Nailed Opinion Mining

High Level Design Document

by Hakan Akdag <hakanakdag46@gmail.com>

Version: 0.1

# Revision History

| Date | Author | Version | Change Reference |
| --- | --- | --- | --- |
| 05/01/2019 | Hakan Akdag | 0.1 | Initial Version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc10241155)

[1. Introduction 4](#_Toc10241156)

[2. Goals, Objectives, and Rationale for the System 5](#_Toc10241157)

[2.1. Project Purpose 5](#_Toc10241158)

[2.2. System Goals and Objectives 5](#_Toc10241159)

[2.3. Proposed System 5](#_Toc10241160)

[2.3.1. System Scope 5](#_Toc10241161)

[2.3.2. Business Processes Supported 5](#_Toc10241162)

[2.3.3. High-Level Functional Requirements 6](#_Toc10241163)

[2.3.4. Summary of Changes 6](#_Toc10241164)

[3. Factors Influencing Technical Design 7](#_Toc10241165)

[3.1. Relevant Standards 7](#_Toc10241166)

[3.2. Assumptions and Dependencies 7](#_Toc10241167)

[3.3. Constraints 7](#_Toc10241168)

[3.4. Design Goals 7](#_Toc10241169)

[4. Proposed System 8](#_Toc10241170)

[4.1. High-Level Operational Requirements and Characteristics 8](#_Toc10241171)

[4.1.1. User Community Description 8](#_Toc10241172)

[4.1.2. Non-Functional Requirements 8](#_Toc10241173)

[4.2. High-Level Architecture 9](#_Toc10241174)

[4.2.1. Application Architecture 10](#_Toc10241175)

[4.2.2. Information Architecture 11](#_Toc10241176)

[4.2.3. Technology Architecture 13](#_Toc10241177)

[4.2.4. Security and Privacy Architecture 13](#_Toc10241178)

# 1. Introduction

This document describes the high level architecture for the Nailed Opinion Mining project. Purpose of this document is to base project’s goals, technical design, proposed system, high level architecture, and analysis of the proposed system.

This document is the entry point to the project. It is highly related to the low level design document. The intended audience for the document is managers and the development team. The high level technical design is completed during the concept phase and it is intended to describe the conceptual design of the proposed system.

This document provides a framework for more detailed requirements and design activities in later phases of the project.

The requirement is to evaluate the feedbacks from the users using opinion mining.

# 2. Goals, Objectives, and Rationale for the System

This section describes the purpose of the project, its goals, and the proposed system.

## 2.1. Project Purpose

The fundamental purpose of this project is to create a new system to evaluate user feedbacks taken for various products.

## 2.2. System Goals and Objectives

Track and centralize user feedbacks, evaluate them, and creating a feedback service which uses opinon mining. Feedbacks from the users carry a very important in today’s World. Feedbacks may be scatterred accross the different application for a single product or service. Evaluating and centralizing them is the solution which this project brings.

## 2.3. Proposed System

In this system, users will see for the product of their interest a more accurate evaluation. System will centralize feedbacks from the different sources, and give points by using opinion mining.

### 2.3.1. System Scope

Scope of the initial version of the system is limited to products only. Products will have various categories. Categories and products will be entered manually. Feedbacks will be added to the system manually via selecting the related product. Later in the future versions of the system, products and their feedbacks will be collected via a data collector service.

### 2.3.2. Business Processes Supported

Main flow of the system will start with selecting the product of interest, and then entering the feedback into the feedbacks section. Later, administrator will approve the given feedback. Then, opinion mining service will evaluate the feedback.

Entering a new product into the system will start with selecting (or entering a new one if it does not exist) category and entering the product information.

Approving a feedback will start by looking at the feedbacks list/grid, selecting one to approve, viewing it and then approving if it is appropriate.

### 2.3.3. High-Level Functional Requirements

* Anonymous users will be able to view products and their feedbacks, but they cant contribute.
* Anonymous users will be able to register to the system or adminis-trator user will create their accounts manually.
* Registered user can view a product and add a feedback.
* A feedback can be editted or deleted until it will be approved. After approval, no changes will be allowed.
* A registered user can only enter single feedback for a product.
* Administrator user will add products and their categories to the system.
* Administrator user will approve or disapprove feedbacks entered by users.
* System will be open to the web but opinion mining service will be accessible internally only.

### 2.3.4. Summary of Changes

As this is initial version, no changes.

# 3. Factors Influencing Technical Design

This section describes the standards, assumptions, and constraints that influence the technical design of the proposed system.

## 3.1. Relevant Standards

Database – Relational (Dapper) and/or non relational (MongoDB)

Inputs – Entered through text field and stored in database.

Security – Username and password are required for access to the system. Bearer token will be used for secured API calls.

Quality – By keeping the interface simple and direct, quality should be kept at maximum.

## 3.2. Assumptions and Dependencies

Main assumption of the system is that system will accept unique products and their categories. Anything found in the web sites or product catalogs are subject to input of this system.

System is highly dependent on real user comments. Therefore, another mechanism to collect user comments from other sites is necessary.

## 3.3. Design Goals

User Interface will be very simple and user friendly. UI will be developed by mobile first approach. UI and backend will be fully decoupled (NextJS).

System must be scalable.

System will be hosted in a cloud solution (AWS).

# 4. Proposed System

This section describes the operational requirements and technical design of the proposed system.

## 4.1. High-Level Operational Requirements and Characteristics

System is designed to handle over a million of users and thousand requests per minute.

### 4.1.1. User Community Description

| User Group | Description/Expected Use of System | Total Users | Concurrent Users |
| --- | --- | --- | --- |
| Admin | Content editor  Feedback Approvement  User Management | 1 | 1 |
| Anonymous | Not registered users  Able to view products and their feedbacks | \* | \* |
| Registered User | Registered users  Able to view products and their feedbacks  Able to enter new feedback for each product | \* | \* |

Table 1 - User Community Description

### 4.1.2. Non-Functional Requirements

* Performance: System must response in 2seconds mostly
* Scalability: System must be scalable
* Capacity: Over a million of products and users can be stored
* Availability: 99%, maintanence will be monthly and for 2 hours.
* Serviceability: Opinion mining will be wrapped by a service which will be available to user by 3rd parties. Feedback service will be available to user by 3rd parties.
* Security: Only authorized users will be able to enter feedbacks

## 4.2. High-Level Architecture

Application will be a multi-page application ([NextJS](https://nextjs.org/)). User interface and the backend will be decoupled. UI will access to the API gateway ([Ocelot](http://threemammals.com/ocelot)). Gateway will send request to the correct API service.

API services may be implemented in any stack and they will have their own databases. Services will not be able to access other databases then their own database. Services may communicate each other using an event bus ([RabbitMQ](https://www.rabbitmq.com/)).

API services will be deployed inside Docker containers so that they can be duplicated if needed in future.

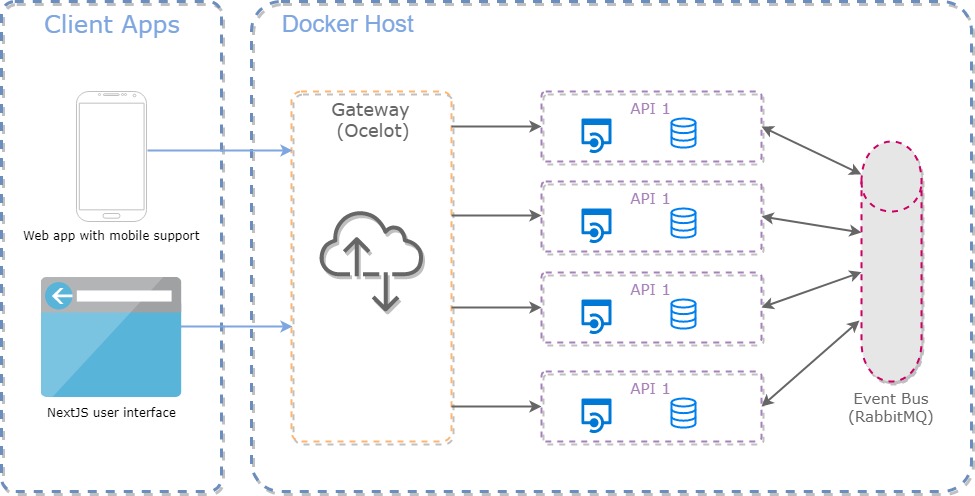


Diagram 1

### 4.2.1. Application Architecture

| Diagram ID | Application Component | Description | Type | Strategy | Pros | Cons |
| --- | --- | --- | --- | --- | --- | --- |
| Diagram 1 | Web Application | Runs on modern web browsers  Mobile first approach  User friendly  Every feature will be usable on the mobile browsers | Online  Operational | Build  Reuse | Browser and mobile friendly approach to reduce development time for mobile devices. | Hard to keep error prone for both mobile devices and desktop browsers. |
| Diagram 1 | API Gateway (Ocelot) | Handles requests from the web application.  Redirect requests to the related API endpoint. | Online  Operational | Build  Reuse | Helps scalability. | Current selected structure doesn’t support feature to show traffic to the API endpoints. |
| Diagram 1 | APIs | Independant Microservices for each business feature. | Online  Operational | Build  Reuse | Highly used APIs can be multiplied across different servers. | Many many services might be hard to maintain in the future. |
| Diagram 1 | Event Bus (RabbitMQ) | Event based messaging system for communication between APIs. | Operational | Reuse | It's fast and it works with good metrics/monitoring.  Ease of configuration.  User friendly admin UI. | Too complicated cluster config and management.  Doesn’t support re-read of consumed messages. |

Table 2 - Description of Application Components

### 4.2.2. Information Architecture

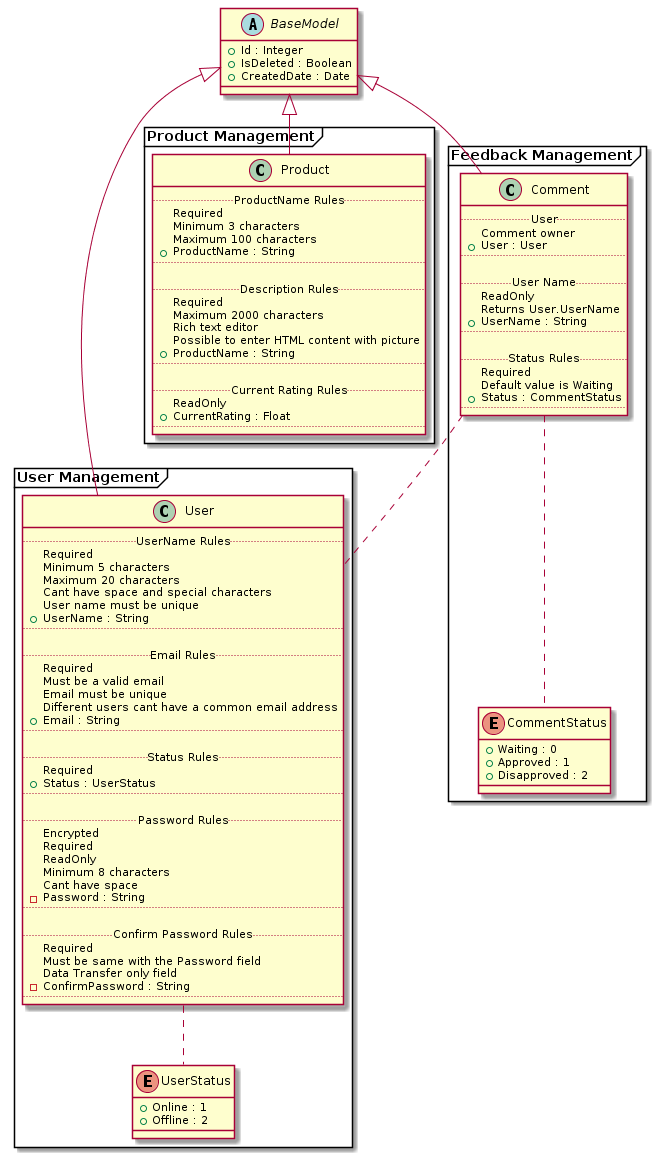


Diagram 2 Classes

| Diagram ID | Conceptual Information | Description | Type of Data Store | System of Record? | Data Acquisition Approach |
| --- | --- | --- | --- | --- | --- |
| Diagram 2 | Product | Core model of the system.  Holds information about the products. | Transactional or NoSql | Through system (Stage I)  Added from external sources (Stage II) | Admin user adds the entries. (Stage I)  Bot adds entries. (Stage II) |
| Diagram 2 | Comment | Second most important model of the system.  Anything user writes about the product. | Transactional or NoSql | Through system (Stage I)  Added from external sources (Stage II) | Registered users add their feedbacks.  Bot adds entries. (Stage II) |
| Diagram 2 | User | Indicates registered users. | Transactional | Through system | Users may register themselves or admin user may create new users. |

Table 3 - Description of Information Components

### 4.2.3. Technology Architecture

There wont be a mobile application. Therefore, web UI must support mobile devices. SOA based architecture is targetted. Because of this requests from UI will be handled at first by a gateway. Gateway will direct requests to the relevant services.

Gateway and Services will be Dockerized. Current design supports only adding service instances manually.

Services will communicate each other via service broker (RabbitMQ).

### 4.2.4. Security and Privacy Architecture

Some of the content (Products and their comments) will be accessable to the anonymous users. Entering new products and feedbacks will be authenticated and authorized.

To prevent spamming, both user interface and backend will take necessary controls. In the UI there will be captcha or Google’s “I’m not a robot” solution and in the backend request queue limit per session strategy will be included.

Authentication will be handled with Bearer token. After the login, token will be created and prevent unauthorized access to the system.

Some of the views will be only accessable by admin user.

Data encryption is not a subject to this project.