Software Design Document
for
Machine Learning Based Spam Detection

Version 2.0 approved

Prepared by
Benny Berba, Syam Jason Bonela,
Leonard Garcia, Saiyang Liu,
Mary Semerdjian

California State University, Los Angeles

September 12, 2022

Revision History

Name	Date	Reason For Changes	Version
Mary Semerdjian	09/12/2022	Initial Draft	1.0
Mary Semerdjian	12/12/2022	Final Draft, editing and revising	2.0

1. Introduction

1.1 Purpose

This Software Design Document (SDD) aims to outline and explain the functions of Machine Learning Spam Detection software (Version 2.0). This document will cover all aspects-concepts of the software, which include: an overall description of the ML Spam Detection software, and it serves to expand the knowledge and understanding of our project's functionalities, architectural strategies, system architecture, user interface, and libraries used in our software.

1.2 Document Conventions

The document contains headings, subheadings, and body text in Times New Roman.
The headings are font size 20pt and bold, subheadings are 14pt and bold, and the body text is 12pt to make the document's layout more readable for the reader. Bullet points are used throughout the document to contour specifications so that the person reading the document can read through it quickly and easily.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document includes developers, machine learning researchers, users, project managers, marketing staff, and testers.

This SDD is organized as follows:

- Developers and ML researchers-users: Interested in reading the entirety of the document and following each section in sequential order.
- Marketing staff and testers: Suggested that they might be interested in high-level descriptions of the project, such as the Introduction and Overall Description.
- Project manager: Helps understand the requirements and steer the development of the web application.
-

1.4 System Overview

The project's goal is to create Machine Learning Based Spam Detection that will be able to identify emails or text messages as spam or ham. The project aimed to understand complex Machine Learning Model Algorithms, as well as being able to train and test a big dataset. Lastly, creating a Graphical User Interface for Spam-Identifier.

2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

2.1 Assumptions and Dependencies

- Related software or hardware
    - Certain GUI issues and limitations
        - Limited amount of training data and testing data, only able to identify old fashion spam messages.
        - Limited features.
- Operating systems
    - Windows
    - Linux
    - MacOS
- End-user characteristics
    - Understanding what the user wants to do when using the GUI
        - Make it easy for the user to understand the functionalities of the GUI
        - Do not complicate the installation, configuration, features, issues and limitations as this will lead to more confusion for the user.
- Possible and/or probable changes in functionality

2.2 General Constraints

- Hardware or software environment
    - Application software that has access to the internet or Wi-Fi
        - Runs on
            - Google Colab
            - Jupiter Notebook
            - Python
- End-user environment
    - Up-to-date web browsers
        - Mozilla Firefox
        - Apple Safari
        - Internet Explorer

        - Google Chrome
        - Opera
        - Microsoft Edge
- Availability or volatility of resources
    - CSV files for testing
- Standards compliance
- Interoperability requirements
- Interface/protocol requirements
- Data repository and distribution requirements
    - Cost of using:
        - Google Colab
- Security requirements (or other such regulations)
    - Anonymization of Personal Data (Emails-Text Messages)
    - Access to Google
- Memory and other capacity limitations
    - 8 GB RAM with 256 GB of space
    - 16 GB RAM with 500 GB or 1-2 TB of space
    - 32 GB RAM with 500GB or 1-2 TB of space
    - 64 GB RAM with 500GB or 1-2 TB of space
- Performance requirements
    - 2.2 GHz and above
    - 2.6 GHz and above
    - 3.2 GHz and above
    - 3.6 GHz and above
    - 4.0 GHz and above
- Hardware requirements
    - Windows/Mac/Linux users with access to Google Collab and Jupiter Notebook
    - Linux Machine
    - NVIDIA GPU
    - AMD GPU
    - M1 GPU

    - M2 GPU
    - Intel CPU
    - AMD CPU
    - M1 CPU
    - M2 CPU
- Network communications
    - The user must have a stable internet connection
- Verification and validation requirements (testing)
- Other means of addressing quality goals
- Other requirements described in the requirements specification

2.3 Goals and Guidelines

One of the goals of the project was that the working version of the software has a mandatory delivery date of December 7, 2022, which is the date of the final presentation. Another goal/guideline of the project was that the software is light on the computer, meaning that it is a relatively small sized program and can run on any modern computer. We also wanted our software product to be intuitive and easy for any consumer to use, whether or not they have a technical background.

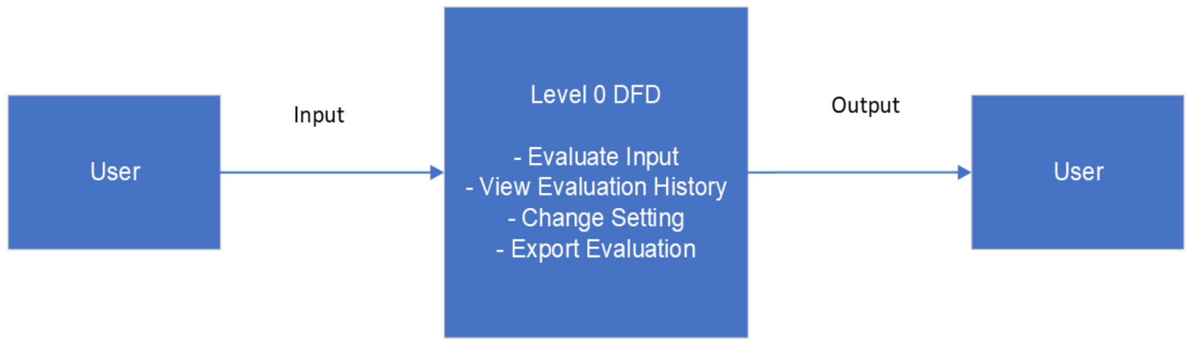
2.4 Development Methods

The method we used for this software design was the Waterfall Model. We followed a variation of the steps: requirements, design, implementation, verification, and maintenance. In the requirements stage, we did requirements gathering and analysis to see what our goals were for the project, and what further steps needed to be taken. Next, we designed a mockup of our GUI, as well as conducted exploratory data analysis of the datasets. At the implementation stage, we completed the training of the machine learning models, as well as created a working GUI based on the mockup. In the verification stage, we conducted testing of the machine learning models, choosing the best performing model to be used in the GUI. Lastly, we routinely tested the published version of our GUI with spam messages found online to see if the model is outputting the correct results, or if further updates are needed.

3. Architectural Strategies

The software application was done and tested using the IDLE (Integrated Development and Learning Environment) of Python. This application was coded in Python, and Python was the primary language used. The app did not use databases. Note that some of the libraries used were Excel for previewing the data and the User Interface library that was used was tkinter-custom tkinter. Aside from the UI and Excel, many different Machine Learning libraries were used, such as the scikit-learn, Natural Language Toolkit, Pandas, NumPy, and Matplotlib. In addition to these Machine Learning libraries, a deep learning library was used such as Keras.

4. System Architecture



4.1 Level 0 DFD:

1. User Input

The user represents the individual that is using the application (i.e., user that uses the application).

2.Evaluate Input

The user can type in an input and evaluate it.

3.View Evaluation History

The user can view the evaluation history.

4. Change Setting

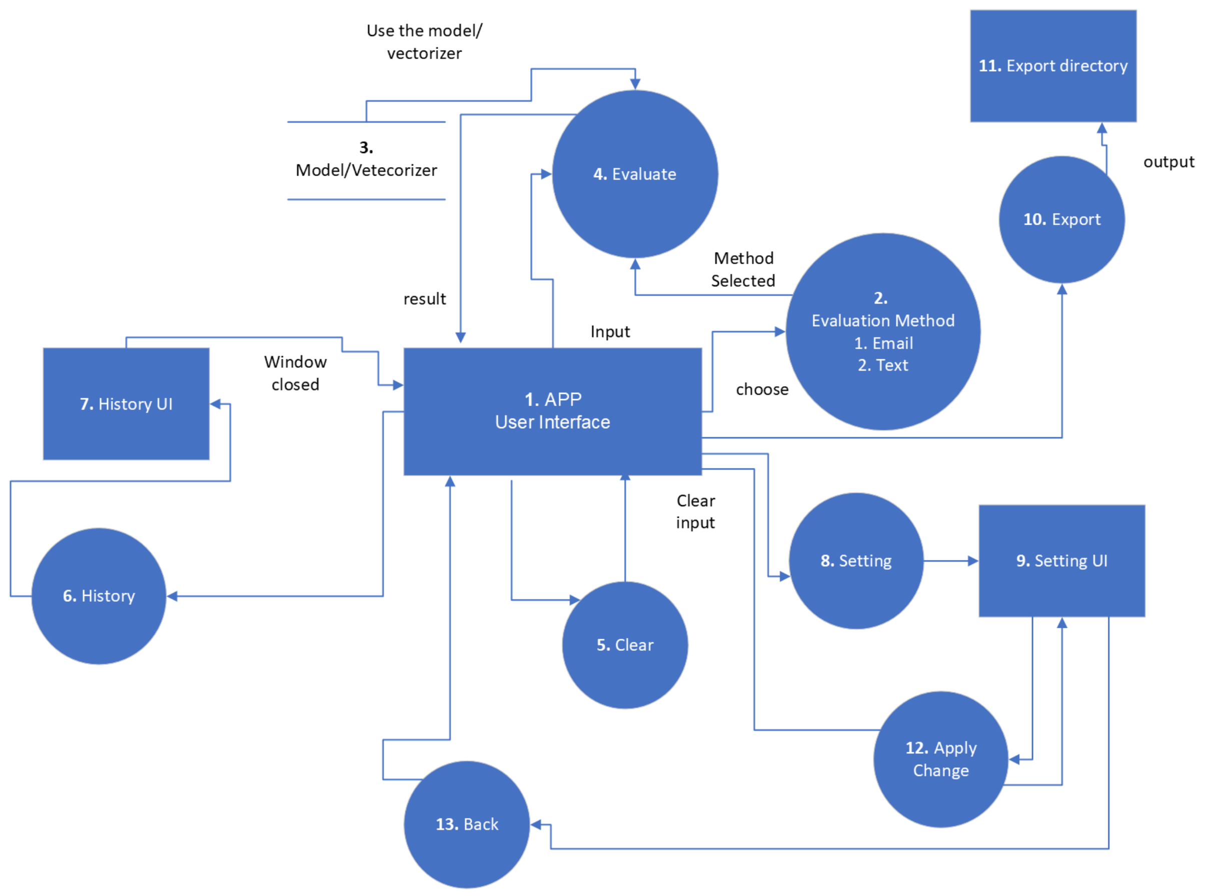
The user can change the setting of the application.

5. Export Evaluation

The user can export all the evaluation history.

6. User Output

The user outputs the results.



4.2 Level 1 DFD:

1.App/User Interface

The application itself, by default, shows the main user interface page.

2. Evaluation Method

Radio buttons allow the user to choose based on which type of input they want to use as an evaluation tool.

3. Model/Vectorizer

The exported machine model/vectorizer files that were loaded to the application for input evaluation.

4. Evaluate

Evaluation method takes the input and the corresponding evaluation method (3) to evaluate the input. Once the process is done, return the result to the user.

5. Clear

Clear the input that the user typed.

6. History

Leading the user to the History UI page (7).

7. History UI

History UI in a new pop-up window that shows all the prior input evaluation results including input Message, result, types of evaluation. When closed, return to the main page.

8. Setting

Leading the user to the setting page (9).

9. Setting UI

Setting page that allows the user to adjust different settings including window resolution, export directory.

10. Export

Exporting all the prior evaluation that can be viewed through the History UI (7) at the target location (11)

11. Export Directory

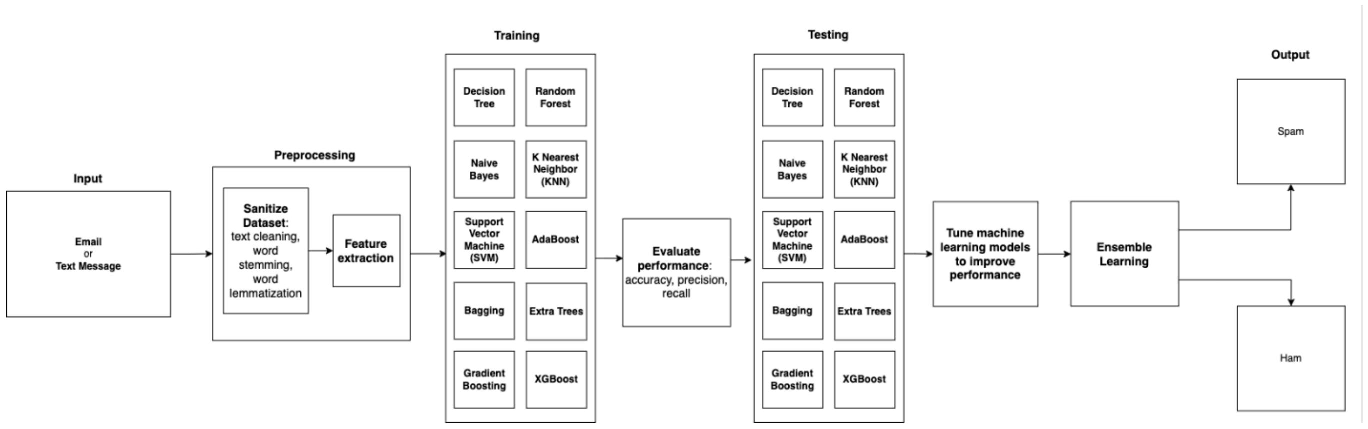
The file export location after running Export (10), the file will be in a .csv extension. Export Directory can be changed in the Setting UI (9).

12. Apply Change

After selecting new setting for the application, the user can apply those changes and it will affect the whole application (Main Interface (1) and Setting Interface (9)) except History UI (7). User will continue to stay at the Setting Interface after the change applied.

13. Back

Leading the user back to the Main Interface (1). If any unsaved change exists, discard it and return to the Main Interface (1).



4.3 Level 2 DFD:

1. Input

Input takes input as Email or Text Messages.

2. Preprocessing

Preprocesses the data. Note that there are two preprocessing methods: sanitizing the dataset and feature extraction.

3. Training

The training trains the model. There are a total of ten Machine Learning classifiers trained.

4. Evaluate

Evaluate the performance of accuracy, precision, and recall.

5. Testing

The testing tests the model. There are a total of ten Machine Learning classifiers tested.

6. Tune machine learning models

The tune machine learning model tunes the machine learning models to improve performance.

7. Ensemble Learning

Ensemble Learning is used for better prediction results.

8. Output

Output the Spam or Ham due to the machine learning models used.

5. Policies and Tactics

5.1 Choice of which specific products used

IDE

- Anaconda
- Google Colab
- PyCharm

Software

- Jupiter Notebook
- Google Colab
- Python
- GitHub

Programming Language

- Python

Import/Export Files

- CSV files
- IPYNB files

5.2 Plans for ensuring requirements traceability

The group will be using Google Colab since the program runs on a faster GPU. Whenever a group member makes an "update" to the IPYNB file, other members in the group will have automatic access to the file to view.

In addition to Google Colab, the group also used GitHub for making changes to the GUI.

5.3 Plans for testing the software

The plan for testing the software is to aim for professionalism and to have the users-executives read and gather information from the dashboards.

5.3.1 Step-Plan for Testing

- i. Test for Basic Functionality
    1. User Interface Elements
        - a. Buttons
        - b. Text fields
        - c. API Calls
    2. Not testing all edge cases
- ii. Code Review
    1. Review Code in pairs
        - a. Go Line by Line
        - b. Explain what is happening

        c. Fix any errors
iii. Static Code Analysis
iv. Unit Testing
    1. Create test cases for each component
        a. Test Basic Functions
        b. Test Edge Cases
    2. Review any failed tests
        a. Repeat Process
v. User Testing
    1. Allow users to use application
        a. Report any bugs or issues found
        b. Resolve any issues

6. Detailed System Design

6.1 Machine Learning Models

6.1.1 Responsibilities

Uses Machine Learning models to test and train Spam Texts/Emails. Specifically focused on Machine Learning algorithms/techniques/methods such as: pre-processing, individual model training, and ensemble learning.

6.1.2 Constraints

6.1.2.1 Pre-processing

- In data pre-processing, first our datasets undergo cleaning for use in the ensuing modules.
- Any invalid entries, such as those with null values, must be removed.
- The dataset must be reduced and all unnecessary characters, punctuation, numeric characters, etc. must be removed.
- Unnecessary words must also be removed.
- This includes stop words, commonly used words that do not add much meaning to a sentence.
- Count Vectorization converts a given set of strings into a frequency representation.
- Performed by the CountVectorizer function of the sklearn library.
- Term Frequency - Inverse Document Frequency (TF-IDF) provides a numerical representation of how important a word is for statistical analysis based on word frequency.
- Performed by the TfidfVectorizer function of the sklearn library.

6.1.2.2 Individual Model Training

- Each of the machine learning models is trained on the pre-processed training data.

6.1.2.3 Ensemble Learning

- Voting Ensemble combines the predictions from multiple models.
- It combines predictions in two ways: hard voting and soft voting.

6.1.3 Composition

6.1.3.1 Pre-processing

- The sanitized data is entered into feature extraction algorithms for use in training.
- These results are then sent to various machine learning models.
- Word stemming and lemmatization is performed.
- Word lemmatization groups together different forms of a word for analysis as a single word.
- Term Frequency - Inverse Document Frequency (TF-IDF) identifies words that are less important for analysis, further reducing the dataset.
- Once the dataset has been reduced and feature extraction has been performed, it is split into training and testing datasets.

6.1.3.2 Individual Model Training

- Once trained, each model will perform binary classification on the testing data, predicting whether each entry is “spam” or “ham,” and be evaluated for accuracy.
- After the initial training, parameter tuning will be performed on the various models to improve accuracy.

6.1.3.3 Ensemble Learning

- Hard voting sums up the predictions for each class label and chooses the label with the most votes.

6.1.4 Uses/Interactions

Used to create Machine Learning models of Spam Based Detection.

6.1.4.1 Pre-processing

- The datasets and machine learning models are used-tested to identify those best able to achieve high performance in classification.
- The diverse set of high performing models are used in an ensemble.
- Once the dataset has been reduced to only these most important words, feature extraction can be performed.
- This feature extraction will convert our dataset consisting of strings into numeric characters.

- The training data is then fed into the machine learning models for training which is then used in the pre-processing stage.

6.1.4.2 Individual Model Training

- Once the parameters that provide the best performance have been determined, the best performing, diverse models will be used to perform ensemble learning.

6.1.4.3 Ensemble Learning

- Soft voting sums up the predicted probabilities for each class label and chooses the label with the probability.

6.1.5 Resources

6.1.5.1 Data

6.1.5.1.1 Text Message

6.1.5.1.2 Email

6.1.5.2 Software

6.1.5.2.1 Operating System

- Windows
- Mac
- Linux

6.1.5.2.2 Python

- Jupyter Notebook
- Google Colab

6.1.6 Interface/Exports

6.1.6.1 File Output

Machine Learning Model

6.1.6.2 Visual Output

Visually see that the Machine Learning model works in Google Colab or Jupyter Notebook

7. Detailed Lower-level Component Design

Does not apply.

8. Database Design

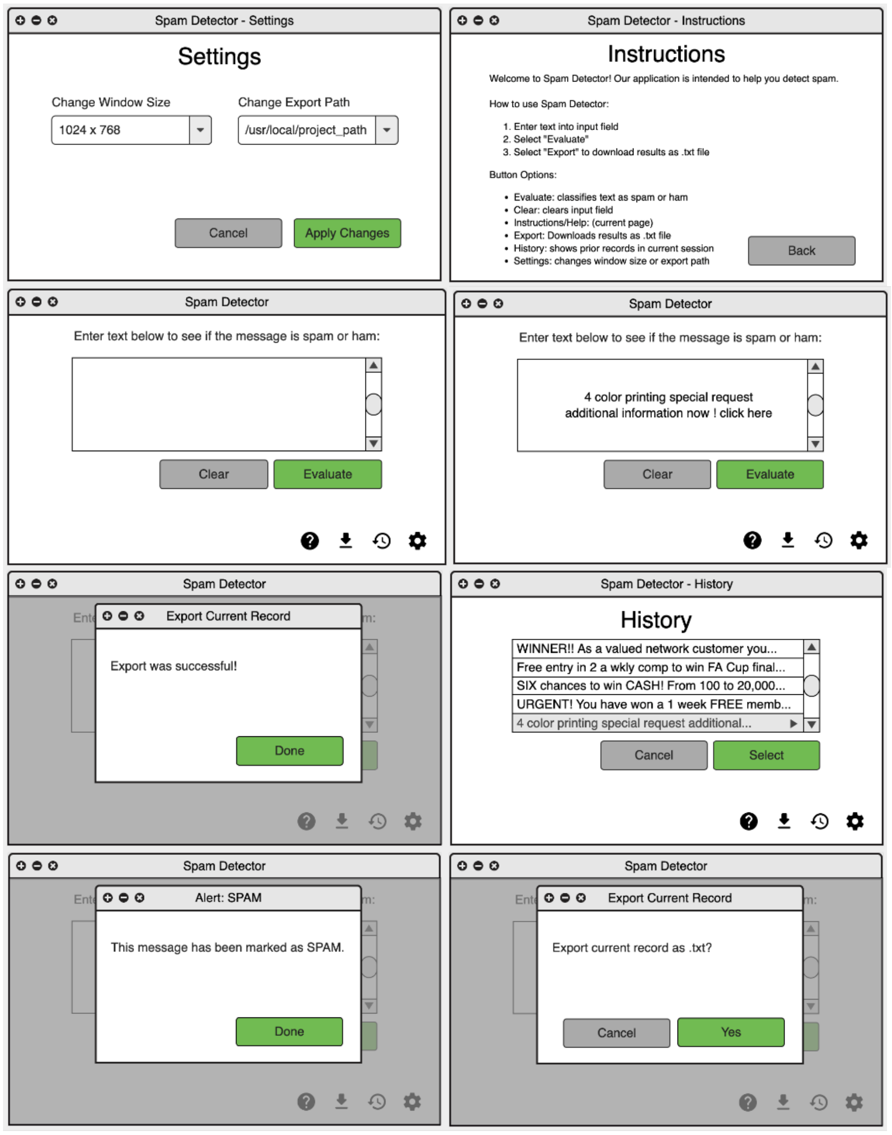
No Database was used in this project; therefore, this does not apply.

9. User Interface

9.1 Overview of User Interface

The user will be able to access the Instructions screen to understand how to use the User Interface. If the user wants to change the window size or the export path, they can access the Settings screen to make those changes. The Main screen includes a text box that allows the user to input the text they want to analyze, as well as several icons representing the different features of the application: Instructions icon, Export icon, History icon, Settings icon. The Export icon allows the user to export the current record as a .txt file, while the History icon leads to the History screen and allows the user to access previously analyzed messages.

9.2 UI Mockup



Change Window Size: 1024 x 768
Change Export Path: /usr/local/project_path

Welcome to Spam Detector! Our application is intended to help you detect spam.

How to use Spam Detector:

1. Enter text into input field
2. Select "Evaluate"
3. Select "Export" to download results as .txt file

Button Options:

- Evaluate: classifies text as spam or ham
- Clear: clears input field
- Instructions/Help: (current page)
- Export: Downloads results as .txt file
- History: shows prior records in current session
- Settings: changes window size or export path

Enter text below to see if the message is spam or ham:

4 color printing special request
additional information now ! click here

Export was successful!

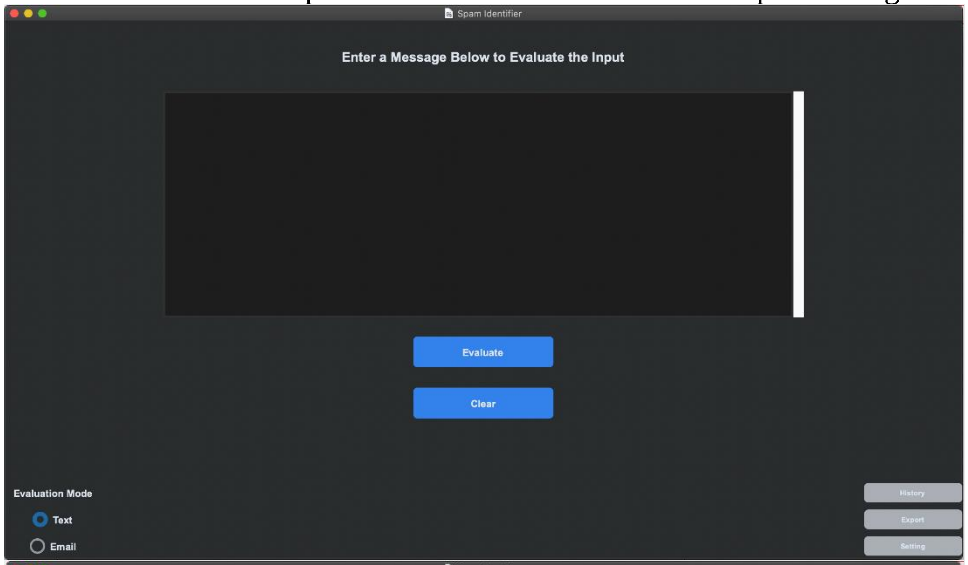
WINNER!! As a valued network customer you...
Free entry in 2 a wkly comp to win FA Cup final...
SIX chances to win CASH! From 100 to 20,000...
URGENT! You have won a 1 week FREE memb...
4 color printing special request additional...

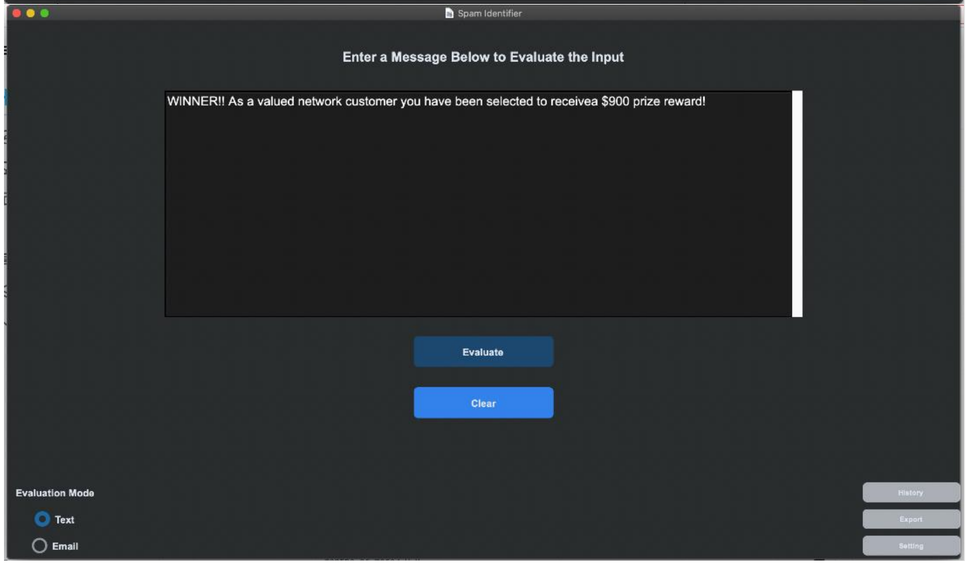
This message has been marked as SPAM.

Export current record as .txt?

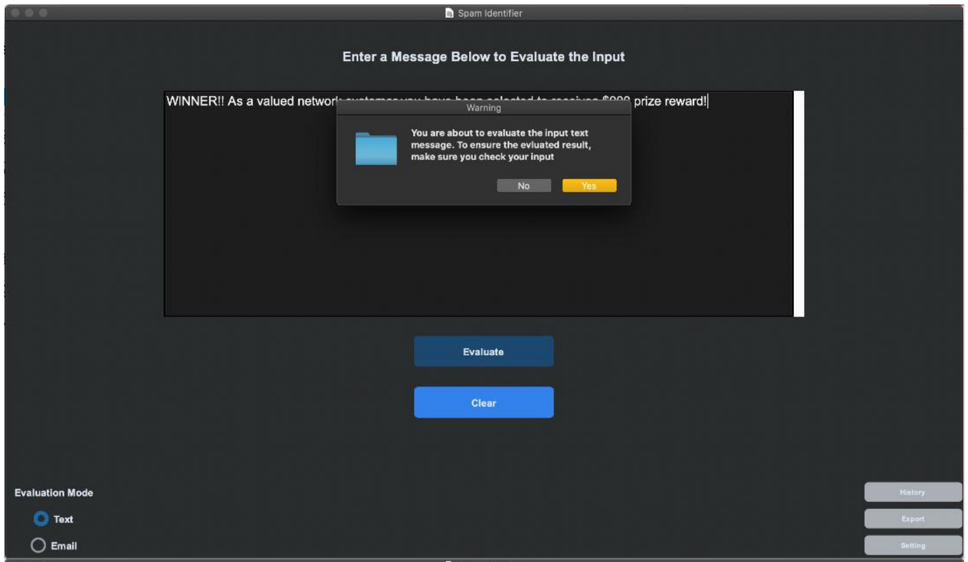
9.3 Current Version UI Screens

Below are the current Spam Identifier UI screens on a computer using MacOS on dark mode.



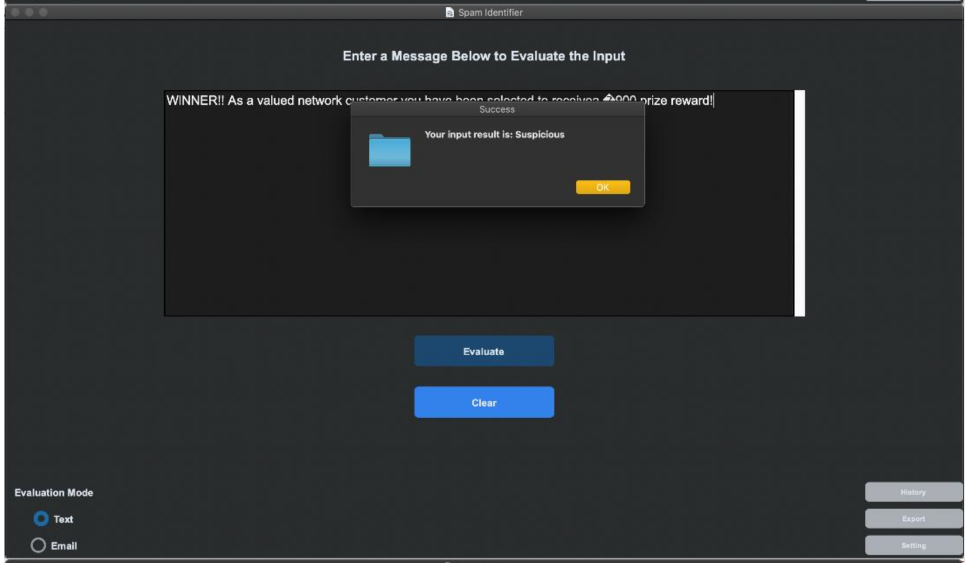


WINNER!! As a valued network customer you have been selected to receivea $900 prize reward!



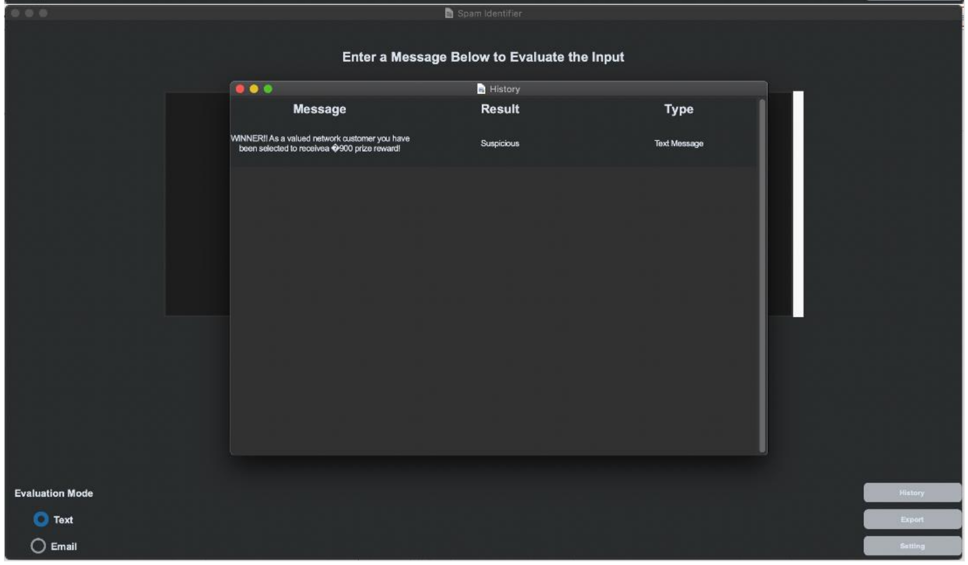
WINNER!! As a valued network customer you have been selected to receivea $900 prize reward!

You are about to evaluate the input text message. To ensure the evluated result, make sure you check your input

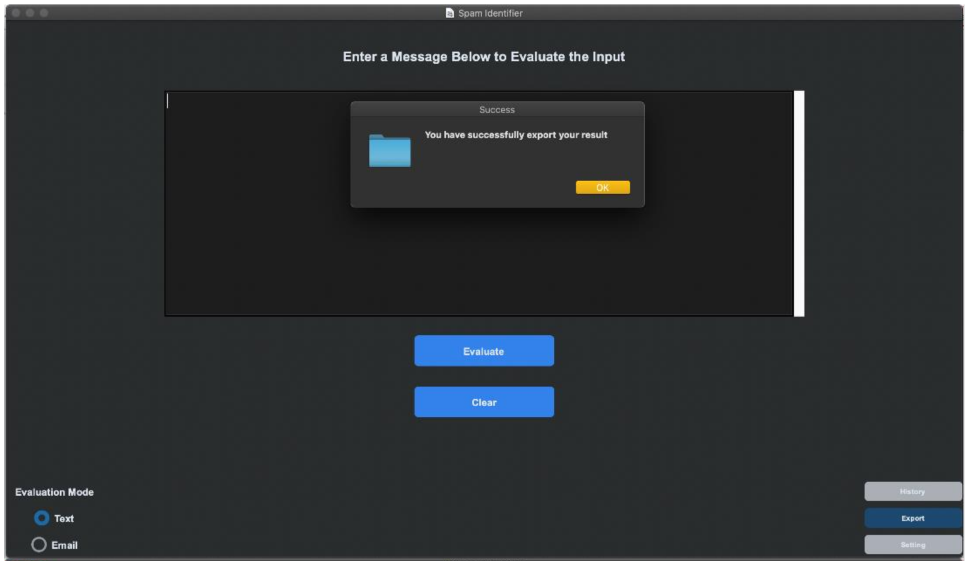


WINNER!! As a valued network customer you have been selected to receivea �900 prize reward!

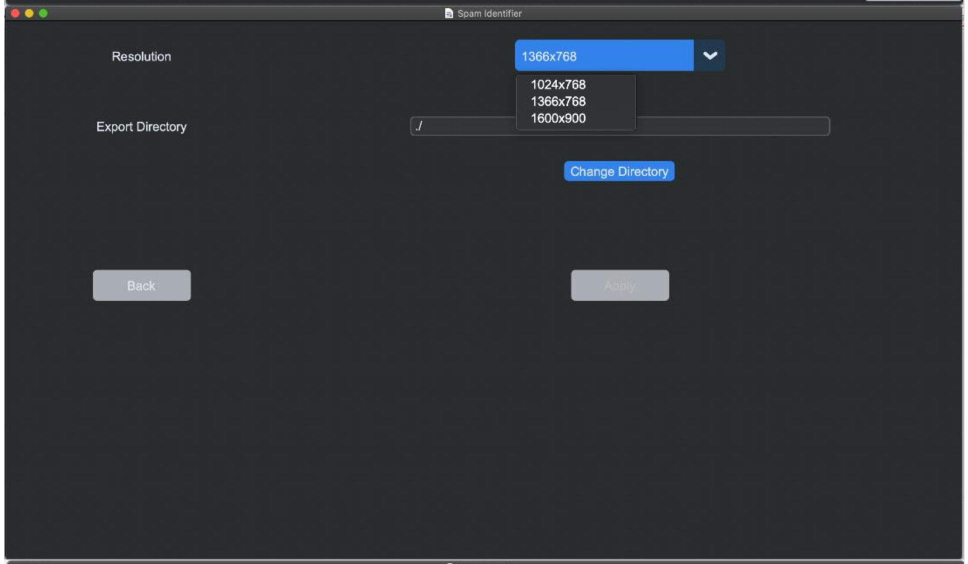
Your input result is: Suspicious

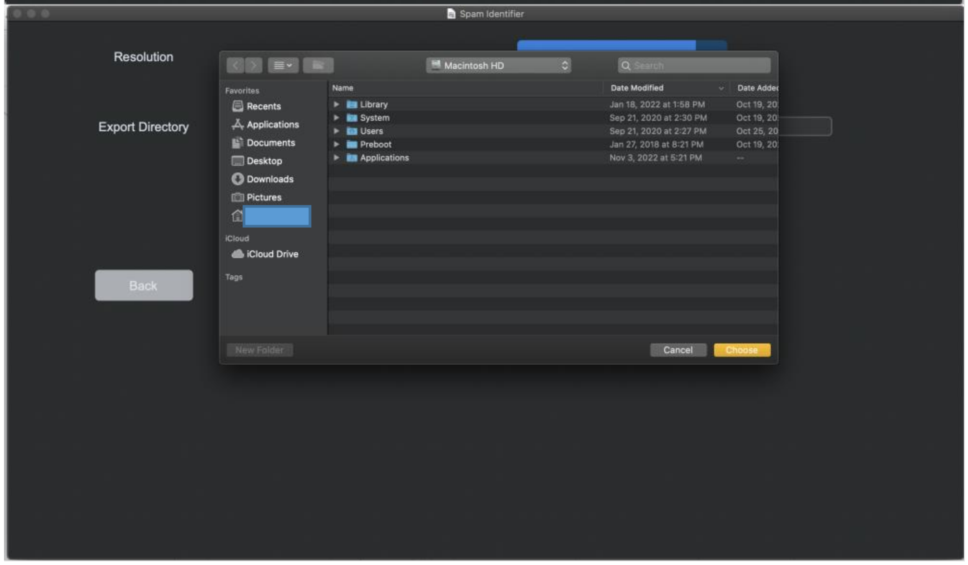


Message	Result	Type
WINNER!! As a valued network customer you have been selected to receivea �900 prize reward!	Suspicious	Text Message



You have successfully export your result

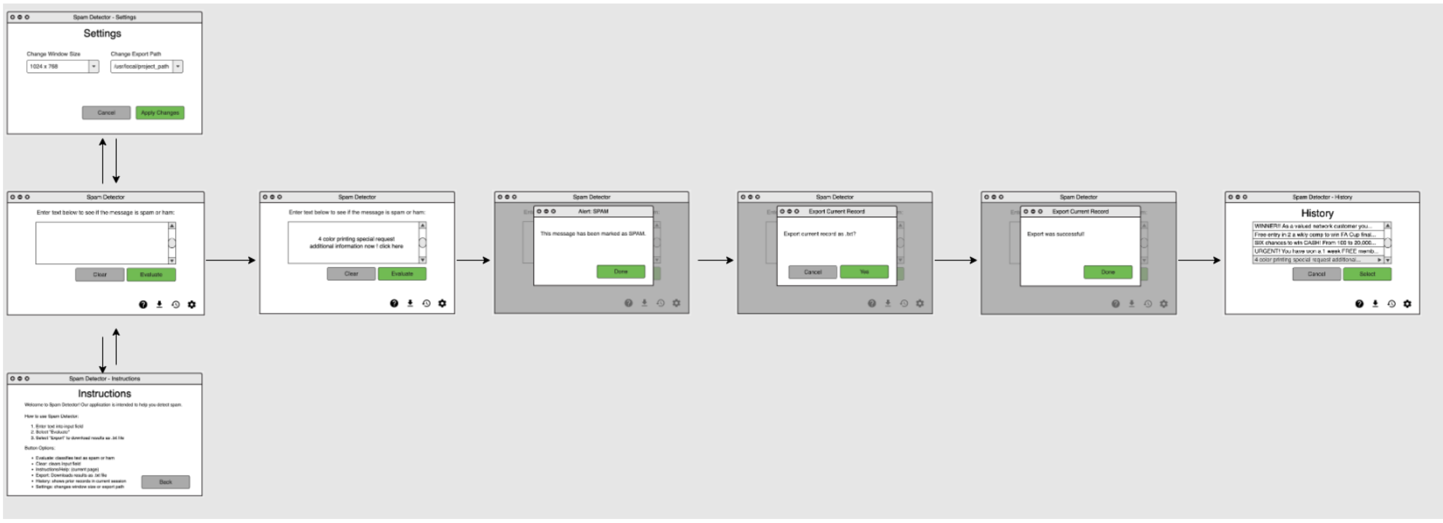




Name	Date Modified	Date Added
Library	Jan 18, 2022 at 1:58 PM	Oct 19, 20
System	Sep 21, 2020 at 2:30 PM	Oct 19, 20
Users	Sep 21, 2020 at 2:27 PM	Oct 25, 20
Preboot	Jan 27, 2018 at 8:21 PM	Oct 19, 20
Applications	Nov 3, 2022 at 5:21 PM	--

9.4 User Interface Flow Model

Below is a discussion of screen objects and actions associated with those objects. It should include a flow diagram of the navigation between different pages.



Enter text below to see if the message is spam or ham:

4 color printing special request additional information now ! click here

This message has been marked as SPAM.

Export current record as .txt?

Export was successful!

WINNER!! As a valued network customer you...
Free entry in 2 a wkly comp to win FA Cup final...
SIX chances to win CASH! From 100 to 20,000...
URGENT! You have won a 1 week FREE memb...
4 color printing special request additional...

Welcome to Spam Detector! Our application is intended to help you detect spam.

How to use Spam Detector:

1. Enter text into input field
2. Select "Evaluate"
3. Select "Export" to download results as .txt file

Button Options:

- Evaluate: classifies text as spam or ham
- Clear: clears input field
- Instructions/Help: (current page)
- Export: Downloads results as .txt file
- History: shows prior records in current session
- Settings: changes window size or export path

1. Main Screen
2. Display the Main Screen for User Interface
    a. Options
        i. Enter text for spam identification
        ii. Instructions
        iii. Export
        iv. History
        v. Settings
3. Results pop-up window (SPAM or HAM)
4. Back to Main Screen

10. Requirements Validation and Verification

Requirement Number	Description	Component Module / UI Element	Testing Method
4.1.1.1	The system shall show the main user interface page and allow the user to choose which type of input they want to use as an evaluation tool: • Email • Text message	Main UI page	• Run Spam Identifier software • Check that the Main UI page loads correctly • Check that the input toggle button loads and works • Check that either email or text message can be selected
4.1.1.2	The system evaluation method shall take and evaluate the input and return the result to the user. It should also clear the message, if decided to do so.	Main UI page	• Input message as text message or write a sentence • Output should use a Machine Learning Model to detect the message as spam or ham • Clear the message you have inputted
4.1.1.3	The system shall have a History UI page that will allow the user to show all prior input evaluation results, including: • Input message • Result • Types of evaluation	History UI page	• Shows the type of evaluations such as Text Message or Email • Keeps data of inputted message/s and sentence/s • Shows the results such as suspicious or normal
4.1.1.4	The system shall have a settings page that allows the user to adjust different settings including: • Window resolution	Settings page	• Checks that the resolution of the windows screen size is correct

	• Export directory		• Export your result
4.1.1.5	The system shall have an export function that allows the user to export all prior evaluation viewable through the History UI to the target location	Export page	• Checks if the export directory has been exported

11. Glossary

- Accuracy
    - Accuracy is one metric for evaluating classification models
- Anaconda
    - Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment
- Classification model
    - A type of model that distinguishes between two or more discrete classes
- Comma-Separated Values (CSV)
    - A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values
- Data pre-processing
    - A component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure
- Exploratory Data Analysis (EDA)
    - Refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations
- Emails
    - Messages distributed by electronic means from one computer user to one or more recipients via a network
- Ensemble learning
    - Process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem

- Feature extraction
    - Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set
- Google Colab
    - Colaboratory, or “Colab” for short, is a product from Google Research and allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education
- Graphical User Interface (GUI)
    - A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics)
- IPYNB
    - Notebook document file type for Jupyter Notebook
- Jupyter Notebook
    - Is the original web application for creating and sharing computational documents
- Local host
    - Localhost is a hostname that refers to the local machine currently making the request
- Machine Learning (ML):
    - A branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy
- Natural Language Processing (NLP):
    - A component of AI in the field of linguistics that deals with interpretation and manipulation of human speech or text using software. It enables the computer to understand the natural way of human communication by combining machine learning, deep learning and statistical models
- Precision
    - The quality, condition, or fact of being exact and accurate
- Pickle
    - The quality, condition, or fact of being exact and accurate
- Python
    - A high-level general-purpose programming language.
- Recall
    - Calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples

- Spam
    - Unnecessary, unwanted, or repetitive content that clogs inboxes and clutters social media feeds
- User Interface (UI)
    - The point of human-computer interaction and communication in a device.
- Word stemming
    - Process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma
- Word lemmatization
    - Grouping together of different forms of the same word

12. References

1. “Ham.” *Ham - SPAMASSASSIN - Apache Software Foundation*, https://cwiki.apache.org/confluence/display/spamassassin/Ham#:~:text=%22Ham%22%20is%20e%2Dmail,non%2Dspam%22%2C%20instead.
2. “Machine Learning Glossary | Google Developers.” *Google*, Google, https://developers.google.com/machine-learning/glossary.