Communication Strategy Memo

Team Members:

Ian Nieto

John Gornick

Jerry “Tre” Kelley

Jasmine Flowers

Client: Jesslynn Stull

Mentor: Bailey Hall

Table of Contents

Introduction

The United States healthcare system is one of the largest and most critical sectors of the national economy, representing more than $4 trillion in annual expenditures. It is a system filled with multiple providers, insurers, and patients, and one in which pricing for the services provided is usually not very straightforward or consistent. It is based on this vast and complex industry that an important disconnection between the consumers of care and the cost of that care exists. Too often, patients are asked to make major healthcare decisions without access to what many consider a basic consumer right: knowing the price of a service before the service is performed. The result is a process through which individuals receive care only to be confronted with confusing and unexpected bills months later, with little recourse. This lack of transparency results in widespread confusion and surprise financial burdens, and it contributes to an overall erosion of trust in the healthcare ecosystem.

Our project, undertaken in partnership with our client, who brings deep personal conviction to this problem, addresses this critical market failure head-on. Though new to entrepreneurship in the healthcare technology sector, she has first-hand experience with pricing opacity and has thus embarked on a highly dedicated mission to create a solution. This application provides an easy-to-access source that patients can use to compare procedure costs and insurance copays among different providers, prioritizing the Arizona market, so any launch will be feasible and controlled. By transforming opaque and fragmented data into clear, actionable information, we aim to reduce patient stress and financial hardship, eventually fostering a more informed and equitable health marketplace.

At its core, the application must satisfy several key user-level requirements: patients need the ability to search for specific medical procedures, compare costs across multiple providers within their geographic area, and understand their anticipated out-of-pocket expenses based on their insurance coverage. From these user needs emerge a set of derived functional and performance requirements that shape the technical foundation of the solution. The system must integrate and normalize pricing data from disparate healthcare provider sources, maintain an up-to-date database of insurance plan structures and copayment rules, and deliver search results with sufficient speed to support real-time decision-making. Performance requirements center on scalability to handle concurrent user queries, data refresh cycles that ensure pricing accuracy, and response times that meet user expectations for modern web applications. Environmental constraints further define the project scope: the initial deployment targets the Arizona healthcare market exclusively, the solution must operate within the regulatory framework governing healthcare price transparency, and the architecture must accommodate future expansion to additional states and insurance networks without fundamental redesign. Together, these elements present the essential requirements context in condensed form.

Implementation Overview

We’re developing a web-based system to make sense of the craziness that exists in healthcare pricing structures, combining provider pricing, insurance types, and patient experiences in one simple and transparent online destination. While the overall vision for what we’re building is pretty lofty, we’re being pragmatic about how we get there by starting with Arizona-first: in other words, we’re focusing on Arizona as our starting point to make sure we can manage scope

while working to solidify what this core technology will be and how it will function in the future, when we add things like medical bill analysis and real customer reviews. We are currently working in two-week sprints to ensure there is progress on a weekly basis. We plan to be market-ready by the end of May 2026.

Essentially, it is based on the idea of separation of concerns, ensuring that the user interface is simple and fast, even as we add more information. By using this approach, the design is structured so that the front and back ends work well together via the API, we can keep our database clean and secure and grow this entire design without ever having to recreate anything.

When looking at the front end, we decided to go with React because it will allow us to build a dynamic, single-page app that feels responsive. There should be no need for page reloads when they're filtering procedures or looking at personalized cost estimates due to our data taking too long to load. We'll use a library to help us out with styling and other components; it'll keep things nice and clean without having to define all that on our own. Whereas, when looking at the back end, the server side will run on Node.js, which is a solid, widely used environment that’s great for getting an API up and running quickly. This layer handles all the business logic: authenticating users, processing requests, crunching data, and serving up everything the front-end needs.

An important section to emphasize would be the database. We are going to store everything in PostgreSQL. Prices that will be displayed include: procedure price, insurance plan details, providers, cash costs, and user-submitted experiences. Since PostgreSQL is extremely reliable, it manages relationships well and provides strong data integrity, which is important when working with healthcare cost information. Lastly, although we're starting with core features, we are leaving the door open for expansion. The bill analysis tool, for example, will

later be built out as a separate microservice, perhaps in Python. The modular setup means we can add in services like that without breaking what's working or interfering with the current code.

Architectural Overview

The system follows a clean separation between the frontend and backend. The frontend is a Next.js React application hosted on Azure Static Web Apps. It handles the user interface, state management, and client-side routing. The backend runs as a separate Azure Functions API that handles search logic, AI processing, and database queries. This split keeps the frontend lightweight and allows each part to scale independently.

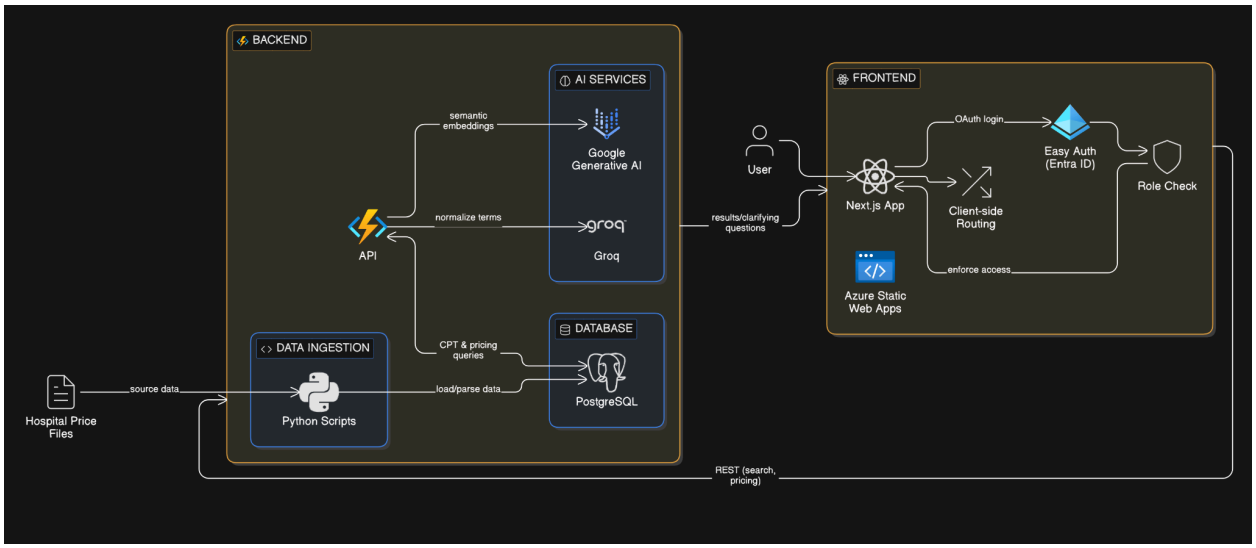
When a user searches for something like "knee surgery," the frontend sends the query to the search API. The backend uses AI services to understand what the user means and map the request to the correct CPT code. We use Google's Generative AI to create semantic embeddings and Groq for chat completions that normalize medical terms. If the system is unsure, it returns clarifying questions instead of guessing. Once a CPT code is identified, pricing data is fetched from PostgreSQL and returned to the frontend.

The database stores procedure codes, clinic information, insurance plans, negotiated rates, and cash prices. PostgreSQL was chosen because it handles relationships well and maintains data integrity, which is important when working with healthcare pricing. The data comes from hospital price transparency files. Python scripts in the altored-parsers repository collect this data, parse it into a clean format, and load it into the database.

Authentication is handled by Azure Static Web Apps using Easy Auth with Microsoft Entra External ID. Users sign in through a standard OAuth flow, and identity claims are mapped

to roles like “whitelisted” or “admin” using a custom auth function. This removes the need to manage sessions or tokens manually. The frontend checks role access and redirects users when needed.

The frontend communicates with the backend through simple REST calls. The search endpoint accepts a POST request and returns either a CPT match, clarifying questions, or no results. A second endpoint returns pricing details for a specific CPT code. Both endpoints are stateless, keeping the backend simple and scalable.



Component-Level Design

Backend Data Collector

The backend data collector will collect data from hospital information databases and will periodically update the system’s backend on the changes to keep all data up to date with hospital information.

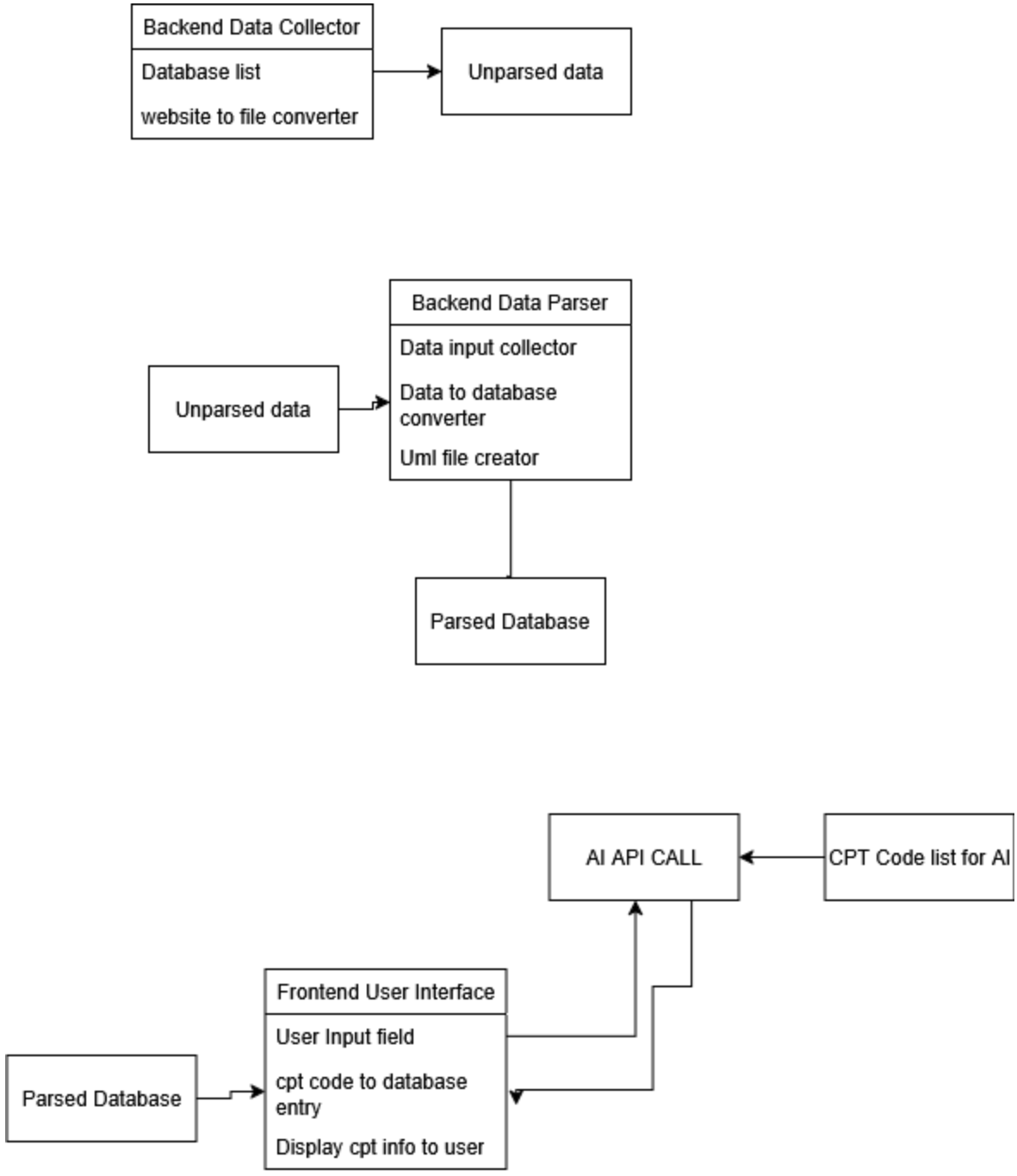
Backend Data Parser

The backend data parser will use the data collected by the backend data collector. It will parse the data into a minimalist sorted dataset for the Frontend user interface to connect to.

Frontend User Interface

The frontend user interface is the portion of the product that user's will interact with directly. It will take user input and reference the database in order to provide the user with healthcare information. The frontend user interface may connect to ai backends to find suggested cpt codes based on user input.

4.2 - UML Diagram



Public interface

The system's public interface will have a main purpose to show users information about healthcare services and the pricing of those services at specific healthcare facilities. It should be as simple as asking for an issue and getting an answer relevant to the problem.

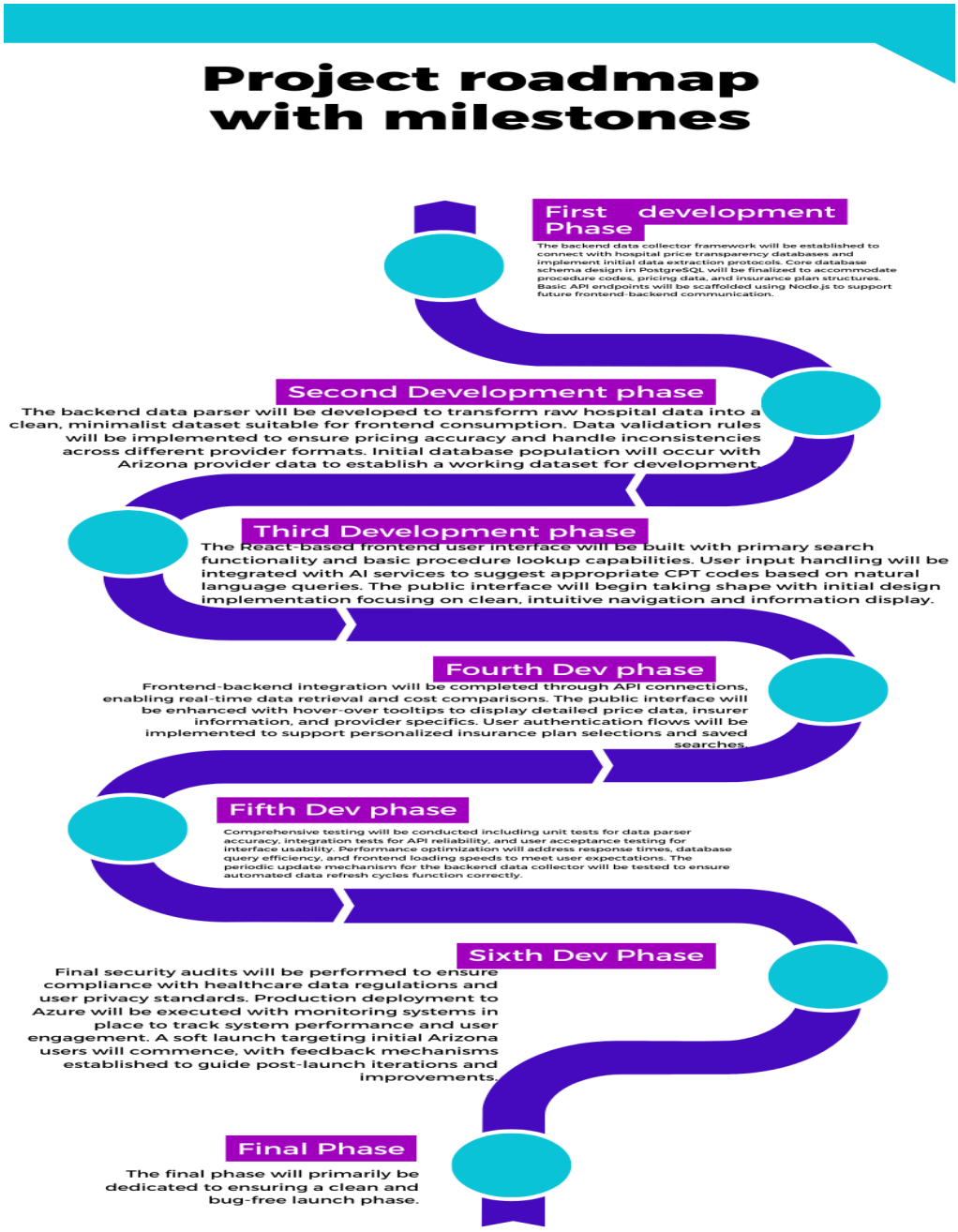
The user interface will be very informative with a decent amount of hover options to show users information such as the price data and insurer information.

Implementation Plan

Our project will be split into six phases. Each phase will represent two weeks of development. All of this work allows our developers to work in tandem and on separate issues, all the while developing each part of the project. It's important to note that this version of our roadmap is not absolute. This is a perfect world situation in which no problems occur during development. The likelihood is that this will look significantly different, as everyone on the team takes different coding tasks as they're capable of. This is why our development is designed this way. Every important task has been given a certain weight, which will carry into our workflow. If every task is completed, we will have a functional product, no matter the order in which things are completed. Our biggest constraint and dependency is our ability to have a working data parser.

If we do not have a working data parser, frontend work will have a much harder time being completed without functional sample data. Beyond the data parser dependency, several additional technical risks warrant acknowledgment. Hospital price transparency files vary significantly in format and structure across providers, which may require provider-specific

parsing logic rather than a generic solution. This trade-off prioritizes faster initial development over long-term maintainability, though we're mitigating risk by starting with a limited subset of Arizona providers to establish baseline patterns. Additionally, our reliance on external AI services for CPT code suggestions introduces availability concerns. Scalability constraints present another consideration as the system grows beyond Arizona. While PostgreSQL handles our current scope effectively, nationwide expansion may demand query optimization, enhanced indexing strategies, or architectural adjustments like read replicas. Our authentication approach through Azure Easy Auth reduces development complexity but creates vendor dependency that could limit future deployment flexibility.


Project roadmap with milestones
First development Phase
The backend data collector framework will be established to connect with hospital price transparency databases and implement initial data extraction protocols. Core database schema design in PostgreSQL will be finalized to accommodate procedure codes, pricing data, and insurance plan structures. Basic API endpoints will be scaffolded using Node.js to support future frontend-backend communication.
Second Development phase
The backend data parser will be developed to transform raw hospital data into a clean, minimalist dataset suitable for frontend consumption. Data validation rules will be implemented to ensure pricing accuracy and handle inconsistencies across different provider formats. Initial database population will occur with Arizona provider data to establish a working dataset for development.
Third Development phase
The React-based frontend user interface will be built with primary search functionality and basic procedure lookup capabilities. User input handling will be integrated with AI services to suggest appropriate CPT codes based on natural language queries. The public interface will begin taking shape with initial design implementation focusing on clean, intuitive navigation and information display.
Fourth Dev phase
Frontend-backend integration will be completed through API connections, enabling real-time data retrieval and cost comparisons. The public interface will be enhanced with hover-over tooltips to display detailed price data, insurer information, and provider specifics. User authentication flows will be implemented to support personalized insurance plan selections and saved searches.
Fifth Dev phase
Comprehensive testing will be conducted including unit tests for data parser accuracy, integration tests for API reliability, and user acceptance testing for interface usability. Performance optimization will address response times, database query efficiency, and frontend loading speeds to meet user expectations. The periodic update mechanism for the backend data collector will be tested to ensure automated data refresh cycles function correctly.
Sixth Dev Phase
Final security audits will be performed to ensure compliance with healthcare data regulations and user privacy standards. Production deployment to Azure will be executed with monitoring systems in place to track system performance and user engagement. A soft launch targeting initial Arizona users will commence, with feedback mechanisms established to guide post-launch iterations and improvements.
Final Phase
The final phase will primarily be dedicated to ensuring a clean and bug-free launch phase.

Conclusion

The healthcare pricing transparency problem represents more than a technical challenge, as it is a fundamental issue of consumer rights and market equity that affects millions of Americans navigating an increasingly complex and costly system. This document has presented a comprehensive design for a web-based solution that addresses this critical gap through a carefully architected system combining modern frontend technologies, robust backend processing, and intelligent data management. The design prioritizes separation of concerns through a React-based user interface, Node.js API layer, and PostgreSQL database foundation, ensuring that each component can evolve independently while maintaining system cohesion and reliability.

The architectural decisions outlined in this document reflect a pragmatic approach to building a scalable, maintainable solution within real-world constraints. By focusing initially on the Arizona market, implementing AI-powered search capabilities for procedure identification, and establishing clear data collection and parsing pipelines, the design supports incremental delivery while maintaining flexibility for future expansion. The phased implementation plan acknowledges both the technical dependencies that must be resolved early, particularly around data parsing, and the risks inherent in integrating disparate healthcare data sources, external AI services, and cloud infrastructure. These considerations inform a development approach that balances immediate functionality against long-term sustainability.

Ultimately, the proposed design supports successful implementation and project outcomes by providing clear technical direction, identifying critical dependencies, and establishing a foundation that can grow with user needs and market demands. It is through this structured yet adaptable approach that we can deliver on our client's vision: a tool that empowers patients with the pricing information they need to make informed healthcare decisions, reduces

financial anxiety and surprise billing, and contributes to a more transparent and equitable healthcare marketplace. The technical framework presented here serves not as an end in itself, but as the means to achieve a broader social good: restoring trust and clarity to one of the most important and personal decisions individuals face.