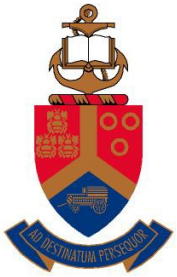

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Squeaky Clean
Requirements Specification

10 July 2017

Contents

1 Project Background

Squeaky Clean is a company focused on providing car wash services to the community of Hatfield. Sqeaky clean has requested that Computer Science students of UP create a static, informative website for the marketing of the business. This website should include information about the companies vision, trade and other relevant information.

2 Vision and Scope

The system should be a simple, informative website that highlights the key components of the company. The aim of the website is to attract more customers for the client through a web footprint. Users can post queries on the website. When posting a query, both the relevant people of the company and the customer should be notified.

3 Design Specifications

The system should be designed in such a way that the mailing service can easily be plugged in and out of the system i.e. it should be loosely coupled from the rest of the website. The following are the design specifications for the website:

- Any updates to the website will be handled by the administrator; No admin interface is required.
- The Anonymous User interface will allow anyone who navigates to the site to browse the entire website
- When a user posts a query, he/she should be given the opportunity to opt-out of receiving further notifications. However, the user should expect to receive a response to the query posted. The relevant people within the company should respond to the email query.

4 Design Technologies

The following technologies should be used to implement the system:

- Html 5 (Html and Bootstrap CSS)
- Angular2
- Spring Boot
- Spring MVC
- Apache Maven
- Git (Github)
- PostgreSQL Database

5 Architectural Technologies

The web application will consist of two subsystems that communicate via HTTP using REST Framework. The Java/Spring Boot application will be known as the "backend" application. The HTML5/Angular2 application will be known as the "frontend" application. The backend application is expected to communicate with the database and use Hibernate which can be imported into Maven, a dependency management tool whereas the frontend application will be hosted in the browser and NodeJS is expected to manage packages required for the application to run successfully.

There are 2 choices on how to implement the backend application. The first choice is to use a monolithic architecture and the second one is to use a microservices architecture.

5.1 Monolithic Implementation

The monolithic implementation will consist of one backend **spring boot** application that will be deployed as a single unit. Apache Maven will be used as the dependency management tool.

5.2 Microservices Implementation

The microservices implementation will consist on n spring boot applications where n = no. of modules or components in the system. These applications will be integrated by using spring cloud and Netflix OSS Zuul Integration. The integration server will be regarded as one component of the n components to be implemented.

5.3 Front-end Application

The frontend application will be an Angular2 application that will communicate with the backend application via the HttpModule frovided by angular2. There are a number of generators that one can use to create an angular2 skeleton application. Links listed in the *Further Reading* section.

6 Further Reading

Below is a list of links that you can read up on to familiarise yourself with the relevant technologies. Take special note of the links related to the monolithic and microservices implementations.

- **Spring Boot** - `https://spring.io/guides/gs/spring-boot/`
- **Spring Cloud** - `https://piotrminkowski.wordpress.com/2017/02/05/part-1-creating-microservice-using-s`
- `http://www.baeldung.com/spring-rest-with-zuul-proxy`
- **Angular2** - `http://www.angular2.com/`
- `https://angular.io/tutorial`
- **Maven** - `https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html`

  Additions to the user interface may be added if desired but these additions must serve a purpose.