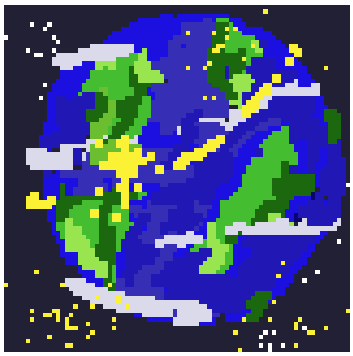
Technological Feasibility

Final Copy



Date: September 19th, 2024

Team name: HelloWorldByMe

Sponsored By: Kevin Daily

Faculty Mentor: Brian Donnelly

Team Members:

Elizabeth Knight

Joey Banaszak

Jessica Maldonado Olivas

Samantha Madderom

Table of Contents

Introduction

Chronic homelessness is a significant issue, particularly in Tucson, Arizona. Between 2019 and 2020, there was a 300% increase in homelessness in Pima County. While many nonprofits offer services to help those in need, the lack of communication between service providers limits their overall impact. To bridge this gap, nonprofits employ "Navigators" who go out into the community, gather information on individuals experiencing homelessness due to drug addiction and mental health issues, and connect them with available services.

Our project aims to streamline this process by providing a system that allows Navigators to update and access a shared database of the people they interact with. This ensures that, if someone has been previously engaged, their information is readily available to assist in future interactions. Additionally, shelters will be able to update the database with real-time information on available beds, enabling Navigators to secure spots in advance for individuals in need. Shelters can also log details about the services they offer, such as support for overcoming addiction or mental health care. Available services and collected demographic information on those in need will not be segregated by organization. Instead, all organizations will be connected and sharing information. In this way, we hope to break down some silos of communication.

Moreover, the data collected will be shared with local government agencies to contribute to existing research and secure funding. By fostering better communication among service providers and supporting data-driven decision-making, we hope this project will lead to improved outcomes for both the organizations and the people they serve.

Several major problems that need to be solved to ensure the success of this project are cybersecurity, compatibility across platforms, and a messaging system that allows Navigators to communicate out on the field. For cybersecurity, we want to make sure that everyone's data is secure. This applies to the data collected on people the Navigators speak to, information the non-profits upload about their services, and any messages Navigators send. We also want to support compatibility across platforms so that the data is more easily accessible on various devices. Finally, we need a messaging system to allow people to contact each other through WorldByMe using their roles. They should be able to search people and send messages based on role and name, as well as actual contact.

Component Diagram

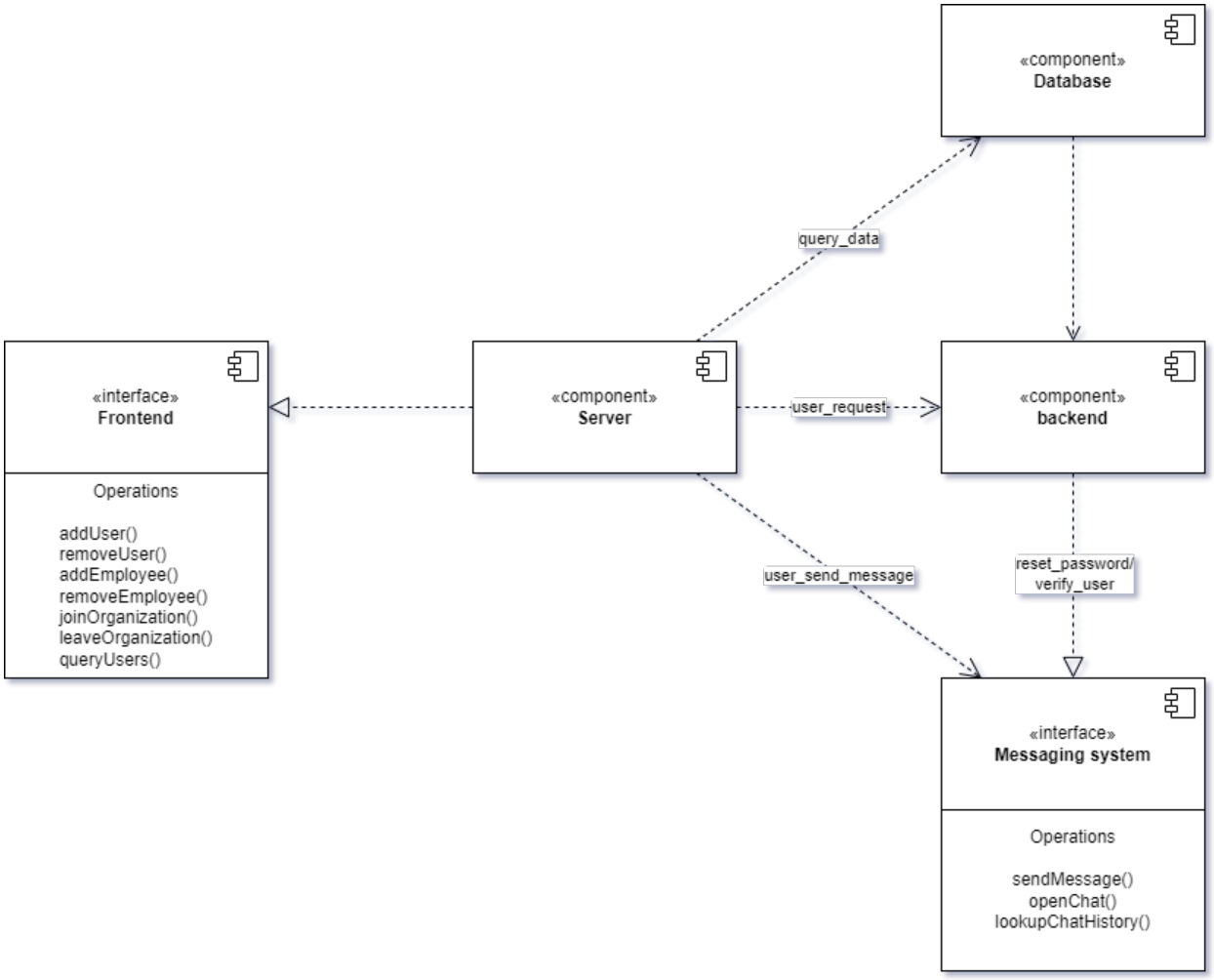


Figure 1: Component Diagram of Current Infrastructure

Technological Challenges

This section will outline the major technical challenges we expect to encounter when implementing the many aspects of this project. As a team, we have identified six core challenges, all varying from the infrastructure to the front-end features that could affect the overall architecture of a role-based system. By addressing these challenges, the team will have opportunities to envision solutions to ensure that any of the following concerns can be resolved.

One major challenge is ensuring that the platform that we create supports varying devices without compromising functionality. Though wrappers and bridges are known to be a good place to start when addressing this issue, it could lead to performance difficulties between the original system and the cross-platform layers. Ensuring that devices are compatible across different operating systems is also important, as it could create limitations within various user experiences. Therefore the system and interface must be responsive and adaptable to different screen sizes.

Since the main feature of the new role-based system is communication, that implies the usage of messages and notifications. The system will support real-time messaging, emojis, and multimedia embedded within the messages. A key aspect of the messaging system is that it needs to perform efficiently under high usage. Involving communication protocols will be essential when implementing this subsystem, however, more complexity will be added when the feature of embedded images and files is involved. Security and storage measures must also be taken into account when thinking about adding a multimedia ability.

Along with messages, emails also need to be included when creating the messaging system. Though the messages and emails fall under the same scope, implementing the email system has its own set of challenges. Being able to incorporate reliable delivery, manage message queues, and integrate the API that the team has to design points to a crucial issue that will inevitably be encountered. The messaging system that includes emails, messages, and embedded multimedia will have to be designed in a way that can be scalable as the usage of the whole platform grows and ensures that the users will receive notifications and alerts.

Moving away from the subsystems and looking more at the big overall picture, designing and implementing a secure but flexible role-based access control system is essential in this project. The biggest challenge is finding a way to support intertwined permission hierarchies while maintaining scalability and ease of communication between users. The system shouldn't be too complicated in the user's eyes since it could make the usage of the platform not engaging, however, security protocols also need to be in place. Security is important when managing each role's sensitive data and handling communication channels.

One of the project's key aspects is effective security and to do that data needs to be managed by secure entities. Managing each user's sensitive data and communication is doable through secure data storage, protective user privacy protocols, and defenses against potential cyber-attacks. Currently, the lack of knowledge within the team concerning cybersecurity will serve as a big technical challenge. Therefore, implementing

cybersecurity is one of the most important but also the where the most work needs to be done.

Technological Analysis

As previously mentioned, the privatized silos within local governments create strain for service providers when combating community issues like homelessness. By developing a new system that will revolutionize how citizens receive help and services, features such as real-time messaging, role-based access control, and cross-platform compatibility are needed. However, for these features to be implemented successfully without compromising scalability, performance, and security, current technological challenges will be addressed. This section will go more in-depth about ideal solutions and any alternatives that were found to create a more robust and efficient system.

User Interface

To build an application that seamlessly operates across both web and mobile interfaces, selecting the right UI option is crucial for delivering an optimal user experience. We need a scalable and accessible solution that ensures consistent performance across platforms. The chosen UI must not only be visually appealing but also fully functional, accommodating the diverse needs of our users. The flexibility and functionality of our user interface system is crucial for the success of our application as it ensures we can meet user expectations and maintain a smooth experience regardless of the device being used based on the correct frameworks chosen.

UI Foundation and Frameworks

It was imperative that we had a reliable and cross-platform application to display information in an easy-to-navigate and organized manner. The options we explored highlighted the need for a front-end system that not only created a simple and effective user interface but also provided a great overall user experience. A well-designed interface is crucial for ensuring that users can effortlessly access the information they need. A thorough analysis of the frameworks required for both client-side and server-side components, in addition to the foundational presentation layer, is essential to determine the best options for an optimal user experience.

Presentation Layer

HTML5

HTML5, also known as HyperText Markup Language, serves as the backbone of web content (Dev, 2024)[1]. There is a lot of information and data needed to be accessed through this skeleton, so having an easy to use and interactive framework was the most ideal choice in our opinion. We ultimately decided that the user interface through the application should carry its structure with HTML5. HTML5 showed to be one of the best options, and one of the only options to create the basic structure of our website.

CSS

CSS, or Cascading Style Sheets, allows the website to be customized to what we see fit as the best user interface. By using CSS, we can separate the skeleton from the design for editing purposes, making it easier to update styles without altering the underlying HTML structure, which enhances maintainability and flexibility (Electronic Accessibility, n.d.)[2]. With CSS, we can implement responsive design techniques, ensuring that our website looks great on a variety of devices and screen sizes. This approach not only saves time during development but also enhances the overall user experience by delivering a visually appealing and functional interface.

Bootstrap

Just using HTML and CSS would not allow for compatibility across devices, as traditional styling often results in fixed layouts that do not adjust well to different screen sizes. This is where Bootstrap comes in, offering a responsive grid system that automatically adapts layouts to various devices. Bootstraps responsive approach not only saves development time but also enhances user experience by providing a consistent interface, regardless of the device being used (Ouellette, 2023)[3].

Javascript

In order to make the website not only have a solid structure and a pleasant user experience but also a working aspect to it, JavaScript was considered to add functionalities to the structure of the website. Since the main emphasis on this system is through a web application, javascript is one of the only frontend languages that is supported in most browsers which would further add to the maintainability and cross platform compatibility aspect that we are looking for in our application (MozDevNet,

---

[1] Joswellahwasike. (2024). The Role of HTML in Web Development. Retrieved from https://dev.to/joswellahwasike/the-role-of-html-in-web-development-3k0k

[2] Electronic Accessibility. (n.d.). Retrieved from https://www.ucop.edu/electronic-accessibility/web-developers/advanced-tips/use-cascading-style-sheets-css-to-present-page-content.html#:~:text=CSS%20gives%20developers%20the%20ability,screen%20readers%20access%20the%20content.

[3] Ouellette, A., Alexandre Ouellette Writer for The CareerFoundry BlogAlexandre Ouellette is a Backend Developer for Petal Health in Ottowa, Alexandre Ouellette Writer for The CareerFoundry Blog, Ouellette, A., Blog, W. for T. C., & Alexandre Ouellette is a Backend Developer for Petal Health in Ottowa. (2023). What is Bootstrap? An Awesome Beginner's Guide. Retrieved from https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/

n.d.)[4]. Not only does JavaScript allow for this compatibility across devices, but it also is known for being able to handle intensive jobs and client-side requests in a very efficient manner. This versatility, combined with its widespread support across browsers, makes JavaScript an essential component for our web application.

Presentational Layer Implementation

The presentation layer will rely on HTML, CSS, Bootstrap, and Javascript as the technologies for our website structure and styling. According to an October 2024 survey by W3Techs, CSS is used in the styling of 96% of all websites, with 77% of those sites utilizing Bootstrap to enhance responsive design (W3Techs, 2024)[5]. On top of that, JavaScript was also reported to be used on 98% of websites (W3Techs, 2024)[6]. With these statistics in mind, implementing HTML, CSS, Bootstrap, and JavaScript ensures our web application is both responsive and engaging for users.

The presentation layer for the frontend system was selected based on their widespread adoption among websites and their ease of integration. However, more customizable frameworks were also explored to address the more demanding aspects of the web application. To ensure optimal functionality, each potential technology was evaluated based on the specific requirements needed to be implemented on our end.

Key aspects we considered for frontend implementation included:

- Scalability: The website's framework should be able to handle large amounts of traffic without any issues.
- Concurrency: The website should be able to handle multiple processes at the same time to provide a more efficient way to handle specific requests.
- Real-Time Updates: The website framework should be able to update in a dynamic fashion without any issues.
- Flexibility: The website framework should be flexible when it comes to handling updates and new data implementations.
- Ease of Use: The framework should be a relatively easy implementation for efficient development
- Cross-Platform: The framework should be able to be used across different devices, browsers and mobile applications for the best user experience.

---

[4] MozDevNet. (n.d.). What is JavaScript? - Learn web development: MDN. Retrieved from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[5] Usage statistics of CSS for websites. (n.d.). Retrieved from https://w3techs.com/technologies/details/ce-css

[6] Usage statistics of JavaScript as client-side programming language on websites. (n.d.). Retrieved from https://w3techs.com/technologies/details/cp-javascript

Client Side Possible Frameworks

React

React is a Javascript library that is used to create dynamic and responsive user interfaces. The reason why we decided to potentially use React as an additional resource is because React allows us to create components that aid in both the server and client aspects of the website. These components can be reused throughout the interface which allow for efficiency and maintainability across sites which is what we are looking for. React is able to simplify these components to return the specific HTML snippet we need in a syntax called JSX. React also uses dynamic edits of the document for improved performance during development (Martinez, 2024).[7]

Angular

Angular is considered a full-fledged framework, meaning that it has all the applicable needs like routing, data handling, user authentication, and much more that would most of the time need external libraries to complete. Though this sounds like an ideal solutions since it would reduce the load with any external libraries needed to be imported onto the web application, Angular is only used for single-page applications which would create an issue with both scalability and compatibility with some devices, which would take away from the cross-platform aspect we are looking for. Angular also has an extremely steep learning curve, which may take away from the ease of use that we want (Neagoie, 2018)[8].

Vue

Vue.js is considered a progressive framework ideal for creating scalable, interactive user interfaces. It is often simpler to integrate into existing projects than Angular or React, thanks to its flexible and minimal setup requirements. However, the scope of a project like WorldByMe, which likely requires a high performance framework, might be better served by a more powerful framework. While Vue is great in lightweight applications, it may not offer the same level of support for large, data-intensive applications as Angular or React do. Vue may also limit the cross-platform feature we are looking for in the front end due to its limited features (Joshi, 2023)[9]

Client Side Frameworks	Scalability	Concurrency	Security	Real-Time Updates	Flexibility	Cross Platform	Ease Of Use

[7] Martínez, A. (n.d.). Vue vs React vs Angular: A Comprehensive Comparison. Retrieved from https://www.paralleldevs.com/blog/vue-vs-react-vs-angular-comprehensive-comparison/

[8] Angular vs React vs Vue: The Best Framework for 2024 is... (n.d.). Retrieved from https://zerotomastery.io/blog/angular-vs-react-vs-vue/

[9] Angular vs React vs Vue: Core Differences. (2023). Retrieved from https://www.browserstack.com/guide/angular-vs-react-vs-vue

React	✔	✔	✔	✔	✔	✔	✔
Angular		✔	✔	✔	✔	✔	
Vue	✔		✔				✔

Table 1: Client Side Frameworks

Chosen Implementation

Based on the consideration above, React is a top choice over Vue and Angular for its flexibility, performance, and overall ease of use. Unlike Angular, which has a steeper learning curve, React is lightweight and component-based, allowing for greater customization. Vue offers simplicity and a similar component model, but React's widespread use means there's a larger variety of resources, making it an ideal framework for our web application.

Server Side Possible Frameworks

Django

Since Python was decided to be the main language that we would dedicate our server to, Django was the best choice to look at when it came to full stack development. Django, a high level Python web framework, comes with many features included that would normally be implemented as a whole separate aspect on a project, like admin interfaces, authentication, and caching features for example (Mantosh, 2023)[10]. Not only this, the application that we are creating is working hand in hand with extensive data and requires a database to manage all the information at once. Django provides a powerful ORM, or Object-Relational Mapping, that makes our server's interaction with our database to be a simple and seamless experience for us (Sakib, 2024)[11].

Flask

Flask is another Python web framework that we were considering. This is more known to be a micro web framework and not a complete full-stack operation like its counterpart Django. With this, Flask is a more lightweight framework that is a bit more flexible personalized web development. Though this option may seem to be great with its minimalist design and its lack of full-fledged framework, it does not have the necessary components to support our specific needs with our application. Our application will contain a more complex structure that may require a framework that has all the necessary components that would comply with the application specifications.

---

[10] Django vs FastAPI: Which is the Best Python Web Framework?: The PyCharm Blog. (2024). Retrieved from https://blog.jetbrains.com/pycharm/2023/12/django-vs-fastapi-which-is-the-best-python-web-framework/

[11] FastAPI vs Flask vs Django: Which Framework to Choose? (2024). Retrieved from https://www.developerchronicles.com/fastapi-vs-flask-vs-django-which-framework-to-choose

FastAPI

FastAPI is the last python framework that is known for building data heavy and high performance API's. FastAPI is also known for its asynchronous programming which is one of the reasons it makes it able to handle multiple requests simultaneously. Though this framework has a main emphasis on creating API's, it is also able to support HTML pages while also including form submissions, authentication, and more server side aspects (Paty, 2024)[12]. There are some limitations with using FastAPI as its main implementation is not web applications, but it was still considered based on its abilities.

Server Side Frameworks	Scalability	Concurrency	Security	Real-Time Updates	Flexibility	Cross Platform	Ease Of Use
Django	✔	✔	✔	✔	✔	✔	✔
Flask	✔	✔	✔				✔
FastAPI	✔	✔		✔			

Table 2: Server Side Frameworks

Chosen Implementation

Based on the considerations above, Django is the best option for our main server-side framework due to its scalability and flexibility. Unlike Flask, which is a smaller framework, and FastAPI, which focuses on high-speed APIs, Django provides a better all around option. These include ORM and user authentication, which enhance development, speed, and security. Django also supports real-time updates through Django Channels. Its cross-platform support and ease of use make Django a solid choice for creating our application. **Image 1** is able to show how the presentation layer and frameworks will work together throughout the website. The HTML skeleton and CSS/Bootstrap styling is able to give an easy-to-navigate interface and the Javascript, React, and Django implementation are able to work with the input fields to process whatever the user submits on the interface.

[12] Paty. (2024). FastAPI vs Flask: The Ultimate Comparison. Retrieved from https://www.clickittech.com/developer/fastapi-vs-flask/

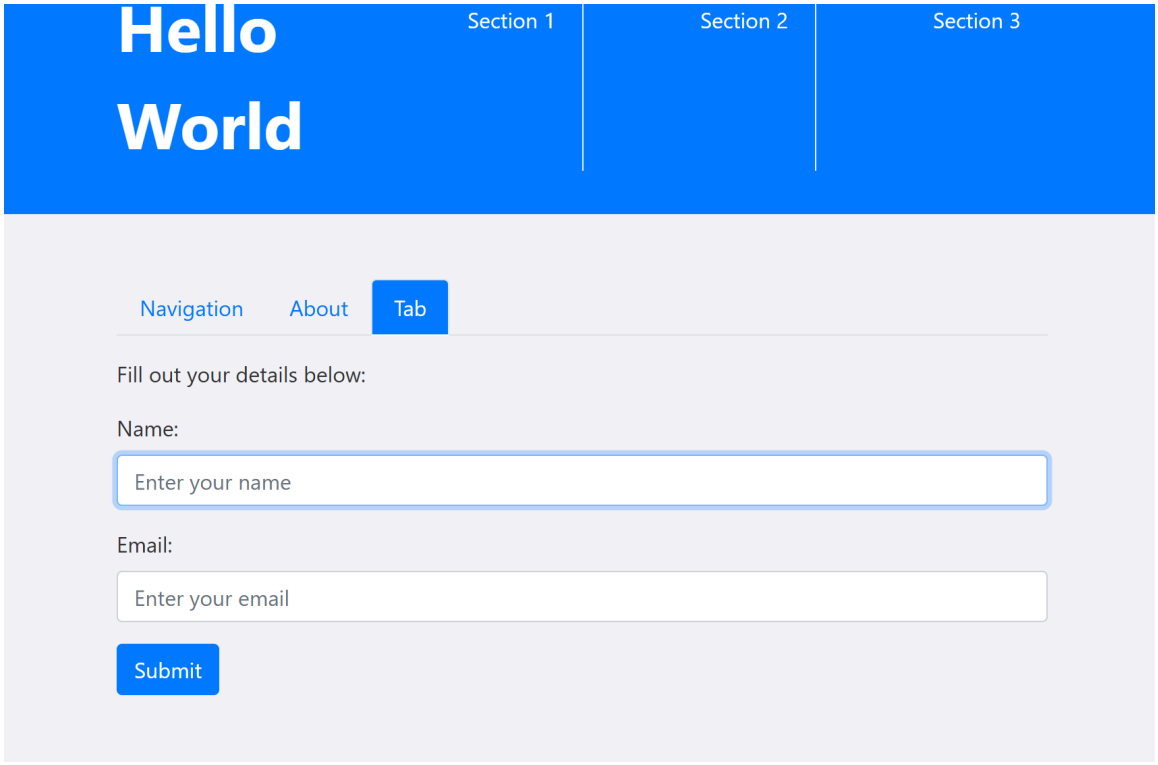


Image 1: User Interface Demo

Database

We need a database capable of storing large volumes of data and performing efficiently while handling multiple users. Not only will the necessary data concerning organizations be stored but various levels of access must be given to the user based on the role assigned. With specific reading and writing permissions a user could have, having a role-based access infrastructure within the database is especially important when handling sensitive information. The database is additionally required to have the ability to update its contents simultaneously for accurate data recording and resource availability counts within multiple service provider organizations. As WorldByMe evolves, its ability to be scalable is essential as the volume of data grows.

Ideal Solution

Scalability -

The database should be able to handle large volumes of data without decreasing the overall performance. Therefore relational databases are perfect for managing humongous amounts of data and adjusting the max capacities without degrading the database. Scalability would be even more achievable by implementing partitioning and horizontal sharding when dealing with larger data sets.

Role-Based Access Control -

Implementing a role-based access control infrastructure would be essential when granting certain permissions to specific users. Having more fine-grained control over maintaining and adjusting the data will prevent data security from being breached. Utilizing SQL seems to be the most efficient language to ensure that appropriate permissions are being granted and revoked.

Concurrency -

Being able to handle multiple users simultaneously is an essential aspect of the database, especially since connectivity between people in various scopes will be the main drive of this project. Implementing ACID principles within the database will assist in maintaining the current data while conducting simultaneous updates. Relational databases handle ACID transactions natively which in turn increases the accuracy and integrity of the current data and the new data that is being put into the database.

Secure -

Keeping the users' data privacy in mind when developing the database infrastructure is essential when applying the overall project to a government setting. Therefore, when preventing any kind of data integrity issues, encryption is required to be implemented on data whether it's sitting in the database or being updated or altered. Another method to be considered when utilizing security protocols is having periodic security checks, along with maintaining role-based access control.

Real-Time Updates -

Having consistently updated information about the service providers' availability and resources is one of the most important requirements for the database. Utilizing dynamic data within the database scope is key to accomplishing this, as it is necessary to notify the navigators in the field about the available resources and facilities. Real-time updates will assist not only the navigators but also the person in need of services.

Flexibility -

As this database infrastructure is bound to grow as time goes on, the database must hold flexibility when it comes to adjusting and maintaining the data model. This could include adding and deleting fields, adjusting existing ones, and maintaining the relationships between certain fields. Flexibility could be achieved in two different ways, utilizing a schema-less design or compromising by retrieving data using JSON.

Ease of Use -

The database should be extremely straightforward when it comes to setup, usage, and maintenance. That is because minimizing the time and resources spent on training, troubleshooting, and operating will enhance the overall efficiency of the database. Implementing a database system that has plentiful documentation, a strong user community and built-in tools for maintaining the data leads to good usability. Overall, providing the users with a database that is easy to use in terms of adjusting to needs and troubleshooting will be essential.

Possible Solutions

PostgreSQL

When trying to decide on an appropriate database for our project, PostgreSQL seems to be a leader in providing various options when implementing sharding, partitioning, and access control among users. Considering PostgreSQL's versatility in what can be done when molding the database to our needs, it is a strong candidate for handling large volumes of data without slowing down the whole data model. Unlike MySQL, PostgreSQL is fully compliant with SQL standards which highlights strong capabilities with data that demand more attuned SQL features such as INTERSECT, EXCEPT, or BETWEEN. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Additionally, implementing indexing and stored procedures will be essential when maintaining the different responsibilities the database will have. Moving on to flexibility, PostgreSQL is perfect when it comes to handling both structured and unstructured data such as JSON and XML, therefore providing PostgreSQL a good advantage when implementing complex queries. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Compared with MySQL's lack of SQL compliance, PostgreSQL makes up for its limitations by making large-scale data handling and complex querying possible.

Since there are large volumes of sensitive data that have to be considered when choosing the database, security protocols are required when maintaining data integrity. Though both MySQL and PostgreSQL have role-based access control features such as user permissions, PostgreSQL provides additional security options such as SSL-based connectivity and can be easily integrated with external authentication systems. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Another aspect of PostgreSQL that cements its credibility as a database infrastructure is its long history and extensible communities of users that enhance any support a newcomer could receive when first learning how to use PostgreSQL. Originally developed in 1987 by Professor Michael Stonebraker of the University of California Berkeley,

PostgreSQL has undergone multiple iterations, usually addressing the users' concerns and feedback. **(*Documentation: 17: 2. A Brief History of PostgreSQL*, n.d.)** With its deprecation in 1994 due to its project branch wanting to focus on other projects, PostgreSQL's popularity has persevered, and the official PostgreSQL organization emerged. Overall, PostgreSQL being open-source for a long time with a global user community has made it what it is today.

MongoDB

Being renowned for not following the relational database model, MongoDB implements a scalable and flexible design that works well with large volumes of unstructured and semi-structured data. **(*Our Story*, n.d.)** Especially since our project requires a database that will need to be flexible to suit the project's needs as it grows, MongoDB sheds light on another option when considering extreme scaling and flexibility capabilities. Like PostgreSQL, sharding and indexing are also available for large volumes of data, however, MongoDB goes in a different direction when it comes to stretching its limits with scaling. Instead of having one point of data, it can utilize multiple servers and maintain multiple databases simultaneously. **(Jain, 2023)** Since there are a lot of layers to what users can access in a typical MongoDB system design, role-based access control would be required to ensure that data is being adjusted and maintained safely. MongoDB seems to provide a native role-based access control system, having granular control over what a user has access to. **(*Built-In Roles in Self-Managed Deployments*, n.d.)** For example, a user could have read, and read-write permissions but an admin could only have privileges that relate to the database itself. MongoDB is also capable of implementing a database owner role that has access to all privileges regarding the database while the user admin is the only portion of admins that deals with anything concerning the users, such as granting and revoking permissions.

When it comes to handling multiple users at once, MongoDB is a strong candidate for concurrent access. Operating severely different from regular relational database locking, MongoDB implements a read-write locking mechanism that allows multiple users to access the same resource such as the internal collections or database. **(*FAQ: Concurrency*, n.d.)** Leading to the overall security capabilities of MongoDB, which has various options that could be implemented with our database. Of course, role-based access is another aspect of security but multiple layers of security protocols could be added as well such as encryption of queries and data, and SCRAM, which is an authentication system. **(*Security*, n.d.)** Finally, MongoDB has a built-in feature called ad-hoc queries which not only enhances performance but also retrieves real-time analytics. **(Jain, 2023)** Additionally, MongoDB is also well-known for its flexible nature towards schemas and adjusting to the project needs data-wise. Overall, MongoDB is a good choice if scalability and flexibility are the two main key requirements for our project but falls behind PostgreSQL when it comes to ACID compliance and extreme concurrency.

MySQL

Modeled after mSQL but with enhanced speed and flexibility to keep up with modern applications, MySQL has grown in popularity due to the improvement of utilizing a new SQL interface. As a result, speed and stability are usually at the forefront when it comes to thinking about MySQL. **(*MySQL :: MySQL 8.4 Reference Manual :: 1.2.3 History of MySQL*, n.d.)** Moving on to scalability and flexibility, MySQL seems to be behind in that regard, as it cannot handle large volumes of data like PostgreSQL and MongoDB can. Therefore, it is a good choice if the application only requires small amounts of data to be functional, which is not possible in our case. Additionally, the capabilities of sharding and indexing are limiting when utilizing a MySQL database, which would not be beneficial in the long run. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Though scalability and flexibility might be out of the question for MySQL, there is a built-in role-based access control system where users can be granted certain access rights to different parts of the database. Compared to MongoDB and PostgreSQL, the amount and variety of roles are not limited to maintaining and setting up the user privileges to the client's liking would not be difficult at all. **(*MySQL :: Security in MySQL :: 4.10 Using Roles*, n.d.)** Leading to being able to handle multiple users at the same time within the database, MySQL also supports multiple users reading and writing in the database simultaneously due to how the ISAM storage engine supports concurrency. **(*MySQL :: MySQL 8.4 Reference Manual :: 10.11.3 Concurrent Inserts*, n.d.)** Additionally, when it comes to implementing security within a MySQL database there are options, such as role-based access and encryption. However, there have been instances where MySQL's data integrity has been questioned, which has led to degraded trust within the user community.

When it comes to implementing real-time updates within the database, being able to have access to the most recent information is essential for this project. Luckily, MySQL does provide an UPDATE statement that could be triggered each time new data is put in the database. However, implementing the UPDATE statement could be inefficient in the long run since excessive CPU and memory usage would be consumed to make the updates happen. **(*MySQL :: MySQL 8.4 Reference Manual :: 15.2.17 UPDATE Statement*, n.d.)** From a big-picture perspective of using MySQL as the database, MySQL involves ease of use including the setup, utilization, and even uninstalling. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Overall, MySQL would not be the most optimal solution however, there are workarounds for some of the issues an admin or user could encounter.

Considering all of the above requirements for the solution, each database will be given a checkmark if scalability, role-based access control capabilities, concurrency limits, security protocols, real-time updates functionality, and flexibility are achievable. Therefore, each category is given a checkmark based on the guidelines below:

- ***Scalability*** - *If the data capacity needs to increase, is it possible with this database?*

- ***Role-Based Access Control*** *- Does this database have a good ability to define and enforce roles and permissions effectively?*

- ***Concurrency*** *- Is it able to handle multiple simultaneous data transactions without degrading data integrity?*

- ***Security*** *- Is this database able to support encryption, authentication measures, and/or auditing?*

- ***Real-Time Updates*** *- Are real-time updates and notifications possible to implement?*

- ***Flexibility*** *- Is the database able to receive and implement data schema/data model changes effectively?*

Database	Scalability	Role-Based Access Control	Concurrency	Security	Real-Time Updates	Flexibility	Ease of Use
PostgreSQL	✔	✔	✔	✔	✔	✔	✔
MongoDB	✔	✔	✔	✔	✔	✔	
MySQL		✔	✔	✔			✔

Table 3: Comparing the three databases by requirements

Tech Demo

For the tech demo, a test database that stores test data that greatly resembles the data that will be collected by navigators will be set up and interacted with. Implementing a small dataset and then scaling it up in size will put the scalability to the test. Additionally, different roles with various user permissions will be granted to test users and those roles will be displayed. When it comes to concurrency, a python script will be created and implemented to ensure that multiple users can access the same database simultaneously. Security protocols can be set up easily and can be shown by displaying how password hashing and encryption is applied to the database. When demoing the real-time updates, event triggers can set up where when new data is put in the database, then the database is automatically updated. Lastly, flexibility can be shown by adjusting fields, columns, rows, and overall structure of the database within the pgAdmin4 interface.

Analysis

PostgreSQL

Recalling from the contents of Table 2, PostgreSQL is a strong contender when it comes to deciding on the database we are going to use. PostgreSQL is extremely

well-known for its scaling and flexibility capabilities which is essential for our project since there will most definitely be large-data handling.

When it comes to implementing scalability through PostgreSQL when deployed on AWS specifically, the database is able to support large volumes of data in a way that optimizes performance and improves cost. For example, an AWS general purpose SSD (gp3) has a concrete performance baseline at 3,000 input/output operations per second, utilizing around 125 mebibytes per second. **(*PostgreSQL Benchmark Observations and Considerations - Optimizing PostgreSQL Running on Amazon EC2 Using Amazon EBS*, n.d.)** That is super impressive considering the white paper that recorded the performance benchmark tested the limits of PostgreSQL and AWS EBS with multiple deployments and worked them to the max amount of CPU power, which was 128GB. In case there are more critical workloads, an AWS whitepaper concerning the PostgreSQL performance benchmark also highlights how various available general SSDs through AWS provide customizable input/output operations levels. Therefore, the whitepaper highlights how PostgreSQL's performance does not degrade under high levels of stress easily.

Implementing concurrency is also very possible by utilizing PostgreSQL, which is essential since multiple navigators at once need to be able to access the database. Looking more at the AWS PostgreSQL performance benchmark, concurrency within PostgreSQL was tested within multiple instances, in which even the lowest version of a general SSD (gp2) was able to handle over 440 connections at once, processing over 10,238,208 transactions per millisecond and implementing 5,868 transactions per millisecond. **(*PostgreSQL Benchmark Observations and Considerations - Optimizing PostgreSQL Running on Amazon EC2 Using Amazon EBS*, n.d.)** With these numbers in mind, it is clear that PostgreSQL is more than capable of handling multiple concurrents simultaneously.

Moving on to overall performance of security, PostgreSQL's security options involved implementing role-based access control and encryptions such as data partitioning, SSL host authentication, and client-side encryption. **(*Documentation: 17: 18.8. Encryption Options*, n.d.)** In the long run, security could be even more enhanced using multiple layers of security such as server layer security protocols. In case a navigator is looking for new information about the services that are available, events can be created and triggered specifically when new information is inserted into the database. If the data model needs to be adjusted in the future, then PostgreSQL aligns with SQL-compliance well enough that editing fields, tables, and even the database infrastructure can happen at a push of a button. Overall, PostgreSQL seems to align with all of our ideal solution requirements, which highlights it as the top choice.

MongoDB

With MongoDB being a viable choice for a database as highlighted in Table 2, this specific database excelled in almost every category except ease of use. However, with its

extreme granularity concerning roles and flexibility capabilities going hand in hand with the amount of high-stress performance it can still implement without crashing the whole database, it is no wonder that MongoDB has great potential to help our project succeed.

Starting with scalability and concurrency, it was found that MongoDB's horizontal scaling capabilities can cause issues when handling large datasets, which is a huge requirement for our project. In a cloud environment, MongoDB Atlas supports auto-scaling, adjusting the resources to meet demanding workloads. For example, for a single instance MongoDB environment on AWS can handle over 1,000 input/output operations per millisecond and can support over 1,000 concurrent users simultaneously without compromising any functionalities of MongoDB. **(Seybold, n.d.)** Leaning to role-based access, MongoDB offers built-in role access control where the granularity can even include the owner of the database, database admins, user admins, and the users themselves. **(*Built-In Roles in Self-Managed Deployments*, n.d.)** Therefore, implementing the level of granularity concerning roles within our data infrastructure could make the overall granting and redaction process easier.

Moving on to security, MongoDB utilizes basic security protocols when it comes to protecting data integrity, but those options could only extend when paired with the server layer's security protocols. **(*Security*, n.d.)** There are built-in authentication systems such as SCRAM, encryption and data partitioning are also available on multiple levels of a MongoDB infrastructure. Along with implementing real-time updates, MongoDB utilizes a different approach as SQL is thrown out the window when dealing with the data, which involves learning MongoDB Atlas, which implements an auto real-time update mechanism. Lastly, flexibility is something that MongoDB excels in immensely as the schema-less design allows many of the structures to be what the developer/client wants. **(*Our Story*, n.d.)** Overall, MongoDB would take our project in an interesting direction, allowing us to take role-based access within a secure environment to another level. However, the learning curve that is associated with learning MongoDB would present a critical challenge, especially since the team is mostly familiar with SQL.

MySQL

Our third option is also a strong contender as displayed in Table 2 due to its ease of use, its ability to implement roles throughout the data model, concurrency, and its security protocols. MySQL stood out in regards to its natural ACID-compliance, and its ability to handle structured data. Therefore, there are a few requirements that MySQL fulfills when implementing our extremely structured, role-based system.

When it comes to scalability within a MySQL database, it is doable but there would be many challenges in doing so with the amount of data that we would have. There would be two options available according to the AWS MySQL Performance Benchmark when it comes to scaling, which would be to add more memory, CPU, and disk storage to the interface, or have multiple servers each with instances of the MySQL database. **(*MySQL Benchmark Observations and Considerations - Optimizing MySQL Running on Amazon EC2 Using Amazon EBS*, n.d.)** With that in mind, the AWS benchmark came with Moving on to

role-based access control capabilities, MySQL does provide the usual GRANT statements and SQL commands that come with it. Though the GRANT statement does get the general idea of roles implemented, it doesn't provide as much granular detail as MongoDB does. Moving on to concurrency, MySQL can implement 500 mebibytes per millisecond and supports around 100 users simultaneously. **(*MySQL Benchmark Observations and Considerations - Optimizing MySQL Running on Amazon EC2 Using Amazon EBS*, n.d.)** When it comes to security there are the usual methods such as encryption however MySQL has been under fire for a long time concerning security breaches on multiple levels.

Keeping the fact that navigators need to see the amount of available service that nonprofits can provide in real-time, it is good to know that MySQL usually has the UPDATE statement available. **(*MySQL :: MySQL 8.4 Reference Manual :: 15.2.17 UPDATE Statement*, n.d.)** However, with the limitations of how the updates can happen, it doesn't seem efficient in the long run when it comes to CPU and cost. Lastly, despite MySQL being a relational database, due to it not being completely SQL compliant, it can be difficult to adjust the data model if the project needs it. **(*PostgreSQL Vs MySQL: Partitioning, Replication, Query Optimization, and More*, 2024)** Therefore, flexibility is out of the question once a whole database is developed. Overall, MySQL does have advantages such as being able to implement roles, support multiple users at once, and being easy to use, however, it doesn't align well with all the requirements.

Initial Choice

Looking back at the score table above the analysis, it is clear that PostgreSQL has ranked the highest in most of the categories. Its robust role-based access control system goes hand in hand with PostgreSQL's overall amazing security system. Out of all the databases, PostgreSQL has proven to be the database that is most capable of scaling up despite the large volumes of data being stored. Additionally, flexibility and the real-time update scoring might not have been the best out of all three options, but paired up with the security protocols and role-based access control system will even out the disadvantages.

Conclusion

Based on the ranking table and in-depth analysis of each database, PostgreSQL has emerged as the best choice for HelloWorldByMe. Due to its strong capabilities regarding security, scalability, role-based access, and flexibility, the handling of large amounts of sensitive data and structured data aligning with real-time updates will be doable. If PostgreSQL proves to have too many unforeseen challenges, MongoDB could be seen as a solid second solution. MySQL could be another option if a simpler, but reliable fallback option is needed. Overall, PostgreSQL will be the database in the final design with MongoDB coming in as a secondary option and MySQL as a last resort. [13]

---

[13] *Benefits at a Glance*. (n.d.). AWS. Retrieved October 31, 2024, from https://aws.amazon.com/application-hosting/benefits/
Built-In Roles in Self-Managed Deployments. (n.d.). MongoDB. Retrieved October 30, 2024, from https://www.mongodb.com/docs/manual/reference/built-in-roles/

Server

While having a scalable and secure database is essential, implementing a server that is responsible for hosting and managing our evolving data storage is equally important. Centralized hosting allows accessibility of the application through different locations and various devices without degrading the performance of the multiple components. That being said, the server will have to be scalable to accommodate the varying amounts of traffic, ensuring that our project's application will be responsive under different workloads. Security is another feature that will need to be taken into consideration as the server will be the backbone of all our transactions intertwining all the components. Therefore role-based access will be looked into when maintaining the data integrity of our server when managing sensitive data. As the progression and behavior evolve, the server will need to have flexibility when it comes to the customization of specific details such as operating system, network configurations, and other server needs. With the customizations in mind, the reliability and uptime of our server will also be accounted for when circumstances of any complications arise within the components. Minimal downtime is essential when making sure the project remains available when users need to use our features. Lastly, our server needs to be able to integrate with external applications like APIs with ease, especially since the monitoring, fallback, and other email, messages, and location-based services will be key when maintaining the condition of our application.

Ideal Solution

Scalable -

As the application continues to grow in terms of users, scalability is key to making sure that the server is still functional despite the traffic increase. Having a scalable server will ensure that the capacity can adjust to the demands which will prevent performance delays. Additionally, implementing dynamic allocating behavior when meeting the needs

Documentation: 17: 18.8. Encryption Options. (n.d.). PostgreSQL. Retrieved October 31, 2024, from https://www.postgresql.org/docs/current/encryption-options.html

Documentation: 17: 2. A Brief History of PostgreSQL. (n.d.). PostgreSQL. Retrieved October 30, 2024, from https://www.postgresql.org/docs/current/history.html

FAQ: Concurrency. (n.d.). MongoDB. Retrieved October 30, 2024, from https://www.mongodb.com/docs/manual/faq/concurrency/

Jain, S. (2023, December 1). *MongoDB Advantages & Disadvantages*. GeeksforGeeks. Retrieved October 30, 2024, from https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/

MySQL benchmark observations and considerations - Optimizing MySQL Running on Amazon EC2 Using Amazon EBS. (n.d.). AWS Documentation. Retrieved October 31, 2024, from https://docs.aws.amazon.com/whitepapers/latest/optimizing-mysql-on-ec2-using-amazon-ebs/mysql-benchmark-observations-and-considerations.html

MySQL :: MySQL 8.4 Reference Manual :: 10.11.3 Concurrent Inserts. (n.d.). MySQL :: Developer Zone. Retrieved October 30, 2024, from https://dev.mysql.com/doc/refman/8.4/en/concurrent-inserts.html

MySQL :: MySQL 8.4 Reference Manual :: 1.2.3 History of MySQL. (n.d.). MySQL :: Developer Zone. Retrieved October 30, 2024, from https://dev.mysql.com/doc/refman/8.4/en/history.html

MySQL :: MySQL 8.4 Reference Manual :: 15.2.17 UPDATE Statement. (n.d.). MySQL :: Developer Zone. Retrieved October 31, 2024, from https://dev.mysql.com/doc/refman/8.4/en/update.html

MySQL :: Security in MySQL :: 4.10 Using Roles. (n.d.). MySQL :: Developer Zone. Retrieved October 30, 2024, from https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/roles.html

Our Story. (n.d.). MongoDB. Retrieved October 30, 2024, from https://www.mongodb.com/company/our-story

PostgreSQL benchmark observations and considerations - Optimizing PostgreSQL Running on Amazon EC2 Using Amazon EBS. (n.d.). AWS Documentation. Retrieved October 31, 2024, from https://docs.aws.amazon.com/whitepapers/latest/optimizing-postgresql-on-ec2-using-ebs/postgresql-benchmark-observations-and-considerations.html

PostgreSQL vs MySQL: Partitioning, Replication, Query Optimization, and More. (2024, September 23). EDB. Retrieved October 30, 2024, from https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features

Security. (n.d.). MongoDB. Retrieved October 30, 2024, from https://www.mongodb.com/docs/manual/security/

Seybold, D. (n.d.). *Performance Benchmarking of MongoDB*. benchANT. Retrieved October 31, 2024, from https://benchant.com/blog/mongodb-benchmarking

of the server can help reduce costs and resources when the demand is low and can also result in cost efficiency. Increasing the effectiveness of the application later on by developing additional features, services, or geological data will require a scalable server, as scopes can be adjusted without interruption.

Secure -

When handling sensitive data within a local government and service provider organization, security protocols will need to be addressed. Increasing scalability on the server also means increasing security by implementing more complex methods. Therefore, considering the infrastructure of this project, role-based access control is the obvious choice. One of the most important underlying features of our application is that only certain people with the correct permissions will be able to access specific components. Making sure that there are security measures in the circumstances of security breaches, hacking, ransomware, etc. will be beneficial in the long run.

Customizable -

Customization is essential when adjusting the environment to the project's needs, ensuring that optimizing security and performance is a priority. Looking into the fact that each component of the server will have differing processing capabilities, storage requirements, and memory, implementing customization will prevent unnecessary workloads and utilize maintenance schedules.

Compatible integration with email/messaging APIs

Having the ability to communicate between entities and within organizations is important when it comes to sending and receiving essential information about the services available. Additionally, having compatibility with the APIs will pave the way for role-based access control from the front-end perspective. Involving certain users to interact with the appropriate features whether it be on the field or behind the government desks will ensure that the correct group of people can access the correct information.

Reliable -

Another part of the ideal solution would be that the server has to be reliable under high-stress circumstances. By finding methods that minimize downtime, users will be able to access the project whenever they need it. A reliable server also ensures that the data is being collected, processed, stored, and retrieved without any issues even under high loads. Additionally, the server needs to be able to increase the number of transactions and requests that can be handled within a specific period as the application gains more users. Having a server that is capable of this will highlight how performance will not be affected despite the number of users active while also reducing the amount of downtime should a huge complication occur.

Automation -

Automation is important when it comes to maintaining the peak state of the application. With this in mind, automating workflows such as software updates, data backups, and security checks would assist in reducing human error and ensuring the infrastructure is up to date. Depending on how the server ends up being set up, data synchronization will also need to be automated so that relevant organizations or components within the server can be on the same page. Additionally, there might be scheduled notifications or messages when it comes to keeping information about service availability recent for the navigators. Overall, automation would keep the project's performance efficiency in check and make sure that background tasks are being completed.

Cost -

Evaluating the overall cost of the server is also critical, as expenses should ensure the overall project is maintained. The initial choice of hosting should not only keep the performance and scalability of the server in mind but also the budget. Fortunately, there are two main options when it comes to cost, which are pay-as-you-go or fixed pricing. Even after the initial setup is complete the cost of maintaining and operating the server might come at a separate cost. Therefore, finding a balance between affordability and performance will be a key factor in the project's success.

Possible Solutions

AWS

At first glance, AWS provides a couple of server options that align well with what the project requires which will be addressed in the following analysis. However, looking at the general overview of AWS as a whole, it is not hard to find that exceptional scalability is offered. There are multiple tools available that could help with the scaling such as AWS Auto Scaling and Elastic Load Balancing which allows applications to scale up or down depending on the project's needs. **(*Benefits at a Glance*, n.d.)** Moving on to the overall security options, AWS leads the server community in that regard as an end-to-end approach is implemented. Therefore, various aspects of a project could be covered when it comes to security, such as physical, operational, and software security. **(*Benefits at a Glance*, n.d.)** What makes AWS stand out from other server services is that many of the packages offered are extremely flexible, allowing developers to only pay for what is required for the result they are trying to achieve.

Therefore, it is not surprising that AWS has many APIs regarding messaging and email available and even provides support for external third-party APIs in case the desired one is not offered already. **(*Benefits at a Glance*, n.d.)** Keeping the capabilities of the messaging and email options available in mind, it is good to also remember that automation and reliability are essential when keeping the overall health of the project intact. Since AWS has multiple points of presence, it will not be difficult to keep the application stable infrastructure-wise. **(*Benefits at a Glance*, n.d.)** Lastly, the cost of an AWS server varies greatly on the package selected and how much of the package is utilized, which greatly affects how the overall server is priced. Therefore the pricing is very flexible

in most cases, which will be beneficial for the project in the long run since the goal is to have the costs as close to free as possible.

Google Cloud

Being well-known for flexibility and scaling, Google Cloud seems to be another option that could be beneficial for this project if implemented correctly. Since Google Cloud implements an open-cloud stance, infrastructure complexity can be reduced by managing multiple clouds with different workloads of various sizes. (***7 Reasons To Move To Google Cloud Platform*, n.d.**) Giving Google Cloud a try by creating multiple cloud instances for each component seems to be an interesting, achievable option but also the most risky. Google Cloud was developed with businesses in mind, and the default settings and configurations could clash with what the team is trying to do with our server. Moving on to Google Cloud's security capabilities, Google Cloud's security command center would make it easy to implement security protocols and be able to determine the level of security needed.

Unfortunately, Google Cloud has limits when it comes to customization and compatibility with messaging and email APIs. Compared to AWS, Google Cloud is very focused on having different varieties of different cloud services that fit a specific setup. That is because business owners usually don't have any tech experience. **(*7 Reasons To Move To Google Cloud Platform*, n.d.)** On the other hand, implementing automation and reliability is possible as the security command center dashboard is available and automation is known to be easy to set up within Google Cloud. Another aspect that is vastly different from AWS is the cost of the cloud services, in which the cost doesn't seem to be worth the services that are being offered. Fixed pricing is available but any Google product is expensive. Overall, Google Cloud is not a bad option, however, some limitations would need to be addressed if Google Cloud were to be implemented within our project.

Azure

Though Azure has received high criticism from users in recent years, it still has the potential to become a good candidate for acting as the server for our project. Starting with scalability and security, Azure has simplified the process of scaling a project up or down depending on what the project requires. **(*The Pros and Cons of Microsoft Azure: Cloud Services for Businesses*, n.d.)** With Azure being a leader when it comes to IaaS security, Azure's cybersecurity protocols go hand in hand with the standard security model which includes: detecting, assessing, diagnosing, stabilizing, and closing. **(*The Pros and Cons of Microsoft Azure: Cloud Services for Businesses*, n.d.)** However, there have been multiple data breaches all linked to Azure, so the trust in security might have been degraded within the Azure user community. When it comes to implementing messaging/email, there are built-in APIs meant to deal with that specific aspect.

However, when it comes to reliability and cost, there have been multiple blogs highlighting mixed feelings about those two parts of our ideal solution. Due to the conflicting reports of these two highly important aspects of any server, it would be best in

the long run if Azure was not implemented. **(*The Pros and Cons of Microsoft Azure: Cloud Services for Businesses*, n.d.)** Overall, Azure is not the best solution but it does consist of features that our project would need to be successful.

Heroku

Originally developed to resolve issues with deploying Ruby on Rails applications, Heroku has changed the game when it came to implementing a serverless approach. Utilizing dynos or containerization instead, there is a downside to becoming "serverless" and that is the ability to scale applications becomes impossible. **(*The Story of Heroku*, 2022)** That is because evolving projects like our own do not translate well in monolithic applications like dynos. When it comes to security, there have also been frequent security breaches and performance-related issues that have affected how its user community perceived the product. **(*The Story of Heroku*, 2022)**

With that in mind, there is an advantage to Heroku in that the beginning stages of our project would be perfect for the infrastructure that Heroku has. Additionally, migration would not be especially difficult since Heroku is built upon AWS, should an application need to switch to AWS. **(*The Story of Heroku*, 2022)** With the overall infrastructure of Heroku in mind, it is not surprising that the automation options, pricing, and ability to implement messaging and emails are super limited, given that Heroku serves a very specific niche of application. **(*The Story of Heroku*, 2022)** Overall, Heroku would not be a strong candidate for hosting our architectural design as many of the requirements are not met through Heroku. However, the beginning stages of developing our application would be perfect with Heroku as it provides a lot of support for small versions of projects.

Considering all of the above requirements for the ideal solution, our server will be ranked by how scalable it is, the security protocols available, how well the server could be customized, the effectiveness of the messaging API, how reliable the server is in times of crisis and how well automation can be carried out. Each category is given a checkmark based on the guidelines below:

- ***Scalability*** - *Can the server adjust the sizing of the components of the application?*
- ***Customizable*** - *Is there any kind of way that the server's packages compromise with some sort of customization?*
- ***Security*** - *Does the server provide encryption support, some kind of amount of security compliance, and integrated access controls?*
- ***Automation*** - *Will the background tasks and maintenance be effective and executed on time?*
- ***Emails/Messages*** - *Are the server's APIs going to be compatible with various components regarding notifications/messaging?*

- ***Reliability*** *- In times of crisis, how well can the server provide assistance/tools for an event to be tracked and dealt with?*

- ***Cost*** *- Are the prices available for each server going to be reasonable in the long term? (Close to free as possible)*

Server	Scalability	Customizable	Security	Automation	Emails/Messages	Reliability	Cost
AWS	✔	✔	✔	✔	✔	✔	✔
Google Cloud	✔		✔	✔	✔	✔	
Azure	✔			✔	✔		
Heroku			✔	✔			✔

Table 4: Comparing the three servers by requirements

<u>Tech Demo</u>

For the tech demo, we will utilize our choice of server by hosting our components on it. Scalability can be found when the capacity of different resources are adjusted and AWS is customizable to different packages and needs, so the team could show which package and customizations are chosen. When it comes to security and automation, the team could set up security layers within the server and automation could be applied when updating various parts of the infrastructure. This could include when various API and real-time updates are called within the server. Emails and messaging are supported through customization of APIs when using AWS. Finally, when it comes to reliability and cost, we could demonstrate reliability stats and the cause and effects of certain circumstances when they occurred.

Analysis

AWS

From the get-go, AWS has been a strong choice for a server, as it aligns with all of the requirements. Not only are the scalability capabilities, reliability, and security protocols proof that AWS has the potential to provide an extremely stable backbone to our application, but AWS sticks out from the rest of the servers for its customizability. Therefore, Table 2 highlights how AWS fulfills all of the requirements.

Scalability is a category that AWS stands out the most for, as its dynamic nature to scale up or down depending on what's needed is beneficial for the project in the long run. Since the application is predicted to grow in size, adaptable scaling is important. AWS allows for auto scaling through the Amazon EC2 instance which is an option that could be taken. Utilizing the auto scaling feature will allow the application to respond to high traffic in short periods. For instance, AWS can support 15 reading replicas and store up to 64 TB

per instance. **(Wurden, 2021)** Moving on to the customizable category, AWS has various packages that can be adjusted to the project's needs at varying price ranges.
When it comes to security, AWS provides multiple layers of security, such as encryption, user authentication, and even role-based access control. With automation and reliability in mind, AWS is very efficient when it comes to handling transactional workloads especially when utilizing the gp3 SSD. According to the AWS performance benchmark, an Amazon EC2 r5b. a 16xlarge instance with a gp3 SSD can have up to 1.79 times higher transactional output and 1.76 times less transactional latency compared to the average Azure transactional output and latency. **(Wurden, 2021)** When it comes to implementing messaging and emails, AWS provides a built-in SES API that works well with maintaining emailing and messaging systems. Lastly, when it comes to cost, AWS is one of the best options for more flexible budgets. For example, the regular EC2 instances with the usual gp3 storage usually cost $46.50 on average for every 1000 users that are to be supported. However, Azure servers with premium storage capacity usually are priced around $91.93, which is around a $46 difference. **(Wurden, 2021)** Overall, AWS is a strong contender as it aligns with all of the requirements.

Google Cloud

Google Cloud is another strong choice when deciding on a server for our project as the scalability and security are one of the best in the industry. Compared to AWS, Google Cloud implements more of a cloud infrastructure than a concrete server architecture. Therefore, if Google Cloud is chosen as the database for our project, an interesting direction might be taken.

Starting with scalability, Google Cloud offers various solutions for different input/output commands. Depending on the protocols utilized, different behaviors were sure to be found. For example, a 64 KiB block size utilizing the SMB protocol maxed at 5,130 MiBps for reading in data and 1,790 MiBps for writing data. **(*Performance Benchmarks*, n.d.**) Therefore proving that Google Cloud is very scalable when updating database parameters. Though a cloud server is not very customizable, Google Cloud has made many advancements in security. While implementing a centralized security dashboard, Google Cloud changes the game within the server industry by allowing business owners to troubleshoot an incident instantly. **(*Google Cloud Security Overview*, 2022)** Automation and reliability also go hand in hand when incorporating Google Cloud as the simplified user interface makes automation and tracking reliability as simple as clicking a button.

Since Google Cloud is very focused on certain categories of cloud servers, not only would it make implementing a cloud-based server not very customizable but make it not suitable to host messaging and emails. The cost for a cloud server is a fixed amount but each type of cloud server would have a different budget range, which would not be good if all the parts of the package are not used. Overall, Google Cloud is a strong contender but it's not fully suitable for our project.

Azure

Scoring the least of all of the servers, Azure has been known to have methods to implement scalability like horizontal sharding, but there seems to be a massive bug regardless. Therefore, if the team were to utilize Azure, it would be very hard to scale up as much as the project required to be. For example, compared to an Intel Core computing instance, which scales very well, Azure only measures up to 430% in scalability limits from the original size compared to Intel's 797%. **(*Performance Benchmark of Over 30 Azure VMs in AKS*, 2024)** Additionally, though Azure has its role-based access control system, it is unclear from the research whether or not the system is effective enough. That is because the large amounts of data that will be stored might not be compatible with the limited scalability. Implementing a server that is limited in the aspect of having role-based access control is not good for a project that also requires multiple layers of security.

When setting up the various levels of security it would be imperative for our team to set them up correctly the first time as it would cause many issues if installed incorrectly. Additionally, Azure is facing many directions of controversy as a huge data breach through Azure has compromised many Microsoft accounts, including Microsoft executives. With the data integrity issues from Azure in mind, it would not be fully safe to trust it with all our sensitive data. Fortunately, Azure automation and email/messaging APIs are also known to be effective.

Since those resources would be available through Azure, much of the automation and any monitoring would be recorded natively. Knowing this, it is okay to say that Azure has a lot of issues with security and capacity which are two key aspects of this application backend perspective. **(*Performance Benchmark of Over 30 Azure VMs in AKS*, 2024)** However, if role-based access control, automation, and email/messaging APIs were a bigger priority, then Azure might be a good option. However, it would not be a good idea to sacrifice security and scalability in the meantime.

Heroku -

Compared to Azure, Heroku is more well-versed in scalability as it has a native auto-scaling system that implements a certain amount of dynos to reach the 95th percentile response time threshold. For example, Heroku is able to support up to 124 dynos in one instance, which is unheard of with Heroku's design infrastructure. **(*Heroku Dynos: Standard Vs Performance*, 2020)** However, on the other hand, Heroku did not score as well in role-based access since it is a server that is not focused on specific security methods like that. With the amount of sensitive data that will be utilized throughout the infrastructure, role-based access control is a must. Keeping role-based access control in mind, it is an interesting thing to rank the security of Heroku a higher score.

However, it is because there are other effective methods besides role-based access such as hashing, encrypting, etc. Therefore, there are certain levels of security that Heroku provides, but it might not be enough for what our project is going to be dealing with. Heroku's security could be better but its automation capabilities are still well-known in the server world. That is because the original developers noticed how Ruby on Rails is hard to automate and deploy, so Heroku was launched. Since automation is guaranteed to be good, implementing emails and messages might prove to be a challenge.

Keeping track of the reliability of the project would not be hard with Heroku, however, that feature still seems to be in testing so it might not be entirely accurate. So far

with the current performance benchmarks that are coming out, it was found that Heroku's dynos are able to decrease latency by 43% and the 99th percentile latency by 56% compared to Azure. **(*Heroku Dynos: Standard Vs Performance*, 2020)** Furthermore, since there is no concrete evidence that reliability Heroku offers recently is effective, it seems to not be a good idea to give Heroku the benefit of the doubt. Overall, Heroku is known to be very good at specific things that don't fully apply to the magnitude of our project, such as scalability and automation.

Initial Solution

Given the project's requirements, the ranking shows that AWS is the most suitable solution. Outdoing Google Cloud, Azure, and Heroku in multiple categories such as scalability, reliability, automation, etc., AWS is going to be able to provide the team and client with what they need to complete the project. Specifically, AWS's EC2 instances and Beanstalk instances seem to be the most reasonable as they provide extensive support for automation and implementing more customizable infrastructure. Despite the learning curve that will undoubtedly happen with the team, the amount of documentation and flexibility even out the doubts.

Google Cloud came in as a second choice, with this cloud server offering almost the same requirements that AWS fulfills, but had different fallbacks with the API compatibility and automation. Azure on the other hand has been ranked the worst due to how many external controversies with data security and integrity that it didn't seem like a good choice. Heroku is better suited for the early stages but would not do well as the project was developed more. Overall, each server had its advantages and disadvantages but fulfilling the requirements was the most important.

Conclusion

After taking the time to compare the different servers, AWS has stood out as the most suitable option. It can include new fields and features should it need to be added, and the security protocols implemented enhance the project. Looking at how the background tasks would be handled along with the software updates, the benefits of having some sort of automation make AWS a winner. In case there are complications, Google Cloud is another stable option but lacks automation and email/messaging APIs. Overall, utilizing AWS would not only fulfill the requirements but it is the most stable choice among the whole group.[14]

---

[14] *Google Cloud Security Overview*. (2022, June 22). Google Cloud. Retrieved October 31, 2024, from https://cloud.google.com/blog/topics/developers-practitioners/google-cloud-security-overview

Heroku Dynos: Standard vs Performance. (2020, December 24). Code 402. Retrieved October 31, 2024, from https://code402.com/blog/heroku-standard-vs-performance-dynos/

Performance benchmark of over 30 Azure VMs in AKS. (2024, January 28). AugmentedMind.de. Retrieved October 31, 2024, from https://www.augmentedmind.de/2024/01/28/benchmark-azure-vm-in-kubernetes/

Performance benchmarks. (n.d.). Google Cloud. Retrieved October 31, 2024, from https://cloud.google.com/netapp/volumes/docs/performance/performance-benchmarks

Wurden, F. (2021, December 9). *Performance Benchmark – SQL Server Workload on AWS and Azure | Amazon Web Services*. AWS. Retrieved October 31, 2024, from

https://aws.amazon.com/blogs/modernizing-with-aws/performance-benchmark-sql-server-workload-on-aws-and-azure/#

Navigation Services

Ideal Solution

The application that is being developed requires navigation as one of its most important components. There are many factors that are to be considered when choosing the best navigation service for our website implementation. The first one is having a navigation service that can handle large amounts of traffic and data while also remaining cost-effective. Additionally, the navigation service should offer features such as route optimization and customizable options for specific user needs. Integration capabilities with other APIs and services are also essential in order to ensure a seamless functionality and navigation experience.

Key aspects we considered for navigation implementation included:

- Features: We want a navigation service that includes a wide range of features accessible to users
- Functionality: The navigation service must remain functional through high data loads and processes across multiple platforms
- Performance : The navigation services should be able to have high performance based on the amount of traffic on the website
- Customization: The navigation service should be customizable and less restrictive on how the maps should be implemented.
- Pricing: The navigation services should be priced fairly based on the websites potential data usage with navigation services
- Ease of Use: There should be a relatively easy learning experience with the chosen map integration for project efficiency and implementation

Possible Solutions

Google Maps API

Google Maps API would allow us to integrate the Google Maps interface onto our application. Google Maps provides a sense of familiarity and high accuracy in their systems as well as being able to be cross platform across web applications and mobile devices, working with both IOS and Android apps (Google Maps Platform, 2024)[15]. Though these are all features that we would want to incorporate into our application, there are some negatives to using Google Maps API.

Google Maps API advertises itself to be free, however, this is dependent on the usage that an application would need. The web application being developed is incredibly extensive in navigation usage, consequently making it a high traffic application, passing the free tier usage that Google Maps API has to offer. Another downside is that there are

[15] Google Maps Platform, (2024). Retrieved from https://developers.google.com/maps/documentation/ios-sdk/get-api-key#:~:text=Note%3A%20You%20can%20use%20the,Places%20SDK%20for%20iOS%20apps.

request limits with the navigation which would ultimately hinder user experience if one of the main aspects of the application has a limit to it. Due to these high limitations, Google Maps API would be on the lower list of potential options. (Khalimonchuk, 2022)[16]

MapBox

Mapbox is another mapping API that is able to create maps and have applications able to implement their mapping data. The biggest reason why we would consider using MapBox is due to its customizable map feature which allows for developers to tailor specific maps on their application to their needs, this would make it more customizable than Google Maps and allow for us to tailor to our navigation needs. (Ahmad, 2024)[17].

Some limitations to this API would be the cost. Like Google Maps, it also has the free tier but goes up in price when the application is high in traffic. This application also has limited coverage in some areas which may not be ideal in certain situations. These limitations make MapBox also a lower option on our potential navigation lists.

Leaflet

Leaflet is a JavaScript library for interactive maps. This application is lightweight and easy to use, allowing for maps to be customized based on the application's needs. It supports multiple device platforms, ensuring a responsive experience for users on desktops, and smartphones alike. Additionally, Leaflet allows for maps to incorporate external plugins, which further enhance the customization options available. (Stepnov, 2021[18]) This flexibility makes it an ideal choice for our application as it creates the best option for creating tailored mapping experiences.

The only potential drawback from using this library comes from the lack of routing that it has. Compared to its competitors, leaflet is a more simple application so the complex aspects like routing or even geolocations may be complicated to implement in the systems.

Navigation Services	Features	Functionality	Performance	Customization	Pricing	Cross Platform	Ease Of Use
Google Maps API	✔	✔		✔		✔	✔
MapBox	✔	✔	✔	✔	✔	✔	

[16] Mapbox vs. Google Maps: Choose Your Perfect Map API. (2022). Retrieved from https://fulcrum.rocks/blog/mapbox-vs-google-maps

[17] Mapbox Vs Google Maps: Best Map API Comparison 2024. (2024). Retrieved from https://wpmaps.com/blog/mapbox-vs-google-maps-comparison/

[18] Stepnov, E., & LLC., F. (2024). Top Mapping and Maps APIs for Your Application. Retrieved from https://flatlogic.com/blog/top-mapping-and-maps-api/

Leaflet		✔	✔	✔	✔	✔	✔

Table 5: Navigation Services

Chosen Implementation

Based on the considerations above, Leaflet is the ideal choice over Google Maps API and Mapbox due to its features and functionality. Unlike Google Maps API, which can be costly for high-usage applications, Leaflet is free of cost. While Mapbox offers powerful features and high customization, Leaflet provides a lightweight, flexible framework that can easily integrate without the learning curve. Its efficient performance and responsive design make it the best choice for our web application implementation. **Image 2** is able to show how Leaflet is able to be integrated with the website design. It is able to have the necessary functionalities on zooming into areas as well as dropping pins to ensure the precise location is found.

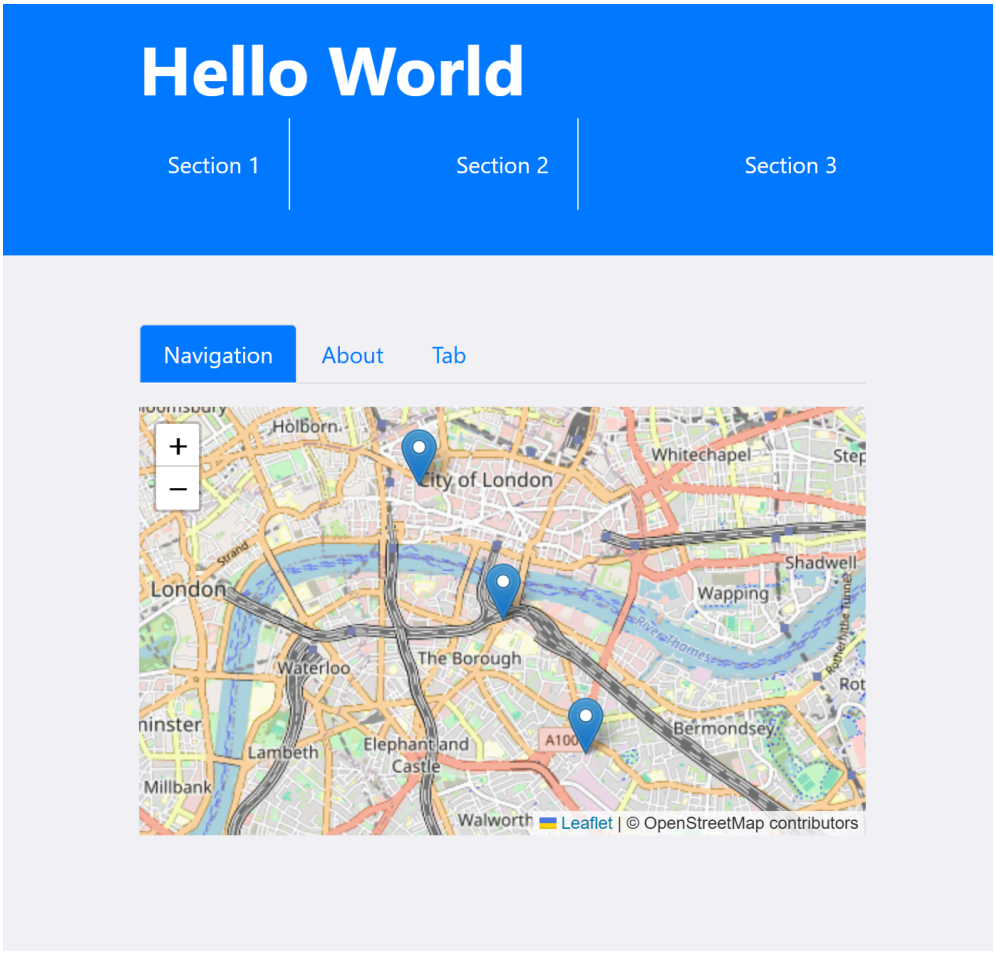


Image 2: Navigation Services Demo

Role-Based Access Technology

The system we are building grants special access to individuals based on their assigned roles within the system. It also allows individuals to contact people based on roles within the system. Different organizations can have their own defined roles, and will also have access to a common bank of roles. These roles are assigned outside their own system, but can still carry meaning within the system. For example, a Navigator can enter someone into the system as a "client", and all the service providers will be able to view them under that role, as well as assign them to a bed or other services, which will also be roles. This logic will be hosted on the server, as seen in Figure 1.

Ideal Solution

Scalability - This solution should be able to be scaled up without editing the basic structure of the program. This would allow it to be expanded to other organizations in the future.

Organization-defined roles - Administrators for an organization should be allowed to create roles to assign to people.

Grant individuals access to resources based on role - The system should allow individuals to access certain pieces of information based on their role or roles within the system. This should probably be some sort of check that looks at each role the person has before allowing them to access certain data or resources, or take certain actions.

Contact people based on role - Users should be able to contact people, or a group of people, based on the role they have.

Assign roles to users - Certain people within the service provider organizations will have the ability to assign roles to individuals.

Services/beds defined as roles - Navigators, as well as employees at the centers, will be able to assign these roles to clients.

Update roles as needed - Certain people within the service provider organizations will be able to edit the roles within their organization.

Possible Solutions and Analysis

Role-based access control (RBAC) is a method of restricting access to data based on the person's role within an organization.[19] Everyone will have at least one role, called the personal role. Their personal role will then be connected to any other roles they are assigned to. Each role will have different levels of access for different organizations.

---

[19] Zhang, E. (2023, May 5). *What Is Role-Based Access Control (RBAC)? Examples, Benefits, and More*. Digital Guardian. https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more

Option	Scalability	Organization-defined roles	Grant access based on role	Contact based on role	Assign and update roles	Beds/ Services as roles	Client approval
A	?	✓	✓	✓	✓	✓	✓
B		✓	✓	✓	✓		

Table 6 - Table compiling analysis of role-based design options

Option A

Although we are designing this system for a specific use case, we should be considering how the scope could be expanded. For now, we are creating the product for the service providers and navigators as they talk to people and gather data. However, in the future, this could be expanded to other companies. Say a Navigator also works at an engineering company that utilizes WorldByMe. The user would need a way to switch from their Navigator role to their engineering role. To make things simpler, the system will default to the last role the person used, with the option to switch their roles if desired.

Organizations are allowed to create their own roles. They can include whatever information should be associated with that role, as well as what information that role has access to. For the purposes of contact, it might be desirable to map certain roles to other roles within other organizations. This mapping could be done by a WorldByMe administrator.

Certain people within an organization will be allowed to assign roles to others. Anyone can create a personal role. That doesn't need to be assigned, it simply represents someone signing up for WorldByMe. The personal roles will be validated with email. If a role is unvalidated, it can still be active in the system. This is because the unsheltered might not have access to an email address to verify their account. For assigning other roles, it depends on what the role is. For example, the role of "Pima County, Unsheltered" will likely be assigned by a Navigator. However, it could also be assigned by someone at the service provider center, if they are simply walking in on their own. A Navigator could also assign a role "Bed" to an unsheltered person they meet. This bed could be located at any service provider in the area. For roles like "Navigator" and "Accountant", these should probably be assigned by someone in HR or the hiring manager. Having the ability to assign roles would itself be a function of someone's role. Certain people will also be able to update roles, including the name, information, and what the role has access to, as needed. This could be necessary for updating organizational roles, as well as updating the demographic information that should be gathered for the unsheltered, which is based on government funding requirements and is subject to change.

Option B

Another way to arrange the information is to create a general organization representing the area. For this project's scope, the area would be Pima County. Within this organization could be smaller organizations representing the service providers and local government. There could be at least one standard role that all service providers would have access to, namely the role of a client, or someone living on the streets that is put into the system by a Navigator. The main reason the client role would be standardized is because of the information that must go into that role. Each Navigator needs to get certain demographic information on each client for funding purposes. This demographic information is defined by the government and is subject to change. When it was updated, they would have access to the updated version immediately.

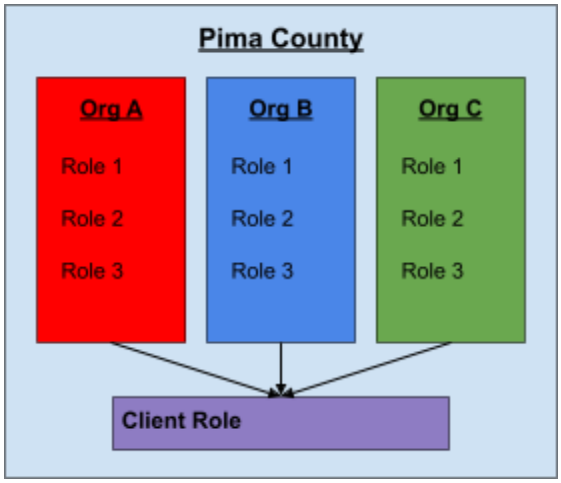


Image 3 - Visual Representation of Option B

Continuing with this method of defining roles outside any particular organization, there would need to be roles defined for a specific organization. Ideally, these roles couldn't be assigned to anyone outside the organization. However, for the purposes of contact, there would be several types of roles that can be mass-contacted. To solve this problem, we could introduce the idea of adding tags. These tags could be added when the role was initially created by an organization. This way, two different organizations create the same role with different names based on their preference (secretary, receptionist), and they can still be contacted collectively if necessary.

Tags could be added to each role to specify the organization it belongs to. When someone goes to assign a role to a person, they are only allowed to see the roles that are part of the same organization they are in. This limits the number of roles a person must be assigned, reducing the amount of corporate overhead necessary for that person to be able to do their job effectively. It should be noted that roles could have multiple tags, thus allowing for general role tags and a tag associated with an organization. Navigators would be responsible for assigning the role of "Unsheltered" to the people they meet, as well as assigning them the role of "bed", or another service.

Roles should be able to be created and updated by people within an organization. When a role is created, it would be assigned a tag and other information that goes along

with the role. They would also specify what types of information people with that role should have access to. This would be accomplished through some sort of UI. Roles can only be edited by people whose role matches the organizational tag of that role. Roles would be able to have multiple tags, so they could be associated with an organization as well as a general role tag. If someone within an organization were creating a role, it would automatically be classified under that organization's role tag.

As far as scalability goes, the larger area could be expanded to encapsulate multiple smaller areas depending on scale. It would also be possible to create a separate area for a different county that could either remain separate or be added to the larger area at some point.

Initial Choice

For this implementation, we will be using option A, as it fulfills the requirements more neatly, specifically for scalability. When considering including other organizations, option B constricts the scalability to service providers, while option A allows other organizations to join if they wish, without having to update the existing architecture. Option A is also the preferred option of our client. As far as implementation goes, we can build some dummy roles to test giving people access based on the role they use to access the server.

Messaging System

Ideal Solution

One of the major goals of this project is to better improve communication between people in different organizations; therefore, the messaging system is a critical feature for HelloWorldByMe. Through collaboration with the client we have identified some characteristics of the messaging system. Firstly, users should be able to chat with other users that they are 'connected' with. This means that users will be able to look up other users in another or their own organization. We also want users to have a log of previous messages, so that they are able to recall previous conversations. Additionally, users should be able to send a variety of messages, so the messaging system must also be able to send and receive various message types, whether it be a chat message, or an image, or any other sort of attachment- we want users to be able to send it. The messaging system should also be scalable, we don't know what the future holds for this project, but we want a solution that can easily be scaled up to accommodate a wide range of users at a given time.

Firebase

The first option we came across was firebase. Firebase is a python supported package that is able to satisfy all of the requirements that we previously identified. Firebase offers support for IOS and Android devices, so the issue of cross compatibility is resolved. Firebase also supports email style messages as well as 'chat' style messages in a manner that supports multiple users at once. The package also stores old messages by default, so all requirements are addressed with firebase. With all of these features, firebase is a compelling option, but firebase also sends messages over a HTTPS connection, ensuring that messages sent are secure.

NiceGUI

NiceGUI is another python package that we came across, although it offers less features than firebase, niceGUI does offer a more robust solution to the chat messaging system. NiceGUI uses TCP sockets to establish a connection with another user, it then uses end-to-end encryption to send the message to the recipient. Although niceGUI is not able to send email style messages, niceGUI does offer a compelling solution to the 'chat' aspect of the messaging system.

Smtplib

To supplement niceGUI's shortcomings in email capabilities, smtplib is a python package that supports an email messaging system. Although being able to send emails was not a rigid requirement, since we are storing a hashed version of a user's password we won't be able to look up a user's password in the event that they forget their password. In this event we will need to send an email to the user for them to be able to reset their password. Additionally, we will need to verify users when they first create an account since a users primary key in the database will be their email address.

Analysis

Package name	Sends email	Store old messages	Multiple message types	scalable	'Chat' style messages
firebase	✓	✓	✓	✓	✓
NiceGUI		✓	✓	✓	✓
smtplib	✓	✓		✓	

Table 7: messaging API options

Based on these options, firebase is the best option going forward. Although niceGUI is a good option with smtplib to support it by sending emails, firebase is best. As laid out in table 7, firebase is able to satisfy all required features of the project. Comfort is a concern with these decisions, however the entire team is comfortable in python, so all of these API's are good choices in terms of the team's comfort but firebase will best assist us in creating a solution, we will consider niceGUI with smtplib as an alternative to firebase in the event that we need to pivot technologies as a team, although we plan to move forward with firebase as our selected technology for the messaging system.

Cybersecurity

Ideal solution

Given the sensitive nature of the information involved, it is crucial to ensure the security of user data. While the client's initial requirements were relatively broad, we identified several key security measures for HelloWorldByMe. First, we will implement string hashing for password verification. We have decided that a SHA256 hash of the passwords will be best for the project and for securing users passwords. Additionally, the messaging system will be secured through end-to-end encryption. Other cybersecurity techniques will be integrated throughout various components of the project. To protect the database, we will establish strict user permissions and employ data validation to prevent potential attacks. With the messaging system encrypted end-to-end, the primary focus for cybersecurity will be on password hashing.

Hashlib & PyCrypto

Our primary solution for the issue of cybersecurity is hashlib. It is a library native to python and offers full support for python 3. This will offer the team and project the least resistance in terms of implementation and functionality. Hashlib offers support for SHA1, SHA224, SHA256, SHA384, SHA512, the SHA-3 series as well as RSA's MD5 algorithm, all of which see use in industry. For our backup option we are considering pycrypto, which is a python package that supports RIPEMD160, as well as the SHA256 hash function. Although the library is no longer maintained, in the event of a cyberattack the library would be obsolete.

Analysis

API name	SHA-256 encryption	Currently maintained	No history of cyberattacks
Hashlib	✓	✓	✓
PyCrypto	✓		✓

Table 8 - comparing hashlib and PyCrypto

Since we mostly decided on the actual hashing function we would be using, the decision relies mostly on if the technology is able to use the SHA256 hashing function. Both of the technologies offer support for it, however hashlib is a native library and supports additional hash functions. Hashlib is overall the better option, as laid out in table 8, PyCrypto is no longer maintained[20]. Which can and will likely lead to issues in the future. With PyCrypto no longer being maintained, and with cybersecurity being paramount to this project, hashlib will be the library the HelloWorldByMe deploys.

[20] Litzenberger, Darsey. "This Software Is No Longer Maintained. | PyCrypto." *Www.pycrypto.org*, 24 May 2022, www.pycrypto.org/. Accessed 24 Oct. 2024.

Programming Languages/Libraries

We require a programming language to program the backend of our website. This language should be able to dialog with the other technologies we have chosen. It must be able to dialog with a database. The language and accompanying libraries will be used for the server side code for connecting to the database, writing the backend logic, and implementing the messaging system, as seen in Figure 1.

Ideal Solution

Comfortability - Everyone on the team should be relatively familiar with the language we choose.

Community Support - Community support is important for the success of the project, as we each have minimal experience with web development.

Cost - Ideally the language should be free and open source.

Interoperability - The language we choose should be able to interface well with the other technologies we have chosen, specifically the database.

Readability - Readability is important for creating the scripts, and for debugging and testing purposes.

Possible Solutions

C++

The C programming language was developed by Dennis Ritchie in the 1970s while working at Bell Laboratories. It was designed to allow software developers to work more closely with the CPU in order to get better performance out of their operating systems. [21]Based on this, the C++ language was later developed as an extension of the C language by Bjarne Stroustrup, and was first released in 1985, but has expanded over time. C++ now has object-oriented functionality, as well as low-level memory capabilities. It is a compiled language. [22]

Python

Python was initially released in 1991. It was created by Guido Van Rossum and began as a hobby project to keep him busy during the holidays. It was a successor to the ABC programming language, which Rossum had also helped create previously. There were several flaws with the ABC programming language, so Python took most of the syntax and features and fixed all the major complaints people had. At the time of release, it had the

---

[21] *Dennis Ritchie, 70, Dies, Programming Trailblazer - The New York Times*. (2021, July 25). Web.archive.org. https://web.archive.org/web/20210725032530/https://www.nytimes.com/2011/10/14/technology/dennis-ritchie-programming-trailblazer-dies-at-70.html

[22] Stroustrup, B. (n.d.). *Stroustrup: FAQ*. Www.stroustrup.com. https://www.stroustrup.com/bs_faq.html#invention

capability for classes and inheritance, as well as functions and exception handling. It is an interpreted language. [23]

Analysis

Technology	Comfortability	Community Support	Cost	Interoperability	Ease of use
C++	✓	✓	✓	✓	
Python	✓	✓	✓	✓	✓

Table 9 - Table compiling analysis of C++ and Python

C++

There are several advantages to C++. Firstly, it is open source, so there is no cost. C++ is an object-oriented language. This provides an easy method of defining roles within the system. Object-oriented programming also provides encapsulation, meaning that people only have access to the data they need, and the rest is hidden to prevent the user from damaging things they shouldn't have access to, and don't strictly need access to. It also allows for inheritance of classes, which could also be useful for this project, especially if we want to have some base roles that other roles build on top of. For example, we could have a base role for all roles, assuming every role needs to have certain information.

Everyone on the team is somewhat familiar with C, which has most of the same syntax as C++. C++ adds classes and an object-oriented structure, which we all have some exposure to as well. However, C++ is less commonly used for web programming, so there is less community support available. This is based on data from the Developer Nation's Q1 2023 report, which stated that the combined number of C and C++ developers around the time were around 13.3 million, compared to Python's 17 million active developers.[24]

There are several libraries for connecting to databases in C++. The main one is called SOCI, and it provides a common method of accessing different types of databases, including MySQL, PostgreSQL, SQLite, and Oracle. The downside to this library is that it is dependent on other dependencies. This library is open source and free. However, it requires a rather complicated download and install on Windows machines, which we are all working with. This is one reason it did not get a check in the "ease of use" category in Table 8. Another reason was that the syntax is generally harder to understand. This makes things a little more complicated. Other options either require a purchased license or don't work for the database options we are considering.

One general advantage of C++ is that it allows for more direct memory management. This is useful in some projects, but in this project it is not necessary. In fact, it somewhat complicates things. The price C++ pays for more direct control over memory

[23] Pramanick, S. (2019, May 2). *History of Python - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/history-of-python/

[24] Fiverr. (2023, August 21). *Python vs. C++: Which Programming Language is Best for Your 2024 Project?* Fiverr.com; Fiverr. https://www.fiverr.com/resources/guides/programming-tech/python-vs-cpp

management is there is less automatic garbage collection. Garbage collection is an automatic form of memory management. This means that if we used C++, we would have to manage the memory ourselves, and run the risk of memory leaks that could use up extra memory and potentially crash the system. We don't need any extensive memory management for this project, so this is not a metric we considered.

Python

Python is a popular open source language, so it is free to use. It is also an object-oriented language, which means we benefit from all the things mentioned above, including inheritance and encapsulation. Python is also an interpreted language instead of compiled. This means the execution is a little slower because it converts each line to machine code as it's being run, but not by much for this project. This also means that the code is more easily updated, which will be important for this project as we work with the client. Python also has more robust automatic garbage collection or memory management. Finally, Python has dynamic typing, which is easier to modify down the line.

As listed in Table 8, everyone on the team is familiar with Python. While classes in Python might be a weak spot for some people, the syntax of Python is generally easier to learn and understand. There is also more community support for Python in general, according to the statistics mentioned above.

There are several Python libraries that would make the project easier. Firstly, a datetime library allows you to access the current time, which will be helpful information to store in our system. Python also has several logging libraries, which will help store data about how the system is working. This is useful for debugging any issues we encounter and storing any problems that may occur in the system. Python also offers socket libraries for network communication.

Python also has several libraries for working with SQL databases. SQLAlchemy is a more general library that allows the user to work with many different databases, and it abstracts the SQL querying so allows you to access the database using Python objects. There is also PyMySQL, which is a simpler library for working with MySQL databases by sending SQL query messages. Finally, there is Psycopg2, which is widely used to interact with PostgreSQL databases using SQL messages. There is also a library for interacting with MongoDB, which is called PyMongo.

One of the major disadvantages of Python is that it has high memory consumption or large amounts of overhead. According to tests done by pythonspeed.com, Python takes about 35 MB of memory to store an array of 1 million integers instead of the expected 8 MB. This is because in Python integers are objects, and thus require more memory overhead to store. However, there is a way around this. The NumPy library allows for arrays that store the integers as just numbers, and not as objects, which reduces the memory overhead.[25] However, in the end it doesn't really matter, since memory consumption is not something we are very concerned about.

Initial Choice

---

[25] Turner-Trauring, I. (2020, July 6). *Massive memory overhead: Numbers in Python and how NumPy helps*. Python⇒Speed. https://pythonspeed.com/articles/python-integers-memory/

Based on all the information, Python is our first choice for the backend programming language for several reasons. Both languages are object-oriented, but Python manages more of the memory for us. For this project, we won't need to manage much memory ourselves, so it makes sense to pick a language that does that for us. Python also has a more extensive list of libraries we can utilize. We will be using the Psycopg2 library for database connections, as we all have experience with SQL commands, and don't want to waste time using SQLAlchemy to abstract things if we don't need to. Finally, everyone on the team is familiar with Python, and the syntax is easy to understand for things we don't have much experience with.

Implementation

PostgreSQL is the most likely database we will select, and there are two different libraries we could use to access that database. They are SQLAlchemy and Psycopg2. SQLAlchemy abstracts tables in a database to class objects. This can make it easier to work with the SQL commands. However, you have to manually create classes representing each table and value, which would be time consuming. It took much longer to get SQLAlchemy code to connect to a postgres database, whereas the psycopg2 library simply sends the SQL query as is, so there was less work to convert things over to objects. It's not necessary to abstract anything as we all have experience with SQL commands, so we will just use the Psycopg2 library. The python code was much easier to understand than the C++ code, and it didn't require extra work to install extra libraries to get it work.

To demonstrate this for our tech demo, we can write a python script using the psycopg2 library, and show it querying a test database we set up using postgreSQL. This python script could both insert and retrieve data to and from the database.

Technological Integration

User Interface

In order to have a comprehensive, functional, and visually appealing user interface, The technologies chosen all catered to their own personal needs while working together to create the interface for the best user experience possible. With HTML5 and CSS as the integration that creates the base of the website, the initial skeleton is able to be created. The application will also integrate technologies like Bootstrap, which is able to keep the applications skeleton to the same across devices and Javascript which adds functionality to the application layout. That makes up the main presentational layer for the front end systems.

The client side framework chosen was React. React works hand in hand with the presentation layer, specifically Javascript to build responsive user interfaces for a dynamic and efficient application and cross platform experience. Lastly, implementing Django will allow us to tie our client and server together to provide functionality to our interface and application.

Centralized Database

We have selected PostgreSQL as our database. It will be connected to our server using Python. Python will interact with the database using the Psycopg2 library, which is a Python library specifically designed to work with PostgreSQL.

Scalable Server

Amazon EC2 would allow the database (Postgres) created for the web application to be hosted on the EC2 instance. EC2 would be the underlying structure to allow both the database and web applications infrastructure to run smoothly.

Navigation Services

Leaflet is the chosen navigation service for our web application. Leaflet is able to work directly as a javascript library in order to implement the desired map functionalities into the application.

Role-based Access Control

The logic for the role based access control technology will be implemented in Python on the server. It will manage requests for data, and will make a decision on the data to return based on the role of the user.

Real-Time Messaging System

We will be implementing a messaging system using the firebase python package on the EC2 server. This will manage any messages that need to be sent from one user to another, or multiple users.

Security Protocol

WorldByMe will be using the hashlib library for security purposes regarding passwords. Any password a person creates will be hashed using this library, and stored in the database.

Conclusion

In recent years, the problem of homelessness has become more noticeable, particularly in areas like Pima County. This up-and-coming crisis has been compounded by the lack of communication and collaboration between local governments and service providers. Therefore, further limits the effects of helping impacted individuals. In an attempt to resolve this issue, non-profit organizations that are service providers hire individuals called navigators. These navigators are people who connect people experiencing homelessness and mental illness with available services. As of right now, there are communication silos that inhibit their efforts to provide help. As illustrated in the figure below **(Figure 2)**, there are four key findings from the analysis:

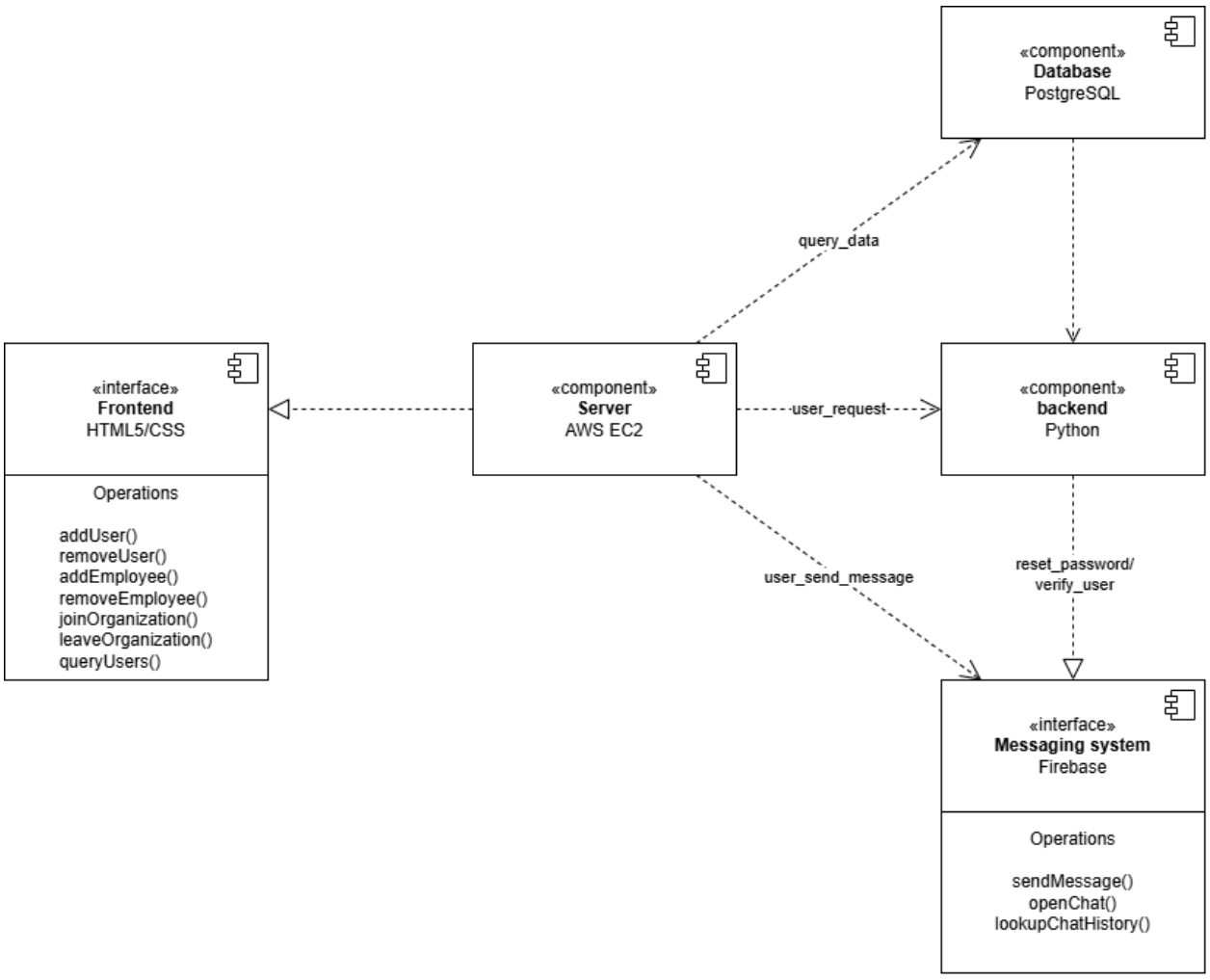


Figure 2: Updated Component Diagram with Technologies Chosen

1. Communication gaps: The current system that is supposed to provide help to local communities is fragmented. Due to how local governments and service providers don't work together, ineffective communication and missed opportunities are

common. **Figure 2** illustrates a centralized AWS EC2 setup that would provide a unifying platform for local governments and non-profit service providers.

2. Technological integration: The suggested solutions above are designed to enhance communication and collaboration within and between organizations. The proposed integrations are designed to streamline the necessary processes of providing services to communities. By implementing the architecture diagram of **Figure 2**, each component infrastructure would maintain the processes connecting them which would ensure that services would be provided to communities efficiently.
3. Cybersecurity concerns: Throughout the analysis section, the importance of implementing security protocols is evident, especially when talking about the handling of sensitive data. Ensuring that security is implemented within the entirety of the infrastructure is important when gaining and maintaining trust with the organizations and clients. Through the frontend and backend components, secure communication lines would be protected as outlined in **Figure 2**.
4. Scalability and Flexibility: For most of the sections of the analysis, the scalability and flexibility of each component have been highlighted, as the application is expected to grow in size as development continues. The overall architectural design shown above should be able to adapt to new user capacities and utilize large data volumes without compromising overall performance.

In conclusion, the main focus of this project is to improve communication and collaboration between local governments and service providers. By ensuring that sensitive data is protected in a careful way and that our overall schema will be able to be adjusted if need be, this project has extreme potential to make an impact on addressing the problems of homelessness and lack of mental illness care in the community.