Design Doc Template

Author(s): xyz

Date: 22/05/2019

Revision: 0

Document Status: Draft [Draft, Completed, Submitted, Reviewed, Final]

Project Status: In-Progress [In Review, Approved, In-Progress, Completed]

Revision History

Date	Revision	Description	Author
22/05/2019	0	Initial draft of the design doc template	xyz

Table of Contents

Addendum

Introduction

Summary

The present public transport system(esp. in Hyderabad),suffers from issues such as :

People risking their lives to reach destinations on time (hanging from the entrances, termed "foot-boarding"), but still being late - just because they can never guess when the next bus will come, if there is any.

Also, TSRTC, the organization associated with public transport in this state, is running in losses due to low utilization of City buses resulting from the above problems.

Background

The problem statement, what are the issues you are trying to solve, what gaps currently exist, etc.

We are trying to provide a solution to the existing operational inefficiency of the organization in running City buses, by attempting to reduce the communication gap existing between the passengers and the organization.

What are the existing tools that address these issues or gaps, and why do they not satisfy this problem? Why can't the existing tools be updated to solve this problem.

Long ago, at least not in the recent past, manual surveying and laying out of bus routes according to passengers' convenience took place (in a less-synchronized way), and that is probably the method being followed till today, to update the schedules on various routes. This can be inefficient, since this takes a long time to complete and also the requirements of the passengers are bound to change over time.

What overlaps will your solution have and how is it differentiated from the existing tooling? What are the gains / wins from creating this tool / solution?

Our solution also helps in updating the schedules of the bus routes in a much similar way as to the manual surveying presently done by the organization - with the difference being that analysis of the data provided by our solution can be used to update them more frequently, than what is possible now.

Definitions, Acronyms, and Abbreviations

Definition of terms that will help readers understand the documents, or acronyms common in your project area.

Solution: the product we are developing, and any future versions of it, will be referred to as thus.

Blynk App: a third-party application which provides our solution's interface.

GPS Tracker: a device used to get the location of any object in real-time, by using the Global Positioning System.

NodeMCU: an open-source firmware and development kit used to prototype our solution.

Arduino:

Design Overview

Requirements

Project requirements, this can include requirements from customers, partners, or overseeing teams. The requirements for the project, this may include subsections for various types or sources of requirements.

Arduino, NodeMCU, breadboard,connecting wires,ultrasonic sensor,buzzer,GPS Tracker,LED Lights, USB cable, switch and batteries.

Tracking the buses to notify the users about its location.

Getting the statistics stored in a representable form.

Documentation

If the project requires any wiki pages, code comments, presentations, etc. that information should be included here

Minimum Viable Product

A detailed description of the deliverable for this project, this is the minimal functionality required for the project to be considered successful and should not include stretch goals or future work.

Stretch goals

Stretch goals include functionality beyond the scope of the minimum viable product that should be include in the project should time and budget permit. Unlike future work, stretch goals would be smaller tasks for features in support of the minimum viable product.

Future work

This may include ongoing support, expansion of the original scope, work that requires transitions in project ownership, or details of projects designed to be broken up into multiple phases.

Architectural Diagrams

UML diagrams describing the project. You can look for any open source diagramming tool for the same.

System Diagrams

This section, sometimes referred to as a 'System Context Diagram', would typically consist of a UML diagram which illustrates the boundary and interaction with external systems. For a CLI this may show which backend systems provide the data, for services it may indicate dependencies such as databases, Kafka, or downstream services.

Application Programming Interface

For services, libraries, and command line interfaces that present an interface which can be wrapped

Recommendations

Using a versioned endpoint simplifies the process of making future backwards incompatible API changes;

/api/v0

User Interface

For frontends, a mockup can be attached to illustrate the user interface. Command line interfaces may include a list of subcommands and their options.

Data Models and Storage

For projects requiring messages queue such as Kafka, MySQL, etc.

Kafka

- How many partitions are needed for this topic?
- How many days of retention will be needed?
- What will the partitioning key become?
- How much data will be written to the topic during peak hours?
- What type of Kafka cluster will be needed? (E.g. aggregate, queuing, tracking, metrics, logging)

MySQL

- What does the table schema look like and how are they all tied together (provide a UML)?
- What sort of updates will be made to the tables?
- How will users make queries to the tables? (e.g. Complex joins, pre-filtering, single record gets)
- What the strategy for indexing?

Service Operability

Key Performance Indicators

Key performance indicators (KPI), describe how a service should be monitored and how its performance can be gauged. This would typically include an overview of the types of metrics an application will need to emit, call time, error rate, etc.

Service Level Objectives

Service level objectives (SLOs), set targets for various KPI through alerts via email or SMS, these targets may provide early indicators of approaching a capacity limit, changes in load patterns through various phases of an application, changes in duration of offline processing, etc.

Project Overview

Communication and Tracking

Any relevant distribution lists, slack channels, taiga projects, etc

Risks

Anything that may put the project at risk; potential delays, dependence on work done by other teams, hardware procurement, or reviews.

Risks may also include assumptions of the project's external dependencies. These may be lower level project dependencies that are tracked outside of the immediate project scope and assumed to already be in place. For example, "project x assumes OS version y will be available in the market."

Milestones

Estimated dates when planning steps, deliverables, and reviews will be completed

Project Phases

For projects that are better tracked and reported on in multiple phases because of extended timelines, external dependencies, etc

Cost

Level of effort, number of resources, number of hours or weeks, unlike milestones which tracks project time cost should only include engaged time.

1) For the all tasks which are deliverables/visible on user-end side needs to be documented as stories.

2) Need to guess/estimate the time required in number of hours for the completing that stories which can be captured in taiga.

3) Assign that task to the right person and document the actual time taken for completing that task.

Frequently Asked Question

References

Links to any supporting documentation, other projects, or reference material

Addendum

Additional diagrams or details that do not particularly belong in the body of the design doc. This could also be a place to describe additional examples that would otherwise bloat the introduction section. More specifics on APIs could also be placed here for engineers to reference.