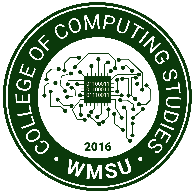
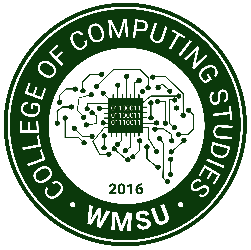
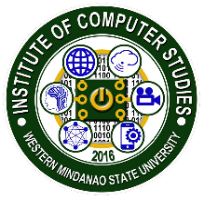
****

Republic of the Philippines

Western Mindanao State University

**COLLEGE OF COMPUTING STUDIES**

DEPARTMENT OF INFORMATION TECHNOLOGY

**PhishTi: An ML-Based Mobile Application for Real-Time SMS Phishing Detection with URL Filtering**

A CAPSTONE PROJECT PROPOSAL

Presented to the Faculty of the

College of Computing Studies

Western Mindanao State University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Information Technology

ALIBASA, SHEIK AISHIR S.

HAJIHIL, ALKHABIR G.

ABILUL, DANILO S.

Researcher

CATADMAN, JASON A.

Adviser

June 2026

***Note:*** *The information in this sample documentation is provided solely as a guide for students in developing their capstone project and research documentation. It should not be copied and pasted. The content is intended to help students understand the structure and elements of a comprehensive project report and will also be used during lecture discussions to illustrate key concepts and best practices.*

TABLE OF CONTENTS

Chapter Page

[**1** **INTRODUCTION** 4](#_Toc200011267)

[1.1 Project Context 4](#_Toc200011268)

[1.2 Statement of the problem 7](#_Toc200011269)

[1.3 Purpose and Description 8](#_Toc200011270)

[1.4 Objectives 10](#_Toc200011271)

[1.4.1 Specific Objective 10](#_Toc200011272)

[1.5 Scope and Limitation 11](#_Toc200011273)

[1.6 Significance of Study 12](#_Toc200011274)

[1.7 Definition of Terms 13](#_Toc200011275)

[**2** **REVIEW OF RELATED LITERATURE AND SYSTEMS** 15](#_Toc200011276)

[2.1 Introduction 15](#_Toc200011277)

[2.1.1 AdviseMe: An Intelligent Web-Based Application for Academic Advising 15](#_Toc200011278)

[2.1.2 Online Degree Auditing for Undergraduate College Students 16](#_Toc200011279)

[2.1.3 SAAS: Creation of an e-Advising Tool to Augment Traditional Advising Methods 17](#_Toc200011280)

[**AdviseMe: An Intelligent Web-Based Application for Academic Advising** 18](#_Toc200011281)

[2.2 Summary 18](#_Toc200011282)

[2.3 Synthesis 19](#_Toc200011283)

[**3** **TECHNICAL BACKGROUND** 20](#_Toc200011284)

[3.1 Conceptual Framework 20](#_Toc200011285)

[3.2 Software and/or Hardware Requirements 23](#_Toc200011286)

[**4** **DESIGN AND METHODOLOGY** 25](#_Toc200011287)

[4.1 Research Design 25](#_Toc200011288)

[4.2 Developmental Methodology 25](#_Toc200011289)

[4.3 System Architecture 25](#_Toc200011290)

[4.4 Data Collection Methods 25](#_Toc200011291)

[4.5 Requirements Analysis 25](#_Toc200011292)

[4.5.1 Context Diagram 26](#_Toc200011293)

[4.5.2 Data Flow Diagram 27](#_Toc200011294)

[4.5.3 Flowchart 27](#_Toc200011295)

[4.5.4 Use Case 28](#_Toc200011296)

[4.6 Requirement Specification 30](#_Toc200011297)

[4.6.1 Functional Requirements 30](#_Toc200011298)

[4.6.2 Non-Functional Requirements 32](#_Toc200011299)

[4.7 System Design 35](#_Toc200011300)

[4.7.1 Database Design 35](#_Toc200011301)

[4.7.2 Entity Relationship Diagram 36](#_Toc200011302)

[4.7.3 Database Fields 36](#_Toc200011303)

[4.8 Tools and Technology 39](#_Toc200011304)

[4.9 Evaluation and Testing 39](#_Toc200011305)

**CHAPTER 1**

# **INTRODUCTION**

## Project Context

Cybersecurity threats have rapidly evolved, and phishing remains one of the most prevalent and damaging forms of attack. Phishing typically involves tricking recipients into revealing sensitive information such as banking credentials, personal identifiers, and account passwords. In recent years, **SMS-based phishing, also known as smishing**, has become more widespread as mobile devices serve as the primary means of communication for many individuals. Attackers craft fraudulent SMS messages that impersonate banks, government agencies, or companies to deceive users into clicking malicious links or sharing private data.

In the Philippines, mobile subscribers regularly encounter suspicious SMS claiming that they have won prizes, offering fake job opportunities, or requesting sensitive information. Despite the presence of built-in spam filters in mobile devices and telecommunication services, many individuals still fall victim to these attacks. Traditional defenses such as static blacklists or keyword-based detection often fail against more sophisticated phishing techniques, such as obfuscating words or using newly registered domains.

Machine learning (ML) has emerged as a powerful solution to address these limitations by enabling systems to learn and adapt to patterns found in phishing attempts. By training models on large datasets of phishing and legitimate SMS, ML can provide more accurate and adaptive detection compared to rule-based systems. However, most existing research only focuses on classification and detection without integrating practical enforcement mechanisms that ensure users are truly protected.

This project, **PhishTi**, addresses this gap by developing a **machine learning-based mobile application** that intercepts SMS messages, classifies them as phishing or legitimate, and archives messages identified as phishing. Additionally, when an SMS contains a URL, the system evaluates the link using the same ML-driven analysis to identify potential malicious websites. By combining SMS interception, ML classification, and URL filtering, PhishTi provides real-time protection and reduces the likelihood of users becoming victims of smishing attacks.

## Statement of the problem

Phishing continues to be one of the most prevalent cybersecurity threats, with SMS phishing—or smishing—emerging as a particularly dangerous attack vector. In the Philippines, fraudulent text messages claiming prizes, impersonating banks, or urging users to click on suspicious links are common. Existing filtering measures, such as keyword-based detection or blacklist systems, often fail to adapt to attackers’ evolving strategies. Cybercriminals employ techniques like obfuscating words, using short-lived domains, and creating convincing replicas of legitimate messages, which allow phishing attempts to bypass traditional defenses.

Furthermore, current SMS spam filters typically focus only on detection and alerting but lack enforcement mechanisms that can block or archive phishing attempts. As a result, even if users are notified, they may still receive and interact with the same malicious content repeatedly, increasing their exposure to scams. Additionally, while there are solutions that detect phishing in web browsers, very few extend their capabilities to analyze URLs embedded within SMS, leaving a critical gap in mobile protection.

Given these challenges, there is a pressing need for a more robust and intelligent system that not only detects phishing SMS in real time but also provides active protection by archiving or blocking malicious messages. By incorporating machine learning, such a system can adapt to new phishing techniques and enhance detection accuracy. At the same time, integrating URL analysis within SMS ensures users are safeguarded against one of the most common delivery methods of phishing attacks. Addressing these problems is essential to protect mobile users from financial fraud, identity theft, and data breaches caused by smishing.

## Purpose and Description

The purpose of this study is to design and develop PhishTi, a mobile application that uses machine learning to detect phishing SMS and analyze embedded URLs for malicious content. The app will intercept incoming SMS, classify them using a trained ML model, and automatically archive those identified as phishing. If a URL is detected within an SMS, the app will further evaluate the link, ensuring that users are not exposed to harmful websites.

PhishTi is designed as a user-friendly mobile application for Android devices, given their widespread use and open system that allows SMS interception. The app provides users with an archiveofdetectedphishingmessages, notifications for blocked attempts, and a history of system decisions for transparency. By combining ML-driven detection with enforcement measures, PhishTi ensures that users are both informed of potential threats and actively protected from them.

## Objectives

The primary objective of this project is to develop an Android mobile application called PhishTi that proactively detects and prevents phishing attempts delivered through SMS messages, with the capability to analyze embedded URLs for malicious content. The system will integrate machine learning–based classification with real-time enforcement mechanisms, ensuring that phishing messages are intercepted, archived, and prevented from reaching users repeatedly. By combining intelligent detection with proactive protection, the project aims to safeguard users against smishing attacks and enhance the overall security of mobile communication.

### Specific Objective

* To collect and preprocess datasets of phishing and legitimate SMS, including messages with embedded URLs.
* To design and train ML models (transformer-based for SMS, hybrid approaches for URL detection).
* To develop a mobile application that intercepts SMS and archives those classified as phishing.
* To implement URL filtering for links contained in SMS to block access to malicious websites.
* To evaluate the system’s performance using accuracy, precision, recall, and F1-score.
* To conduct software testing and user evaluation to ensure usability and effectiveness.

## Scope and Limitation

This project focuses on the development of PhishTi, an Android mobile application that intercepts SMS messages and classifies them as either legitimate or phishing using machine learning models. The core functionality lies in its ability to detect phishing messages in real time and automatically archive those identified as malicious to prevent repeated delivery. A secondary feature includes the analysis of URLs embedded within SMS, enabling the system to identify and block access to malicious links that are often used as phishing vectors. The app also provides a phishing archive that allows users to review flagged messages and understand the nature of detected threats.

However, the system will be limited to Android devices only, as iOS does not permit third-party applications to intercept SMS messages. The application will not cover other forms of phishing, such as email-based or voice call phishing, nor will it provide a comprehensive browser-level filtering system beyond the URLs contained within SMS messages. Furthermore, the detection accuracy of the system depends on the quality and diversity of datasets used in training the machine learning model, meaning that highly sophisticated or zero-day phishing attempts may still evade detection.

In addition, PhishTi’s functionality will not extend to providing global network-level protection; it is designed solely as a device-level application for end-users. The project also excludes advanced user authentication or cloud-based synchronization features, focusing instead on local archiving and detection. By defining these boundaries, the project establishes realistic expectations and ensures that development efforts remain aligned with the primary goal of creating an intelligent, mobile-based SMS phishing detection and filtering system.

## Significance of Study

The development of PhishTi is significant because it directly addresses the growing problem of SMS phishing, or smishing, which continues to deceive users despite existing security measures from mobile providers. By leveraging machine learning, the system is capable of adapting to evolving phishing strategies, providing a more intelligent and reliable approach than traditional keyword-based or blacklist methods. For everyday mobile users, PhishTi offers real-time protection by intercepting and archiving phishing SMS, reducing the likelihood of falling victim to scams, identity theft, and financial fraud.

For the academic and research community, this study contributes to the growing body of knowledge on the practical application of artificial intelligence in cybersecurity. Unlike many purely experimental models that exist only in controlled environments, PhishTi demonstrates how machine learning models can be integrated into a real-world mobile application to provide immediate, user-centered protection. This research also sets a precedent for future studies exploring AI-driven approaches to mobile security, particularly in regions where smishing is prevalent.

From a broader societal perspective, PhishTi supports digital safety and helps build user trust in mobile communication. By actively reducing the risks posed by fraudulent SMS and malicious URLs, the system promotes safer use of mobile technology, which is increasingly essential in the digital economy. Its implementation can inspire further innovation in AI-based security tools and serve as a model for future solutions that aim to counteract phishing and other cyber threats, ultimately contributing to the collective effort of strengthening cybersecurity at the individual and community levels.

## Definition of Terms

Defined terms are those whose meanings in the study are particular or unique. Give a minimum of fifteen. The researcher must credit his sources for any definitions he uses, even if they are from dictionaries, books, journals, newspaper articles, and other sources. Technical and conceptual definitions are those that are derived from published sources.

Table 1: Definition of Terms

| **Term** | **Definition** |
| --- | --- |
| 1. Phishing | is a fraudulent attempt to obtain sensitive information such as usernames, passwords, or financial details by disguising oneself as a trustworthy entity in electronic communication (Jagatic et al., 2007). |
| 1. SMS Phishing (Smishing) | A type of phishing attack delivered through SMS messages, where fraudulent texts are used to lure victims into clicking malicious links or sharing personal data (Almomani et al., 2013). |
| 1. Malicious URL | A hyperlink crafted by attackers that directs users to harmful websites, often used in phishing campaigns to steal data or distribute malware (Cova et al., 2010). |
| 1. Machine Learning (ML) | A branch of artificial intelligence that enables systems to automatically learn patterns from data and improve detection accuracy without explicit programming (Mitchell, 1997). |
| 1. Transformer Model | A deep learning architecture that uses attention mechanisms for natural language processing tasks, including text classification such as detecting phishing SMS (Vaswani et al., 2017). |
| 1. URL Filtering | A security technique used to analyze and block access to malicious or suspicious links, preventing users from visiting harmful websites (Provos et al., 2007). |
| 1. PhishTi | The proposed mobile application developed in this study that leverages ML-based models to intercept SMS, classify messages as phishing or legitimate, and archive malicious messages for user protection (Researchers, 2025). |
| 1. SMS Interception | The process of capturing incoming SMS messages by an application to analyze and determine whether they are safe or potentially harmful (Google Android Developers, 2023). |
| 1. Archiving | The storage of phishing SMS messages in a separate section of the application to prevent repeated delivery and allow users to review detected threats (Almeida et al., 2011). |
| 1. Dataset | A collection of phishing and legitimate SMS messages and URLs used to train and test machine learning models for classification tasks (Hidalgo et al., 2013). |
| 1. Preprocessing | The data preparation step that involves cleaning, tokenizing, normalizing, and labeling SMS or URL datasets before training machine learning models (Han et al., 2011). |
| 1. Precision | A performance metric in classification that measures the proportion of correctly identified phishing messages out of all messages classified as phishing (Sokolova & Lapalme, 2009). |
| 1. Recall | A performance metric that evaluates the ability of a model to correctly identify all phishing messages in a dataset (Sokolova & Lapalme, 2009). |
| 1. F1-Score | The harmonic mean of precision and recall, providing a balanced measure of model performance for phishing detection tasks (Sokolova & Lapalme, 2009). |
| 1. Real-Time Detection | The ability of a system to analyze and classify SMS messages instantly as they are received, providing immediate protection to users (Stallings, 2018). |

**CHAPTER 2**

# **REVIEW OF RELATED LITERATURE AND SYSTEMS**

## Introduction

Phishing attacks delivered through SMS remain one of the most persistent and damaging threats to digital security. SMS-based phishing, also known as smishing, deceives mobile users through fraudulent messages that impersonate trusted organizations, luring them into revealing sensitive information or clicking on malicious links. While SMS spam filters and blacklist-based defenses exist, these methods are often limited in scope and fail to detect zero-day phishing attempts or obfuscated text patterns that cybercriminals use to bypass detection. Similarly, URLs embedded in SMS continue to be effective phishing vectors, yet many existing solutions do not address this layer of risk.

In recent years, researchers have proposed various datasets, machine learning models, and hybrid approaches to improve phishing detection. These studies have significantly advanced detection accuracy and demonstrated the potential of AI-driven security. However, many of these works remain experimental, focusing mainly on model performance without extending their application to mobile platforms where smishing is most prevalent. Furthermore, enforcement mechanisms—such as automatically intercepting and archiving phishing SMS—are often absent, leaving a gap between detection and practical user protection.

This chapter reviews related literature and systems that have contributed to the development of SMS and URL phishing detection technologies. It outlines their contributions, strengths, and limitations while identifying gaps that PhishTi aims to address. Unlike previous works, PhishTi is designed as a real-time, ML-based mobile application that not only classifies phishing SMS but also enforces protection by archiving malicious messages and analyzing embedded URLs. Through this integration, the system provides a more practical and user-centered approach to combating smishing in everyday mobile communication.

The following literature and systems exhibit comparable structures and thematic relevance.

### Contributions to the Study of SMS Spam Filtering: New Collection and Results

Almeida, Hidalgo, and Yamakami (2013) introduced the SMS Spam Collection dataset, which has become a benchmark for SMS spam and phishing research. Their study compared machine learning algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Decision Trees. While the models performed well on spam classification, they struggled with sophisticated phishing messages crafted to evade static filters.

The core features of the study include:

* Creation of the SMS Spam Collection dataset.
* Evaluation of classical ML algorithms.
* Provided benchmark results for SMS filtering research.
* Identified limitations of static keyword detection.

“The system was able to highlight the need for adaptive learning methods beyond static filtering, showing that traditional classifiers, while effective against spam, were insufficient for evolving phishing strategies.” (Almeida et al., 2013)

### Towards SMS Spam Filtering: Results Under a New Dataset

Hidalgo et al. (2013) extended the earlier study by analyzing how adversarial modifications affected SMS spam filters. Their findings revealed that keyword-dependent systems failed when attackers introduced deliberate variations, symbols, or obfuscation, underscoring the importance of adaptive models.

The core features of the study include:

* Evaluation of spam filters under adversarial and obfuscated text.
* Performance analysis using the SMS Spam Collection dataset.
* Demonstrated weaknesses of keyword-dependent systems.
* Recommended adaptive, ML-based methods.

“Keyword-dependent systems show significant weaknesses when confronted with adversarial text variations, underscoring the importance of adaptive filtering methods.” (Hidalgo et al., 2013)

### Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages

Canali et al. (2011) developed **Prophiler**, a large-scale malicious website filter that relied on blacklists such as Google Safe Browsing. The system effectively blocked known phishing websites but was reactive in nature and failed to detect zero-day phishing URLs.

The core features of the system include:

* Large-scale filtering of malicious web pages.
* Use of blacklists for phishing detection.
* IEffective against known phishing URLs.
* Identified vulnerability to zero-day attacks.

“Blacklist-based systems are valuable as a first defense but are inherently reactive, leaving users exposed to new and rapidly changing phishing domains.” (Canali et al., 2011)2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 2.0 Review of Related Works and System Comparison Table** | | | | |
| **Features** | **Contributions to the Study of SMS Spam Filtering: New Collection and Results** | |  | | --- | |  |   **Towards SMS Spam Filtering: Results Under a New Dataset** | **Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages** | **Proposed Study** |
| |  | | --- | | Dataset Creation |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | SMS Classification |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Handling Obfuscation /  Adversarial Text |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Blacklist Usage |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | URL Phishing Detection |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Large-Scale Filtering |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Machine Learning Models |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Transformer/Deep  Learning |  |  | | --- | |  | |  |  |  |  |
| |  | | --- | | Enforcement Mechanism  (Archiving/Blocking) |  |  | | --- | |  | |  |  |  |  |

The table shows that while existing studies contribute important features such as dataset creation, SMS classification, and URL filtering, they remain limited in adaptability, enforcement, and user privacy. The proposed system, PhishGuard, builds on these works by combining SMS and URL phishing detection into one mobile app, using deep learning models, real-time enforcement, and optional user authentication.

## Summary

The reviewed studies highlight progress in phishing detection, from benchmark datasets for SMS spam to advanced hybrid and deep learning models for phishing websites. These works demonstrate the value of machine learning and feature integration in improving detection accuracy. However, most existing systems are either reactive, focusing on detection without enforcement, or limited in scope, addressing only SMS or only URLs. Furthermore, privacy and personalization features for end users remain largely unexplored in previous research.

## Synthesis

From the review, three gaps become clear: the lack of unified mobile solutions for SMS and URL phishing detection, the absence of real-time enforcement mechanisms, and the neglect of user privacy and personalization. The proposed system, **PhishGuard**, addresses these gaps by combining SMS and URL phishing detection within a single Android application. Powered by transformer-based and hybrid deep learning models, it ensures adaptive classification accuracy while introducing enforcement features such as archiving phishing SMS and blocking malicious URLs. Additionally, the inclusion of **optional user authentication** provides privacy and control over user logs, making the system not only technically advanced but also user-centered.

By bridging detection, enforcement, and privacy, PhishGuard advances the field of phishing prevention and delivers a practical, deployable solution tailored to mobile users.

**CHAPTER 3**

# **TECHNICAL BACKGROUND**

## Conceptual Framework

*Figure 3.1*

The PhishGuard mobile application is designed to provide mobile users with proactive protection against phishing attempts delivered through SMS and malicious URLs. The system operates on a three-layered architecture: Presentation Layer, Application and Logic Layer, and Database Layer, ensuring both functionality and security.

At the input stage, the system utilizes phishing and legitimate datasets of SMS and URLs. These datasets undergo data preparation, which includes data cleaning, labeling, and splitting into training, validation, and testing sets. The preprocessed data is then used to train machine learning models—a transformer-based model for SMS classification and a hybrid deep learning model (CNN + lexical features) for URL detection.

The application integrates these models into two major modules: (1) the SMS Phishing Detection Module, which intercepts incoming text messages, classifies them, and archives phishing attempts; and (2) the Browser Integration Module, which operates either as an in-app browser or integrates with the device’s default browser to detect and block malicious URLs in real time.

To enhance user trust and privacy, the application includes an optional user authentication feature. Users can log in to secure their phishing logs and preferences, or continue in local-only mode without registration. This ensures flexibility while maintaining security.

The outputs of the system are an Android mobile application capable of AI-driven phishing detection, real-time enforcement mechanisms (archiving SMS and blocking URLs), and enhanced privacy through optional authentication.

**3.1.1 Presentation Layer**

The presentation layer of PhishGuard serves as the primary interface through which users interact with the system. Developed as an Android mobile application, this layer is designed to be simple, intuitive, and user-friendly to ensure accessibility for both technical and non-technical individuals. Through the interface, users are able to view their incoming SMS messages, receive phishing alerts, and access logs of archived phishing attempts. In addition, the application provides a dedicated in-app browser where users can safely open links, while also being notified if a website has been flagged as malicious.

Another important feature in the presentation layer is the optional user authentication system. This allows users to register or log in to secure their phishing logs, settings, and preferences. For individuals who prefer not to register, the application still operates fully in local mode, storing detection results directly on the device. By combining usability with flexible security options, the presentation layer ensures that PhishGuard can cater to a wide range of users, delivering both convenience and protection.

**3.1.2 Application and Logic Layer**

The application and logic layer is the core of the PhishGuard system, where all essential processing and decision-making take place. At this layer, the SMS phishing detection module intercepts incoming messages through Android APIs and applies a transformer-based machine learning model trained on phishing and legitimate datasets. Messages identified as phishing are immediately archived, ensuring that users are not exposed to repeated or harmful content. Similarly, the browser integration module employs a hybrid deep learning model that combines CNNs with lexical and structural features to evaluate the safety of URLs accessed either within the in-app WebView or through supported browser integrations. When a link is classified as phishing, the system automatically blocks access, preventing users from being redirected to harmful websites.

Beyond detection, the application and logic layer implements critical enforcement mechanisms that distinguish PhishGuard from traditional detection-only systems. These mechanisms include automatic archiving of phishing SMS, disabling of suspicious URLs, and real-time user notifications to inform individuals of attempted attacks. The logic layer also incorporates authentication and privacy management, ensuring that user credentials and logs are encrypted and protected. Together, these functions create a proactive defense system that not only identifies threats but actively prevents users from engaging with them.

**3.1.3 Database Layer**

The database layer supports the operation of PhishGuard by securely storing all necessary information related to detection, enforcement, and user preferences. For users operating in local mode, the database stores phishing logs, archived SMS, and blocked URLs directly on the mobile device using a lightweight and efficient structure. This ensures quick retrieval and minimal reliance on external servers, which enhances both speed and privacy.

When users enable the optional authentication feature, the database additionally stores credentials and personalized settings. To ensure the protection of sensitive data, the system applies encryption and hashing techniques, securing both user information and phishing-related logs. This dual storage approach—local for quick operations and encrypted storage for authenticated sessions—provides a balance between functionality and security. By managing information in this structured and secure manner, the database layer supports the seamless operation of the PhishGuard mobile application and guarantees the integrity of data processed by the system.

**3.1.4 User Interactions**

User interaction with PhishGuard is designed to be straightforward and efficient, with the goal of providing maximum protection with minimal effort from the end user. Once the application is installed, users can immediately benefit from phishing detection and enforcement features without the need for prior configuration. Incoming SMS messages are automatically scanned, and phishing attempts are archived in a separate log to prevent repeated exposure. When users click on a URL, the browser integration module evaluates the link in real time, allowing safe websites to load normally while blocking or disabling access to suspicious ones. Notifications are sent whenever phishing content is intercepted, keeping users informed without overwhelming them.

For individuals who enable the optional authentication feature, interaction is extended to include login and account management. This allows users to secure their phishing logs and preferences, ensuring that only they have access to sensitive records. Even without authentication, users retain the ability to review detection results locally on their device. Overall, user interaction with PhishGuard balances automation with control, delivering a system that is both user-friendly and highly protective.

## Technology Requirements

PhishGuard relies on a set of modern technologies to ensure functionality, reliability, and security. The mobile application is developed using Android Studio with either Java or Kotlin, guaranteeing compatibility with a wide range of Android devices running Android 8.0 (Oreo) and higher. Machine learning models for SMS and URL phishing detection are trained separately using Python, employing frameworks such as TensorFlow and PyTorch for deep learning model development. Supporting libraries including Scikit-learn, NumPy, and Pandas are used for preprocessing, feature extraction, and performance evaluation. Once trained, these models are exported and embedded into the Android application for on-device phishing detection.

For data storage, the application uses SQLite, a lightweight and reliable mobile database that stores phishing logs, detection results, and user settings. When optional authentication is enabled, cryptographic methods such as bcrypt hashing and AES encryption are applied to protect sensitive data and ensure privacy. Version control and collaboration during development are maintained using Git, allowing for efficient source code tracking and project coordination. By combining these technologies, PhishGuard ensures a secure, scalable, and efficient architecture capable of real-time phishing detection and prevention.

**Software Requirements**

1. Operating Systems:

The PhishGuard application runs on the Android operating system, supporting Android 8.0 (Oreo) and higher. For development purposes, the machine learning models are trained on desktop or laptop environments running modern operating systems such as Windows 10/11 or Ubuntu Linux.

1. Database Management System:

The system uses SQLite, a lightweight embedded database suited for mobile environments. This database manages local storage of phishing logs, archived SMS, and detection results, ensuring quick data retrieval without the need for a dedicated server.

1. Development Tools:

The mobile application is developed using Android Studio. For model development, Python with TensorFlow and PyTorch serves as the primary environment. Libraries such as Scikit-learn, Pandas, and NumPy are used for preprocessing, feature engineering, and evaluation.

1. Programming Languages and Libraries:

Java/Kotlin**:** For mobile application development.

Python**:** For training machine learning models.

Security Libraries**:** Including **bcrypt** for hashing and **AES** for encryption of sensitive user data.

1. Version Control:

Git is used for collaborative development, version tracking, and code management.

Hardware Requirements

1. Client Devices:

The PhishGuard application is designed to run on Android smartphones with at least 2 GB RAM, quad-core processors, and 100 MB of available storage space. The minimum supported operating system is Android 8.0 (Oreo), though Android 10 or higher with 4 GB RAM and an octa-core processor is recommended for optimal performance.

1. Development Workstation:

Developers require a workstation with at least 8 GB RAM, a multi-core processor, and 256 GB storage to handle Android Studio and Python model training. Higher specifications improve training speed and allow parallel execution of development tasks.

1. Networking:

A stable internet connection is required during development for downloading dependencies, accessing phishing datasets, and managing Git repositories. For end users, PhishGuard can function in offline mode, but periodic internet access is recommended for updates and synchronization.

The PhishGuard mobile application not only automates the detection and prevention of phishing attempts but also ensures accurate and efficient protection of user communication data. By integrating machine learning–based SMS classification and URL analysis within a single mobile platform, the system provides a comprehensive security solution that benefits end users by minimizing exposure to fraudulent content. The use of advanced technologies such as transformer-based models, hybrid deep learning approaches, and mobile database management ensures that the application is both innovative and reliable, addressing the growing cybersecurity challenges faced by mobile users. Furthermore, its flexible architecture—capable of functioning in both offline and online modes—paves the way for future advancements in mobile security systems, offering a scalable and user-centered defense against evolving phishing threats.

**CHAPTER 4**

# **DESIGN AND METHODOLOGY**

## Research Design

*Explain and discuss the methodology used in the development of the project. Provide a detailed account of the approaches, techniques, and processes employed to achieve the project's objectives. If advanced techniques such as machine learning algorithms, AI approaches, or other sophisticated methods are utilized, ensure to include a thorough explanation of these components. Describe how these techniques were implemented and integrated into the application or project. This section should give a clear understanding of the technical framework and the step-by-step procedures followed during the development process.*

## Developmental Methodology

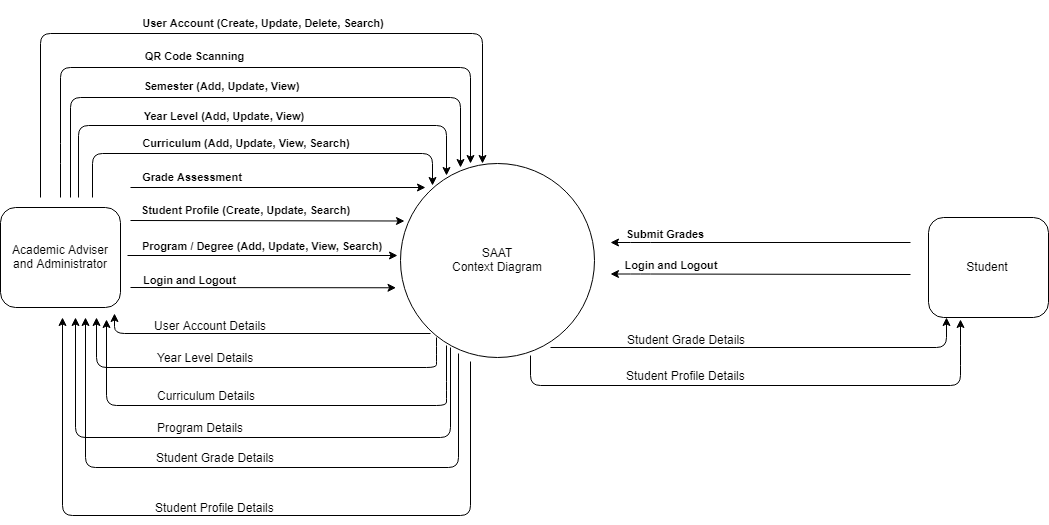
## System Architecture

## Data Collection Methods

## Requirements Analysis

Requirements analysis is a critical phase in the development of the Academic Advising Evaluation System for Undergraduate Students. This process involves gathering, documenting, and analyzing the needs and expectations of the stakeholders to ensure the system meets their requirements. The goal is to define clear, actionable, and measurable requirements that guide the development process and ensure the final product addresses the needs of its users effectively.

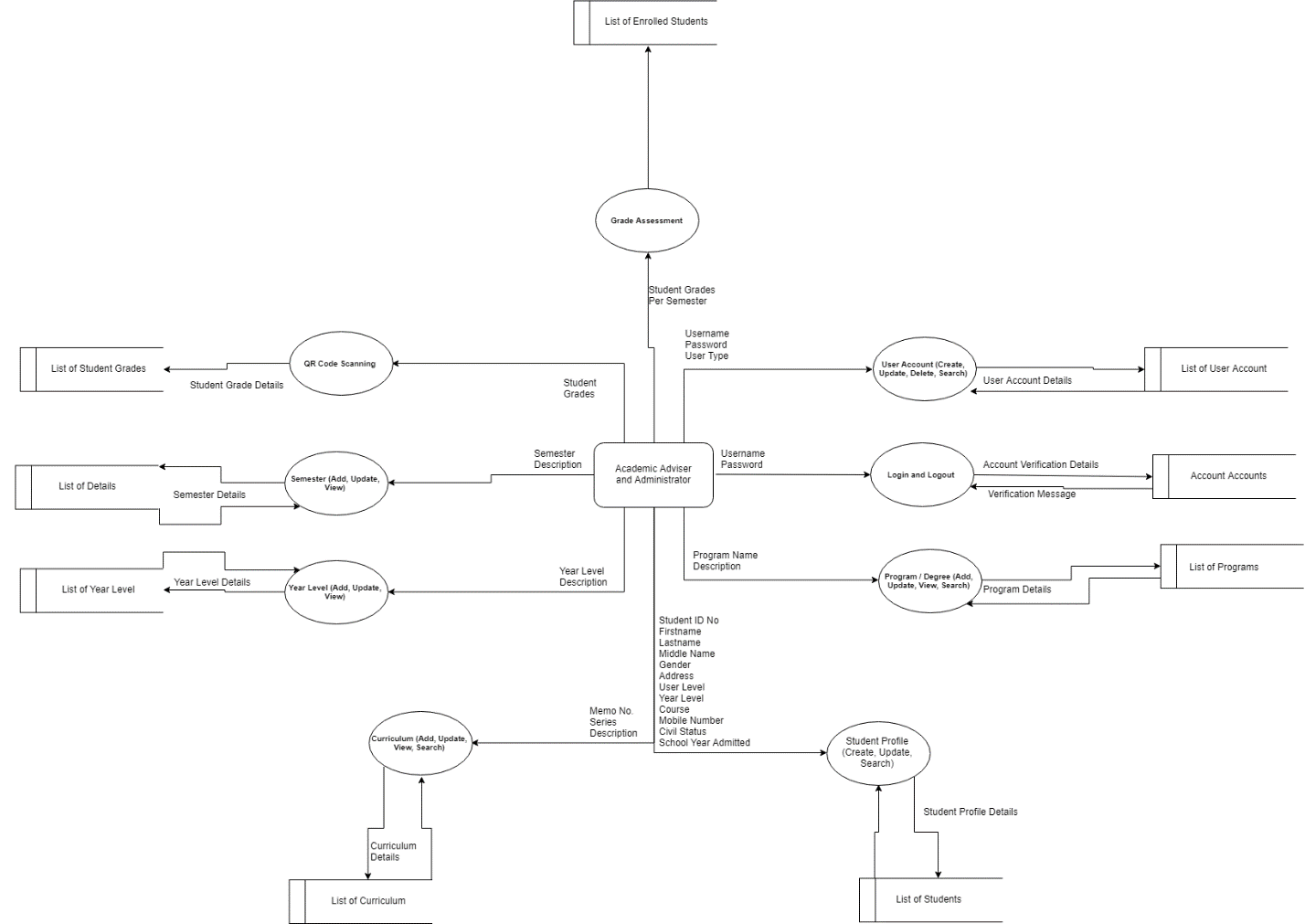
### Context Diagram



*Figure 4.2*

***(Explain / Discuss the Diagram)***

### Data Flow Diagram



*Figure 4.2*

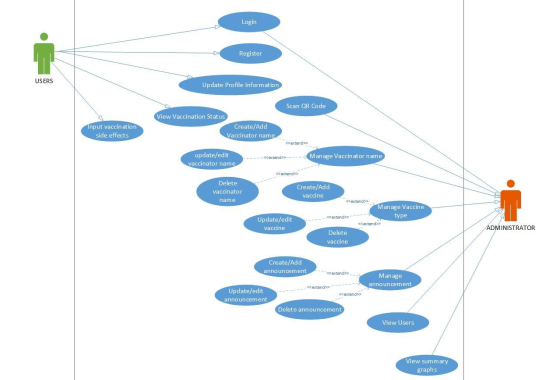
***(Explain / Discuss the Diagram)***

### Flowchart

*Figure 4.2*

***(Explain / Discuss the Flowchart of every process)***

### Use Case



*Figure 4.2*

*(****Explain / Discuss the Diagram****)*

**Use case # 1: Log-in**

**User:** Administrator, Bookkeeper, Coop-member

**Description:** The system shall allow the user to have access on the system.

**Fit Criterion:** The user inputs a validated username and password, and it should match to access the assigned module after log-in.

Use case scripts:

1. Log-in form is loaded.
2. Enter username and password.
3. If log-in button is clicked the system will verify the inputted values, if one of the values are invalid a message box will prompt “ACCESS DENIED: Incorrect username and/or password”. If the entered values are correct, a message box will prompt “ACCESS GRANTED: You have successfully logged-in.”

**Use case # 4: Terminate User Account**

**User:** Administrator, Bookkeeper

**Description:** The system must allow deleting of user accounts that are no longer in use.

**Fit Criterion:** The user can delete selected record and will not be visible when loading again the list of accounts. The deleted record will not be able to use to have access on the system.

Use case scripts:

1. Click ‘User Accounts’ from the Main form.
2. Select a record from the list and click ‘Delete’ button to delete.
3. A message box will prompt, “You are about to delete (1) record. If you click YES, you won't be able to undo this delete operation. Do you want to proceed?”
4. If ‘Yes’ button is click, a message box will prompt “Record was successfully deleted.” Else, record will not be deleted.

**Use case # 3: Update User Account**

**User:** Administrator, Bookkeeper

**Description:** The system must allow modification of existing user account information.

**Fit Criterion:** User select account on the list and make necessary changes to the selected user account record. The user can view the modified account in the list of user accounts.

Use case scripts:

1. Click ‘User Accounts’ from the Main form.
2. Account form is loaded.
3. Select a record from the list and click ‘Update’ button.
4. Update Account form is loaded.
5. Fill up given fields and click ‘Update’, a message box will prompt “This will update the account. Do you want to proceed?” if ‘Yes’ button is clicked, a message box will prompt “Record was successfully updated.”.
6. Else, changes will not be save.

**Use case # 2: Register User Account**

**User:** Administrator, Bookkeeper

**Description:** The system must allow the user to create new user account.

**Fit Criterion:** The administrator and bookkeeper should input all the necessary information of the new user on the user account record form. The user can view the newly added accounts for the user in the list of accounts.

Use case scripts:

1. Click ‘User Accounts’ from the Main form.
2. Account form is loaded.
3. Click ‘Add’ button from the Account form to add new account.
4. Add account form is loaded.
5. Fill up all the fields. When done, click “Save” button.
6. A message box will prompt “This will save new account. Do you want to proceed?” if ‘Yes’ button is click, a message box will prompt “New account was successfully saved.” Else, record will not be save.

## Requirement Specification

The Requirement Specification for the Academic Advising Evaluation System for Undergraduate Students outlines the detailed functional and non-functional requirements that the system must meet to fulfill its intended purpose. This document serves as a comprehensive guide for developers, stakeholders, and project managers to ensure that all aspects of the system are well-defined and understood. The specification includes functional requirements, which describe the behaviors and functions of the system, and non-functional requirements, which detail the system's performance and operational criteria.

### Functional Requirements

Functional requirements define the specific behaviors, functions, and capabilities that the system must possess. These requirements ensure that the system performs the tasks needed to support academic advising effectively. Key functional requirements for the system include:

#### Software Functionality

In this section, it shows the modules that the Academic Advising Evaluation System uses to meet the goals and objective in developing the application.

##### Login and Logout

This feature allows the user to gain access to the application. All user level can access the application provided they have their username and password created. Logout will allow the user to exit the application.

##### Student Profile

This feature allows the user to create, update, and search student profile.

* *Create Student Profile*

This sub feature allows the user to add new student profile.

* *Update Student Profile*

This sub feature allows the user to update existing student record.

* *Search Student Profile*

This sub feature allows the user to search a specific student record.

##### Academic Evaluation

This feature will generate the subjects for next semester based on the student’s current academic grades.

It will also provide the list of subjects the student might need to take based on the assumption that the student has an irregular status or has pending subjects that needs to be completed to finish the course requirement.

#### User Characteristics

* Academic Advisor

The academic advisor is the primary user of the application and has the full access to all the functionalities of the application and has the highest access level. The one responsible for helping students to choose what major and minor subjects to enroll and ensuring that they meet all the requirements to complete the academic program.

* Administrator

The administrator has the full access to all the functionalities of the application and has the highest access level.

* Student

The student has the authority to use the application but with limited access to the functionalities of the application.

|  |  |  |  |
| --- | --- | --- | --- |
| **User Role** | **User Name** | **Priorities** | **Features** |
| Key User | Administrator | Highest priority | All Features |
| Primary User | Academic Advisor | Highest priority | Student Profile Management  Academic Evaluation  Prospectus Management |
| Secondary User | Student | Medium priority | Academic Standing  Course Enrollment Plans  QR Code Scanning |

### Non-Functional Requirements

Non-functional requirements define the system's performance and operational criteria, ensuring that it meets quality standards and provides a satisfactory user experience. Key non-functional requirements include:

#### Technical Requirements

There are two types of application that will be developed one is a web-based application and the second one is a mobile application. Since it has a web-based application and an application server, it will be deployed online and it requires an internet connection to access the application.

Android Studio will be used to develop the application for android mobile platform and HTML, CSS and JavaScript for the web-based application and Node JS for the web server of the application. MySQL database will be used to store the data. The android app will be deployed in android mobile phones (at least the operating system version is Kitkat) and for the web application it can be accessed using desktop and/or mobile browsers (preferably google chrome).

The server machine should have a powerful CPU and high-speed internet access so that it can handle multiple server requests at the same time and has high storage capacity. Higher storage space means more request it can handle and bigger workspace per user so higher the storage, better the performance. And the database should handle large volume of data.

#### Performance Requirements

The application deployment will be done in the machine in the cloud, machine should have a powerful CPU and high-speed internet access so that it can handle multiple server requests at the same time and has high storage capacity. The database should handle large volume of data.

The application response to users request or command will take about 2 to 5 seconds (depending on some circumstances that may interfere with it). The application response time shall be fast to avoid delay of work flow. It should have the capability that can cater the concurrent or simultaneous access with it. The application shall provide accurate outputs and provide precise data results for the user.

#### Assumptions and Dependencies

Designing this project, the developer(s) make some assumptions related to the software, hardware and the environment. First of all, the program will run on an android mobile phone and a web browser(s). Below are other assumptions made while designing the project:

* Project will follow RAD methodology throughout the development
* The application will utilize REST API architecture
* The developer(s) will write the application in Java for android application, HTML, CSS, JavaScript for web application and SQL scripts for database queries.
* The application should have a database where all the data and other information will be saved
* The application will use the existing environment for testing
* The application must meet user’s needs
* The application must be easy to interact
* The application must give the user’s correct information for better decision making

#### Security Requirements

Given that the application will be hosted on a cloud server, data security and user privacy are paramount. Several measures are implemented to ensure the security of user data and to prevent unauthorized access:

**User Authentication and Role-Based Access Control:** A secure login mechanism is in place to prevent unauthorized access. User access varies depending on their role (academic advisor, administrator, or student), following the principle of least privilege to ensure users can only access data relevant to their role.

**Data Encryption:** All user credentials and sensitive data are encrypted both in transit and at rest to protect against data breaches.

**Input Validation and Error Handling:** The application rigorously validates all user inputs to prevent malicious data from being entered. This includes protecting against SQL injection attacks by using prepared statements and parameterized queries. Proper error handling mechanisms ensure that any errors are managed gracefully without exposing system vulnerabilities.

**SQL Injection Protection:** The application uses secure coding practices to defend against SQL injection. This includes validating and sanitizing all inputs, using parameterized queries, and employing ORM (Object-Relational Mapping) frameworks that inherently protect against such vulnerabilities.

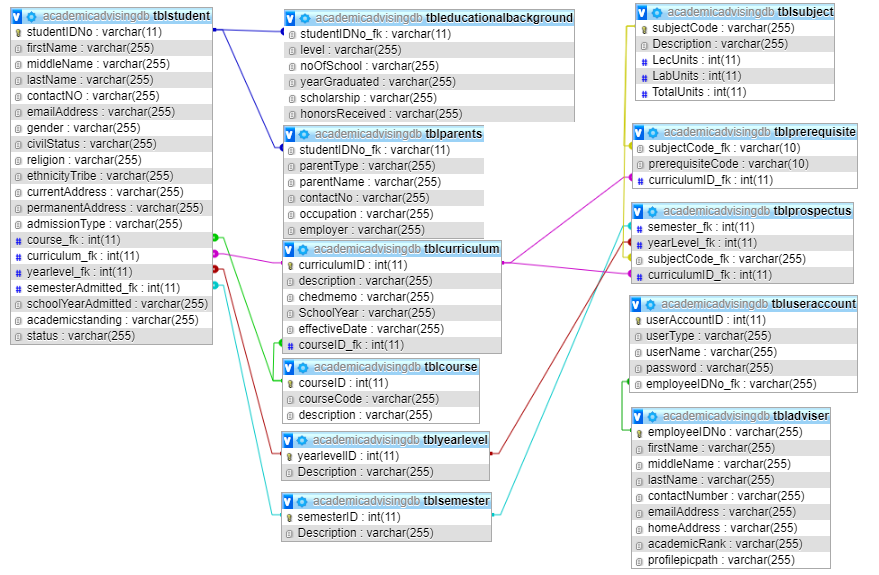
**Access Control:** Access to the system is controlled based on user roles, ensuring that users can only view and interact with data pertinent to their responsibilities. This minimizes the risk of data being accessed by unauthorized users.

These security measures ensure that the Academic Advising Evaluation System not only enhances the advising process but also protects the integrity and confidentiality of the data it manages.

## System Design

The System Design phase is a crucial step in the development of the Academic Advising Evaluation System for Undergraduate Students. This phase translates the gathered requirements into a blueprint for constructing the system, detailing how the system will be structured and how its components will interact. The design phase involves both high-level architectural design and detailed design of system components, ensuring the final product is robust, scalable, and maintainable.

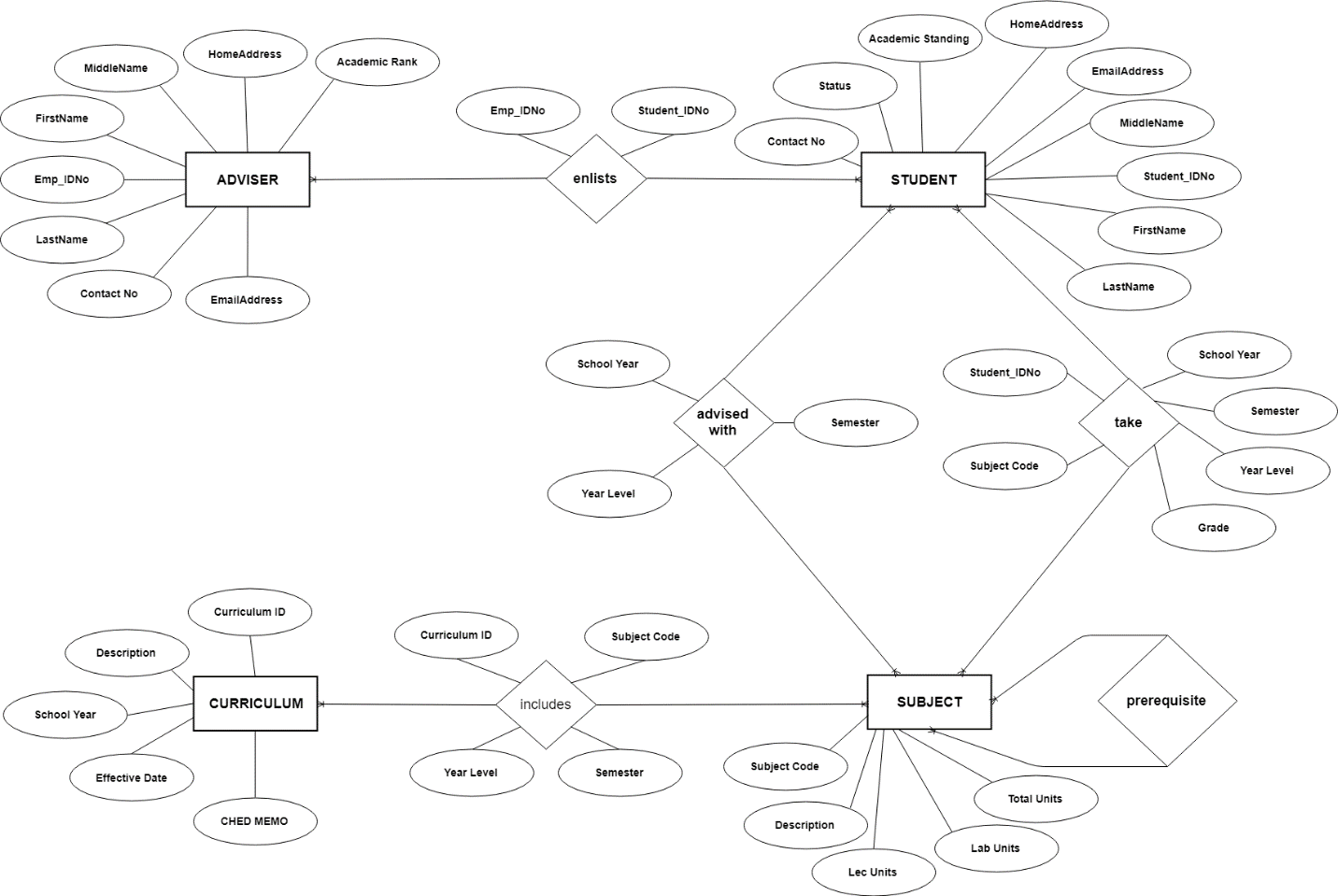
### Database Design



*Figure 4.2*

*(****Explain / Discuss the Database Design****)*

### Entity Relationship Diagram



*Figure 4.2*

*(****Explain / Discuss the ERD****)*

### Database Fields

A field is a data structure for a single piece of data. Fields are organized into records, which contain all the information within the table relevant to a specific entity. To keep the data consistent from one record to the next, assign the appropriate data type to each column. The primary key or the table is also AutoNumber data type, which automatically generates a unique number in each row. A primary key (PK) is a unique identifier for a given entity.

TABLE tblAdviser

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| emp\_IDNo | INTEGER | 11 | **Primary Key** | No |
| firstName | VARCHAR | 255 |  | No |
| middleName | VARCHAR | 255 |  | No |
| lastName | VARCHAR | 255 |  | No |
| contactNo | VARCHAR | 255 |  | No |
| emailAddress | VARCHAR | 255 |  | No |
| academicRank | VARCHAR | 255 |  | No |

*Figure 4.2.1*

TABLE tblStudent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| studentIDNo | INTEGER | 11 | **Primary Key** | No |
| firstName | VARCHAR | 255 |  | No |
| middleName | VARCHAR | 255 |  | No |
| lastName | VARCHAR | 255 |  | No |
| contactNo | VARCHAR | 255 |  | No |
| emailAddress | VARCHAR | 255 |  | No |
| academicStanding | VARCHAR | 255 |  | No |
| status | VARCHAR | 255 |  | No |
| emp\_IDNo | INTEGER | 11 | **Foreign Key** | No |

*Figure 4.2.2*

TABLE tblPrerequisite

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| SubjectCode | VARCHAR | 10 | **Foreign Key** | No |
| PrerequisiteCode | VARCHAR | 10 | **Foreign Key** | No |

*Figure 4.2.3*

TABLE tblCurriculum

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| curriculumID | INTEGER | 11 | **Primary Key** | No |
| Description | VARCHAR | 255 |  | No |
| SchoolYear | VARCHAR | 255 |  | No |
| EffectiveDate | VARCHAR | 255 |  | No |
| ChedMemo | VARCHAR | 255 |  | No |

*Figure 4.2.4*

TABLE tblSubject

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| SubjectCode | VARCHAR | 10 | **Primary Key** | No |
| Description | VARCHAR | 255 |  | No |
| LecUnits | INTEGER | 11 |  | No |
| LabUnits | INTEGER | 11 |  | No |
| TotalUnits | INTEGER | 11 |  | No |

*Figure 4.2.5*

TABLE tblGrades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Max Length** | **Key Type** | **Null** |
| Grade | VARCHAR | 255 | **Primary Key** | No |
| School Year | VARCHAR | 255 |  | No |
| Semester | VARCHAR | 255 |  | No |
| Year Level | VARCHAR | 255 |  | No |
| TotalUnits | VARCHAR | 255 |  | No |
| studentIDNo | INTEGER | 11 | **Foreign Key** | No |
| SubjectCode | VARCHAR | 10 | **Foreign Key** | No |

*Figure 4.2.6*

TABLE tblProspectus

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **Data Type** | **Size** | **Max Length** | **Key Type** |
| SubjectCode | VARCHAR | 255 | **Foreign Key** | No |
| curriculumID | INTEGER | 11 | **Foreign Key** | No |
| Semester | VARCHAR | 255 |  | No |
| Year Level | VARCHAR | 255 |  | No |

*Figure 4.2.7*

## Tools and Technology

*This section outlines the tools, platforms, and technologies used throughout the development and deployment of the system. The choice of tools was based on their compatibility with the project requirements, ease of use, availability of resources, and support for integration and scalability.*

* ***Programming Language****: PHP was used for backend development due to its open-source nature and wide support for web-based applications.*
* ***Database****: MySQL served as the primary relational database management system, enabling efficient data storage, retrieval, and manipulation.*
* ***Frontend Technologies****: HTML, CSS (Tailwind CSS), and JavaScript were utilized to design a responsive and user-friendly graphical user interface (GUI).*
* ***Version Control****: GitHub was used for source code management and collaboration.*
* ***Development Environment****: XAMPP was used as the local server environment for development and testing.*
* ***Others****: AJAX was integrated to enable asynchronous data loading, improving responsiveness; and Google Drive was used to manage and share documentation files.*

*These tools facilitated the efficient development, testing, and deployment of the system, ensuring seamless integration across different modules.*

## Evaluation and Testing

*This section describes the strategies and procedures employed to evaluate the functionality, usability, and performance of the system. The goal was to ensure the system meets user requirements and operates reliably in real-world conditions.*

* ***Unit Testing****: Individual modules and components were tested to verify that each function behaves as expected. This included rule matching logic, database interactions, and API responses.*
* ***Integration Testing****: Modules were combined and tested as a group to check for proper data flow and interaction between the inference engine, user interface, and database.*
* ***User Acceptance Testing (UAT)****: Selected users, such as quality assurance staff or designated evaluators, tested the system in a controlled environment to validate that it meets their operational needs.*
* ***Performance Testing****: The system was evaluated for response time, resource usage, and stability under typical workloads.*
* ***Accuracy and Reliability****: For expert systems or inference-based projects, accuracy of the inference results (e.g., clause-action mapping, rule execution) was cross-checked against predefined expectations or expert-reviewed outputs.*

*Feedback from these evaluations was used to refine the system and ensure it provides reliable support in the context of Quality Management or expert decision-making.*