



## CS 521 – Project Implementation

Arabic SMS Spam Filtering Using ML with NLP Text Classification Methods

By

Name	ID
Omar Alsantali	2190005929
Khalid Alsantali	2190002994
Badr Alkhaldi	2190000220
Bandar Almutairi	2190001857
Sakher Alroque	2190005038

Year 4 of the Computer Science Program

Term 3 – Academic Year: 2022/23

**Supervised by**

Dr. Irfan Ullah Khan

## **DECLARATION**

We hereby declare that this project report entitled “Arabic SMS Spam Filtering Using ML with NLP Text Classification Methods” is based on our original work except for citations and quotations, which have been duly acknowledged. We also declare that it has not been previously or concurrently submitted for any other degree or award at any university or any other institution. This project work is submitted in the partial fulfillment of the requirements for the degree of “Bachelor of Science in Computer Science” at the Computer Science Department, College of Computer Sciences and Information Technology, Imam Abdulrahman Bin Faisal University.

# Table of Contents

DECLARATION .....	2
Abstract.....	8
Chapter 1 - Introduction.....	9
1.1 Introduction .....	9
1.2 Problem Statement.....	9
1.3 Dataset.....	10
1.4 Aims & Objectives .....	10
1.5 Scope / Limitation of the Study .....	11
1.6 Project Organization .....	12
1.6.1 Communications: .....	12
1.6.2 Team Meetings: .....	12
1.7 Definitions, Acronyms, and Abbreviations .....	12
Chapter 2 – Background and Literature Review .....	14
2.1 Introduction .....	14
2.2 Algorithm Review .....	14
2.3 Literature Review .....	14
2.4 Conclusion.....	16
Chapter 3 – Proposed Methodology .....	17
3.1 Introduction .....	17
3.2 Proposed Solution.....	17
3.3 System Analysis .....	18
3.3.1 Product Perspective.....	18
3.3.2 Functional Requirements.....	19
3.3.3 Non-functional Requirements.....	19
3.3.4 Operating Environment .....	20
3.3.5 Constraints .....	20
3.3.6 Assumptions .....	20
Chapter 4 – Software Design Specifications.....	21
4.1 Introduction .....	21
4.2 System Description.....	21

4.2.1 System Architecture .....	21
4.2.2 System Users .....	21
4.2.3 System Flowchart.....	22
4.3 Design Considerations .....	22
4.3.1 Used Software & Hardware .....	22
4.3.2 End-user Expectation .....	23
4.4 General Constraints.....	23
4.4.1 Hardware & Software Environments .....	23
4.4.2 End-user Environment.....	23
4.4.3 Standard Compliance .....	24
4.4.4 Interoperability Requirements .....	24
4.4.5 Interface Requirements.....	24
4.4.6 Performance Requirements.....	24
4.4.7 Verification and Validation Requirements .....	24
4.5 User Interface .....	25
Chapter 5 – Takeaways.....	28
Chapter 6 - Software Test Plan .....	29
6.1 Objectives.....	29
6.2 Testing Strategy .....	29
6.3 Scope.....	30
6.4 Test Items .....	30
6.4.1 Program Modules.....	30
6.4.2 User Procedures .....	32
6.5 Features to Be Tested.....	32
6.6 FEATURES Not to Be Tested.....	33
6.7 APPROACH.....	33
6.7.1 Component Testing .....	33
6.7.2 Integration Testing .....	33
6.7.3 Conversion Testing .....	34
6.7.4 Job Stream Testing.....	34
6.7.5 Interface Testing .....	34

6.7.6 Security Testing .....	34
6.7.7 Recovery Testing .....	34
6.7.8 Performance Testing .....	34
6.7.9 Regression Testing .....	35
6.7.10 Acceptance Testing .....	35
6.7.11 Beta Testing .....	35
6.8 Pass / Fail Criteria .....	35
6.8.1 Suspension Criteria .....	35
6.8.2 Resumption Criteria .....	36
6.8.3 Approval Criteria .....	36
6.9 Testing Process .....	36
6.9.1 Test Deliverables .....	36
6.9.2 Testing Tasks .....	37
6.9.3 Responsibilities .....	37
6.9.4 Resources .....	37
6.9.5 Schedule .....	38
6.10 Environmental Requirements .....	38
6.10.1 Hardware .....	39
6.10.2 Software .....	39
6.10.3 Security .....	39
6.10.4 Tools .....	40
6.10.5 Publications .....	40
6.10.6 Risks and Assumptions .....	40
6.11 Change Management Procedures .....	40
6.12 Plan Approvals .....	41
Chapter 7 – Implementation Details .....	41
7.1 Data Collection & Preprocessing .....	41
7.2 Algorithm .....	42
7.3 Android Application .....	42
7.4 Results & Benchmarking .....	43
Chapter 8 – User Manual .....	44
8.1 Recommended Hardware Requirements .....	44
8.1.1 Processor .....	44

8.1.2 Memory .....	44
8.1.3 Storage Space.....	44
8.2 Screen Items .....	45
8.3 Application Installation steps .....	46
Chapter 9 – Conclusion .....	49
9.1 Findings & Contributions.....	49
9.2 Limitations .....	49
9.3 Lessons & Skills Learned.....	49
9.4 Recommendations for future works.....	50
References.....	51
Appendix .....	53
Code snippet:.....	53

## List of Figures

Figure 1 - Communication Structure .....	12
Figure 2 - Product Perspective .....	18
Figure 3 - System Flowchart.....	22
Figure 4 - Main Page .....	25
Figure 5 - Spam Page .....	26
Figure 6 - Chatting Page .....	27
Figure 7 - System Design.....	31
Figure 8 - First Step.....	47
Figure 9 - Second Step.....	47
Figure 10 - Third Step .....	47
Figure 11 - Fourth Step.....	47
Figure 12 - Fifth Step .....	48
Figure 13 - Sixth Step.....	48

## **List of Tables**

Table 1 - Definitions.....	13
Table 2 - Hardware Requirements.....	22
Table 3 - Software Requirements.....	23
Table 4 - Modules .....	32
Table 5 - Tested Features.....	33
Table 6 - Types of tests.....	36
Table 7 - Table of Responsibilities.....	37
Table 8 - Schedule.....	38
Table 9 - Recommended Hardware .....	39

## Abstract

This document is a culmination of the work spanning an entire semester year. Included in it are the project proposal, SPMP, SRS, SDS, STP, and user manual. We set out to accomplish a very critical piece of research in the field of text classification that simultaneously was under-researched and had wide implications towards users' needs. SMS spam classification is a widely used and solved problem, but very little has been done to further that solution in Arabic text. We proposed a Naïve Bayes based model with an accompanying Android application that would implement and actually effectively utilize the model. We have accomplished that as discussed in this paper. We implemented carefully considered preprocessing steps and overcame the limitations we discussed in the proposal below. Limitations such as a large research gap and limited available datasets. We collected a dataset ourselves and completed important research. In the end, we managed to produce a text classification algorithm with 96% accuracy, 97% precision, and 90% recall. Such a model and application can greatly reduce the risk of harm coming to the lay person using their phone and receiving harmful messages.



# Chapter 1 - Introduction

## 1.1 Introduction

The SMS (Short Messaging Service) messaging functionality comes as standard in every modern smartphone. Due to the massive popularity of such technology, it is natural that it should suffer from fraudulent and deceitful users exploiting it. According to [1], there were 10.89 billion spam texts in the US in August 2022. Such schemes come in many different forms, including spam messages that attempt to get the receiver to divulge sensitive and financial details. In this project, we aim to develop an AI algorithm that leverages ML and uses NLP text classification to categorize Arabic SMS messages based on their contents.

A common approach for text classification is the Naïve Bayes algorithm, as proposed by previous research [2], based on Bayes' rule. As a start, that is the algorithm that this project will be utilizing. It is a great algorithm for limited datasets [3], considering the low amount of Arabic SMS messages data available online. In this case, it works by recording independent words in each message, spam or legitimate, and calculating the probabilities of these words occurring in each type. This is only a very brief description of how it works. More will be covered later on.

The practical outcome of this project will be perceived by a proof-of-concept Android application. An iOS application is not feasible in this case due to API restrictions when developing on non-iOS devices. The application will solely serve as a vehicle to showcase the success of the algorithm by only displaying legitimate messages and storing spam messages in a separate section.

## 1.2 Problem Statement

SMS Text Messaging is a widely used and essential feature of all mobile phones. Many service providers offer free unlimited messaging through SMS as part of their plans, and so users of all sections of society are subject to SMS messages. This also means a wide range of tech literacy and as such users with a low degree of technical understanding can be faced with messages that exploit that misunderstanding.

According to the Pew Research Center, 79% of survey respondents indicated that they used text messaging on their phones, and of that 79%, 69% say they get unwanted spam or text messages [4]. That is a massive number of people that are receiving messages that could be very harmful. In March of 2022, spam text messages tracked by RoboKiller totaled 11 billion [5]. This massive number of messages means that even if a tiny percentage of users fall victim to the scam, it's still a huge amount of money stolen in general. Between December 23-31 of 2021, 790 clients of OCBC Bank in Singapore fell victim to an SMS phishing scam, with losses totaling up to about \$10.1 million USD. That's an average \$12,800 USD per victim [6].

This problem affects a huge number of people and not just the elderly either. According to research from TSB Bank. Three quarters of people struggled to identify common scams. With the highest percentage of people who would respond to at least one message pretending to be their bank being 18-34 year-olds, with 41% of them falling victim to a theoretical scam message [7].

This is why research on Arabic SMS spam use is of paramount importance. Very little research exists and service providers do little to combat the rising tide of scams. Our project aims to alleviate this problem by creating an algorithm that would detect spam messages and stop users from falling victim to these scams.

### 1.3 Dataset

Scavenging online for a dataset that contained Arabic SMS messages proved to be difficult. There is next to none, as we have only found 40 separate Arabic SMS messages available for use. None of the major dataset sources (Kaggle, huggingface), nor smaller ones contain the desired dataset. Moreover, research that has previously used Arabic SMS datasets does not publicly share them. Therefore, we will be manually collecting the dataset through local smartphones (friends, family & relatives), along with the aforementioned dataset. However, we will continue to look for any data that we can come by online and could be of use.

### 1.4 Aims & Objectives

Our aim is to develop a proof-of-concept Android application that showcases a filtered SMS message feed of strictly messages that are relevant to the user. The app will hide all spam messages in a separate section. The criteria in which the app will determine a message is spam or not will be through the to-be-trained ML model that we will conceive using the chosen algorithm (Naïve Bayes classifier initially), which will take the SMS message that is received from the smartphone as a pure text input and produce an output of “spam” or “not spam”. Below is a list of the project’s objectives:

- Develop and improve an Arabic spam detection algorithm.
- Reduce the number of victims to such schemes by hopefully avoiding the reach of scammers completely.
- Collect spam messages so that it can be used to analyze any new patterns and techniques the scammers may apply.

### 1.5 Scope / Limitation of the Study

Our project revolves around solving the problem of SMS spam through a special Android messaging app that filters down SMS messages into a feed of legitimate messages. That said, our application focuses primarily on Arabic spam messages. As such, we will have a very restricted Dataset to work with. Our Arabic spam message app has few resources for datasets because it is targeting the Arabic language. Despite Kaggle having so many datasets to pick from, the possibility of finding Arabic datasets is low. Hence, we must gather Arabic spam messages on our own to help increase the amount of data our algorithm can work with.

Furthermore, we will restrict ourselves to Android rather than developing for iOS due API restrictions. Android’s open-source nature will allow us to develop the application with immense flexibility as it relies on Java and enables full customization. On the other hand, iOS requires XCode for native iOS development as well as a Mac to run XCode. This is a problem in both flexibility and cost. [8]

## 1.6 Project Organization

### 1.6.1 Communications:

The team communication is as follows: the team discusses the project and then the team leader communicates our findings and questions to the supervisor Dr. Irfan. A figure is provided below showcasing the team structure in detail.

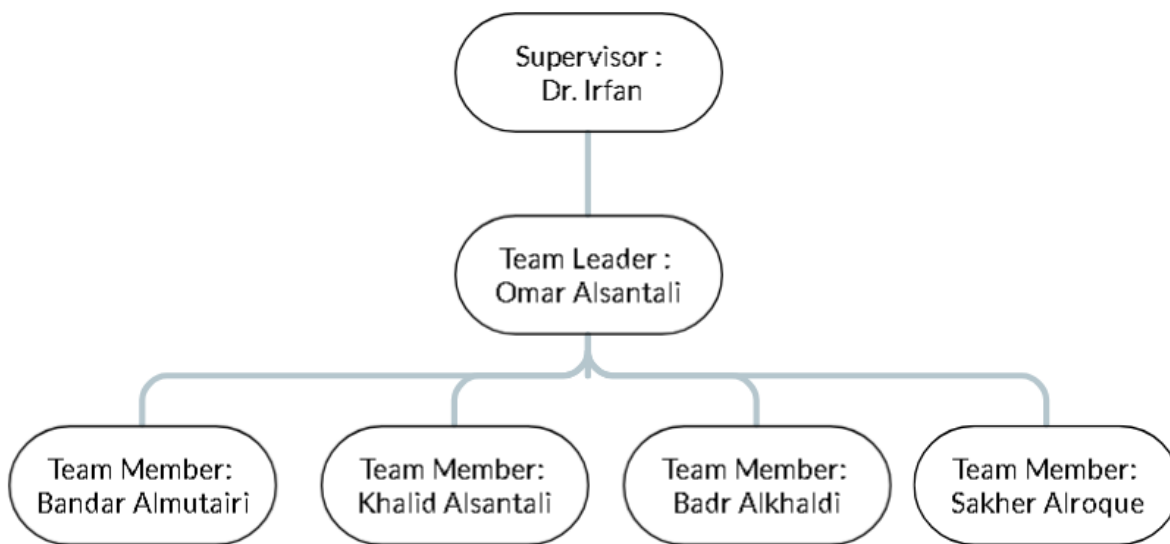


Figure 1 - Communication Structure

### 1.6.2 Team Meetings:

The team meetings are separated into two different categories: weekly meetings that are done through Zoom where we review our work throughout the week and share feedback, and daily communication using WhatsApp where we plan and provide answers for any questions that the team members have.

## 1.7 Definitions, Acronyms, and Abbreviations

Expression	Meaning
------------	---------

<b>SMS</b>	Short Message Service
<b>ML</b>	Machine Learning
<b>NLP</b>	Natural Language Processing
<b>API</b>	Application Programming Interface
<b>CNN</b>	Convolutional Neural Networks
<b>LTSM</b>	Long Short-Term Memory
<b>SVM</b>	Support Vector Machine
<b>KNN</b>	K-Nearest Neighbors

*Table 1 - Definitions*

## Chapter 2 – Background and Literature Review

### 2.1 Introduction

This chapter covers research conducted in the past concerning Arabic SMS spam filtering. It is important to note that the amount of research regarding this is little in contrast to English SMS spam filtering. Therefore, literature regarding both languages will be researched, with Arabic being the main concern, and English as supplementary knowledge of the various algorithms used.

### 2.2 Algorithm Review

As it currently stands, most research and practical application of SMS spam filtering is mainly based on two main algorithms: SVM and the Bayesian network to construct spam classifiers [9]. We see that this is corroborated based on the literature reviewed below. Other algorithms were also devised in this field, such as the hybrid CNN-LTSM model provided in one of the studies [10].

Although some research suggests that some algorithms are better than others, this project, as previously stated, will leverage the Naïve Bayes classifier as a starting point. Moreover, little research has used this algorithm for the Arabic language, which is more reason to support starting out with such an algorithm.

### 2.3 Literature Review

[10] is the most informative piece of literature regarding Arabic SMS spam filtering. The researchers proposed a hybrid deep learning model for detecting spam SMS messages. It is based on a combination of two deep learning models, which are CNN and LTSM. It was devised with the intention of recognizing two languages, Arabic and English. The authors also compared this algorithm to other common methods, of which are SVM, KNN, Multinomial Naïve Bayes, and many others. This study used two separate datasets for each language, the first one being an English dataset from the UCI repository, and the other being one the authors collected using local

smartphones in Saudi Arabia. Unfortunately, the latter is nowhere to be downloaded and used by us in this project. Regarding the authors' conclusion, they found that their proposed CNN-LTSM hybrid algorithm performed the best with a 98.37% accuracy compared to 97.83%, 90%, and 97.83% for SVM, KNN, and Multinomial Naïve Bayes respectively.

[11] is a paper describing a proposed system for filtering SMS messages that uses a Naïve Bayes filter as a first classifier and a Neural Network as a second classifier. The paper details the methodology the authors used. There are two methods used to collect SMS spam messages:

1. Collaboration-based method in which users submit feedback and experience regarding the data. Eliminating guess work or other more time-consuming tasks to determine the type of SMS message.
2. Content-based method which relies on analyzing the textual content of messages. This is more common due to the difficulty of acquiring data related to SMS user experience in general and for Arabic specifically.

Preprocessing is also mentioned as a critical part of the algorithm's preparation. White spaces are removed and stop words are extracted and used as features to help filter SMS messages more efficiently. Six features are described in the paper such as the presence of a phone number, and non-alphanumeric characters, etc.

The results of this system for Arabic filtering had 95% accuracy with six features and 70% of the dataset for training. This paper describes an important part of AI filtering which is preprocessing, and the multiple steps taken to prepare the data for the classifiers.

[12] is another paper that utilized Naïve Bayes filter as a classifier for a proposed Arabic SMS spam filtering system. The researchers adopted a workflow of multiple phases before feeding the NB filter with the SMS messages. The first phase is the preprocessing phase which eliminates white spaces and stop words. The second phase is feature extraction in which the researchers used a total of 15 features, some of which had a higher effect on how NB classified SMS messages. For example, one of the extracted features is the words ratio in a message and it had the most effect on

classifying non-spam messages, while the number of non-alphanumeric characters had the highest effect on classifying spam messages.

After extracting the features normalization will be applied to reduce the variance of values, and then features will be selected before classification to reduce the number of irrelevant and redundant features. Removing such features proved to have an increase in the system performance, the researchers applied TF, IG, and GR selection methods. As for the findings, with a dataset of 400 messages in which 70% were considered for training an accuracy of 83% was reached, and it was raised up to 88% after applying TF with 12 features. And thus, concluding that to achieve the best results it is suggested to use a feature selection method along with NB classifier.

## 2.4 Conclusion

As shown, the amount of research regarding this subject is relatively sparse. It is important to note that SMS spam filtering in general is thoroughly researched and applied though with languages other than Arabic. This is why we initially decided to research this issue, and we do not believe that English-based spam filtering research is pertinent to our specific case, hence our exclusion of it in the review.



## Chapter 3 – Proposed Methodology

### 3.1 Introduction

Arabic SMS Spam filtering, as previously mentioned, has been remedied before by several researchers, each using different methodologies (in this case, different algorithms). We know that a solution is possible to a certain degree of accuracy depending on the approach. The datasets needed, while lacking in online availability, are easily extractable from several local mobile phones. Many instances of Arabic SMS spam are present in most smartphones today, and are fortunately easily identifiable due to their uncommon wording and the anonymity of the sender.

### 3.2 Proposed Solution

The proposed solution for the problem statement is, at its core, an AI algorithm that can receive an Arabic SMS message and classify it in a binary fashion as a SPAM or NORMAL message with a level of confidence. The eventual goal is to achieve as high of a level of confidence as possible.

The machine learning algorithm will leverage the Naïve Bayes classifier for categorizing SMS messages. It is a probabilistic classifier that applies Bayes' theorem along with heavy independence assumptions, meaning previous results are not considered when evaluating the next.

The aforementioned algorithm will exist as a backend that takes the input when an SMS message is received, determines its category, and delivers a verdict to an android application. This mobile application will serve as the forefront that displays the SMS messages by their category. Normal SMS messages (ones that are evaluated as safe/not spam by the algorithm) will be displayed in the main section, while spam messages will be displayed in another.

### 3.3 System Analysis

#### 3.3.1 Product Perspective

Below is a figure illustrating how the different system modules interact with each other. The system is comprised of 4 main parts, 2 of which we are responsible for creating (SMS App and ML Model). The user enters the SMS application, after which the app fetches the current existing SMS messages (both spam and not-spam) from the smartphone's local database, then sends each message to be evaluated as spam or not by the ML model using an API. The ML model sends back an evaluation of whether the sent message is spam or not, along with a degree of accuracy/confidence.

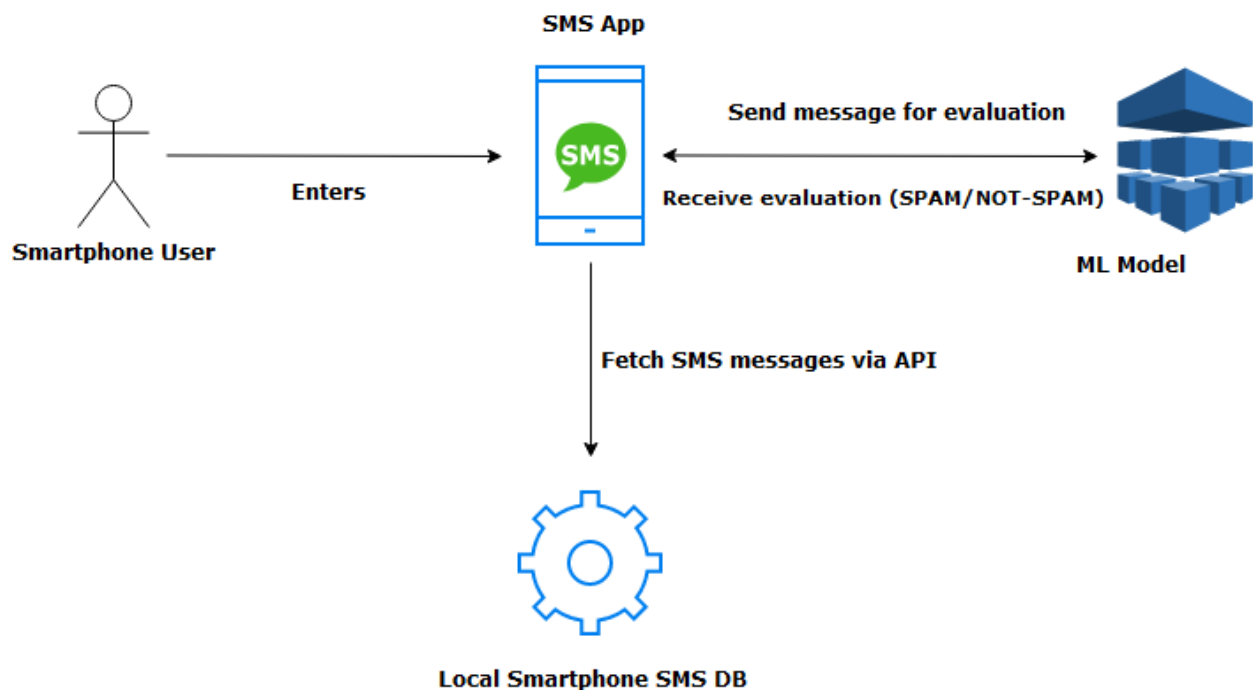


Figure 2 - Product Perspective

### 3.3.2 Functional Requirements

The functional requirements for this project are going to include both the ML algorithm and the application presenting it.

- **The application**
  - a. The user is able to view all SMS messages received.
  - b. The user can view two categories of SMS messages, chat and spam, at any time.
  - c. The application receives the spam confidence level from the algorithm and places the message in the correct category.
- **The ML algorithm**
  - d. The algorithm must generate a result that a message is either spam or not spam with a confidence level.

### 3.3.3 Non-functional Requirements

The non-functional requirements apply to both the algorithm and the application.

- **Accuracy:** Important messages can be caught by the spam filter mistakenly. So the algorithm and the application must be highly accurate to avoid those mistakes. An algorithm accuracy above 90% is important and the application should have a failsafe mechanism to categorize ambiguous messages as non-spam.
- **Usability:** The application and algorithm must be easy to understand and use so users can fully benefit from the features.
- **Speed:** The speed of the algorithm must be unnoticeable to the user. Such that if they're waiting for a message to come in they wouldn't have an impression the application and algorithm are making things more difficult to use.

- **Localization:** The algorithm should only be used on Arabic language SMS messages and so the application will only categorize Arabic messages into either spam or not spam.

### 3.3.4 Operating Environment

The algorithm will be written in Python and the application will run on Android.

### 3.3.5 Constraints

- **Dataset:** The biggest constraint of this project. Lack of ready, high-quality datasets of Arabic language spam is a constraint that we must keep in mind as we collect data and analyze it.
- **Language:** Arabic is the only language support by the algorithm, but the application will be able to display English and Arabic.
- **Tools:** Python with NLP and text classification libraries alongside Numpy.
- **Time:** The length of the Project Implementation course is 11 weeks.

### 3.3.6 Assumptions

Our goal is to create an application which serves as a front-end for a machine learning text classification algorithm that classifies Arabic SMS messages into spam and non-spam categories. The application will only function as a way to view the SMS messages and will not have wide functionality to serve as a replacement for the default messaging app.

When the device receives a text message, the application will read it and analyze it through the algorithm and decide whether it's spam or not spam and appropriately place it in the correct category.

## Chapter 4 – Software Design Specifications

### 4.1 Introduction

Software design specifications is a document that contains a detailed plan for developing a piece of software. In this chapter, we will discuss all the design aspects of our project. Starting with specifying the system's layout and architecture and providing some of the constraints and considerations in terms of design. And finishing off with a prototype representation of the final project's user interface.

### 4.2 System Description

#### 4.2.1 System Architecture

Our system will utilize the Model-view-controller architectural pattern commonly known as MVC. The main idea of this architecture is that the user can see and interact through the view component which sends prompts to the controller, in which the controller updates the model, and the model sends any changes to the view.

#### 4.2.2 System Users

Our system will be implemented into android mobile devices, and the end user will be able to use the system to identify spam messages stored in the device by filtering it through the ML model.

### 4.2.3 System Flowchart

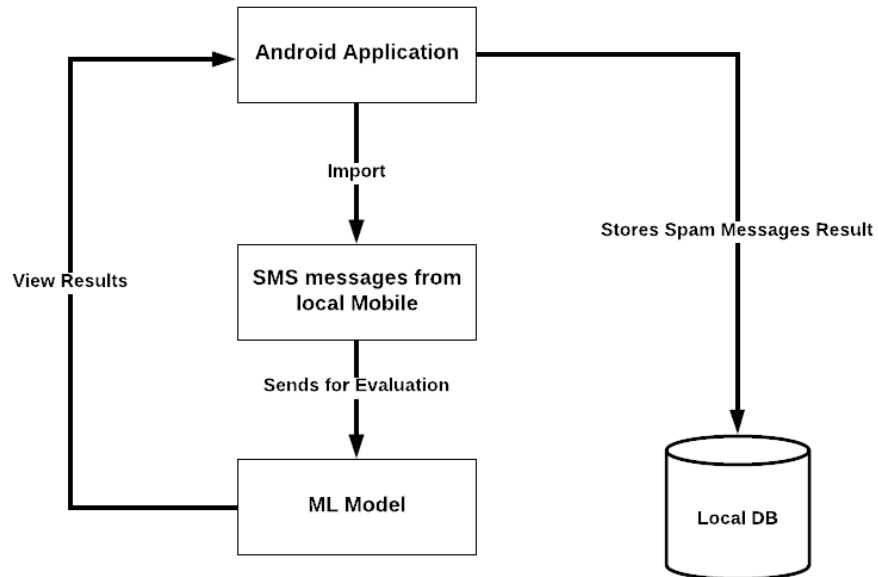


Figure 3 - System Flowchart

## 4.3 Design Considerations

In this section we specified the main considerations required from us to be able to build this application, both hardware and software.

### 4.3.1 Used Software & Hardware

Hardware requirements:

Hardware	Requirement
RAM	A minimum of 4GB or Higher
CPU	i5 processor or Higher

Table 2 - Hardware Requirements

Software requirements: for the operating system we will be using windows 10 and below are the software tools that we will be using:

Software	Version	Description
Python	3.9.13	Python is an interpreted, object-oriented, high-level programming language
Numpy	1.23.4	a Python library used for working with arrays.

*Table 3 - Software Requirements*

#### 4.3.2 End-user Expectation

ASMS will provide an app that will solve the problem the end-users face when dealing with spam SMS messages.

### 4.4 General Constraints

This section will focus on every aspect that limits or constraints our application from performing its intended purpose or slows down our development progress.

#### 4.4.1 Hardware & Software Environments

We currently are restricting our application to Android devices as the OS is open source, allowing for ease of development. iOS devices on the other hand requires a Mac device to develop applications and to program them through XCode (Apple's IDE).

#### 4.4.2 End-user Environment

Android's nature as an open-source device means that it can operate on a dozen different devices with their own specs and limitations. This would be problematic if we wish to release the application as it would require constant checking on the compatibility with various android phones.

#### 4.4.3 Standard Compliance

The application will comply with the standards set forth by the university to ensure that the application is of high-quality, will be secure to use, and is effective at its intended objective.

#### 4.4.4 Interoperability Requirements

As mentioned earlier, it would be immensely difficult to try and make our application function for both Android and iOS. The former's open-source nature makes it easy to develop on, but it also winds up making it challenging to maintain compatibility on various devices with differing versions of Android. And the latter has its unique problem of requiring special software and hardware to develop or port applications for.

#### 4.4.5 Interface Requirements

The application's Machine Learning algorithm would require messages from the phone's SMS database and a dataset that possesses spam message samples to run its analysis with.

#### 4.4.6 Performance Requirements

The application is lightweight, so running it won't cause any slowdowns or require bigger RAM.

#### 4.4.7 Verification and Validation Requirements

Our application will run through various tests to ensure that it performs within expected accuracy and causes no issues that may lead to an unintended result. That way, we can check if it meets all our needed requirements.



## 4.5 User Interface

User interface design is the procedure designers use to build customized interfaces in software, every design should always cater to the simple and straightforward approach so the normal everyday user can use the app easily and intuitively without confusion.

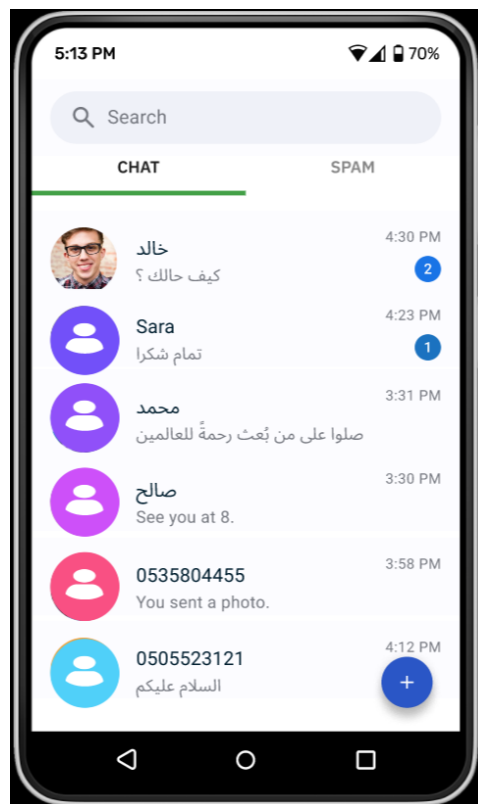


Figure 4 - Main Page

As we can see from the figure above this is a simple straightforward main page, in this main page there is the short display of the latest messages, also you can create a new message or if the user wants to find a specific message the search function can be used.



Figure 5 - Spam Page

As we can see from the figure above this the spam page, this page usage is so the user can differentiate normal messages from suspicious messages, this can be done by using Artificial intelligence that detects any suspicious messages it gets transferred to this section.



Figure 6 - Chatting Page

This is the chatting page, where the user can see the conversation between them and the other number, this page has the functionality of displaying and writing messages.

## Chapter 5 – Takeaways

In conclusion, spam SMS messages are incredibly dangerous and harmful. The studies we have cited have shown how much these scam messages can cost. We plan to tackle this problem through a machine learning algorithm that can help users to determine whether or not to trust SMS text messages. We discussed the limitations of our project such as data set procurement difficulties and lack of prior research on this subject.

Our plans to solve the issue of spam begin in our study and research of prior attempts and current technologies. We conducted a literature review where we gathered pertinent information such as the prevalence of the use of Naïve Bayes Classifiers for classifying text into categories. We also gathered valuable information in regard to preprocessing the data so it can be used efficiently by machine learning algorithms.

We further discussed our proposed methodology by further elaborating on the specific requirements and goals of this project which can be summarized as an Android application acting as a front-end for a Machine Learning algorithm.

## Chapter 6 - Software Test Plan

### 6.1 Objectives

The test plan for ASMS is comprised of many objectives, from the initialization of the testing process to finalization of and implementation of the various tests for the application's features. The following list specifically defines each overarching objective of this plan:

- Define what purpose the testing plan will serve, as well as the scope of the plan.
- Make clear which items, features and components of the system are to be tested.
- Divide the decided-on tasks to team members.
- Allocate a schedule to be strictly followed by all members contributing to the testing plan.
- Provide information resources to consult during the testing plan.
- Set a strict and consistent testing environment in which all tests must be performed on.
- Define all the tools necessary to conduct the testing plan.

### 6.2 Testing Strategy

The testing strategy for this project will be one of segmentation. Components of the system will be tested separately and will have defined criteria for their success or failure. Moreover, each test needs to have an explanation on why it is being tested and what the most important things to look out for in that testing case are.

A pass/fail system is also implemented and is a core part of our testing strategy. The importance of this module is paramount, due to it being the cornerstone of how we accept and eliminate components of our system. For each component, we define certain criteria that the components, after being tested, will be judged based on. Entire components may either be added, removed, or completely revised based on the result of this module.

Lastly, we address the more managerial concerns. The most prominent example of which is our management approach. How we will confront the testing plan is of utmost importance. Moreover, we will bring up the division of roles and responsibilities, as well as more management-related concerns such as scheduling, milestone deliveries and associated risks.

## 6.3 Scope

The scope of this testing plan covers almost the entirety of the project. Everything from user interactions to backend functionality is tested. Security, network, and authentication tests are all included.

## 6.4 Test Items

These are the documents that will be referenced by the testing procedures to ensure predefined specifications are met and no feature is missing/not working as intended.

- Project Proposal Document.
- Software Design Specifications (SDS).

The documents above were previously shared with the intended recipients (this project's evaluators).

### 6.4.1 Program Modules

Here, we will define each major module to be scrutinized by the testing procedures to ensure that it is working properly. No testing strategy, procedure, or criteria will be mentioned here. This section is specifically to list the program's modules and what their uses are.

As reference, below is the overarching design of the system.

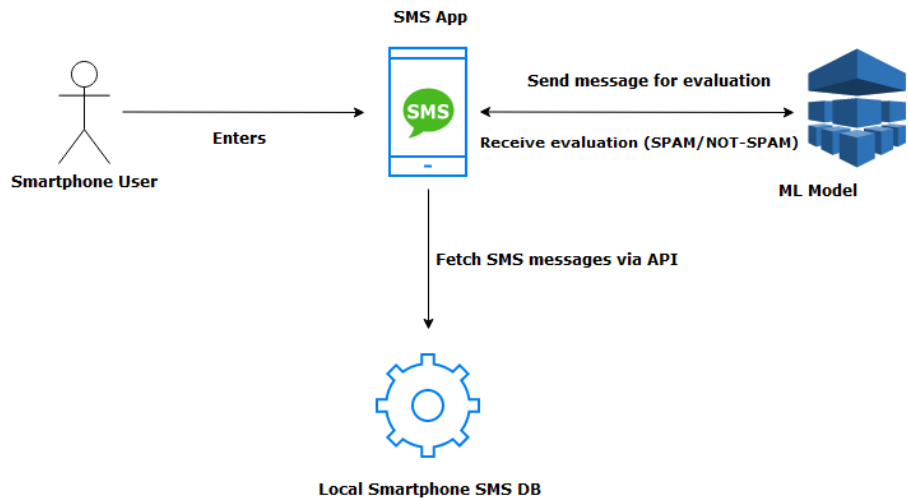


Figure 7 - System Design

The two main modules of the system from the above figure which are to be tested are the SMS application and the ML model. The bulk of the testing is going to be performed on the application, as it is prone to minuscule errors that could potentially be fatal to the working of the program.

The following are the individual modules and what purpose they serve. To note, these aren't all the system's modules. These represent the major modules which are to be tested.

Module	Functionality
SMS retrieval	Fetching the SMS messages for the smartphone's local DB and populating the application.

<b>Message display</b>	Displaying the fetched SMS messages in either the 'safe' section or the 'spam' section.
<b>Message categorization</b>	Categorizing each SMS message as spam or not. The classification is done by the ML module, but this refers to their UI placement.
<b>Model communication</b>	Communicating with the ML model to send an SMS message and receive a categorization of that message (spam or not)

*Table 4 - Modules*

#### 6.4.2 User Procedures

As the scope of this project is research-based and the application will not be published for commercial use, the user-base is comprised of the evaluators and ourselves. Therefore, user procedures are irrelevant in this case.

#### 6.5 Features to Be Tested

<b>Model Accuracy</b>	<b>Model Speed</b>
The AI Model's accuracy in determining whether a message is spam or not. Tests will be conducted using testing data collected which includes spam and non-spam SMS messages.	The AI model is designed to be ran on mobile devices and therefore speed and efficiency are important. Tests will be conducted on a variety of models to determine the most suitable one for mobile device performance.
<b>SMS Message Display</b>	<b>Spam Alert Functionality</b>
The accompanying application will display SMS messages received by the SIM card. Tests will be conducted to determine if the display functionality is error free.	SMS messages which are detected as spam will be tagged as such and shown on the application as an accompanying icon/alert to the suspected SMS message



Message Retrieval
Tests regarding the ability of the application to retrieve SMS messages from the phone's internal storage systems for storing SMS messages. Message decoding functionality should be working correctly.

*Table 5 - Tested Features*

## 6.6 FEATURES Not to Be Tested

The following features will not be tested:

- Application responsiveness: The application will not be released to the public for general use as this is a research project. So application responsiveness is not an important metric

## 6.7 APPROACH

The overall approaches to testing will prioritize both speed and efficiency to ensure the software will function in a real mobile device environment, as well as accuracy to reduce the number of possible errors. In the case of our tests, we shall implement dummy data to guarantee privacy.

### 6.7.1 Component Testing

We test the accuracy of our AI model's spam detection by utilizing data containing both spam and non-spam messages to ensure that it can differentiate between the two types of messages without error.

### 6.7.2 Integration Testing

To ensure that our software can run on mobile devices without slowdowns in performance, we will test out different models to pinpoint which is the best fitting for mobile architecture.

For retrieving SMS messages from the mobile device's internal storage system, we perform tests to guarantee that our device can indeed take the messages from the device's internal storage with no roadblocks by creating automated tests for the API.

#### 6.7.3 Conversion Testing

To ensure our program's message decoding capability, we will test that it can convert the SMS messages into readable data for our model to utilize.

#### 6.7.4 Job Stream Testing

The result of the previous spam detection will make it more accurate to determine if the next message inputted into the model is spam or not.

#### 6.7.5 Interface Testing

Our program must interact through the API to the phone device's storage to gather SMS messages for our model to run on. As such we need to make automated calls to the API to speed up the process.

#### 6.7.6 Security Testing

To ensure that our program will not leak SMS messages and compromise the end-user's privacy, tests will be performed to make sure the output data is scrubbed from the model's database after spam detection.

#### 6.7.7 Recovery Testing

The program will have auto backup to make sure messages can be retained in the case of data corruption. We shall test it out by verifying that the program can backup data and restore it.

#### 6.7.8 Performance Testing

One of our goals with this software is performance as we want the model to detect spam messages as seamlessly and as fast as possible. To test that out, with our Integration testing, we will run the software on multiple devices to tweak the performance in case of slowdowns.

### 6.7.9 Regression Testing

To prevent this kind of error, automated tests will be made to ensure that the modified code didn't affect other parts of the software.

### 6.7.19 Acceptance Testing

To ensure the end-user's goals with the software are met, we will test out each and every feature of our application to make sure they perform as expected. For example, our main function being spam detection will be run through automated tests to determine if it is indeed accurate and prevent the program from adding non-spam messages in the spam category.

### 6.7.11 Beta Testing

As our program will only be utilized by us and the evaluators, and being for non-commercial use, beta testing will be done by us.

## 6.8 Pass / Fail Criteria

Based on the below criteria the testing team decides if the testing phase is successful or not, Pass or fail criteria:

- All essential components need to be tested.
- Defects (bugs) detected during the testing phase should be resolved or fixed.
- Models used to process data must pass an efficiency threshold decided by the tester.
- Interface components must pass an interactivity test.
- Security measures must be secure with no data leakage or weaknesses.
- Failsafe measures such as back-ups must pass a testing of operability.

### 6.8.1 Suspension Criteria

- Major defects that have a major effect on the output of the testing phase.
- Different testing output result from expected.
- API connection failure.
- Component communication issues.

### 6.8.2 Resumption Criteria

- All the issues of suspension must be resolved.
- All testing outputs must match the expected results.
- The API connection must work flawlessly.
- Components must work together seamlessly.

### 6.8.3 Approval Criteria

- A table for all the essential requirements that must be met.
- A complete documentation of the testing process.
- The tester's approval of the testing results.

## 6.9 Testing Process

Test type	Evaluation criteria for testing results
Unit test	Testing the adequate functionality of each unit test.
Integration Testing	Detection of issues in the integrated unit's interaction.
System Testing	Calculation of performance and efficient usage of resources.
Acceptance Testing	Assessing the system based on the essential requirements.

*Table 6 - Types of tests*

### 6.9.1 Test Deliverables

- Testing Plan.
- Testing Use Cases.
- RTM (Requirements Traceability Matrix).
- Testing implementation.

- Testing summary report.

### 6.9.2 Testing Tasks

- Report any bugs detected.
- Solve any issues faced during the testing phase and document them.
- Test the program with unit, components, and integration testing.
- Test use cases for unexpected results and outliers.
- Test various models for optimal data model.
- Achieve desired and expected results after the conclusion of the testing phase.

### 6.9.3 Responsibilities

Role	Responsibilities
Test Manager	Supervise the testing process and direct the testing process based on the schedule.
Software Test Engineer	Tests the system using specialized tools and techniques.
Test Analyst	Create use cases and test conditions for the Software Test Engineer.

*Table 7 - Table of Responsibilities*

### 6.9.4 Resources

- Testing personnel (e.g. Test Manager, Software Test Engineer, and Test Analyst)

- Device used for the testing process: Personal desktop computer.
- Programming language used in testing: Java.
- Automated testing Tool: e.g. Junit
- Operation System used: Windows 10.
- Virtual mobile device environment: a low-range Android phone (e.g. Samsung Galaxy A03s).

#### 6.9.5 Schedule

Activity	Duration	Starting Date	Finish Date
Testing Plan	5 days	Sunday 22/1/2023	Friday 27/1/2023
Testing Use Cases.	3 days	Sunday 29/1/2023	Tuesday 31/1/2023
RTM (Requirements Traceability Matrix).	1 day	Wednesday 22/2/2023	Thursday 23/2/2023
Testing implementation.	10 days	Friday 24/2/2023	Sunday 5/3/2023
Testing summary report.	4 days	Monday 6/3/2023	Thursday 10/3/2023

*Table 8 - Schedule*

#### 6.10 Environmental Requirements

Here we will specify all the requirements to build a proper testing environment for the application. For our project the testing environment will be limited to mobile android devices. The testing environment will offer the supplies needed to produce accurate tests. As previously mentioned, a collection of dummy data will be used as testing messages.

### 6.10.1 Hardware

A 5 years old or newer android mobile device is recommended, and the following hardware is required for testing:

Component	Requirement
RAM	8 GB of RAM is recommended as higher-end devices have this amount of RAM to make sure the test results reflect how the app will run on average mobile devices.
Memory	64 GB of memory or higher are more than sufficient.
Internet Connection	Basic 4G Wi-Fi connection will be needed for the messages to arrive to the device and for the app to communicate with the ML model.
Processor	No specific processing power is required since all the calculations are done via the ML model, and so an average mobile CPU is more than enough.

*Table 9 - Recommended Hardware*

### 6.10.2 Software

For our project, we will be testing the software manually. As such, no automated tools or scripts will be used. Each tester will take over the role of the end-user and will navigate and experience the application services in search for bugs or any unexpected behavior. As for the android OS it will be the latest version which is currently Android 12.

### 6.10.3 Security

As this is a research project and is not intended for commercial use, not many security measures are needed for the testing environment. As for the assets, the SMS messages will be kept private and will not be saved inside the ML model database.

#### 6.10.4 Tools

This section is not a requirement in our case since we will not be using any testing tools and will go with the manual approach through testing with an android mobile device.

#### 6.10.5 Publications

During the execution of the test plan, a set of documents and reports will be created mainly for the purpose of tracking the testing process. These documents include:

- Test strategy
- Test plan
- Test traceability matrix
- Test summary

#### 6.10.6 Risks and Assumptions

Since the scope of our project is limited and the vision for the application and how we will implement it is well-defined we're safe to say there are no risks of late delivery of the project. But some issues can arise like the team not having access to android devices (some of our team have android devices but not all). In this the members of the testing team may conduct the testing process via emulators on desktop computers.

### 6.11 Change Management Procedures

- Change initiation: Any changes in the STP are applied, the transition to the new STP begins.
- Change review: Assure that no regression has occurred. Review all requirements and deliverables.
- Change authorization process: Acquire authorization to confirm changes to STP



## 6.12 Plan Approvals

The testing plan must be approved by Dr. Irfan Ullah Abdur Rab, the instructor for our project. As well as the evaluators Dr. Ma'mon Ibrahim and Dr. Farmanullah Mohammad.

## Chapter 7 – Implementation Details

### 7.1 Data Collection & Preprocessing

To begin this project we of course needed a dataset and as previously planned we collected 495 SMS messages including spam messages, advertisements, and informational and conversational messages such as log-in codes. Our dataset categorized those messages into 0 and 1, with 0 denoting a non-spam message and 1 denoting a spam message. We collected the dataset by exporting messages from phones and categorizing them manually.

To preprocess these messages, we first removed all English characters and all numerals from a text message. Then, after removing some other non-letter characters such as punctuation and emojis, we removed diacritics and tatweel characters (-, َ), which are Arabic specific and do not add any significant meaning to the words. After collecting stop word lists we removed those from the messages and we applied stemming to reduce the variance of the dataset. Stemming is the process of transforming a word into its root, so words like “jumped” and “jumps” are not counted as two different words because they share the same meaning in a text classification context.

Finally, the messages are vectorized. Vectorization separates each word in a text message into a vector such that each word is given its own attribute and its number of occurrences in a single tuple is counted and recorded. For example, the sentence “Out of the trunk, the branches grow; out of them, the twigs” is preprocessed into “Out of the trunk the branches grow out of them the twigs” which is then vectorized into a table as so.

Out	of	the	Trunk	branches	grow	them	twigs
2	2	3	1	1	1	1	1

Which is then further encoded into:

0	1	2	3	4	5	6	7
2	2	3	1	1	1	1	1

This is the final result of the preprocessing step and has proven to be by far the best way we have found to prepare Arabic text for classification.

## 7.2 Algorithm

The vectorized text is fed into a Naïve Bayes classification algorithm. Briefly, a Naïve Bayes classification algorithm is a probabilistic classifier based on applying Bayes' theorem with the "Naïve" assumption of conditional independence between pairs of features. We used a multinomial Naïve Bayes implementation which is suited for text classification. Bayes' theorem states the following, given class variable  $y$  and dependent feature vector  $x_1, \dots, x_n$  : [13]

$$P(y | x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

The algorithm returns either a 0 or a 1, depending on the final result of the NB classifier, which either denotes a non-spam or spam message.

## 7.3 Android Application

All of the mentioned above is wrapped in an Android application to further prove and extend the usability of this algorithm and model we have developed. Using Chaquopy, which is a Python SDK for Android which allows full integration of Python and Java inside Android. Using Chaquopy we integrated a special Python file within an Android app that retrieves all SMS messages on a phone and classifies them using our model. The Android app runs a python environment which runs the model and is able to call a predict function that classifies each message.

The Android app is written in Java and uses Android System API to retrieve the SMS messages and store them. Messages are only classified when read by a user to save memory and increase performance.

## 7.4 Results & Benchmarking

We used three metrics to judge our model and compare it to other potential classifiers. First, accuracy, which is the percentage of correct predictions the classifier had in comparison to all predictions. Our final model has an accuracy of 96% using testing data. Second, precision, which is the accuracy of positive predictions (in this case, a message being a spam message is a positive prediction). Our model achieved 97% precision. Third, recall, which measures how many of actual positive (spam) instances were correctly identified. Our model has a recall score of 90%.

Accuracy: 96% Precision: 97% Recall: 90%		Actual	
		Spam	Non-spam
Predicted	Spam	37	1
	Non-spam	4	82

These results should provide high confidence in the model to accurately predict whether a message is potentially harmful to the user or not and reduce the risk of harm as the result of falling victim to such messages. Snippets of code and results can be found in Appendix.

Benchmarking for the Android application was also conducted on several devices and we found that the application takes very little processing power owing to its Naïve Bayes implementation which is a very quick and efficient algorithm. The major bottleneck of the application is loading the messages from system into RAM, which takes a few seconds, and initializing the Python environment also takes in practice upwards of 2 seconds, but after which the program runs almost instantly and very smoothly. Profiling for the RAM usage was also done and was found to be less than 200MBs overall due to the recycling of resources when displaying and classifying messages.

## 7.5 Security

A major advantage of having a locally running model on a device is the near elimination of the risk of leaking personal information. The dataset used in the model is highly obfuscated through the use of the vectorizer and the data scanned by the model is only stored locally on the device. Eliminating a major attack vector.

## Chapter 8 – User Manual

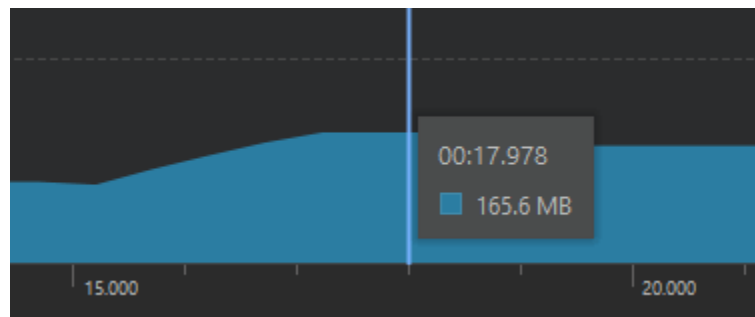
### 8.1 Recommended Hardware Requirements

#### 8.1.1 Processor

The application was tested on a Huawei Honor 8X mobile phone, which had a Hisilicon Kirin 710 processor. The performance observed on the devices was acceptable but could use some improvements. Therefore, a processor equivalent to the Kirin 710 is recommended to achieve the intended experience.

#### 8.1.2 Memory

The application takes up very little memory, due to the most significant data being loaded are the SMS messages, which are usually only text. The graph below is from the Android Studio Profiler, which provides diagnostics on CPU and RAM usage.



The application takes up 150-170 MB of ram at startup and after running the python environment for the first time, which is the point of highest load in the app. As more messages/chats are opened and exit from, RAM usage grows due to the messages being saved.

Therefore, if we assume the average user opens 10-20 messages at a time, we can conclude that the application needs about 230 MB of free RAM as the minimum amount to run this application.

#### 8.1.3 Storage Space

After exporting the application into APK format, the size required to install was 185 MB. Therefore, we recommend at least 200-210MB of available space, accounting for possible future updates.

## 8.2 Screen Items

ASMS App have 2 main interfaces, the first one is the message feed which contains the list of SMS messages obtained from the local SMS app and the second interface is the chat that is shown when the user clicks on a specific message, and it gives a warning if the message is considered spam. Below are all the screen items in detail:

- The first item that appears in both screens is a thin dark purple rectangle at the very top of the screen, it contains the current time located at the far left and the battery indicator at the far right.

### Message Feed Screen:

- The screen has a purple rectangle at the top which contains a header that reads “Messages”.
- Below is the message feed which is constructed as a stack of messages that you can scroll down through.
- Each message has 3 elements starting with the message content itself, which is shown in the middle of the message, note that the longer messages are not shown completely and are split by 3 dots at the end.
- The mobile number of the sender, located at the top left.
- The date in which the message was sent, located at the top right with dd/mm/yyyy format.

### Chat Screen:

- Like the message feed, the chat screen has a purple rectangle at the top with a header that reads “Chat” and an arrow pointing to the left for going back to the previous screen.

- Below is a white rectangle that contains the number of the sender in the center in a gray color.
- Below is the chat section which contains a single message in a light gray rectangle.
- A red rectangle with rounded corners appears above the message as a warning if the message is considered spam, the content of the warning message is “This message has a high likelihood of being spam! Be careful not to share sensitive information.”.
- Also, when the message is too long to be shown on the screen, a scroll is added so the user can scroll down and up to read the whole message.

### 8.3 Application Installation steps

The link for the drive containing the application:

[https://drive.google.com/drive/folders/1\\_1YhcrxWJuMkqp4W2vTE8KxIn6X7ecLX?usp=sharing](https://drive.google.com/drive/folders/1_1YhcrxWJuMkqp4W2vTE8KxIn6X7ecLX?usp=sharing).

- **First Step:** Open the google drive link written above using an android smartphone.
- **Second Step:** Click on the ASMS Spam Detector.apk to download it.
- **Third Step:** After the download finishes, the user will be prompted to confirm and allow google drive as a trusted source for downloading android applications.
- **Fourth Step:** After finishing step three a pop-up window will be shown to user with two options cancel and install click the latter.
- **Fifth Step:** After the program finishes installing open it.
- **Sixth Step:** after that the user will be shown a permission pop up, give the program permission to send and view SMS messages, finally the user will be able to see and use the program as intended.

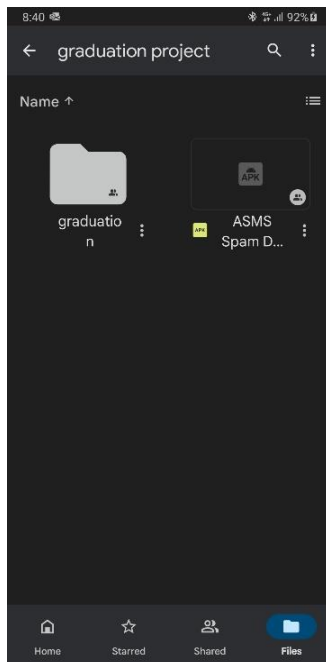


Figure 8 - First Step

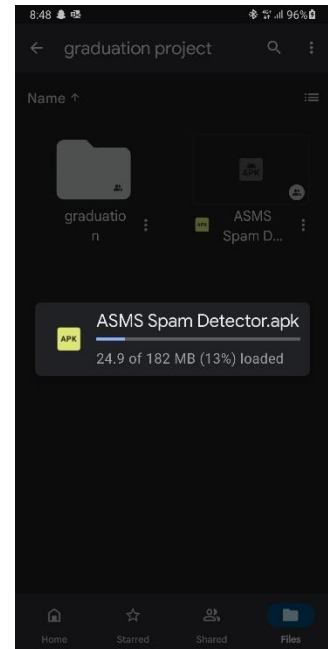


Figure 9 - Second Step

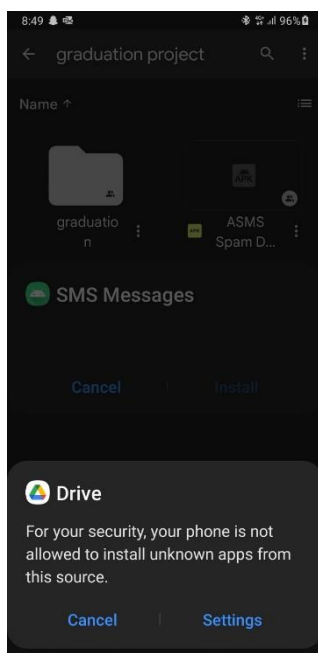


Figure 10 - Third Step

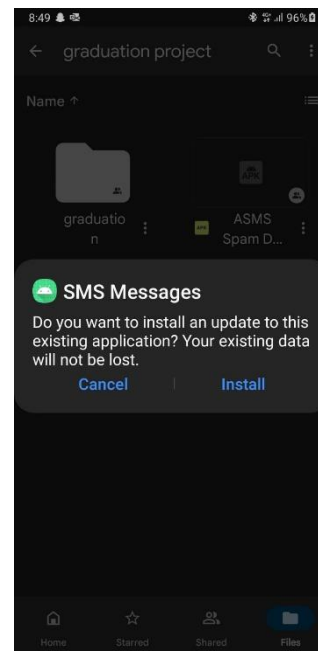
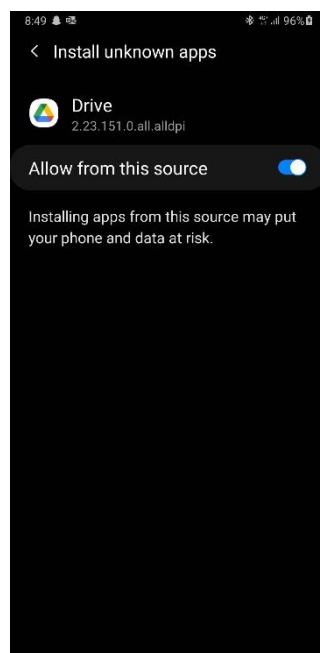


Figure 11 - Fourth Step

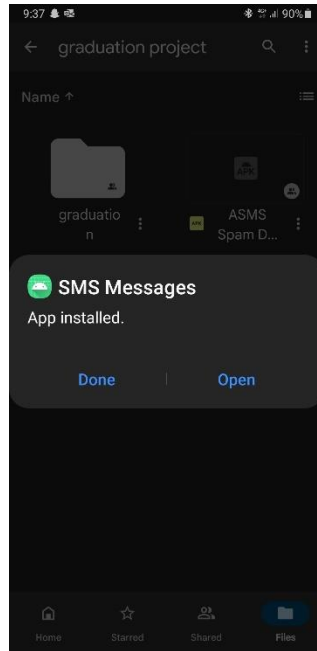


Figure 12 - Fifth Step

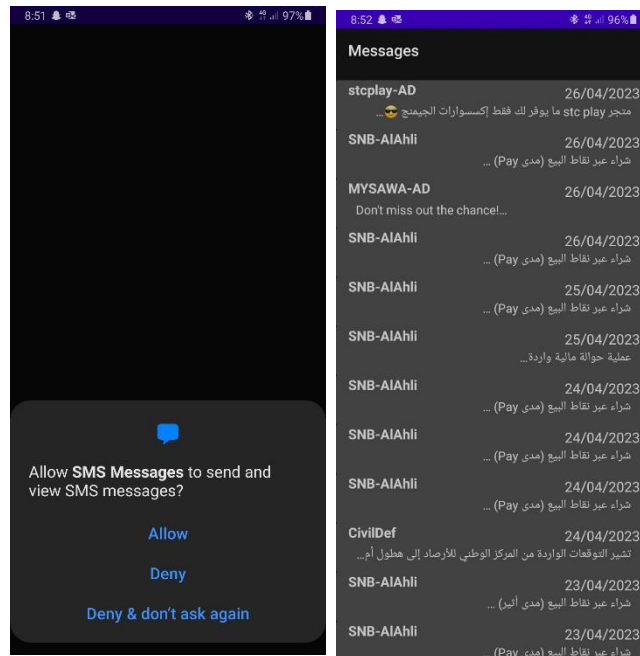


Figure 13 - Sixth Step



## Chapter 9 – Conclusion

### 9.1 Findings & Contributions

As discussed in the results & benchmarks section. We developed a model that runs in an Android application that can classify whether an Arabic SMS message is spam or non-spam with 96% accuracy. As well as narrowing a very large research gap we also produced a complete dataset that can be incredibly useful in future research in this field or even other fields such as general sentiment analysis.

Developing an Android application also allows this model to be used very easily by any user. A lightweight algorithm such as Naïve Bayes also allows a wide array of devices to be able to run this application as discussed in the benchmarking section.

### 9.2 Limitations

Time limitations were very apparent during the course of this project. The lack of any meaningful prior research and datasets made it difficult to navigate. Because of this, we could not develop the Android application to be as feature rich as possible and so it remains, as planned in the proposal, to be merely a proof that such a model could be easily run on any Android device.

### 9.3 Lessons & Skills Learned

During the course of this project, we gained a great deal of skills and expanded the ones we already had a head start on. Machine Learning applications are very complicated and need a lot of care and detail to implement in an efficient manner. Mobile application development was another completely new skill we had to learn, as it was an important part of our research. Alongside the very useful skills of collecting, categorizing data, and research which we used over and over throughout the three semesters.

## 9.4 Recommendations for future works

Although we have created a high accuracy model for Arabic SMS messages, it is still not a perfect model obviously. Due to a lot of the limitations discussed, there might be edge-cases where the model fails to recognize spam. The fact that the model's learning ability is essentially locked-in and can't continue to grow means that, in the future, it could be circumvented very easily. Therefore, a more complex and intensive model that can use a reporting system and which continues to learn would be an excellent continuation of our work.

## References

- [1] "2022 United States spam text trends," RoboKiller, 2022. [Online]. Available: <https://www.robokiller.com/spam-text-insights>. [Accessed 2022].
- [2] S. Mishra and D. Soni, "Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis," *Future Generation Computer Systems*, vol. 108, pp. 803-815, 2020.
- [3] "Understanding Naive Bayes Classifier," simplilearn, 2022. [Online]. Available: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/naive-bayes-classifier>.
- [4] J. Boyles and L. Rainie, "Mobile Phone Problems," Pew Research Center, 2012. [Online]. Available: <https://www.pewresearch.org/internet/2012/08/02/mobile-phone-problems/>.
- [5] RoboKiller, "Spam Text Messages Soar 28% In March, According to RoboKiller," RoboKiller, [Online]. Available: <https://www.prnewswire.com/news-releases/spam-text-messages-soar-28-in-march-according-to-robokiller-301520331.html>.
- [6] N. Choy, "OCBC completes arrangements for full payouts to all 790 SMS phishing victims; S\$13.7m in losses," The Business Times, 2022. [Online]. Available: <https://www.businesstimes.com.sg/banking-finance/ocbc-completes-arrangements-for-full-payouts-to-all-790-sms-phishing-victims-s137m>.
- [7] TSB Bank, "TSB research finds almost three quarters of people struggle to identify common scams, with young people most at risk," TSB Bank, 2020. [Online]. Available: <https://www.tsb.co.uk/news-releases/three-quarters-of-people-struggle-to-identify-common-scams/>.
- [8] R. Anwar, "ANDROID VS IOS DEVELOPMENT: WHICH PLATFORM SHOULD I DEVELOP FOR FIRST?," Eastern Peak, 13 11 2020. [Online]. Available: <https://easternpeak.com/blog/android-vs-ios-development-which-platform-first/>. [Accessed 2 10 2022].
- [9] S. M. A. e. al., "A Review on Mobile SMS Spam Filtering Techniques," *IEEE Access*, vol. 5, pp. 15650-15666, 2017.
- [10] A. Ghourabi, M. A. Mahmood and Q. M. Alzubi, "A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages," *Future Internet*, vol. 12, p. 156, 2020.
- [11] H. H. Mansoor and S. H. Shaker, "Using Classification Techniques to SMS Spam Filter," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 12, pp. 1734-1739, 2021.
- [12] M. B. e. al., "researchGate," October 2018. [Online]. Available: [https://www.researchgate.net/publication/329017526\\_Towards\\_building\\_an\\_anti-spam\\_Arabic\\_SMS](https://www.researchgate.net/publication/329017526_Towards_building_an_anti-spam_Arabic_SMS). [Accessed 3 October 2022].

- [13] scikit-learn, "Naive Bayes - Scikit-learn," [Online]. Available: [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html). [Accessed April 2023].

## Appendix

### Code snippet:

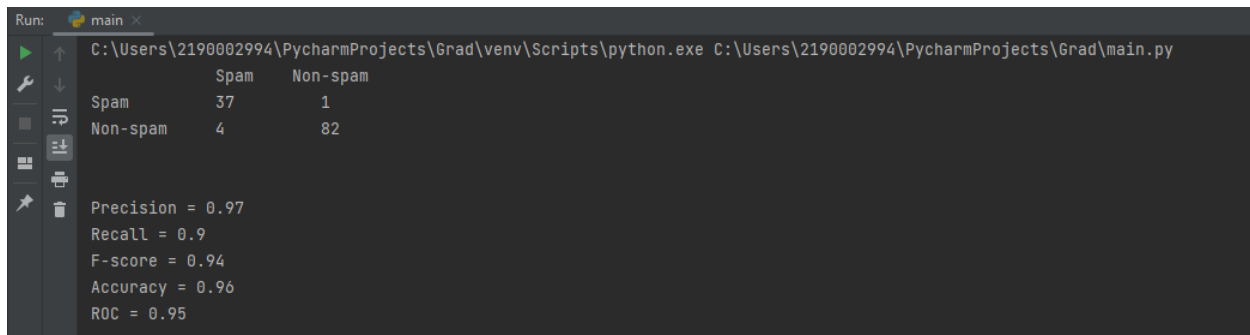
```
model.fit(train_df, train_labels)

y_pred = model.predict(test_df)

tn, fp, fn, tp = confusion_matrix(test_labels, y_pred).ravel()

print("\t\t\t Spam \t Non-spam")
print("Spam \t\t " + str(tp) + "\t\t\t" + str(fp))
print("Non-spam \t " + str(fn) + "\t\t\t" + str(tn))
print("\n")
print("Precision = {:.2}".format(precision_score(test_labels, y_pred)))
print("Recall = {:.2}".format(recall_score(test_labels, y_pred)))
print("F-score = {:.2}".format(f1_score(test_labels, y_pred)))
print("Accuracy = {:.2}".format(accuracy_score(test_labels, y_pred)))
print("ROC = {:.2}".format(roc_auc_score(test_labels, y_pred)))
print("\n")
```

### Output:



Run: main x

C:\Users\2190002994\PycharmProjects\Grad\venv\Scripts\python.exe C:\Users\2190002994\PycharmProjects\Grad\main.py

	Spam	Non-spam
Spam	37	1
Non-spam	4	82

Precision = 0.97  
Recall = 0.9  
F-score = 0.94  
Accuracy = 0.96  
ROC = 0.95