**AnyChange**

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

# Architectural goals and philosophy

The architecture of AnyChange has to be adaptable to integrate with many different external product selling platforms and has to be scalable to meet performance criteria of growing customer base. It should provide a secure way of communicating between the users and application software.

# Assumptions and dependencies

The team is composed developers with varying experience. The project will be built from ground up so it is assumed that there won’t be any legacy interfaces that will be depended on. AnyChange will be a web based software product. It is assumed that there won’t be any publicly available API’s to pull price data from when collecting product data, if there is however, the system should be able to use that as well.

# Architecturally significant requirements

* All data, hence the database applications and clients are expected to be secure.
* Functional uptime of the system will be 166 hours per week.
* System shall scale up to support 10000 concurrent transactions.

# Decisions, constraints, and justifications

* In order to achieve flexibility required to accommodate several different platforms with the assumption that there is no publicly available API’s to pull price data from, AnyChange will use an external scripting platform that can scrape data from price sources. This will enable us tailor custom solutions for each price source so that we can monitor wide category of products and sources.
* AnyChange will be built using Model-View-Controller architectural pattern. It is a pattern readily supported by Spring Boot framework and allows for separation of data, business logic and presentation of data. This will allow us to develop and create an initial product faster.
* AnyChange will be developed using Java to enable cross-platform operation.
* AnyChange will use Python as is external scripting platform.

# Architectural Mechanisms

## **Scripting Platform**

This mechanism is intended to provide fast to develop, flexible platform to develop web scraping solutions assuming the price sources do not provide free, easy to access API’s to pull price data for products.

# Key abstractions

System will be composed of products and users. Each user will be authenticated to use the system and users will be able to monitor products that are added to the system. The products can have many different price sources. Each price source is responsible for providing price points at any point of time. A product and price source will have price information, which includes the products current price and price history.

# Layers or architectural framework

When using MVC pattern, the product and users will form the model part of the architecture. The controller part will be provided by the software and it will be responsible for reacting to user events and doing necessary business logic that will result in data that can be presented to user. The view will be made to represent the data produced by the controller.

# Architectural views

