# Healthcare Navigator: A Knowledge Graph for Integrated Patient and Provider Services

Rahul Marathervar, Balaji Radhakrishnan Padmanabhan, Akshat Aggarwal, Yogesh Sangtani, Charani Tirumalareddy
rmarath4@asu.edu, bradhak5@asu.edu, aaggar66@asu.edu, ysangtan@asu.edu, vtiruma3@asu.edu
*Computer Software Engineering*
*Arizona State University*
Tempe, Arizona

***Abstract*—Patient navigation of the U.S. healthcare system is impeded by the fragmentation of provider, facility, and service data across disconnected silos. This paper introduces the Healthcare Navigator, a knowledge graph-driven framework designed to address this challenge by creating a unified, queryable healthcare ecosystem. Finding a suitable physician often requires consulting multiple sources for provider specializations. We construct a novel ontology to integrate heterogeneous public datasets, including physician specializations and hospital affiliations from the Centers for Medicare & Medicaid Services (CMS), patient satisfaction ratings, pharmaceutical dispensary locations, and symptom-to-precaution mappings. The resulting knowledge graph powers a decision-support application that enables multi-faceted, context-aware queries. Users can search by symptom to receive preliminary precautionary information and recommendations for relevant, highly-rated specialists in their geographic vicinity. The system leverages geospatial indexing to visualize results on an interactive map interface and suggest nearby pharmacies, completing the patient's care-seeking journey. Healthcare Navigator demonstrates a semantic approach to transforming siloed data into an actionable, patient-centric resource, significantly lowering the barrier to informed healthcare decisions.**

## I. Introduction

Navigating the U.S. healthcare system is a complex and often frustrating journey for patients. Critical information needed to make informed decisions is scattered across numerous, disconnected sources. A patient seeking a specialist must typically consult one website for insurance coverage, another for physician reviews, a third for hospital quality ratings, and separate mapping services to understand location and accessibility. This fragmentation of data forces the patient to manually cross-reference information, a process that is inefficient, prone to error, and presents a significant barrier to receiving timely and optimal care [1].

The main challenge comes from "data silos", separate datasets that can't easily share or work with each other [2]. A physician directory is unaware of a hospital's patient satisfaction scores, and a hospital database is not linked to the locations of nearby pharmacies. This disconnect prevents users from performing holistic, multi-faceted queries that reflect real-world needs, such as, "Find a highly-rated cardiologist for my symptoms who is affiliated with a top-tier hospital near my home and is accessible via public transit." Current systems lack the underlying semantic structure to answer such a question in a single step.

To address these challenges, this paper presents the Healthcare Navigator, a knowledge graph-driven system designed to create a single, unified, and intelligent source of healthcare information. At its core, our approach is to apply semantic web technologies to build a comprehensive ontology that formally defines key healthcare entities such as *Physician, Hospital, Pharmacy, and MedicalCondition* and the rich relationships between them, like *isAffiliatedWith, treatsCondition, and locatedNear* [3], [4].

We construct our knowledge graph by integrating several heterogeneous, large-scale public datasets. These include provider and facility data from the Centers for Medicare & Medicaid Services (CMS), which contains physician specialties and hospital quality ratings; authoritative symptom-to-precaution knowledge from health resources; and geospatial data for medical pharmacies and public transit networks. By converting this data into a cohesive Resource Description Framework (RDF) graph, we create a powerful, queryable network of interconnected information, a method increasingly used for large-scale public health analysis [5], [6].

The Healthcare Navigator system utilizes this knowledge graph to power a patient-centric web application. A key feature is the ability for users to search by symptoms. The system responds with preliminary precautionary advice and a ranked list of relevant, highly-rated specialists. These results are visualized on an interactive map interface, showing the locations of providers, their affiliated hospitals, and the nearest pharmacies, thereby providing a complete and actionable guide for the patient's care journey. This paper details the design of our ontology, the data integration pipeline, and the architecture of the final application, demonstrating a powerful model for transforming siloed data into a seamless healthcare navigation experience.

## II. Problem Definition

The core challenge addressed by this research is the persistent fragmentation and lack of semantic interoperability across healthcare information systems in the United States. Data concerning physicians, hospitals, and pharmacies is dispersed among numerous independent repositories,

each employing distinct data formats, identifiers, and terminologies. This heterogeneity inhibits the formation of a unified, patient-centric understanding of healthcare services and introduces a semantic disconnect between patients' complex informational needs and the isolated datasets currently available [2], [7]. Consequently, existing healthcare systems are unable to efficiently support complex, cross-domain queries that require reasoning over multiple dimensions—such as identifying highly rated specialists affiliated with top-performing hospitals within a given geographic area. The Healthcare Navigator project aims to address this issue by designing a unified, semantically enriched data model and constructing a healthcare knowledge graph that integrates publicly available datasets from sources such as CMS, NPPES, and HCAHPS. This knowledge-driven framework facilitates interoperability, enables intelligent query processing, and transforms fragmented healthcare data into a coherent, interconnected ecosystem capable of delivering comprehensive insights to patients, caregivers, and healthcare organizations alike [8]–[10].

This framework is realized as the Healthcare Navigator, a patient-centric web application and decision-support system designed to address these challenges through three key use cases. First, it provides a symptom-based search that offers preliminary precautionary information, guiding users on immediate steps. Second, it features a multi-faceted query engine that allows users to find relevant, highly-rated specialists based on their reported symptoms, location, and other filters. Finally, it includes an interactive map interface to visualize the locations of providers, their affiliated hospitals, and nearby pharmacies, thus completing the patient's care-seeking journey.

## III. Literature Review

Research on the use of semantic web technologies to address data fragmentation in the healthcare industry is well established, with the main objective being the development of unified ontologies that will enable more sophisticated query systems. The fundamental idea of using semantic models for provider matching and service composition was validated by pioneering work in this field. To match patient symptoms with reputable specialists, Sethuraman et al. [11], for example, presented a novel framework utilizing a multi-agent system that combined sentiment analysis and machine learning clustering with an RDF-Schema. The system only offered a single recommendation, failing to incorporate the larger context of hospital affiliations or pharmaceutical services, even though this showed promise for integrating quality metrics. In the same way, Fayçal and Abdelkamel's work [4] clearly validated the use of a formal ontology to logically link diseases with corresponding physicians and drugs; however, its practicality was limited due to its dependence on a static, conceptual dataset. Subbulakshmi et al. [3] advanced this paradigm by introducing a context-aware system that dynamically composed healthcare services by integrating geospatial awareness. However, their model was designed for local emergency coordination, demonstrating the usefulness of location-based semantics but failing to address the problem of navigating non-emergency care on a national level. Together, these studies created a strong basis for semantic provider matching, but they also made it evident that a more comprehensive, data-rich strategy was required to navigate the entire patient care pathway.

As the field has developed, the emphasis has moved from creating custom ontologies to creating extensive, diverse healthcare knowledge graphs. Abu-Salih et al.'s systematic review [1] thoroughly documents this trend. This recent research wave shows how combining complex and varied datasets can yield deeper insights. Majdalawieh et al. [5] demonstrated the enormous potential for sophisticated public health analytics by creating a Semantic Knowledge Graph that combined clinical disease records with demographic and environmental data. Similar to this, Shi et al. [6] concentrated on building a knowledge graph from unstructured medical texts, like electronic health records (EHRs), in order to produce a comprehensive Medical Knowledge Model for complex clinical query answering and interpretation. Although these studies highlight the analytical capabilities and scalability of contemporary knowledge graph construction, their applications are primarily intended for researchers, public health officials, and clinicians rather than as direct, patient-facing navigation tools.

The significance of knowledge-graph-driven approaches extends beyond academia, reflecting a growing recognition in industry of semantic technologies as essential for addressing healthcare interoperability challenges [2], [7]. Leading technology providers are currently advancing solutions that utilize graph databases to analyze and manage intricate healthcare data standards such as FHIR (Fast Healthcare Interoperability Resources), as evidenced by technical publications from Amazon Web Services and other industry leaders [9], [10]. The application of this technology is evolving, with recent studies emphasizing the integration of knowledge graphs and advanced AI techniques, such as Retrieval-Augmented Generation (RAG), to enhance the querying of complex FHIR data [8]. The significant industry adoption underscores the viability and importance of semantic integration; however, the primary emphasis continues to be on backend data management, clinical support, and enterprise-level analytics, resulting in a deficiency of tools aimed at directly empowering patients.

By bridging the semantic divide between fragmented data and direct patient decision-making, our Healthcare Navigator combines these disparate lines of inquiry and practice. We operationalize this idea at a national level using sizable, publicly available CMS datasets, building on the fundamental validation of ontology-based provider matching found in the works of Fayçal and Abdelkamel [4] and Sethuraman et al. [11]. Our framework is intended for national use, in contrast to Subbulakshmi et al.'s localized system [3]. Moreover, we go beyond the objectives of large-scale knowledge graphs such as those in Majdalawieh et al. [5] and Shi et al. [6] that are centered on research and clinicians. By doing this, we close the gap between the real-world, everyday needs of patients

and the robust backend systems that are discussed in the industry [9], [10]. Our system fills a crucial gap left by current academic and commercial systems by offering a unique, end-to-end patient care journey with interactive visualization and quality-driven decision support.

## IV. Approach and System Design

Our approach utilizes a "Static, No-ETL" data strategy, treating our knowledge base as an immutable release 1. Data is curated offline into a versioned "Dataset Bundle," which contains the formal ontology and all RDF data, and is stored in AWS S3. Live application servers then perform a one-time seed from this bundle. This design eliminates complex ETL jobs and ensures a stable, repeatable deployment.

*a) **High-Level Architecture**:* The high-level system architecture (HLD) comprises five logical components. S3 (Data Store) stores the immutable, versioned dataset bundles. Ontotext GraphDB (Knowledge Graph) serves as the source of truth; an RDF/OWL 2 database storing the healthnav.owl ontology and all semantic triples for complex graph queries. MongoDB Atlas (Cache) stores denormalized "views" and response caches, loaded once during the seed to ensure "snappy UI loads". FastAPI (Backend) is the orchestration layer, acting as a "SPARQL proxy" to GraphDB, and manages caching, geospatial calculations, and result ranking. Finally, a React (Frontend) client-side application (React, Vite, TypeScript) provides the UI, Mapbox GL JS visualization, and communication with the backend.

The user flow follows this design: A user search on the React frontend calls the FastAPI backend. The API first checks MongoDB for a cached result. On a "miss," it queries GraphDB via SPARQL, computes distances, ranks the results, stores them in the MongoDB cache, and returns the data to the client.

A key principle of this architecture is the strict separation of data. The Knowledge Graph (in GraphDB) is static and contains **no** personal patient information; it is built entirely from public, anonymized provider and facility data. Personal user data, such as an email for login or a saved `InsurancePlan`, is handled separately by the FastAPI application layer and stored in MongoDB for user session management and preferences. A patient's self-reported symptom is used *at query time* to search the knowledge graph but is not ingested or stored within the graph itself.

*b) **Low-Level Design**:* The low-level design (LLD) specifies our data models and service logic.

**1) Data Models:** We employ a dual data model: a normalized graph model (RDF) for semantic integrity and a denormalized document model (JSON) for performance. The GraphDB (RDF Model) defines key classes (Physician, Hospital, Symptom, Precaution) and object properties (affiliatedWith, treatsCondition, hasSymptom) that allow for traversing the complex healthcare graph. The MongoDB (Cache Model) stores pre-built JSON objects to minimize query time. The providers_cache, for instance, stores "one doc per provider-hospital tuple" 2 and contains pre-joined data like physician specialties and hospital HCAHPS scores 3.

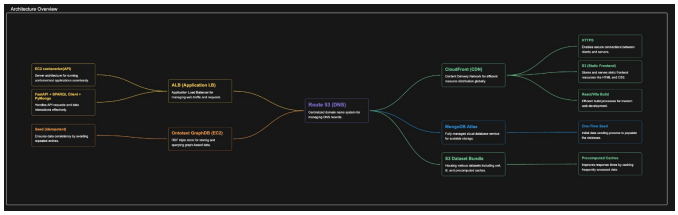


Fig. 1. High Level Architecture

```
@prefix : <http://example.org/healthnav#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema
    #> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
     .

:Physician a owl:Class ; rdfs:subClassOf :Person ,
    [ a owl:Restriction ;
      owl:onProperty :hasSpecialty ;
      owl:someValuesFrom :Specialty ] ,
    [ a owl:Restriction ;
      owl:onProperty :treatsCondition ;
      owl:someValuesFrom :MedicalCondition ] .

:affiliatedWith a owl:ObjectProperty ;
    rdfs:domain :Physician ;
    rdfs:range :Hospital .
:hasSymptom a owl:ObjectProperty ;
    rdfs:domain :MedicalCondition ;
    rdfs:range :Symptom .
:recommendedPrecaution a owl:ObjectProperty ;
    rdfs:domain :Symptom ;
    rdfs:range :Precaution .

:hcahpsOverallScore a owl:DatatypeProperty ;
    rdfs:domain :Hospital ;
    rdfs:range xsd:decimal .
```

**2) Backend (FastAPI):** The backend handles all business logic via key API routes like POST /api/search/symptom 4. The core logic for this endpoint executes a SPARQL query on a cache miss. This query is designed to 5: filter by the input Symptom name 6, traverse the graph from Symptom to MedicalCondition to Physician, and use an OPTIONAL block to retrieve the Physician's affiliated Hospital data, including its hcahps score and location (lat, lon) 7. After retrieving the SPARQL results, the Python service performs two final steps: computing the haversine distance for each provider, and applying a weighted rank function to normalize and combine the HCAHPS score and distance, producing a final user score 8.

**3) Frontend (React):** The frontend uses React Query for server state management, providing a "stale-while-revalidate" cache strategy. The UI features a search form and a synchronized map/list view, where hovering a list item highlights the corresponding map marker. The application state is stored in URL parameters to ensure all search results are shareable links.

| Domain | Property | Range | Description |
|---|---|---|---|
| **Patient** | hasPrimaryPhysician | **Physician** | Links a patient to their primary physician. |
| **Physician** | performsProcedure | **Procedure** | Specifies medical procedures performed by a physician. |
| **Hospital** | affiliatedWith | **Organization** | Shows organizational affiliation (e.g., health system or net work). |
| **Organization** | locatedAt | **Address** | Connects the organization to a postal address. |
| **InsurancePlan** | hasRating | **Rating** | Associates an insurance plan with user or performance rating s. |
| **GeoLocation** | longitude | **decimal** | Defines the longitudinal coordinate of the location. |
| **Address** | addressLine | **string** | Describes the full street address. |
| **MedicalCondition** | hasSymptom | **Symptom** | Connects a medical condition to its symptoms. |
| **Rating** | ratingValue | **decimal** | Numeric score assigned to an entity. |
| **Procedure** | performedBy | **Physician** | Connects a procedure to the physician who performs it. |

Fig. 2. TABLE I: EXAMPLES OF ONTOLOGY PROPERTIES

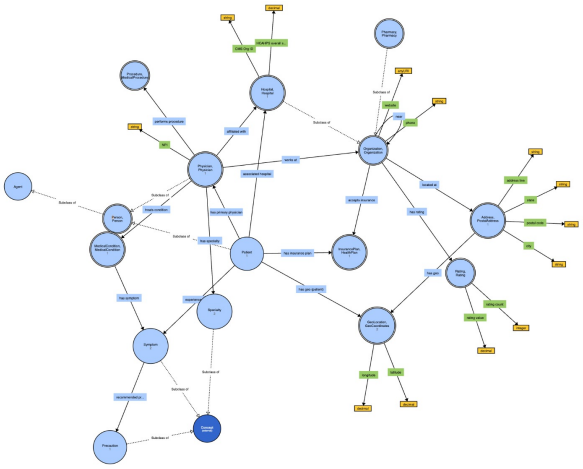

Fig. 3. Visualization of the Healthcare Navigator Ontology, showing the patient-centric model. Core classes are shown as blue circles, data properties as green rectangles, data types as yellow rectangles, and object properties (relationships) as blue rectangles on the edges.

## V. Ontology Design

The Healthcare Navigator Ontology was developed to represent and interlink key entities within the healthcare ecosystem—including patients, physicians, hospitals, organizations, insurance plans, symptoms, and medical conditions. It provides a semantic framework that allows structured reasoning and interoperability between healthcare data systems.

*a)* ***Classes****:* The ontology is structured around several major classes. At the center is the Patient class, representing individuals receiving care. This class is semantically linked to care providers (Physician), institutions (Hospital), and administrative or service-providing entities (Organization, which includes pharmacies) . The model also includes InsurancePlan to cover treatments; GeoLocation and Address for spatial details; Symptom and MedicalCondition for clinical data; Procedure and Prescription for medical activities; and Specialty to represent areas of medical expertise .

*b)* ***Properties****:* Object and data properties connect these classes. A selection of key properties, showing their domain, range, and description, is presented in Table I. These properties, such as hasPrimaryPhysician and hasSymptom, form the semantic links that power the knowledge graph.

*c)* ***Visualization****:* The ontology visualization (Fig. 1) represents this structure, with core classes, data properties, data types, and object properties . This diagram emphasizes a **patient-centric model** where the individual is the core of the healthcare network, linked semantically to care providers (hasPrimaryPhysician), insurance (hasInsurancePlan), location (hasGeoPatient), clinical data (experiencesSymptom), and required expertise (needsSpecialty)

## VI. Data Collection and Processing

1) Our project's architecture is founded on a "Static, No-ETL" data strategy, where the dataset is treated as an immutable, versioned release. This approach was chosen to ensure stability, performance, and repeatable deployments, avoiding the complexity of live ETL pipelines. All data collection and processing are performed once, offline (e.g., in a notebook) to create a "Dataset Bundle" . This bundle contains the final, clean RDF data files in Turtle format, as specified in our architecture , and the pre-built JSON caches .

2) The project integrates several key datasets. For Doctors and Clinicians, we use the national downloadable files from the CMS Provider Data Catalog [12], [13] and the NPPES Data Dissemination service [14] . Integration between these sources is achieved using the National Provider Identifier (NPI) as the common key. From these files, we extract essential provider identity data, including NPI, full name, and specialty (based on NPPES taxonomy codes), as well as physician-to-hospital affiliation mappings. All financial data, such as procedure volume, and the weekly incremental updates are explicitly excluded. This is to maintain a fixed dataset focused on our provider-finding use case, rather than financial analysis or real-time updates. The cleaned data is mapped to our ontology by our offline scripts and exported as `physicians.ttl` and `specialties.ttl`.

3) For Hospitals and Ratings, we use the national hospital-level HCAHPS dataset from CMS [15], [16], which provides patient survey experience scores. The CMS ID (or CCN) is used as the primary key for this data. We include the hospital's name, CMS ID, and address. For our quality rating, we extract only the primary HCAHPS overall score. All detailed sub-metrics (e.g., "nurse communication," "room cleanliness") are excluded to simplify the backend ranking model, which requires a single, normalized score . The addresses are also geocoded to extract latitude and longitude, which is critical for the geospatial query logic. This data is exported as `hospitals.ttl` and `hospitals_hcahps.ttl`.

4) Pharmacy locations are sourced from OpenStreetMap (OSM) [17]. We fetch this data by querying the Overpass API [18], [19], specifically requesting nodes and ways tagged

with `amenity=pharmacy` . We extract only the name and geospatial coordinates. Our processing script converts the API's JSON results into RDF resources matching our ontology and exports them as `pharmacies.ttl`.

5) Finally, the Symptom/Condition Mappings are a manually curated dataset created for this project. This knowledge base provides the semantic "glue" that is not available in structured public data. It links a `Symptom` (e.g., "Chest Pain") to a `MedicalCondition` ("Coronary artery disease") and a `Precaution` ("Call 911"). This data is authored directly as `conditions_symptoms.ttl` and `symptoms_precautions.ttl`.

All processed `.ttl` files are combined with the `healthnav.owl` ontology and stored in the S3 "Dataset Bundle" , which serves as the single source of truth for the `ops/seed.py` script to populate GraphDB.

## VII. Implementation Plan

The implementation adopts a "Static, No-ETL" data strategy, treating the dataset as an immutable, versioned release. The plan comprises three primary components: (A) data model curation, (B) backend infrastructure and service deployment, and (C) frontend user interface.

### A. Data Curation and Seeding

The foundation of the system is a one-time, offline data curation process. Initially, the formal ontology, healthnav.owl, was defined to specify all classes (e.g., Physician, Hospital, Symptom) and object properties (e.g., affiliatedWith, treatsCondition). Subsequently, offline scripts were developed to process raw source data from CMS, HCAHPS, and OpenStreetMap. This process transforms the heterogeneous data into two standardized formats: a set of RDF Turtle (.ttl) files (e.g., physicians.ttl, hospitals.ttl, symptoms_precautions.ttl) corresponding to the ontology, and pre-computed JSON files (e.g., providers_cache.json) for the denormalized MongoDB cache.

These files are bundled into a versioned release and stored in an AWS S3 bucket (e.g., s3://healthnav-static/v1/). The live databases are populated by executing a single ops/seed.py script. This "One-time Seed ETL" loads the healthnav.owl ontology and all .ttl data files into the Ontotext GraphDB repository, while simultaneously populating the MongoDB collections from the JSON cache files. A one-time SHACL validation is performed after seeding to ensure data integrity.

### B. Backend and Infrastructure

The system infrastructure is deployed on AWS. The Ontotext GraphDB instance operates on an EC2 server within a private VPC, while the cache layer utilizes MongoDB Atlas. The backend consists of a FastAPI (Python) application that serves as the orchestration layer. Its primary function is to handle the POST /api/search/symptom request. The logic for this endpoint first checks the MongoDB query_cache for a matching result. If there is a cache miss, the service constructs and executes a SPARQL query against the GraphDB repository. The Python application processes the SPARQL results, calculates geospatial distance using the Haversine formula, and applies a weighting algorithm to rank providers. The final processed result is stored in the Mongo query_cache and returned to the user. This API is containerized and deployed to EC2 behind an Application Load Balancer.

### C. Frontend User Interface

The frontend is implemented as a React application using TypeScript, Vite, and Tailwind CSS. Data fetching is managed by React Query, which handles server state and implements a stale-while-revalidate cache strategy. The user interface consists of a search form and a synchronized map-and-list view, utilizing Mapbox GL JS to display clustered markers.

Application state, including search filters, is stored in the URL parameters to ensure all searches are shareable links. The static application is built and deployed to S3 and served globally via a CloudFront CDN.

## VIII. Roles and Responsibilities

This work is a collaborative effort, with tasks distributed among the team members based on the core components of the architecture. The team's roles are logically divided into four main areas to align with our implementation plan.

The **Data and Ontology** role is responsible for the foundational data engineering. This includes the one-time, offline curation of source data (CMS, HCAHPS, OSM) into clean RDF `.ttl` files and denormalized JSON caches . This role also defines the formal `healthnav.owl` ontology and develops the `ops/seed.py` script to load the entire dataset bundle into the live databases .

The **Backend** role focuses on developing the FastAPI application to serve as the orchestration layer. This involves implementing the SPARQL query logic to interface with GraphDB , managing the MongoDB cache , and performing real-time processing, such as Haversine distance calculations and result ranking .

The **Frontend** role builds the entire client-side React application using TypeScript and the `shadcn/ui` component library. This includes managing server state with React Query and building the interactive Mapbox GL JS visualization .

Finally, the **Infrastructure and Ops** role manages the deployment of all cloud services. This includes deploying the Ontotext GraphDB and FastAPI application on EC2, hosting the static frontend on S3/CloudFront , and configuring MongoDB Atlas and CloudWatch for monitoring .

A detailed breakdown of individual contributions and task assignments for each team member is provided in Table II.

## References


[1] B. Abu-Salih, M. AL-Qurishi, M. Alweshah, M. AL-Smadi, R. Alfayez, and H. Saadeh, "Healthcare knowledge graph construction: A systematic review of the state-of-the-art, open issues, and opportunities," *Journal of Big Data*, vol. 10, no. 1, p. 81, 2023.

[2] A. Masood, "The role of knowledge graphs in healthcare," https://medium.com/@adnanmasood/the-role-of-knowledge-graphs-in-healthcare-7109a8c33122, 2021, accessed: 2025-10-26.

| S.No | Team Members | Roles | Resposnibility |
|---|---|---|---|
| 1 | Rahul Marathervar | Frontend, Infrastructure and Ops | 1.Builds the complete client-side application using React, Vite, and TypeScript, including the Mapbox GL JS interface and React Query for data fetching .<br>2.Manages the frontend's cloud deployment, which involves hosting the static build on AWS S3 and configuring CloudFront for global CDN delivery . |
| 2 | Balaji Radhakrishnan Padmanabhan | Backend, Data and Ontology | 1.Develops the FastAPI (Python) server, including implementing all API routes (e.g., POST /api/search/symptom) and the core business logic (Haversine distance, ranking) .<br>2.Defines the formal healthnav.owl ontology and performs the one-time, offline curation of raw CMS, HCAHPS, and OSM data into the clean RDF .ttl files (e.g., physicians.ttl) . |
| 3 | Akshat Aggarwal | Backend, Data and Ontology | 1.Develops the FastAPI (Python) server, including implementing all API routes (e.g., POST /api/search/symptom) and the core business logic (Haversine distance, ranking) .<br>2.Defines the formal healthnav.owl ontology and performs the one-time, offline curation of raw CMS, HCAHPS, and OSM data into the clean RDF .ttl files (e.g., physicians.ttl) . |
| 4 | Yogesh Sangtani | Data and Ontology, Infrastructure and Ops | 1.Generates the pre-built, denormalized mongo/*.json cache files (e.g., providers_cache.json) for MongoDB and writes the ops/seed.py script that loads the entire "Dataset Bundle" into the live databases .<br>2.Deploys and manages all backend infrastructure, including the Ontotext GraphDB on EC2, the MongoDB Atlas cluster, the FastAPI application, and CloudWatch monitoring . |
| 5 | Charani Tirumalareddy | Data and Ontology, Infrastructure and Ops | 1.Generates the pre-built, denormalized mongo/*.json cache files (e.g., providers_cache.json) for MongoDB and writes the ops/seed.py script that loads the entire "Dataset Bundle" into the live databases .<br>2.Deploys and manages all backend infrastructure, including the Ontotext GraphDB on EC2, the MongoDB Atlas cluster, the FastAPI application, and CloudWatch monitoring . |

Fig. 4. TABLE II: TEAM ROLES AND RESPONSIBILITIES


[3] M. Subbulakshmi, K. Krishnan, and S. Sreereshmi, "Contextual aware dynamic healthcare service composition based on semantic web ontology," *IEEE Access*, vol. 8, pp. 218 652–218 663, 2020.

[4] Z. Fayçal and T. Abdelkamel, "Building a semantic web services ontology in the pharmaceutical field using the owl-s language," in *2021 International Conference on Information Systems and Advanced Technologies (ICISAT)*. IEEE, 2021, pp. 1–8.

[5] S. Majdalawieh, A. A. Masri, and M. Moh'd, "Semantic knowledge graph framework for knowledge discovery and acquisition in public health systems," *IEEE Access*, vol. 12, pp. 44 585–44 599, 2024.

[6] L. Shi, S. Li, X. Yang, J. Qi, G. Pan, and B. Zhou, "Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services," *BioMed Research International*, vol. 2017, p. 2858423, 2017.

[7] Ontotext, "Semantic technologies and knowledge graphs in healthcare," https://www.ontotext.com/blog/semantic-technologies-and-knowledge-graphs-in-healthcare/, 2024, accessed: 2025-10-26.

[8] S. Schifman, "Rag on fhir with knowledge graphs," https://medium.com/@samschifman/rag-on-fhir-with-knowledge-graphs-04d8e13ee96e, 2024, accessed: 2025-10-26.

[9] Tenasol, "Fhir & healthcare graph databases," https://www.tenasol.com/blog/hl7-fhir-graph-database, 2024, accessed: 2025-10-26.

[10] Amazon Web Services, "Analyze healthcare fhir data with amazon neptune," https://aws.amazon.com/blogs/database/analyze-healthcare-fhir-data-with-amazon-neptune, 2023, accessed: 2025-10-26.

[11] R. Sethuraman, G. Sneha, and D. S. Bhargavi, "A semantic web services for medical analysis in health care domain," in *2017 International Conference on Information Communication and Embedded Systems (ICICES)*, 2017, pp. 1–5.

[12] Centers for Medicare & Medicaid Services, "Provider data catalog: Doctors & clinicians," https://data.cms.gov/provider-data/topics/doctors-clinicians, 2025, accessed: 2025-11-10.

[13] ——, "Provider data catalog: National downloadable file (doctors)," https://data.cms.gov/provider-data/dataset/mj5m-pzi6, 2025, accessed: 2025-11-10.

[14] ——, "Nppes: Npi data dissemination files," https://download.cms.gov/nppes/NPI_Files.html, 2025, accessed: 2025-11-10.

[15] ——, "Provider data catalog: Hcahps - hospital," https://data.cms.gov/provider-data/dataset/dgck-syfz, 2025, accessed: 2025-11-10.

[16] ——, "Cahps (hcahps) program," https://www.cms.gov/data-research/research/consumer-assessment-healthcare-providers-systems/hospital-cahps-hcahps, 2025, accessed: 2025-11-10.

[17] OpenStreetMap Wiki, "Tag:amenity=pharmacy," https://wiki.openstreetmap.org/wiki/Tag%3Aamenity%3Dpharmacy, 2025, accessed: 2025-11-10.

[18] ——, "Overpass api by example," https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_API_by_Example, 2025, accessed: 2025-11-10.

[19] ——, "Overpass ql," https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL, 2025, accessed: 2025-11-10.