Architecture / Technical Design Template Document

Document Overview

This document contains the technical architecture and data model that will be developed during the *[XXXX]* development project. The architecture here will be described at a sufficient level to help structure development. The architecture will be described with the use of four major views into the structure of the application.

- A **logical view** of the system – will provide the reader with an overview of the major components of the system.
- A **physical view** of the system – a description will be given about how the application will be structured and what technologies will be developed in and configured for use in the system. This view will also discuss the use of an infrastructure in the application deployment.
- A **development view** of the application – the reader is provided with some summary information about how code will be developed and the general development workflow.
- A **process view** of the overall system – in this view of the application the document will cover the various runtime aspects of the application including how the application may be scaled. Additionally it will describe at a high level the Agile design and development processes being used to define and implement the application.

How to read this document

[This section is optional.
You'll want to give the reader a description of how to read your document, I like the approach of giving each audience who is expected to use this document to be aware of what to read and what to skip to provide them a map of the application and how this documents it.]

Summary of Functional Scope

[This should be a 2-4 paragraph description of the project and high level KPIs and product goals of the changes and if applicable how will the end users interact with the system that the rest of this document is defining the architectural approach of implementing. NOTE: you know you've given a good summary when the technical documentation below fully supports the summary given here - if what you've defined below doesn't support the functionality listed here, you've either not described the functional scope well enough or you've given too much technical detail below.]

Reference Documents

To aid the review of this document please review the following documentation to understand the functional scope and vision of the site. These documents, including this document, are living documents that are subject to change during the life of the project.

[List out all of the product scope documents relevant to understanding this architecture - please be exhaustive and don't assume the reader knows which document or which version is relevant, simply list it out]

Non-functional Requirements

[*In this section please document all of the systems non-functional requirements such as:*

- *If a web application please include*
  - *Technical SEO requirements and considerations*
  - *ADA requirements*
  - *Browser support*
  - *Site performance metrics of interest and targets*
  - *Any interesting notes about the user interactions which are driving technical decisions*
  - *any relevant security requirements or uptime requirements*
  - *how we are handling user experience in system failure states*
  - *site performance characteristics*
- *If a service or process please include*
  - *references to the api interfaces, if there's a sketch*
  - *domain models being met by the interfaces*
  - *service performance expectations*
  - *security requirements for service*
- *For all systems also please include information about the following areas*
  - *type of data being managed or handled in the system*
  - *if any part of the data is considered PII, if you are unsure please work with department head to identify*
  - *references domain models being met by the interfaces*
  - *references to the data sets being interacted with or modified as a result of this architecture*
- *And any other system need or requirement that is driving decisions in the technical direction*]

Additional notes about the architecture and major decisions

This document is meant to be a guide to understanding the various parts of the system. Because the site requirements are evolving and there are ongoing technical discussions, this document will

be updated to reflect the outcomes of conversations, decisions, and learnings. As such, this document should be considered a living document and maintained throughout the life of the project this was created for. **It is the responsibility of the technical lead on the project work to maintain it.**

[The above note is important, please update this section with links to relevant meeting notes (please no slack links to private channels, etc) that capture the context of the decisions made and the updates to the document. e.g., "Added a section to define testing approach because it's a change from our current practice of using bullwinkle"]

Logical View

As described above, this view of the system will provide an overview of the major functional parts of the overall application.

[Generally this should be a boxes and arrows diagram - feel free to freehand it if you don't have a preferred diagramming tool. The key here is to layout all of the moving parts and how they relate to each other. Diagram styles to consider:

- *Component Diagram*
- *Overviews or details of relevant Domain Models*
- *Activity Diagrams*

As the goal here is to give an overview of the technical design, you don't need to give a lot of detail. It's also recommended to not only include diagrams but to also include a written explanation of the diagram so that the document can stand on its own.]

High-level Enterprise View

[This section is optional

A bird's eye view of a change to help the reader (and future readers) of the scope of change and how it relates to the rest of the overall enterprise architecture.

For example, if this document is meant to define a new section of the website, for example a user management section of the site, which also requires the inclusion of a new subsystem or system, or integrating with an existing one, and having that service support a new use case. It's good to include how the new scope "fits" in with the rest of the system.]

Domain Model View

[This section is optional, if documentation of the domain, business or data modeling lives elsewhere, simply include a link to it here. It is the expectation that the linked document is current and accurate - if it is not, update those documents as it may be easier to keep those up-to-date]

Here's a good article for how to iteratively design a domain model: http://agiledata.org/essays/agileDataModeling.html

Currently there is no standard so one's best effort is fine, there is one example one can use as a model: Employer Ecosystem "Project Cauldron" Company and Profile Domain Model]

Reused Components or Systems

[The purpose of is to simply make a list of the different tools you are using, and most importantly modifying to implement the functional scope of this architecture. Between this section and the brand new components below, every element in the high-level diagram should be discussed between these two sections with some level of description and detail about why it exists and the value that each component provides.

A bulleted list with a description of what you are changing. e.g.

if it's technical

- *a software agent - to be able to collect additional performance metrics automatically the team will be updating the client to collect on the page load event the user's browser who has loaded a page*

or if a large application is being modified - the following for a segment of the application

- *admin pages - updating one of the forms to support a the configuration of a client's configuration*

It's recommended to scale the amount of description you provide to the amount of effort / scale / lines of code that will be updated per the component. For example if a segment of code only requires a few lines of code updated, list out the change, similarly if a component is getting an overhaul be descriptive of the overhaul with some detail to help the reader get a sense of what and why the changes are.]

Brand New Components or Systems

[Similar to the reused components do the same for the brand new components. For these, be more descriptive of the scope of the component being built. Provide any nuanced business and nonfunctional requirements especially if they are not self evident but provide motivation for the choices made]

Physical View

This section of the architecture provides a map of how the application or technical scope will behave when deployed in staging and production. What you will see below are the deployable components and how different repositories and systems interact with each other to implement the functional and nonfunctional scope described in the Summary of Functional Scope above.

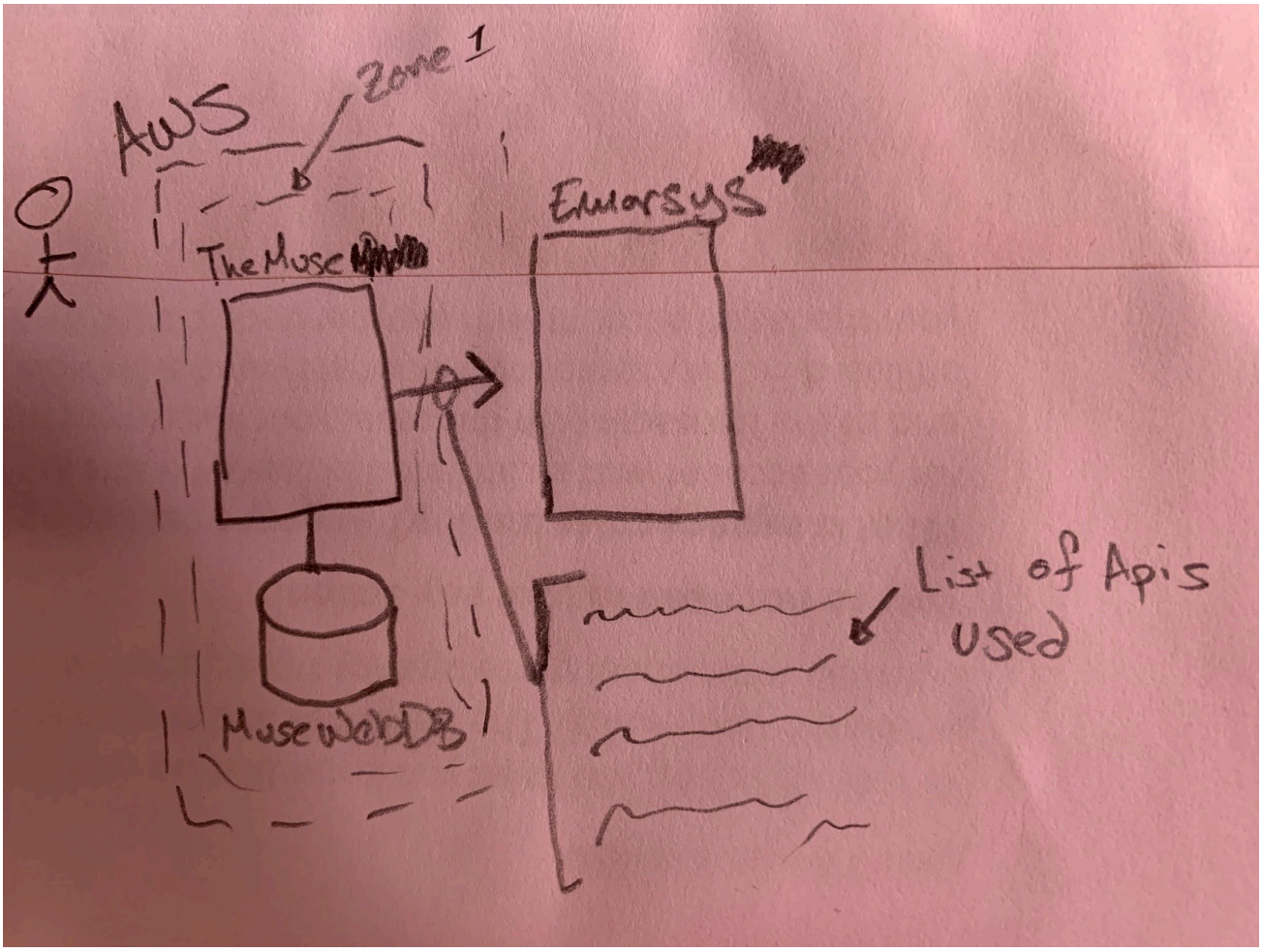
System Architecture

Below is a visualization of the major technologies used in the application and a general outline of how they may integrate with each other.

[For the most part this section of the document should have the goal of describing the various parts of the system as it functions in production, with references of key interaction or activity diagrams.

An example for our Email Form capture:

Below is a rough sketch of the deployment approach for the email capture form.



The various deployment artifacts are:

- *XXX repo will be deployed as normal. The following AWS environment variables were added to make sure we are able to update the api endpoint and to inject our private keys into the deployment*

    - *EMARSYS_USER - our user id*
    - *EMARSYS_ACCOUNTID - our account id available via the emarsys management console*
    - *EMARSYS_API_URL - the url of emarsys's api*
    - *EMARSYS_CERTIFICATE - an ssh key credential for our account*
- *Within AWS we will need to open the firewall to allow connections to the emarsys*
- *...*

This example is a little contrived though the quality of the diagram is not important as long as we've captured the overall design in both visual and textual manager to help avoid ambiguity
]

API design

[If your system is either creating new APIs or modifying existing ones list out the APIs and the changes you are making to the APIs. You should be detailed about the changes being contemplated as apart of this design, what those changes are, why you need to make them, and how those changes support the described functionality

NOTE: this is one of those sections that if there's a need to change the plan it is a requirement that those changes need to be informed with consultation of our platform team, and the head of engineering is notified of a change in plan, in addition to this document being updated to reflect the current state of thinking - This document should be a living document.]

Database Model Updates

[If the changes you are working on require data model updates, those model updates should be described below. Big things to include are DB creation, tables being added or removed. No need to include specific table fields unless they articulate or show data available to support a featurer or function described in the scope above

Separately you should also describe the process by which you will deploy these changes to production, and if there are any data migration scripts needing to be run, a loose understanding of the testing approach / plan prior to going to production.]

Staging and Production

[Capture any meaningful differences between production and staging here. And what approach is being taken for testing and validation that the changes are being verified prior to production. This is a great section to discuss performance testing as a meaningful distinction between staging and production is usage volume and data volume]

Cost Estimation

[what are the costs associated with the work outlined in this proposal. Anything that is going to add more than 120 dollars in a year in spend should be reviewed prior to moving forwards with the head of the department.

Things to consider when estimating :costs

- *Storage costs - S3, DB sizing, etc*
- *Networking costs, if we need to go to the public cloud or make anything available in the public cloud, how "big" would it be? Note this is hard to do if the physical infrastructure hasn't been thoroughly vetted*
- *CPU costs - what type of infrastructure will be needed to support these changes? is it incremental? or will it change the trajectory of how much we spend. To deploy the scope will new containers be required, how many, etc.*
- *Logging and Operational data costs - are we planning to log a lot of data for monitoring or performance monitoring?*
- *New software and services. Are we purchasing new software or adding costs (e.g., new user seats) as the result of these changes?*
- *Anything that would end up being additional costs for the company should be listed*

Ideally these costs are broken out into the cost source, and mapped out monthly for the next 12 months.

Here's an example spreadsheet to clone and fill out:

This is important because we should also be calculating costs before we deploy if they are appreciable (and if not you should be to show how you affirmed that they are not if asked).]

Development View

[The development of the system can be broken down into a number of parts:

- *local code development*
- *source code management*
- *continuous integration*
- *unit and functional testing*
- *etc.*

The subsections below will expand on the activities that a team will need to build for when implementing a new system or rebuilding a subsystem, and shall describe the tools used to facilitate them.]

Local Development Environment

[*In this section please answer the following questions:*

- *What is the local development environment like?*
- *What repositories will be required to build the system? Are any new repositories?*
- *Are there any dependencies beyond items like docker and github?*
- *How does one build the system?*
- *Will the developer need to "prep" their system to be able to install and run the application locally with environment variables or additional configuration like updating the hosts file?*
- *Does there need to be access to services or api endpoints for the locally running instance to work correctly? If so, which ones?*
- *Does the developer need to be on the VPN? Do they need to have permissions to external systems?*]

Continuous Integration

[How we are utilizing CCI?
This really should include anything that doesn't already conform to the standards laid down here: Docker and Scripting Standards Link XXX]

Unit and Functional Testing

[*What unittesting and functional testing is being planned for?*

- *What isn't or being deemed unnecessary?*
- *Are there any particular challenges that will need to be overcome during the build, such as building testing fixtures and data sets so that testing can be executed both locally and in circle ci?*]

Process View

In this view of the architecture we will look at the process aspects of the application: including any content migration processes, development process, and the operational processes of the application.

Operational Requirements

We recommend for purposes of planning:

1 A **recovery point objective** (RPO), or the point in time prior to an application outage that data is captured, should be about XXX hours' worth of data loss. This

site will be a free, publicly facing site and if the site goes down, there is a reasonable expectation that data loss should be contained to XXX.

2 The **recovery time objective** (RTO), or the maximum amount of time the site should be down unexpectedly, should be about XXX in the case of something catastrophic (such as an earthquake near the data center). If the failure is not as the result of a catastrophe, the RTO should be closer to XXX hrs.

3 The **recovery capacity objective** (RCO), or the amount of capacity the site should have in a failure mode, for the initial release is 50%; if there is a catastrophic failure at the data center.

[*For the items above please get into details below here, please include:*

- *how you came to the conclusion of the need?*
- *what aspects of the system will support achieving these goals?*
- *how does the system meet these goals in a failure mode? what failure modes are you planning for, are there any you are aware of that you are not?*]

Security & Privacy

[*Please work with the platform team in filling this out, and if you are planning not to please review with department head why these concerns do not apply*]

Operation Processes

[*Once live, what does logging and monitoring looking like?*
Things to include in this section:

- *What are the major moving pieces of the application and how are they composed to execute their major tasks.*
    - *In the case of a web request it could be an activity diagram to show how the different architectural components interact to return to a user the user's response.*
    - *In the case of an ETL process this would be the ETL flow (if not defined in a section above)*
- *How will the system support logging and debugging when in production?*
- *How is the system being monitored in production?*
- *Is caching being used? If so why (or why not) and,*
    - *what is being cached,*
    - *what is the caching duration,*
    - *what is the caching volume,*
    - *what is the target cache hit rate?*
- *Are there any particular system needs for the application in regards to system sizing? Do you need a particular number of processors or memory size? Why?*
- *etc.*]

System Sizing

[*To help with understanding the services and scale of the application it is also important to capture those aspects of the system that give a sense of scale, or change in scale. As such in this section please create subsections to share what the current understanding of system scale this architecture is meant to support. This section and Cost Estimation are usually heavily related. Areas of interest include:*

- *Data Volume: e.g., number of records for key tables, number and count of large objects being stored in S3*
- *API request volume: if we are interacting with internal or external apis - what is the request volume per hour and per day,*
- *site request volume: if the design is meant to be user facing, what is the planned for number of unique visitors, individual requests*
- *payload size: what is the target size of a request payload, for a service this may be measured in K, in the case of web requests this may be on the scale of MBs*
- *logging volume: how much logging will this application create?*
- *monitoring event volume: how many monitoring events are expected to be created?*
- *if this is a data transform process, understanding what the expected time duration of the process is informative.*

For all calculations please include average and peak usage calculations including links to a spreadsheet with the math so we can understand how the final numbers were calculated.]

Data/Content Migrations

[This section to list out and detail the strategy and approach being taken for all data migrations needing to be done to execute and complete the architecture being defined here.

[*For each migration, please include:*

- *A brief description of the migration - in layman's english. Eg, "Migrating the email subscriptions out of the user tables and into a separate table structure as defined in section XXX. This will be accomplished by using XXX"*
- *How will this be done?*
- *Testing approach for the migration both to QA the migration process as well as to QA the final run in a production mode? A reference to a migration specification or documentation on how "correctness" of the migration will be tested.*
- *How will the migration be reverted or backed out if it fails? If this requires a back before the migration occurs, what things will need to be backed up*
- *Does any client or end user facing portion of the site need downtime for the migration?*]

Logging and Monitoring

Logging

[*This section is not meant to define what we log in any given error or warning. This section should outline the architecture logging strategy. It should ideally describe and be able to answer the following questions:*

- *What are the logging goals of this application?*
- *Are there specific actions or events that will be logged to support system debugging and monitoring?*
- *Are there any parts of the system that will require logging to help with system performance analysis?*
- *Are there any production system support needs that will need to be logged to provide the account management team or another team visibility into the system health?*]

Monitoring

[*Similar to Logging, this section should outline a strategy that will be used for monitoring the site health and alarming. This section also in part should be aligned with or driven by the sections: Non-functional Requirements, Cost Estimation, and Operational Requirements.*

Things to consider:

- *How are we measuring and monitoring system uptime?*
- *How can we be using monitoring to ensure the nonfunctional requirements are being met, and when they are not?*
- *How do we measure/monitor if we are meeting our system sizing needs?*
- *How do we measure / monitor our production process characteristics? E.g., cache hit rate, or process duration?*
- *How are errors and error types monitored? When should they alert those on product support?*
- *Are there changes we need to make/introduce to our operational monitoring tools to support the monitoring goals? If so, what are they?*
- *What systems will include health checks, what will those health checks monitor? Are there parts of the system that will not be monitored?*
- *… feel free to include other aspects not otherwise mentioned*

For the above responses what are acceptable thresholds? Do those thresholds need to be monitored or alerted if surpassed?]