Email Security Scanner

Design Document

Dev. Team: Chris Deck | Tom Miller | Chris Porch | Steve Leonetti

Product Owner: Dan Smith

Scrum Master: Michael Bayruns

Document Written By: Chris Deck, Michael Bayruns

Software Engineering - Eagles Design Document

Version: 1

Table of Contents

1 Purpose

This document presents the Solution Development Lifecycle (SDLC) Design / Installation Information for a Project affecting the ASRC Email Scanner.

2 Scope

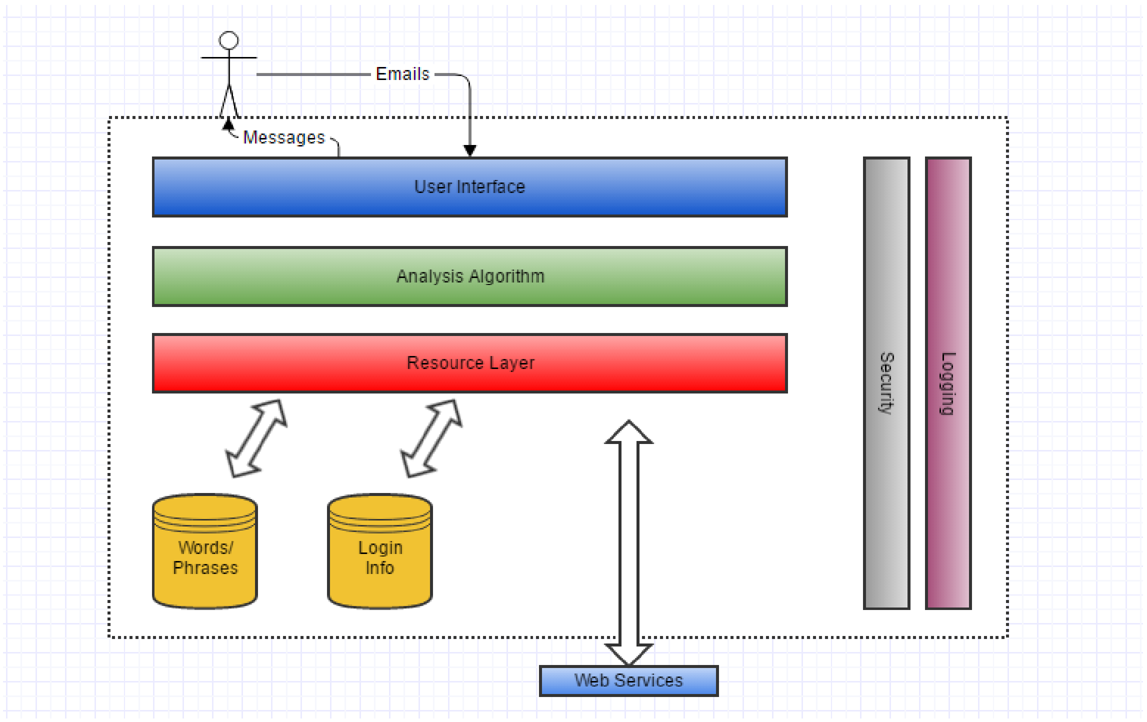
The scope of this document includes the code that the development team has written, and the structure of the project as it is today. Some Architecture in the large diagrams may change as the product progresses.

2.1 Exclusions, Assumptions, and Limitations

This Document does not cover any code written by third-parties. These libraries are assumed to be documented by their own developers and the team is just using their services. These libraries include java libraries, Apache Lucene, WordNet, etc. These libraries will be explained to show how they work, and being implemented.

3 Solution Design Overview

The Security Email Scanner has one main goal: to determine if an E-Mail is confidential, and if so, confirm that the user wishes to send confidential information. It is not a mechanism to *block* e-mails from getting out, more of a measure to assume responsibility. Below are some Architecture in the large views of the system.



4 Technical Implementation

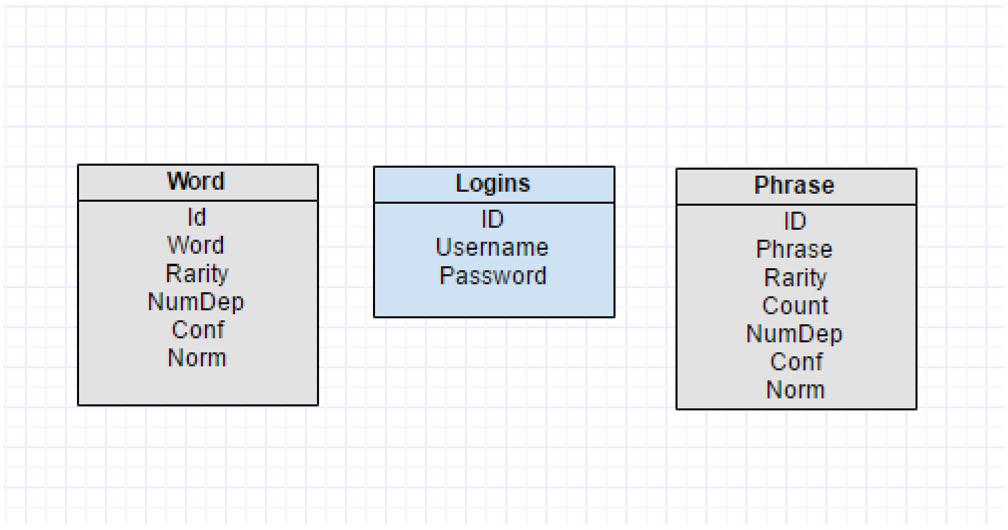
The current implementation of our program features four main GUIs. The Login GUI requires you input a username and password to access all features of the program. The password entered is hashed and compared to the ones stored in the database using B-crypt. The scanner GUI takes input of words or phrases and returns a score based on the level of confidential material entered into it. This is accomplished through parsing the input, hashing each word, and then comparing the hashed values with the ones in the database. The Database GUI can be used to add additional terms to the database. Finally the training GUI can be used used to train the program and modifies the values of each confidential word stored in the database. This training GUI can be used to load in several sample emails and score them individually to modify the values stored in the database.

4.1 Software Requirements

All dependencies should be pulled by Maven when you first run anything. *Code requires Java 1.8.* If you package the project in the command line using the `mvn package command,` it should generate a sweng-eagles-1.0.jar file in the target directory that should open the Email text checking window. As of right now, that is how the program is run. The eventual design is to implement it server-side, or as a Microsoft Plug-in as previously stated.

4.2 Middleware Hosting

A SQL database is needed for this application. The database settings should be configured in the Graphical User Interface of the application. The databases tables and relationships are defined in the figure below.



5 Security Design

The program uses SHA-512 & B-Crypt hashing algorithms to one-way hash passwords, and confidential words & phrases. There is currently a Hashing-Whitepaper up on our GitHub page for more information. The link to the GitHub is in the Appendix.

6 Solution Design

The following diagrams show what our system does, and how it works. There will be a small explanation with each diagram for further explanation.

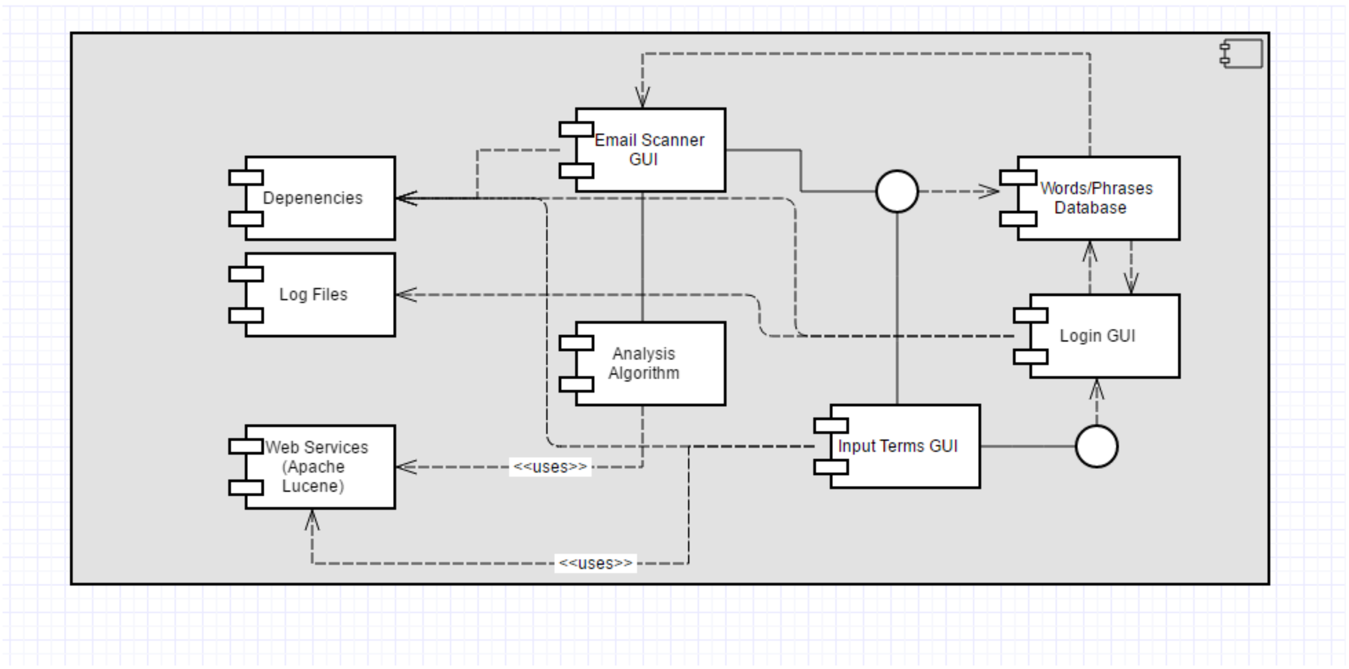


Figure 6.1 - Component Diagram

Apache Lucene is a web service the program uses to stem words and phrases down to the root word(s). It takes away unnecessary words like "the" & "a" and can be configured to take off plurals, which we have included in our program. Below is an object diagram to show an example of how it stems.

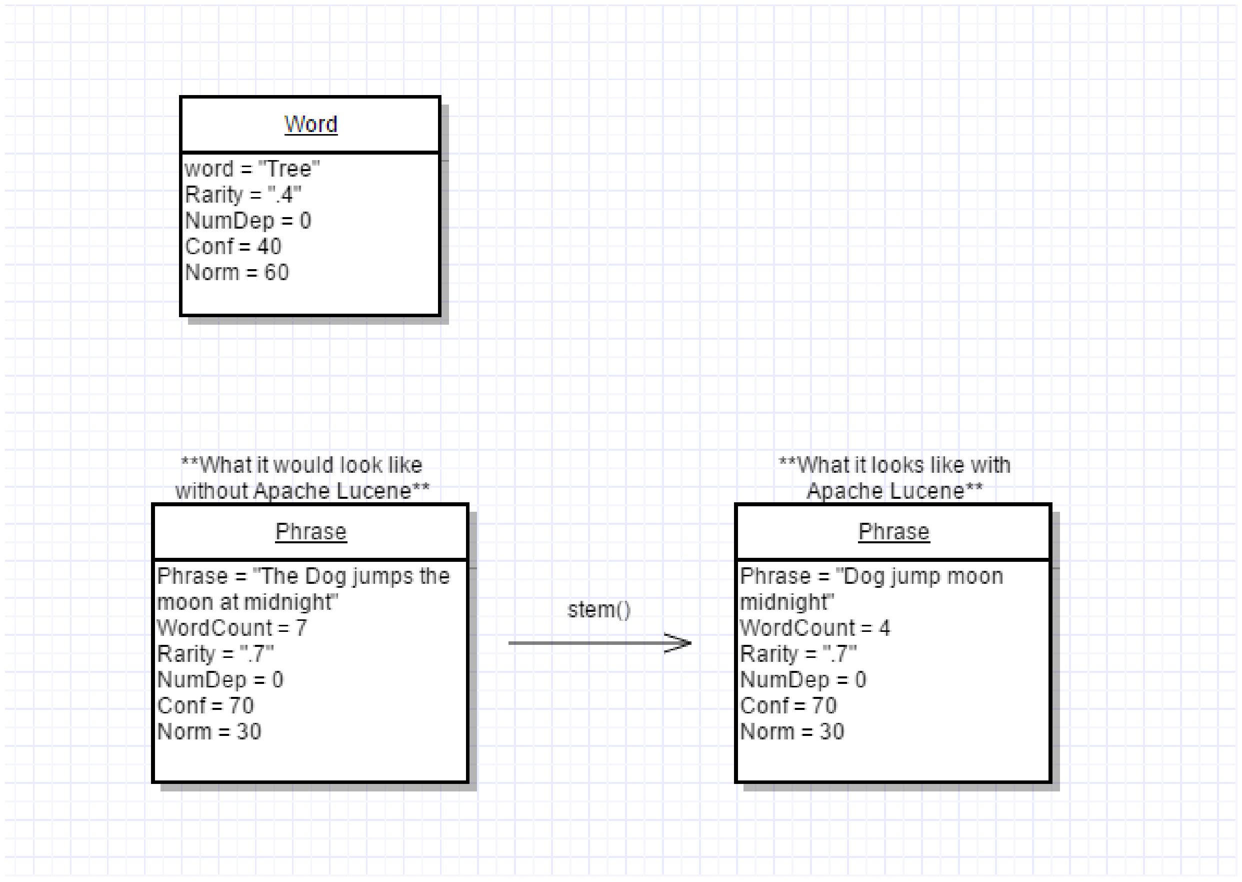


Figure 6.4 - Object Diagram

This diagram accurately represents how the words and phrases are stored and what values they have.

More information on the project can be found on our GitHub repository, including the ReadMe document, the code, and the Hashing Whitepaper. As previously stated, the link to the repository is in the Appendix.

7 Roles and Responsibilities

The *IT Solution Delivery Lifecycle* Procedure, describes the Roles and Responsibilities for developing, verifying, and implementing a Solution. There are no additional roles specific to this document.

8 Terms and Definitions

The IT Glossary of Terms maintains the common terms in this document. There are no additional terms and definitions to this document.

9 Revision History

Update this table each time this document is revised. Where possible, include a Change Number or Project related to the document change. Entries should provide the reader with only an indication of what changed. Include section where a change took place. Add rows as necessary.

Version	**Version Date**	**Revisions**
1.0	11/20/16	Initial Revision
1.1	12/15/16	Final Revision – changed document to correctly reflect the program

Appendix A

GitHub Repo - https://github.com/cbporch/sweng-eagles

Development View - https://www.gliffy.com/go/share/s3a3e8w1fyasifvh4ifq

`