PetPeople

PetPeople

Social network

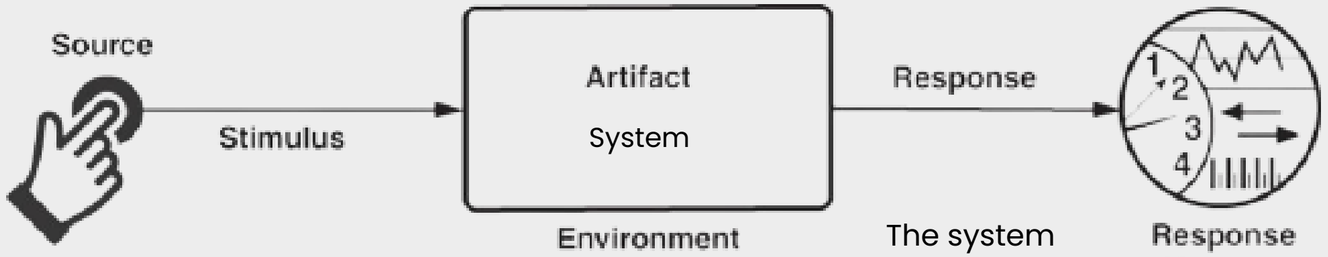
MarketPlace

Information Hub

For your pet

Framework

- Number of concurrent requests (Performance)



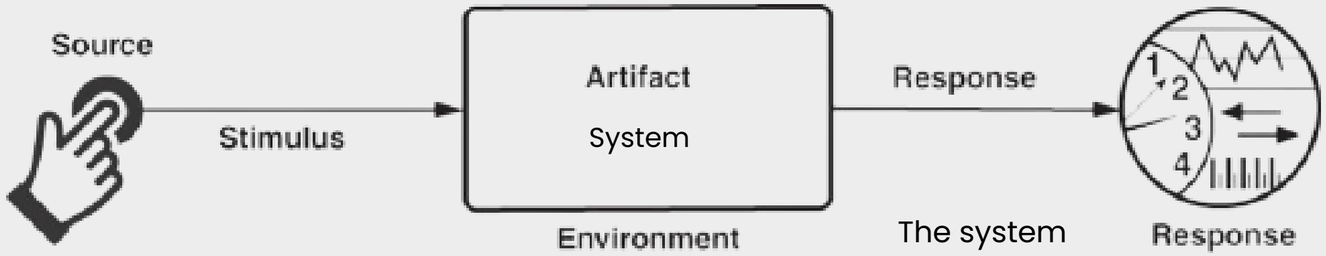
The users	perform concurrent requests	Normal operation	The system responds by handling each request independently and assynchronously	Response Measure: The system checks if the system can handle at least 800 requests per second

- **Express** is known for its simplicity and speed, making it an excellent choice for building lightweight, high-performance applications.

- **Spring** offers a broad range of features and modules, making it suitable for building complex applications that require scalability and high performance.

Framework

- Up time (Availability)

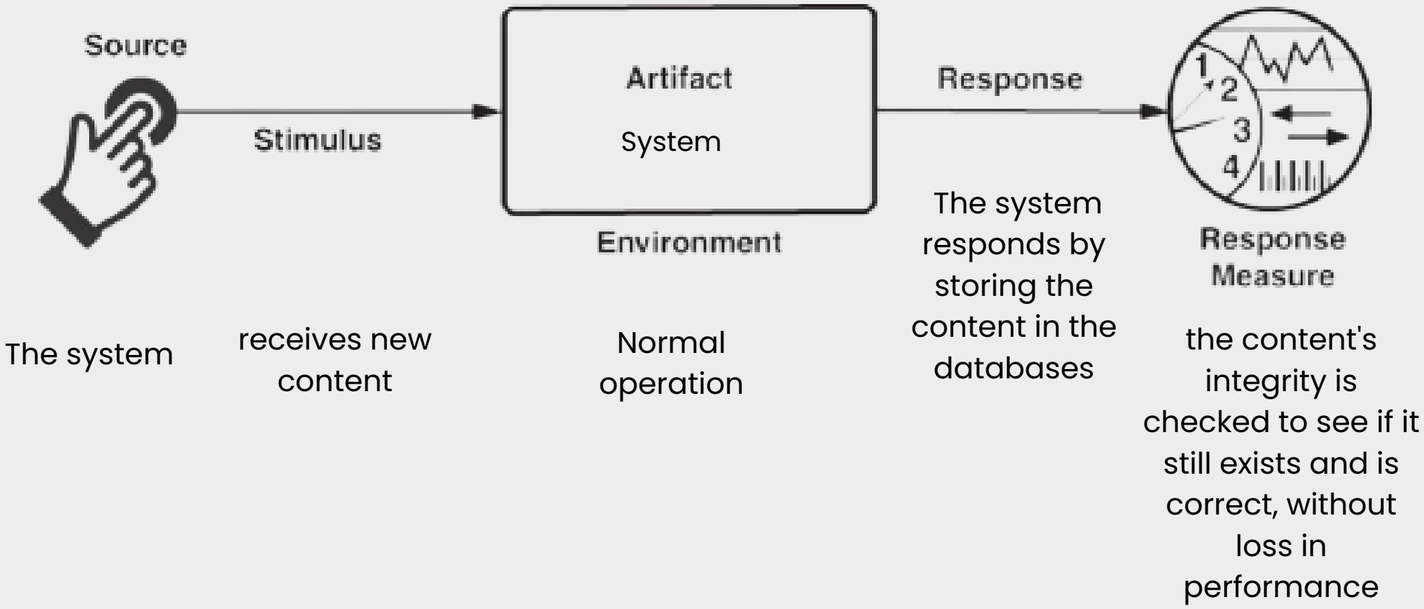


The system's software	fails/crashes	Normal operation	The system responds by entering a previous safe state.	Response Measure
				The system checks if the restart occured in less than 1 minutes after the initial crash.

- **Spring, Django, Laravel,** and **Ruby on Rails** all offer built-in fault tolerance features, while **Express** requires additional third-party libraries and modules. However, the specific fault tolerance features and configuration may differ between frameworks.

Framework

- Sustain an annual content growth of at least 200% in content (Scalability)

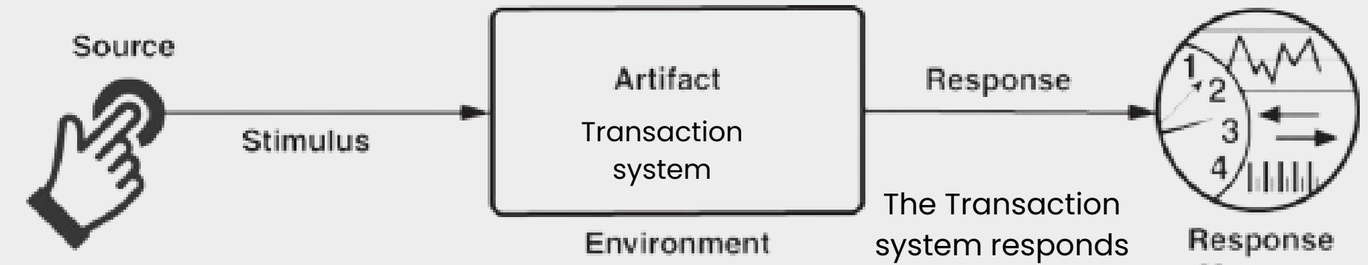


- **Express** is a lightweight and flexible framework that can be easily integrated with different databases and libraries, making it a popular choice for building web applications.

- **Django** has a well-established architecture and comes with a lot of built-in features and plugins, making it easy to develop and deploy web applications.

Framework

- Number of transactions (Performance)



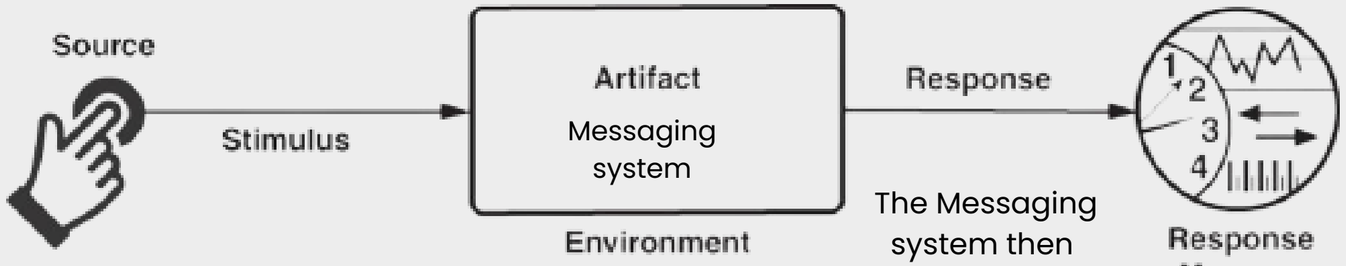
The users	perform transactions simultaneously	Normal operation	The Transaction system responds by orchestrating the incoming transactions	Response Measure: The system checks if the system can hold up to 1000 transactions per minute

- **Spring** is known for its support for distributed systems and microservices, making it a good choice for building transactional systems that can handle a large volume of transactions.

- **Express**, on the other hand, can handle a high volume of requests concurrently, making it a suitable option for systems that require high concurrency.

Framework

- Secure messaging (Security)



The user | sends a private message to another | Normal operation | The Messaging system then responds by safely sending the message encrypted to the other user | it checks if no messages are ever intercepted by unwanted third parties

- **Spring and Django** are known for their strong security features, making them a good choice for building secure messaging systems. **Express, Laravel, and Ruby on Rails** can also be used, but you may need to add additional security features to ensure that messages are not intercepted by unwanted third parties.

Framework

- Low Funds
  - As a Client that is starting a business, I want to save the most I can seeing that I do not have a lot of funds

Considering that we are the developers of the platform, for cost savings and a faster development process, the best framework for us would be one with which we are familiar, such as **Express** and **Ruby on Rails**.

Framework

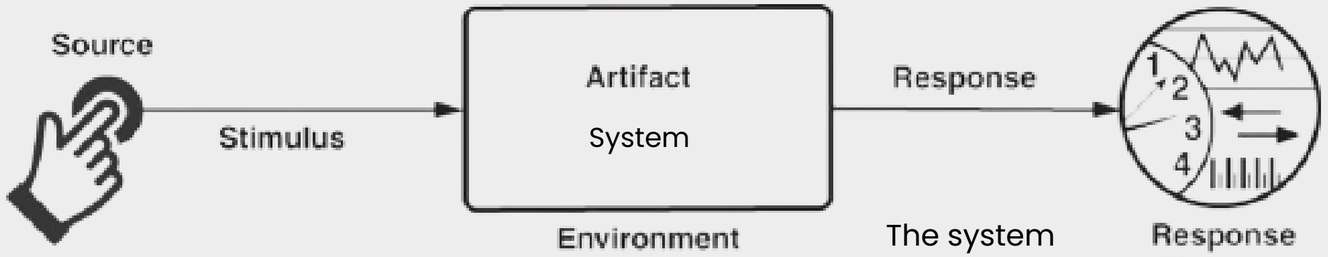
Decision

Express

JS

Database

- Number of concurrent requests (Performance)

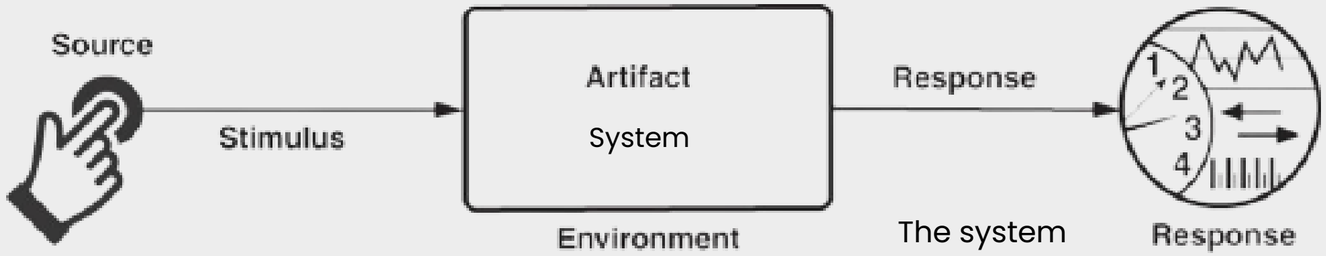


The users	perform concurrent requests	Normal operation	The system responds by handling each request independently and assynchronously	Response Measure: The system checks if the system can handle at least 800 requests per second

Document-oriented databases are designed for high performance and scalability, and they can handle concurrent requests efficiently. They store data in a flexible, semi-structured format, making them a good fit for web and mobile applications that deal with large volumes of unstructured or semi-structured data. **MongoDB**, in particular, has features like sharding and replication, which allow it to scale horizontally and handle high volumes of traffic.

Database

- Up time (Availability)

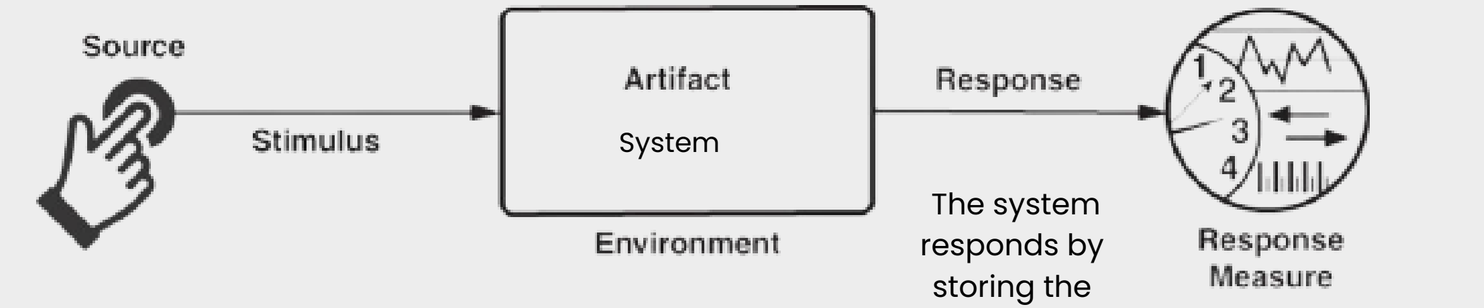


The system's software	fails/crashes	Normal operation	The system responds by entering a previous safe state.	The system checks if the restart occured in less than 1 minutes after the initial crash.

MySQL, **PostgreSQL**, and **Oracle** all have relatively mature mechanisms for recovery, such as replication and clustering, which can provide high availability and reduce downtime. **MongoDB** and **Redis** have similar mechanisms, but they are less mature and have some limitations.

Database

- Sustain an annual content growth of at least 200% in content (Scalability)

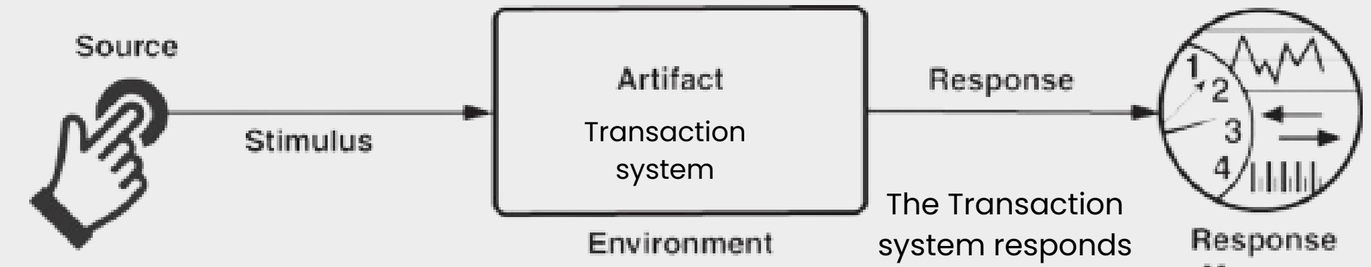


The system | receives new content | Normal operation | The system responds by storing the content in the databases | the content's integrity is checked to see if it still exists and is correct, without loss in performance

As we are building a social media platform that needed to store and process large amounts of user-generated content, such as photos and videos, a **NoSQL** database like **MongoDB** or **Cassandra** would be a better fit because it can handle large amounts of unstructured data and scale easily to meet the needs of a growing user base.

Database

- Number of transactions (Performance)



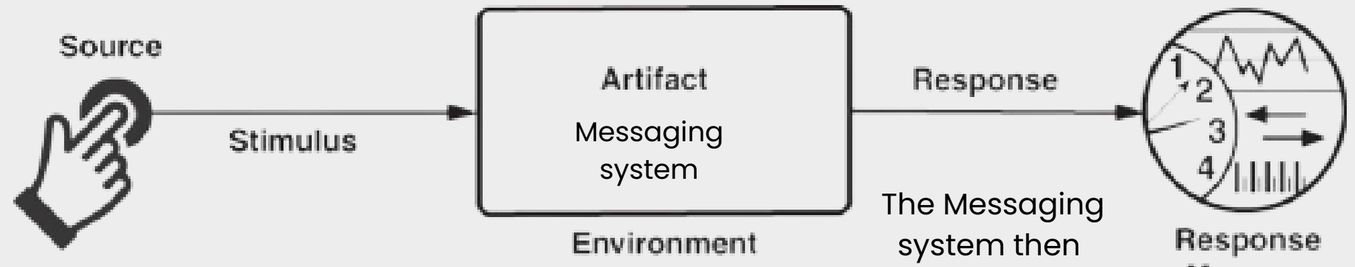
The users	perform transactions simultaneously	Normal operation	The Transaction system responds by orchestrating the incoming transactions	The system checks if the system can hold up to 1000 transactions per minute

PostgreSQL and **MongoDB** are strong contenders for handling high volumes of concurrent transactions, with **PostgreSQL** being a good choice for traditional relational workloads and **MongoDB** being a good choice for handling complex data structures and distributed transactions.

In terms of scalability, **MongoDB** is often considered better because of its horizontal scaling capabilities and sharding feature, which allows for distributing data across multiple machines.

Database

- Secure messaging (Security)



The user	sends a private message to another	Normal operation	The Messaging system then responds by safely sending the message encrypted to the other user	it checks if no messages are ever intercepted by unwanted third parties

PostgreSQL, **MongoDB**, and **Oracle** offer more comprehensive built-in security features, including encryption capabilities. These databases provide better options for ensuring the confidentiality and integrity of the messages.

Database

- Low Funds
  - As a Client that is starting a business, I want to save the most I can seeing that I do not have a lot of funds

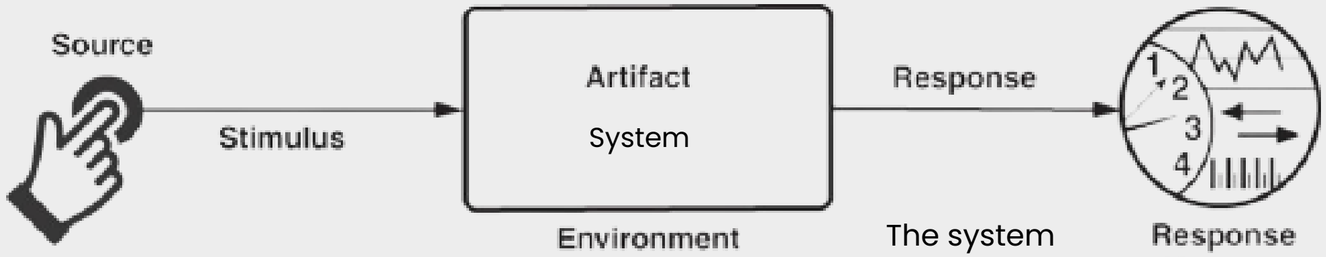
Considering that we will be the developers of the platform, as mentioned previously in the framework section, in order to save funds and attain a faster development, the most suitable database would be one with which we are familiar, such as **MongoDB** or **PostgreSQL**.

Database

Decision

Architecture Type

- Number of concurrent requests (Performance)



perform concurrent requests

Normal operation

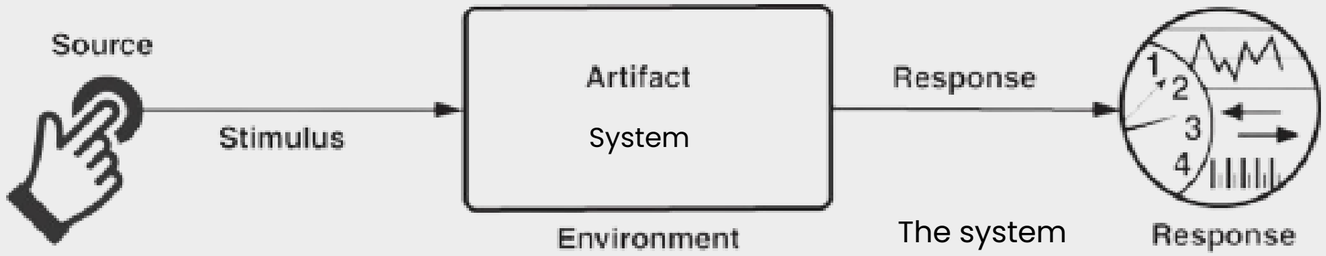
The system responds by handling each request independently and assynchronously

The system checks if the system can handle at least 800 requests per second

Event-driven architecture and **space-based architecture**

Architecture Type

- Up time (Availability)

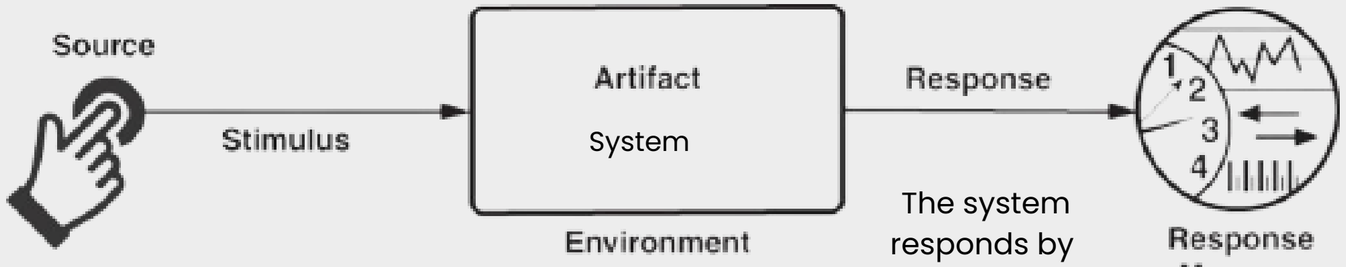


The system's software	fails/crashes	Normal operation	The system responds by entering a previous safe state.	Response Measure
				The system checks if the restart occured in less than 1 minutes after the initial crash.

A **microservices architecture** or **service-based architecture** can be beneficial since these architectures can provide fault isolation and allow for components to be restarted independently, reducing the chances of a full system failure.

Architecture Type

- Sustain an annual content growth of at least 200% in content (Scalability)

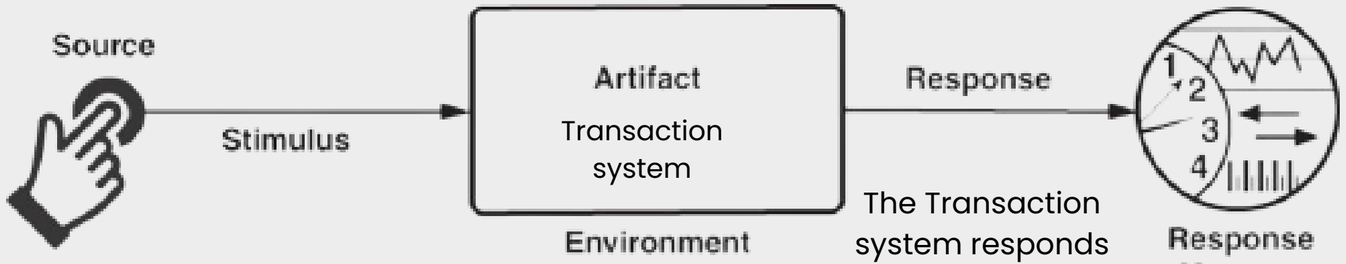


The system | receives new content | Normal operation | The system responds by storing the content in the databases | the content's integrity is checked to see if it still exists and is correct, without loss in performance

Event-driven architecture and **space-based architecture**

Architecture Type

- Number of transactions (Performance)



The users perform transactions simultaneously

Normal operation

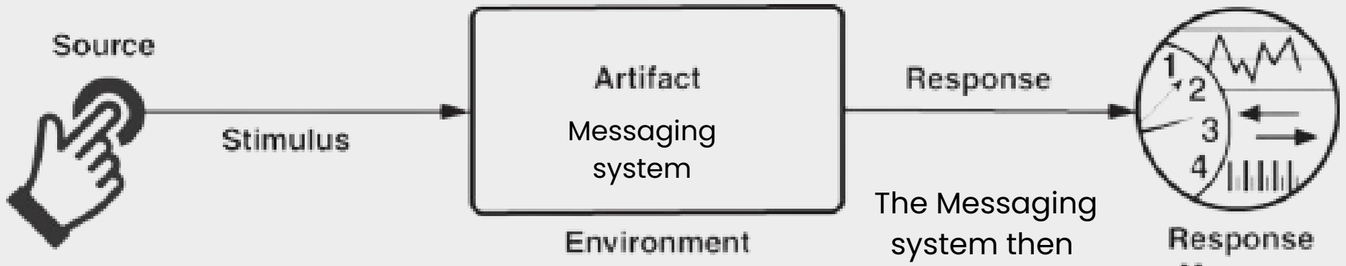
The Transaction system responds by orchestrating the incoming transactions

The system checks if the system can hold up to 1000 transactions per minute

Event-driven architecture and **space-based architecture**

Architecture Type

- Secure messaging (Security)



The user	sends a private message to another	Normal operation	The Messaging system then responds by safely sending the message encrypted to the other user	it checks if no messages are ever intercepted by unwanted third parties

Service-Based Architecture or a **Microservices Architecture**. In both of these architectures, each service is responsible for performing a specific task, and communication between services is done through APIs, which can be secured using encryption and authentication.

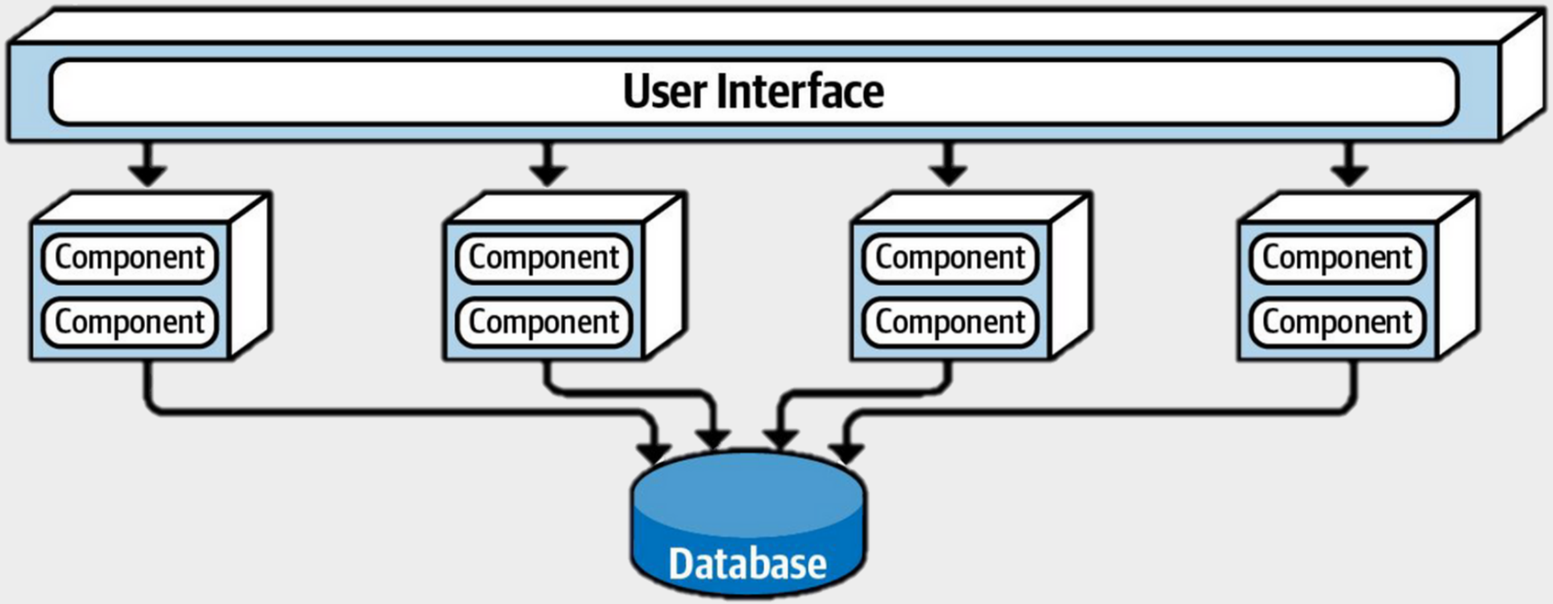
Architecture Type

- Low Funds
  - As a Client that is starting a business, I want to save the most I can seeing that I do not have a lot of funds

Layered architecture, **pipelined architecture**, **microkernel architecture** and **Service-based architecture**

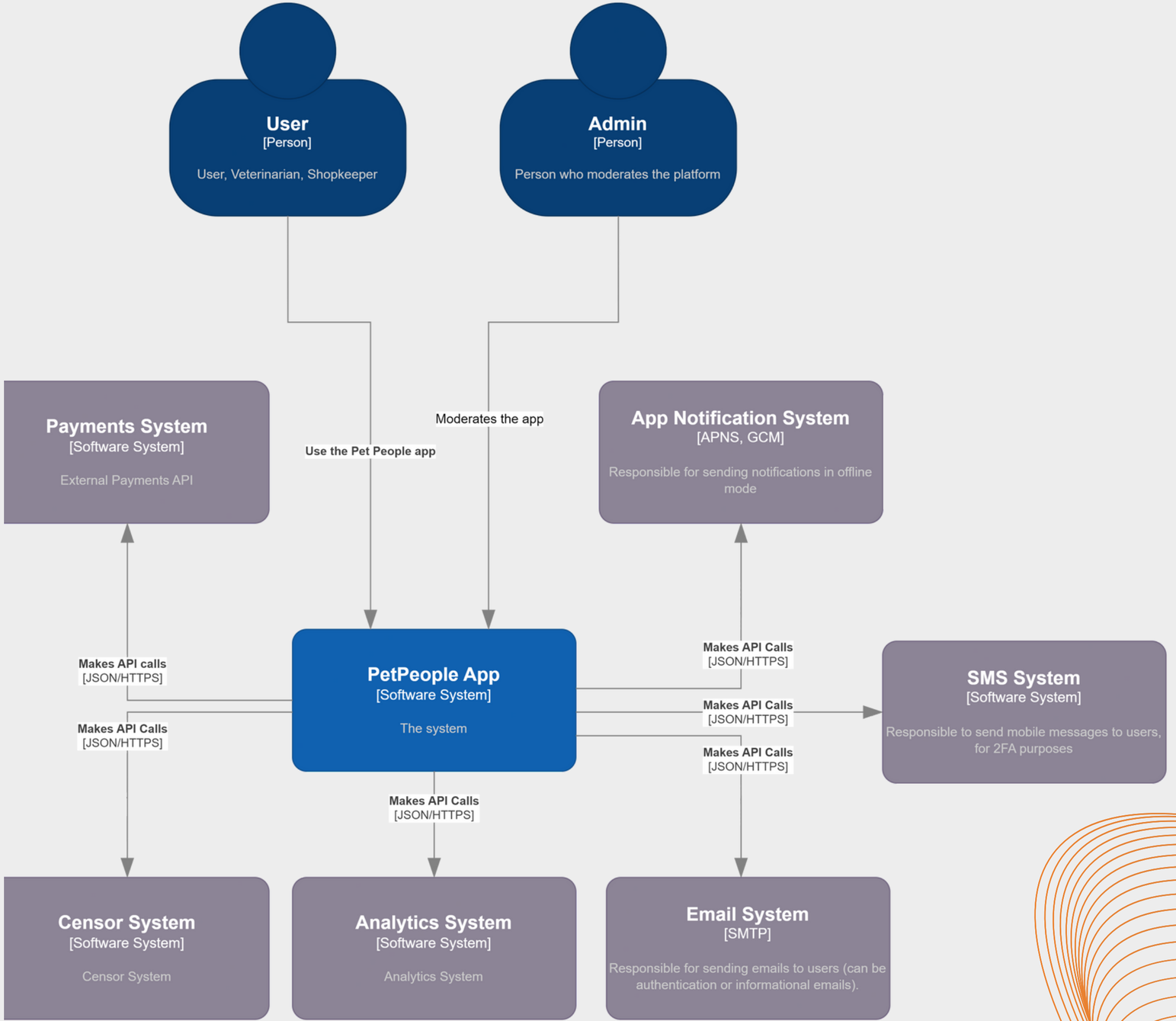
Architecture Type

Decision

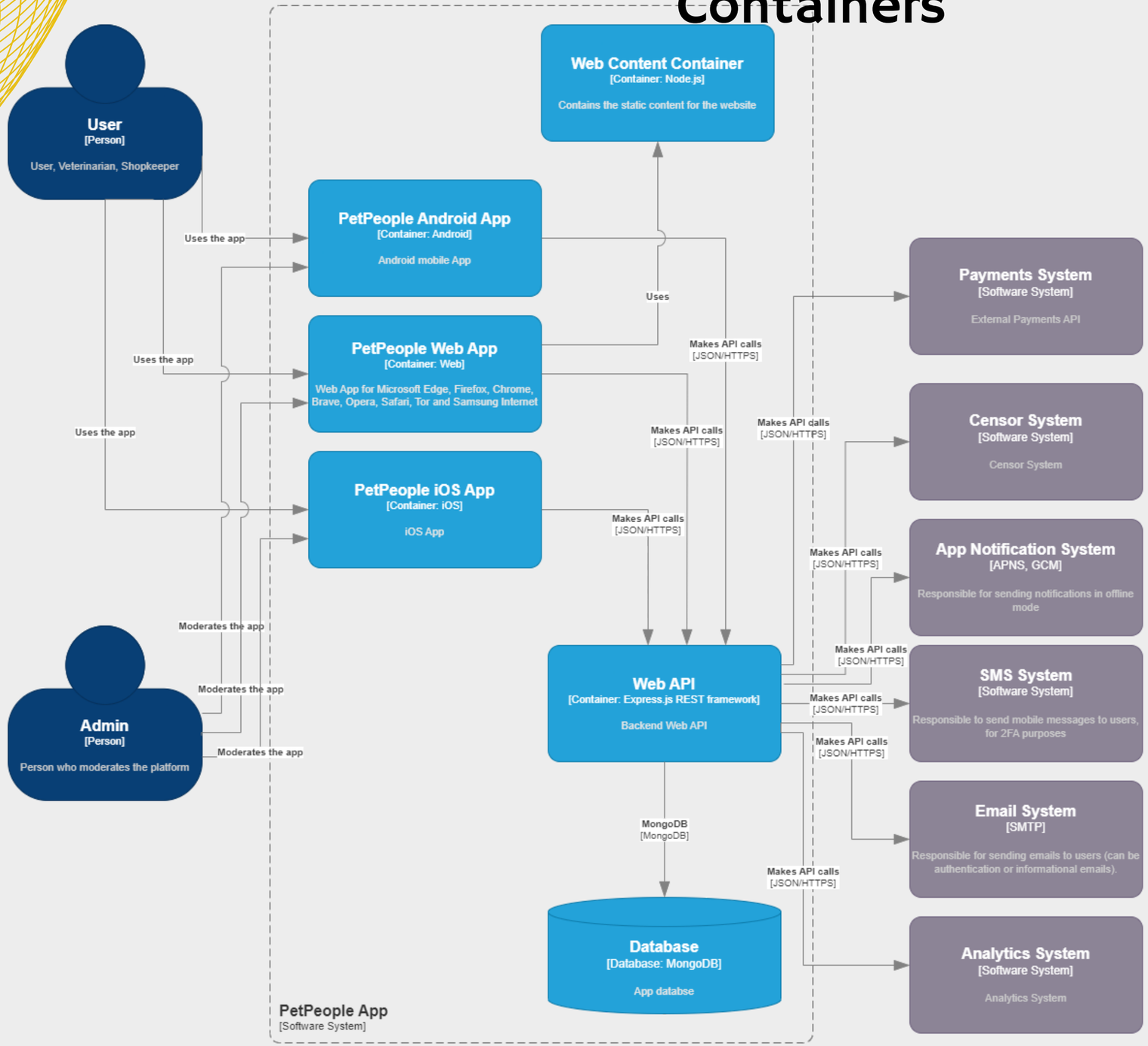


Service-Based architecture

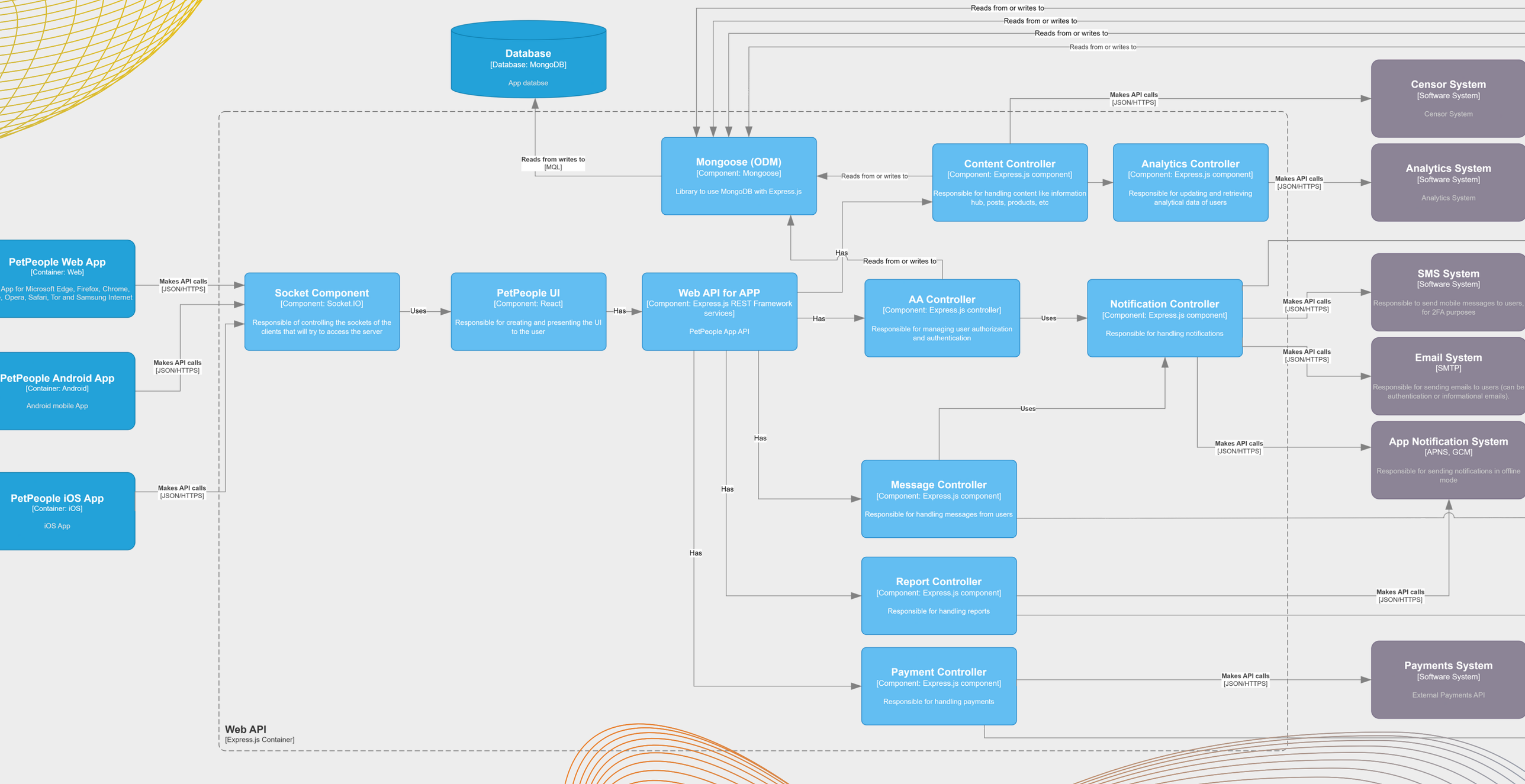
Context


User
[Person]
User, Veterinarian, Shopkeeper
Admin
[Person]
Person who moderates the platform
Payments System
[Software System]
External Payments API
Use the Pet People app
Moderates the app
App Notification System
[APNS, GCM]
Responsible for sending notifications in offline mode
Makes API calls
[JSON/HTTPS]
PetPeople App
[Software System]
The system
Makes API Calls
[JSON/HTTPS]
Makes API Calls
[JSON/HTTPS]
Makes API Calls
[JSON/HTTPS]
Makes API Calls
[JSON/HTTPS]
SMS System
[Software System]
Responsible to send mobile messages to users, for 2FA purposes
Makes API Calls
[JSON/HTTPS]
Censor System
[Software System]
Censor System
Analytics System
[Software System]
Analytics System
Email System
[SMTP]
Responsible for sending emails to users (can be authentication or informational emails).

Containers

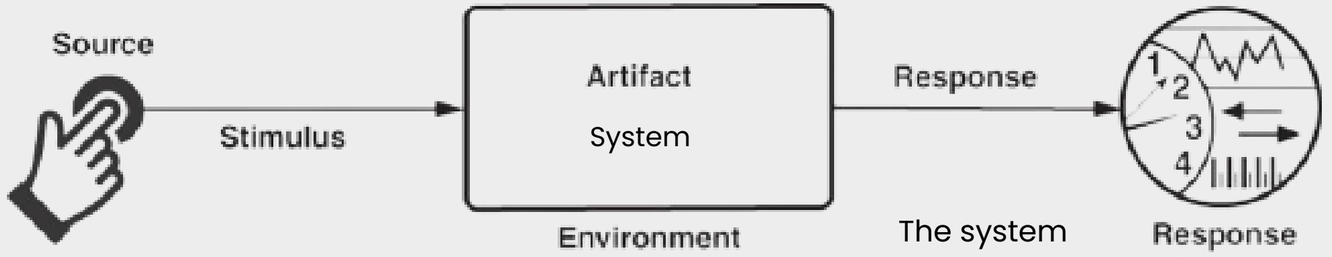

User
[Person]
User, Veterinarian, Shopkeeper
Admin
[Person]
Person who moderates the platform
Web Content Container
[Container: Node.js]
Contains the static content for the website
PetPeople Android App
[Container: Android]
Android mobile App
PetPeople Web App
[Container: Web]
Web App for Microsoft Edge, Firefox, Chrome, Brave, Opera, Safari, Tor and Samsung Internet
PetPeople iOS App
[Container: iOS]
iOS App
Web API
[Container: Express.js REST framework]
Backend Web API
Database
[Database: MongoDB]
App databse
PetPeople App
[Software System]
Payments System
[Software System]
External Payments API
Censor System
[Software System]
Censor System
App Notification System
[APNS, GCM]
Responsible for sending notifications in offline mode
SMS System
[Software System]
Responsible to send mobile messages to users, for 2FA purposes
Email System
[SMTP]
Responsible for sending emails to users (can be authentication or informational emails).
Analytics System
[Software System]
Analytics System
Uses the app
Uses the app
Uses the app
Moderates the app
Moderates the app
Moderates the app
Uses
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
MongoDB
[MongoDB]

Web API Components



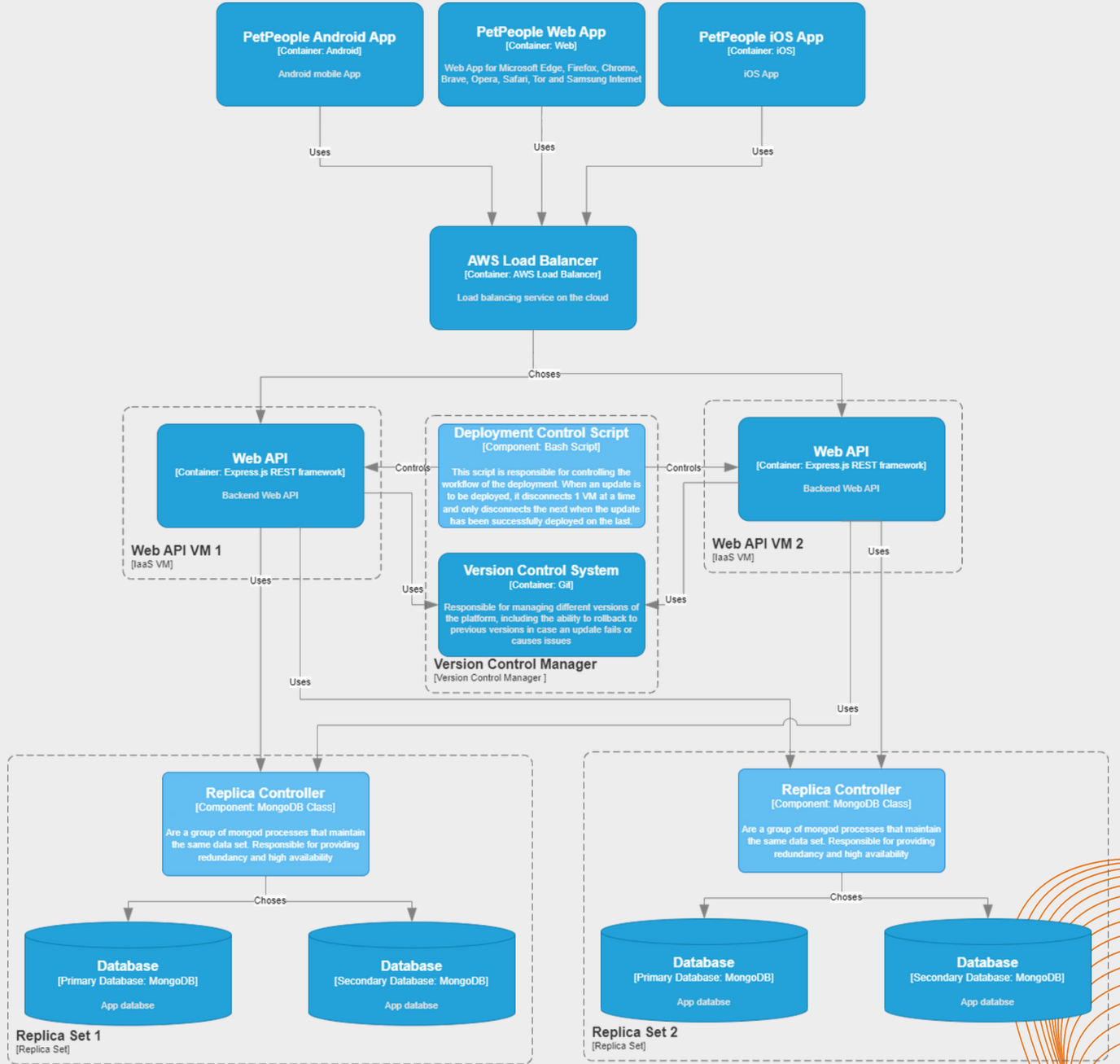
Scenario

- Up time (Availability)



The system's software	fails/crashes	Normal operation	The system responds by entering a previous safe state.	The system checks if the restart occured in less than 1 minutes after the initial crash.

Deployment


PetPeople Android App
[Container: Android]
Android mobile App
PetPeople Web App
[Container: Web]
Web App for Microsoft Edge, Firefox, Chrome, Brave, Opera, Safari, Tor and Samsung Internet
PetPeople iOS App
[Container: iOS]
iOS App
Uses
Uses
Uses
AWS Load Balancer
[Container: AWS Load Balancer]
Load balancing service on the cloud
Choses
Web API
[Container: Express.js REST framework]
Backend Web API
Controls
Deployment Control Script
[Component: Bash Script]
This script is responsible for controlling the workflow of the deployment. When an update is to be deployed, it disconnects 1 VM at a time and only disconnects the next when the update has been successfully deployed on the last.
Controls
Web API
[Container: Express.js REST framework]
Backend Web API
Web API VM 1
[IaaS VM]
Uses
Uses
Version Control System
[Container: Git]
Responsible for managing different versions of the platform, including the ability to rollback to previous versions in case an update fails or causes issues
Uses
Version Control Manager
[Version Control Manager]
Web API VM 2
[IaaS VM]
Uses
Uses
Uses
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 1
[Replica Set]
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 2
[Replica Set]

Architecture decisions

Load Balancer

- AWS Load Balancer
  - designed to handle large amounts of traffic;
  - health checks;
  - auto scaling features to help monitor and adjust to which machines to send traffic;
  - integrated in the AWS global network, which has a 99.99% to 99.999% availability per year.
- NGINX
  - designed to handle large amounts of traffic;
  - often used with other tools such as Docker and Kubernetes;
  - lightweight footprint and easily scale.

Both technologies impact on availability is similar, but due to the application being deployed on AWS, it makes a lot more sense to use a AWS load balancer, seeing that it would require some additional architecture components and configurations to use Nginx, which could delay the application's release date impacting BC2-Time to market, which imposes a deadline of 1 year.

Validation

The advertised uptime of the each of the services, according to documentation

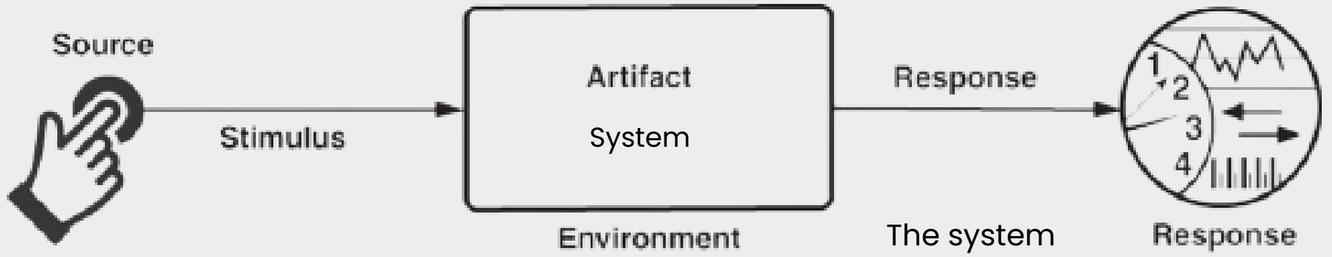
- Load Balancer - 99.99% to 99.999%
- 2 Virtual Machines - 99.99% to 99.999%
- Version Control System (Git repository) - 99.9%
- MongoDB Replicas - 99.995%

P((VM1 Down AND Vm2 Down) OR GitHub Down OR MongoDB Atlas Down OR Load Balancer)

	Worse expected downtime	**Best expected downtime**
Hardware	0.01%	0.001%
Software	0.1342985%	0.10500095%
Availability	99.8557%	99.89399%

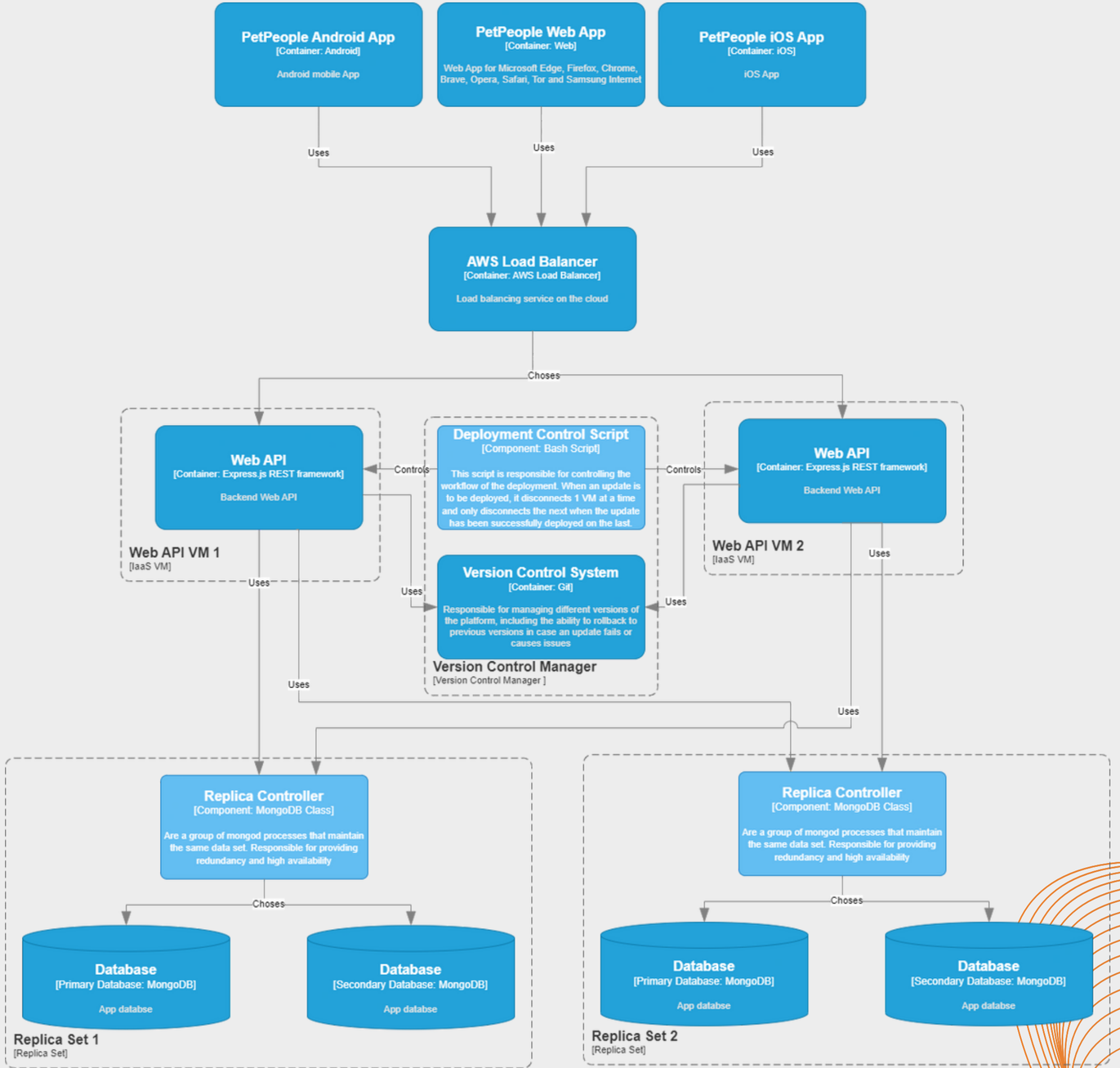
Scenario

- Number of concurrent requests (Performance)



The users	perform concurrent requests	Normal operation	The system responds by handling each request independently and assynchronously	Response Measure The system checks if the system can handle at least 800 requests per second

Deployment


PetPeople Android App
[Container: Android]
Android mobile App
PetPeople Web App
[Container: Web]
Web App for Microsoft Edge, Firefox, Chrome, Brave, Opera, Safari, Tor and Samsung Internet
PetPeople iOS App
[Container: iOS]
iOS App
Uses
Uses
Uses
AWS Load Balancer
[Container: AWS Load Balancer]
Load balancing service on the cloud
Choses
Web API
[Container: Express.js REST framework]
Backend Web API
Controls
Deployment Control Script
[Component: Bash Script]
This script is responsible for controlling the workflow of the deployment. When an update is to be deployed, it disconnects 1 VM at a time and only disconnects the next when the update has been successfully deployed on the last.
Controls
Web API
[Container: Express.js REST framework]
Backend Web API
Web API VM 1
[IaaS VM]
Uses
Uses
Version Control System
[Container: Git]
Responsible for managing different versions of the platform, including the ability to rollback to previous versions in case an update fails or causes issues
Uses
Version Control Manager
[Version Control Manager]
Web API VM 2
[IaaS VM]
Uses
Uses
Uses
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 1
[Replica Set]
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 2
[Replica Set]

Architecture decisions

- Framework
- Load balancer
- Hosting provider
  - Amazon Web Services
  - Google Cloud Platform
  - Microsoft Azure
  - DigitalOcean

Almost all hosting providers offer similar features that guarantee high performance, reliability, security, and scalability. Despite this fact, we believe that Amazon Web Services (AWS) is the most comprehensive hosting provider of them all and offer a wide variety of services in total. Furthermore, we have decided to select additional AWS components; thus our choice was to go with the AWS service.

Validation

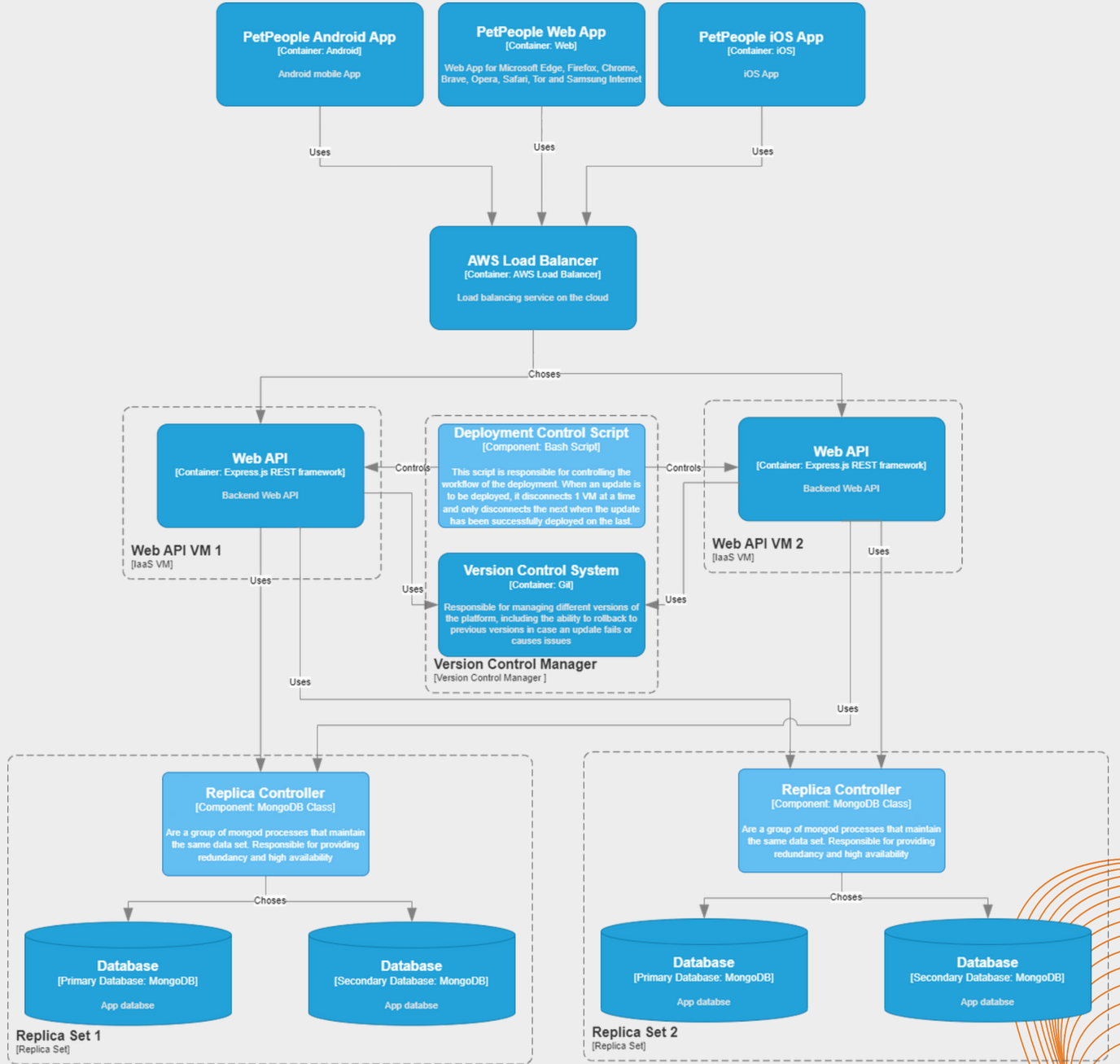
- Amazon EC2 server
  - 4 c6g.4xlarge instances - compute optimized instance that ensures enough power for CPU-intensive workloads
- Elastic Load Balancer - distributes incoming traffic evenly across multiple instances, allowing each instance to handle a portion of the requests and prevent any single instance from becoming overloaded by an abundance of requests
- Amazon Elastic Block Store (EBS)
  - IO2- tailored towards I/O-intensive workloads like databases while providing consistent low-latency performance

Scenario

- Sustain an annual content growth of at least 200% in content (Scalability)

The system	receives new content	Normal operation	The system responds by storing the content in the databases	the content's integrity is checked to see if it still exists and is correct, without loss in performance

Deployment


PetPeople Android App
[Container: Android]
Android mobile App
PetPeople Web App
[Container: Web]
Web App for Microsoft Edge, Firefox, Chrome, Brave, Opera, Safari, Tor and Samsung Internet
PetPeople iOS App
[Container: iOS]
iOS App
Uses
Uses
Uses
AWS Load Balancer
[Container: AWS Load Balancer]
Load balancing service on the cloud
Choses
Web API
[Container: Express.js REST framework]
Backend Web API
Controls
Deployment Control Script
[Component: Bash Script]
This script is responsible for controlling the workflow of the deployment. When an update is to be deployed, it disconnects 1 VM at a time and only disconnects the next when the update has been successfully deployed on the last.
Controls
Web API
[Container: Express.js REST framework]
Backend Web API
Web API VM 1
[IaaS VM]
Uses
Uses
Version Control System
[Container: Git]
Responsible for managing different versions of the platform, including the ability to rollback to previous versions in case an update fails or causes issues
Uses
Version Control Manager
[Version Control Manager]
Web API VM 2
[IaaS VM]
Uses
Uses
Uses
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 1
[Replica Set]
Replica Controller
[Component: MongoDB Class]
Are a group of mongod processes that maintain the same data set. Responsible for providing redundancy and high availability
Choses
Database
[Primary Database: MongoDB]
App databse
Database
[Secondary Database: MongoDB]
App databse
Replica Set 2
[Replica Set]

Architecture decisions

- Replica Controller

The replica controller coordinates the behaviour of each replica set and is responsible for electing the current primary database in case of failure of the previous one. It is also responsible for ensuring that the data is synchronized across all the replicas, guaranteeing that any read operation will have the same results, independent of which replica we are currently reading from.

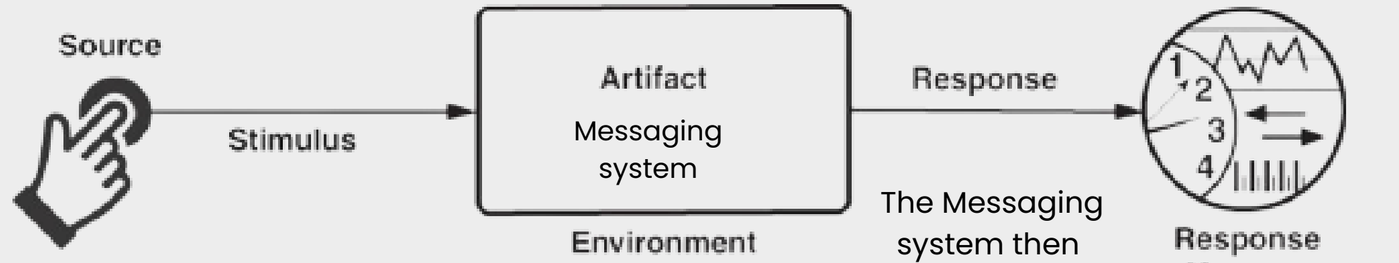
The fact that MongoDB Atlas allows for up to 50 replicas, in multiple replication zones, allows it to scale much more efficiently.

Validation

- The amount of data produced in the first year is estimated at 643 terabytes.
- With this in mind, the use of sharding is essential in order to horizontally scale, making it possible to handle large amounts of data.
- According to some documentation, each database can hold up to 64 terabytes of information. We shall use the Amazon Elastic Block Store (EBS) to store the databases, whose maximum storage suffices. The recommended maximum size for a shard is 50 terabytes.
- For our MongoDB cluster, we only need to create new shards when required, having virtually no limit. As such, scalability is assured in our architecture.

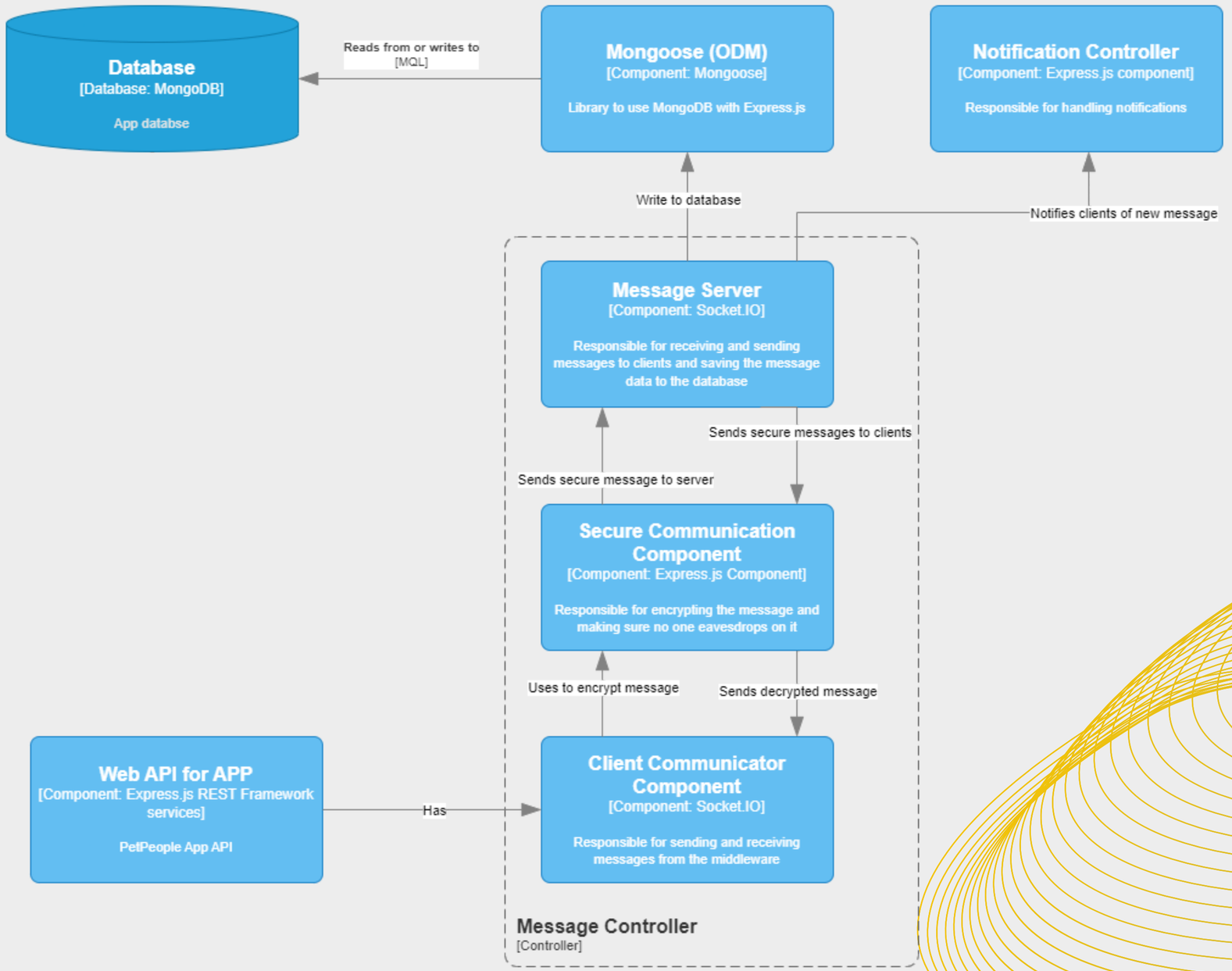
Scenario

- Secure messaging (Security)



The user	sends a private message to another	Normal operation	The Messaging system then responds by safely sending the message encrypted to the other user	it checks if no messages are ever intercepted by unwanted third parties

Message Controller



Architecture decisions

- End-to-end encryption
  - Is the standard for ensuring a secure and private communications.
  - The data is encrypted on the message sender's device and is only decrypted on the message receiver's device.
  - On top of using this method, the SCC will make use of SSL/TLS encryption protocol to safeguard all transmissions.
- Message-level encryption
  - each message is individually encrypted using a several encryption certificates, making sure that only the receiver can decrypt it using a private key
  - Seeing that it was made having sensitive transaction data in mind (Personal Identification Information, for example) and has a considerable overhead

Validation

	Security Checklist	Component
User Authentication	1. Verify that users are authenticated before accessing the messaging system	Message Controller
Encryption	4. Verify that messages are encrypted using a secure encryption algorithm	Secure Communication Component
	5. Verify that encrypted messages can only be decrypted by authorized users	Secure Communication Component
	6. Verify that encryption keys are securely stored	Secure Communication Component
Message Integrity	8. Verify that message checksums are used to ensure message integrity	Message Controller
	9. Verify that the messaging system can detect and report any tampering attempts	Secure Communication Component
Access Control	10. Verify that users can only access messages that they are authorized to see	Message Controller

Risks

Risk Description	Likelihood	Impact	Severity	Mitigation Plan	Owner
Application might not be able to scale to the amount of requests by users. The number of initial MongoDB shards and Amazon Elastic Block Store may not be suficient	1	5	5	Monitor resource utilization and if it forecasted that there will be a need for more resources, add more shards or increase the size of the Elastic Block	Database Admin
The censor system needs to be tested in the context of animal photos, seeing that Rekognition's documentation is not clear if it detects animal cruelty and abuse. This needs to be verified for a safe and secure platform utilization	3	5	15	Initially have many active admins that respond to reported content. An admin's decision should be informed to the Censor System, so that it can evolve. Once the efficiency of the algorithm has been calculated, the number of admins should be adjusted accordingly. If there are a lot of false positives, then we need to change the Censor System's dataset. If there are a lot of false negatives, then we need more data.	Tester
The architecture may lack streamlined deployment processes, leading to a complex deployment process	5	3	15	Configure CI/CD pipelines to automate the deployment testing process.	Developer
High mobile data usage. Due to the complexity behind the features, it is somewhat likely that the application has a high mobile data usage	3	3	9	Reduce the quality of images and videos in order to reduce the data needed to load them	Developer
There might be perfomance botlenecks in the fetching of data. There needs to be verification of the efficiency of the Inteligent Model and its performance in terms of speed.	5	1	5	Have a defined set of preloadable content (may be old posts that are still in cache) for each user that loads in case the Inteligent System is taking too long.	Developer

Assumptions

- We assume that 1000 ad impressions lead to 8$ of revenue.

Dependencies

- External Systems
  - Payment systems
  - Censor System
  - Google Analytics
  - Email system
  - SMS system
  - Notification system

Process

1. Prioritize requirements;
2. Chose framework, database and architecture type;
3. Follow functional requirements and make model C4;
4. Refine the architecture with quality attributes;
5. Chose the right test to validate each quality attribute;
6. Validate the architecture.

Team

Edgar Duarte

Sofia Neves

Tatiana Almeida

Thank you!

Pet People