Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster

Email Spam Filter using Machine Learning

Design Manual

Author: Hazel Murphy

Student ID: C00230058

Project Supervisor: James Egan

Date: Friday 11th December 2020

Table of Contents

Abstract

The purpose of this project is to create a web application that will filter spam emails by using supervised machine learning algorithms. We are giving the user the ability to view their emails in their inbox already filtered, like an email client application. The classifier will run periodically in the backend to ensure all emails are correctly classified based on the machine learning algorithms. The admin should be able to view a spam report to find out how many spam mails there are in each inbox overall and view all the registered users of the application. The spam and ham classification will never be shown to the user, this will all take place in the backend. They will just see their emails in the inbox or spam sections of their interface.

Introduction

This Design Manual document is a document that provides detailed information on how the email spam filter tool will function and what the expected behaviour of it will be. This document will provide an architecture overview, database design, detailed use cases and system sequence diagrams. The purpose of this document is to give the reader a deep understanding of how the entire system provides its functionality specified in the functional specification document. To determine if an email is spam or not spam is an extremely important aspect for all email users. For any daily email user their primary purpose is to send and receive emails which are safe and secure to view. Opening emails which are spam can cause major impacts to the user e.g., download malware on their devices, the hacker could gain access to their sensitive information. The email spam filter tool will be a user-friendly system to use. It will display an easy-to-use graphical user interface (GUI) to any user who logs in. Each user will have access to receive and view emails as often as they wish. Users will not need to worry if an email they have received in their inbox is spam and may contain for example malware. This is because the email spam filter tool will have classified this email as spam and the email will appear in the spam folder. The tool will classify the emails by using a supervised machine learning algorithm such as Naive Bayes.

This document presents relevant UML diagrams such as class diagrams and sequence diagrams. Additionally, it provides User Experience (UX) which is considered with the entire process of acquiring and integrating a web application including design, usability, and functions. It also will provide pseudo-code of implementing the spam filtering system in the spam catcher web application.

Project plan

A project plan also known as project management plan clearly outlines the project goals and objectives, specifies tasks and how goals will be achieved, identifies what resources will be needed and associated budgets and timelines for completion.

Sprint #	Plan	Due Date	Deliverable
0	Complete Necessary Research	13/11/2020	Research Manual
1	Spam classification machine learning model	20/11/2020	Discovery for spam classification machine learning model
2	Complete Necessary Research on how the project will function	27/11/2020	Functional Specification
3	Implement Naïve Bayes and SVM model	4/12/2020	Build python function for Naïve Bayes and SVM model
4	Complete Necessary Research on the design of the project	11/12/2020	Design Manual
5	Implement Random Forest and Logistic Regression model	16/12/2020	Build python function for Random Forest and Logistic model
6	Presentation 1	18/12/2020	Present my project to my supervisor and receive feedback
7	Implement Word embedding	22/12/2020	Build python function for Word embedding
BREAK	**Christmas Holiday**		
8	Plan tests	08/01/2021	Plan tests to create confusion matrix, type I error, type II error, precision and recall to determine which is best to use
9	**Presentation 2**	**15/01/2021**	**Key date: by this date the project should be ready for a live demonstration.**

10	Research flask application	20/01/2021	Discovery for flask application
11	Design flask application	05/02/2021	A front end for the website that is easy to use and looks good
12	Build initial flask application	19/02/2021	Implement the web app with all the required features such as secure login page
13	Research hMailServer	23/02/2021	Discovery for hMailServer setup
14	Implement hMailServer	05/03/2021	Build the hMailServer
15	Train models	06/03/2021	Get datasets and train the 4 models, getting all scores for analysis
16	Implement cron-job	12/03/2021	Embed the python function as periodic cron-job into flask application
17	Implement models into cron-job	13/03/2021	Embed the machine learning models into the cron-job
18	Test the development of all functionality in application	26/03/2021	Iterate on Flask application until all functionality in application
19	Test spam classification	31/03/2021	Test all spam classification features
20	Test spam report	02/04/2021	Test spam report features
21	Exploratory testing phase	10/04/2021	Test full application, report any flaws found
22	Review	15/04/2021	Complete any final tasks, review final project
23	Technical Manual	19/04/2021	Complete the technical manual and submit to the supervisor

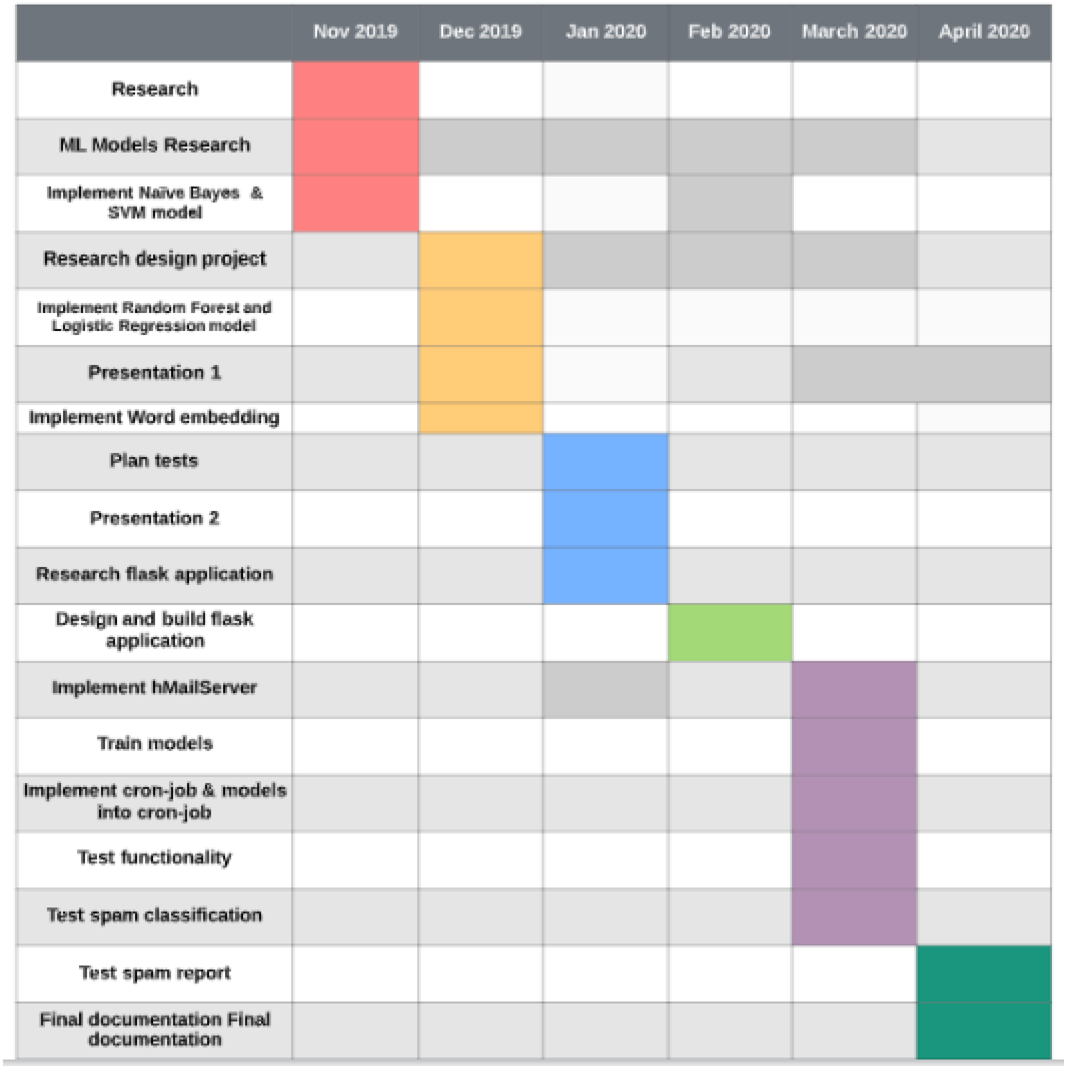
24	Final Report	19/04/2021	Complete the final report for the project and submit to my supervisor
25	Final Product	23/04/2021	Final product completed successfully
26	Research Poster or Usability	23/04/2021	Complete the research poster for the project and submit to my supervisor
27	**Project Submission**	**30/04/2021**	**The project should be reviewed. Receive feedback from friends, family, and daily email users.**

Figure 1: Project Plan

(4 Tips for an Effective Project Management Plan, 2017)

Email Spam Filter Gantt Chart

An estimated Gantt chart is provided to outline the dates for the deliverables of my project.



	Nov 2019	Dec 2019	Jan 2020	Feb 2020	March 2020	April 2020
Research	■					
ML Models Research	■					
Implement Naïve Bayes & SVM model	■					
Research design project		■				
Implement Random Forest and Logistic Regression model		■				
Presentation 1		■				
Implement Word embedding		■				
Plan tests			■			
Presentation 2			■			
Research flask application			■			
Design and build flask application				■		
Implement hMailServer					■	
Train models					■	
Implement cron-job & models into cron-job					■	
Test functionality					■	
Test spam classification					■	
Test spam report						■
Final documentation Final documentation						■

Figure 2: Gantt chart

(planner, 2021)

System architecture

The following is a high level three tier diagram to show how this is going to be implemented. It shows the sender's client sending an email which reaches hMailServer which acts as an SMTP, MTA and deals with IMAP and POP3 protocols. This information is stored in a message repository in .eml format. Our web application will be hosted on a local web server, which has access to a local MySQL database. The flask application will host code to periodically check for new emails, parse the stored email messages into storable string format and store it in the MySQL database. Spam scoring will take place in the flask application and this score will also be stored in the MySQL database. The front end of the flask application will provide the user a way to log in, when logged in there will be pages separating the inbox and spam messages.

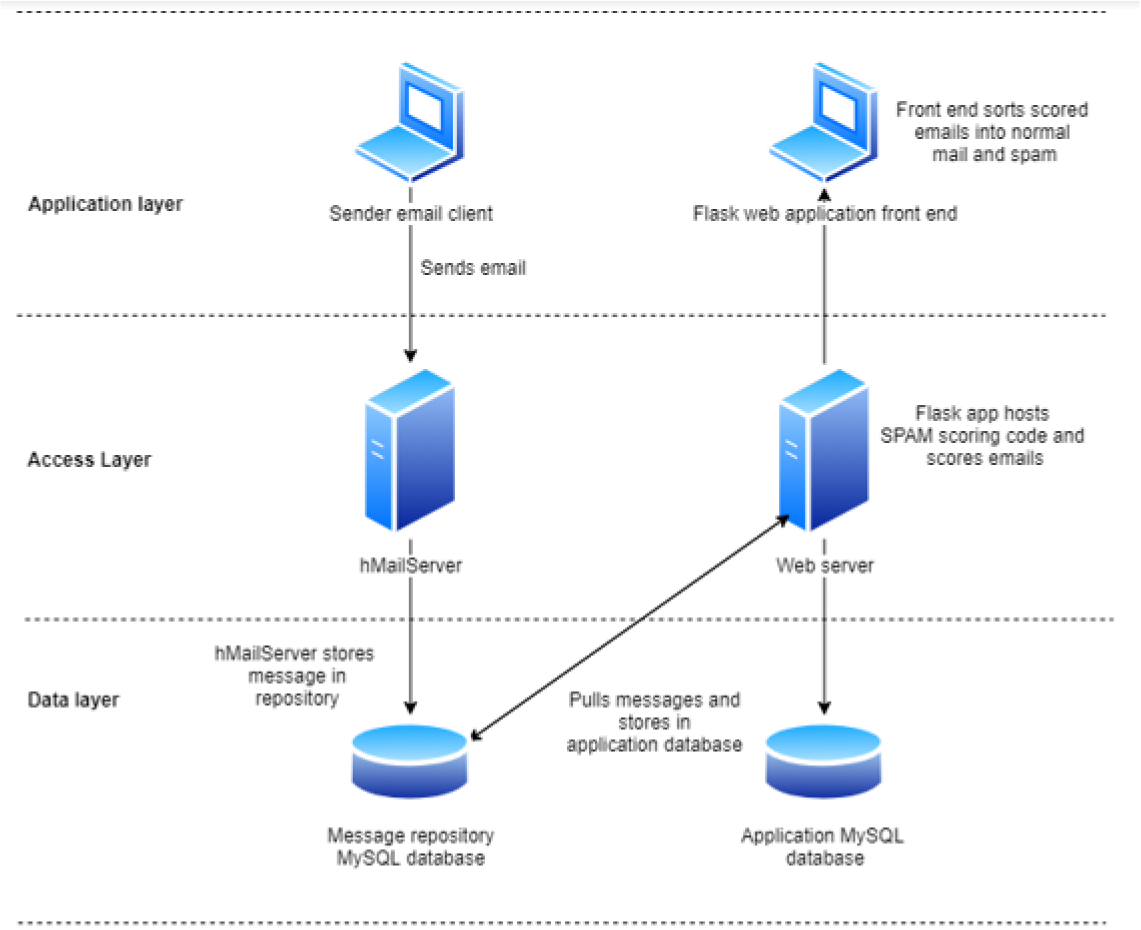

Application layer
Sender email client
Sends email
Flask web application front end
Front end sorts scored emails into normal mail and spam
Access Layer
hMailServer
Web server
Flask app hosts SPAM scoring code and scores emails
Data layer
hMailServer stores message in repository
Pulls messages and stores in application database
Message repository MySQL database
Application MySQL database

Figure 3: How each technology will be implemented

Email server software

In order to make a local email server, an email server software must be used to deal with the SMTP, IMAP and POP3 protocols, sending, receiving and storing emails. During the course of research, I considered many different email server software's, with the main requirements being that you can set up a local server and connect to an external database.

hMailServer

hMailServer is a free open-source email server software for Windows. It is written in C++ and C# (Knafve, 2020). It seems to be an attractive option for usage as it takes only a few minutes to install, deals with SMTP, POP3 and IMAP protocols. The software allows you to store the emails in an external database of your choice (Knafve, 2020). The software allows you to set up a local server and resolve issues on local domains that don't exist in a DNS server (Knafve, 2020).

Storage

A database will be required to store the emails the email server is sending and receiving, it will also store the user login information for the web application and the spam classification information. The database will also need to be used by the web application so the web application can display the emails, the content of the emails, email spam score and display them in the web application as either normal mail or spam.

MySQL

MySQL is an open-source relational database management system. It is possible to set up a MySQL database server on a windows machine. (MySQL :: MySQL Community Edition, 2020)It is compatible with hMailServer.

Web application

When researching the technology to build the web application and spam classification module of this program, there were requirements that needed to be met such as extensive libraries, machine learning etc. The language I use needs to be a good choice for machine learning, a good choice for use as a web application and needs to be thoroughly documented to assist in creating the web application.

Python

Python is a high level, easy to use, programming language. It is commonly used in data science projects due to its fast-prototyping attributes and the endless amounts of packages it has to assist you in what you need to do fast. For example, scikit-learn is a machine learning library for use with classification, regression and clustering (scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation, 2020). Python also has libraries such as pandas and numpy for working with big data sets. (Top Python Libraries: Numpy & Pandas, 2020). These kits seem to be used in spam classification, one method I found showed the use of NLTK, Pandas and Numpy as the three packages required to classify spam using python.(Spam Classifier in Python from scratch, 2020). It also has web framework libraries that you can use to create web applications easily, such as Flask and Django.

Flask

Flask is a micro web framework for python. It is used for creating web applications and APIs using python. A 'micro' framework means that the whole web application does not need to fit into one file, it just means flask keeps the core simple but extensible. It has a website of extensive documentation; it is deemed good for use as a web application technology. (Foreword — Flask Documentation (1.1.x), 2020)

SQLAlchemy

SQLAlchemy is an object relational mapper for use with Python. It gives developers full power and flexibility with SQL databases right from their code. It is commonly used in conjunction with Flask. (Object-relational Mappers (ORMs), 2020). Considering the web application will need to pull email data from a MySQL database this type of tool will be required to work alongside flask.

Entity–relationship Diagram

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types and specifies relationships that can exist between entities.

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

Entities

A real-world thing either living or non-living that is easily recognizable and nonrecognizable are known as entities. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event, or a concept, which stores data in the database. The characteristics of entities must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Attributes

It is a single-valued property of either an entity-type or a relationship-type.

For example, a user might have attributes: name, id, address, etc.

Relationship

Relationship is nothing but an association among two or more entities.

Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships

(ER Diagram: Entity Relationship Diagram Model | DBMS Example, 2021).

In the diagram below we have the following entities: user and email. We can see the relationship between each entity.

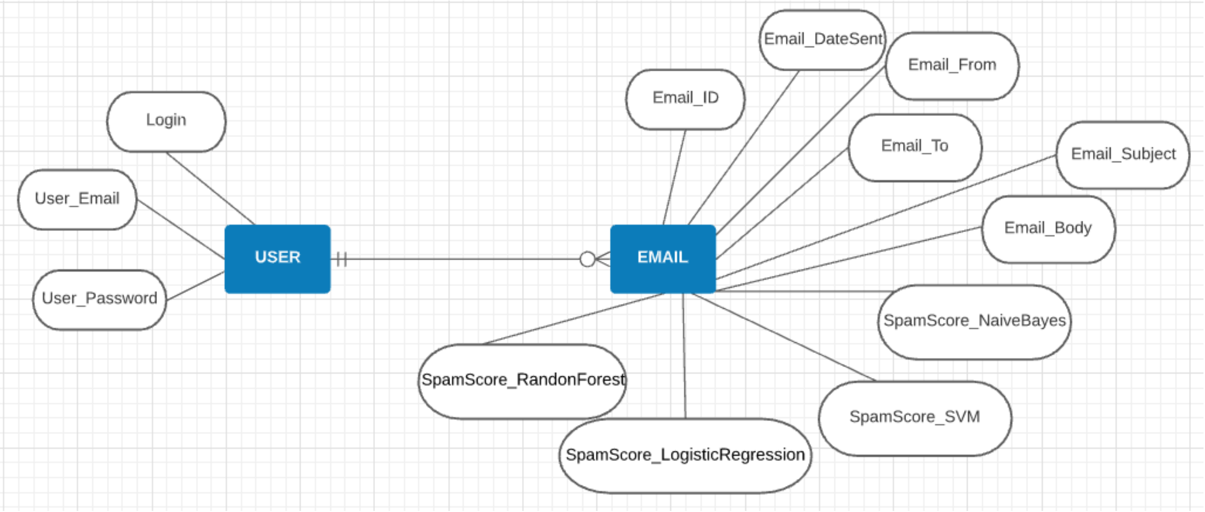


Figure 4: Application ER diagram

Class diagram

A class diagram is a type of static structure diagram. It is like an ER diagram. The diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

This type of diagram is essential. Before the development of any system this diagram can be very informative.

It not only delivers the idea to system owners on how the system will operate and how each component will interact with the other, but it also gives the developer a few good hints as to what they will be developing.

The following class diagram is designed for the Email spam filter system.

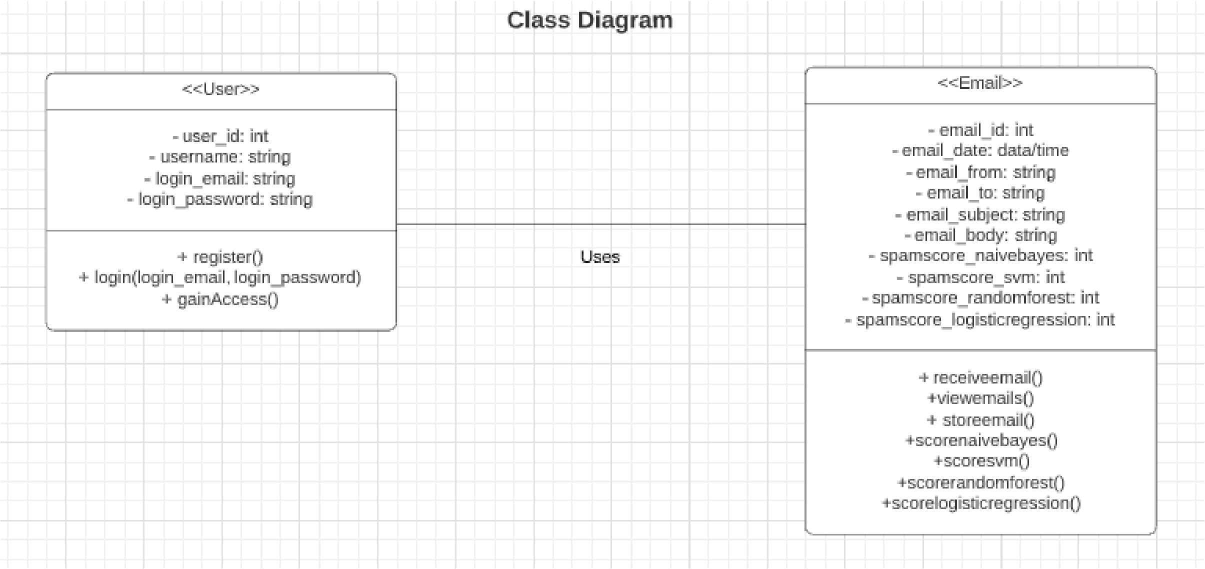


Figure 5: Application class diagram

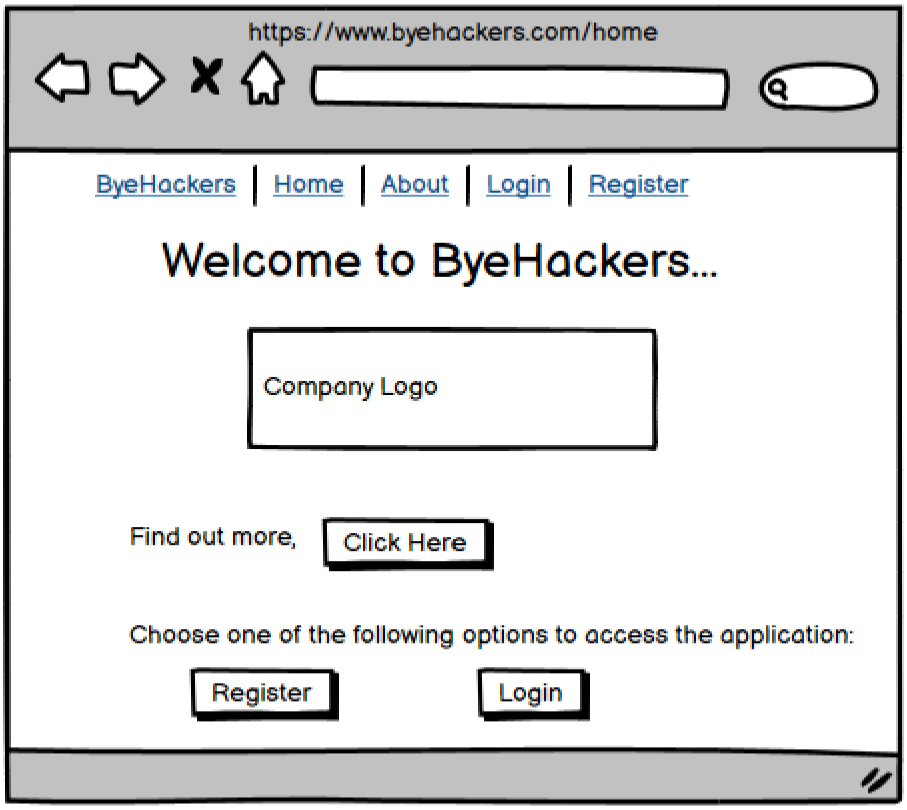
(The Easy Guide to UML Class Diagrams | Class Diagram Tutorial, 2020)

User interface

The interface design should be as simple as possible, informative, and easy to use. The flask web application will have simple colours to reflect positively and give a more user-friendly atmosphere. The web application will be attempted to be responsive, meaning it can be used on any device capable of running a browser. A few design possibilities are provided below. The following diagrams are simple and clear prototypes of how the Email Spam filter web application should look like and how the application will flow.
This section contains figures 6 - 26.

Home screen

Anyone who visits the application will be presented with the home screen. From this page they can visit three other pages which are about, register and the login page.



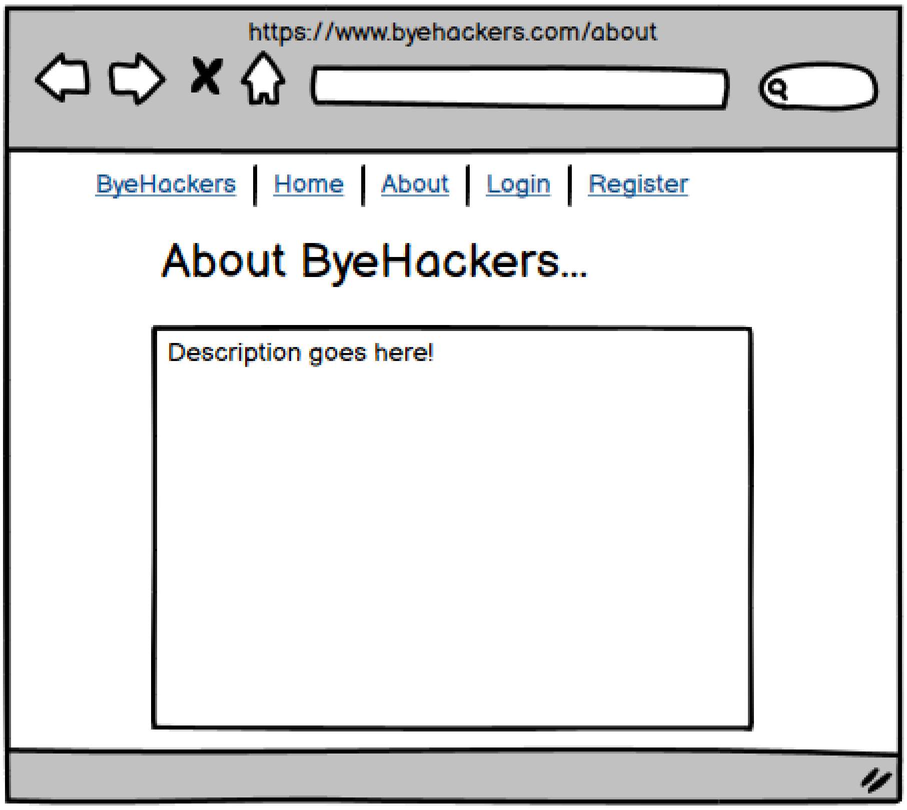
Welcome to ByeHackers...

Find out more,

Choose one of the following options to access the application:

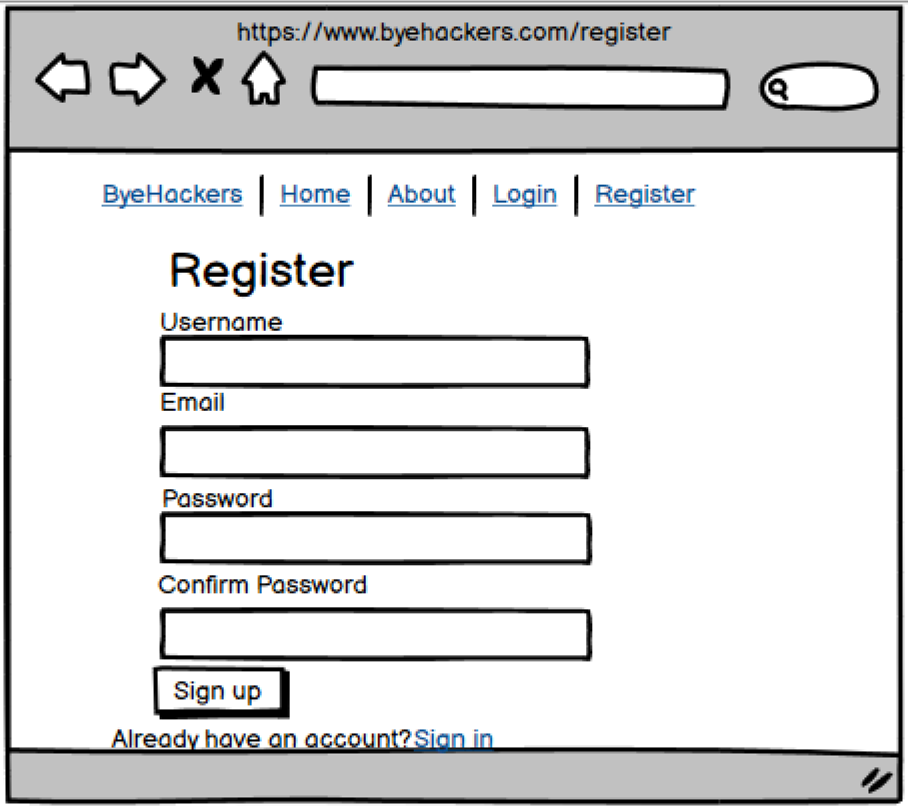
About page

The user can find out information about the application on this page. They can also navigate back to the home page or move to the register or login page from here.


https://www.byehackers.com/about
ByeHackers | Home | About | Login | Register
About ByeHackers...
Description goes here!

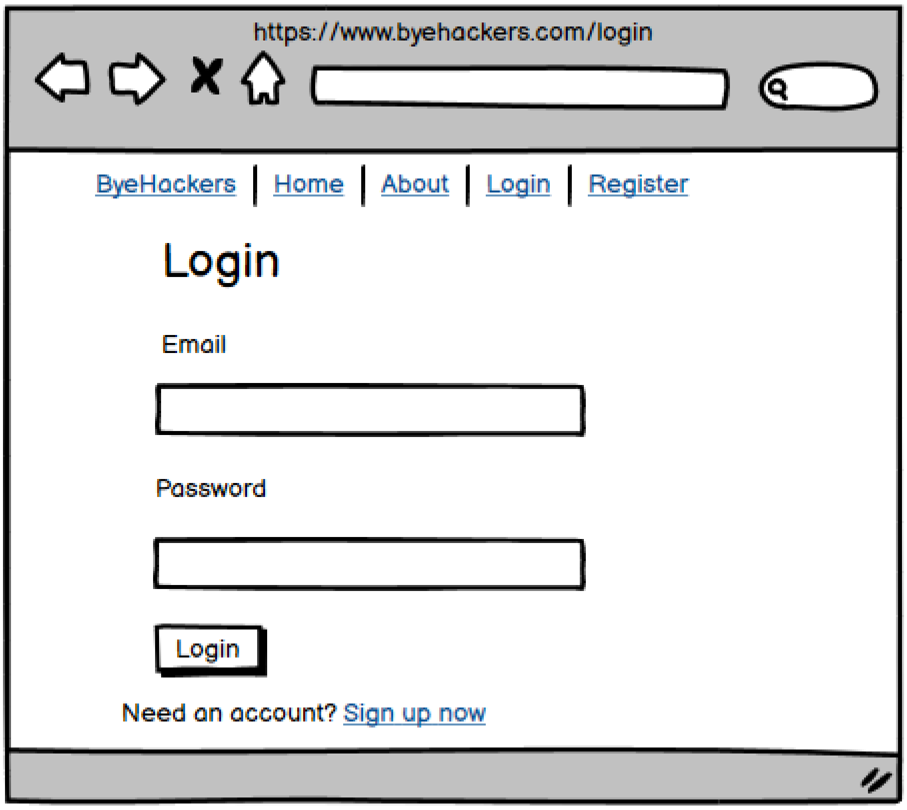
Register Screen

The user must register to gain access to the application. When registering they will need to provide the following information: username, email address and password. The same username and email cannot be used for two accounts.

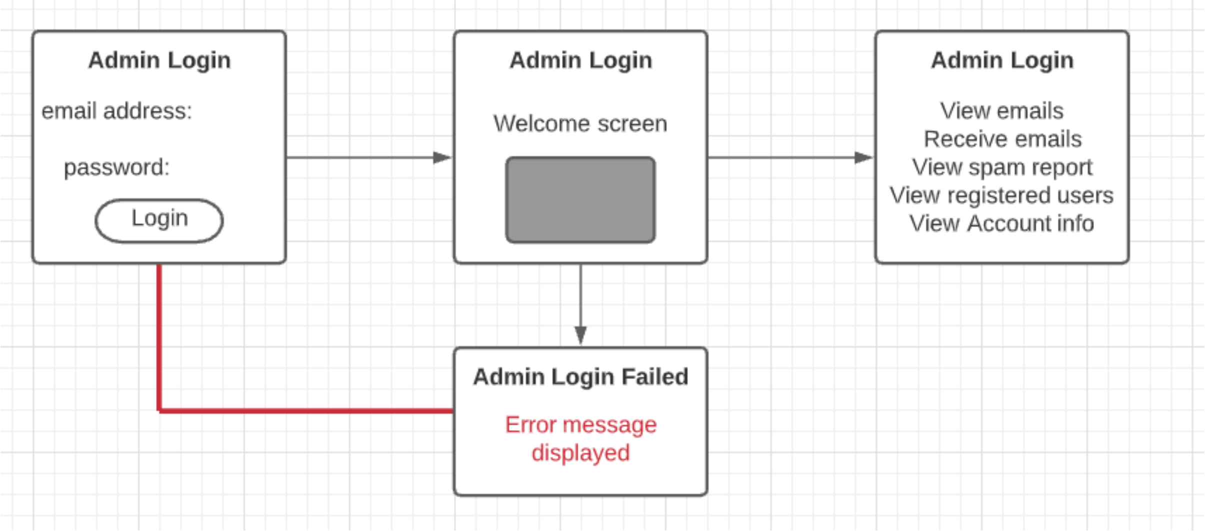

https://www.byehackers.com/register
ByeHackers | Home | About | Login | Register
Register
Username
Email
Password
Confirm Password
Sign up
Already have an account? Sign in

Login Screen

After the user has registered, they should login to validate their information and get access to the system. When the user is logging in, they will ned to provide the email address and password they used to register with the application. If they provide incorrect details an error message will display for them.


https://www.byehackers.com/login
ByeHackers | Home | About | Login | Register
Login
Email
Password
Login
Need an account? Sign up now

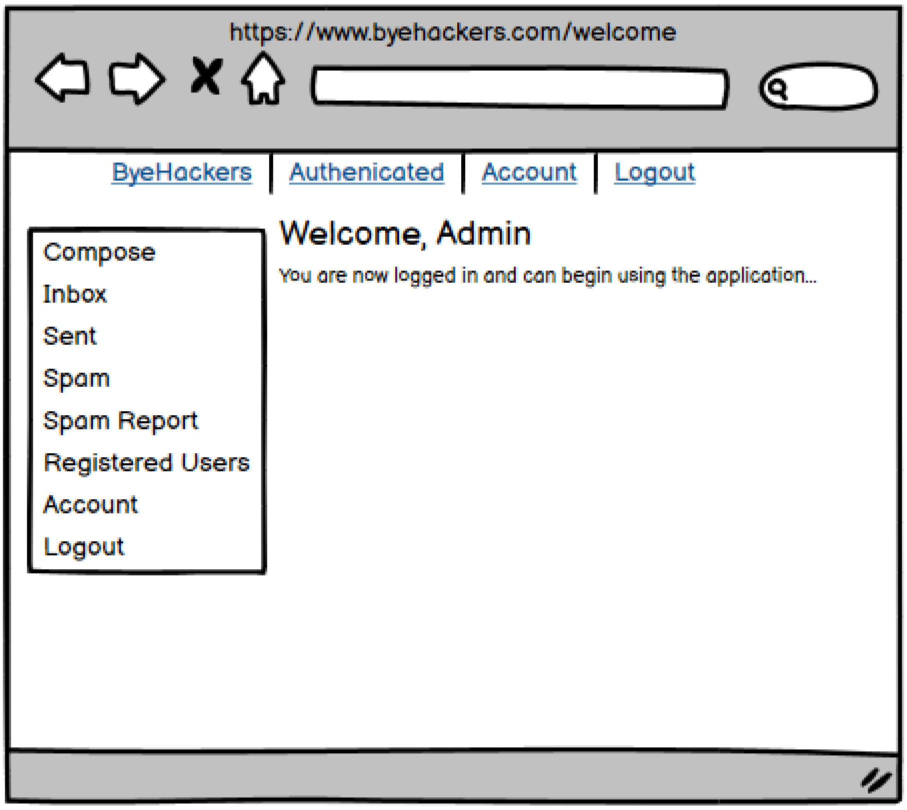
Admin Login Screen and Flow



The following diagrams are simple prototypes of each page of the web application for an admin user. The design will be simple to avoid over-complexity and achieve good user experience.

Welcome screen

Once the admin has successfully logged into the application, they will be presented with the welcome screen. From this page they can pick where they would like to navigate to from the nav-bar on the left of the screen.

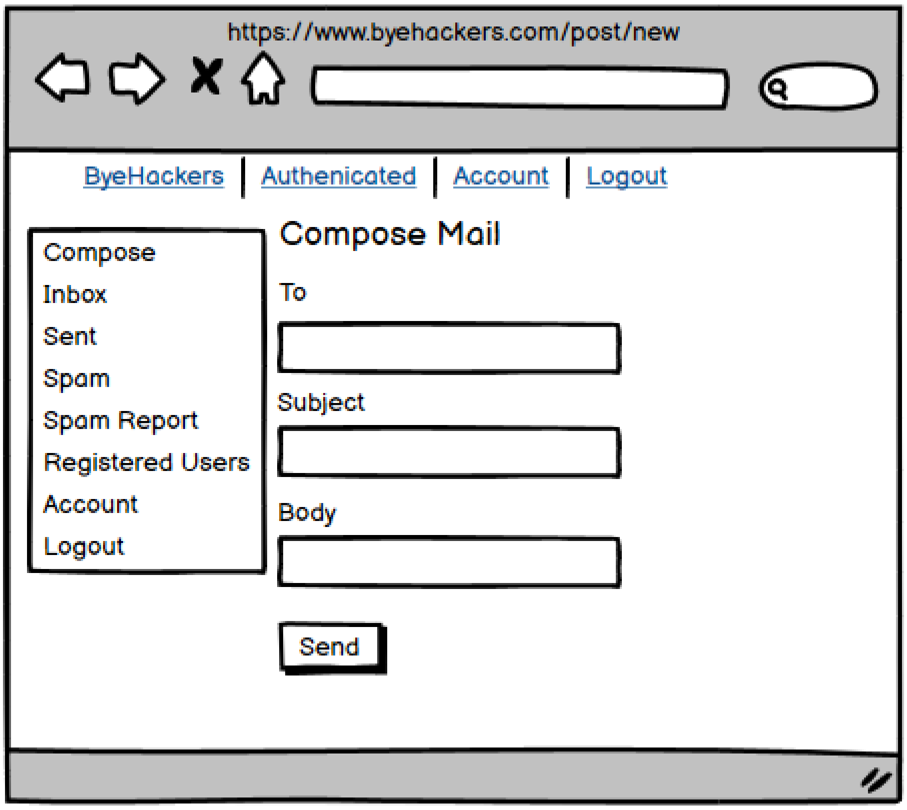


Welcome, Admin

You are now logged in and can begin using the application...

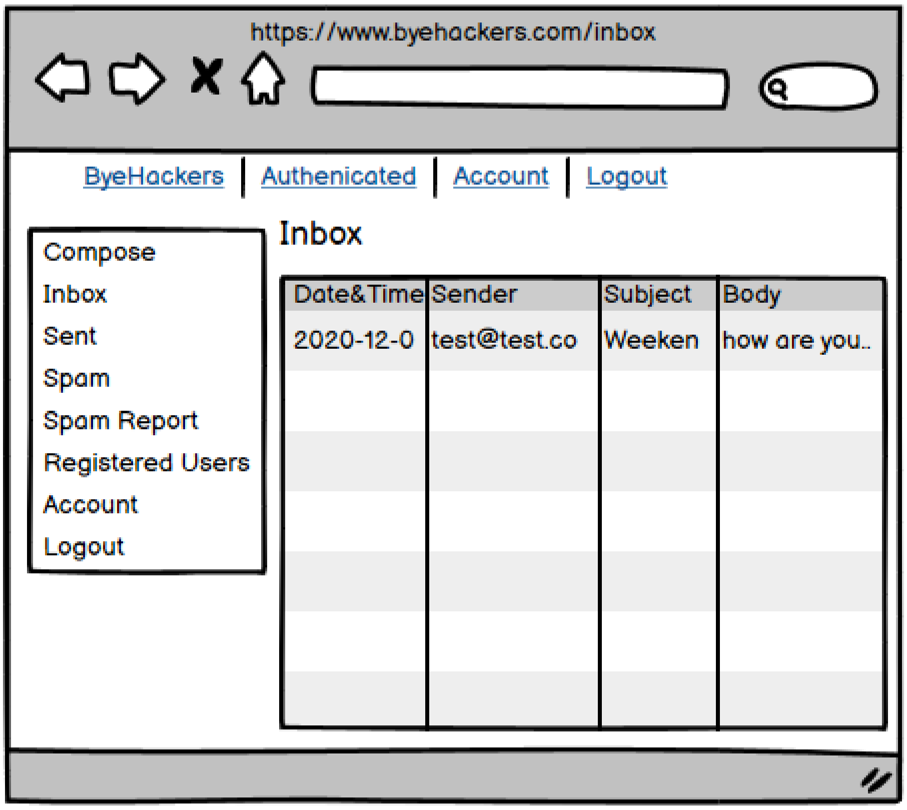
Compose Mail

The admin user can send an email to the account of their choice by filling in the fields on this page and click “Send”.



Inbox

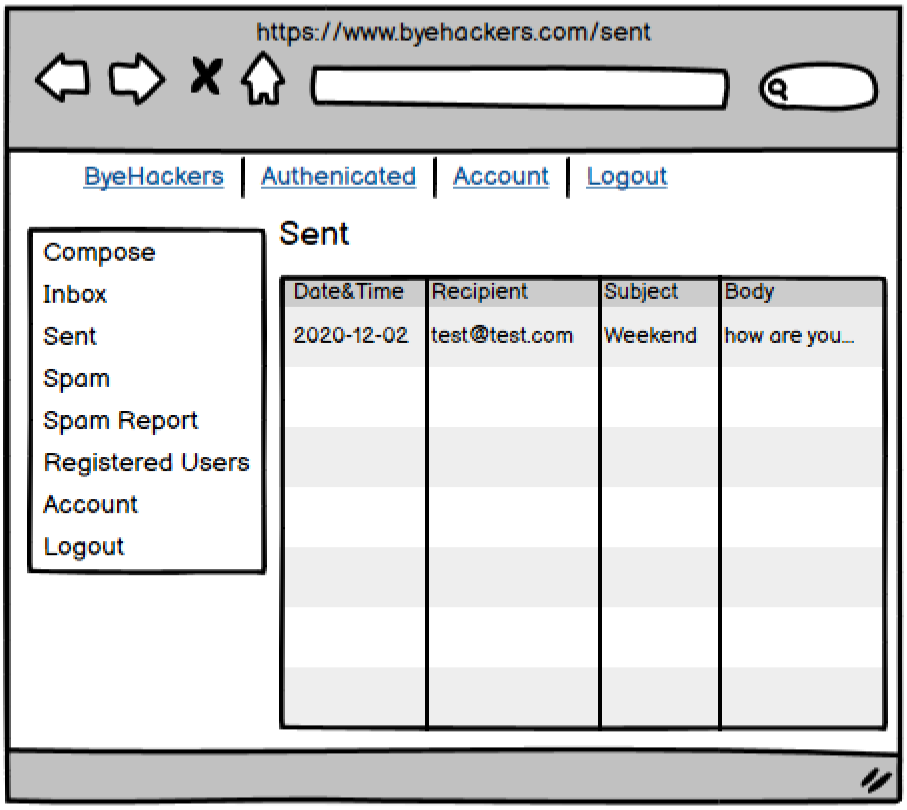
All the admins mail they have received will display here. No emails classified as spam will appear on this page. The most recent email will appear at the top of the list. For each email the following details will be displayed: date & time received, sender, subject and the body of the email.



Date&Time	Sender	Subject	Body
2020-12-0	test@test.co	Weeken	how are you..

Sent mail

The admin can view all the emails they have sent and who they have sent these emails too.

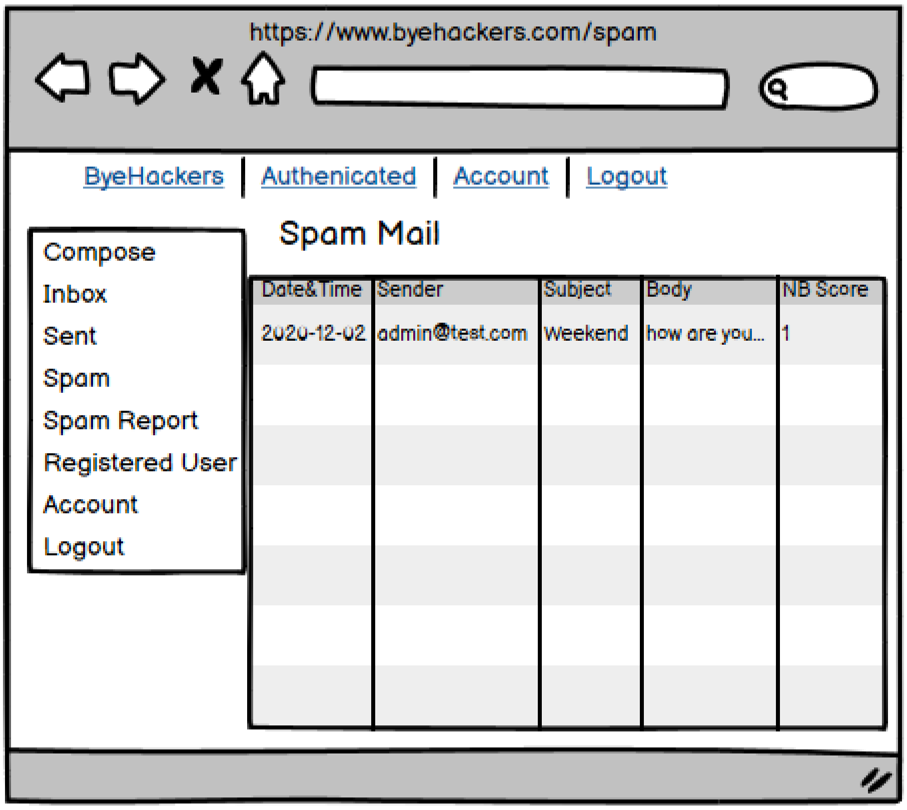

https://www.byehackers.com/sent
ByeHackers | Authenicated | Account | Logout
Compose
Inbox
Sent
Spam
Spam Report
Registered Users
Account
Logout

Sent

Date&Time	Recipient	Subject	Body
2020-12-02	test@test.com	Weekend	how are you…

Spam Mail

On this screen the admin can view the spam mail they received. If emails appear on this screen this means they have been classified as spam by the machine learning algorithms.

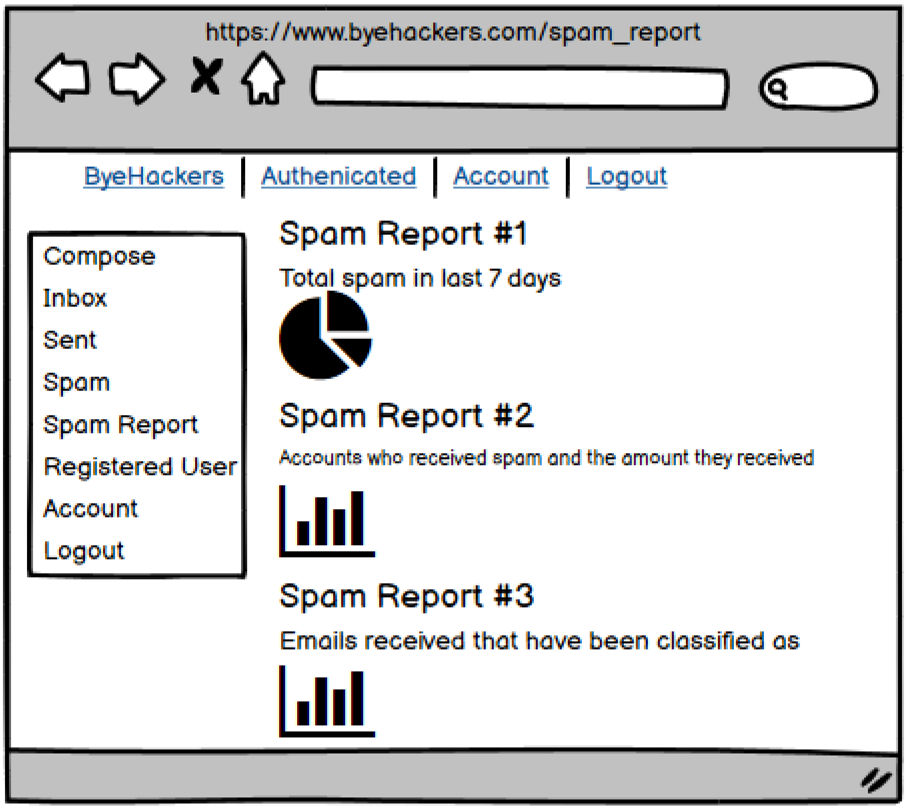

https://www.byehackers.com/spam
ByeHackers | Authenicated | Account | Logout
Compose
Inbox
Sent
Spam
Spam Report
Registered User
Account
Logout

Spam Mail

Date&Time	Sender	Subject	Body	NB Score
2020-12-02	admin@test.com	Weekend	how are you...	1

Spam Report

Only the admin user has access to this page. They can generate report as often as they wish. This will allow them to see how much spam is being received from all the registered users accounts.



Spam Report #1

Total spam in last 7 days

Spam Report #2

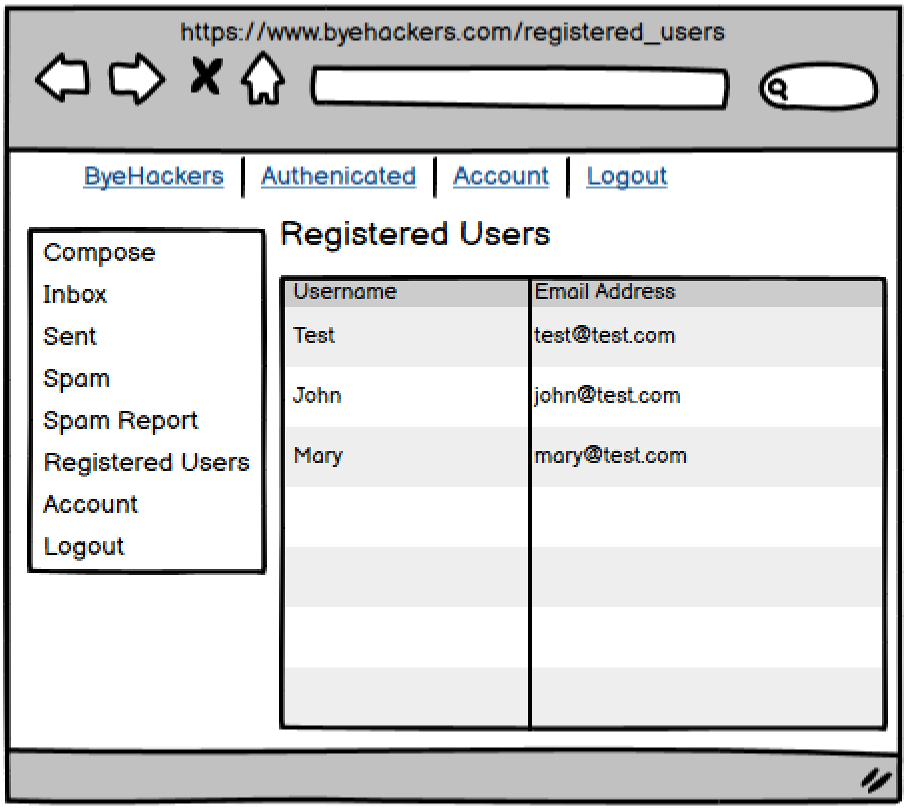
Accounts who received spam and the amount they received

Spam Report #3

Emails received that have been classified as

Registered Users

Only the admin user has access to this page. They can view all the registered users of the application.



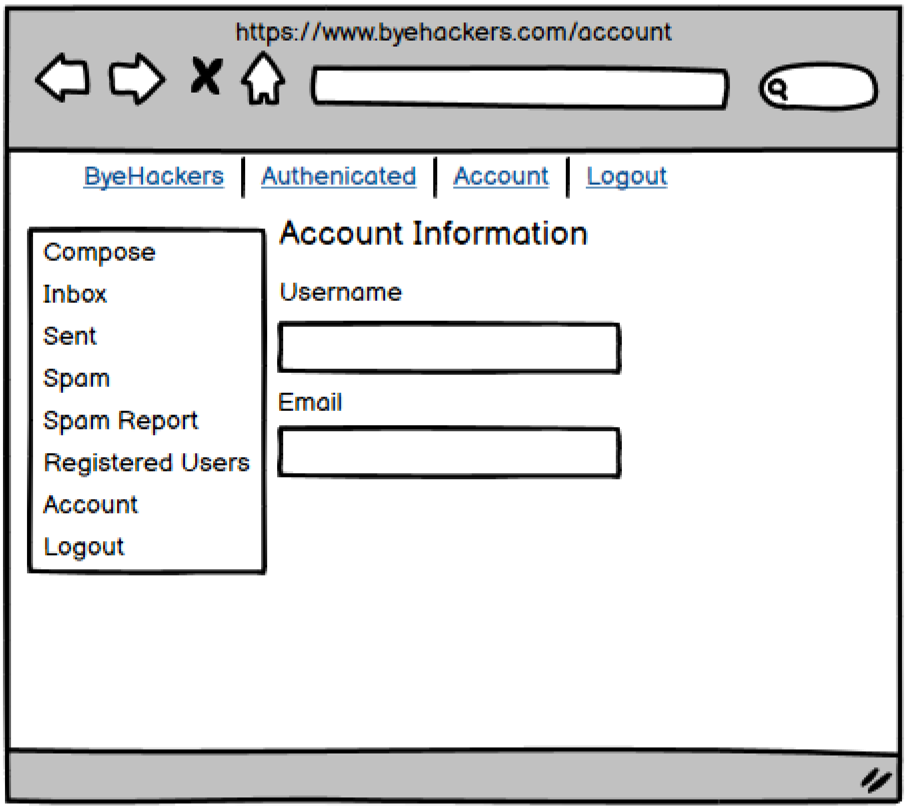
https://www.byehackers.com/registered_users
ByeHackers | Authenicated | Account | Logout
Compose
Inbox
Sent
Spam
Spam Report
Registered Users
Account
Logout

Registered Users

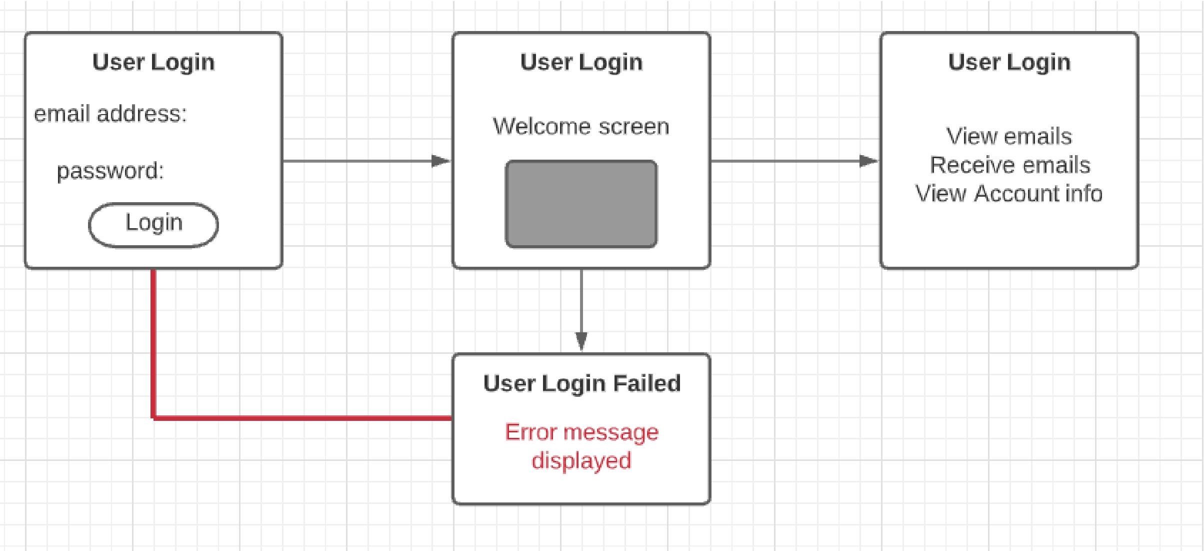
Username	Email Address
Test	test@test.com
John	john@test.com
Mary	mary@test.com

Account Information

This screen will display the admins account username and password that they registered with.



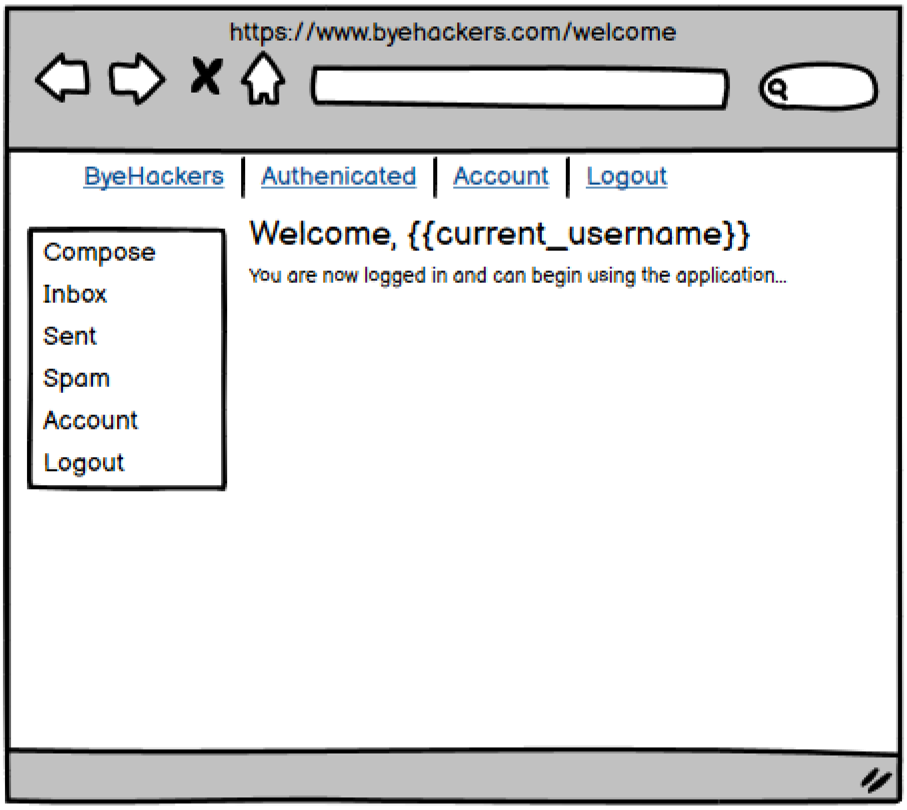
User Login Screen and Flow



The following diagrams are simple prototypes of each page of the web application for regular user. The design will be simple to avoid over-complexity and achieve good user experience. They will not have access to spam reports or view all the registered users of the application like the admin.

Welcome screen

Once the user has successfully logged into the application, they will be presented with the welcome screen. From this page they can pick where they would like to navigate to from the nav-bar on the left of the screen.

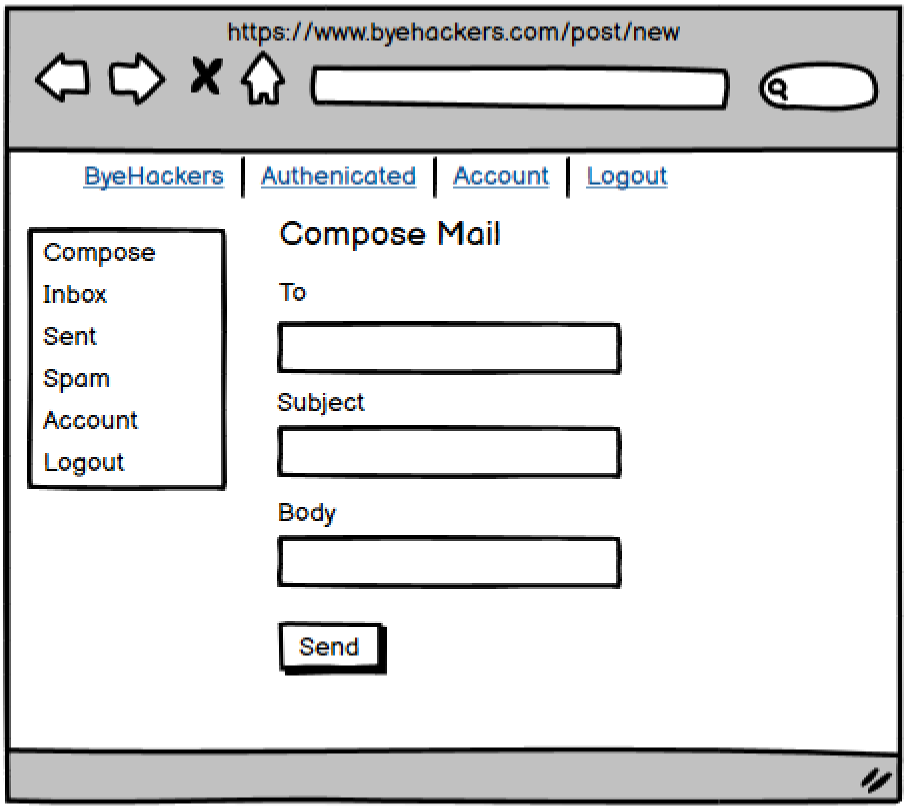


Welcome, {{current_username}}

You are now logged in and can begin using the application...

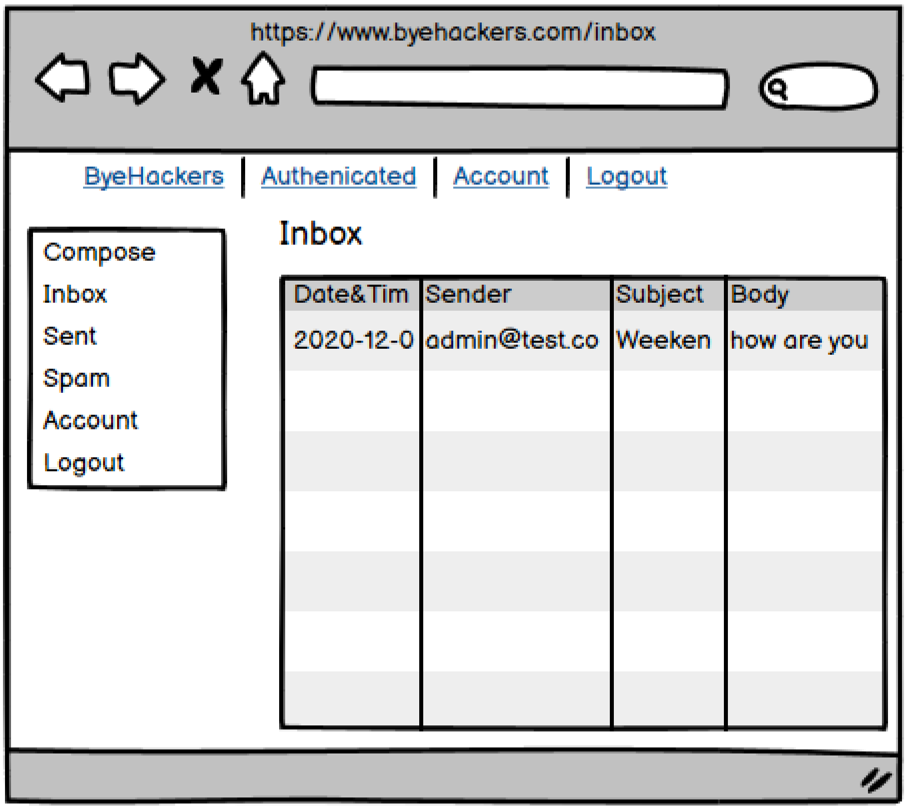
Compose Mail

The user can send an email to the account of their choice by filling in the fields on this page and click “Send”.



Inbox

All the users mail they have received will display here. No emails classified as spam will appear on this page. The most recent email will appear at the top of the list. For each email received the following details will be displayed: date & time received, sender, subject and the body of the email.

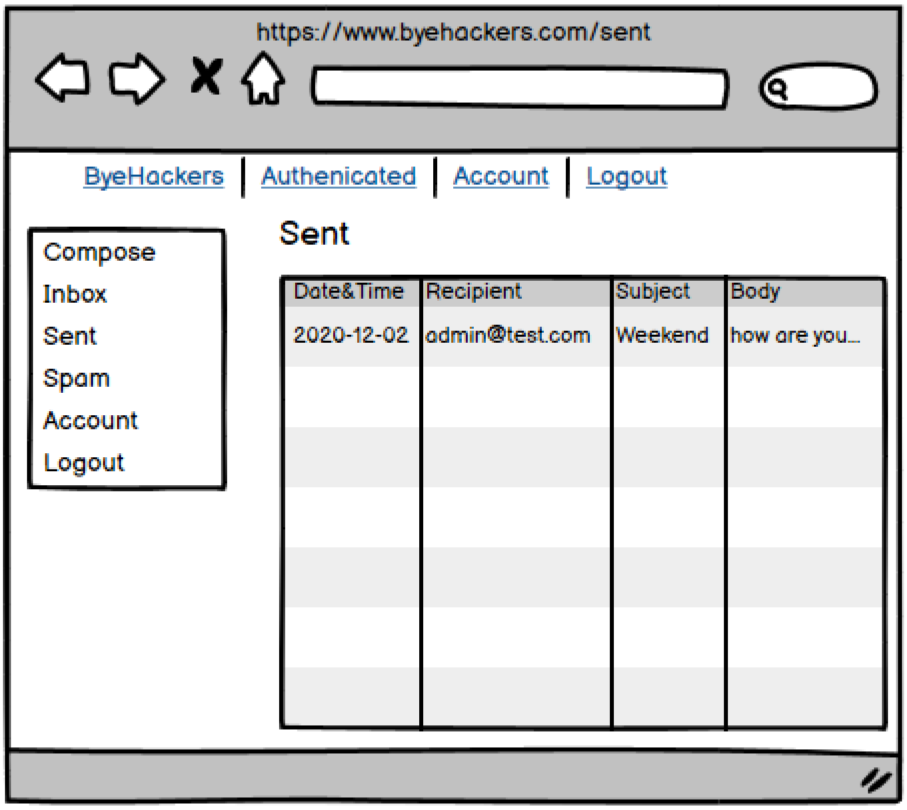


Inbox

Date&Tim	Sender	Subject	Body
2020-12-0	admin@test.co	Weeken	how are you

Sent mail

The user can view all the emails they have sent and who they have sent these emails too.



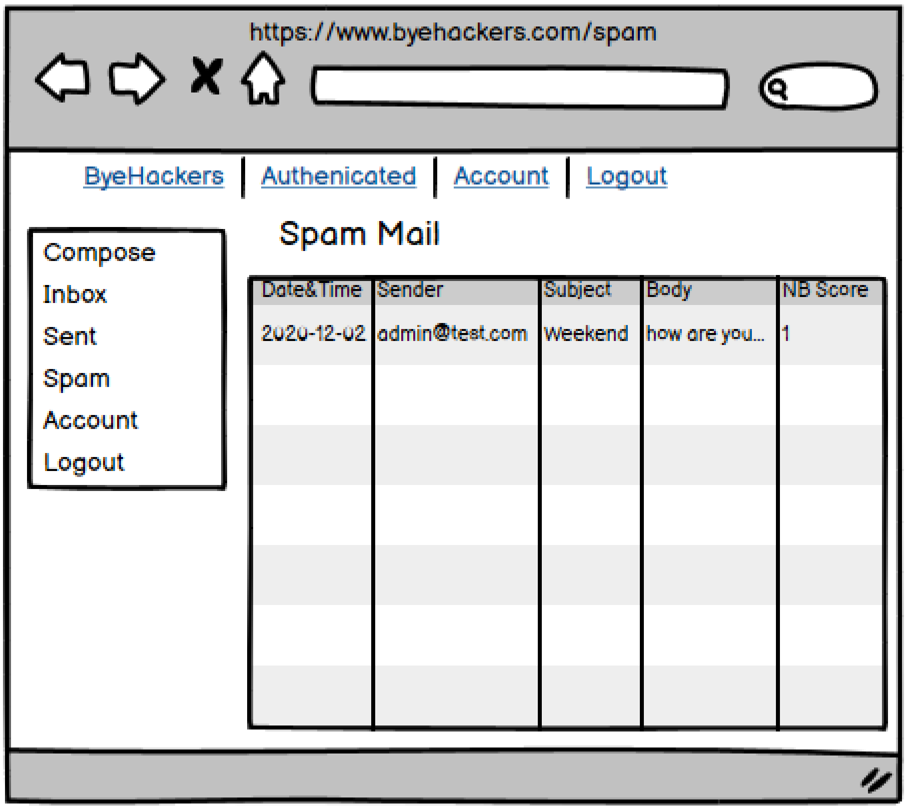
https://www.byehackers.com/sent
ByeHackers | Authenicated | Account | Logout
Compose
Inbox
Sent
Spam
Account
Logout

Sent

Date&Time	Recipient	Subject	Body
2020-12-02	admin@test.com	Weekend	how are you...

Spam Mail

On this screen the user can view the spam mail they received. If emails appear on this screen this means they have been classified as spam by the machine learning algorithms.

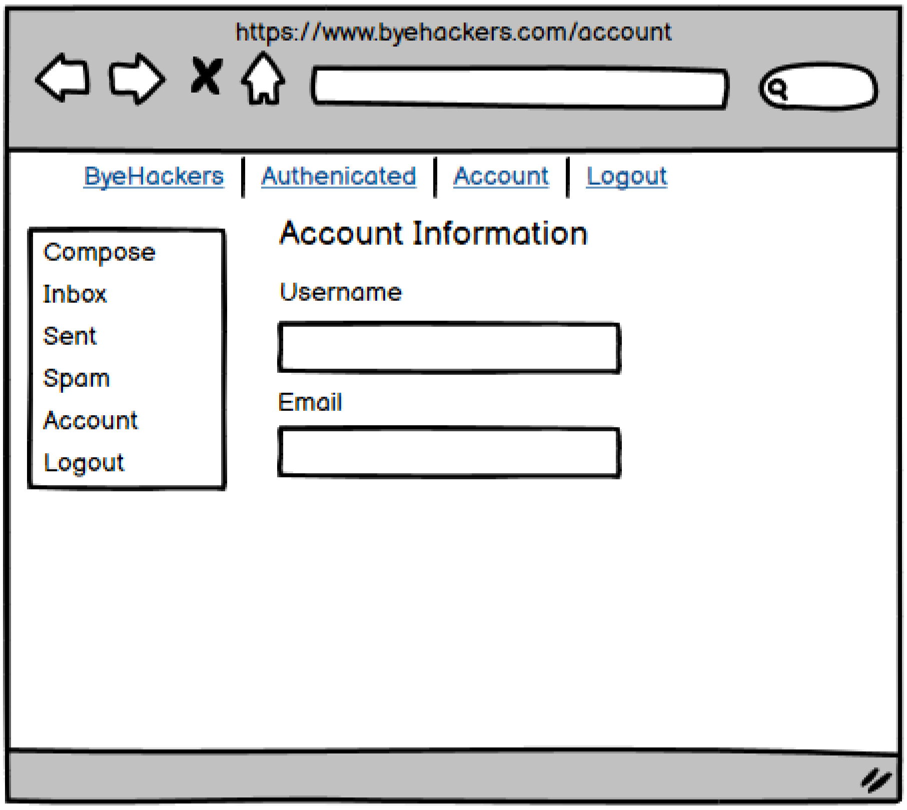


https://www.byehackers.com/spam
ByeHackers | Authenicated | Account | Logout
Compose
Inbox
Sent
Spam
Account
Logout
Spam Mail

Date&Time	Sender	Subject	Body	NB Score
2020-12-02	admin@test.com	Weekend	how are you...	1

Account Information

This screen will display the users account username and password that they registered with.



(3 rules for a good user experience (UX), 2020)

Database Design

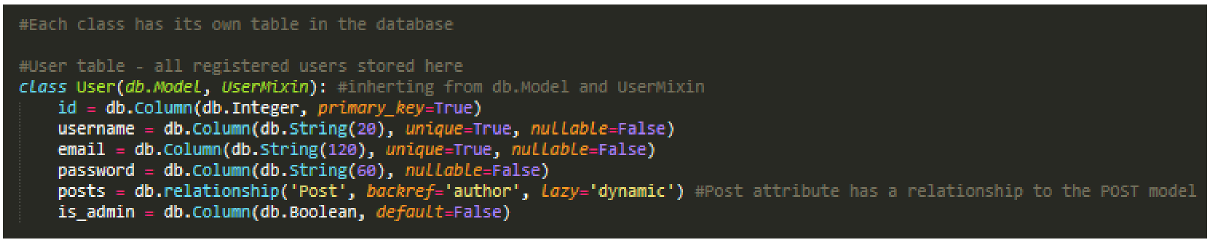
There is going to be one database created for this project. The database will be easily to follow and will be queried to display mail and reports on the web application. The database will be designed using SQL Alchemy, which is an object relational mapper that can be used with flask web framework. This will control the creation of tables and relationships from the code. Any changes made to the code will be migrated to the MySQL database. The section contains figure 27 – 33.

Users Table

Database Name: pythonlogin

Tables Name: user

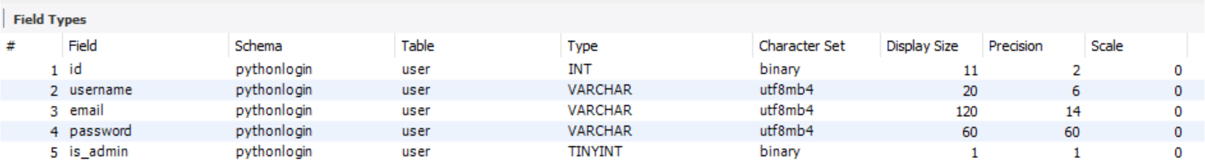
Description: This table stores the registered users' details. This table is used to allow access to the web app where the user can view their emails.



```
#Each class has its own table in the database

#User table - all registered users stored here
class User(db.Model, UserMixin): #inherting from db.Model and UserMixin
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(60), nullable=False)
    posts = db.relationship('Post', backref='author', lazy='dynamic') #Post attribute has a relationship to the POST model
    is_admin = db.Column(db.Boolean, default=False)
```

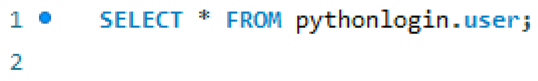
Table Structure:



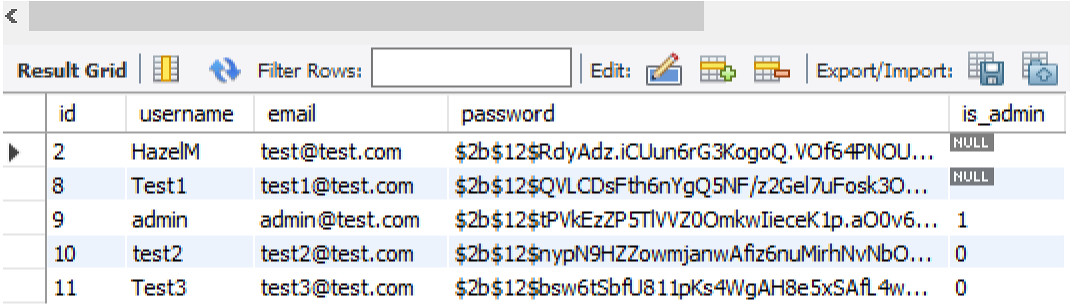
Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	pythonlogin	user	INT	binary	11	2	0
2	username	pythonlogin	user	VARCHAR	utf8mb4	20	6	0
3	email	pythonlogin	user	VARCHAR	utf8mb4	120	14	0
4	password	pythonlogin	user	VARCHAR	utf8mb4	60	60	0
5	is_admin	pythonlogin	user	TINYINT	binary	1	1	0

Sample Data:



```
SELECT * FROM pythonlogin.user;
```



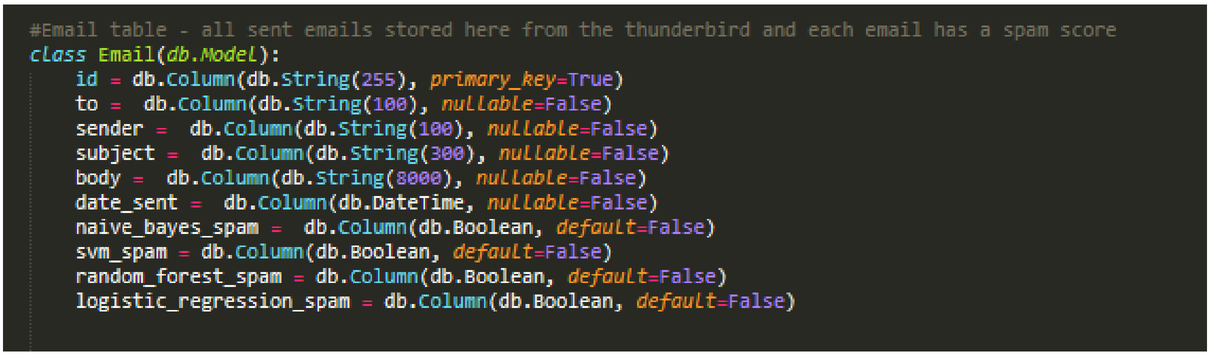
id	username	email	password	is_admin
2	HazelM	test@test.com	$2b$12$RdyAdz.iCUun6rG3KogoQ.VOf64PNOU...	NULL
8	Test1	test1@test.com	$2b$12$QVLCDsFth6nYgQ5NF/z2Gel7uFosk3O...	NULL
9	admin	admin@test.com	$2b$12$tPVkEzZP5TlVVZ0OmkwIieceK1p.aO0v6...	1
10	test2	test2@test.com	$2b$12$nypN9HZZowmjanwAfiz6nuMirhNvNbO...	0
11	Test3	test3@test.com	$2b$12$bsw6tSbfU811pKs4WgAH8e5xSAfL4w...	0

Email Table

Database Name: pythonlogin

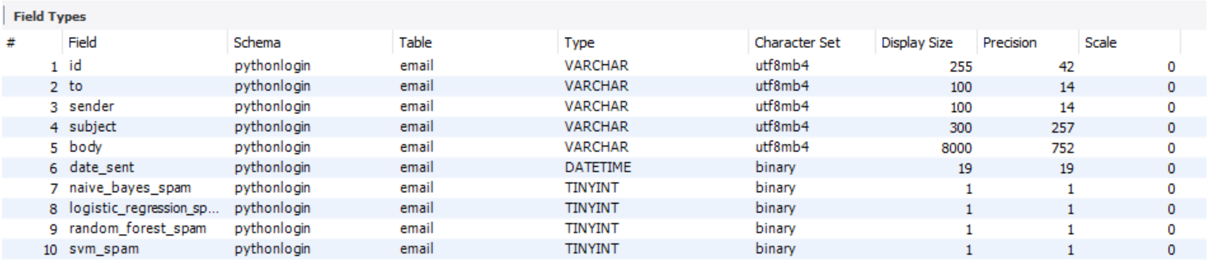
Tables Name: email

Description: This table stores emails and each emails spam score for the machine learning algorithms.



```
#Email table - all sent emails stored here from the thunderbird and each email has a spam score
class Email(db.Model):
    id = db.Column(db.String(255), primary_key=True)
    to =  db.Column(db.String(100), nullable=False)
    sender =  db.Column(db.String(100), nullable=False)
    subject =  db.Column(db.String(300), nullable=False)
    body =  db.Column(db.String(8000), nullable=False)
    date_sent =  db.Column(db.DateTime, nullable=False)
    naive_bayes_spam =  db.Column(db.Boolean, default=False)
    svm_spam = db.Column(db.Boolean, default=False)
    random_forest_spam = db.Column(db.Boolean, default=False)
    logistic_regression_spam = db.Column(db.Boolean, default=False)
```

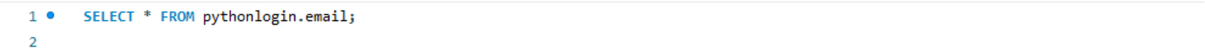
Table Structure:



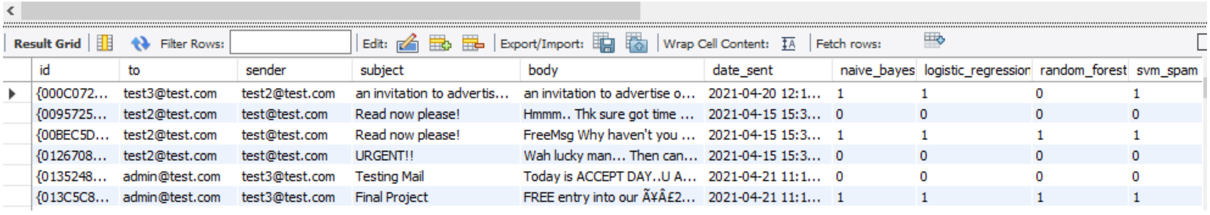
Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	pythonlogin	email	VARCHAR	utf8mb4	255	42	0
2	to	pythonlogin	email	VARCHAR	utf8mb4	100	14	0
3	sender	pythonlogin	email	VARCHAR	utf8mb4	100	14	0
4	subject	pythonlogin	email	VARCHAR	utf8mb4	300	257	0
5	body	pythonlogin	email	VARCHAR	utf8mb4	8000	752	0
6	date_sent	pythonlogin	email	DATETIME	binary	19	19	0
7	naive_bayes_spam	pythonlogin	email	TINYINT	binary	1	1	0
8	logistic_regression_sp...	pythonlogin	email	TINYINT	binary	1	1	0
9	random_forest_spam	pythonlogin	email	TINYINT	binary	1	1	0
10	svm_spam	pythonlogin	email	TINYINT	binary	1	1	0

Sample Data:



```
SELECT * FROM pythonlogin.email;
```



id	to	sender	subject	body	date_sent	naive_bayes	logistic_regressior	random_forest	svm_spam
{000C072...	test3@test.com	test2@test.com	an invitation to advertis...	an invitation to advertise o...	2021-04-20 12:1...	1	1	0	1
{0095725...	test2@test.com	test@test.com	Read now please!	Hmmm.. Thk sure got time ...	2021-04-15 15:3...	0	0	0	0
{00BEC5D...	test2@test.com	test@test.com	Read now please!	FreeMsg Why haven't you ...	2021-04-15 15:3...	1	1	1	1
{0126708...	test2@test.com	test@test.com	URGENT!!	Wah lucky man... Then can...	2021-04-15 15:3...	0	0	0	0
{0135248...	admin@test.com	test3@test.com	Testing Mail	Today is ACCEPT DAY..U A...	2021-04-21 11:1...	0	0	0	0
{013C5C8...	admin@test.com	test3@test.com	Final Project	FREE entry into our Ã¥Â£2...	2021-04-21 11:1...	1	1	1	1

Detailed use cases

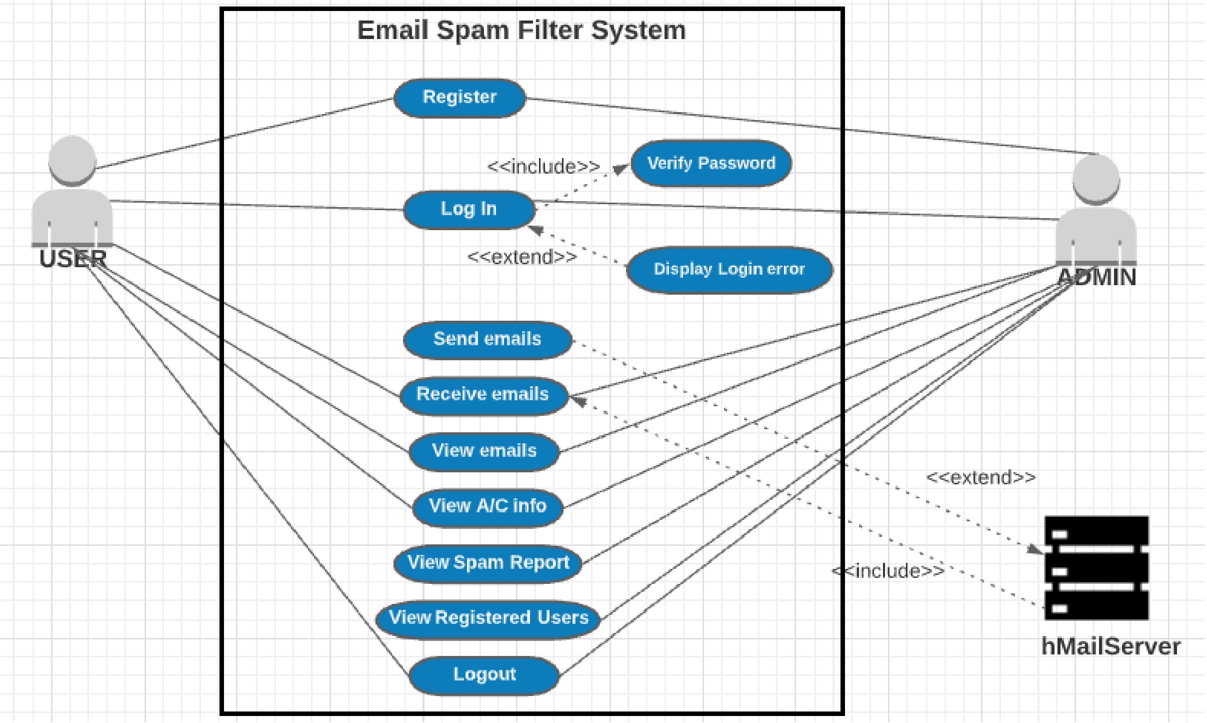


Figure 34: Detailed use-case

1. Register

Primary Actors: User, Admin

Precondition: The user has successfully registered to the web application.

Steps:

1. User clicks on register button on the desktop application
2. User enters their username, email address and password twice and clicks register

Expected Result: The user has registered to the system successfully

Alternative(s):

1. Invalid information is entered by the user e.g. the email address entered is already in use

2. Login

Primary Actors: User, Admin

Precondition: The user has successfully registered with the system.

Steps:

1. User visits the website
2. User enters their email address and password on the login screen

3. User clicks Login
4. If credentials match the database for this user a successfully login will occur
5. The user will have access to their emails

Expected Result: The user successfully logs into the application and is presented with the welcome screen.

Alternative(s):

1. The user enters incorrect information - message to the screen “Login unsuccessful. Please check your email and password”.

3. **Send Emails**

Primary Actors: User, Admin

Precondition: The user has an account setup on the hMailServer

Steps:

1. The user will login to hMailServer
2. The user will create a new email
3. The user will enter the required fields to send the email
4. The user clicks send

Expected Result: The user has successfully sent an email.

Alternative(s):

1. The user enters an invalid recipient email address - An email is received to the user “Address not found”

4. **Receive Emails**

Primary Actors: User, Admin

Precondition: The recipient is successfully logged into the system. An email is successfully sent to the recipient.

Steps:

1. The user will login to the system
2. The user will navigate to their inbox folder to view their email
3. The users most recent email received will appear at the top of the list on the inbox screen

Expected Result: The email is successfully received and is visible to the user in the correct folder.

Alternative(s):

1. The email is not received. The email could be classified as spam and will be placed into the spam folder instead.

5. **View Emails**

Primary Actors: User, Admin

Precondition: The user is successfully logged into the system.

Steps:

1. The user will login to the system
2. The user will navigate to the specific folder (sent, inbox or spam folder).

Expected Result: An email which seems suspicious is displayed in the spam folder and has a spam score of 1 in the database. An email received with spam score of 0 will be displayed in the inbox folder. All the users sent mails will be displayed in the sent folder.

Alternative(s):

1. An email with a spam score of 1 displays in the inbox folder.

6. **View Account Information**

Primary Actors: User, Admin

Precondition: The user is successfully logged into the system.

Steps:

1. The user is successfully logged into the system
2. The user visits the account page.
3. The users username and email address they used to register for the application will display.

Expected Result: Their account information displays.

Alternative(s):

1. The data fields have not populated. An issue occurred to retrieve these from the database.

7. **View Spam report**

Primary Actors: Admin

Precondition: The administrator has successfully logged in to the system.

Steps:

1.The administrator is successfully logged into the system

2.The administrator visits the report page.

Expected Result: The spam report is displayed to the Administrator

Alternative(s):

1. An error occurred while querying the database and incorrect data is displayed.

8. View Register report

Primary Actors: Admin

Precondition: The administrator has successfully logged in to the system.

Steps:

1. The administrator is successfully logged into the system
2. The administrator visits the register page

Expected Result: All registered users are displayed to the Administrator

Alternative(s):

1. An error occurred while querying the database and registered users are missing from the list.

9. Logout

Primary Actors: User, Admin

Precondition: The user wants to logout of the system.

Steps:

1. The user clicks the “Logout” button
2. The user will be redirected to thew homepage of the application.

Expected Result: The user successfully logs out of the system.

Alternative(s):

1. The user may press the “Logout” button by mistake – the user will have to log back into the system to view their emails again.

Sequence diagram

A sequence diagram is a type of interaction diagram. This type of diagram states clearly how and in what order a group of objects works together.

Software developers and business professionals use these diagrams mainly for a new system or to document an existing process.

Register sequence diagram

This diagram displays the process when registering for the application.

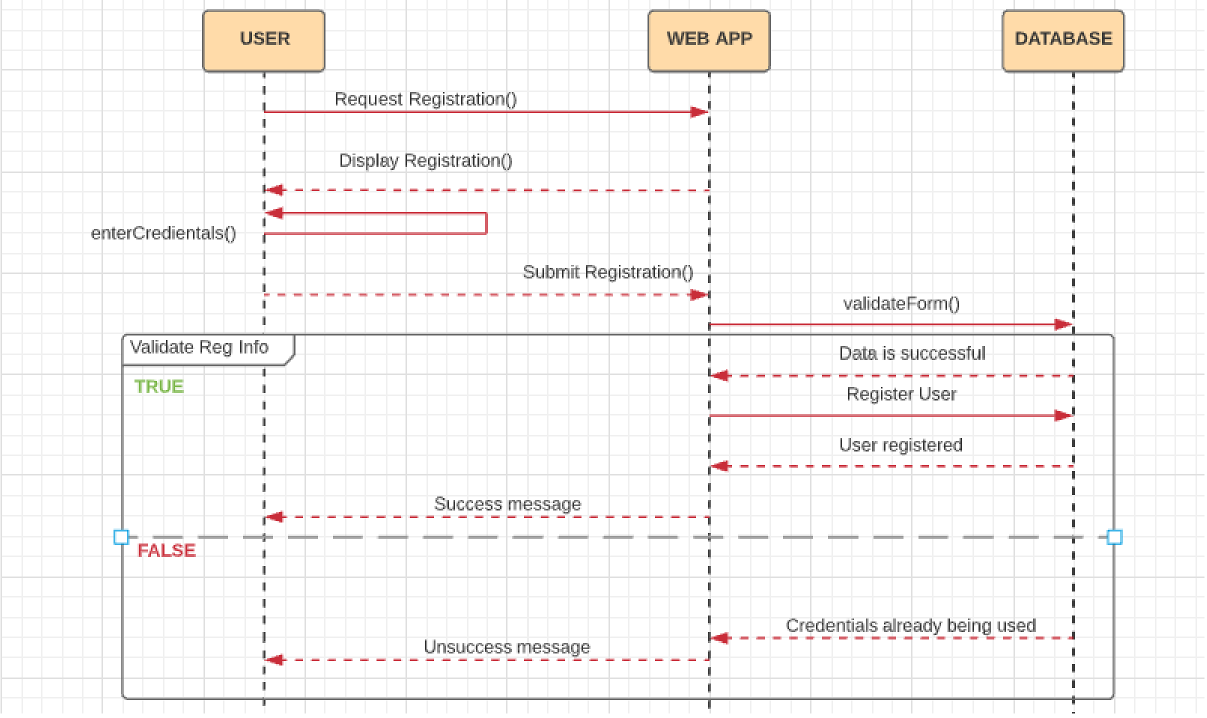


Figure 35: Register

Login sequence diagram

Once the user has registered with the system, they then need to complete to the login form and authenticate with the system to use the systems functions.

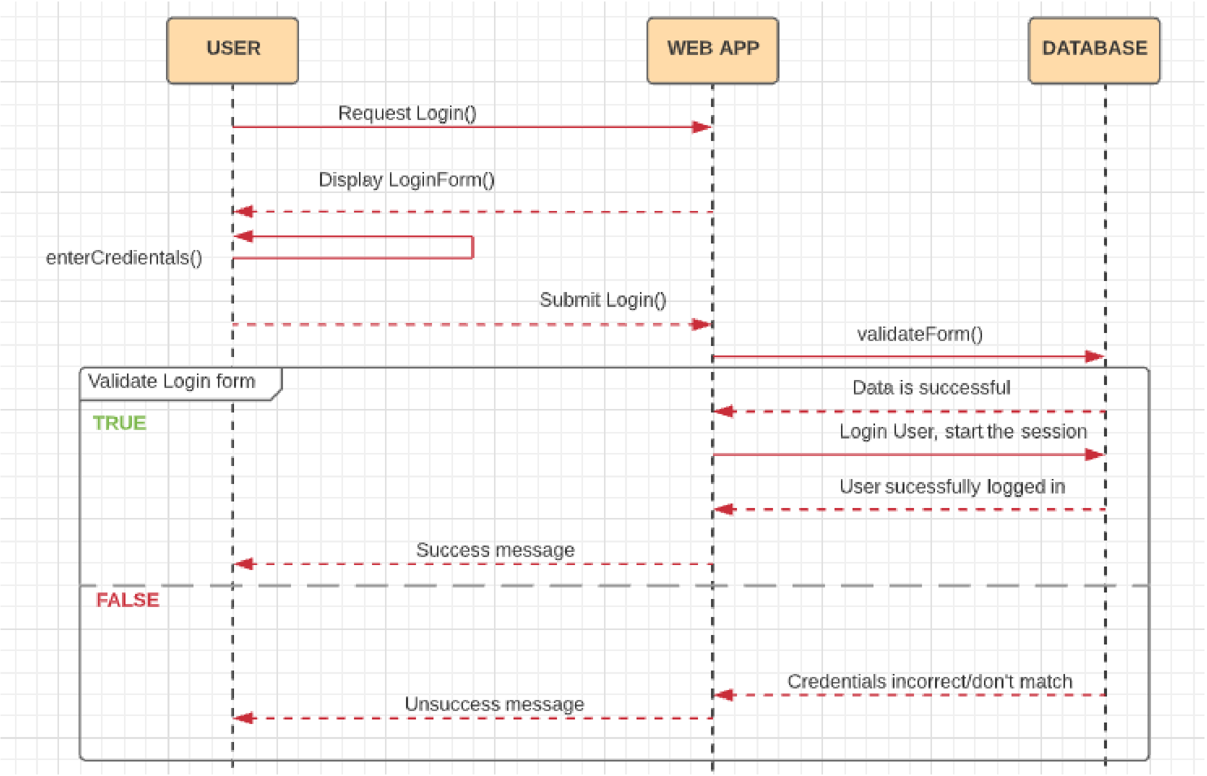


Figure 36: Login

General use sequence diagram

Users will use this spam filter tool to view emails safely. The process below shows how this application maybe used by a user.

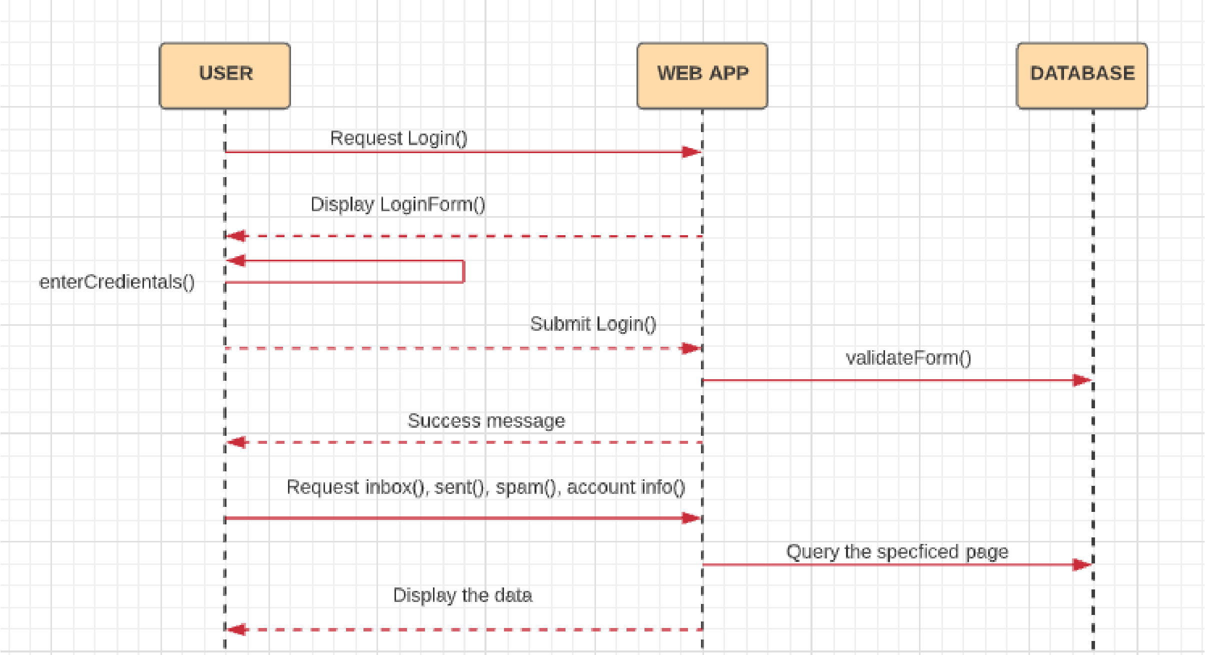


Figure 37: General Use

Display Spam Report Sequence Diagram

This diagram displays the process involved when the administrator requests spam reports.

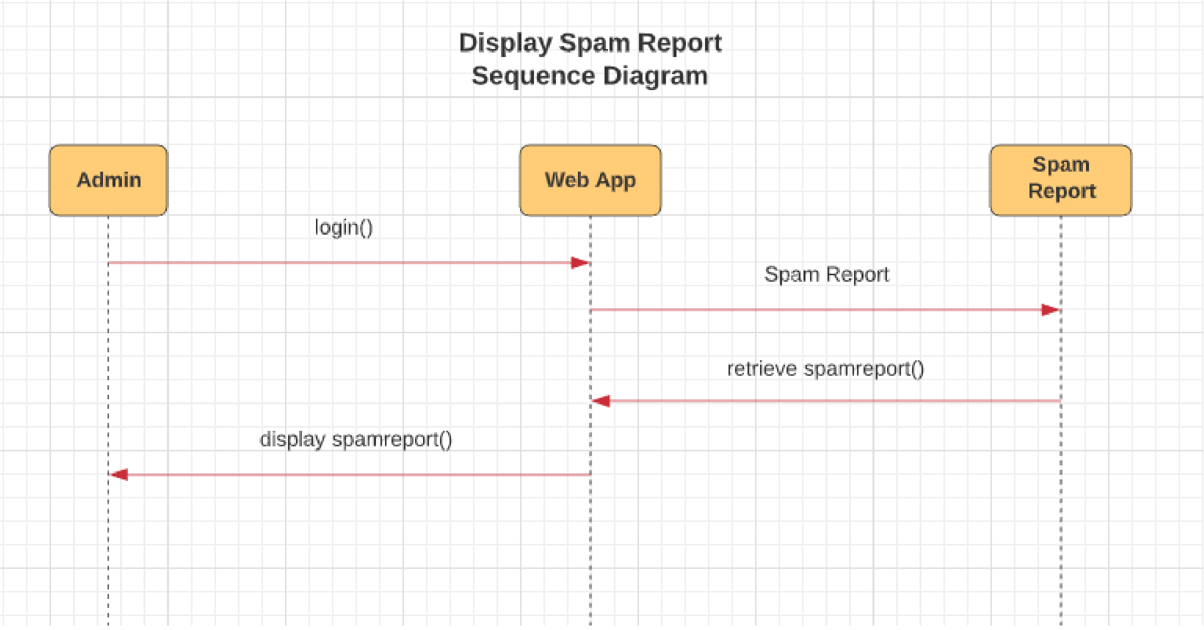


Figure 38: Spam report

Scoring Spam Sequence Diagram

This diagram shows the process which takes place when scoring emails as spam or ham. This process occurs every 15 seconds.

1) The scheduler will check the check table in the database for a list of email IDs.
2) The scheduler will then check the mail sever for emails that are not in the list.
3) Emails which are not in the list are then scored as spam or ham.
4) The result for this email is stored in the database.

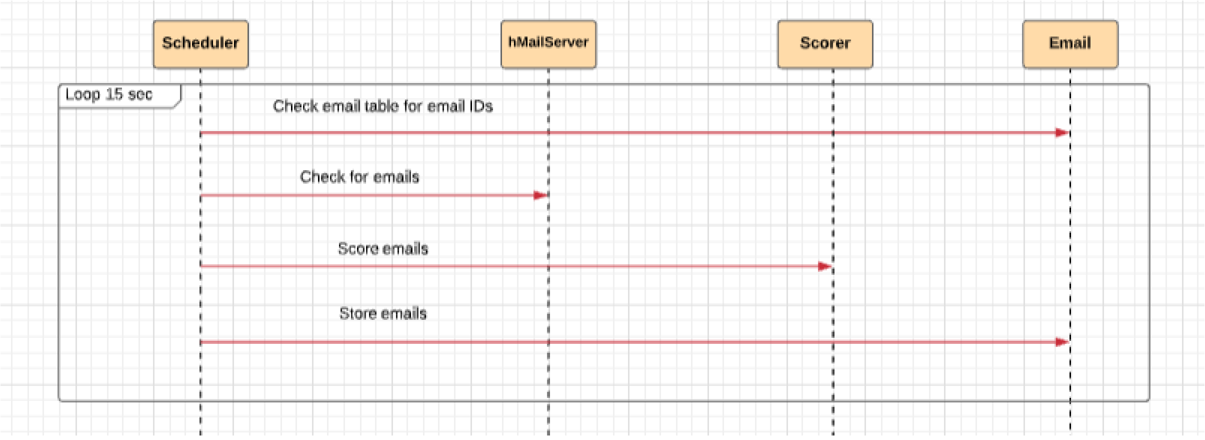


Figure 39: Score emails

Logout Sequence Diagram

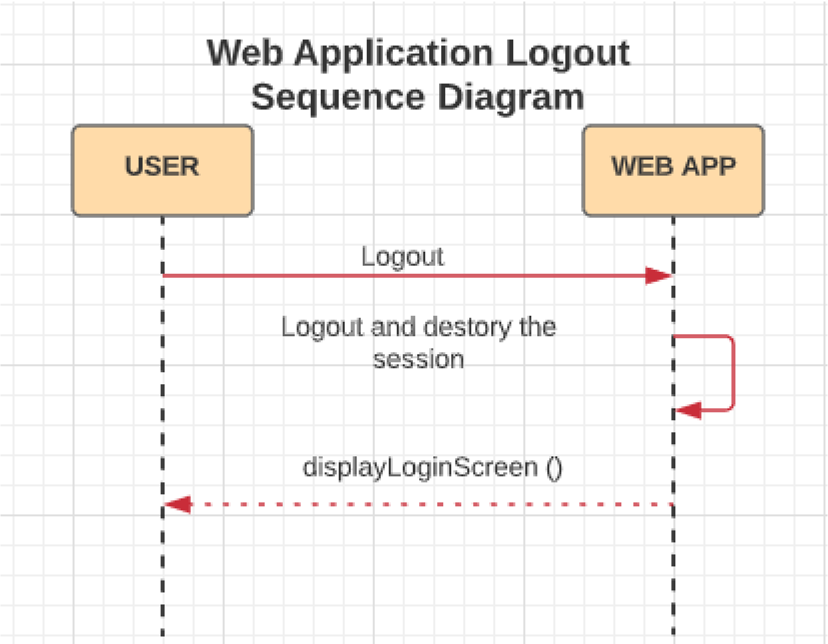

Web Application Logout Sequence Diagram
USER
WEB APP
Logout
Logout and destory the session
displayLoginScreen ()

Figure 40: Logout

Data encrypted and stored in the database

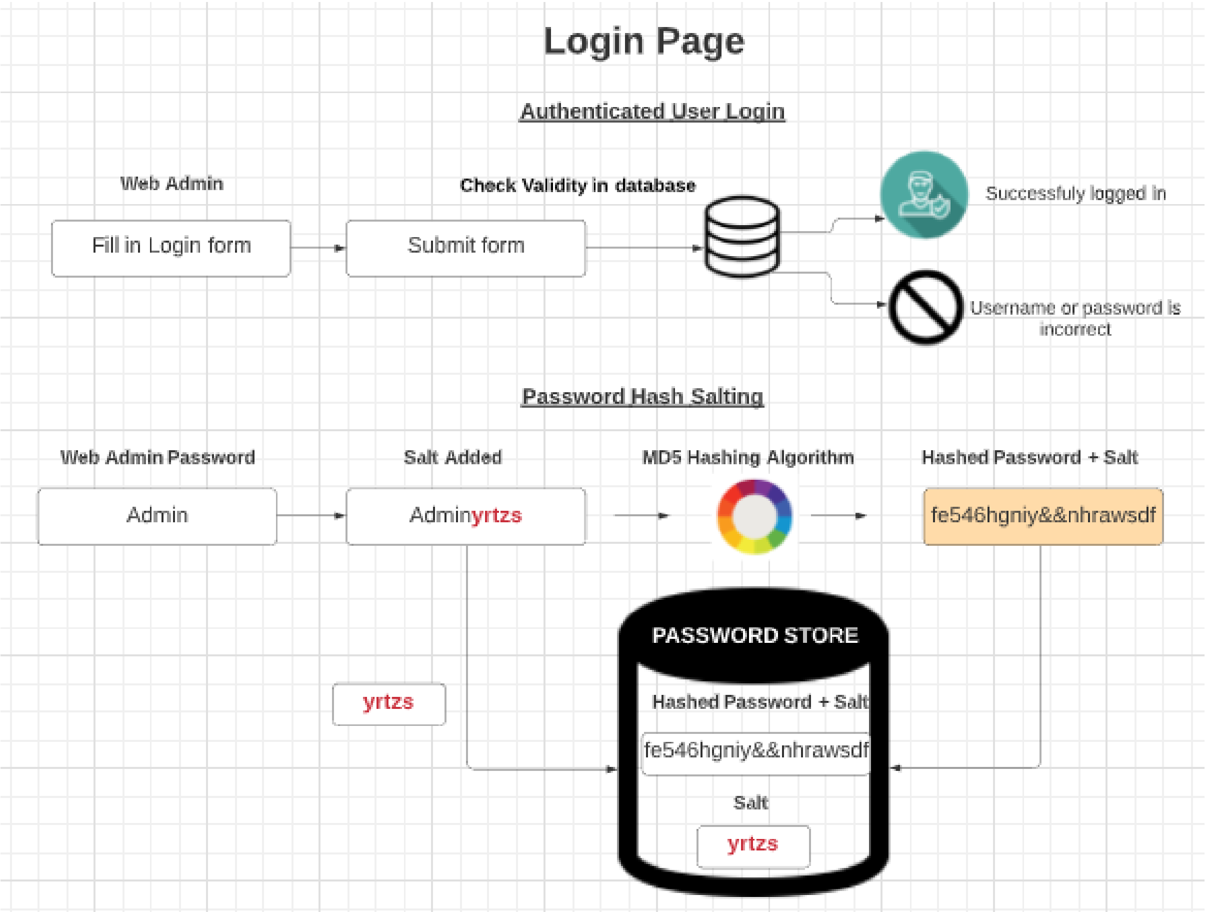


Figure 41: Data encrypted

(Unified Modeling Language (UML) | Sequence Diagrams - GeeksforGeeks, 2020)

Background Functionality

Pre-setup

1. Firstly, I had to research the best fit dataset to use to train and test the machine learning models. I decided to use the following: https://www.kaggle.com/uciml/sms-spam-collection-dataset

2. This dataset needs to be split. I need a training set and testing set. The first 4457 entries will be used as the training set and the remaining entries will be used as the testing set.
3. The four machine learning modules will be trained using the training dataset. The four models are Naïve Bayes, SVM, Random Forest, Logistic Regression.
4. Once these models were trained, they were added to pickle files for quicker access to the models.

Execute scrape function

The scrape function is setup as a periodic task and executes on schedule every 15 seconds.

1. The task scans through the hMailServer directories to try find emails which have not been scored and stored in the MySQL database already.
2. The email data is then transformed to be stored in the correct format in the database.
3. The four machine learning models are then called by the function to score each email as spam or not spam. A score of 1 means the email is spam and a score of 0 means the email has been classified as not spam.
4. The score of the emails is then added.
5. All data is then stored in the database.

Bibliography

Wrike.com. 2017. *4 Tips for an Effective Project Management Plan*. [online] Available at: <https://www.wrike.com/blog/4-tips-effective-project-management-plan/> [Accessed 1 December 2020].

Knafve, M., 2020. *Hmailserver - Free Open Source Email Server For Microsoft Windows*. [online] Hmailserver.com. Available at: <https://www.hmailserver.com/> [Accessed 10 November 2020].

Knafve, M., 2020. *Connect To Mysql - Hmailserver - Free Open Source Email Server For Microsoft Windows*. [online] Hmailserver.com. Available at: <https://www.hmailserver.com/documentation/v5.4/?page=howto_connect_to_mysql> [Accessed 10 November 2020].

Knafve, M., 2020. *Set Up Local / Stand-Alone Server - Hmailserver - Free Open Source Email Server For Microsoft Windows*. [online] Hmailserver.com. Available at: <https://www.hmailserver.com/documentation/v5.6/?page=howto_set_up_local> [Accessed 10 November 2020].

James.apache.org. 2020. *Apache James Server 3.0 - Apache James Server 3 - Installation*. [online] Available at: <https://james.apache.org/server/3/install.html> [Accessed 10 November 2020].

Mysql.com. 2020. *Mysql :: Mysql Community Edition*. [online] Available at: <https://www.mysql.com/products/community/> [Accessed 10 November 2020].

Scikit-learn.org. 2020. *Scikit-Learn: Machine Learning In Python — Scikit-Learn 0.23.2 Documentation*. [online] Available at: <https://scikit-learn.org/stable/> [Accessed 10 November 2020].

Medium. 2020. *Top Python Libraries: Numpy & Pandas*. [online] Available at: <https://towardsdatascience.com/top-python-libraries-numpy-pandas-8299b567d955> [Accessed 10 November 2020].

Medium. 2020. *Spam Classifier In Python From Scratch*. [online] Available at: <https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73> [Accessed 10 November 2020].

Flask.palletsprojects.com. 2020. *Foreword — Flask Documentation (1.1.X)*. [online] Available at: <https://flask.palletsprojects.com/en/1.1.x/foreword/> [Accessed 10 November 2020].

Fullstackpython.com. 2020. *Object-Relational Mappers (Orms)*. [online] Available at: <https://www.fullstackpython.com/object-relational-mappers-orms.html> [Accessed 10 November 2020].

Bhatia, R., 2020. *Why Do Data Scientists Prefer Python Over Java?*. [online] Analytics India Magazine. Available at: <https://analyticsindiamag.com/why-do-data-scientists-prefer-python-over-java/#:~:text=In%20terms%20of%20toolset%2C%20Java,of%20concurrency%2C%20Java%20beats%20Python.> [Accessed 10 November 2020].

Guru99.com. 2021. ER Diagram: Entity Relationship Diagram Model | DBMS Example. [online] Available at: <https://www.guru99.com/er-diagram-tutorial-dbms.html> [Accessed 1 December 2020].

The Easy Guide to UML Class Diagrams | Class Diagram Tutorial. [Blog] Available at: <https://creately.com/blog/diagrams/class-diagram-tutorial/> [Accessed 2 December 2020].

3 rules for a good user experience (UX). [Blog] Available at: <https://www.zdnet.com/article/3-rules-for-a-good-user-experience-ux/> [Accessed 5 December 2020].

GeeksforGeeks. 2020. Unified Modeling Language (UML) | Sequence Diagrams - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/#:~:text=A%20sequence%20diagram%20simply%20depicts,objects%20in%20a%20system%20function.> [Accessed 7 December 2020].