Software Requirements Specification

Yahoo! Project
Team E

Version 1.0

Table of Contents

[1] Introduction 1

[1.1] Project Overview 1

[1.2] Team Members 1

[1.3] Document Purpose and Overview 1

[1.4] Application Overview 2

[1.4.1] Objectives 2

[1.4.2] Business Process 2

[1.4.3] User Roles and Responsibilities 2

[1.4.4] Interactions with Other Systems 3

[1.4.5] Replacement of Legacy Systems 3

[1.4.6] Operating Environment 3

[1.4.7] Design and Implementation Constraints 3

[1.4.8] Assumptions and Dependencies 3

[2] System Features 4

[2.1] Extract and Display Aggregation Statistics 4

[2.2] Anomalies 4

[2.2.1] Extract and Display 4

[2.2.2] Anomaly State 4

[2.3] Bug Information 4

[2.3.1] Extract and Display 4

[2.4] Saving Data Changes 4

[3] Use Cases 4

[3.1] Create Bug 4

[3.2] Re-Open Anomaly 5

[3.3] Re-Open and Edit Bug 5

[3.4] Switch User 5

[3.5] User Login 5

[3.6] Stop Alerts 6

[3.7] Enable Alerts 6
[3.8] Refresh Data 6
[3.9] Filter Anomalies and Bugs 6
[4] External Interface Requirements 7
[4.1] User Interfaces 7
[5] Other Nonfunctional Requirements 7
[5.1] Performance Requirements 7
[5.2] Process Requirements 7
[5.3] Security Requirements 7
[5.4] Software Quality Attributes 7

[1] Introduction

[1.1] Project Overview

Yahoo! Inc. is a leading global Internet brand and one of the most trafficked Internet destinations worldwide. Yahoo! is focused on powering its communities of users, advertisers, publishers, and developers by creating indispensable experiences built on trust. Yahoo! is headquartered in Sunnyvale, California.

Within Yahoo, the Strategic Data Solutions (SDS) business unit receives, manipulates, and reports audience data internally. This data includes usage statistics (page views, clicks, time spent, etc.) for all properties (ex: Y! Finance, Y! Taiwan Sports, Y! Brazil Mail) across Yahoo!. Yahoo! uses this data for many purposes including SEC filings, property engagement, and business forecasting. Given these many uses, Yahoo! must ensure the data provided by the audience data is accurate, complete, and consistent. Thus, SDS created a team solely dedicated to data quality within this pipeline.

As part of ensuring data quality, the DQ team uses a variety of statistical modeling and trending to ensure accurate data. If an anomaly occurs, the DQ team notifies owners of the properties. Unfortunately, each property must access a website to view the data anomalies; no application exists to receive data quality updates directly on their computer desktop. Fortunately, Yahoo develops an open application called widgets. Widgets allows for custom applications built upon the widget architecture. Assuming that the widgets architecture is installed on the host machine, the end user can install any created widget onto their desktop. The data quality team wants a widget which will provide data quality information onto end user desktops. This widget will allow each property owner to see his/her property trending data, volume data, and alerts as well as allow for signing off on alerts.

[1.2] Team Members

- Rob Shaw *Team Coordinator* [rjshaw@purdue.edu]
- Jake McDorman *Project Leader* [jmcdorma@purdue.edu]
- Robert Read *Technologist* [rread@purdue.edu]
- Brad Van Dyk *Editor* [bvandyk@purdue.edu]

[1.3] Document Purpose and Overview

The software requirements specifications document is a high-level view of the detailed requirements needed for the widget that is to be designed. It includes the overall description of the project, system features, use cases, broad range of requirements, etc.

[1.4] Application Overview

[1.4.1] Objectives

T he application will provide a solution to Yahoo's difficulty with accessing statics collected from the various systems offered by Yahoo. The domain of this tool is statistic analysis, and as such, the motivation of this software is to aid the user with analysis by providing trending data, volume data, and alerting mechanisms.

[1.4.2] Business Process

Every time a person on the internet accesses a Yahoo site, an entry is logged into the servers. At specified time intervals, a mechanism kicks off to gather all of those entries from across the world. This data is then extracted, transformed into a more readable format, and loaded into a data warehouse. At this point, the data is used in many places around Yahoo.

The data quality team has placed special extraction mechanisms along the pipeline. The team then loads this data into the data quality database for additional trending analysis. Thus, there will be a count of usage statistics from servers, etl activities/warehouse, and internal reporting systems.

After the data quality database is populated, the property owner will be able to use the application to view the statistics collected. This will allow the user to keep track or view anomalies within the database at real-time.

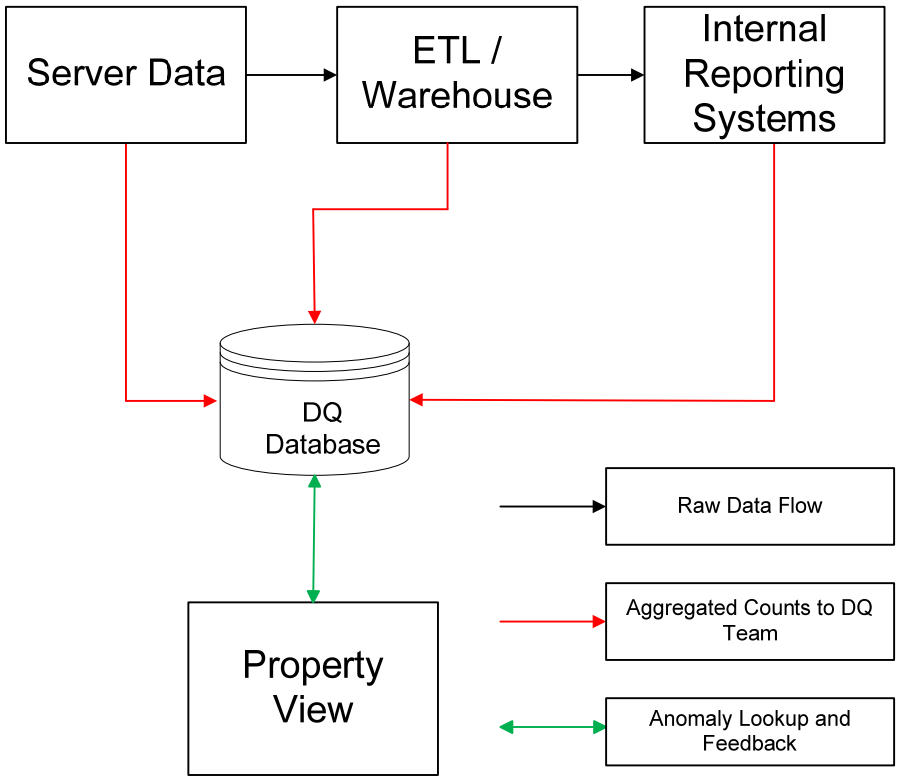
[1.4.3] User Roles and Responsibilities

The property owner's role is to monitor and explain statistical anomalies according to page views, clicks, time spent, etc for properties such as Y! Finance, Y! Brazil Mail, and Y! Taiwain Sports across Yahoo!.

The application will help the property owner explain statistics. The application will provide a view of the data at different points within the pipeline. This feature will help the property owner establish a possible starting point of an anomaly or issue.

The application will assist the property owner in monitoring statistical anomalies. The application will allow the user to view the current state of open data quality issues. This feature will explain any open data quality issues and alerts for their selected property(ies). The property owner will also have the ability to write feedback pertaining to a particular anomaly. This feedback of anomalies is critical to keep knowledge between property teams and data quality teams consistent.

[1.4.4] Interactions with Other Systems



The Property View application will interact with the data quality database to read and write information pertaining to anomalies.

[1.4.5] Replacement of Legacy Systems

The application will replace the legacy system. Currently, the property owner must interface with a website to view statistical anomaly information. There will be no bootstrapping required when switching production from the legacy system to the Property View application. The Property View application simply runs on top of the end-user's desktop.

[1.4.6] Operating Environment

This application shall use Yahoo! Widgets 4.5 which will allow for portablility between Mac and Windows clients. The Yahoo widget framework uses Javascript and XML.

[1.4.7] Design and Implementation Constraints

The developer's system will require the Konfabulator SDK from the Yahoo! Widgets website as well as Yahoo! Widgets 4.5 to be installed locally. The developer will also need Oracle 10 installed locally.

[1.4.8] Assumptions and Dependencies

Yahoo Widgets 4.5 will be installed on the end-user machine.

[2] System Features

[2.1] Extract and Display Aggregation Statistics

The widget should extract various statistical data from different data quality databases.

The information should be displayed in a text or graphical format.

[2.2] Anomalies

[2.2.1] Extract and Display

The widget should extract, bundle, and display open data anomalies.

The anomaly data extracted should only be anomaly data requested by the user.

[2.2.2] Anomaly State

The widget should allow the user to sign-off (or close) currently opened data anomalies.

The user should have the option to stop receiving anomaly alerts until a date specified.

The user should have the ability to create bugs associated with anomalies.

[2.3] Bug Information

[2.3.1] Extract and Display

The widget should be able to extract and display bug information.

The user should have the ability to enter a bug ID number

[2.4] Saving Data Changes

Any changes made to bug or anomaly data by the user in the widget should save the appropriate changes to the database.

[3] Use Cases

[3.1] Create Bug

Related Use Cases: None

Use Case Steps:

Actor Actions:

1. Create a new bug
4. Enter bug ID and additional bug comments
5. Choose to create bug

System Responses:

2. Gather information about anomaly
3. Display anomaly information
6. Send bug information to database and store it

[3.2] Re-Open Anomaly

Related Use Cases: None

Use Case Steps:

Actor Actions:	System Responses:
1. Search for anomaly	2. Display anomaly information
3. Change anomaly state from closed to open	5. Write anomaly state change to database
4. Choose to save changes	

[3.3] Re-Open and Edit Bug

Related Use Cases: None

Use Case Steps:

Actor Actions:	System Responses:
1. Search for bug by ID	2. Display bug information
3. Change bug state from closed to opened, edit comment, or ID	5. Write bug changes to database
4. Choose to save changes	

[3.4] Switch User

Related Use Cases: [3.5] User Login

Use Case Steps:

Actor Actions:	System Responses:
1. Choose to logout	2. Close all open properties
4. Follow login use case	3. Display login screen

[3.5] User Login

Related Use Cases: None

Use Case Steps:

Actor Actions:	System Responses:
1. Open widget	2. Display login screen
3. Type in username and password	4. Check database for valid username and password
6. Repeat if needed	5. Show login success/failure

[3.6] Stop Alerts

Related Use Cases: [3.7] Enable Alerts

Use Case Steps:

Actor Actions:	System Responses:
1. Choose to stop alerts 2. Enter date when user will start receiving alerts from the data source again	3. Update alert settings approximately on user's computer

[3.7] Enable Alerts

Related Use Cases: [3.6] Stop Alerts

Use Case Steps:

Actor Actions:	System Responses:
1. Choose to enable alerts	2. Update alert settings appropriately on user's computer

[3.8] Refresh Data

Related Use Cases: None

Use Case Steps:

Actor Actions:	System Responses:
1. Refresh anomaly and bug data	2. Synchronize UI data with database

[3.9] Filter Anomalies and Bugs

Related Use Cases: None

Use Case Steps:

Actor Actions:	System Responses:
1. Choose filters for which anomalies and bugs to display and receive alerts for	2. Update filter settings appropriately on user's computer

[4] External Interface Requirements

[4.1] User Interfaces

At any point in time, there should be 1+ views of different data-sets displaying property statistics and other useful information to the property owner.

The data-sets view should have different types of statistics which can be represented in a textual and/or graphical way. The data included in this view should be acquired from the Oracle database which stores the data from varying areas within the Yahoo! network.

A view shall be provided that allows the most recent anomalies detected to be downloaded from the database and presented to the user via an alert.

Anomalies should be given a view where the property owner can add a bug id and comment on the anomaly.

A view shall be provided to give the property owner the option to search for a bug by the anomalies id.

[5] Other Nonfunctional Requirements

[5.1] Performance Requirements

During the download process of data and computing of graphs, a 'Processing' message will be displayed to inform the property owner the application is working on the request.

[5.2] Process Requirements

Yahoo! did not request or require any specific process requirements.

With no specific process requirements, the team will implement whichever processes seem reasonable and effective.

[5.3] Security Requirements

Each property owner must log-in using their Yahoo! user id; this user id will be passed along to the database. Based on that input, the database will return the appropriate user properties.

Besides the user login, the application will not implement other security measures. The application is a desktop application used inside the Yahoo! internal network; thus, no further security considerations will be made. It is assumed the desktop environment will be secure and the Yahoo! employees are not malicious.

[5.4] Software Quality Attributes

It is expected that the application will not crash unexpectedly.

If any type of error is encountered which renders the program useless, a warning should be displayed to the user with a description and possible cause on exit.

The widget should have the flexibility to switch the data sources in which the information is extracted from.

[5.5] Troubleshooting

Yahoo! provided no requirements for troubleshooting.