**Solution Architecture Document**

**Stove and Oven Application**

Document Version V1.0

Submission Date: 2/18/20

Document Control Information

|  |  |  |  |
| --- | --- | --- | --- |
| Date | File Version | Change Description | Author |
| 2/18/20 | V1.0 | Initial Document Creation | Robert Chin |
| 4/28/20 | V1.1 | Updated physical view, Added Context View, and updated Architectural / Logical View | Robert Chin |

Reviewers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Change Req # | Review Notes | Review Date | Decision |
|  |  |  |  |  |

Table of Contents

[1. Overview 3](#_Toc38968200)

[1.1 Executive Overview 3](#_Toc38968201)

[2. Architecture Exceptions & Tech Debt 3](#_Toc38968202)

[2.1 Exceptions 3](#_Toc38968203)

[3. Applications and Technology 3](#_Toc38968204)

[3.1 Applications 3](#_Toc38968205)

[3.2 Technologies 3](#_Toc38968206)

[4. Requirements 4](#_Toc38968207)

[4.1 Reference Architecture Mapping 4](#_Toc38968208)

[4.2 System Quality Attributes / Non-Functional Requirements 4](#_Toc38968209)

[4.3 Monitoring Requirements 7](#_Toc38968210)

[4.4 Capacity Estimates and Summary 7](#_Toc38968211)

[4.5 Performance Requirements 9](#_Toc38968212)

[5. Solution Overview 9](#_Toc38968213)

[5.1 RAID Analysis 9](#_Toc38968214)

[5.2 Architecture Release Strategy 10](#_Toc38968215)

[5.3 Solution Architecture Views 10](#_Toc38968216)

[5.3.1 Context 10](#_Toc38968217)

[5.3.2 Architectural / Logical View 10](#_Toc38968218)

[5.3.3 Process View 11](#_Toc38968219)

[5.3.4 Development View 11](#_Toc38968220)

[5.3.5 Physical View 12](#_Toc38968221)

[5.3.6 Operational View 12](#_Toc38968222)

[5.3.7 Security View 13](#_Toc38968223)

[5.4 Security Architecture 14](#_Toc38968224)

[5.5 Data Architecture 15](#_Toc38968225)

[5.5.1 Data References 15](#_Toc38968226)

[5.6 Solution Integrations 15](#_Toc38968227)

[5.6.1 API Integration Catalogue 15](#_Toc38968228)

[5.6.2 Data Integrations 15](#_Toc38968229)

[6. Accessibility Assessment 15](#_Toc38968230)

[7. Glossary 15](#_Toc38968231)

[8. References 16](#_Toc38968232)

[9. Appendix 16](#_Toc38968233)

# Overview

## Executive Overview

Last semester we had developed a MVP for the Stove and Oven web application. It was a tool designed and developed to help users plan their weekly meal prep by providing some recipe examples. The tool also accounted for their refrigerator / ingredients they had, and accounted for key user attributes like favorites and meal preferences.

This semester we will look at additional features, performance scaling, productionizing, and security of the application.

# Architecture Exceptions & Tech Debt

## Exceptions

|  |  |  |
| --- | --- | --- |
| # | Description | Link |
|  |  |  |

There are no predefined target standard architectures / technologies requiring exceptions.

# Applications and Technology

## Applications

|  |  |
| --- | --- |
| Application Name | Architecture Requirements / Exceptions from Application Development Team |
| Stove and Oven |  |

## Technologies

|  |  |  |
| --- | --- | --- |
| Manufacturer | Product Name | Version |
| NodeJS |  |  |
| ExpressJS |  |  |
| ReactJS |  |  |
| Heroku |  |  |
| MongoDB |  |  |

# Requirements

## Reference Architecture Mapping

|  |
| --- |
| Alignment Comments |
|  |

## System Quality Attributes / Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Attribute | Description | Explain Approach and if there are any tradeoffs or tech debt to meet the requirement (if applicable) |
| Usage | Usability | Effectiveness for consumer | The team will do some usability testing |
|  | Localization | Experience tailored to region | The recipes are broad, accounting for certain dietary constraints/restrictions, but not regional.  It could deter certain types of users, but you don’t have to search by dietary constraints or restrictions |
|  | Accessibility | Support for disabled users | N/A |
|  | Personalization / Customizability | Audience specific experience | Users are able to account for their favorite recipes and profile pic, and meal plan / refrigerator ingredients |
|  | Reusability | Potential to save additional development effort | The Express API provides an avenue for if another kind of web application to query our lists of recipes publicly. |
|  | Configurability | Runtime behavior modification |  |
|  | Interoperability | Ease of information exchange | The exchange of information is done via HTTP/S APIs (for the React and Express) |
| Development | Manageability | Build process, source code and tooling alignment |  |
|  | Maintainability / Upgradeability / Flexibility / Agility | Ease of rapid change | Any code committed/pushed to the Github is compiled onto the Staging website  Additional work to create Production site (TBD) |
|  | Extensibility | Ease of adding functionality | The web application is open-sourced on Github. People can choose to add / modify code, as long as you’re granted the access. |
|  | Testability | Ensure functionality per specification |  |
| Operation | Performance | Responsiveness at least per SLA |  |
|  | Reliability / Availability / Resiliency | Usage window and mean time to failure (see metal rating) |  |
|  | Scalability | Increase throughput without reducing performance |  |
|  | Recoverability | Failure response complexity and SLA |  |
|  | Supportability | Ease of: deployment, operation, monitoring, debugging, or environment change (portability) |  |
| Security | Auditability | Track activity in all tiers | The tracking of users happens at the Heroku and the MongoDB  You could add user account management for the customer users, but additional tooling would need to be developed. |
|  | Attack surface area | All vectors that can be exploited | This is a web application.  All components are hosted on Internet, and communicate over Internet  HTTPS encryption between User/React, React/Express (TBD) or the mongo+srv between Express and DB  Proper whitelist controls should be implemented on Mongo, and or Express services. |
|  | Lease Privilege / Default Configs and Permissions | No user or process can do more than they should | RBAC on the Heroku for React and Express (Enterprise, if added), RBAC on the MongoDB  Users only can access their profile’s information. |
|  | Fail Securely | Protect from data leak or failure cascade |  |
| Tradeoffs |  | EX: Performance vs Maintainability, Interoperability vs Configurability |  |
| Other |  |  |  |

## Monitoring Requirements

|  |  |  |
| --- | --- | --- |
| Platform | Review Items | Response |
| On-Prem Application | Confirm that existing monitoring capabilities can handle all requirements? | N/A – Not On-Premises |
|  | Does this application involve any non-standard data sources? | N/A – Not On-Premises |
|  | List of Health Checks which need to be in place. E.g. Server startup, load balancer verification, message broker, web tier, batch trail, etc | N/A – Not On-Premises |
|  | Does the solution involve any batch functionality or peak times which would be recognized? | N/A – Not On-Premises |
| SaaS / Hosted | Does the solution come with native monitoring and performance management we can pull using open API? | Heroku and MongoDB have dashboards for monitoring and performance management.  Heroku Logplex  MongoDB project Events could be retrieved using API |
|  | Can the solution send events natively to our event management system? | Yes, this is possible. But we don’t have an event management system |
|  | Are internal users required to log on to vendor applications to check monitoring dashboards or correct issues? | Yes, unless integration with to be developed centralized event management / operational support systems |

## Capacity Estimates and Summary

|  |  |  |
| --- | --- | --- |
| Characteristics | Target value | Constraints / Notes |
| Total number of users | 10 | MongoDB user DB collection support 10 users  If Google Auth has a cap on how many users can authenticate through it |
| Number of concurrent sessions | 10 | Load Balanced ReactJS Heroku, Load Balanced ExpressJS Heroku |
| Number of total transactions per day | 100 | Added the inquiry and updates. |
| Number of inquiry transactions per day | 50 | Number of GET commands that happen on the Express to populate frontend React  This also probably isn’t an average |
| Number of update transactions per day | 50 | Number of PUT/DELETE commands that happen on the Express to populate Refrigerator, Favorites, Meal Plan  However, this probably isn’t an average (but rather once a week) |
| Batch processing windows | N/A | On a usual basis, our recipes are not batch updated from the Spoonacular API.  However, when doing so updating the Recipe DB should have a batch processing window. |
| % annual transaction increase | 1% |  |
| % annual data volume increase | 1% |  |
| Supported years of expansion | Only 6 months for until the end of this semester | Stevens Spring 2020 |

## Performance Requirements

These types of requirements should be discussed with the developers, I put placeholders.

|  |  |  |
| --- | --- | --- |
| Characteristics | Target Expectation | Constraints / Notes |
| Integration Response time for inquiry transactions | X sec | API interface System A to Sys B |
| Max utilization rate | X% | Headroom threshold |
| Response Time to Establish User Session | Less than X minutes |  |
| Response Time for Inquiry Transactions | X secs | From cold start |
| Response Time for Update Transactions | X secs | Per leg of transaction |
| Timeout if no response from host | X minutes |  |
| Log out if idle | X minutes |  |
| Max utilization Rate | X% |  |
| Response Time to get a report for viewing | X secs |  |
| Max Utilization Rate | X% |  |

# Solution Overview

## RAID Analysis

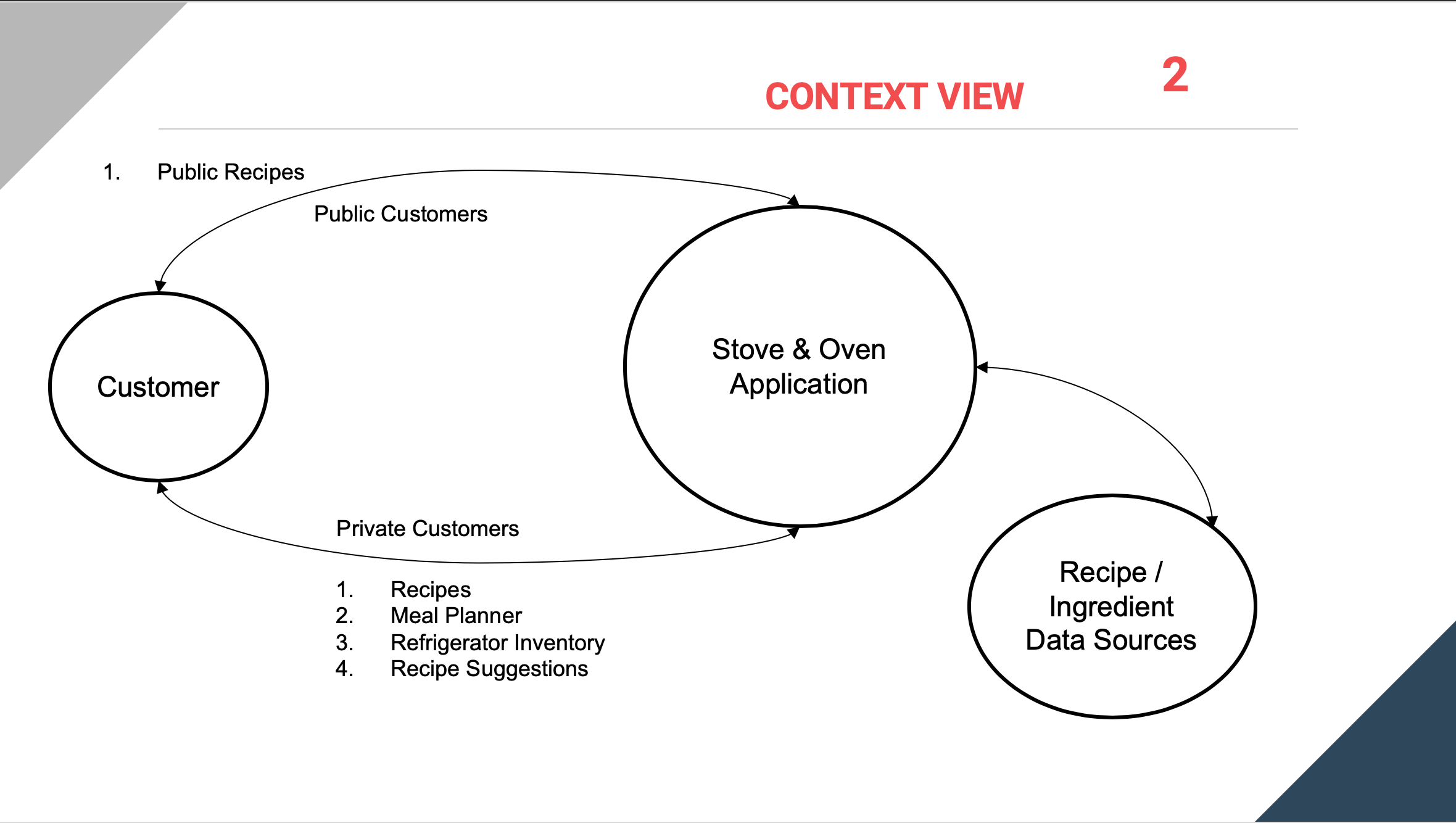
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Description | Type (Risk, Assumption, Issue, Dependency) | Criticality | Next Actions / Decisions / Status |
| 1 | Refactoring of ExpressJS code to support multiple databases | Dependency | High |  |
| 2 | Creation of new MongoDB Cloud Atlas accounts / projects | Dependency | High |  |
| 3 | Investigation of cost/benefits of Heroku Enterprise to support LB | Risk | High |  |
| 4 |  |  |  |  |

## Architecture Release Strategy

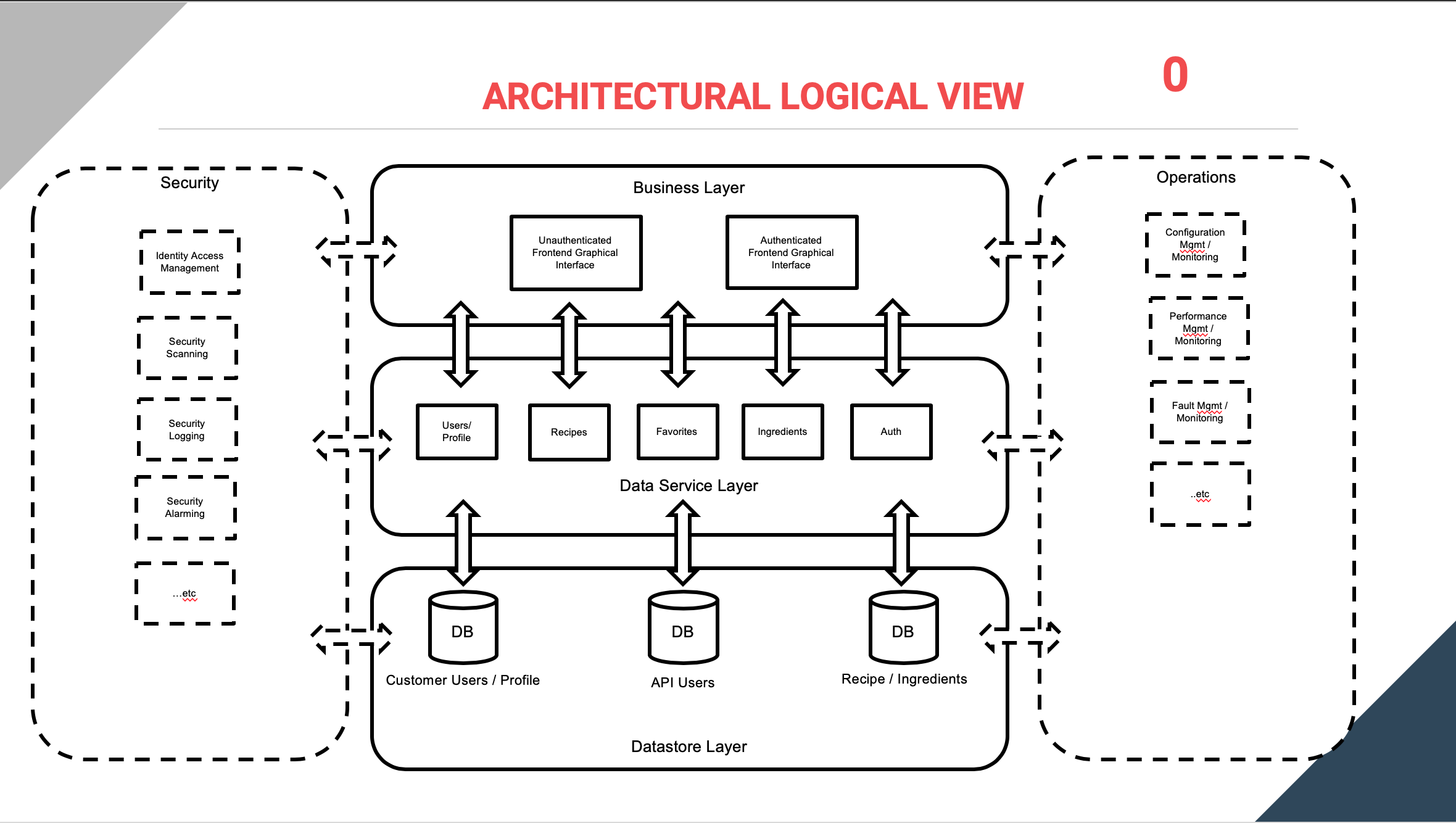
|  |
| --- |
| Release Strategy |
|  |

## Solution Architecture Views

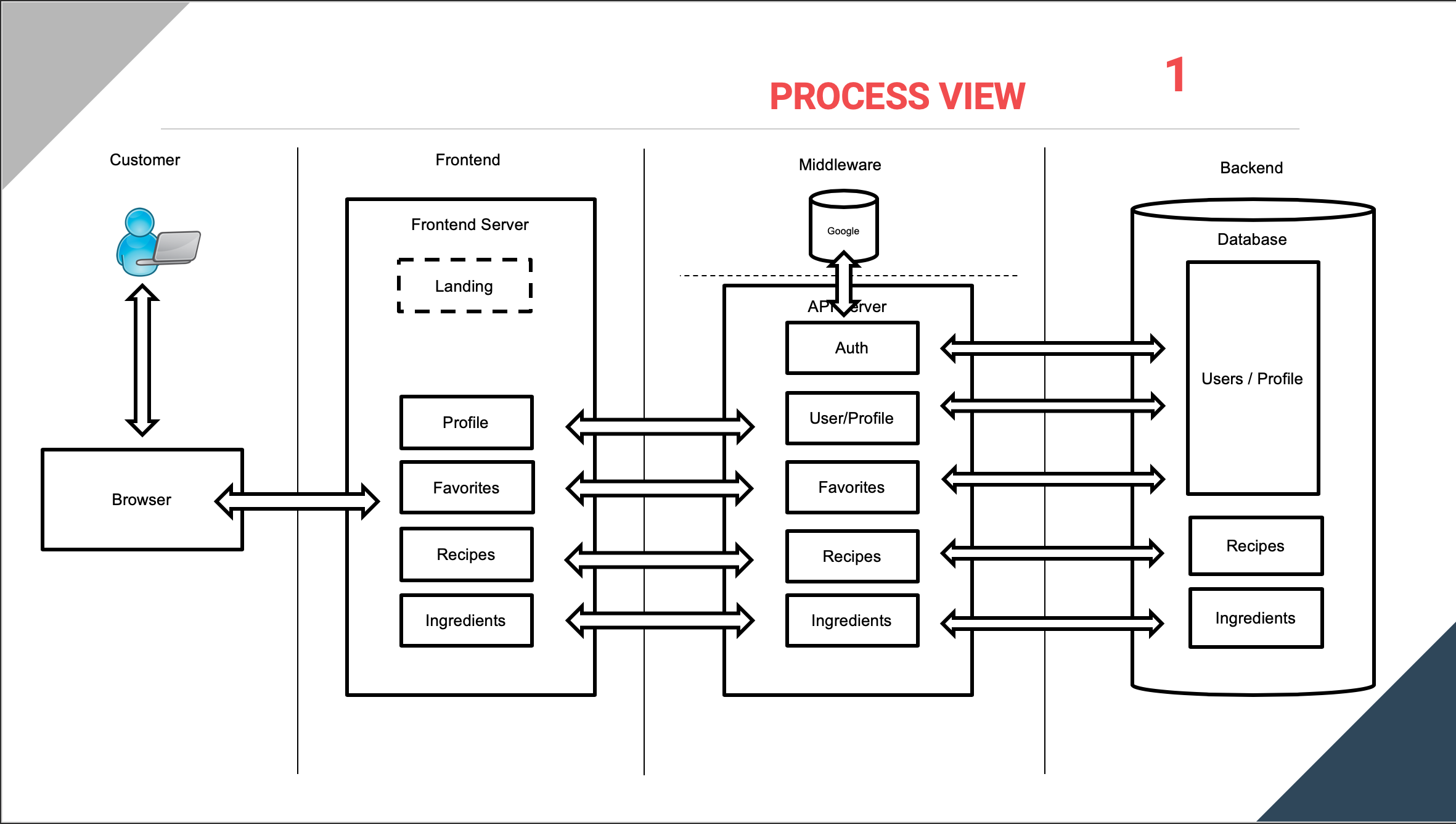
### Context



### Architectural / Logical View



### Process View

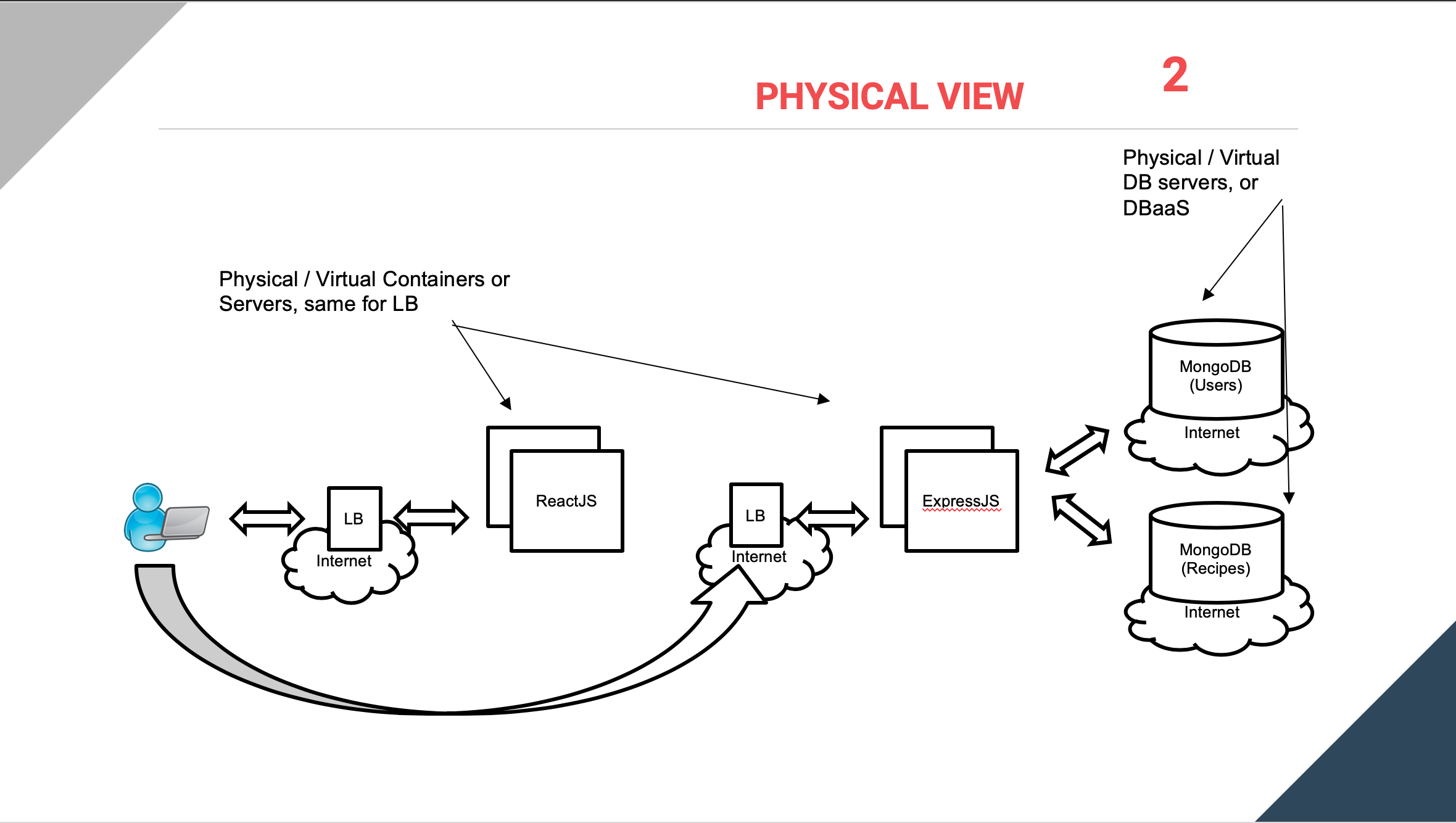


### Development View

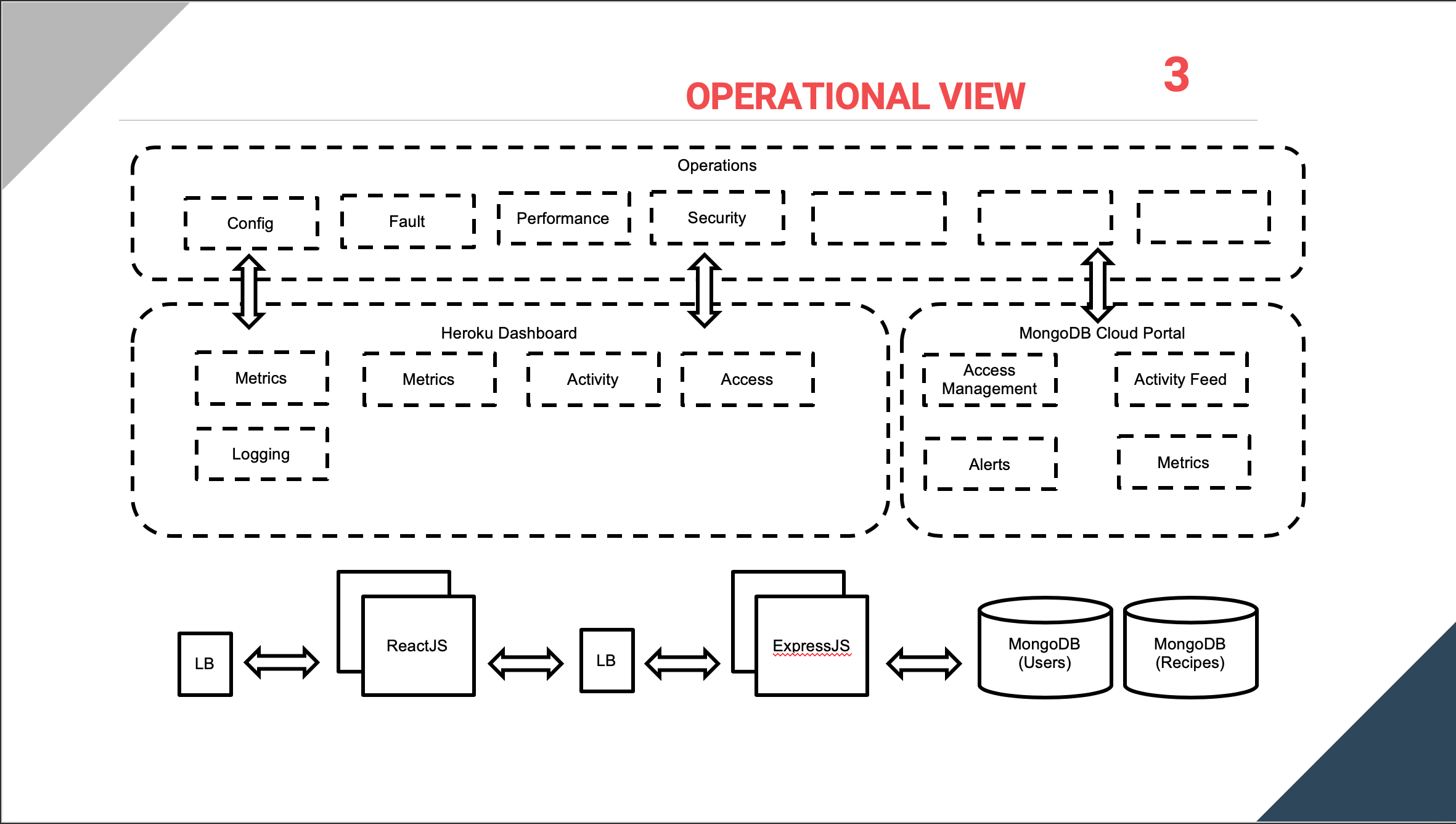
The existing application/system consists of:

1. ReactJS 16.7.0
2. ExpressJS 4.17.1
3. NodeJS v10.x
4. MongoDB Cloud Atlas / MongoDB 4.2

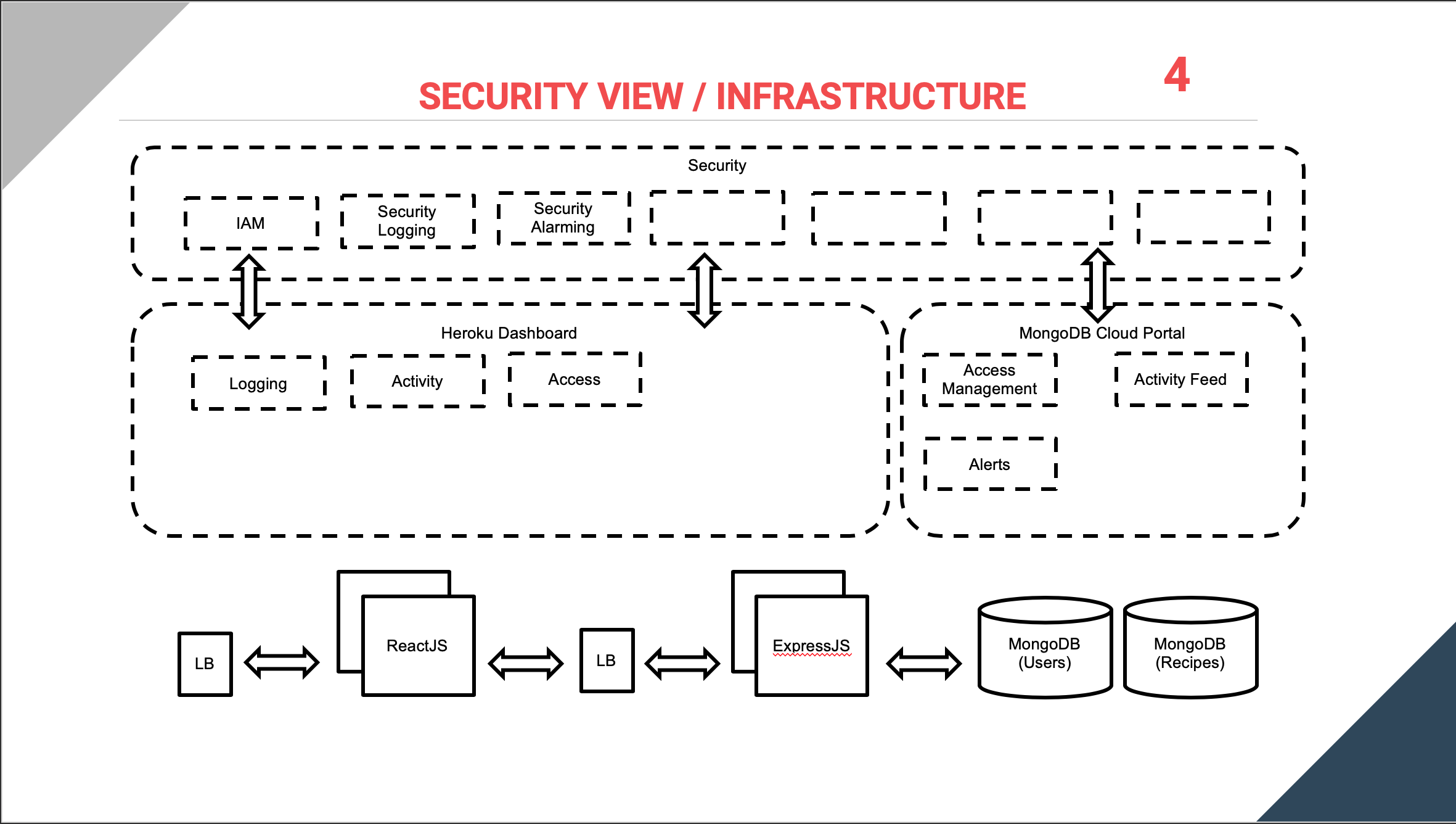
### Physical View



### Operational View



### Security View



## Security Architecture

|  |  |
| --- | --- |
| Question | Response |
| How will SSO be implemented? | The customer authentication is done via a Google Auth.  There are plans for local customer authentication.  No, SSO will not be implemented. |
| Describe how the solution is leveraging API security standards? | OWASP API standards should be considered when developing  <https://owasp.org/www-project-api-security/> |
| What secure channels are being used for file transfer (ex. sftp)? | There’s no file transfer happening, but HTTPS is should be used |
| If data at rest will be stored somewhere other than an internal managed store (ex. Vendor hosted, or cloud hosted), please list and describe the encryption methods used (ex. AES-256) and provide other security protections | Currently, data is stored not internally managed but on MongoDB Cloud Atlas.  Cost/Benefit analysis to enable encryption of data at rest should be investigated |
| Describe the encryption method for data in transit (ex. TLS 1.2) | Data to the user is HTTPS / TLS 1.2  Data between Express and DB (TLS 1.2) |
| If the solution has External User facing sites, is Multi-Factor Authentication (MFA) enabled? | No. |

|  |  |  |  |
| --- | --- | --- | --- |
| User Type | User Role | Authorization | Certification and Recertification |
|  |  |  |  |

Heroku Access (Enterprise version)

1. View
2. Deploy
3. Operate
4. Manage

MongoDB Cloud Atlas Project Access:

1. Project Owner / Cluster Mgr
2. Data Access (R)
3. Data Access (RW)
4. Project (R)

## Data Architecture

### Data References

The application’s data consists of:

1. User Profile
   1. Credentials
   2. Favorites
   3. Ingredients in their Refrigerator
2. Recipe / Ingredient Data

The User Profile’s Credentials should consider encryption methods in database

The rest of the data where feasible could also consider encryption at rest, but not necessary – it’s public data. You could consider more secure handling of a User’s private information of ingredients in their Fridge or their favorites

## Solution Integrations

### API Integration Catalogue

N/A

### Data Integrations

N/A

# Accessibility Assessment

|  |  |
| --- | --- |
| Question | Response |
| Does your application / project / solution involve an external facing web application | Yes |
| Does your (application / project / mobile app) comply with the respective country’s Web Accessibility Laws and Policies? |  |

# Glossary

|  |  |
| --- | --- |
| Terms | Definitions |
|  |  |

# References

|  |  |
| --- | --- |
| Item | Link / Reference Source |
|  |  |

# Appendix