SOFTWARE ARCHITECTURE OVERVIEW

ZaeTae

Platform for Medical Tourism

June 29th, 2016

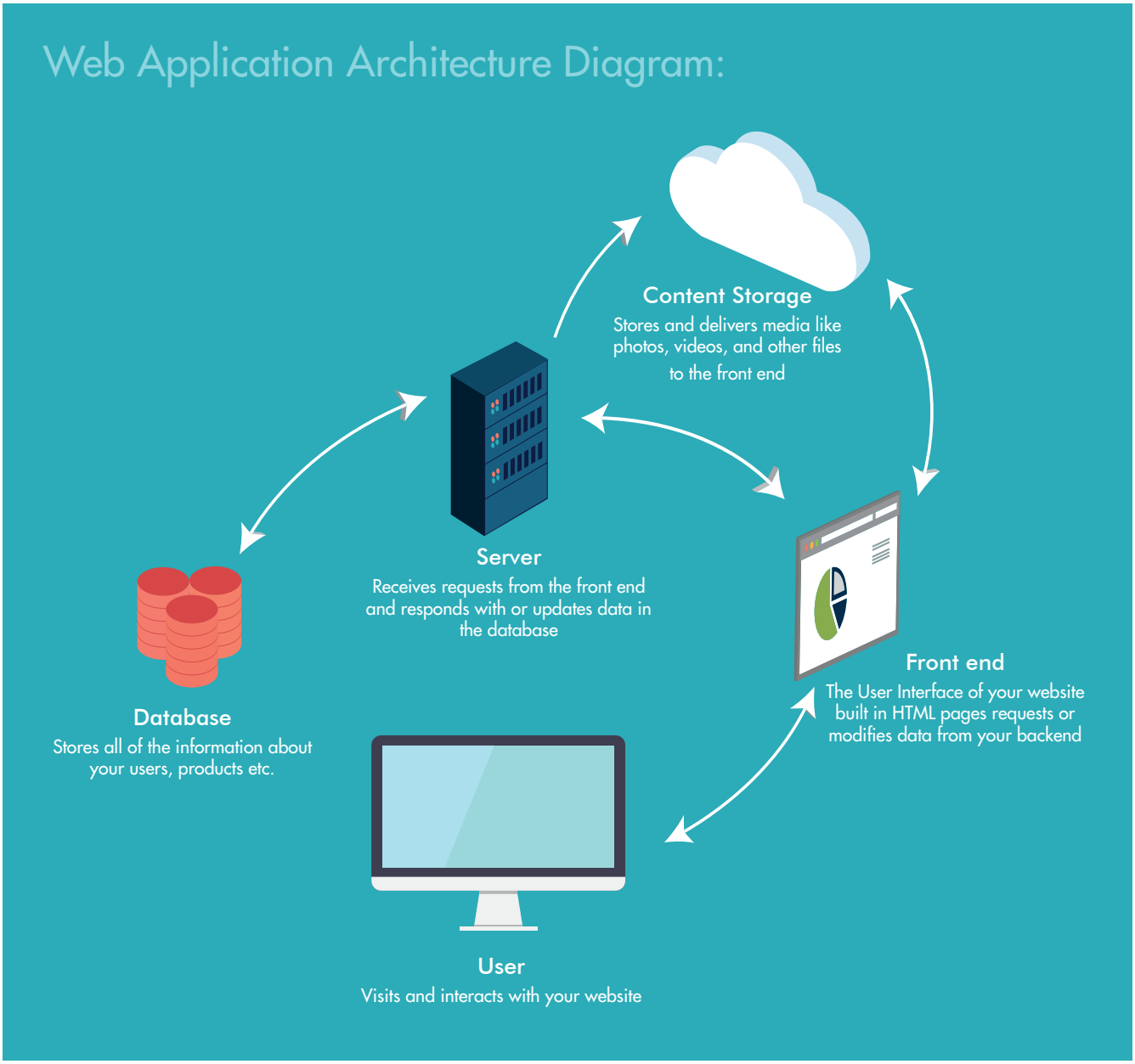
Andrew Wong, SimpleCTO

SOFTWARE ARCHITECTURE OVERVIEW

Project Name	**ZaeTae**
Project Description	Platform for medical tourism.
Prepared On	June 29th, 2016
Prepared By	Andrew Wong, SimpleCTO

ARCHITECTURAL RECOMMENDATIONS

ZaeTae is a modern marketplace web application that will consist of the following components: Backend server, Database, Frontend, Content Storage/Delivery, and a Hosting service for each of these components.



We will make recommendations for each of these components and go into detail on the pros and cons and reasoning behind each decision.

Choosing the Right Backend Language

The backend of a web application can be thought of as the engine of a car. It can't be seen without opening the hood, but it's what powers the machine to get from point A to point B. In a web application, the backend is responsible for processing the data it receives from visitors to your website and either saving information in the database, or sending back a web page for your visitor to view.

When choosing a language to build your backend with, there are a lot of choices: Ruby + Ruby on Rails, Python + Django, Golang, Node + Express, Java, PHP, Elixir + Phoenix, and Scala just to name the more popular ones. If you aren't familiar with these, don't worry, they can be thought of as cars of different makes and models - they're all achieving the same end goal with slight differences in speed, luxury, and form.

To choose the right language, it is important to find the balance between three factors: the needs of your application, the availability of developers in the language, and the maturity of the language's ecosystem.

For example, Elixir + Phoenix can handle a very high number of concurrent users, however, it is a relatively new language so it is very difficult to find skilled developers who have experience with it. Furthermore, the language's ecosystem is not yet developed enough to significantly speed up development time.

Ruby on Rails, on the other hand, provides a middle ground. It is a very mature language with a high availability of skilled developers and a well-developed ecosystem that can shorten the development time by half or even more. Additionally, it's popularity amongst skilled developers results in a lower risk of being unable to find developers to continue the project in the future.

The minor shortcoming of Ruby on Rails is that it's not the most high performance choice available in terms of processing speed and concurrent requests (multiple users accessing your application simultaneously). However, while it may not be the Ferrari of backend languages in terms of speed, it is still highly performant and more than capable to handle the majority of practical web applications.

ZaeTae
Backend Recommendation

ZaeTae is a perfect example of a modern marketplace web application. We expect users to make the following interaction requests to the backend: creating accounts, logging in, updating their profiles, uploading images, saving text information in their profiles, sending messages to providers, browsing procedures, comparning procedures, and searching for procedures.

These types of requests, even at scale, require at most an average request rate, and a low data processing rate. Therefore, because this application does not have high performance constraints, we recommend using a popular modern framework with a developed ecosystem that will be easy to find developers for.

Preferred Backend Choice: Ruby on Rails

Other Excellent Choices: Node + Express, Django + Python

Backends to avoid: Golang, .NET, Java

Reasoning

Ruby on Rails is the perfect choice for this application because it has a huge ecosystem of existing modules (called gems) and documentation available which will greatly speed up the development process. It is also one of the easiest language + framework combinations to develop in and was designed exactly with this type of application in mind. The only drawbacks of Ruby on Rails are performance constraints involving extremely high volume concurrent requests, or heavy duty computing power requirements (i.e. image processing or machine-learning algorithms). However, this is not a concern for most practical web applications including a marketplace application like AsapNinja. Therefore we recommend Ruby on Rails because its advantages greatly outweigh its disadvantage.

Popular applications built on Ruby on Rails: AirBnb[1], Twitter[2], Shopify[3]

[1]AirBnb's Tech Stack http://stackshare.io/airbnb/airbnb
[2] Twitter's Tech Stack http://stackshare.io/twitter/twitter
[3]Shopify's Tech Stack http://stackshare.io/shopify/shopify

Backend Languages

Common Backend Programming Languages:

Python: A modern language that is easy to build upon with a great track record and ecosystem. Popular web development frameworks include Django and Flask. Python is widely used in data science and machine learning as well as in web applications.

Wordpress: A content management platform based on PHP that can be very well suited to building a quick and cheap MVP. If you are building an ecommerce news, or blogging platform with little customization, Wordpress is ideal.

Node.js: Allows developers to use JavaScript, the language of the web browser, on the backend of a web application. While this language is fully-featured, it tends to be used mainly when the backend is small or uncomplicated.

Java: One of the most popular languages during the early days of the web, with a lot of developers, especially overseas. Java is still used to build high-performance software, but it takes a lot longer to write code than in newer languages.

PHP: The go-to language for web 1.0 applications, with a lot of developers available at bargain prices. PHP is generally falling out of favor and should therefore be avoided unless you have a low budget.

Ruby on Rails: A web application framework built on the Ruby programming language that was created specifically for companies that want the best combination of development time and performance/quality.

Choosing the Right Frontend Framework

The frontend of your web application is how your user interacts with and gets use from the web application. It's everything they see interact with in their web browser. You can think of the frontend as every part of the car that isn't the engine. It's the steering wheel, the driver's seat, and the slick coat of paint.

Choosing the right framework for your frontend follows similar guidelines to choosing the right backend. You want a framework that will support the functionality your application needs, as well as one that also has a rich ecosystem and a high availability of experienced developers.

The balance with building the frontend is knowing the constraints of your budget and deadlines with the features your application needs. In most cases, we advise building the absolute bare minimum features in the beginning to get your application in front of your users. Once you have an audience, all the bells and whistles can be added from there. This way you don't waste time and money building features on your frontend until they're directly providing value to your users.

It's like with the analogy of the car, in the beginning you only need to give your users a steering wheel and a driver's seat so they can experience the essence of your invention - getting from point A to point B. Once your users have fallen in love, then you can continue to wow them by adding power locks and windows, leather seats, and a high fidelity sound system.

ZaeTae Frontend Recommendation

The majority of the frontend can be built with the static HTML and CSS theme/template already purchased. HTML and CSS are the building blocks of every web page - they are the text, images, buttons and graphics that we see. The HTML places the objects there, and the CSS styles and positions them in the correct places.

Any static web page, meaning pages that don't have any interaction other than clickable links can be built simply with just HTML and CSS. The home page and the procedure booking page will most likely require custom pages to be built instead of using the template. These pages can be built in HTML and styled with CSS.

For the procedure sorting page we recommend using a modern frontend framework to improve the performance and user's experience. A framework is simply a set of tools to make building a complex user interface easier for the developer. For frontend frameworks the popular ones are all great: React, Ember, Backbone, or Angular. All of these frameworks are highly developed and will come with a reasonable availability of experienced developers as well as a thriving ecosystem and capability. When choosing a frontend framework, the location you're expecting to look for developers in plays the biggest role.

Preferred Frontend Choice: React

Other Excellent Choices: Angular, Backbone, Ember

Popular applications built with React: Facebook[4], Airbnb[5], Uber[6]

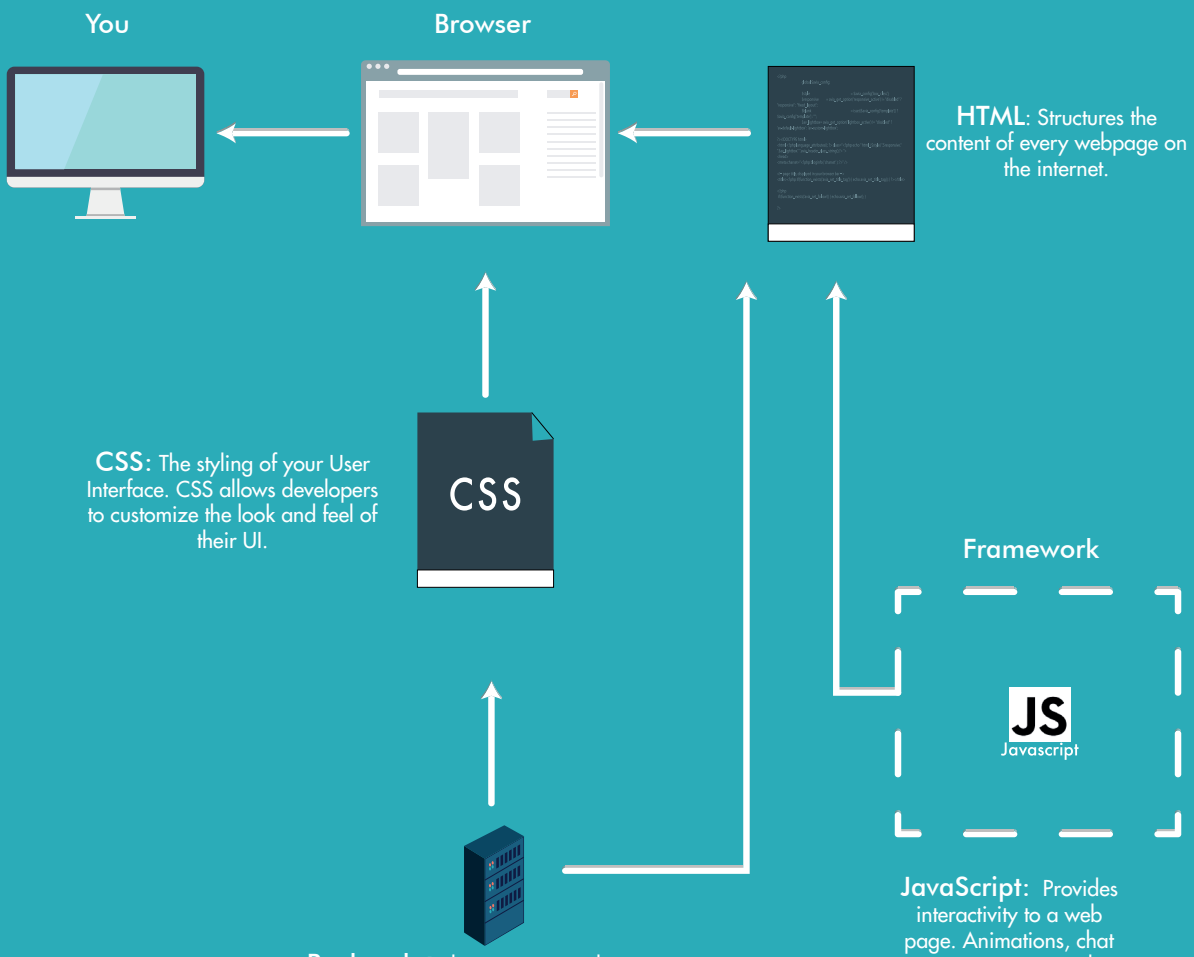
Reasoning

The popular frameworks are all excellent solutions for building **ZaeTae**. Because you're based in San Francisco, we recommend using React. React is highly popular in the San Francisco Bay Area which will make finding developers easier both now and in the future. It is also extremely well equipped to handle highly interactive user interfaces and has a fully mature ecosystem.

[4]Facebook's Tech stack http://stackshare.io/facebook/facebook
[5] Airbnb's Tech Stack http://stackshare.io/airbnb/airbnb
[6]Uber's Tech Stack http://stackshare.io/uber/uber

Frontend Frameworks

Front End Architecture Diagram:



Choosing the Right Database

A database is your web application's memory - it's where all the information is saved. Your user's information, images, the procedures they've booked, and every other piece of information is stored in your database. The majority of databases come in two different types: relational and NoSQL.

Relational databases were the original type of database created and are fast and optimized for very structured data[7]. You can think of relational databases as large spreadsheets with many rows and columns. As such, relational databases shine when all of the items in the spreadsheet have the same structure.

On the other hand, NoSQL databases shine for datatypes that are unpredictable, change often, or rely heavily on interconnected data[8]. These types of databases shine when we aren't sure of the structure of data being saved in use cases like document storage or social graphs. It's also important to note that NoSQL databases are still less standard than traditional relational databases, making it more difficult to find developers who are comfortable with them. As such, for most general use cases a relational database is the right tool for the job. Common relational databases are either MySQL or PostgreSQL.

Preferred Database Choices: MySQL or PostgreSQL

Databases to avoid: MongoDB, Neo4j, NoSQL

Reasoning

Relational databases are the simplest to use, and are perfect for highly structured data. ZaeTae will access and save information in a highly structured way making it perfect for a relational database. Seeing as all of the data needed for ZaeTae will be structured and there will be no need for a distributed database system, using a NoSQL database such as MongoDB would be using the wrong tool for the job.

Choosing the Right Media Storage Solution

Most web applications not only need to store text, but also need to store and retrieve images, videos, and other media types. These types of media can't effectively be stored in the database like your other information (user emails, item descriptions, etc.) and therefore must be stored in a proper media storage solution. This way, when a user views your webpage, your backend can send the viewer the right images along with the rest of the web page.

Preferred Media Solution: Amazon S3

[7] A Relational Database Overview https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html
[8] SQL vs NoSQL Differences https://www.sitepoint.com/sql-vs-nosql-differences/

Reasoning

Amazon S3 is an industry standard for media storage, and is very easy to set-up and manage with the other technologies you'll be using.

Choosing the Right Hosting Platform

Once your web application has been developed, it needs to be uploaded somewhere so that when users go to your website they can interact with your application. Hosting companies essentially have rooms full of powerful computers where they host web applications like yours. To do this, they lease a portion of their computing power to many different companies running web applications like **ZaeTae.** When a user visits your website, they will be connected to the hosting company and their supercomputers will begin processing the information and sending the appropriate web pages to your visitors.

Preferred Hosting Choice: Heroku

Another Excellent Choice: Google App Engine

Hosting to Avoid: Amazon Web Services (AWS)

Reasoning

We recommend hosting **ZaeTae** on Heroku. Heroku takes away a lot of the complexity involved in hosting a web application making it perfect for small and medium companies. While AWS is more popular amongst larger tech companies, it is much more expensive to set up, and in addition will require a DevOps professional to oversee. The overhead cost of hiring a full-time employee to maintain your servers on AWS is simply not practical for a company in its early-stages. Heroku takes away a lot of these maintenance and management concerns and can be accessed by any experienced developer.

Payment Processing

Because **ZaeTae** is a marketplace, it will require a payment processor to handle the transactions of renting items between locals and travelers. Payment processing platforms are extremely complex and have a wide-range of security concerns involved. Two popular payment processing services we recommend are Stripe and Braintree. Stripe and Braintree will both allow your web application to accept credit card payments, and will store your user's credit card information in a secure manner.

Security Concerns

Because your marketplace will be handling credit card information which is highly-sensitive data you must ensure all transactions and authentication requests happen over HTTPS[9]. Additionally a secure authentication solution must be properly implemented to ensure your user's passwords are safely encrypted and meet security strength standards. Any competent developer should know how to handle these constraints.

[9]HTTP to HTTPS | What is a HTTPS Certificate - SSL https://www.instantssl.com/ssl-certificate-products/https.html

DEVELOPMENT TIMELINE

This section provides a breakdown of the features that should be developed for **ZaeTae**. It also provides a general overview of how the development should be approached, as well as providing recommendations for paying developers in a process that protects you.

Total Estimated Development Time[10]: 148 Hours

PROJECT DELIVERY

This project is broken down into two sprints, each of which will result in a deliverable milestone that is complete and functional. We recommend splitting the payment into 3 equal payments - initial payment, after the first sprint is delivered, and after the second sprint is delivered. The deliverable after each sprint should be fully functional in case you must change developers any reason.

DEVELOPMENT BREAKDOWN

SPRINT 1	
Backend	
Initial Application Setup	2 hours
Database Migrations	2 hours
Authentication	8 hours
Users (Patients)	6 hours
Users (Providers)	8 hours
Procedures	6 hours
Hosting/Deployment	6 hours
Frontend	
Homepage	8 hours
Users (Patients) Sign Up	4 hours
Users (Patients) Profile	4 hours
Provider/Procedure New View	8 hours
Procedure View	12 hours
QA and Debugging	2 hours
Total Development	76 hours

SPRINT 2	
Backend	
Procedure Comparison	4 hours
Payment Integration	12 hours
Messages	8 hours
SSL/Security Setup	6 hours
Frontend	
Procedure Search/Sort	16 hours
Comparison View	6 hours
Image Lightbox	6 hours
Messages Frontend	8 hours
QA and Debugging	6 hours
Total Development	72 hours

[10]This is a rough estimate based on the description of the application provided. Actual development time can vary greatly depending on feature requirements, the experience of the developers, and the development methodologies used.

Finding Developers

Excellent work, you're now ready to start looking for the right development team to build your software. Teams can range from one-man shops to global organizations with hundreds on staff. Prices can also range from the low thousands to a few hundreds of thousands. Finding the right team is all about finding a team you can trust. While a general rule of thumb is that you will receive higher quality software the more you pay, it's not always true in the below $25,000 price range.

Look for a development team who has your company's interests in mind, and will be communicative with you so you're always in the loop with the development process. The last thing you want is to waste your time and money with a developer who doesn't spend the time to figure out the specifics of the software you want built.

There are many excellent developers out there, and conversely there are dishonest developers and agencies out there as well. By reading this document, you now have a greater understanding of the underlying process involved in developing your application. While this knowledge will help you navigate the development landscape, **the surest way to find a good developer is through a recommendation** from a trusted source.

Sample Script

For your convenience, you may use the following sample script when contacting developers and development agencies.

"Dear [the developer or agency's name],

How are you? I'm writing to see if you're available to help me develop the software for a company I'm starting. The software is a typical modern web application where locals can rent items to travellers via a marketplace. I have already worked with SimpleCTO to write out the specifications and technology stack required for the application.

The software will be built on Ruby on Rails and React for the frontend. We estimate the project should take around 150 hours from start to finish. Please let me know if you're available and I can send over the project specifications along with more detail of the project.

Thank you very much,
[Your name]"

APPENDIX-RESOURCES

1. MVC Architechture - Google Chrome https://developer.chrome.com/apps/app_frameworks
2. RESTful Web Services: The basics http://www.ibm.com/developerworks/library/ws-restful/
3. Why Use Ruby on Rails? A Senior Dev Explains the Benefits https://www.toptal.com/ruby-on-rails/after-two-decades-of-programming-i-use-rails
4. AngularJS 2.0 Status and Preview http://ng-learn.org/2014/03/AngularJS-2-Status-Preview/
5. A Relational Database Overview https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html
6. Rails Hosts: Amazon AWS vs. Digital Ocean vs. Heroku vs. Engine Yard https://www.airpair.com/ruby-on-rails/posts/rails-host-comparison-aws-digitalocean-heroku-engineyard
7. HTTP to HTTPS | What is a HTTPS Certificate - SSL https://www.instantssl.com/ssl-certificate-products/https.html

Database Models

Based on your description of **ZaeTae**, your database will use the following relational models to represent your user interactions:

User (type: Patient)

- Email
- Password
- First Name
- Last Name
- Gender
- Birthdate
- Photo
- Location
- **Messages**
- **Saved Procedures**

User (type: Provider)

- Email
- Password
- Facility Name
- Facility Type
- Facility Mission Statement
- Location
- **Procedures**

Procedure

- Type
- Statement
- Photos
- Location
- Price
- **Amenities**
- **Doctors**

SimpleCTO
www.simplecto.com