# Design Document Template

Copy everything below this line to a new document, then fill in the sections. Instructions are in block-quotes. Remove them once each section is either complete or has been removed. Not all sections are always needed, use some judgement.

Changes to your service’s architecture or any key functionality must be reflected in this document, and reviewed if appropriate.

# 1 Design Inputs

The subsections in this section are considered **inputs** to the design process, and should not change based on the design selected.

Delete this block once complete.

## 1.1 Problem Statement

Describe the problem this design is trying to solve. Be brief, but complete enough to describe the challenges involved.

Delete this block once complete.

## 1.2 Definitions

Define the taxonomy e.g.  
Purchasing Service - The front-end software that an End User is directly using.  
Purchase Order - a data object representing a purchase of an asset by a customer.

Delete this block once complete.

## 1.3 User Roles

Defines the customers for the feature being added or modified.

e.g.

End User

Community manager

Service Developer

Delete the block-quoted text once complete.

## 1.5 Description

Provide a high level description of service/feature/system

Note that this is not a description of the implementation, and not just a restatement of the problem statement.

It’s ok for this to be very short, if everything that needed to be said was already said in the problem statement.

Delete this block once complete.

## 1.6 Scope and Limitations

Briefly describe what this feature or change does not encompass. Describe the boundaries of this functionality. The intent is to describe the input that drives the choice of your design, not the limits of the design. Answer the question “What part of the problem are we not attempting to solve?”

Quite often the assumptions we are making as part of our design process should be represented in this section, or in the [1.7 Constraints](https://quip-amazon.com/w8xXABrjDAOR#temp:C:JGaa3e599fb4b1f4309b15c85336) section. The difference between the two is that this section describes self-imposed choices, while Constraints describes choices imposed by others.

Delete this block once complete.

## 1.7 Constraints

When building systems, often we need to interact with other services, and those services will impose constraints on our design. Those constraints are another input into the design process.

See the [1.6 Scope and Limitations](https://quip-amazon.com/w8xXABrjDAOR#temp:C:JGa65dd4ff5f7144f8eac4f49cc0) section description for a description of how assumptions map into these two sections.

Delete this block once complete.

# 2 Design Details

The subsections here are the “**output**” of the design process. The subsections under this heading can be modified, adding or removing as necessary. This allows the template to be adapted for use in both high-level designs as well as low-level.

This section should have exactly the set of things that are needed to implement the design.

**If there is extra information that provides background, put it in an appendix section.**

**If there is extra information that provides justification of design choices, put it in an appendix section.**

Delete this block once complete.

## 2.1 Design Summary

Write a paragraph or two that gives a brief overview of what your design changes. After reading this section a reviewer should be able to have a high-level understanding of what changes are to be made and why. This section should be brief.  
  
Provide a list of services/components proposed to be built, what technology will be used.   
E.g  
**AutomationService** - REST based service written in Java  
**TranscriptsAccessor** - Library to used to retrieve call/chat trasncripts securely from S3, written in Java.

Delete this block once complete.

## 2.2 User Experience

If the system being designed has any user-facing components, other than an API, they should be described in sufficient detail that they can be implemented. A link to a UX specification is sufficient. If there are no user-facing components other than an API, write ‘N/A’.

Delete this block once complete.

## 2.3 Data Model

What is the data model you are proposing with this design?

Delete this block once complete.

## 2.4 API

What is the programmatic external-facing interface for this design? Is this external-facing interface public? The API should be described in sufficient detail to be implemented, including information about data structures and error handling, as well as the expectations that a consumer should have when using the API. External-facing public APIs should follow the AWS API standards: [https://w.amazon.com/bin/view/AWS/API\_Standards/](https://w.amazon.com/bin/view/AWS/API_Standards/CommonTerms/#Deprecated_verbs)

When this service or change is implemented link to your documented smithy model or other API definition here. There is no need to update the individual request/response details from your initial review.   
The list of APIs must be kept up to date. Deprecated APIs must also be kept until there are no longer any consumers.

Delete this block once complete.

## 2.5 Architecture

Provide appropriate design details. These may include a representation of the static structure (system diagram, etc), a representation of the behavior (sequence diagram, etc), Diagrams should still have text descriptions that accompany them - a diagram by itself is not complete.  This section should answer the question: How does this design work? It should have sufficient detail that a developer other than the author can implement the design with a reasonable chance of success.

Delete this block once complete.

## 2.6 Telemetry

### 2.6.1 Health and Performance

What metrics will we use to measure health and performance? Are they default metrics provided by AWS services in CloudWatch? Are there specific data points that you want to capture in code (if so, how will you capture those?)  
How do you want to instrument your code?

Delete this block once complete.

### 2.6.2 Business Metrics

List of business metrics and their definitions. Where will they be instrumented?

Delete this block once complete.

## 2.7 Key Decisions

What are the major decisions that you made in this design process? For each decision, describe the alternative solutions, and the reason for going in the direction you chose.

Delete this block once complete.

# 3 Software quality attributes

Design flaws are often only visible when viewed through an appropriate lens. This section provides a set of lenses which you will use to view different aspects of your design. Working through these sections will force you to think about the design from each perspective, helping to expose flaws that might exist.

How does this design address each of the following software quality attributes? Where are trade-offs or compromises being made? Note that there are many more quality attributes of interest but these are particularly relevant to our system and development. The sections under this heading are intended to force you to think about each of the different attributes described, so that you can be sure that they were addressed in the design. Most of the text in these sections should refer to details in the “Design Details” sections above, rather than being the original source of the information presented.

The subsections here are the best place to capture the “why” for some of your design aspects, especially those that cross-cut multiple parts of the design. It’s easy to see the “what” part of the design, that’s what the [2 Design Details](https://quip-amazon.com/w8xXABrjDAOR#temp:C:JGa77ca9f51172749d9b2fef777c) section is all about. This section is all about “why” we made the choices we did.

Information in this section should not be required reading for implementing the design.

**If there is information that would be required to implement the design, put it in the design details section.**

Delete this block once complete.

## 3.1 Correctness and Reliability

Correctness is the ability of the system to produce expected results. Reliability is the ability of the system to perform correctly for a period of time. Use this section to describe why we believe the design will produce correct results, and will continue to produce correct results over time.

Think of this section in terms of a process, where the consumer is expecting to provide some inputs and get an expected output. If your design can’t be thought of in terms of that model, then this section could just contain “N/A”.

Delete this block once complete.

## 3.2 Performance and Scalability

Some of the relevant facets of performance are latency, throughput, and resource utilization. Scalability is the ability of the system to handle an increasing work load. We are concerned with both vertical and horizontal scalability.

Delete this block once complete.

## 3.3 Testability

Testability is the ease with which the software can be verified and validated. We should consider this at various scopes (unit testing, functional testing, integration testing, and acceptance testing) and from multiple perspectives (Service developer, QA developer, customer). If the test strategy is non-trivial, then a part of the design should be dedicated to describing the design of the tests.

Delete this block once complete.

## 3.4 Security

There are a number of concerns that are security related. Security must be a design consideration at every scope. This section should describe how the design handles these concerns.

Customer data can be sensitive, and is subject to a number of access controls. This section should identify which data is sensitive, and which data is not, along with the relevant treatment for them. This might include encryption, specific access controls, logging, etc.

Bad actors can negatively impact This section should include the threat model for the service/feature, which should refer to the architecture diagram in the main body of the design.

Delete this block once complete.

## 3.5 Failure Handling

Failures are inevitable in a distributed system. A design should consider failure modes and effects. Unlike Reliability and Correctness, failure handling focuses on the result when something goes wrong. This section should discuss the potential failures and how to handle and mitigate the impact of failures.

Think about these questions when writing this section: what should happen when something goes wrong? What if there is an LSE that takes down one or multiple dependencies? What is the customer impact of a failure and how to mitigate? How will it recover from an outage? What level of automation should be used for recovery? How to reduce the blast radius? Have you considered both the impact of failures and likely frequency of the failures?

Delete this block once complete.

# 4 Operating Cost Estimate

Both building and operating a system cost development effort and time. Striking a balance between being frugal and “frupid“ requires knowing when to make trade-offs between design choices based on those costs. One of the first steps to doing this is to estimate the cost of operating the system that your design imposes.

Operating cost refers to the cost incurred by the service owner to provide service to customers.

This section should identify the portions of the design that have operating cost, and how those relate to control factors such as data volume, call volume, concurrency, and latency.

Delete this block once complete.

# 5 Dependencies

Is this change dependent on other changes or actions? Are there prerequisites?

Delete this block once complete.

# 6 Approvals

This section tracks approvals for the design. Some designs will need only one approval (internal to a team, for example), while others may require several.

In general, don’t sign someone else’s name in this table. Let them add their entry, so that the edit history shows that they did the approval. Also, don’t trust an approval that wasn’t added by the person listed in the reviewer column unless you ask them if they approved.

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewer** | **Date Approved** | **Review Purpose** | **Notes** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 7 Appendices

Add supporting detail in this section that would otherwise break up the flow of the document.

Delete this block once complete.

## 7.1 Related Documents

Are there any existing design docs which this design impacts that need updating?

It’s highly beneficial to have our design docs reflect the reality of the system, or where the system is headed. If we change course, they should be updated. This helps with ramp up, knowledge sharing, and driving implementation decisions.

Delete this block once complete.

# 8 References

Provide links to relevant reference material.

Delete this block once complete.

# 9 Open Issues

Track unresolved questions and/or actions here, remove or strike them out when they have been resolved.

Delete this block once complete.