# Overview

Problem statement is to build a package indexer which keeps track of package dependencies. Clients will connect to the server and inform which packages should be indexed, and which dependencies they might have on other packages. We want to keep our index consistent, so the server must not index any package until all of its dependencies have been indexed first. The server should also not remove a package if any other packages depend on it.

# Design Rationale

Goal of this doc is to provide insight into how the system would work in production. Current solution submitted runs on a single host with an in-memory datastore but when this is deployed to production, several things will need to be considered.

## Host Configuration

Real production system typically has a geo load balancer backed by multiple hosts. It is fair to assume we'll have multiple app servers running on multiple hosts.

In a real production system, you'll also have to think of security implications. A threat model is a good way to do it. A simple example would be adding things like rate limiting or configuring the load balancer to accept x number of connection to prevent DDOS.

## Data Recovery

Submitted solution implements an in-memory datastore which will not work with multiple hosts. In addition, if the application crashes we would lose the index without the ability to rebuild it.

In a production system, we need a persistent distributed data storage. Several options are available in the market including graph & KV databases. Current solution does implement the in-memory datastore behind an interface so you could seamlessly switch the implementation.

## Code Improvements

1. Currently, we do not have a Package class; but it will be good to have a Package class, if more information about the package is to be stored. Example, it's size, checksum, author, dependencies, etc.
2. Logging can be improved with log4j or some other external logging libraries.
3. Edge case that can be handled : What will happen if a package is a dependency of itself. (Not handled currently).