csv`) được quy định sẵn và xử lý trực tiếp trong mã nguồn của back-end.
 - Hệ thống không có chức năng tải file CSV mới từ người dùng.
2. Hỗ trợ các tiêu chí tra cứu:
 - **Số tiền (credit hoặc debit):** Tìm các giao dịch có giá trị được nhập bởi người dùng.
 - **Thông tin chuyển khoản (detail):** Tìm các giao dịch có chứa từ khóa trong trường thông tin chi tiết (detail).
 - **Mã giao dịch (trans_no):** Tìm kiếm theo mã giao dịch
 - **Ngày giờ giao dịch (data_time):** Lọc giao dịch dựa trên khoảng thời gian. Có thể kết hợp tìm kiếm theo thời gian giao dịch kết hợp với các thông tin khác.
3. Hiển thị kết quả:
 - Kết quả tra cứu được hiển thị dưới dạng bảng.
 - Các trường được hiển thị: ngày giờ giao dịch, mã giao dịch, số tiền, và nội dung giao dịch.
4. Tìm kiếm hiệu quả với tập dữ liệu lớn:
 - Hỗ trợ tìm kiếm chính xác và nhanh chóng thông qua các thuật toán tối ưu như Binary Search và Boyer-Moore.

2.1.2 Yêu cầu phi chức năng

1. Hiệu năng cao:
 - Tốc độ xử lý nhanh ngay cả khi tập dữ liệu lớn.
 - Tiết kiệm bộ nhớ thông qua các thuật toán và cấu trúc dữ liệu hợp lý.
2. Giao diện thân thiện:
 - Hệ thống hỗ trợ giao diện web đơn giản, trực quan để nhập tiêu chí tìm kiếm và hiển thị kết quả.
 - Giao diện có tính tương tác cao, giữ được tỉ lệ trên nhiều môi giao diện Web khác nhau.
3. Khả năng mở rộng:
 - Dễ dàng tích hợp thêm các tiêu chí tra cứu mới trong tương lai.
 - Cấu trúc mã nguồn linh hoạt, dễ bảo trì.

2.2 Kiến trúc hệ thống

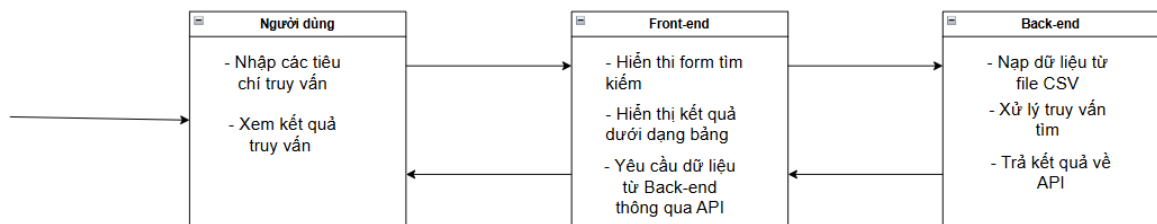
Tổng quan kiến trúc: Hệ thống được xây dựng với kiến trúc client-server gồm 2 thành phần chính:

1. Back-end:
 - Xử lý file `chuyen_khoan.csv` và nạp dữ liệu vào các cấu trúc dữ liệu được tối ưu cho tìm kiếm.
 - Triển khai các API phục vụ tìm kiếm dựa trên tiêu chí do người dùng nhập.
 - Sử dụng Node.js với framework Express để xử lý yêu cầu và trả kết quả cho front-end.

2. Front-end:

- Giao diện web hiển thị form tìm kiếm và kết quả.
- Giao diện tối giản nhưng trực quan, hỗ trợ người dùng nhập các tiêu chí như khoảng số tiền, tên người gửi, nội dung, hoặc ngày giờ giao dịch.
- Sử dụng React và CSS/Java Scripts để hiện thực.

Sơ đồ kiến trúc:



Hình 1: Sơ đồ kiến trúc hệ thống

Vì dữ liệu được nạp và sử dụng trực tiếp tại server backend nên Server (trong mô hình Client-Server) trong trường hợp này sẽ là Backend server, với Client là user, yêu cầu và nhận phản hồi với Server thông qua Frontend server và các API trung gian.

2.3 Luồng hoạt động

1. Nạp dữ liệu từ file CSV (khi khởi động server):

File CSV được xử lý theo quy trình sau:

- Xóa BOM marker (Byte order marker) nếu tồn tại trong file CSV.
- Tách dữ liệu theo từng dòng và bỏ qua dòng tiêu đề.
- Lưu trữ từng giao dịch dưới dạng object JSON, ví dụ như sau:

```

{
  "data_time" : "05/09/2024_5218.29906",
  "trans_no" : 74,
  "credit" : "6000",
  "debit" : 0,
  "detail" : "120232.050924.144826.NGUYEN HUYEN MY chuyen tien, ma GD 247619865"
}
  
```

Hình 2: Một Object JSON được trích xuất từ giao dịch có mã GD 74

- Các giao dịch này được lưu vào các cấu trúc dữ liệu như mảng, map object để hỗ trợ tìm kiếm hiệu quả.

2. Người dùng gửi yêu cầu tìm kiếm:

- Người dùng nhập các tiêu chí tìm kiếm thông qua giao diện web.

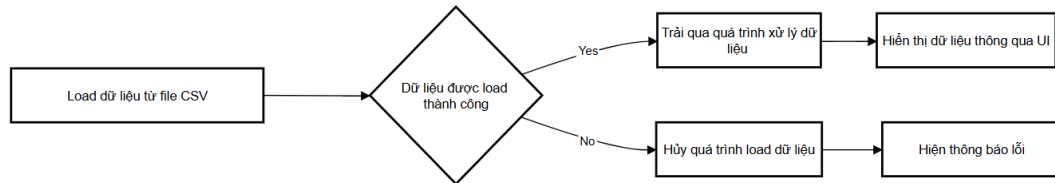
3. Xử lý truy vấn tìm kiếm:

- Hệ thống sử dụng cấu trúc dữ liệu và thuật toán phù hợp (Boyer-Moore, Binary Search) để tìm kiếm dữ liệu thỏa mãn yêu cầu.

4. Trả kết quả:

- Kết quả được trả về front-end dưới dạng JSON và hiển thị thành bảng cho người dùng.

Có thể tóm tắt ngắn gọn luồng đi của dữ liệu trong hệ thống thông qua sơ đồ hoạt động sau:



Hình 3: Tóm tắt sơ đồ hoạt động

2.4 Class Diagram cho hệ thống

2.5 Công cụ sử dụng

Ngôn ngữ và framework:

- Back-end: Node.js, Express.
- Front-end: HTML/CSS/JavaScript (Kết hợp React để tăng tính tương tác trên giao diện người dùng).

Thư viện và công cụ:

- Cors: Để xử lý chia sẻ tài nguyên giữa các nguồn khác nhau.
- fs (Node.js): Để xử lý file CSV và đọc dữ liệu từ hệ thống file.

Công cụ hỗ trợ:

- Trình duyệt web: Chrome, Firefox để kiểm tra giao diện.
- Postman: Kiểm thử API tìm kiếm.

Công cụ kiểm tra hiệu năng: K6 hoặc Vegeta để kiểm tra khả năng chịu tải của hệ thống.

3 Mô tả giải pháp

3.1 Cấu trúc dữ liệu được sử dụng

1. Mảng động (Array):

- Sử dụng để lưu trữ dữ liệu đọc được từ file CSV dưới dạng danh sách các object JSON.
- Mỗi object JSON chứa các trường: `data_time`, `trans_no`, `credit`, `debit` và `detail`.
- Mảng động là nền tảng để tổ chức và xử lý dữ liệu, cho phép duyệt tuần tự hoặc sắp xếp nhanh chóng.

2. Map Object Data (Map Object): Được sử dụng để tạo các bảng tra cứu nhanh theo trường dữ liệu:

- **ByDate:** Map với key là giá trị ngày giờ (`data_time`), value là danh sách các giao dịch tương ứng.
- **ByTransNo:** Map với key là số giao dịch (`trans_no`), value là danh sách các giao dịch tương ứng.
- **ByCredit:** Map với key là giá trị (`credit`), value là danh sách các giao dịch tương ứng.
- **ByDebit:** Map với key là giá trị (`debit`), value là danh sách các giao dịch tương ứng. Ngoài ra, các key trong Map Object được tổ chức và sắp xếp để hỗ trợ tìm kiếm nhị phân (Binary Search).

3. Map Object Keys (Map Object):

- Lưu danh sách key duy nhất cho từng trường, ví dụ: tất cả các giá trị `credit` không trùng lặp.
- Mỗi danh sách được sắp xếp theo thứ tự tăng dần, hỗ trợ tìm kiếm nhanh bằng thuật toán Binary Search.

3.2 Thuật toán tìm kiếm được áp dụng

1. Thuật toán tìm kiếm Boyer-Moore:

Mô tả: Boyer-Moore là một thuật toán tìm kiếm chuỗi con (substring) trong chuỗi lớn (string). Đây là một trong những thuật toán nhanh nhất trong trường hợp trung bình vì nó không kiểm tra từng ký tự mà tận dụng thông tin bổ sung từ bảng dịch (shift table).

Cơ chế hoạt động: Thuật toán hoạt động dựa trên hai chiến lược chính:

- Bad Character Heuristic (Bảng ký tự xấu):
 - Khi ký tự trong chuỗi cần tìm (pattern) không khớp với chuỗi lớn (text), thuật toán dịch chuỗi cần tìm sao cho ký tự không khớp trong chuỗi lớn thẳng hàng với lần xuất hiện cuối cùng của nó trong chuỗi cần tìm.
 - Nếu ký tự đó không tồn tại trong chuỗi cần tìm, thuật toán dịch chuỗi cần tìm qua toàn bộ độ dài của nó.
- Good Suffix Heuristic (Bảng tiền tố tốt):
 - Nếu một đoạn cuối cùng (suffix) của chuỗi cần tìm khớp, nhưng phần còn lại không khớp, thuật toán dịch sao cho phần đoạn này thẳng hàng với lần xuất hiện khác trong chuỗi cần tìm.
 - Nếu đoạn đó không xuất hiện nữa, thuật toán dịch qua toàn bộ độ dài của nó.

Ứng dụng: Dùng để tìm kiếm chuỗi con (substring) trong trường "detail".

- Tạo bảng bad character để giảm số bước dịch chuyển trong quá trình tìm kiếm.
- Tìm kiếm hiệu quả trên chuỗi dài nhờ giảm thiểu số lần so khớp ký tự.

- Tối ưu cho trường hợp tìm chuỗi trên tập dữ liệu lớn, với độ phức tạp trung bình: $O(m+n)$ (m là độ dài chuỗi cần tìm, n là độ dài chuỗi lớn).

2. Thuật toán Binary Search:

Ứng dụng: Tìm kiếm nhanh theo các trường credit, debit và data_time và lọc giao dịch nằm trong khoảng giá trị được nhập bởi người dùng (ví dụ: credit từ 1000 đến 1000000).

Cách hoạt động: Tìm ra được khoảng trong mảng lấy từ Map Object Keys với trường tương ứng. Các giá trị trong khoảng này sẽ thỏa với yêu cầu tìm kiếm. Với những giá trị đó ta sẽ lấy được những item tương ứng từ Map Object Data.

Hiệu quả: Giảm độ phức tạp trong việc tìm kiếm khoảng trong mảng lấy từ Map Object Keys từ $O(n)$ (tìm tuyến tính) xuống $O(\log n)$ nhờ sắp xếp trước. Sau đó lấy các item trong khoảng đó tốn $O(\text{số item thỏa yêu cầu})$.

3. Thuật toán sắp xếp (Tim Sort/Merge Sort):

Ứng dụng: Tìm kiếm nhanh theo các trường credit, debit và data_time và lọc giao dịch nằm trong khoảng giá trị được nhập bởi người dùng (ví dụ: credit từ 1000 đến 1000000).

Cách hoạt động: Sử dụng hàm so sánh để sắp xếp mảng theo thứ tự tăng dần.

Hiệu quả: Đảm bảo dữ liệu sẵn sàng cho tìm kiếm nhị phân, đảm bảo độ phức tạp luôn xấp xỉ mức $O(n \log n)$.

3.3 Quy trình xử lý truy vấn:

1. Nhận đầu vào từ người dùng: Người dùng nhập tiêu chí tìm kiếm trên giao diện (số tiền, mã giao dịch hoặc nội dung) hoặc chọn khung thời gian tìm kiếm trên chức năng lịch.
2. Lựa chọn thuật toán phù hợp:
 - Với trường detail: Áp dụng Boyer-Moore để tìm kiếm với mỗi mẫu, nếu kết quả của mẫu đó trả về khác -1 , thêm nó vào kết quả.
 - Với các trường credit, debit, data_time: Sử dụng Binary Search để tìm nhanh trong Map Object Keys. Lấy kết quả từ Map Object Data bằng cách thêm vào kết quả các giao dịch tương ứng với key trong khoảng tìm kiếm.
3. Trả kết quả: Kết quả được chuyển đổi thành JSON và trả về cho front-end để hiển thị.

3.4 Ưu điểm:

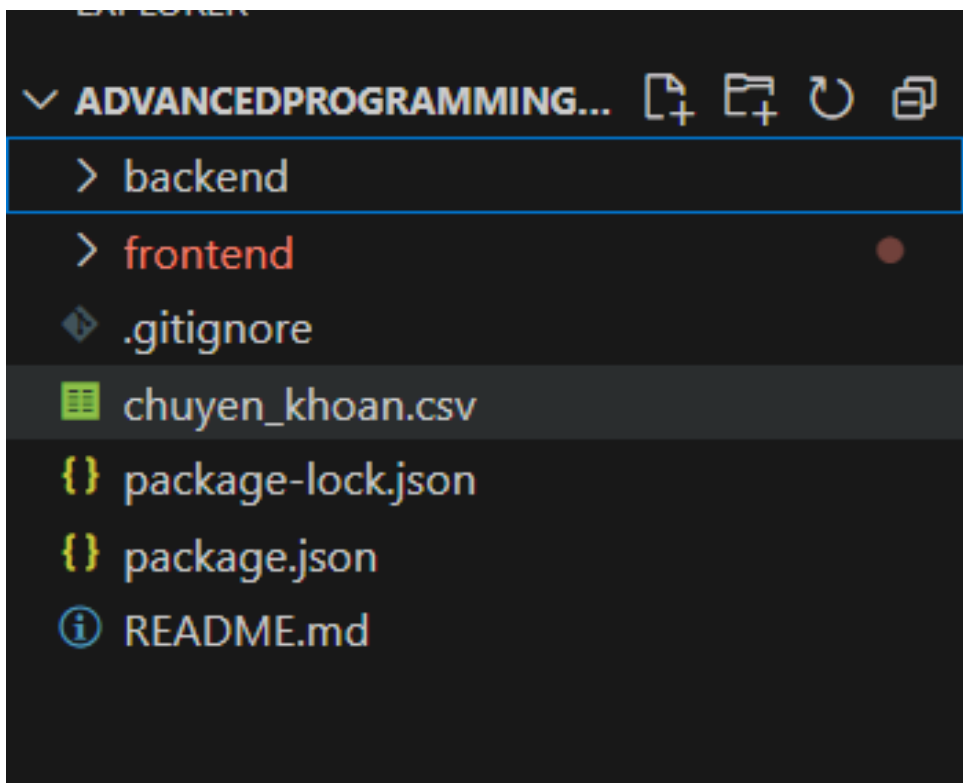
1. Hiệu năng cao: Kết hợp giữa các cấu trúc dữ liệu Map Object và thuật toán tối ưu giúp giảm thời gian tìm kiếm trên tập dữ liệu lớn.
2. Dễ mở rộng: Có thể bổ sung thêm các tiêu chí tra cứu mới mà không cần thay đổi cấu trúc tổng thể.
3. Tiết kiệm bộ nhớ: Tận dụng tối đa các cấu trúc dữ liệu sẵn có trong JavaScript, như mảng động và Map Object.

3.5 Nhược điểm:

1. Thuật toán xử lý file ban đầu có thể tiêu tốn tài nguyên khi nạp dữ liệu lớn.
2. Không hỗ trợ xử lý dữ liệu theo thời gian thực (do file CSV cố định).

4 Cấu trúc mã nguồn và hiện thực thuật toán

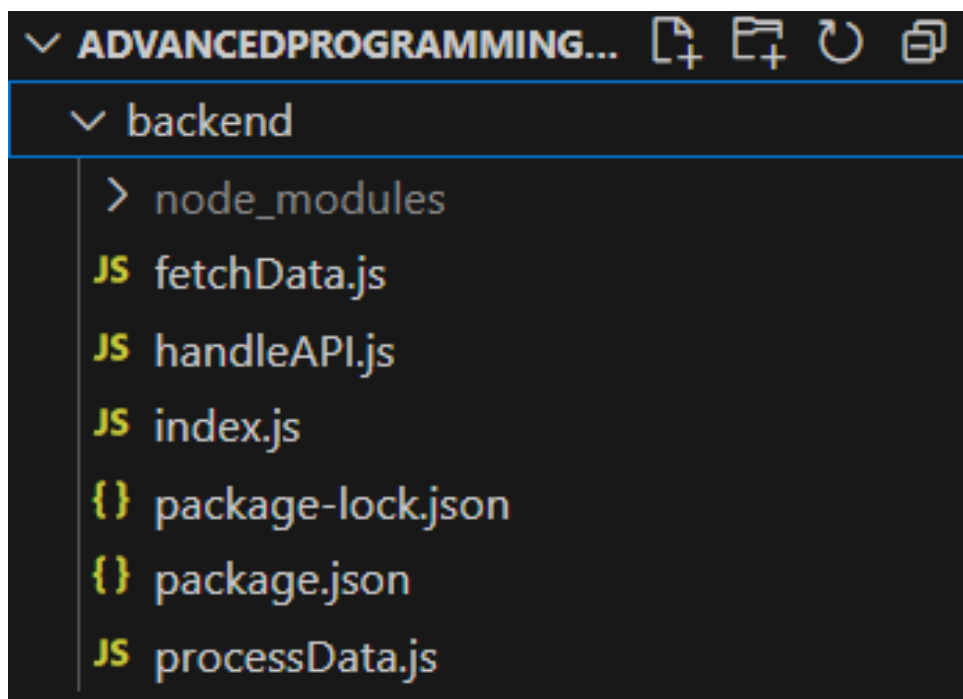
4.1 Tổng quan



Hình 4: Tổng quan mã nguồn hệ thống

Mã nguồn gồm 2 thư mục backend và frontend riêng biệt. File CSV cần truy vấn được đặt cùng một vị trí cho sự tiện lợi khi truy cập. Khi khởi tạo server cho backend hoặc frontend, chỉ cần gọi tới thư mục nói trên tại terminal.

4.2 Backend



Hình 5: Tổng quan thư mục Backend

Thư mục gồm các thành phần sau:

1. `fetchData.js`: Xử lý việc tải dữ liệu từ file CSV. File sẽ tạo một datastream bắt đầu từ dòng thứ 2 của file (để loại bỏ header). Sau đó xóa BOM (Byte Order Mark) nếu có tồn tại. BOM là một marker vô hình được đặt vào đầu file để phân biệt định dạng UTF-8 với các định dạng khác. Sau đó lấy từng dòng của dữ liệu và nhận theo các trường định dạng được quy định nói trên.
2. `handleAPI.js`: Quy định việc xử lý các yêu cầu API từ frontend.
3. `index.js`: File khởi tạo server Backend.
4. `processData.js`: Gồm 2 class chính là BoyerMoore và DataHandler, quy định việc hiện thực thuật toán BoyerMoore và xử lý data query theo các bước dưới đây:
 - Tạo ra một map object với 4 trường ByDate, ByTransNo, ByCredit, ByDebit
 - Tại mỗi trường (gọi là trường A), ta tạo một map object khác, với key là giá trị của trường tương ứng và value là một mảng chứa item mà tại trường ứng với trường A, giá trị của nó bằng với key.

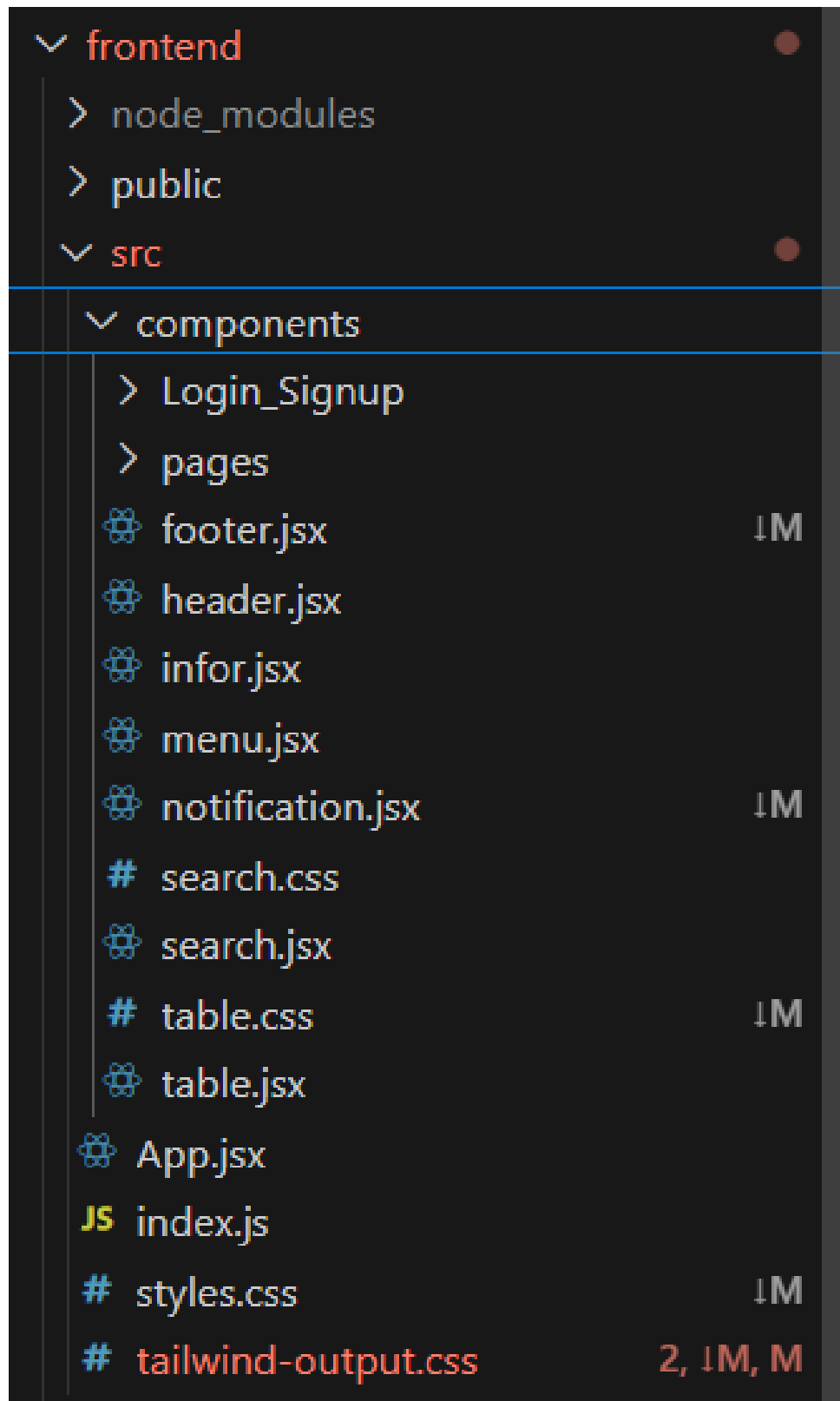
```
inputData.forEach((item) => {  
  this.data["byDate"][item.data_time] =  
    this.data["byDate"][item.data_time] ?? [];  
  this.data["byDate"][item.data_time].push(item);  
  
  this.data["byTransNo"][item.trans_no] =  
    this.data["byTransNo"][item.trans_no] ?? [];  
  this.data["byTransNo"][item.trans_no].push(item);  
  
  this.data["byCredit"][item.credit] =  
    this.data["byCredit"][item.credit] ?? [];  
  this.data["byCredit"][item.credit].push(item);  
  
  this.data["byDebit"][item.debit] =      this.data["byDebit"][item.debit] ?? [];  
  this.data["byDebit"][item.debit].push(item);  
});
```

Hình 6: Hiện thực Map Object

- Tạo thêm một map object (đặt tên là keys), cũng gồm 4 trường ByDate, ByTransNo, ByCredit, ByDebit.với value là một mảng
- Thêm vào mảng các giá trị tương ứng với các trường và sort theo comparator được định nghĩa sẵn (comparator so sánh số nguyên, so sánh ngày tháng), lưu ý mỗi giá trị chỉ tồn tại 1 lần trong mảng (không được lặp lại).

Với cách làm trên, chúng ta sẽ hỗ trợ cho việc tìm kiếm bằng Binary Search. Khi đó, ta chỉ cần thực hiện BinarySearch theo map object keys. Và lấy item trong map object khác được định nghĩa ở trên. Ngoài ra, Với trường details ta sẽ nạp vào cấu trúc dữ liệu Boyer Moore. Xây dựng một bảng và tìm kiếm trên bảng đó. Nó sẽ trả về Vị trí của item trong file .csv và đồng thời những vị trí bắt đầu của key mà người dùng tìm kiếm trong trường detail của item đó.

4.3 Front-end



Hình 7: Tổng quan thư mục Front-end

Thư mục gồm các thành phần sau:

1. Các Component: Các component như footer.jsx, header.jsx, menu.jsx... đóng vai trò như các khối đồ họa được tái sử dụng trong quá trình xây dựng UI.
2. Các file trong thư mục pages: Gồm các file như Contact.js, About.js, chứa thông tin cấu trúc cho các trang hiển thị khác nhau trong hệ thống.
3. index.js: Quy định cách render UI của hệ thống khi server được khởi tạo
4. App.jsx: Quy định dữ liệu / hàm và xử lý các yêu cầu từ người dùng, qua đó gọi API tới backend và yêu cầu dữ liệu cần thiết.
5. Các file css: Quy định một định dạng chung về mặt đồ họa và cấu trúc được tái sử dụng cho các component và trang.

Với cấu trúc như trên, server Frontend của hệ thống có luồng hoạt động cơ bản như sau:

1. **Khởi tạo và điều hướng:** App.jsx quản lý routing với các trang home, about, và contact.
2. **Tìm kiếm dữ liệu:** Người dùng nhập thông tin tìm kiếm tại Search Form, dữ liệu từ đó được gửi tới server API, kết quả sẽ được trả về và hiển thị trong Statement Table.
3. **Hiển thị thông tin:** Số dư, tổng thu nhập và chi phí được hiển thị ở AccountInformation.
4. **Thông báo lỗi:** Mọi lỗi hoặc thông báo thành công hiển thị bằng Notification.

5 Kiểm thử và đánh giá hệ thống

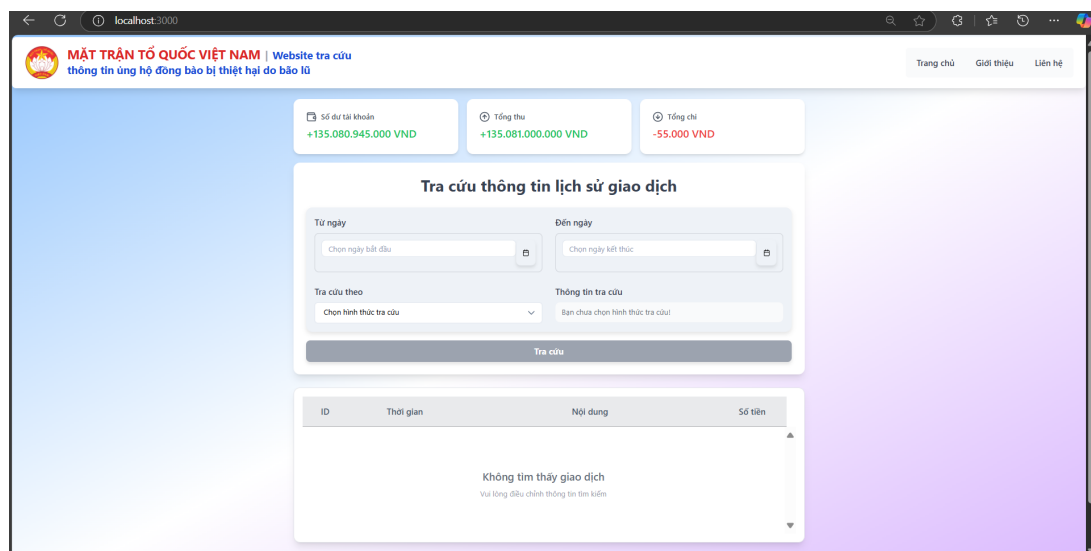
5.1 Khởi động chương trình

Để chạy chương trình trên máy tính cá nhân cần làm như sau:

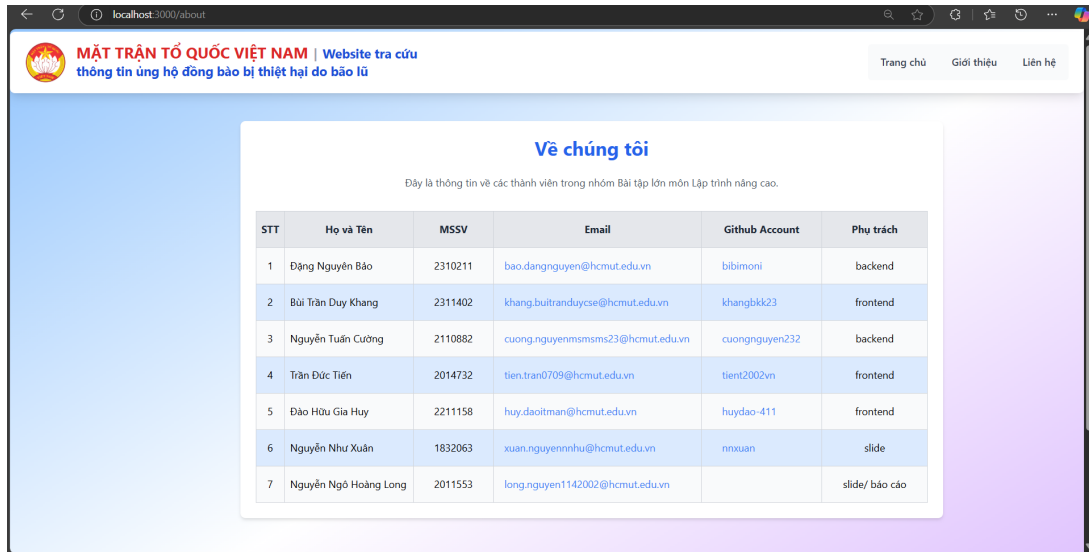
1. Đi đến [Website chính thức của NodeJS](#) và cài đặt NodeJS về máy
2. Mở Terminal và cd vào đường dẫn của project (thư mục `AdvancedProgramming24-Assignment`)
3. Tiếp tục gõ `cd backend\` và gõ `npm i` (nếu đây là lần chạy code đầu tiên), sau đó gõ `node ..`
4. Mở một Terminal mới và cũng cd vào đường dẫn của project.
5. Tiếp tục gõ `cd frontend\`, sau đó gõ `npm start` để chạy webapp. Lúc này terminal sẽ tự mở browser và đi đến <http://localhost:3000/>, nếu không hãy tự gõ đường dẫn vào thanh tìm kiếm của browser.

5.2 Kiểm thử

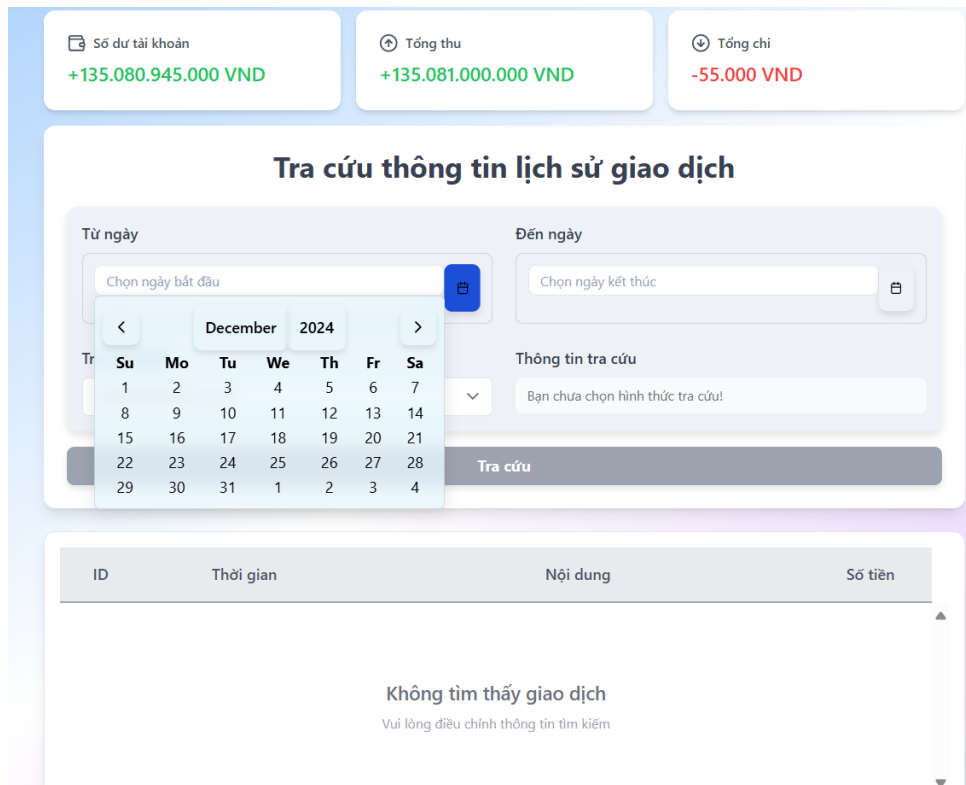
Khi khởi chạy server backend tại local port 8080 và server frontend tại local port 3000, hệ thống hoạt động bình thường và sẵn sàng cho việc kiểm thử.



Hình 8: Giao diện trang chủ của hệ thống



Hình 9: Giao diện giới thiệu của hệ thống



Hình 10: Hệ thống cho phép truy vấn theo thời gian

Tra cứu thông tin lịch sử giao dịch

Từ ngày

Đến ngày

Tra cứu theo

Tìm kiếm...

- Mã giao dịch
- Số tiền thu
- Số tiền chi
- Nội dung chuyển khoản

Thông tin tra cứu

Bạn chưa chọn hình thức tra cứu!

Nội dung	Số tiền
<p>Không tìm thấy giao dịch</p> <p>Vui lòng điều chỉnh thông tin tìm kiếm</p>	

Hình 11: Hệ thống cho phép truy vấn theo 4 tiêu chí (Mã GD, số tiền thu, số tiền chi, Nội dung chuyển khoản)

Tra cứu thông tin lịch sử giao dịch

Số dư tài khoản

+135.080.945.000 VND

Tổng thu

+135.081.000.000 VND

Tổng chi

-55.000 VND

Từ ngày

Đến ngày

Tra cứu theo

Thông tin tra cứu

Tra cứu

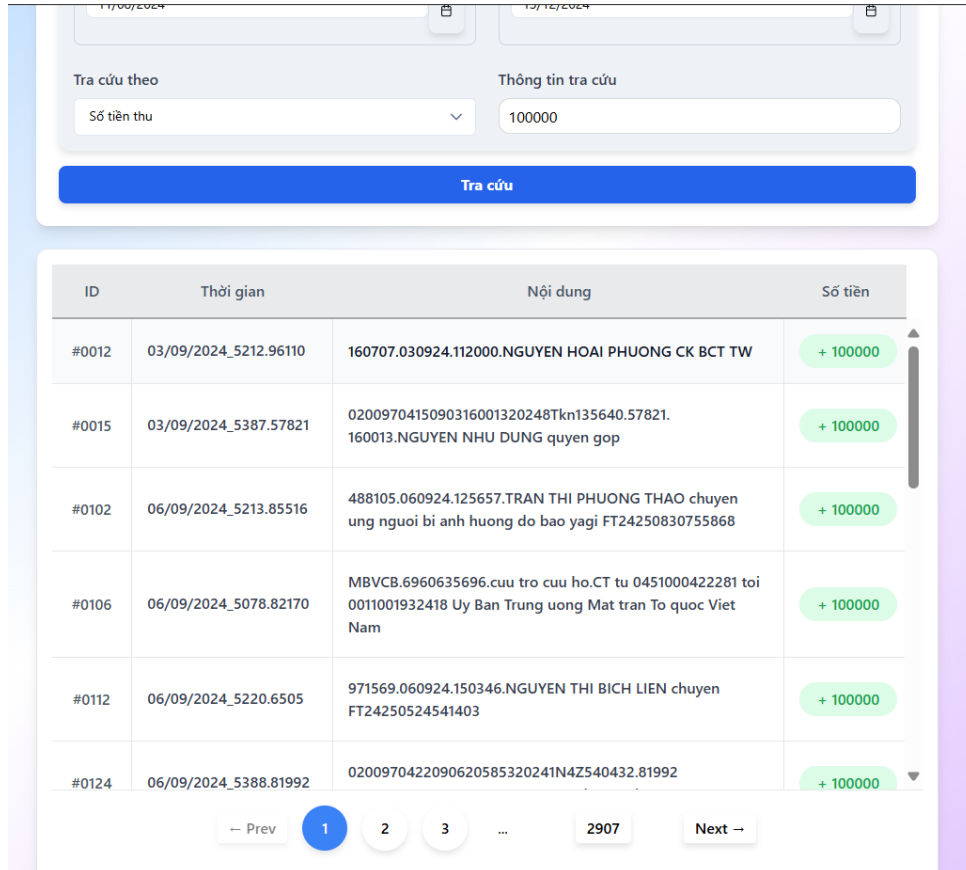
ID	Thời gian	Nội dung	Số tiền
#0114	06/09/2024_5219.86168	975783.060924.154915.LE THI MINH NGOC Chuyen tien	+ 6000

Hình 12: Kết quả khi truy vấn kết hợp giữa thời gian và mã GD 114

Kết quả này đúng với dữ liệu gốc trong file CSV gốc.


```
113 "06/09/2024_5220.6505",112,"100000",0,"971569.060924.150346.NGUYEN THI BICH LIEN chuyen FT24250524541403"  
114 "06/09/2024_5240.62694",113,"20000",0,"MBVCB.6961243776.PHAM XUAN DU chuyen tien.CT tu 0351001148841 PHAM XUAN DU toi 0011001932418  
115 "06/09/2024_5219.86168",114,"6000",0,"975783.060924.154915.LE THI MINH NGOC Chuyen tien"  
116 "06/09/2024_5213.83934",115,"6000",0,"2006577.060924_161258.TRAN VAN TOAN chuyen tien"
```

Hình 13: Đối chiếu kết quả truy vấn với mã GD 114



The screenshot shows a web application with a search form and a table of results. The search form has a dropdown menu for 'Tra cứu theo' (Search by) with 'Số tiền thu' (Revenue) selected, and a text input for 'Thông tin tra cứu' (Search information) with '100000' entered. A blue 'Tra cứu' (Search) button is below the form. The table below has four columns: ID, Thời gian (Time), Nội dung (Content), and Số tiền (Amount). It displays several rows of transaction data, including IDs like #0012, #0015, #0102, #0106, #0112, and #0124, with corresponding times, descriptions, and amounts of 100,000. A pagination bar at the bottom shows 'Prev', '1', '2', '3', '...', '2907', and 'Next'.

Hình 14: Kết quả truy vấn với tiêu chí số tiền thu 200000

Với truy vấn các giao dịch với số tiền thu tương ứng với 200000đ, ta có 2907 trang kết quả với 15 kết quả truy vấn mỗi trang với thời gian ít hơn 1 giây. Điều này cho thấy hệ thống cơ bản đã vận hành đúng với các yêu cầu đề ra.

[Link Source code của hệ thống](#)

5.3 Kết luận

Sau quá trình phân tích, thiết kế, xây dựng và kiểm thử, hệ thống tra cứu dữ liệu sao kê đã vận hành ổn định và đáp ứng đầy đủ các yêu cầu đặt ra. Cụ thể:

Tính năng hoàn thiện:

- Hệ thống hỗ trợ tìm kiếm chính xác và hiệu quả dựa trên các tiêu chí như số tiền, tên người gửi, nội dung giao dịch, và thời gian.
- Kết quả tìm kiếm được hiển thị rõ ràng dưới dạng bảng, trực quan và dễ sử dụng.

Tối ưu hóa:

- Việc sử dụng thuật toán Boyer-Moore để tìm kiếm theo chuỗi và thuật toán Binary Search cho các trường số liệu đã cải thiện đáng kể hiệu suất tra cứu, đảm bảo tốc độ xử lý nhanh ngay cả với dữ liệu lớn.
- Cấu trúc dữ liệu linh hoạt giúp tối ưu hóa bộ nhớ và giảm độ phức tạp xử lý.

Hệ thống ổn định:

- Qua các giai đoạn kiểm thử, hệ thống đã chứng minh được khả năng xử lý ổn định, không phát sinh lỗi nghiêm trọng.
- Các thành phần backend và giao diện người dùng giao tiếp mượt mà, đảm bảo trải nghiệm tốt cho người dùng cuối.

Khả năng mở rộng:

- Hệ thống được thiết kế với kiến trúc module hóa, cho phép mở rộng thêm tính năng hoặc tích hợp với các dịch vụ khác trong tương lai.

Với những kết quả đạt được, hệ thống không chỉ là một công cụ hữu ích để quản lý và tra cứu dữ liệu sao kê mà còn là một nền tảng có tiềm năng phát triển, ứng dụng trong nhiều bài toán quản lý dữ liệu tương tự. Kết quả này là minh chứng cho sự phối hợp hiệu quả giữa các thành viên trong nhóm, áp dụng tốt các kỹ thuật và công cụ lập trình hiện đại.



6 Tài liệu tham khảo

- [1] Boyer Moore Algorithm for Pattern Searching
<https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/>
- [2] Binary Search Algorithm – Iterative and Recursive Implementation
<https://www.geeksforgeeks.org/binary-search/>