**BÁO CÁO XÂY DỰNG HỆ THỐNG QUẢN LÝ TẬP TRUNG VOS3000**

**1. Đặt Vấn Đề**

Trong ngành viễn thông VoIP, VOS3000 là một giải pháp softswitch mạnh mẽ, được sử dụng rộng rãi. Tuy nhiên, việc quản lý đồng thời nhiều cụm máy chủ VOS3000 thường phức tạp, đòi hỏi nhiều thao tác thủ công trên từng server riêng lẻ. Điều này không chỉ tốn thời gian, nhân lực mà còn tiềm ẩn nguy cơ sai sót cao, ảnh hưởng đến hiệu quả vận hành và chất lượng dịch vụ. Các vấn đề thường gặp bao gồm:

* **Quản lý cấu hình phân tán:** Cấu hình Mapping Gateway (MG), Routing Gateway (RG), khách hàng, và các quy tắc định tuyến trên từng server một cách riêng lẻ rất dễ gây ra sự thiếu nhất quán và khó kiểm soát.
* **Thiếu giao diện quản lý tập trung:** Quản trị viên phải truy cập nhiều giao diện khác nhau (thường là CLI hoặc các tool quản lý riêng lẻ của VOS3000) để thực hiện các tác vụ, gây khó khăn trong việc theo dõi tổng thể và ra quyết định.
* **Khó khăn trong quản lý khách hàng và tài nguyên số:** Việc tạo mới, sửa đổi thông tin, cập nhật hạn mức tín dụng (limit money), khóa/mở tài khoản khách hàng, cũng như quản lý (thêm, xóa, tra cứu) các đầu số, số ảo trên nhiều server đòi hỏi thao tác lặp đi lặp lại, dễ nhầm lẫn và tốn nhiều công sức.
* **Phức tạp trong việc dọn dẹp và bảo trì:** Các tác vụ như dọn dẹp số không còn sử dụng khỏi các gateway, tra cứu thông tin số trên toàn hệ thống thường rất mất thời gian nếu thực hiện thủ công.

Để giải quyết những thách thức trên, việc xây dựng một **Hệ Thống Quản Lý Tập Trung VOS3000** với giao diện web thân thiện, trực quan là một nhu cầu cấp thiết. Hệ thống này được kỳ vọng sẽ đơn giản hóa các tác vụ quản trị, giảm thiểu lỗi vận hành, nâng cao hiệu quả sử dụng tài nguyên và cung cấp một cái nhìn tổng thể, minh bạch về hoạt động của toàn bộ cụm máy chủ VOS3000.

**2. Mục Tiêu và Phạm Vi Dự Án**

**2.1. Mục tiêu**

* Phát triển một ứng dụng web tập trung để quản lý các chức năng cốt lõi của nhiều server VOS3000.
* Đơn giản hóa các quy trình cấu hình Mapping Gateway, Routing Gateway và Rewrite Rules.
* Tối ưu hóa việc quản lý thông tin khách hàng, bao gồm hạn mức tín dụng và trạng thái tài khoản.
* Cung cấp công cụ tra cứu thông tin số và dọn dẹp gateway hiệu quả.
* Giảm thời gian thao tác và nguy cơ sai sót so với phương pháp quản lý thủ công.
* Nâng cao năng suất và hiệu quả làm việc của đội ngũ quản trị viên.

**2.2. Phạm vi**

Hệ thống tập trung vào các tính năng chính sau (dựa trên README.md và các module mã nguồn):

* **Cấu hình Server VOS3000:** Xem và cấu hình chi tiết Mapping Gateways (MG) và Routing Gateways (RG).
* **Quản lý Khách Hàng:** Tìm kiếm, xem chi tiết, cập nhật hạn mức, khóa/mở tài khoản khách hàng trên nhiều server.
* **Tra Cứu Thông Tin Số:** Tìm kiếm sự xuất hiện của một số điện thoại (và các biến thể) trong cấu hình MG/RG.
* **Dọn Dẹp Gateway:** Xóa các số điện thoại khỏi MG (CalloutCallerPrefixes) và RG (CallinCallerPrefixes, CallinCalleePrefixes, Rewrite Rules) trên nhiều server.
* **Quản lý Số Ảo (Rewrite Rules):** Tra cứu, sửa đổi, và thay thế số thật cho một Số ảo Đích trong RG.
* **Thêm Server VOS Mới (Trong Phiên):** Cho phép người dùng tạm thời thêm server VOS mới vào danh sách để thao tác trong phiên làm việc hiện tại (tuy nhiên, để lưu vĩnh viễn cần sửa config.py).

**3. Phân Tích và Thiết Kế Hệ Thống**

**3.1. Lựa chọn Công nghệ**

* **Ngôn ngữ lập trình:** Python (phiên bản 3.9+).
* **Framework giao diện Web:** Streamlit – Được chọn nhờ khả năng phát triển nhanh ứng dụng web tương tác trực tiếp từ mã Python, phù hợp cho các công cụ nội bộ và bảng điều khiển quản trị.
* **Thư viện hỗ trợ:**
  + streamlit-option-menu: Tạo menu điều hướng sidebar.
  + pandas: Xử lý và hiển thị dữ liệu dạng bảng.
  + requests: Thực hiện các lời gọi HTTP đến VOS3000 API.
* **Cơ sở dữ liệu:** Hệ thống không sử dụng cơ sở dữ liệu riêng để lưu trữ trạng thái cấu hình VOS3000 mà tương tác trực tiếp với các server VOS3000 thông qua API. Thông tin cấu hình server VOS được lưu trong tệp config.py.

**3.2. Kiến trúc Hệ thống**

Hệ thống được thiết kế theo kiến trúc module hóa, bao gồm các thành phần chính:

* app.py **(Giao diện người dùng - Frontend):**
  + Là file chính chạy ứng dụng Streamlit.
  + Xây dựng giao diện người dùng, bao gồm sidebar menu, các form nhập liệu, bảng hiển thị dữ liệu.
  + Điều phối luồng xử lý, nhận input từ người dùng và gọi các hàm xử lý từ backend.
  + Quản lý trạng thái phiên (st.session\_state) để lưu trữ dữ liệu tạm thời và lựa chọn của người dùng.
* **Các Module Backend (Thư mục** backup/ **hoặc cùng cấp với** app.py**):**
  + config.py**:**
    - Chứa các cấu hình cố định của hệ thống như danh sách VOS\_SERVERS (tên, URL API), DEFAULT\_TIMEOUT cho các lời gọi API, DEFAULT\_ENCODING.
    - Cung cấp các hàm tiện ích liên quan đến thông tin server.
  + api\_client.py**:**
    - Module trung tâm chịu trách nhiệm thực hiện tất cả các lời gọi đến VOS3000 API.
    - Đóng gói logic gửi request (HTTP POST), xử lý headers, payload, timeout.
    - Xử lý các response từ API, kiểm tra retCode (mã trả về của VOS API) và các lỗi HTTP, lỗi kết nối, lỗi giải mã JSON.
  + utils.py**:**
    - Chứa các hàm tiện ích chung được sử dụng bởi nhiều module khác.
    - Bao gồm: phân loại và biến đổi số điện thoại (classify\_phone\_number, transform\_real\_number\_for\_vos\_storage), tạo các biến thể tìm kiếm số (generate\_search\_variants), định dạng tiền tệ (format\_amount\_vietnamese\_style), phân tích và định dạng chuỗi rewrite rules (parse\_vos\_rewrite\_rules, format\_rewrite\_rules\_for\_vos), kiểm tra số ảo (is\_six\_digit\_virtual\_number\_candidate).
  + ui\_utils.py**:**
    - Chứa các hàm tiện ích liên quan đến giao diện người dùng, chủ yếu là safe\_display để hiển thị giá trị một cách an toàn. Các hàm khác liên quan đến CLI có thể là di sản.
  + customer\_management.py**:**
    - Chứa logic nghiệp vụ cho việc quản lý khách hàng.
    - Các chức năng: lấy chi tiết khách hàng (get\_raw\_customer\_details, get\_customer\_details\_for\_display), tìm kiếm khách hàng trên tất cả server (find\_customers\_across\_all\_servers), lấy và cập nhật hạn mức tín dụng (get\_current\_customer\_limit\_money, update\_customer\_credit\_limit), cập nhật trạng thái khóa/mở tài khoản (update\_customer\_lock\_status).
  + mapping\_gateway\_management.py**:**
    - Chứa logic nghiệp vụ cho việc quản lý Mapping Gateway (MG).
    - Các chức năng: lấy danh sách tất cả MG (get\_all\_mapping\_gateways), lấy chi tiết một MG (get\_mapping\_gateway\_details), cập nhật thông tin MG (update\_mapping\_gateway).
    - Hỗ trợ chức năng dọn dẹp MG: xác định MG cần dọn dẹp (identify\_mg\_for\_cleanup\_backend), áp dụng cập nhật sau khi dọn dẹp (apply\_mg\_update\_for\_cleanup\_backend).
  + routing\_gateway\_management.py**:**
    - Chứa logic nghiệp vụ cho việc quản lý Routing Gateway (RG) và số ảo.
    - Các chức năng: lấy danh sách tất cả RG (get\_all\_routing\_gateways), lấy chi tiết một RG (get\_routing\_gateway\_details), cập nhật thông tin RG (update\_routing\_gateway).
    - Quản lý số ảo: lấy tất cả định nghĩa số ảo (get\_all\_virtual\_number\_definitions\_backend), tìm định nghĩa của một số ảo cụ thể (find\_specific\_virtual\_number\_definitions\_backend), tìm kiếm key rewrite rule (find\_rewrite\_rule\_keys\_globally\_backend).
    - Hỗ trợ chức năng dọn dẹp RG: xác định RG cần dọn dẹp (identify\_rgs\_for\_cleanup\_backend), áp dụng cập nhật sau khi dọn dẹp (apply\_rg\_update\_for\_cleanup\_backend).
    - Tìm khách hàng liên kết với một số ảo thông qua MG (find\_customers\_linked\_to\_virtual\_number\_backend).

**3.3. Luồng Hoạt Động Chính**

1. **Khởi tạo:** app.py được chạy, thiết lập giao diện Streamlit và menu. Danh sách server VOS được tải từ config.py (và lưu vào st.session\_state.vos\_servers\_list).
2. **Người dùng tương tác:** Người dùng chọn một chức năng từ menu sidebar.
3. **Xử lý yêu cầu:**
   * app.py thu thập thông tin đầu vào từ người dùng (ví dụ: tên server, ID khách hàng, danh sách số cần thao tác).
   * Gọi các hàm xử lý tương ứng từ các module backend (ví dụ: customer\_management.find\_customers\_across\_all\_servers khi tìm khách hàng).
4. **Tương tác API:**
   * Các module backend sử dụng api\_client.call\_api để gửi yêu cầu đến URL API của server VOS3000 đã chọn.
   * api\_client.py xử lý việc gửi request, nhận và kiểm tra response.
5. **Xử lý dữ liệu:**
   * Các module backend nhận dữ liệu từ API, xử lý (ví dụ: lọc, định dạng bằng các hàm trong utils.py) và chuẩn bị kết quả.
6. **Hiển thị kết quả:**
   * app.py nhận kết quả từ backend và hiển thị lên giao diện người dùng bằng các thành phần của Streamlit (bảng, thông báo, form).

**4. Thực Hiện và Triển Khai Chức Năng**

Dựa trên thiết kế và các module đã phân tích, các chức năng chính được triển khai như sau:

* **Cấu hình Server VOS3000 (Trang "Configure Server"):**
  + Người dùng chọn một server VOS từ danh sách thả xuống.
  + **Tab Mapping Gateway:**
    - Liệt kê/Tìm kiếm MG theo tên.
    - Hiển thị chi tiết MG được chọn (tên, account, calloutCallerPrefixes, v.v.).
    - Cho phép thao tác với calloutCallerPrefixes: đếm, thêm, xóa. Các thay đổi được cập nhật qua API ModifyGatewayMapping.
  + **Tab Routing Gateway:**
    - Liệt kê/Tìm kiếm RG theo tên.
    - Hiển thị chi tiết RG được chọn.
    - Cho phép quản lý rewriteRulesInCaller (thêm mới, sửa, xóa cặp số ảo - số thật), callinCallerPrefixes, callinCalleePrefixes. Các thay đổi được cập nhật qua API ModifyGatewayRouting.
* **Quản lý Khách Hàng (Trang "Customer Information"):**
  + Tìm kiếm khách hàng theo Account ID hoặc Tên KH trên tất cả các server.
  + Hiển thị danh sách khách hàng tìm thấy.
  + Khi chọn một khách hàng, hiển thị thông tin chi tiết (Account, Name, Money, Limit Money, Lock Type, v.v.).
  + Cho phép thực hiện các hành động: Đặt hạn mức, Thêm hạn mức, Bớt hạn mức, Khóa/Mở tài khoản. Các thay đổi được cập nhật qua API ModifyCustomer.
* **Tra Cứu Thông Tin Số (Trang "Number Information"):**
  + Người dùng nhập danh sách số điện thoại/key cần tra cứu.
  + Hệ thống tạo các biến thể tìm kiếm cho mỗi số.
  + Quét qua tất cả các server, kiểm tra sự tồn tại của các biến thể trong calloutCallerPrefixes (MG), callinCallerPrefixes, callinCalleePrefixes, và rewriteRulesInCaller (RG).
  + Hiển thị kết quả chi tiết về vị trí tìm thấy.
* **Dọn Dẹp Gateway (Trang "Gateway Cleanup"):**
  + **Bước 1:** Nhập danh sách số cần xóa và chọn phạm vi quét (MG, RG, hoặc cả hai).
  + **Bước 2:** Hệ thống quét và hiển thị danh sách các MG/RG chứa các số đó, cho phép người dùng chọn các mục cụ thể để xóa.
  + **Bước 3 & 4:** Thực thi xóa và hiển thị log kết quả. Hệ thống gọi API ModifyGatewayMapping hoặc ModifyGatewayRouting để áp dụng thay đổi.
* **Quản lý Số Ảo (Trang "Virtual Number Management"):**
  + Quy trình nhiều bước cho phép người dùng:
    - Tìm kiếm một "Số ảo Đích".
    - Chọn một định nghĩa cụ thể nếu tìm thấy nhiều.
    - Chọn hành động: Sửa trực tiếp số thật (hướng dẫn người dùng qua trang "Configure Server") hoặc Thay thế/Lấy số thật từ một Nguồn khác.
    - Nếu chọn lấy từ nguồn khác:
      * Chọn loại nguồn: Backup tự động (ví dụ: [KeyĐích]bk trong cùng RG) hoặc một Key bất kỳ khác làm nguồn.
      * Nếu chọn Key nguồn thủ công: Tìm kiếm và chọn định nghĩa Key nguồn.
      * Xác nhận số lượng số thật muốn lấy từ nguồn.
      * Xác định hành động cho Số ảo Đích (Thêm vào hay Thay thế hoàn toàn).
      * Xử lý Key nguồn nếu nó trở nên rỗng sau khi lấy số (đặt "hetso", để trống, hoặc xóa key).
    - Xem lại và xác nhận tất cả thay đổi trước khi thực thi.
    - Hệ thống thực hiện cập nhật qua API ModifyGatewayRouting.

**5. Kết Quả Đạt Được**

Việc triển khai Hệ thống Quản lý Tập trung VOS3000 đã mang lại những kết quả tích cực và cải thiện đáng kể hiệu quả vận hành (chi tiết so sánh hiệu quả có thể tham khảo Phụ lục A nếu có, hoặc tài liệu so sánh riêng). Những lợi ích chính bao gồm:

* **Nâng cao hiệu quả quản lý:** Giảm đáng kể thời gian thực hiện các tác vụ cấu hình và quản lý thường nhật. Quản trị viên có thể thao tác trên nhiều server từ một giao diện duy nhất.
* **Giảm thiểu sai sót:** Tự động hóa một phần và chuẩn hóa quy trình qua giao diện web giúp giảm lỗi do thao tác thủ công.
* **Đơn giản hóa quy trình làm việc:** Các tác vụ phức tạp như tra cứu số toàn diện, dọn dẹp gateway hàng loạt, hay quản lý số ảo trở nên đơn giản và nhanh chóng hơn.
* **Tăng cường khả năng kiểm soát và minh bạch:** Giao diện tập trung cung cấp cái nhìn tổng quan hơn về trạng thái hệ thống.
* **Tối ưu hóa việc sử dụng tài nguyên:** Chức năng dọn dẹp gateway giúp giải phóng tài nguyên.
* **Nâng cao năng suất của đội ngũ quản trị:** Giảm gánh nặng công việc thủ công, cho phép tập trung vào các nhiệm vụ chiến lược hơn.

**6. Đánh Giá Hệ Thống**

**6.1. Ưu điểm**

* **Giao diện người dùng trực quan, thân thiện:** Streamlit giúp tạo giao diện web dễ sử dụng, giảm thời gian làm quen.
* **Tính năng quản lý tập trung mạnh mẽ:** Giải quyết hiệu quả bài toán quản lý nhiều server VOS3000.
* **Tự động hóa một phần quy trình:** Các chức năng như tra cứu, dọn dẹp đã tự động hóa các bước tốn thời gian.
* **Giảm thiểu rủi ro vận hành:** Hạn chế thao tác trực tiếp và chuẩn hóa qua giao diện giúp giảm nguy cơ lỗi.
* **Cấu trúc module rõ ràng:** Dễ bảo trì và mở rộng.

**6.2. Hạn chế và Khó khăn**

* **Hiệu suất với dữ liệu lớn:** Một số thao tác quét trên nhiều server có thể chậm nếu khối lượng dữ liệu (số lượng MG/RG, rules) quá lớn.
* **Phụ thuộc vào VOS3000 API:** Chức năng của hệ thống bị giới hạn bởi những gì VOS3000 API cung cấp.
* **Quản lý lỗi API:** Cần cải thiện việc hiển thị thông báo lỗi chi tiết và thân thiện hơn từ VOS API cho người dùng cuối.
* **Thiếu cơ chế rollback tự động:** Các thao tác cập nhật hàng loạt nếu gặp lỗi giữa chừng đòi hỏi rollback thủ công.
* **Bảo mật ứng dụng:** Phiên bản hiện tại chưa có cơ chế xác thực người dùng và phân quyền chi tiết cho chính ứng dụng quản lý.

**7. Phương Hướng Phát Triển và Hoàn Thiện**

Để hệ thống ngày càng hoàn thiện, các định hướng phát triển trong tương lai bao gồm:

* **Nâng cao hiệu suất và khả năng mở rộng:**
  + Tối ưu hóa các lời gọi API, xem xét caching dữ liệu.
  + Triển khai xử lý bất đồng bộ cho các tác vụ quét dữ liệu lớn.
* **Tăng cường Bảo mật:**
  + **Ưu tiên hàng đầu:** Triển khai hệ thống xác thực người dùng (đăng nhập) và phân quyền chi tiết dựa trên vai trò cho ứng dụng quản lý.
  + Ghi log chi tiết các hành động quan trọng.
* **Hoàn thiện tính năng Báo cáo và Thống kê:**
  + Xây dựng module báo cáo tự động về tình hình sử dụng tài nguyên, hiệu suất.
  + Tích hợp công cụ trực quan hóa dữ liệu.
* **Tự động hóa sâu hơn các tác vụ vận hành:**
  + Phát triển kịch bản tự động cho các nhiệm vụ định kỳ (tự động khóa tài khoản, dọn dẹp số ảo không dùng).
* **Hỗ trợ Đa ngôn ngữ:** Mở rộng giao diện để hỗ trợ thêm các ngôn ngữ khác.
* **Cải thiện Trải nghiệm Người dùng (UX):**
  + Liên tục thu thập phản hồi để cải tiến giao diện và luồng thao tác.
  + Thêm hướng dẫn chi tiết (tooltips, tài liệu) ngay trên giao diện.
* **Tích hợp với các Hệ thống Khác:** Xem xét khả năng tích hợp với CRM, Billing.

**8. Hướng Dẫn Cài Đặt Hệ Thống**

Phần này cung cấp các bước cần thiết để cài đặt và chạy Hệ thống Quản lý Tập Trung VOS3000.

**8.1. Yêu Cầu Hệ Thống**

* Python 3.9 trở lên (khuyến nghị 3.10+)
* PIP (Python package installer)

**8.2. Cài Đặt Môi Trường và Các Gói Phụ Thuộc**

1. Tạo Môi Trường Ảo (Khuyến Nghị):

Việc sử dụng môi trường ảo giúp quản lý các gói phụ thuộc tốt hơn và tránh xung đột giữa các dự án.

python -m venv venv

Kích hoạt môi trường ảo:

* + Trên Windows:
  + venv\Scripts\activate
  + Trên macOS/Linux:
  + source venv/bin/activate

1. Cài Đặt Các Gói Phụ Thuộc:

Ứng dụng này yêu cầu một số thư viện Python. Bạn có thể tạo một file requirements.txt với nội dung sau:

streamlit

streamlit-option-menu

pandas

requests

Sau đó, cài đặt các gói này bằng pip:

pip install -r requirements.txt

Hoặc, nếu bạn không có file requirements.txt, bạn có thể cài đặt từng gói một:

pip install streamlit streamlit-option-menu pandas requests

**8.3. Cấu Hình Server VOS**

1. Mở file config.py. Nếu bạn đặt các file backend trong thư mục con (ví dụ: backup), đường dẫn sẽ là backup/config.py.
2. Chỉnh sửa danh sách VOS\_SERVERS để bao gồm thông tin (tên gợi nhớ và URL API đầy đủ) của các server VOS3000 bạn muốn quản lý. Đảm bảo URL kết thúc bằng dấu /.
3. # Ví dụ trong config.py
4. VOS\_SERVERS = [
5. {"name": "VOS-01 (IP: 171.244.56.166)", "url": "http://171.244.56.166:3161/external\_new/server/"},
6. {"name": "VOS-02 (IP: 171.244.56.167)", "url": "http://171.244.56.167:2718/external/server/"},
7. # Thêm các server khác của bạn vào đây
8. ]

**8.4. Cấu Trúc Thư Mục**

Đảm bảo cấu trúc thư mục của dự án được tổ chức đúng cách để app.py có thể import các module backend:

* **Phương án 1 (Tất cả file ngang hàng):** Đặt tất cả các file Python (app.py, config.py, api\_client.py, utils.py, customer\_management.py, mapping\_gateway\_management.py, routing\_gateway\_management.py, ui\_utils.py) trong cùng một thư mục.
  + Nếu theo phương án này, hãy đảm bảo rằng bạn đã **xóa hoặc vô hiệu hóa** đoạn code thêm thư mục backup (hoặc tên thư mục con khác) vào sys.path ở đầu file app.py.
* **Phương án 2 (Sử dụng thư mục con cho backend, ví dụ** backup**):** Nếu bạn đặt các file backend (ngoại trừ app.py) vào một thư mục con (ví dụ: tên là backup), hãy đảm bảo rằng đoạn code sau có trong app.py (thường ở đầu file) để Python có thể tìm thấy các module đó:
* import sys
* import os
* current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))
* # Giả sử thư mục backend tên là 'backup' và nằm cùng cấp với app.py
* backend\_modules\_dir = os.path.join(current\_dir, "backup")
* if backend\_modules\_dir not in sys.path:
* sys.path.insert(0, backend\_modules\_dir)

*Lưu ý: Dựa trên mã nguồn app.py được cung cấp, hệ thống đang được cấu hình để tìm các module backend trong một thư mục con (đoạn mã sys.path.insert(0, backup\_dir) đã có).*

**8.5. Cách Chạy Ứng Dụng**

1. Mở terminal hoặc command prompt.
2. Điều hướng đến thư mục gốc của dự án (nơi chứa file app.py).
3. Nếu bạn sử dụng môi trường ảo, hãy kích hoạt nó (xem Bước 8.2.1).
4. Chạy lệnh sau:
5. streamlit run app.py
6. Streamlit sẽ tự động mở ứng dụng trong trình duyệt web của bạn. Nếu không, hãy mở trình duyệt và truy cập vào địa chỉ Local URL mà Streamlit hiển thị trên terminal (thường là http://localhost:8501).

**8.6. Gỡ Lỗi (Troubleshooting)**

* **Lỗi** ImportError**:**
  + Đảm bảo bạn đã cài đặt tất cả các gói trong requirements.txt (hoặc các gói được liệt kê ở Bước 8.2.2).
  + Kiểm tra kỹ cấu trúc thư mục và cấu hình sys.path trong app.py (xem Bước 8.4). Nếu các file backend nằm trong thư mục backup/, app.py phải có đoạn code để thêm thư mục này vào sys.path.
  + Đảm bảo tên các hàm được import trong app.py và giữa các module backend với nhau là chính xác (không có lỗi chính tả, phân biệt chữ hoa/thường).
* **Không kết nối được VOS Server:**
  + Kiểm tra lại URL và port của các server VOS trong config.py đã chính xác và có thể truy cập được từ máy tính chạy ứng dụng Streamlit.
  + Kiểm tra firewall trên server VOS hoặc trên máy tính của bạn có thể đang chặn kết nối.

**9. Bảng Tổng Quan Các Tính Năng (Trạng thái hiện tại)**

|  |  |  |
| --- | --- | --- |
| **Tính Năng Chính** | **Mô Tả Chi Tiết** | **Trạng Thái** |
| **Quản lý Server VOS3000** | Chọn server VOS3000 từ danh sách cấu hình để thực hiện các thao tác quản lý. Thêm server tạm thời trong phiên. | Hoàn thành |
| **Quản lý Khách hàng Tập trung** | Tìm kiếm khách hàng theo nhiều tiêu chí trên một hoặc nhiều server, xem chi tiết, chỉnh sửa hạn mức tín dụng, khóa/mở tài khoản. | Hoàn thành |
| **Quản lý Mapping Gateway (MG)** | Liệt kê, tìm kiếm MG. Xem chi tiết, thao tác với calloutCallerPrefixes. | Hoàn thành |
| **Quản lý Routing Gateway (RG)** | Liệt kê, tìm kiếm RG. Quản lý callinCallerPrefixes, callinCalleePrefixes, và rewriteRulesInCaller (số ảo). | Hoàn thành |
| **Tra cứu Số Toàn diện** | Tìm kiếm một hoặc nhiều số điện thoại/key trên tất cả server, xác định vị trí cấu hình. | Hoàn thành |
| **Dọn dẹp Gateway (MG & RG)** | Nhập danh sách số, quét và xác định các MG/RG chứa số đó, đề xuất và thực hiện xóa số khỏi các cấu hình liên quan. | Hoàn thành |
| **Quản lý Số Ảo Nâng cao** | Quy trình nhiều bước để thay thế/lấy số thật cho một Số ảo Đích từ một Nguồn khác (backup/key khác). | Hoàn thành |
| Phân tích và Báo cáo Nâng cao | Xây dựng báo cáo tự động, trực quan hóa dữ liệu. | Chưa triển khai |
| Tự động hóa Tác vụ Vận hành | Tự động hóa các công việc định kỳ. | Chưa triển khai |
| Bảo mật và Phân quyền Người dùng | Cơ chế đăng nhập, phân quyền chi tiết cho ứng dụng quản lý. | Chưa triển khai |

**10. Kết Luận**

Hệ thống Quản lý Tập trung VOS3000 được xây dựng bằng Python và Streamlit đã giải quyết thành công nhiều thách thức trong công tác vận hành và quản trị các cụm máy chủ VOS3000. Bằng việc cung cấp một giao diện web duy nhất, trực quan và các tính năng quản lý mạnh mẽ, hệ thống không chỉ giúp đơn giản hóa các tác vụ phức tạp, giảm thiểu sai sót mà còn nâng cao đáng kể hiệu quả làm việc của đội ngũ quản trị.

Mặc dù hệ thống đã đạt được những thành tựu quan trọng, việc tiếp tục đầu tư vào các hướng phát triển đã đề ra, đặc biệt là tăng cường bảo mật, nâng cao hiệu suất và bổ sung các tính năng tự động hóa, sẽ giúp hệ thống trở thành một công cụ không thể thiếu, đóng góp vào sự ổn định và phát triển bền vững của các dịch vụ VoIP dựa trên nền tảng VOS3000.