Spot-On

Architecture/Design Document

**Table of Contents**

[1 Introduction 3](#_Toc131876418)

[2 Design Goals 4](#_Toc131876419)

[3 System Behavior 4](#_Toc131876420)-5

[4 Logical View 5](#_Toc131876421)

[4.1 High-Level Design (Architecture) 5](#_Toc131876422)

[4.2 Mid-Level Design 6](#_Toc131876424)

[4.3 Detailed Class Design 8](#_Toc131876425)

[5 Process View 11](#_Toc131876426)

[6 Development View 12](#_Toc131876427)

[7 Physical View 12](#_Toc131876428)

[8 Use Case View 12](#_Toc131876429)

Change History

**Version:** 0.3

**Modifier:** John Asare, Lam Nguyam, Jonathan Dias

**Date:** 03/01/2018

**Description of Change:** Implemented the backbones of the databease and the code for demo day.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** 1.0

**Modifier: John Asare, Lam Nguyam, Jonathan Dias**

**Date:** 03/22/2018

**Description of Change:** Uploaded everything to git. Got django runnung and the server

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:**

**Modifier:**

**Date:**

**Description of Change:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

This document describes the architecture and design for the Spot-On application and website that is being developed for all user with all ages, Spot-On is a modern website with its mision to perform elegant and gives relevant results. Spot-On has a rich sreach engine algorithm with a strong databes that works together to gives its users a better search results when a user search a product that have a good reviews. What makes Spot-On different from other serach engine is that it does not gives reference of other sites as a results but it actually crawl through references sites and brings the product as a result for the user to purchase.

The purpose of this document is to describe the architecture and design of the Spot-On application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

* Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
* Developers – they want an architecture that will minimize complexity and development effort.
* Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all logic related to GPS coordinates.
* Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no one diagram or model that can easily express a system’s architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the Spot-On application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

# Design Goals

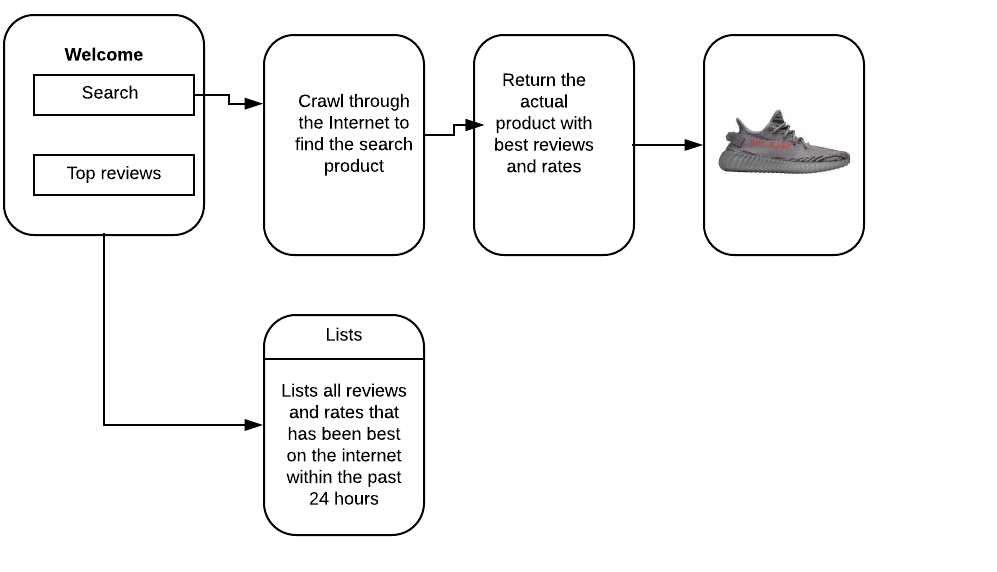
There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The priorities for the design that follows are:

* The design should minimize complexity and development effort.
* The design should take into account the development environment which is 3 small teams with complementary skills that work across time and space. We meet twice a week but we work independently and log our work through Jira. The concerns of each component should be narrow so that each team can specialize on a particular technology or skill.
* The design shouldn’t inhibit reusability. The two previous design goals are more important, but the ability to reuse components is also desirable.
* Simplicity, friendly and powerful is our model and blueprint of our design

# System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows.



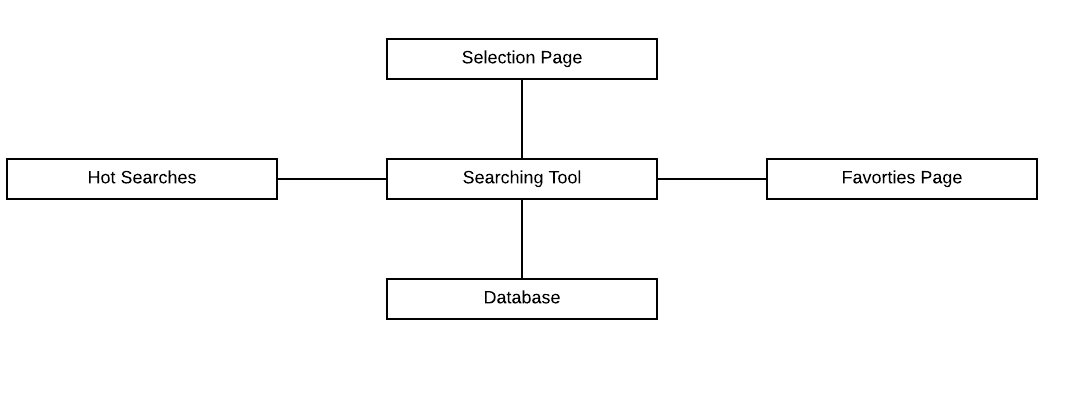
1. **Logical View**

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of Spot-On is first expressed in terms of high level components (architecture) and progressively refined into more detailed components.

* 1. ***High-Level Design (Architecture)***

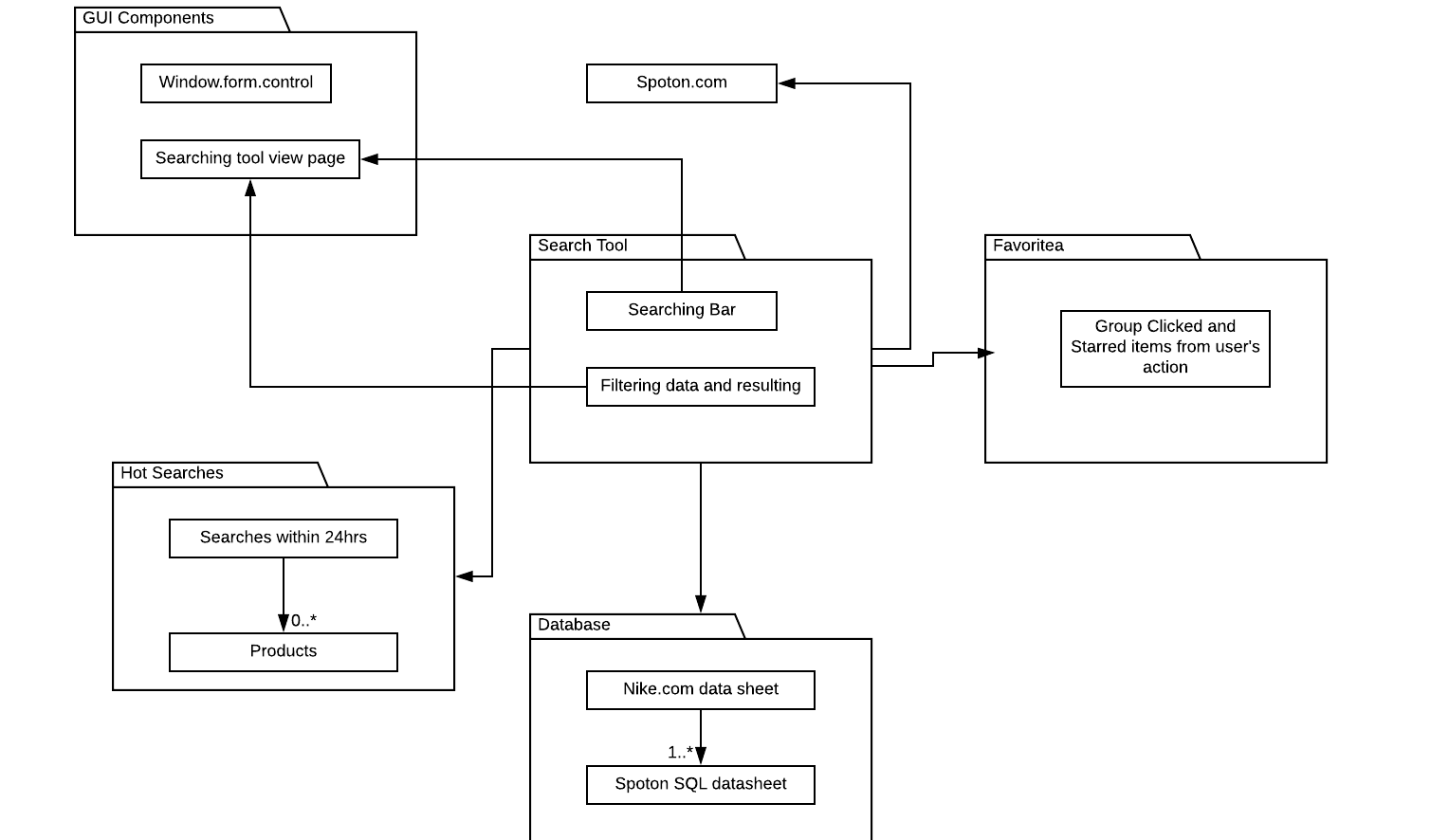
The high-level view or architecture consists of 5 major components:



* The **Selection Sections** provides user’s reasons for visiiting the page. When a user is at the **Selections Sections**, it gives the database a heads up of what the user will be searching for on the site
* The **Searching Tool** is the main page driver of the application. It allows users to filter the database of a specific searches among their typed characters
* The **Database** is a central repository for data on user’s search, crawling through the internet and returning a specific results.
* The Hot Searches provides users a common and highly searches within a timepsan of 24hrs
* **Favorite Page** groups the user’s popular liked and clicked actions

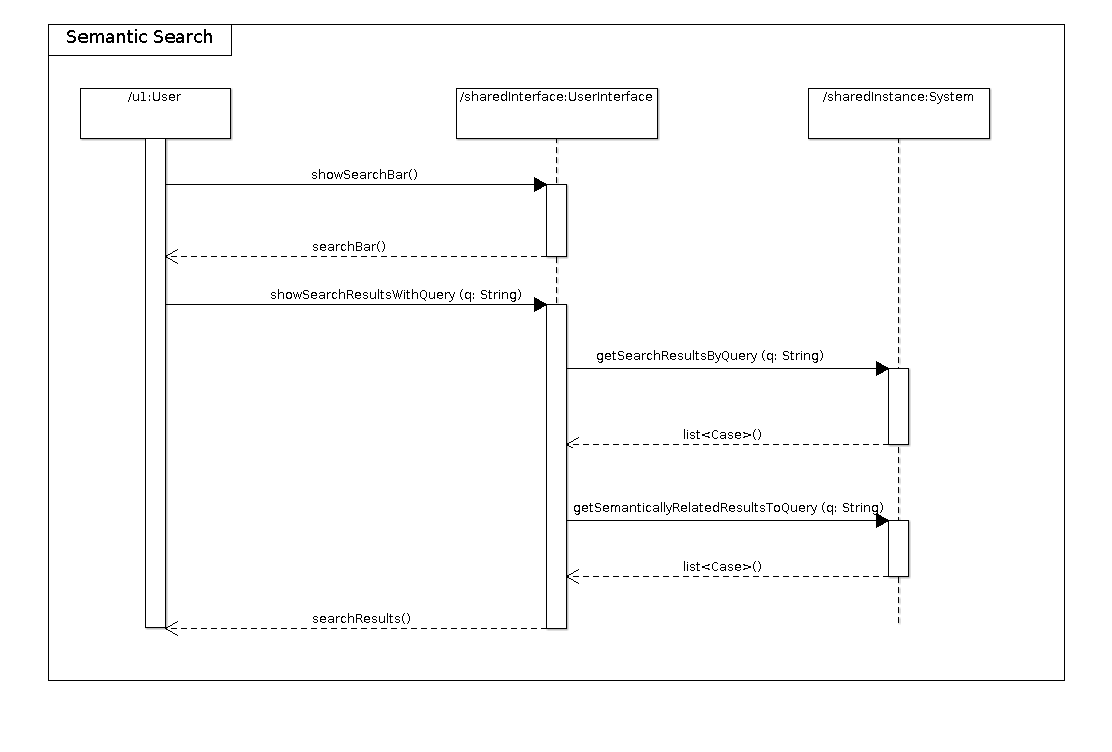
## 

## Mid-Level Design



***This figures shows the mid-level of the project with components and their relationships***

The dynamic behvior of mid-level componenets in basic mode



***This figure shows the dynamic behavior of mid-level componenets in self-directd mode.***