<Project Title>

Design Document

Table of Contents

[Problem Statement 2](#_Toc186724774)

[Use Cases 2](#_Toc186724775)

[Project Scope 2](#_Toc186724776)

[In Scope 2](#_Toc186724777)

[Out of Scope 3](#_Toc186724778)

[Open Questions 3](#_Toc186724779)

[Functional Requirements 3](#_Toc186724780)

[U1: <User story description> 3](#_Toc186724781)

[U1.1: <Title> 3](#_Toc186724782)

[UX Mockups 3](#_Toc186724783)

[Proposed Architecture 4](#_Toc186724784)

[Architectural Overview 4](#_Toc186724785)

[Architectural Diagram 4](#_Toc186724786)

[Chosen Technologies 4](#_Toc186724787)

[Technology 1: <Name> 4](#_Toc186724788)

[Alternative Technologies 4](#_Toc186724789)

[Alternative 1: <Name> 4](#_Toc186724790)

[Service Overview 4](#_Toc186724791)

[API Design 5](#_Toc186724792)

[Public Models 5](#_Toc186724793)

[Model 1: <Name> 5](#_Toc186724794)

[Endpoints 5](#_Toc186724795)

[Endpoint 1: <URI> : <HTTP Method> 5](#_Toc186724796)

[Data Design 5](#_Toc186724797)

[Table 1: <Name> 5](#_Toc186724798)

[Development Operations 6](#_Toc186724799)

[Source Control & Standards 6](#_Toc186724800)

[Coding Language(s) & Standards 6](#_Toc186724801)

[<Component> : <Language 1, ex: Python 3.11> 6](#_Toc186724802)

[Containerization / Build Platform 7](#_Toc186724803)

[Component 7](#_Toc186724804)

[Build Platform 7](#_Toc186724805)

[Component 1 7](#_Toc186724806)

[Ex: Docker 7](#_Toc186724807)

[Testing Details 7](#_Toc186724808)

[Code Review Process 7](#_Toc186724809)

[Deployment Process 7](#_Toc186724810)

[Logging, Monitoring, & Metrics 7](#_Toc186724811)

[Tasks and Milestones 7](#_Toc186724812)

[Milestone 1: <Title> 8](#_Toc186724813)

[Task 1.1: <Title> 8](#_Toc186724814)

[Appendix 1: Definition of Terms 8](#_Toc186724815)

[Appendix 2: References & Citations 9](#_Toc186724816)

[Appendix 3: Design Document Resources <You can delete later> 9](#_Toc186724817)

# Problem Statement

*Explain clearly what problem you are trying to solve.*

# Use Cases

*This is where we work backwards from the customer and define what our customers would like to do (and why). You may also include use cases for yourselves, or for the organization providing the product to customers.*

* U1. As a <product> customer, I want to <result> when I <action>
* U2. …

# Project Scope

*Clarify which parts of the problem you intend to solve. It helps reviewers know what questions to ask to make sure you are solving for what you say and stops discussions from getting sidetracked by aspects you do not intend to handle in your design.*

## In Scope

*Which parts of the problem will you solve with this design? This should include the base functionality of your product. What pieces are required for your product to work?*  
  
*The functionality described above should be what your design is focused on. You do not need to include the design for any out of scope features or expansions.*

* <In Scope Feature 1>
* <In Scope Feature 2>

## Out of Scope

*Based on your problem description, are there any aspects you are not planning to solve? Do potential expansions or related problems occur to you that you want to explicitly say you are not worrying about now? Feel free to put anything here that you think your team can't accomplish during this project, but would love to do with more time.*

* <Out-of-scope Feature 1>
* <Out-of-scope Feature 2>

# Open Questions

*Use this space for any questions you still need to answer as you update/review this document.*

1. <Question 1>
2. …

# Functional Requirements

*Plan further details for your use cases, including Happy Path and Error cases. These requirements are what you will use to create your test plan.*

*This should follow the given/when/then format. (AKA preconditions, action, postconditions/invariants)*

## U1: <User story description>

### U1.1: <Title>

*Title is often in given\_when\_then format (similar to writing test names)*

GIVEN: <Preconditions>

WHEN: <Action>

THEN: <Postconditions / Invariants>

# UX Mockups

*Include mock-ups of the web pages you expect to build. These can be as sophisticated as mockups/wireframes using drawing software, or as simple as hand-drawn pictures that represent the key customer-facing components of the pages. It should be clear what the interactions will be on the page, especially where customers enter and submit data. You may want to accompany the mockups with some description of behaviors of the page (e.g. “When customer submits the submit-dog-photo button, the customer is sent to the doggie detail page”)*

# Proposed Architecture

*Describe broadly how you are proposing to solve for the requirements you described earlier.*

## Architectural Overview

*This should include what web services or other technologies you plan to use, as well as an architecture diagram.*

*You should describe your technical decisions, and argue why the technologies you chose are reasonable. Describe how each component will behave as part of your system.*

## Architectural Diagram

<Diagram>

## Chosen Technologies

### Technology 1: <Name>

## Alternative Technologies

*Describe alternative technologies or strategies, and why you did not choose to use them.*

### Alternative 1: <Name>

## Service Overview

*This may include class diagram(s) showing what components you are planning to build.*  
*You should argue why this architecture (organization of components) is reasonable. That is, why it represents a good data flow and a good separation of concerns. Where applicable, argue why this architecture satisfies the stated requirements.*

# API Design

## Public Models

*Define the data models your service will expose in its responses.*

### Model 1: <Name>

<Class Diagram>

## Endpoints

### Endpoint 1: <URI> : <HTTP Method>

*Describe the behavior of the first endpoint you will build into your service API. This should include what data it requires, what data it returns, and how it will handle any known failure cases. You should also include a sequence diagram showing how a user interaction goes from user to website to service to database, and back.*  
  
*(You should have a separate section for each of the endpoints you are expecting to build)*

Request: <Data Model>

Response: <Data Model>

Error Responses:

* <Error Code>: <Condition>

Sequence Diagram:

<Diagram> (suggest also including source code for it, if possible)

# Data Design

*Define the tables you will need for the data your service will use. It may be helpful to first think of what objects your service will need, then translate that to a table structure. This should include any entity-relationship diagrams to describe how your different tables will be related.*

<Entity-Relationship Diagram(s)>

## Table 1: <Name>

Key Schema:

* <Attribute name> : <Type>

Attributes

* <Attribute name> : <Type>

Foreign Keys:

* <Foreign Table> : <Attribute name> : <Type>

# Development Operations

*Define what tools you will use to build, test, deploy, release, and monitor your service.*

*Some examples of what to include:*

## Source Control & Standards

*Example:*

*We will use Git for our source control platform.*

*We will create 1 feature branch for each milestone, and 1 branch from the milestone branch for each task. Changes from tasks will be pulled to the milestone branch as they are completed. Changes in milestone branches will be pulled to `mainline` on a regular cadence (1 business week, on Mondays at 1pm PST).*

*To maintain clean branches, we will use `rebase` instead of `merge` for pulling changes to other branches.*

*We will follow the following commit message template:[[1]](#footnote-1)*

1. # Title: Summary, imperative, start upper case, don't end with a period

2. # No more than 50 chars. #### 50 chars is here: #

3.

4. # Remember blank line between title and body.

5.

6. # Body: Explain \*what\* and \*why\* (not \*how\*). Include task ID (Jira issue).

7. # Wrap at 72 chars. ################################## which is here: #

8.

9.

## Coding Language(s) & Standards

### <Component> : <Language 1, ex: Python 3.11>

* We will follow the [PEP 8 style guidelines](https://peps.python.org/pep-0008/)
* We will follow the [PEP 257 docstring conventions](https://peps.python.org/pep-0257/)
* We will enforce conventions using the following tools:
  + [Flake8](https://flake8.pycqa.org/en/latest/) for linting
  + [isort](https://pycqa.github.io/isort/) for import sorting
  + [black](https://black.readthedocs.io/en/stable/) for formatting

## Containerization / Build Platform

|  |  |
| --- | --- |
| Component | Build Platform |
| Component 1 | Ex: Docker |
|  |  |

## Testing Details

*Ex:*

*We will use the `unittest` library for unit testing our python components.*

*We will use `pytest` for integration testing our python components.*

*We will maintain 95% code coverage for all python components.*

## Code Review Process

*How many required approvals? SLA time on CR feedback. How long are approved CRs allow to sit before being pushed to the respective branches?*

## Deployment Process

Do you have a continuous deployment pipeline? Do you require a manual approval step? Do you plan to have a blue-green deployment strategy?

## Logging, Monitoring, & Metrics

What platform do you plan to use for monitoring? What alarms do you want to set, and how do you want to respond to them? Where will you store your logs? Do you want to perform any log processing?

# Tasks and Milestones

*Break down the effort this project will require into tasks and subtasks. Then group tasks into milestones.*

*Tasks should cover a vertical slice of your effort – for example, each API endpoint may constitute a task. Creating the database table may be a subtask, as would creating the API, creating the business logic, and so forth.*

*You may also choose to split some tasks or milestones by horizontal slices – maybe you want to create a separate milestone for creating the initial architecture, or one for implementing your UX.*

*If you need to perform any additional investigation, create tasks for those as well.*

## Milestone 1: <Title>

### Task 1.1: <Title>

#### Subtask 1.1.1: <Title>

# Appendix 1: Definition of Terms

*Use this space to define terms or acronyms that your audience may not be familiar with.*

|  |  |
| --- | --- |
| Term | Definition |
| UX | “User Experience” - |
| SLA | “Service Level Agreement” – agreement on what will be delivered, and under what timelines. |
| CR | Code Review |
| URI | “Uniform Resource Identifier – a system for identifying resources. In our case, this means the path to the resource. |
| URL | “Uniform Resource Location” – a system for identifying locations of resources. Often includes a protocol, a domain name, and a path to the resource. |
| PEP | “Python Enhancement Proposals” |
| linting | The process of using a lint tool, or a static code analyzer, to automatically check source code for errors. |

# Appendix 2: References & Citations

1. Cbeams. “How to Write a Git Commit Message.” *Cbeams*, cbeams, 27 May 2023, cbea.ms/git-commit/.
2. “Use a Git Commit Message Template to Write Better Commit Messages.” *Gist*, lisawolderiksen, gist.github.com/lisawolderiksen/a7b99d94c92c6671181611be1641c733.
3. van Rossum , Guido, et al. “PEP 8 – Style Guide for Python Code.” *Python EnhancementProposals (PEPs)*, 5 July 2001, peps.python.org/pep-0008/.
4. Goodger, David, and Guido van Rossum. “PEP 257 – Docstring Conventions.” *Python Enhancement Proposals (PEPs)*, 29 May 2001, peps.python.org/pep-0257/.
5. “Flake 8: Your Tool for Style Guide Enforcement.” *Flake8*, flake8.pycqa.org/en/latest/#.
6. “Isort.” *Isort*, pycqa.github.io/isort/.
7. “The Uncompromising Code Formatter.” *Black 24.8.0 Documentation*, black.readthedocs.io/en/stable/.

# Appendix 3: Design Document Resources <You can delete later>

*Here are some resources to help you create your design document.*

|  |  |
| --- | --- |
| Site | Use(s) |
| plantuml.com | Sequence/Class/Entity Relationship Diagrams |
| https://app.diagrams.net/ | Architectural Diagrams / Mockups / Other Diagrams |
| https://docs.aws.amazon.com/ | AWS Documentation |
| https://aws.amazon.com/what-is/restful-api/ | What is a RESTful API? |

1. Based on “Use a Git Commit Message Template to Write Better Commit Messages.” Gist, lisawolderiksen, gist.github.com/lisawolderiksen/a7b99d94c92c6671181611be1641c733. [↑](#footnote-ref-1)