Intelligent Medical Examination Management System Integrating Recommendation System and Chatbot for Multi-Specialty Clinics

Nguyen Thi Hoang Khanh, Le Hoang Khang, Tran Dinh Kien
Department of Information Technology
Ho Chi Minh City University of Industry
Ho Chi Minh City, Vietnam
nguyenthihoangkhanh@iuh.edu.vn, hoangkhang.dev@gmail.com, kientrandinh.dev@gmail.com

***Abstract*—Vietnam's multi-specialty clinics rely on outdated manual processes, including queuing for registration, handwritten patient records prone to errors, manual clinical notes, and cash/bank transfer payments. Staff manually cross-reference separate ledgers, causing severe congestion during peak hours. These inefficiencies prolong examinations, increase patient waiting times, foster dissatisfaction, risk record mix-ups, and violate priority principles for policy beneficiaries, the elderly, pregnant women, and emergencies, delaying timely care. With Vietnam's population aging rapidly—projected to surpass 20% over 60 by 2030 [1]—coupled with rising chronic diseases and modern lifestyle demands, the traditional system faces immense pressure. Vietnam lags regionally in healthcare access, underscoring the need for digital transformation per the national strategy by 2025. Revita addresses these challenges as an intelligent management application for multi-specialty clinics. It integrates AI to auto-prioritize appointments per legal regulations [2], enables 24/7 online booking, manages real-time doctor schedules with conflict alerts, and streamlines registration, electronic records, and bank transfer payments. Additionally, Revita features an AI chatbot for direct patient assistance during visits and an NLP-based diagnosis recommendation system that extracts symptoms from notes to suggest accurate options. This allows doctors to focus on treatment, reduces waiting times, minimizes errors, and boosts efficiency, ensuring faster, equitable healthcare for all.**

***Index Terms*—Medical Examination Process, Patient Record Management, Medical Text-Based Disease Diagnosis, Next.js, Nest.js, AI Chatbot, Recommendation System**

Tóm tắt—Các phòng khám đa khoa tại Việt Nam hiện vẫn phụ thuộc vào quy trình thủ công lạc hậu, bao gồm xếp hàng đăng ký, ghi chép hồ sơ bệnh nhân bằng tay dễ sai sót, viết tay phiếu khám bệnh và thanh toán bằng tiền mặt/chuyển khoản. Nhân viên phải đối chiếu thủ công các sổ sách riêng lẻ, dẫn đến tình trạng ùn tắc nghiêm trọng vào giờ cao điểm. Những bất cập này làm kéo dài thời gian khám bệnh, tăng thời gian chờ đợi của bệnh nhân, gây bất mãn, nguy cơ nhầm lẫn hồ sơ và vi phạm nguyên tắc ưu tiên đối với người có công, người cao tuổi, phụ nữ mang thai và trường hợp cấp cứu, làm chậm trễ việc chăm sóc kịp thời. Với dân số Việt Nam đang già hóa nhanh chóng – dự kiến vượt 20% trên 60 tuổi vào năm 2030 [1] – cùng sự gia tăng bệnh mãn tính và nhu cầu lối sống hiện đại, hệ thống truyền thống đang chịu áp lực cực lớn. Việt Nam còn tụt hậu so với khu vực về tiếp cận dịch vụ y tế, do đó cần thúc đẩy chuyển đổi số theo chiến lược quốc gia đến năm 2025. Revita giải quyết các thách thức trên bằng ứng dụng quản lý thông minh dành cho phòng khám đa khoa. Hệ thống sử dụng cơ chế phân phối xoay vòng có ưu tiên để tự động ưu tiên cho bệnh nhân khi đến khám theo quy định pháp luật [2], hỗ trợ đặt lịch trực tuyến 24/7, quản lý lịch bác sĩ theo thời gian thực với cảnh báo xung đột, đồng thời đơn giản hóa quy trình đăng ký, hồ sơ điện tử và thanh toán chuyển khoản. Ngoài ra, Revita cung cấp chatbot AI hỗ trợ bệnh nhân trực tiếp trong quá trình khám và hệ thống gợi ý chẩn đoán dựa trên NLP – trích xuất triệu chứng từ ghi chú để đề xuất phương án chính xác. Nhờ đó, bác sĩ tập trung vào điều trị, giảm thời gian chờ, hạn chế sai sót, nâng cao hiệu quả, đảm bảo dịch vụ y tế nhanh hơn, công bằng hơn cho mọi người.

Từ khóa—Quy trình khám bệnh, Quản lý hồ sơ bệnh nhân, Chẩn đoán bệnh dựa trên văn bản y khoa, Next.js, Nest.js, Chatbot AI, Hệ thống gợi ý.

I. GIỚI THIỆU

Quy trình quản lý khám bệnh tại các phòng khám đa khoa ở Việt Nam bao gồm nhiều giai đoạn tuần tự: lấy số thứ tự, hoàn tất thủ tục hành chính để đăng ký và khám bệnh, chỉ định các dịch vụ chẩn đoán và thủ thuật cần thiết, thanh toán chi phí, và cuối cùng là bác sĩ chẩn đoán tình trạng bệnh, kê đơn thuốc cho bệnh nhân. Quy trình này đòi hỏi sự phối hợp chặt chẽ giữa nhiều bên liên quan, bao gồm nhân viên hành chính, bác sĩ, bệnh nhân và thu ngân. Với dân số già hóa nhanh chóng và nhu cầu y tế ngày càng gia tăng, việc duy trì hiệu quả và công bằng trong quản lý trở nên cực kỳ thách thức, đặc biệt khi các bước thủ công dẫn đến ùn tắc và sai sót.

Để giải quyết các vấn đề này, chúng tôi đã phát triển và triển khai *Revita: Hệ thống Quản lý Khám bệnh Thông minh Tích hợp Gợi ý Chẩn đoán và Chatbot dành cho Phòng khám Đa khoa* – một giải pháp công nghệ hiện đại nhằm tối ưu hóa và tự động hóa toàn bộ các giai đoạn trong quy trình khám bệnh. Hệ thống không chỉ giảm tải công việc hành chính mà còn nâng cao trải nghiệm tổng thể cho cả bệnh nhân và bác sĩ trong suốt hành trình chăm sóc sức khỏe.

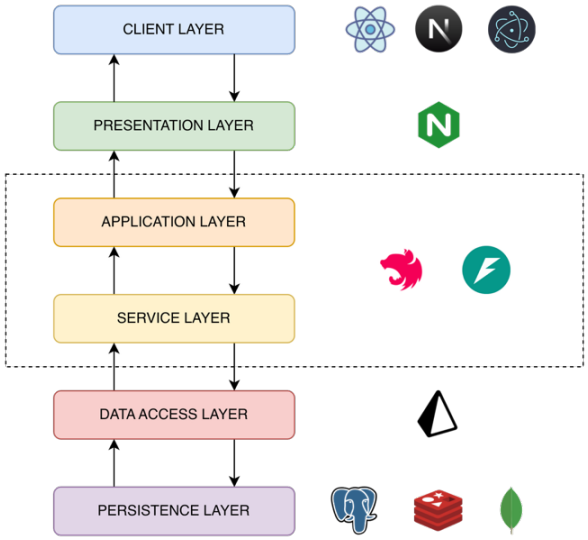
Revita được xây dựng trên các nền tảng web, di động và máy tính để bàn, sử dụng các công nghệ tiên tiến như Next.js, Nest.js, React Native và Electron. Điều này đảm bảo tính linh hoạt và khả năng truy cập, cho phép người dùng tương tác mượt mà mọi lúc, mọi nơi – từ đặt lịch khám trực tuyến đến hiển thị số thứ tự theo thời gian thực trên màn hình kiosk. Mục tiêu chính là nâng cao hiệu quả quản lý, cung cấp công cụ hỗ trợ tự động và tạo ra môi trường hiện đại, thân thiện với bệnh nhân cũng như nhân viên y tế.

Một tính năng nổi bật là hệ thống gợi ý chẩn đoán dựa trên ghi chú y khoa, công cụ thông minh được thiết kế chuyên biệt để hỗ trợ bác sĩ phân tích triệu chứng và đề xuất các tình trạng bệnh phù hợp. Hệ thống sử dụng xử lý ngôn ngữ tự nhiên (NLP) để trích xuất thông tin chính từ văn bản lâm sàng, kết hợp với thuật toán gợi ý dựa trên nội dung nhằm đưa ra các đề xuất chính xác và nhanh chóng – từ đó nâng cao độ tin cậy trong chẩn đoán.

II. CƠ SỞ LÝ THUYẾT

A. Kiến trúc phần mềm

Hệ thống Revita được thiết kế dựa trên kiến trúc phân tầng (layered architecture) [3], một mô hình kiến trúc phần mềm phổ biến giúp tổ chức hệ thống thành các tầng độc lập với trách nhiệm rõ ràng. Kiến trúc này đảm bảo tính mô đun hóa, khả năng bảo trì và mở rộng, đồng thời tách biệt các mối quan tâm giữa giao diện người dùng, logic nghiệp vụ và dữ liệu.



Hình 1. Sơ đồ kiến trúc của phần mềm

Tầng giao diện người dùng (Client Layer): bao gồm ba ứng dụng độc lập về nền tảng nhưng thống nhất về trải nghiệm người dùng. Ứng dụng web được xây dựng trên Next.js với kiến trúc App Router hiện đại [4]; ứng dụng di động được phát triển bằng React Native nhằm tối ưu hiệu năng trên cả hai hệ điều hành chính [5]; ứng dụng desktop được đóng gói bằng Electron để đáp ứng nhu cầu xử lý tác vụ nặng tại các phòng khám. Các ứng dụng này chỉ chịu trách nhiệm hiển thị dữ liệu và thu thập tương tác người dùng, không chứa bất kỳ logic nghiệp vụ nào.

Tầng trình diễn (Presentation Layer): được thực hiện bởi Nginx đóng vai trò cổng giao tiếp duy nhất với thế giới bên ngoài [7]. Nginx đảm nhiệm các chức năng chấm dứt kết nối bảo mật, định tuyến yêu cầu theo kênh người dùng, cân bằng tải, giới hạn tần suất truy cập và chuyển tiếp giao thức WebSocket mà không làm gián đoạn luồng dữ liệu thời gian thực, trong quá trình triển khai thì Nginx của hệ thống được Coolify tích hợp và quản lý tự động.

Tầng ứng dụng (Application Layer): bao gồm các instance NestJS được tổ chức theo mô hình Backend-for-Frontend [8]. Mỗi instance phục vụ riêng một kênh người dùng thông qua tập hợp các Controller chuyên biệt, thực hiện việc tiếp nhận yêu cầu, xác thực phiên làm việc, xác thực dữ liệu đầu vào và điều phối luồng xử lý xuống các tầng bên dưới. Việc tách biệt này cho phép tối ưu hóa cấu trúc phản hồi theo nhu cầu riêng của từng nền tảng mà không làm phức tạp hóa giao diện lập trình ứng dụng. Bên cạnh đó tầng này còn chứa hệ thống gợi ý chẩn đoán được xây dựng bằng FastAPI.

Tầng dịch vụ (Service Layer): tập trung toàn bộ quy tắc nghiệp vụ cốt lõi của hệ thống quản lý bệnh viện. Các Service tại tầng này được triển khai dưới dạng các đơn vị xử lý thuần túy, không phụ thuộc vào cơ sở dữ liệu hay giao thức truyền tải. Tại đây diễn ra các quyết định nghiệp vụ quan trọng như kiểm tra ràng buộc lịch khám, tính toán độ ưu tiên khi phân phối bệnh nhân vào các quầy thủ tục, các phòng khám/buồng khám phù hợp, xử lý thanh toán đa phương thức (tiền mặt/chuyển khoản), mã hóa/giải mã thông tin bệnh án khi lưu trữ/đọc ra, đảm bảo tính nhất quán và khả năng kiểm thử độc lập.

Tầng truy cập dữ liệu (Data Access Layer): được xây dựng trên Prisma Client kết hợp với Repository Pattern. Tầng này đóng vai trò trung gian duy nhất giữa logic nghiệp vụ và các hệ cơ sở dữ liệu, cung cấp giao diện thống nhất và an toàn kiểu dữ liệu cho mọi thao tác đọc ghi. Nhờ cơ chế sinh mã tự động, mọi thay đổi trong mô hình dữ liệu đều được phản ánh tức thì trên toàn hệ thống mà không yêu cầu chỉnh sửa thủ công.

Tầng lưu trữ (Persistence Layer): sử dụng đồng thời ba hệ quản trị cơ sở dữ liệu phù hợp với đặc thù dữ liệu. PostgreSQL đảm bảo tính toàn vẹn và nhất quán cho các thực thể có quan hệ phức tạp phù hợp cho các hệ thống lưu trữ nhiều dữ liệu phức tạp và cần mối quan hệ chặt chẽ đặt biệt trong lĩnh vực y tế; MongoDB phục vụ lưu trữ tài liệu phi cấu trúc ở đây là dữ liệu về thuốc được lấy về từ OpenFDA [19], vì lượng dữ liệu về dược này rất lớn, có cấu trúc đa dạng nên được lưu trữ trong MongoDB để giảm thời gian backup trên cơ sở dữ liệu chính cũng như tận dụng MongoDB để tối ưu hóa cơ chế tìm kiếm thuốc; Redis được sử dụng làm bộ nhớ đệm và sử dụng Redis Stream (đóng vai trò như các message queue) để xử lý các yêu cầu đồng thời với độ trễ thấp, nhận và phát các sự kiện khi người dùng bốc số vào các quầy thủ tục/phòng khám.

B. Hệ thống chẩn đoán bệnh

1) TF-IDF

Trọng số TF-IDF mô hình hoá mức độ "thông tin" của n-gram bằng cách kết hợp cường độ xuất hiện tại tài liệu với độ hiếm trên toàn bộ tập [9]. Để giảm thiên lệch theo độ dài và ổn định số, chúng tôi sử dụng TF dưới tuyến tính (sublinear TF) rồi chuẩn hoá L_2 ở bước cuối. Cho thuật ngữ t trong tài liệu d, với N là số tài liệu huấn luyện và $df(t)$ là số tài liệu chứa t:

$$idf(t) = \log\left(\frac{1+N}{1+df(t)}\right) + 1 \tag{1}$$

TF dưới tuyến tính được xác định bởi:

$$tf'(t,d) = \begin{cases} 0, & tf(t,d) = 0 \\ 1 + \log tf(t,d), & \text{ngược lại} \end{cases} \tag{2}$$

Trọng số TF-IDF cuối cùng là:

$$tfidf(t,d) = tf'(t,d) \cdot idf(t) \tag{3}$$

Sau khi tính cho toàn bộ từ vựng, véc-tơ đặc trưng được chuẩn hoá:

$$\widehat{\mathbf{x}}_d = \frac{\mathbf{x}_d}{\| \mathbf{x}_d \|_2} \tag{4}$$

Trong thiết lập dự đoán ICD từ discharge note, biểu diễn TF-IDF thưa (vocabulary lớn) được ghép với thuộc tính tuổi (chuẩn hoá) và giới tính (one-hot) để tạo biểu diễn đầu vào thống nhất cho các bộ phân loại nhị phân ở các bước sau.

2) One-vs-Rest (OvR) reduction

Chiến lược One-Vs-Rest (OvR) ánh xạ bài toán đa nhãn thành tập các bài toán nhị phân độc lập, mỗi nhãn k đi kèm một bộ phân loại tuyến tính riêng. Với đầu vào $\mathbf{x} \in \mathbb{R}^F$, điểm tuyến tính được chuyển thành xác suất hậu nghiệm:

$$p_k = \sigma\left(\mathbf{w}_k^\top \mathbf{x} + b_k\right) \tag{5}$$

Cấu trúc này cho phép huấn luyện song song theo nhãn và suy luận đồng thời bằng phép nhân ma trận:

$$\mathbf{p} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{6}$$

trong đó $\mathbf{W} \in \mathbb{R}^{K \times F}$ và $\mathbf{b} \in \mathbb{R}^K$. Việc chọn nhãn đầu ra có thể dựa trên ngưỡng τ hoặc Top-K' xác suất lớn nhất. Đối với bộ mã ICD quy mô lớn, OvR duy trì tính tuyến tính theo số nhãn, đồng thời giữ khả năng diễn giải: hệ số lớn theo chiều của một n-gram phản ánh tín hiệu mạnh cho nhãn tương ứng [10].

3) SGD (Stochastic Gradient Descent) optimizer

Tham số của từng bộ phân loại OvR được ước lượng bằng hạ gradient ngẫu nhiên với mất mát logistic và chuẩn hoá L_2, phù hợp dữ liệu thưa và khối lượng lớn. Với mẫu $(\mathbf{x}_i, y_{i,k})$, xác suất được tính bằng hàm sigmoid:

$$p_{i,k} = \sigma\left(\mathbf{w_k}^\top \mathbf{x_i} + b_k\right) \tag{7}$$

và hàm mục tiêu chính:

$$\mathcal{L}_{\|}\left(\mathbf{w_k}, b_k\right) = \frac{1}{n}\sum_{i=1}^{n}\Big[- y_{i,k}\log p_{i,k} - (1-y_{i,k})\log\left(1-p_{i,k}\right)\Big] + \frac{\lambda}{2}||\mathbf{w_k}||_2^2 \tag{8}$$

trong đó $\lambda > 0$ kiểm soát mức phạt để hạn chế quá khớp. Gradient được tính trên minibatch và cập nhật tham số với tốc độ học α để tối ưu hàm mục tiêu. Khi áp dụng cho discharge note, biểu diễn TF-IDF kết hợp thuộc tính tuổi/giới tính tạo điều kiện hội tụ ổn định của SGD, trong khi ma trận trọng số $\mathbf{W}$ cho phép suy luận hàng loạt và giải thích tuyến tính mức đặc trưng [11].

4) Ứng dụng Scikit-learn trong pipeline dự đoán

Quy trình huấn luyện được triển khai bằng hệ sinh thái Scikit-learn [12], theo các bước tổng quát:

- **Chuẩn bị dữ liệu**: nạp bộ hồ sơ thống nhất, san phẳng danh sách ICD và chuẩn hoá nội dung ghi chú (cắt chiều dài, loại bỏ ký tự dư thừa) trước khi tạo trường `text_clean`.
- **Chọn nhãn**: Chỉ giữ các bệnh có mức xuất hiện đủ lớn trong tập dữ liệu nhằm bảo đảm tính ổn định khi huấn luyện.
- **Chia dữ liệu**: dữ liệu được chia thành ba tập độc lập cho train/validation/test. Quá trình chia được thực hiện theo hai bước: đầu tiên tách tập test và giữ nguyên trong suốt quá trình huấn luyện; sau đó từ phần còn lại, tách thêm tập validation. Việc chia dữ liệu được thực hiện theo `subject_id` sử dụng `GroupShuffleSplit` thay vì chia ngẫu nhiên theo từng record, nhằm đảm bảo tất cả các lần nhập viện của cùng một bệnh nhân chỉ xuất hiện trong một tập duy nhất. Cách tiếp cận này tránh hiện tượng rò rỉ dữ liệu (data leakage) khi cùng một bệnh nhân xuất hiện ở cả tập huấn luyện và tập kiểm tra, đồng thời phản ánh đúng mục tiêu thực tế của hệ thống là dự đoán cho các bệnh nhân hoàn toàn mới chưa từng xuất hiện trong dữ liệu huấn luyện. Tập validation được sử dụng để điều chỉnh siêu tham số và đánh giá hiệu năng trong quá trình huấn luyện, trong khi tập test chỉ được sử dụng một lần duy nhất ở giai đoạn đánh giá cuối cùng để đảm bảo tính khách quan.
- **Vector hoá**: áp dụng `TfidfVectorizer` với cấu hình n-gram từ đơn và từ ghép, sử dụng sublinear TF, chuẩn hoá L_2 và giới hạn kích thước từ vựng để kiểm soát bộ nhớ.

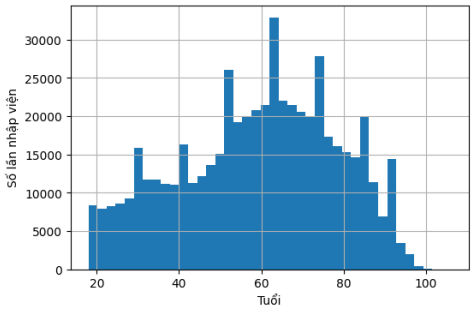
- **Mã hoá nhãn**: `MultiLabelBinarizer` định nghĩa thứ tự lớp thống nhất; các nhãn quá hiếm được tự động loại bỏ khỏi huấn luyện nhằm tránh cảnh báo trong quá trình dừng sớm.
- **Huấn luyện**: `OneVsRestClassifier` bao bọc `SGDClassifier` với hàm mất mát logistic, regularization L_2.
- **Đóng gói**: sau khi huấn luyện hoàn tất, toàn bộ các thành phần cần thiết cho suy luận được đóng gói thành một file duy nhất sử dụng `joblib`, bao gồm: bộ phân loại `OneVsRestClassifier` đã được huấn luyện, vectorizer từ vựng (`word_vec`), vectorizer ký tự tùy chọn (`char_vec`), bộ mã hoá nhãn đa lớp (`MultiLabelBinarizer`) để ánh xạ giữa mã ICD và chỉ số lớp, cùng với các tham số cấu hình. Để tối ưu dung lượng lưu trữ, tất cả các trọng số của mô hình được chuyển đổi sang kiểu dữ liệu `float32` trước khi lưu. File đóng gói này cho phép triển khai mô hình một cách độc lập mà không cần tái huấn luyện, đảm bảo tính nhất quán giữa môi trường phát triển và sản xuất.
- **Suy luận**: quy trình dự đoán được thực hiện qua các bước tuần tự. Đầu tiên, hệ thống nhận thông tin đầu vào bao gồm tuổi, giới tính và ghi chú bệnh án của bệnh nhân. Tiếp theo, văn bản được vector hoá bằng `TfidfVectorizer` đã được huấn luyện trước đó, tạo ra biểu diễn TF-IDF tương ứng. Nếu có vectorizer ký tự, các đặc trưng ký tự được ghép nối với đặc trưng từ vựng để tạo vector đặc trưng cuối cùng. Mô hình `OneVsRestClassifier` tính toán xác suất cho từng mã ICD thông qua phương thức `predict_proba`, sau đó các xác suất được sắp xếp theo thứ tự giảm dần và chọn top-K mã có xác suất cao nhất. Cuối cùng, hệ thống ánh xạ các mã ICD sang tên bệnh tương ứng từ bảng tra cứu và trả về danh sách các chẩn đoán kèm theo xác suất dự đoán cho bác sĩ tham khảo.

C. Công nghệ sử dụng trong hệ thống

- **NextJS** được sử dụng để xây dựng giao diện web quản trị và đặt lịch khám trực tuyến 24/7, tận dụng Server-Side Rendering (SSR) và Static Site Generation (SSG) nhằm tối ưu tốc độ tải trang, cải thiện SEO và đảm bảo trải nghiệm người dùng mượt mà trên mọi thiết bị [4].
- **NestJS** xây dựng hệ thống backend trung tâm của Revita, quản lý toàn bộ logic nghiệp vụ như lịch khám, hồ sơ bệnh nhân, ưu tiên theo quy định pháp luật [2] và thanh toán chuyển khoản [8].
- **React Native** phát triển ứng dụng di động cho bệnh nhân và bác sĩ, cho phép đặt lịch, nhận thông báo, tra cứu hồ sơ trên cả iOS và Android từ một mã nguồn duy nhất [5].
- **Electron** xây dựng ứng dụng desktop cho các màn hình chờ và kiosk tự phục vụ, hiển thị số thứ tự theo thời gian thực [6].
- **Docker** chuẩn hóa môi trường triển khai dịch vụ, bảo đảm tính nhất quán giữa giai đoạn phát triển và vận hành [13].
- **FastAPI** triển khai microservices AI, xử lý gợi ý chẩn đoán từ ghi chú y khoa và chatbot hỗ trợ bệnh nhân [14].
- **Gemini 2.5 Flash** tích hợp làm lõi AI thông minh cho chatbot tư vấn 24/7 hỗ trợ bệnh nhân khi đến khám tại phòng khám [15].
- **PostgreSQL** được sử dụng làm cơ sở dữ liệu chính để quản lý toàn bộ hệ thống nghiệp vụ lớn của Revita [16].
- **Redis** được triển khai như bộ nhớ đệm (cache) hiệu năng cao và sử dụng Redis Streams để lưu trữ luồng sự kiện theo thời gian thực [17].
- **MongoDB** được dùng để lưu trữ và quản lý dữ liệu thuốc với khối lượng lớn, cấu trúc linh hoạt [18].
- **Supabase** cung cấp nền tảng backend-as-a-service cho lưu trữ các tập tin đa phương tiện trong hệ thống. [20].

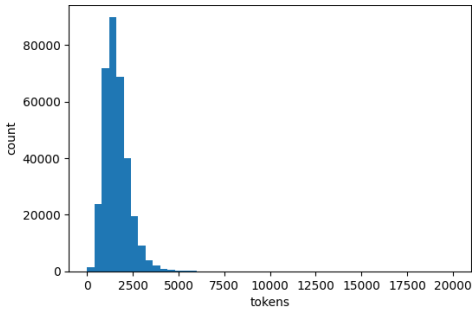
D. Dataset MIMIC-IV

MIMIC-IV là một bộ dữ liệu hồ sơ y tế điện tử (EHR) mã nguồn mở, được phát triển bởi nhóm nghiên cứu tại Beth Israel Deaconess Medical Center (BIDMC) và MIT Laboratory for Computational Physiology vào năm 2020, như một bản cập nhật từ phiên bản MIMIC-III. Bộ dữ liệu này áp dụng kiến trúc modular với các module riêng biệt như hosp (dữ liệu bệnh viện), icu (dữ liệu đơn vị chăm sóc tích cực), và ed (dữ liệu cấp cứu), được tổ chức bằng các bảng SQL để nhấn mạnh nguồn gốc dữ liệu và hỗ trợ tích hợp linh hoạt giữa các nguồn khác nhau. Dữ liệu được thu thập từ hệ thống EHR tùy chỉnh, hệ thống thông tin ICU (MetaVision), và các nguồn bên ngoài, sau đó được biến đổi và khử nhận dạng theo quy định HIPAA. Chức năng chính của MIMIC-IV là cung cấp hơn 250.000 hồ sơ bệnh nhân từ năm 2008-2019, bao gồm đo lường sinh lý, đơn thuốc, chẩn đoán, thủ thuật, và ghi chú lâm sàng, nhằm hỗ trợ nghiên cứu trong lĩnh vực thông tin lâm sàng, dịch tễ học, và học máy, đặc biệt trong phân tích chăm sóc bệnh nhân cấp cứu và ICU [21].



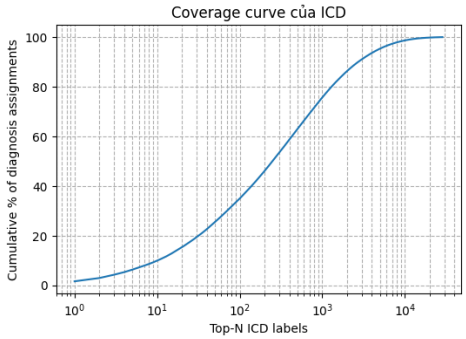
Hình 2. Phân bố độ tuổi nhập viện

Phân bố tuổi bệnh nhân trong MIMIC-IV có dạng đa đỉnh (multimodal), tập trung mạnh ở nhóm 50–80 tuổi, với đỉnh cao nhất tại ∼65 tuổi – phản ánh dân số bệnh nhân ICU chủ yếu là người trung niên và cao tuổi.



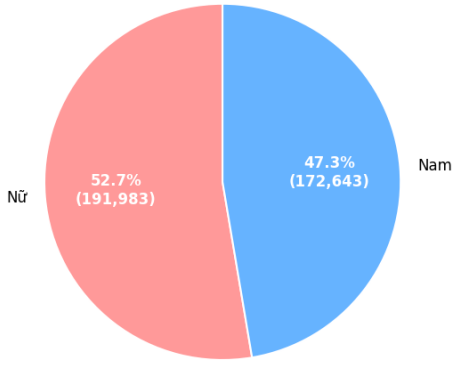
Hình 3. Phân bố độ dài của các bệnh án y khoa

Dữ liệu cho thấy phân bố độ dài văn bản có dạng lệch phải mạnh, trong đó phần lớn bệnh án tập trung ở mức 1.000–3.000 tokens, phù hợp với đặc trưng của ghi chú lâm sàng dài. Một số ít văn bản có độ dài rất lớn (trên 10.000 tokens), tạo thành phần đuôi dài của phân bố. Đặc điểm này cho thấy mô hình cần cơ chế xử lý văn bản dài hoặc rút gọn thông minh để tránh mất thông tin quan trọng.



Hình 4. Phân bố mã ICD

Trong MIMIC-IV, chỉ 100 mã ICD phổ biến nhất đã chiếm ∼65% tổng số chẩn đoán, và 1.000 mã chiếm ∼90%. Đường cong tăng nhanh ở đầu và chậm dần về sau thể hiện hiện tượng đuôi dài mạnh. Với Revita, việc tập trung huấn luyện trên top 100–500 mã đảm bảo độ chính xác cao cho phần lớn ca bệnh thực tế, đồng thời tối ưu tài nguyên khi triển khai hệ thống gợi ý chẩn đoán.



Hình 5. Tỉ lệ giới tính của bệnh nhân

Trong MIMIC-IV, nữ giới chiếm 52.7% (191.383 bệnh nhân) và nam giới chiếm 47.3% (172.643 bệnh nhân) trong tổng số 364.026 bệnh nhân duy nhất. Tỷ lệ này cho thấy sự cân bằng tương đối giữa hai giới, với nữ giới nhỉnh hơn nhẹ – phù hợp với đặc điểm dân số nhập viện ICU. Với Revita, hệ thống gợi ý chẩn đoán và quản lý hồ sơ được thiết kế không thiên vị giới tính, đảm bảo độ chính xác đồng đều trên cả hai nhóm bệnh nhân.

III. HỆ THỐNG QUẢN LÝ QUY TRÌNH KHÁM CHỮA BỆNH

A. Quy trình khám chữa bệnh tại phòng khám

Bước 1: Bệnh nhân có thể đặt lịch khám trước sau đó đến phòng khám, nhập thông tin khám bệnh nếu đăng ký khám lần đầu, nếu đã từng đến khám tại bệnh viện thì bệnh nhân nhập mã hồ sơ khám/ số điện thoại hồ sơ/mã đặt lịch (nếu có) sau đó hệ thống sẽ phân phối bệnh nhân vào các quầy làm thủ tục khám với cơ chế xoay vòng có ưu tiên.

Bước 2: Nhân viên quầy thủ tục nhấn gọi bệnh nhân đến quầy làm thủ tục, bệnh nhân quan sát thông tin của mình trên màn hình chờ để biết khi nào chuẩn bị đến làm thủ tục.

Bước 3: Bệnh nhân sau khi làm thủ tục khám thì cầm theo phiếu chỉ định các dịch vụ đến quầy thu ngân nộp để thanh toán các phí dịch vụ theo yêu cầu. Nhân viên thu nhân nhận phiếu chỉ định và bắt đầu tạo hóa đơn thanh toán với các dịch vụ bệnh nhân muốn thực hiện. Có hai phương thức thanh toán là bằng tiền mặt (nhân viên thu ngân xác nhận thanh toán thành công khi nhận tiền) và thanh toán qua hình thức chuyển khoản nhân viên tạo mã chuyển khoản tương ứng với hóa đơn và đưa mã chuyển khoản cho bệnh nhân, bệnh nhân thực hiện chuyển khoản (thông qua tài khoản ngân hàng hoặc tài khoản ví điện tử) khi chuyển khoản thành công thì hệ thống tự động cập nhật thông tin và xuất hóa đơn tự động.

Bước 4: Nếu bệnh nhân đặt lịch trước đó hoặc là yêu cầu chọn bác sĩ khi ở quầy thủ tục thì khi thanh toán thành công bệnh nhân sẽ được thêm vào hàng chờ của bác sĩ đó. Nếu không thì bệnh nhân sẽ đến trước khu vực khám và checkin tại các máy kiosk hoặc nhờ nhân viên ở tại quầy thủ tục (khu vực khám bệnh) để xác nhận vào khám, khi xác nhận xong thì hệ thống đưa bệnh nhân vào hàng chờ.

Bước 5: Bác sĩ nhấn gọi bệnh nhân vào phòng khám theo thứ tự hàng chờ, sau khi vào khám thì bác sĩ tiến hành chẩn đoán, tạo bệnh án, đơn thuốc cho bệnh nhân. Nếu bác sĩ mong muốn bệnh nhân thực hiện thêm các dịch vụ cho mục đích khám chữa bệnh thì bác sĩ sẽ tạo phiếu chỉ định cho bệnh nhân thực hiện các dịch vụ đó, sau đó quay về bước 3.

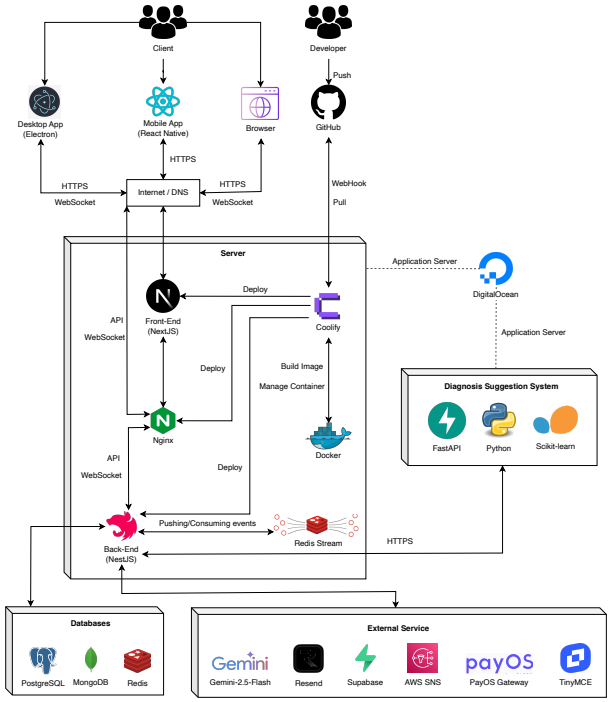
Bước 6: Khi đã hoàn thành các dịch vụ theo yêu cầu của bác sĩ thì bệnh nhân tiến hành quay lại khu vực khám để checkin chờ kết quả.

B. Một số chức năng chính

- Đề xuất/Xác nhận lịch làm việc.
- Đặt lịch hẹn khám bệnh
- Lấy số thứ tự để đến quầy thủ tục
- Cập nhật bệnh án, kết quả sau khi khám/thực hiện các dịch vụ.

- Quản lý phiếu chỉ định, bệnh án, đơn thuốc
- Hệ thống chatbot hỗ trợ bệnh nhân (Không hỗ trợ chẩn đoán, đơn thuốc, … các yếu tố có thể ảnh hưởng đến sức khỏe của bệnh nhân.
- Hệ thống gợi ý các chẩn đoán dựa trên bệnh án của bệnh nhân (chỉ cho các bác sĩ trong hệ thống sử dụng)

C. Sơ đồ triển khai hệ thống



Hình 6. Sơ đồ triển khai hệ thống

Hệ thống được triển khai theo mô hình CI/CD tự động với sự hỗ trợ của Coolify và Docker [13] trên nền tảng DigitalOcean, bám sát các thực hành DevOps hiện đại [22]. Khi lập trình viên đẩy (push) mã nguồn lên GitHub, Coolify nhận webhook và tự động pull phiên bản mới nhất, build Docker image, sau đó triển khai lên các application server. Nginx được tích hợp trực tiếp trong Coolify để quản lý định tuyến HTTPS và WebSocket, đảm bảo kết nối an toàn và real-time giữa client (Web, Mobile, Desktop) và server [7].

Cơ sở dữ liệu bao gồm PostgreSQL (dữ liệu cấu trúc), MongoDB (dữ liệu phi cấu trúc) và Redis (cache, session, stream) được triển khai trực tiếp trên server. Redis Stream đóng vai trò hàng đợi sự kiện (event queue), hỗ trợ xử lý bất đồng bộ các tác vụ như thông báo, cập nhật lịch hẹn, và đồng bộ dữ liệu real-time qua WebSocket.

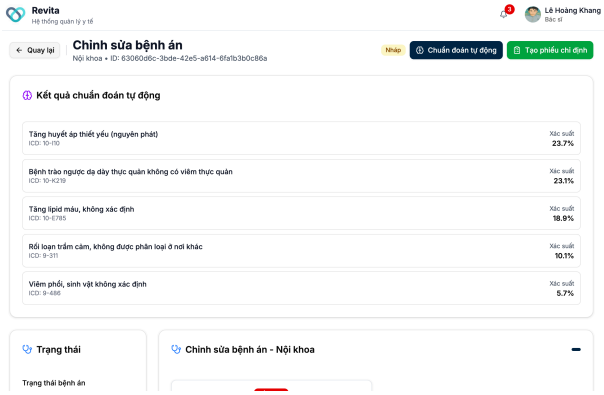
Hệ thống tích hợp nhiều dịch vụ bên thứ ba:

- Gemini-2.5-Flash cung cấp gợi ý AI
- Resend gửi email thông báo [23]
- Supabase lưu trữ ảnh, tài liệu
- AWS SNS gửi mã xác thực OTP qua SMS [24]
- PayOS xử lý thanh toán trực tuyến [25]
- TinyMCE hỗ trợ soạn thảo văn bản phong phú

Riêng hệ thống gợi ý chẩn đoán được huấn luyện bằng Python với thư viện Scikit-learn [12], triển khai dưới dạng microservice sử dụng FastAPI. Khi bác sĩ yêu cầu chẩn đoán, NestJS (Back-End) chuẩn hóa dữ liệu đầu vào, sau đó gọi HTTPS đến FastAPI để nhận danh sách các chẩn đoán gợi ý, đảm bảo tính chính xác và tốc độ phản hồi.

IV. TRIỂN KHAI

A. Một số màn hình giao diện



Hình 7. Chẩn đoán tự động dựa trên thông tin đề xuất

Bác sĩ sẽ chọn bệnh án mà muốn chẩn đoán, và nhấn nút chẩn đoán. Hệ thống sẽ lấy toàn bộ nội dung của bệnh án đó kèm theo các kết quả khi thực hiện các dịch vụ và gọi tới hệ thống gợi ý để chẩn đoán. Kết quả trả về sẽ hiển thị theo thứ tự phần trăm xác suất cao nhất mà bệnh nhân có thể mắc phải.

Hình 8. Tra cứu thông tin khám bệnh bằng chatbot AI

Bệnh nhân có thể sử dụng chatbot AI để được hỗ trợ trong quá trình thăm khám, ví dụ như hỏi bác sĩ nào đó có lịch làm việc hay không, hỏi giá các dịch vụ, … Chatbot sẽ từ chối hỗ trợ những câu hỏi có thể ảnh hưởng tới sức khỏe của người bệnh, ví dụ như thuốc, cách tự chữa trị, …

B. Kết quả thực nghiệm

Mô hình được huấn luyện theo hướng phân loại đa nhãn (multi-label) nhằm dự đoán các mã ICD bệnh lý dựa trên tóm tắt ra viện (discharge summary). Phương pháp: One-vs-Rest Logistic Regression.

- Đặc trưng đầu vào: TF-IDF n-gram (1–2 từ) trên văn bản đã làm sạch.
- Dữ liệu huấn luyện: ∼400 k hồ sơ MIMIC-IV được chia thành ba tập với tỷ lệ 70%/15%/15% cho train/validation/test. Việc chia dữ liệu được thực hiện theo `subject_id` để đảm bảo không có bệnh nhân nào xuất hiện ở nhiều tập khác nhau, tránh hiện tượng rò rỉ dữ liệu.
- Đầu ra: xác suất của từng mã ICD (≈ 40000 nhãn)
- Chiến lược đánh giá: top-K (1, 3, 5, 10)

Bảng I
Kết quả định lượng

Chỉ số	**Kết quả**	**Giải thích**
Tổng số ca kiểm tra	50	Mẫu từ tập test
Hit@1	76%	76% ca có ít nhất 1 mã đúng ở vị trí đầu tiên
Hit@3	94%	94% ca có ít nhất 1 mã đúng trong 3 mã đầu
Hit@5	96%	96% ca có ít nhất 1 mã đúng trong top-5
Hit@10	98%	49/50 ca có ít nhất 1 mã đúng trong top-10
Trung bình số mã đúng / ca	2.62	Mỗi ca dự đoán trúng trung bình ∼2.6 bệnh
Số ca dự đoán đúng hoàn toàn	22/50 (44%)	Toàn bộ mã thật đều nằm trong top-10

V. KẾT LUẬN

Trong bài báo này, chúng tôi đã trình bày chi tiết quy trình xây dựng và triển khai hệ thống quản lý quy trình khám chữa bệnh tại các phòng khám đa khoa. Hệ thống tích hợp các công nghệ hiện đại như NextJS, NestJS, Electron, React Native, FastAPI, Gemini, kết hợp với mô hình dự đoán bệnh được huấn luyện từ tập dữ liệu y khoa nổi tiếng MIMIC-IV. Nhờ đó, hệ thống cung cấp gợi ý bệnh lý chính xác, nhanh chóng dựa trên ghi chú bệnh án, tuổi và giới tính, hỗ trợ hiệu quả cho bác sĩ trong công tác sàng lọc và chẩn đoán ban đầu.

Mặc dù đã đạt được những kết quả khả quan, hệ thống vẫn còn hạn chế lớn là chưa sử dụng dữ liệu thực tế từ chính hệ thống sinh ra trong quá trình vận hành để huấn luyện và cải thiện mô hình do hiện nay cơ chế gán nhãn dữ liệu cho các chẩn đoán của các y bác sĩ là phức tạp, cần sự tham gia thực hiện của con người. Và do hệ thống gợi ý chẩn đoán đang sử dụng dữ liệu huấn luyện đến từ tập MIMIC-IV nên mô hình chẩn đoán sẽ bị một số hạn chế từ tập dữ liệu này, dù là một tập dữ liệu cực kỳ lớn trong lĩnh vực y học nhưng MIMIC-IV chỉ mạnh về một số nhóm bệnh cụ thể, chẳng hạn như: nhóm bệnh tim mạch, nhiễm trùng, suy đa tạng, biến chứng sau phẫu thuật, nhóm bệnh huyết áp…

Trong thời gian tới, nhóm sẽ tập trung: Thu thập và tích hợp dữ liệu nội sinh từ hệ thống để tái huấn luyện mô hình định kỳ, nâng cao độ chính xác và phù hợp với thực tế lâm sàng tại Việt Nam. Cải thiện trải nghiệm người dùng: tối ưu giao diện, tăng tốc độ phản hồi, hỗ trợ đa ngôn ngữ và tích hợp voice-to-text cho ghi chú bệnh án. Nâng cấp mô hình dự đoán: thử nghiệm các mô hình ngôn ngữ y khoa chuyên biệt (như BioClinicalBERT), bổ sung xử lý phủ định và ngữ cảnh phức tạp. Mở rộng nền tảng: tích hợp với hệ thống bệnh viện (HIS), và triển khai thử nghiệm thực tế tại các cơ sở y tế.

TÀI LIỆU THAM KHẢO

[1] Nhật Dương, "Việt Nam sẽ có 18 triệu người cao tuổi vào năm 2030," *VnEconomy*, Jan. 10, 2025. [Online]. Available: https://vneconomy.vn/viet-nam-se-co-18-trieu-nguoi-cao-tuoi-vao-nam-2030.htm

[2] Quốc hội nước Cộng hòa xã hội chủ nghĩa Việt Nam, "Luật Khám bệnh, chữa bệnh (Luật số 15/2023/QH15)," *Công Báo*, no. 489–490, Feb. 19, 2023.

[3] M. Richards, *Software Architecture Patterns*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2022.

[4] A. Tazetdinov, *Next.js Cookbook: Learn How to Build Scalable and High-Performance Apps from Scratch*. New Delhi, India: BPB Publications, 2023.

[5] D. Abbott, H. Djirdeh, A. Accomazzo, and S. Shoemaker, *Fullstack React Native: The Complete Guide to React Native*. San Francisco, CA, USA: Fullstack.io, 2017.

[6] S. Kinney, *Electron in Action*, 2nd ed. Shelter Island, NY, USA: Manning Publications, 2025.

[7] C. Nedelcu, *Nginx HTTP Server*, 4th ed. Birmingham, U.K.: Packt Publishing, 2018.

[8] G. Magolan, J. Bell, D. Guijarro, A. de Peretti, and P. Housley, *NestJS: A Progressive Node.js Framework*. Santa Rosa, CA, USA: Bleeding Edge Press, 2018.

[9] C. D. Manning, P. Raghavan, and H. Schütze, "Scoring, term weighting and the vector space model," in *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008, ch. 6, pp. 110–134.

[10] V. Ashwinkumar, P. P. Arage, R. Jeya, and P. Sudhakaran, "One-vs-Rest vs. Voting Classifiers for Multi-Label Text Classification: An Empirical Study," in *Proc. E3S Web Conf.*, vol. 491, p. 01014, 2024.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[12] R. Garreta and G. Moncecchi, *Learning scikit-learn: Machine Learning in Python*. Birmingham, U.K.: Packt Publishing, 2013.

[13] N. Poulton, *Docker Deep Dive*. Portland, OR, USA: Nigel Poulton, 2017.

[14] G. De Luca, *FastAPI Cookbook*, 1st ed. Birmingham, UK: Packt Publishing, 2024.

[15] Google Cloud, "Gemini 2.5 Flash | Generative AI on Vertex AI | Google Cloud Documentation." [Online]. Available: https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash. [Accessed: Dec. 1, 2025].

[16] R. O. Obe and L. S. Hsu, *PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.

[17] J. L. Carlson, *Redis in Action*. Shelter Island, NY, USA: Manning Publications, 2013.

[18] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

[19] T. A. Kass-Hout *et al.*, "OpenFDA: An innovative platform providing access to a wealth of FDA's publicly available data," *J. Amer. Med. Inform. Assoc.*, vol. 23, no. 3, pp. 596–600, May 2016, doi: 10.1093/jamia/ocv153.

[20] Supabase, "Storage | Supabase Docs." [Online]. Available: https://supabase.com/docs/guides/storage. [Accessed: Dec. 1, 2025].

[21] A. E. Johnson *et al.*, "MIMIC-IV, a freely accessible electronic health record dataset," *Sci. Data*, vol. 10, no. 1, p. 1, Jan. 2023.

[22] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. Portland, OR, USA: IT Revolution Press, 2016.

[23] Resend, "Managing Emails - Resend." [Online]. Available: https://resend.com/docs/dashboard/emails/introduction. [Accessed: Dec. 1, 2025].

[24] Amazon Web Services, *Amazon Simple Notification Service: Developer Guide*. Seattle, WA, USA: Amazon Web Services, Inc., 2025.

[25] PayOS, "PayOS API Documentation," 2025. [Online]. Available: https://payos.vn/docs/api/. Accessed: Nov. 17, 2025.