UPLB Cashier Notification Services: Data Security using Advanced Encryption Standard

Andrea Nicole G. Privado and Concepcion L. Khan

***Abstract*— The study presents an alternative to the SMS notifications currently employed by UPLB Cashier. It succeeded in developing a mobile application for employee users and a web application for admin users. It has also been able to implement end-to-end encryption for mobile notifications and a data encryption algorithm when writing and accessing sensitive data from the database using the Advanced Encryption Standard.**

A mean score of 82.5 (web app) and 92.5 (mobile app) was obtained from the System Usability Testing (SUS) which implies that the system has an above-average to excellent usability. A result of 'Threat Level 1' is obtained from a web vulnerability scan which implies that no severe or critical vulnerabilities were detected in the web application. Lastly, the result 'All tested parameters do not appear to be injectable.' has been obtained from an SQL injection testing performed in one of the server's HTTP request URLs.

***Index Terms*— Mobile Application, Web Application, Encryption Algorithm, Push Notification, In-app Notification**

I. Introduction

A. Background of the Problem

Short Message Service (SMS) alerts are among the conventional notification systems as they can offer a practical and fast method of communication [16]. It is also an ideal notification system if the target recipients are a large number of audiences. Because it requires a lesser advanced and completely no internet connection to receive the alert, it is among the most widely used notification systems.

However, recent news from Inquirer.net last September 2022 stated that scammers have turned to various creative schemes to con other people via SMS Phishing or Smishing. These are personalized text scams. It could be in the form of addressing their victims by their first names instead of simply sending a templated message [11].

Due to the rise in occurrences of this issue, telecommunications providers actively restrict suspicious-looking text messages, particularly those that follow predefined templates. Consequently, individuals may develop a tendency to disregard or question any information received through SMS.

B. Statement of the Problem

One solution for the issues of SMS alerts is to implement mobile app push notifications as an alternative to SMS alerts. Push notifications come from apps installed on a person's phone. Push notifications offer increased security than SMS because of their end-to-end encryption [11]. There are token and certificate-based encryptions in place which ensure that push notifications are sent only to the intended end user.

One of these recognized encryptions for push notifications is the AES or the Advanced Encryption Standard. IOS notifications such as in iMessage use RSA + AES end-to-end encryption to securely send and receive messages from one person to another [6].

C. Significance of the Study

This study offers an alternative solution (mobile app push notifications with end-to-end encryption) to the existing alert system utilized by the UPLB Cashier's Office, specifically addressing the challenges faced by SMS notifications in today's technologically advanced environment, particularly when network providers restrict or discontinue the provision of templated bulk messaging to combat the growing issue of Smishing. Additionally, it presents employees with a mobile app interface to access their notice history, view announcements, and for sending inquiries. For administrators, a web application has been developed to facilitate the management of notice data for the cashier's office and provide a user-friendly interface for sending push notifications to recipients.

D. Scope and Limitation

This study focuses on the implementation of the mobile app push notification as an alternative to the current SMS alerts for the UPLB employees that uses the Advanced Encryption Standard (AES), an algorithm used for data security and protection, to implement end-to-end encryption on the mobile app push notification of the UPLB Cashier Notification Services and the data from its database.

II. Objectives of the Study

The main objective of the study is to implement data encryption on the UPLB Cashier Notification Services.

This study specifically aims to:

1) Develop a mobile application for employee users to receive push notifications, monitor their notice history, and send an inquiry to the cashier's office.
2) Develop a web application for the administrator users to send out push notifications to the recipients and for easy management of data.
3) Implement end-to-end encryption on mobile app push notifications using AES.
4) Implement AES encryption on the data within the database.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

III. Review of Related Literature

SMS Notification System

Short message service (SMS) technology is one of the most stable and widely used mobile communication after phone calls. This is a protocol used in communications that allows short text messages to be exchanged from one mobile phone device to another. SMS alerts are among the conventional notification systems as they can offer a practical and fast method of communication [16]. It is also an ideal notification system if the target recipients are a large number of audiences. Because it requires a lesser advanced and completely no internet connection to receive the alert, it is among the most widely used notification systems.

But messaging has evolved over time, with things becoming more complicated as the number of devices and media formats have emerged. Typically, the relevance of messaging has gone up as companies try to make communications more personalized [6]. But what comes with this personalization is also significant harm to the data privacy of the people involved.

There has been an existing problem, for years, regarding the security of the signaling system used to send SMS. These serious security holes have made the SMS platform less unsafe for sending text messages with sensitive information. There is a possibility that SMS messages can be intercepted and delivered to hackers through different mechanisms: technical mechanisms where the hackers create viruses or malware to be delivered to customer's devices; the technological means wherein the impact of the advanced technologies increases the security risks of SMS; and lastly, social mechanisms where it involves conversing directly to the customer trying to manipulate them to get the sensitive information from the customers [10].

Rise of the Internet and Smartphones

Because of the rapid advancement of technology, people also learned to adapt to these developments. One of these is the increasing demand for smart mobile devices. The number of smartphone users worldwide is 6.648 billion, which means that 83.07% of the world's population owns a smartphone. This is a 49.89% increase in the number of people with a smart cell phone from 2017 to 2022 [4]. In the Philippines, it is estimated that there will be approximately 85 million smartphone users in the year 2022. This is based on the forecast of the Statista Research Department last 2021 from the results of their survey regarding smartphone users from 2017 to 2020. By 2025, it was predicted that there would be about 91.5 million smartphone users in the country, on the grounds that there are approximately 79 million smartphone users in the Philippines in 2020, reflecting an upward trend since 2017 [14]. With these numbers, it is highly observed that people have adapted to the advancements of technology over time and will still continue to progress in the future especially now in this technological era.

Furthermore, people have become so reliant on their smartphones in their daily lives to the point where they treat them as their home. It is suspected that one of the reasons for this is the messaging apps that allow them to stay in touch with their family and friends even when they are from different locations [12]. Because of the wide capabilities of the development of smartphones and mobile apps, it could contribute to the convenience of smartphone users, especially in terms of communication.

But using smartphones and mobile apps has disadvantages, too. One of these is their internet dependence. Most mobile apps rely on an internet connection to function properly, as they send and receive data in real time. These apps act as the interface between the user and the user's data and are not usable without an active internet connection. They are known as native mobile apps [2]. One example of this kind of mobile application is the Facebook Messenger app. In order to send and receive messages, voice calls, or video call another person, a user must be connected to the internet first.

But, even with this disadvantage, there is still a significant increase in Facebook Messenger users in recent years. This is because the public's access to internet connection is also developing along with the advancement of technology. In January 2022, there were 76.01 million internet users in the Philippines, representing a 68.0% penetration rate among the total population [9].

Mobile App Push Notifications

Because of the problems encountered with SMS notification systems today and the widespread development of smartphone and mobile apps in present, mobile app push notifications can be a viable alternative to SMS alerts.

Push notifications are messages that appear on a user's mobile device or desktop to alert them of an event or update related to a specific app. These notifications can be compared to SMS text messages or local mobile alerts, but they are specific to the app being used. Users have the option to disable push notifications in their device's settings, but they are important for many apps so it is important for users to be selective when choosing which apps to allow push notifications for. The technology behind push notifications, known as "server push," involves the server communicating with the client about a specific notice or notification [13].

There are three types of push notifications: mobile app push notifications, web push notifications, and desktop push notifications. Here are some free open-source mobile push notification servers and tools for developers who want to use open-source tools for better control which also will ensure privacy and lower cost [13]:

1) *Uniqush*

   Uniqush is a free and open-source tool that allows you to send push notifications to mobile apps from a server. By using Uniqush, you can send notifications to any mobile platform that is supported by the service.

2) *Pushkin*

   Pushkin is a free and open-source tool for sending push notifications that was designed for speed and flexibility. It was originally developed for use with mobile games, but can be used for any type of application. Pushkin supports both Android and iOS

platforms and is responsive to different services, such as game servers or databases, that can send HTTP POST requests.

3) Pushy

Pushy is a template for integrating push notifications using the Apple Push Notification Server (APNs) into a Swift iOS client. While it has not been updated in three years, Pushy is still a useful resource for iOS developers looking to implement push notifications in their apps.

4) Pigeon

Pigeon is a push notification tool for iOS and Android that is specifically designed for use with the Elixir programming language. It is widely supported and used by many developers. Pigeon is a popular choice for implementing push notifications in Elixir-based projects.

5) Android PN

This project consists of a client and server, both of which are available as free and open-source projects. While the project has not received any updates in recent years, it is still considered to be stable.

6) Pushd

Pushd is a universal mobile push notification tool that allows you to send push notifications to any supported mobile platform, web app, or HTTP server through a single entry point. It is easy to install, configure, maintain, and extend, and is designed to support an unlimited number of subscribable events.

7) AirNotifier

AirNotifier is a user-friendly and powerful application server that allows you to send real-time notifications to mobile and desktop apps. AirNotifier also has features such as API access control, support for an unlimited number of devices, logging of activities, access key management, and a web-based dashboard.

8) Gotify

Gotify is a self-hosted push notification service server that is built using the Go programming language for stability and portability. Gotify can be used on Windows and Linux servers and is also available as a fully functional Docker image.

9) Iris

Iris is a self-hosted notification server that does not rely on Google Cloud Messaging, Firebase Cloud Messaging, or Apple Push Notification Service. This project is currently under development and is not yet ready for production use.

10) Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a messaging solution compatible with multiple platforms, enabling the dependable and cost-free delivery of messages.

Security of Push Notifications in Mobile Apps

Leonard (2017) stated that push notifications offer increased security than SMS because of their end-to-end encryption. There are token and certificate-based encryptions in place which ensure that push notifications are sent only to the intended end user and one of these recognized encryptions for push notifications is the AES or the Advanced Encryption Standard.

If push notifications are to be compared with SMS in terms of security, push notification certainly offers more increased protection than SMS. This is because of the enforcement of end-to-end cryptographic validation on the push notification content. According to Apple support, IOS push notifications, such as in iMessage, use RSA to encrypt the receiving device's public key, and the AES algorithm encrypts the content of the messages in order to send and receive notifications from one person to another securely.

Data Security using AES Algorithm

According to Federal Information Processing Standards (FIPS), AES, or the Advanced Encryption Standard, is a standardized cryptographic algorithm that can be used to secure electronic information. It uses a symmetric block cipher, which means it can both encrypt and decrypt data. When data is encrypted, it is converted into a code called cipher text that cannot be easily understood. Decrypting the cipher text converts the data back into its original form, known as plaintext. AES can use keys of various lengths, such as 128, 192, and 256 bits, to encrypt and decrypt data in blocks of 128 bits.

With this, AES is highly regarded as the most secure symmetric encryption algorithm in the world and is widely used in consumer devices and applications, including SSDs for data storage, Google Cloud Storage, internet browsers like Firefox and Opera, and website security certificates. Many popular apps, including Snapchat and Facebook Messenger, use AES encryption to securely transmit information like photos and messages. AES is also used in well-known file compression programs like 7z, WinRAR, and WinZip to prevent data breaches, and can be found in the libraries of programming languages like Java, Python, and C++ [18].

IV. Materials and Methods

A. Architecture and Technologies

Tech Stack

The tech stack for this project encompasses various technologies to ensure a seamless and efficient development process. On the front end, *React Native* is employed for the mobile application, enabling the creation of a cross-platform experience with its flexibility and reusability. *React JS*, on the other hand, powers the web application front end, offering a robust and interactive user interface. Moving to the back end, *Express JS* serves as the foundation, providing a lightweight and flexible framework for building

scalable server-side applications. The runtime environment, *Node JS*, forms the backbone of the entire stack, enabling server-side JavaScript execution and facilitating smooth communication between the front end and back end. Finally, the *MySQL* database is chosen to handle the storage and retrieval of data, ensuring reliability and efficient querying capabilities. With this comprehensive tech stack, the application can deliver a powerful and smooth experience to its users.

Software Framework

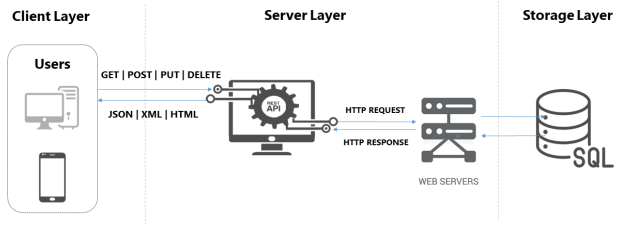

Fig. 1: Software Framework of the UPLB Cashier Notification Services

Figure 1 shows the three layers of the system namely the client, server, and storage layer. This framework visualizes how the data will be sent from the end-users to the database and vice versa. It follows a client-server architecture, where the clients are the admin users (web application) and the employee users (mobile application). The server side of the system is built using Node JS. It handles all the HTTP requests from the web and the mobile application.

Entity Relationship Diagram

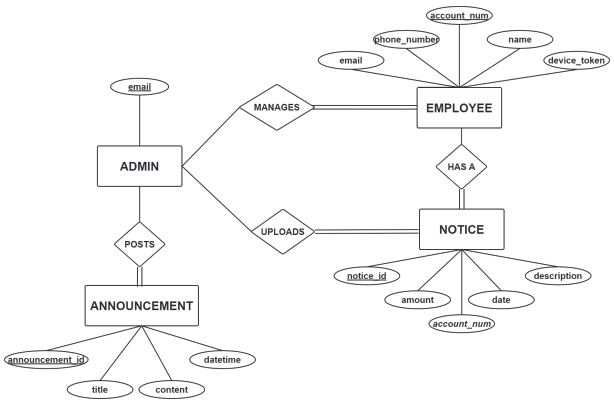

Fig. 2: ERD of the UPLB Cashier Notification Services

Figure 2 depicts the relationships between entities (or objects) within the database and the attributes associated with those entities. The relationships between entities in an ERD are represented by connecting lines between the related entities. Based on the diagram, a user, who has an email address, can manage employees, upload notices, and post announcements. Meanwhile, employees have notices and their own registered devices distinguished by the device_token.

Use Case Diagram

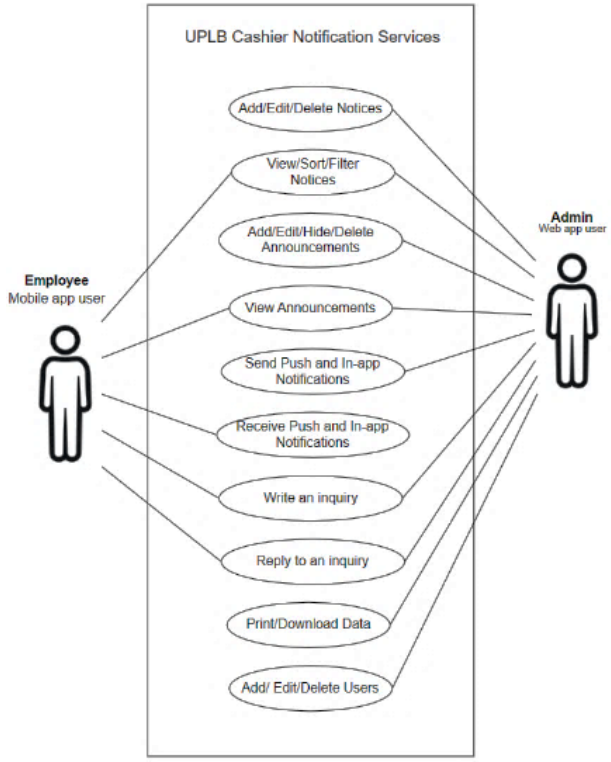

Fig. 3: Use Case Diagram of the UPLB Cashier Notification Services

Figure 3 shows the system's users (employees and admins) and their particular interactions in the system. It also shows what functionalities are available to them when they access the system.

B. User Acceptance Testing

User Acceptance Testing or UAT is a process in which the software system is tested by the end-user or client to ensure that it meets their requirements and is ready for use in the production environment [7]. The UAT has been performed on both the mobile and web application.

Participants

The population for the UAT of the mobile and web application are be UPLB employees. The sampling method used was purposive sampling. This type of sampling is particularly useful when the goal is to quickly reach a specific sample group, rather than to ensure proportional representation within the sample [3]. It can be useful in situations where the main focus is not on proportionality [3].

The participants of the mobile app UAT was selected from the employees and staff in UPLB. This is mainly because they could give useful feedback and suggestions with regard to the overall usability of the mobile application since they are the target users of the system.

Meanwhile, the participants for the web application UAT was selected from UPLB Cashier employees, specifically

those who are directly involved with the operations and tasks using the web application.

Data collection

For the data collection proper, the system usability scale (SUS) is the metric used to perform the UAT. SUS is the most used questionnaire for measuring perceptions of usability. It has become a standard in the industry with over 600 publication references [15].

The SUS is a ten-item questionnaire with five-item possible responses from strongly disagree (1) to strongly agree (5). The survey questionnaire was physically printed and given to participants in person.

Data Analysis

To calculate the SUS score based on the data gathered. The following are to be performed:

1) For odd-numbered items, subtract 1 from the user response.
2) For even-numbered items, subtract the user response from 5.
3) This transforms all values to a range of 0 to 4, with 4 being the most positive response.
4) Add up the converted responses for each user and multiply the total by 2.5 to get the final score, which will be on a scale of 0 to 100 rather than 0 to 40.

SUS scores above 68 are considered above average, while scores below 68 are considered below average [15].

C. Web Vulnerability Scan

A web vulnerability scan is conducted as a fundamental component of the data security testing process. This scan involves systematically assessing a web application for potential vulnerabilities and weaknesses that could be exploited by malicious actors. The primary objective of a web vulnerability scan is to identify security flaws that may exist within the application's infrastructure, codebase, or configuration.

During a web vulnerability scan, specialized software or tools are employed to automatically crawl and analyze the target application. These tools simulate various attack vectors, such as cross-site scripting (XSS), cross-site request forgery (CSRF), and other common web-based vulnerabilities. By examining the application's response to these simulated attacks, the scan can identify potential vulnerabilities that could be leveraged by attackers to compromise the system or gain unauthorized access to sensitive data [21].

The software employed for conducting the web vulnerability scan is Acunetix. Acunetix is a powerful and comprehensive web application security testing tool designed to identify and mitigate potential vulnerabilities and weaknesses in web applications.

Acunetix incorporates a wide range of advanced scanning techniques and algorithms to thoroughly analyze the target application for security flaws. It performs an in-depth examination of the application's architecture, codebase, server configurations, and external dependencies to uncover potential vulnerabilities that could be exploited by malicious actors [22].

Acunetix offers a user-friendly interface that allows security professionals and developers to easily configure and customize the scanning process. It provides options to specify target URLs, authentication parameters, and custom scanning policies based on industry best practices and compliance requirements.

Furthermore, Acunetix generates comprehensive reports that outline the identified vulnerabilities, their severity levels, and detailed recommendations for remediation. These reports assist security teams and developers in prioritizing and addressing the detected vulnerabilities in a timely manner.

D. SQL Injection

In order to ensure the robustness of data security, an SQL Injection Testing is conducted as an essential part of the security assessment process. SQL injection is a malicious technique commonly employed to exploit vulnerabilities in data-driven applications. It involves the insertion of malicious SQL statements into input fields, aiming to manipulate the application's database and gain unauthorized access or extract sensitive information [19].

The tool to be used in SQL Injection Testing is SQLMap. SQLMap is a powerful and widely used tool specifically designed for conducting SQL Injection Testing. It is an open-source penetration testing tool that automates the process of identifying and exploiting SQL injection vulnerabilities in web applications. Its primary purpose is to assist security professionals and penetration testers in assessing the security of web applications by simulating real-world SQL injection attacks. It provides a comprehensive set of features and functionalities that streamline the testing process and enable efficient vulnerability identification [20].

V. Results and Discussion

A. System Development

The web application and the mobile application are developed using the same framework and runtime environment, Node JS, so that data processing and operations will be easier and the modules and libraries to be used are guaranteed to be compatible with each other such as the cryptography module of Node.

The development of both applications is simultaneous which means that each development phase is per module for both mobile and web. The development and testing process involves running the server and the database on a Digital Ocean droplet. The public IP address of the droplet is used to fetch URLs of the web and mobile app HTTP requests.

The system's design focused on simplicity, usability, and efficiency. The user interface was designed to be intuitive, with clear navigation and minimalist design elements. The server-side components were designed to handle a high volume of requests and process them in a timely manner.

For the authentication of admin users and mobile app users, the Google Sign-in method is implemented since most of the UPLB employees and staff has email address. Security limitations lie on the admin users on which email addresses they would recognize to be signed in on the mobile app.

B. Mobile Application

A mobile application has been developed specifically for employee users, providing them with a convenient platform to stay updated with notices and announcements from the Cashier's office through mobile notifications, and an interface to submit inquiries directly to the Cashier's Office. The application offers several key features designed to enhance communication and provide easy access to important information. This streamlined communication and information access contribute to an enhanced user experience, ensuring that employees are well-informed and have a convenient means to interact with the Cashier's office.

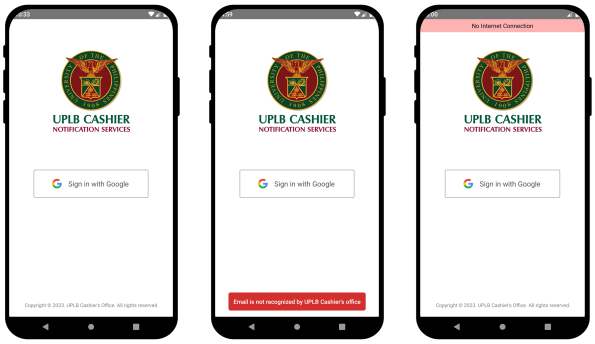

Fig. 4: Login Page

User login implements the Google sign-in method. For security purposes, only those whose email addresses are added to the database (by the Cashier's Office employee) are the ones who can log in to the mobile app. Otherwise, a notification that says 'Email is not recognized by UPLB Cashier's Office' will be shown. Additionally, a device that is not connected to the internet cannot also log in. A prompt regarding connection will also be shown at the top of the app.

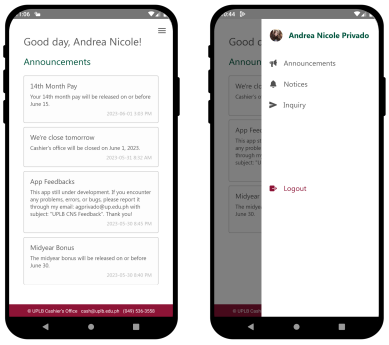

Fig. 5: Announcements Page/Front Page

Once logged in, the user will be navigated to the Announcements page which is the front page of the mobile app. This is where the announcements from the UPLB Cashier's Office will be displayed. The navigation menu, a hamburger menu icon, is located at the upper right side of the screen. From this, the user can navigate to the Account Information page, Announcements Page, Notices Page, and Inquiry Page. The logout button is also found on this menu.

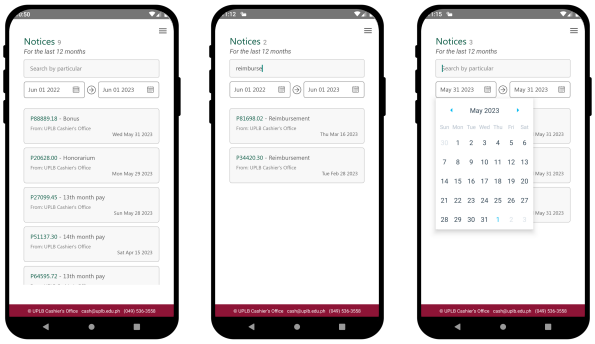

Fig. 6: Notices

The notices page displays a list of all the notices that are under the user's account number regardless of whether it is successfully sent to the device or not. The user can search notices by description and filter them by a chosen date range.

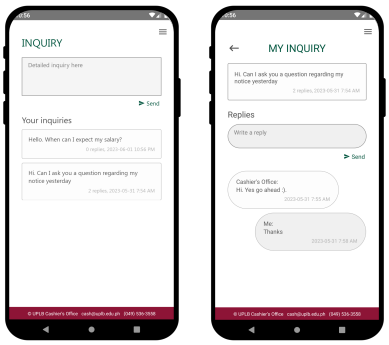

Fig. 7: Inquiries/Individual Inquiry View

The inquiry page provides the users with an interface to send an inquiry to the cashier. It also displays all the past inquiries of the user and an individual view to see the replies. The user is also provided with an interface to write a reply under a particular inquiry.

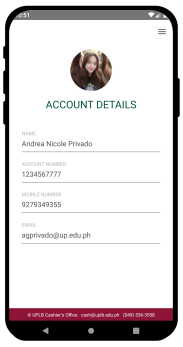

Fig. 8: Account Information

The account information page can be accessed by tapping the name or the photo of the user from the navigation menu. It shows all the information about the signed-in user.

C. Mobile Notifications

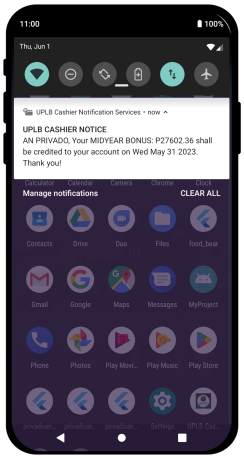

Fig. 9: Push Notifications

Push notifications are messages or alerts that are delivered to users' devices, such as smartphones, tablets, or computers, from a server or application. They are used to provide timely information, updates, or reminders to users even when they are not actively using the application. With push notifications, the system is able to deliver personalized and targeted messages to the mobile users, enhancing the overall user experience. They can be received when the user is signed in on the app but the app is closed. It can appear as a banner notification and a status bar notification or either of the two depending on the user's phone settings.

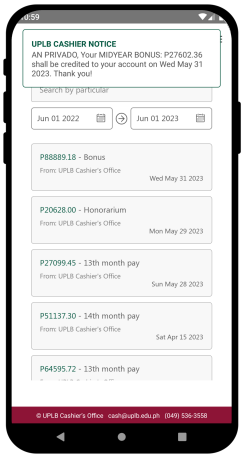

Fig. 10: In-app Notifications

On the other hand, In-app notifications are messages or alerts that are delivered directly within a mobile application to engage and communicate with users. Unlike push notifications that are sent from a server, in-app notifications are generated and displayed within the application itself. In-app notifications can be received when the user is signed in on the app and the app is opened. It will automatically appear as a banner notification as set by the developer. Their design, colors, and font can be customized.

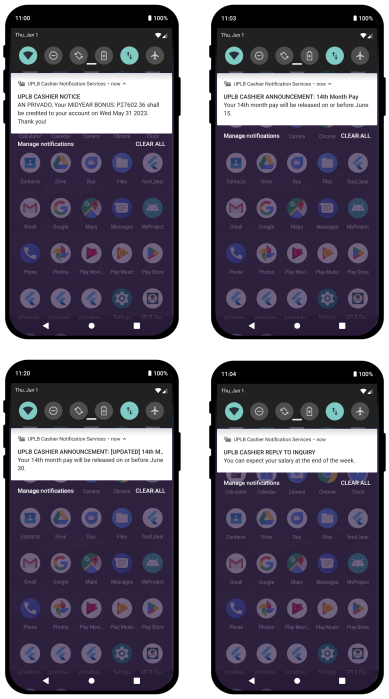

Fig. 11: Notifications Categories

The notifications that users receive can be classified into four categories: new announcements, updated announcements, individual notices, and replies to inquiries. The header or title of each notification reflects its specific category. These categorizations apply to both push notifications and in-app notifications.

D. Web Application

A web application has been developed specifically for administrator users, offering them a platform to perform various tasks and manage data efficiently. The primary functionalities of this application include the ability to send out push notifications to recipients, post announcements, and respond to inquiries.

It also serves as a comprehensive management tool for data. Administrators can organize and manage various types of data within the application, such as user information, announcements, and inquiries. This feature contributes to effective data administration, ensuring easy access, retrieval, and manipulation of information as needed.

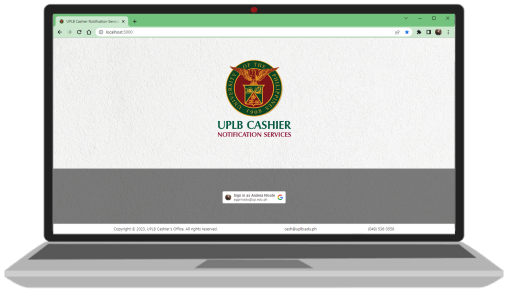

Fig. 12: Login Page

The web app also uses the Google Sign-in method for its sign-in privileges. Only the acknowledged email addresses are allowed to sign in on the web app. Otherwise, a prompt will be shown.

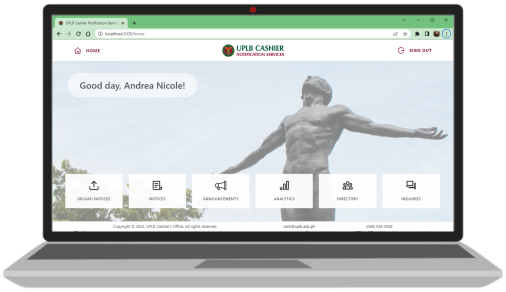

Fig. 13: Homepage

After a successful sign-in, the user will be navigated to the homepage. On this page, the user can navigate through other pages by clicking the boxes below. The user can sign out anytime anywhere by clicking the sign-out button on the interface' header.

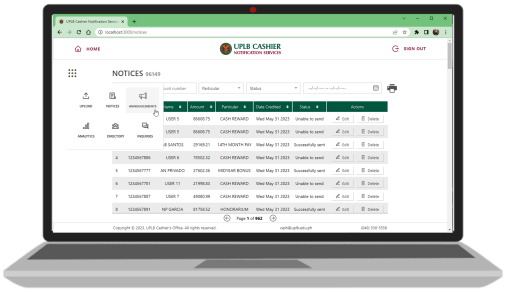

Fig. 14: Navigation Menu

Once the user navigates to other pages. A 9-dot navigation menu will be displayed at the top-left corner of the interface.

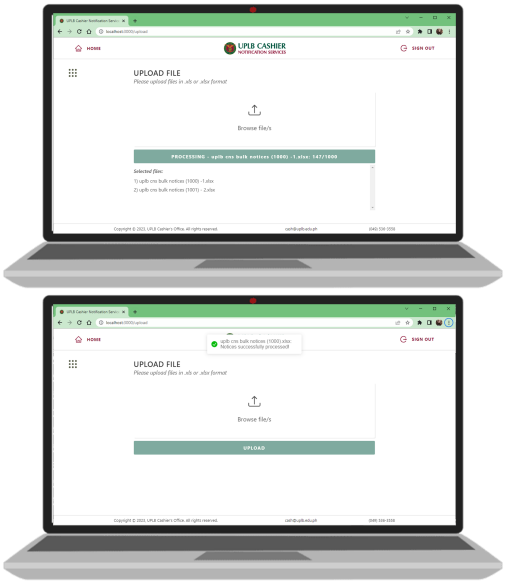

Fig. 15: Upload File Page

On this page, the user can upload files that are in .xlsx or .xls format. The file contains the individual notices of the recipients in a specified valid format. It accepts batch file uploads and shows the progress per file in the green button as shown above. It is also implemented with file error catching wherein invalid file types and invalid table formats are not accepted. After the uploading process, the individual notices will be sent out as notifications (whether push or in-app) to the respective recipients.

	A	B	C	D	E
1	ACCOUNT NUMBER	REGISTERED NAME	AMOUNT	PARTICULAR	DATE CREDIT
2	1234567881	USER 1	36998.93	MIDYEAR BONUS	6/9/2023
3	1234567882	USER 2	93427.55	MIDYEAR BONUS	6/9/2023
4	1234567883	USER 3	87696.52	MIDYEAR BONUS	6/9/2023
5	1234567884	USER 4	32942.28	MIDYEAR BONUS	6/9/2023
6	1234567885	USER 5	86608.75	MIDYEAR BONUS	6/9/2023
7	1234567886	USER 6	78502.32	MIDYEAR BONUS	6/9/2023
8	1234567887	USER 7	49080.99	MIDYEAR BONUS	6/9/2023
9	1234567888	USER 8	55211.42	MIDYEAR BONUS	6/9/2023
10	1234567889	USER 9	70135.91	MIDYEAR BONUS	6/9/2023
11	1234567790	USER 10	53414.51	MIDYEAR BONUS	6/9/2023

Fig. 16: Sample Input File

Figure 16 illustrates an example of how the input file should be saved. The Excel file should consist of five columns, each labeled with corresponding headers. The purpose of having a standardized file format with specific column headers is to ensure consistent data organization and facilitate seamless data processing within the system. By adhering to this structure, the system can accurately interpret and process the data contained in the uploaded file.

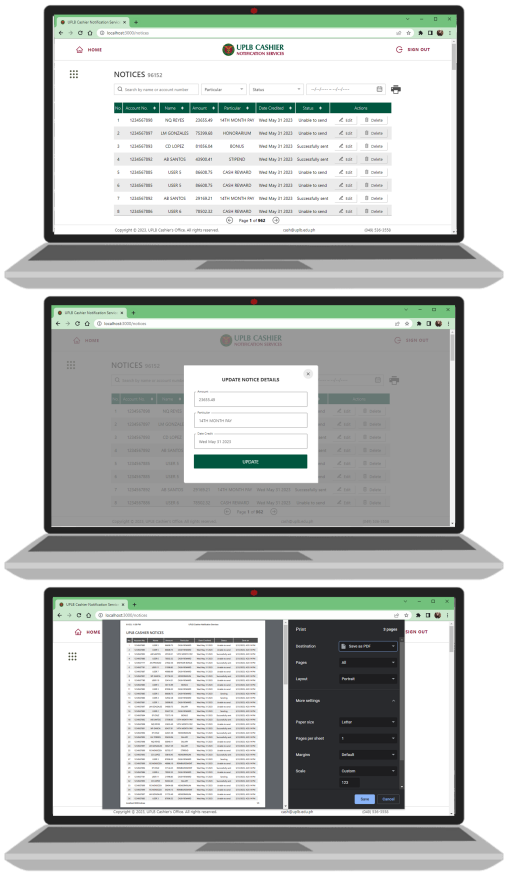

Fig. 17: Notices Page

On this page, the uploaded notices are displayed. It is implemented with pagination that is up to 100 notices per page

since the data is essentially large. The user can also search for notices by account number and name. He can also filter the notices according to the 'particular' field or the description field, the status of the notice whether it is successfully sent or not, and by date range. The user can also order the notices based on the preferred column. He can also perform edit and delete operations for each notice. If the status of the notice is 'Unable to send', once the user hovers over it, it will show the reason why it did not proceed. Ex. 'Account not signed in'. If the status is 'Successfully sent', once the user hovers over it, it will show the timestamp when it is sent. An additional feature is the option to print or save to pdf the displayed notices.

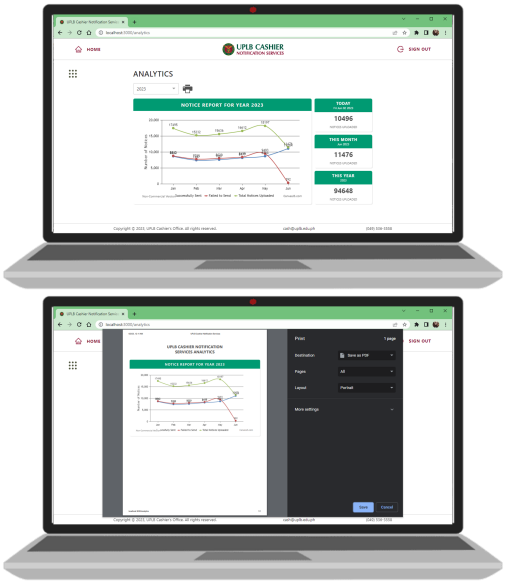

Fig. 18: Analytics Page

This page shows a graph about the notice report per year. The data points included are the successfully sent notices, failed to send notices, and the total notices uploaded for each month of the year. The graph can be changed based on a chosen year. It also shows a report of the total uploads for the day, the current month, and the current year. The graph of the yearly notice report can also be printed or saved as a pdf.

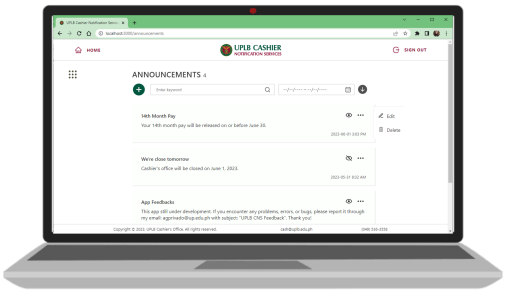

Fig. 19: Announcements Page

On this page, the created announcements are displayed. The user can search for announcements by keyword. He can also filter the announcements by date range. The user can also order the announcements from oldest to latest or vice versa. A new announcement is created by clicking the green plus sign circle button at the top-left. The user can also perform hide, edit, and delete operations for each announcement.

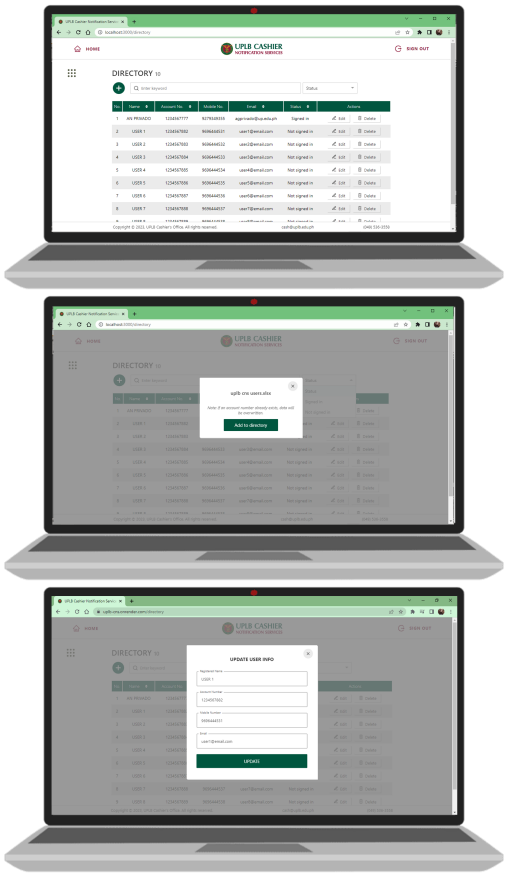

Fig. 20: Directory Page

On this page, the added mobile users are displayed. To add mobile users, it is also by uploading a .xlsx or .xls file with the same file error catching applied with the notice upload. The user can search for mobile users by email, name, or account number. He can also sort the mobile users according to the 'status' field whether they are signed in or not. The user can also order the mobile user based on the preferred column. He can also perform edit and delete operations for each mobile user.

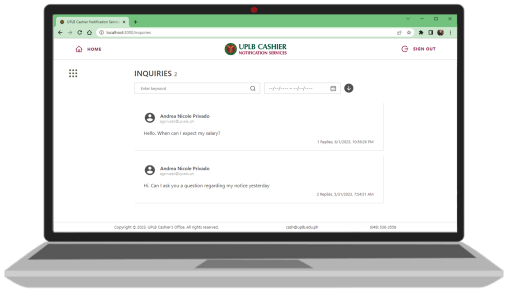

Fig. 21: Inquiries Page

This is where the user can view the inquiries sent by the mobile users. He can search an inquiry by keyword or filter them by date range. He can also order them from newest to oldest or vice versa.

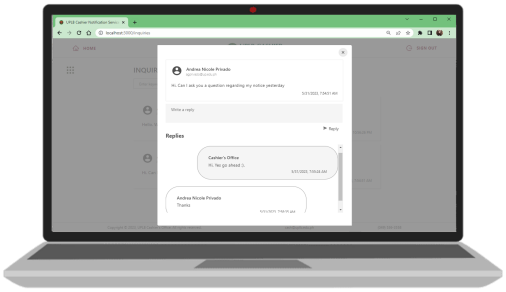

Fig. 22: View Inquiry

A user can click on a particular inquiry if he wishes to view the replies or write a reply to it.

E. System Usability Testing Result

The web application's usability is tested through System Usability Scale (SUS). This assessment is designed to measure the usability of the application by providing responses to statements using a five-point scale that spans from "Strongly Disagree" to "Strongly Agree."

The SUS statements used are the following:

1) I think that I would like to use this application frequently.
2) I found the application to be overly complex without justification.
3) I thought the application was easy to use.
4) I think that I would need the support of a technical person to be able to use this application.
5) I found the various functions in this application were well integrated.
6) I thought there was too much inconsistency in this application.
7) I would imagine that most people would learn to use this application very quickly.
8) I found the application very awkward to use.
9) I felt very confident using the application.
10) I needed to learn a lot of things before I could get going with this application.

After testing, the web component of the system obtained a mean score of 82.5 based on Figure 23 which indicates that the web application has above-average to excellent usability. In addition to scoring, testers were also requested to provide comments and suggestions on potential areas for improvement in the web application.

Respondent	Computed Odd	Computed Even	Score
1	17	15	80
2	19	15	85
Mean Score			82.5

TABLE I: Web Application SUS Score

The potential user/admin of the web application from the Cashier's Office is only one. The second respondent is her supervisor. Both of the respondents gave positive feedback on the overall web application, especially on the functionalities such as the effective searching, filtering, and ordering of data and the choice of color and clean design of the user interface. They also provided points for improvement and helpful recommendations during the testing. One of these is adding the quick reply feature in the inquiries page or an AI to reply for similar inquiries as it is quite a work if they would be responding to them one by one.

Meanwhile, the mobile component of the system obtained a mean score of 92.5 based on Figure 24 which indicates that the mobile app has above-average to excellent usability. Testers were also asked to offer feedback and recommendations regarding possible enhancements in the mobile application.

Respondent	Computed Odd	Computed Even	Score
1	17	20	92.5
2	20	14	85
3	19	20	97.5
4	20	20	100
5	17	8	62.5
6	20	20	100
7	20	20	100
8	20	19	97.5
9	20	16	90
10	20	20	100
Mean Score			92.5

TABLE II: Mobile Application SUS Score

The respondents of the mobile application user testing are UPLB employees and staff. Three of them are from the Learning Resource Center (LRC), four of them are from the Institute of Computer Science (ICS), and three from the Institute of Mathematical Sciences and Physics (IMSP). All of the respondents gave positive feedback on the overall mobile application, especially on the simplicity of the app's functionalities, the intuitiveness of the user interface, and its usability. During the testing phase, some participants shared valuable suggestions for improvement, including the recommendation to incorporate mobile numbers as an additional sign-in method for mobile app users and to ensure the availability of the mobile application on iOS devices.

F. Improved Efficiency

As per the individual responsible for sending out SMS notifications, the traditional process of sending 2000 SMS messages can be quite time-consuming. It typically involves a waiting period of approximately 2 hours before all the messages are sent out. Moreover, this process is often interrupted whenever a reply is received, further prolonging the overall time taken.

However, with the implementation of the UPLB Cashier Notification Services, the efficiency and speed of sending out 2000 notices have significantly improved. On average, it now takes only around 5 minutes and 10 seconds to complete the entire sending process. This drastic reduction in time is a notable achievement.

Furthermore, one of the key benefits of this new system is the immediate receipt of notifications by the intended recipients. Unlike the traditional approach, where delays can occur due to various factors, the UPLB Cashier Notification Services ensure that the notifications reach the recipients promptly.

The implementation of this system has brought about a remarkable improvement in terms of time efficiency and the

delivery of notifications. The reduced processing time and the elimination of delays contribute to an enhanced user experience and increased operational effectiveness for the organization responsible for sending out the notices.

G. System Performance and Scalability

The system is designed to handle a large volume of data, specifically bulk notices, which can reach up to approximately 50,000 notices per month. To ensure that the system is capable of efficiently storing and managing this amount of data, extensive testing has been conducted.

The system has undergone successful testing to validate its performance and scalability. One of the tests involved storing a substantial amount of data, exceeding 100,000 records, within the system's database. Notably, a significant portion of these records contained encrypted data, ensuring the system's capability to handle such data securely.

Additionally, the system was subjected to a substantial workload during testing, simulating real-world usage scenarios. This included handling a considerable number of requests, ranging between 40,000 to 50,000, specifically related to uploading notices and sending out push notifications. The system demonstrated its efficiency and reliability by effectively managing and processing this high volume of requests.

By successfully handling the storage of a large dataset and efficiently managing a significant number of requests, the system has exhibited its robustness and ability to scale. These test results instill confidence in the system's capacity to handle demanding workloads and ensure smooth operations even under heavy usage conditions.

H. Data Security using AES

To ensure the security and protection of sensitive data, the UPLB Cashier Notification Services employ data encryption techniques. When storing sensitive information in the database, a robust encryption algorithm called aes-256-gcm is utilized. This algorithm is widely recognized as one of the strongest encryption methods available to date, offering a high level of data confidentiality and integrity [18].

In addition to database encryption, end-to-end encryption is implemented for the mobile app notifications. This step is taken because Firebase Cloud Messaging (FCM), the platform used for sending notifications, does not inherently support end-to-end encryption. By incorporating end-to-end encryption (AES) within the mobile app, the UPLB Cashier Notification Services ensure that the data transmitted between the server and the app remains secure and protected throughout the entire communication process.

The utilization of aes-256-gcm encryption for sensitive data storage and the implementation of end-to-end encryption for mobile app notifications significantly enhance the security measures of the UPLB Cashier Notification Services. These encryption mechanisms provide a robust shield against unauthorized access and safeguard the confidentiality and integrity of sensitive information. By adopting these strong encryption practices, the system effectively mitigates the risk of data breaches and reinforces the trust and confidence of users in the security of their personal data.

The AES algorithm provides a high level of security and has been extensively analyzed and tested by the cryptographic community. It has become widely adopted and is used in various applications to ensure the confidentiality and integrity of sensitive data.

The basic steps involved in the AES encryption process are as follows:

1) Key Expansion: The original encryption key is expanded into a set of round keys that will be used in each round of encryption.
2) Initial Round: The input data is XORed (combined) with the first round key.
3) Rounds: A specified number of rounds (10, 12, or 14 depending on the key size) are performed. Each round includes the following operations:
    - SubBytes: Non-linear byte substitution using a substitution table called the S-box.
    - ShiftRows: Rearrangement of the bytes within each row of the data block.
    - MixColumns: Mixing of the columns of the data block using a matrix multiplication operation.
    - AddRoundKey: XORing the current round key with the modified data block.
4) Final Round: The final round excludes the MixColumns operation to simplify the decryption process.

The output of the encryption process is the ciphertext, which appears as random and unintelligible data. The ciphertext produced by AES is designed to be secure and resistant to cryptographic attacks. It does not reveal any information about the original plaintext or the encryption key used. The strength of the AES algorithm lies in its ability to effectively obscure the original data, making it extremely difficult for unauthorized individuals to decipher without knowledge of the correct decryption key.

The ciphertext generated by AES encryption can be safely transmitted or stored, as long as the decryption key is kept secure. To retrieve the original plaintext from the ciphertext, the AES decryption process is performed using the correct decryption key. The decryption process applies the inverse operations of the encryption process, transforming the ciphertext back into the original plaintext form.

It is important to note that without the correct decryption key, it is computationally infeasible to recover the original plaintext from the ciphertext produced by AES encryption. This property ensures the confidentiality and security of sensitive data protected by AES encryption.

Fig. 23: Data encryption within the database

The data stored and transmitted within the system holds significant sensitivity as it encompasses a range of critical information such as account numbers, transaction details, and user names. Safeguarding this data is of paramount importance to ensure the privacy and security of the individuals involved.

By applying encryption techniques to the system, the security risks associated with storing and transmitting sensitive data are significantly reduced. Encryption transforms the data into an unreadable format using sophisticated algorithms, making it extremely challenging for unauthorized individuals to access or decipher the information without the appropriate decryption keys.

In terms of data storage, encryption ensures that even if unauthorized access occurs, the encrypted data remains protected and incomprehensible. In the event of a security breach or unauthorized intrusion, the encrypted data acts as an additional layer of defense, preventing unauthorized individuals from gaining meaningful access to sensitive information.

Moreover, during transmission or notification sending, end-to-end encryption safeguards the data from interception and tampering. By encrypting the data before transmitting it over networks or channels, it becomes highly resistant to eavesdropping and unauthorized modifications. This ensures the integrity and confidentiality of the data throughout its journey from the sender to the intended recipient.

The implementation of encryption in the system mitigates the security risks associated with storing and transmitting sensitive data. It strengthens the overall security posture of the system, providing assurance to users that their data is being adequately protected. By incorporating encryption as a fundamental security measure, the system upholds privacy, maintains trust, and safeguards the integrity of the sensitive information it handles.

I. Web Vulnerability Scan Result

To ensure the security of the system, a web vulnerability scan is carried out to proactively identify any potential threats or vulnerabilities. Acunetix software is employed for this purpose. During the testing, using Acunetix, a full scan has been conducted which includes a comprehensive scan consisting of over 21,000 security testing requests in 33 minutes. The URL being scanned is the URL of the client side of the Web Application that is currently rendered using an online rendering service.

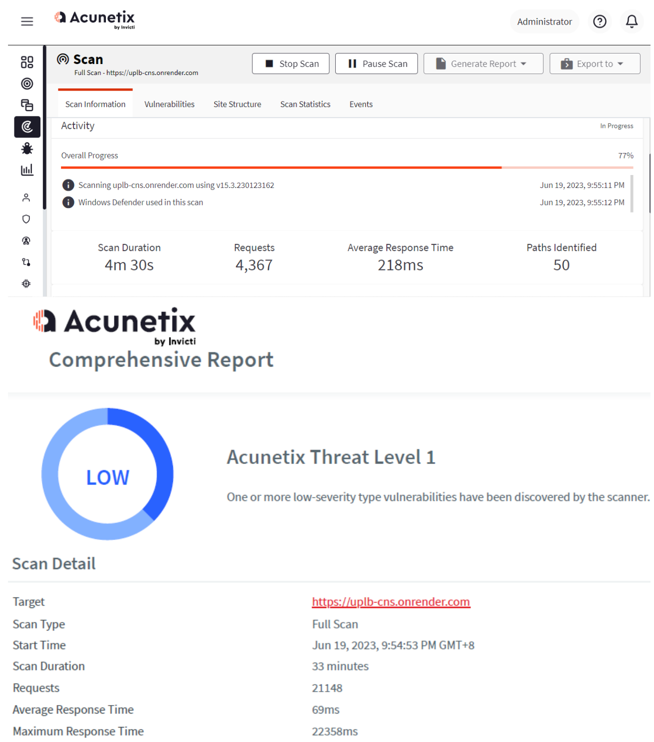

Fig. 24: Web Vulnerability Scan Result

The results of the web vulnerability scan provide valuable insights into the security posture of the system. Fortunately, the scan reveals that the threat level is low, indicating that no severe or critical vulnerabilities were detected during the assessment. This is an encouraging finding, as it indicates that the system has been developed with security measures in place and is resilient against common web-based attacks.

It is important to note that the low vulnerabilities described in the scan result primarily pertain to domain/sub-domain conflicts. These arise due to the system being hosted in the free tier of an online rendering service without having its dedicated domain. While these vulnerabilities are considered low-risk, they are more related to the system's infrastructure and hosting setup rather than inherent flaws within the application itself.

Overall, the results of the web vulnerability scan provide reassurance that the system has undergone thorough security testing and is equipped with appropriate safeguards. By identifying and addressing any potential vulnerabilities, the system can maintain a robust security posture, ensuring the confidentiality, integrity, and availability of the data and protecting against malicious attacks.

J. SQL Injection Result

SQL Injection Testing is carried out as a crucial component of the security evaluation procedure. Using SQLMap, a specified URL with parameters is tested with SQL injection attacks. The tested URL corresponds to one of the HTTP request URLs utilized by the server to retrieve data specifically for the Notices Page.

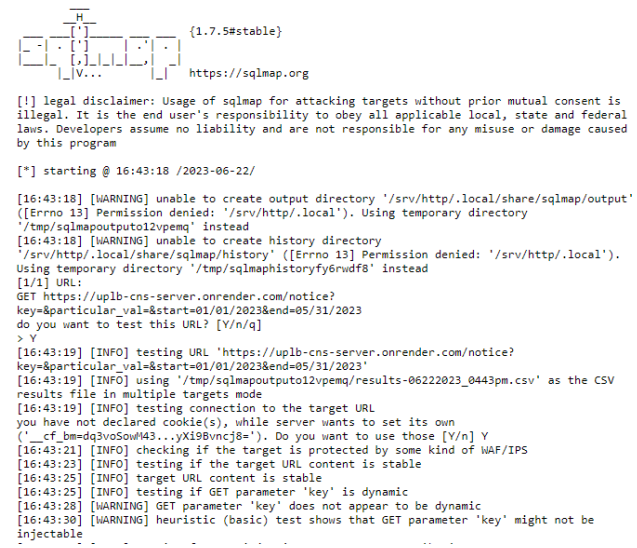

Fig. 25: SQL Injection via SQLMap

The objective of SQL Injection Testing is to identify potential points of weakness where the application's input validation and sanitization mechanisms may be inadequate. By injecting carefully crafted SQL statements into the input fields, the testing process aims to determine if the application is susceptible to unauthorized database access or other malicious activities.

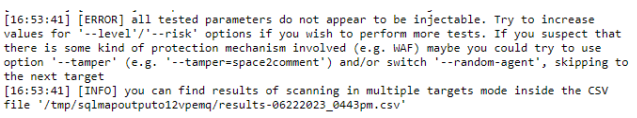

Fig. 26: SQL Injection Testing Result

The result of the SQL Injection Testing, which states that "all tested parameters do not appear to be injectable," indicates that no vulnerabilities were detected during the testing process. This implies that the application's input validation and sanitization mechanisms effectively prevent the execution of malicious SQL statements and mitigate the risk of SQL injection attacks.

This favorable result is encouraging, as it indicates that the tested parameters within the URL demonstrate resilience against SQL injection vulnerabilities. It suggests that the application's developers have implemented robust security measures, such as parameterized queries or prepared statements, to mitigate the risk of SQL injection attacks and ensure the integrity and security of the underlying database.

Nonetheless, it is important to note that regular SQL Injection Testing should be performed as a proactive security measure. This helps to identify any potential vulnerabilities that may arise due to changes in the application's codebase, updates to the underlying database system, or evolving security threats.

By conducting SQL Injection Testing and obtaining results indicating the absence of injectable parameters, the system gains confidence in the security of its data-driven applications. This serves to protect sensitive data, maintain the integrity of the application, and safeguard against potential SQL injection attacks.

VI. Conclusions and Future Work

The system UPLB Cashier Notification Services has been successfully developed and tested to meet its objectives: a web application is developed for admin users, a mobile application is developed for the recipients, end-to-end encryption on the mobile notifications (both push and in-app) is implemented, and the data encryption is implemented when saving sensitive data on the database.

A mean score of 82.5 (web app) and 92.5 (mobile app) was obtained from the System Usability Testing (SUS) which implies that the system has an above-average to excellent usability. A result of 'Threat Level 1' is obtained from a web vulnerability scan which implies that no severe or critical vulnerabilities were detected in the web application. Lastly, the result 'All tested parameters do not appear to be injectable.' has been obtained from an SQL injection testing performed in one of the server's HTTP request URLs.

The implementation of AES encryption in the system mitigates the security risks associated with storing and transmitting sensitive data. It strengthens the overall security posture of the system, providing assurance to users that their data is being adequately protected.

Recommendations

As part of future enhancements, it is recommended to include a mobile OTP (One-Time Password) sign-in method in the mobile application for recipients who don't have an email address. Moreover, considering that a significant number of UPLB staff and employees uses iPhone or iOS devices, it would be beneficial to also deploy the mobile application in an iOS environment to cater to the target audience within the UPLB community. Additionally, it would be valuable to incorporate a quick reply feature or an AI-based response system for admin users on the inquiries page. This feature would streamline the process of addressing similar inquiries, as it can be time-consuming to respond to each one individually.

Deployment

Server and Database

To efficiently host the server and database for both the web app and mobile app, it is advisable to opt for cloud hosting. One recommended approach is to employ a cloud virtual machine (VM) like DigitalOcean Droplet. DigitalOcean provides a low-cost, reliable, and scalable cloud infrastructure that is well-suited for deploying web applications. It provides a user-friendly interface, a console for creating and maintaining the database (MySQL) and for running the server via Express JS. It offers various features to support your deployment needs.

DigitalOcean Basic Droplets		
Price	$6/month	$12/month
CPU	1 vCPU	1 vCPU
RAM	1 GB	2GB
Storage	25 GB	50 GB

TABLE III: DigitalOcean basic droplet pricing

During the development and testing, the $6/month droplet is utilized for running the server and the database of the system. It is successfully tested with storing more than 100,000 data (with most fields containing encrypted data) and with handling around 40,000 to 50,000 requests when uploading notices and sending out push notifications.

Web Application

There are two options for hosting the client-side of the web application: utilizing an online web rendering service or deploying it on a DigitalOcean droplet along with the database and server components.

During the testing, the Render static site rendering service is used. Render is an integrated cloud platform that build and run all your applications and websites. It provides free TLS certificates, a global Content Delivery Network (CDN), DDoS protection, private networks, and automatic deployments from Git. The advantage of this is it is cost-free, the client-side can successfully run in the free tier of the static site rendering. Another is the security of the client-side as it is already provided with TLS certificate and is running in HTTPS. However, there are some drawbacks to hosting the client-side separately from the server. One primary concern is conflicting origins, specifically related to Cross-origin resource sharing (CORS) policies. These measures can sometimes restrict the client-side from sending requests to the server due to the differing origins between the two components. One possible solution to address this issue is to configure the server to allow requests from all origins. However, implementing such a solution can potentially compromise the overall security of the system. Another aspect of concern is the maintenance of the system. With separate hosting environments, it may become more challenging to maintain consistent updates and version control. Ensuring that both the client-side and server-side components are in sync with each other, especially during updates or feature enhancements, can be more complex and time-consuming.

The second option is by deploying the client-side on a DigitalOcean droplet along with the database and server components. It offers a lot of advantages including simplified deployment since updates and new features can be deployed simultaneously, ensuring consistency and reducing the need for coordination between separate hosting environments. Next is performance optimization as it reduces latency and network overhead. Communication between the client and server occurs within the same hosting environment, resulting in faster data transmission and improved overall performance. It also offers enhanced security as it eliminates the need for CORS policies and allows for more fine-grained control over access and data exchange, enhancing overall security measures. However, there is a higher cost involved in hosting the client-side and server-side together. Upgrading from the $6/month droplet to the $12/month droplet is necessary, as it requires a higher machine specification. As indicated in Figure 25, this upgraded droplet offers double the RAM and storage capacity compared to the previous one. With this enhanced configuration, it becomes capable of accommodating both the client-side and server-side of the web application within a single cloud host. The client-side can be accessed either through the droplet's public IP address or via a specified domain that is configured to point to DigitalOcean's nameservers.

Mobile Application

For the development and testing of the mobile app, it is only tested in Android devices using APK. For deploying the mobile application developed in React Native on both Android and iOS devices, the following steps can be performed:

Android Deployment:

1) Generate an APK (Android Package) file for the React Native application.
2) Once the APK file is generated, it can be distributed to users via various methods such as email, app distribution platforms, or direct download links.

iOS Deployment:

1) Generate an IPA (iOS App Store Package) file for the React Native application. For this, you will need an Apple Developer account and a provisioning profile for distribution.
2) Open the Xcode project associated with your React Native application using the following command: open ./ios/YourProjectName.xcworkspace
3) Configure the necessary settings, such as code signing, provisioning profiles, and bundle identifiers in Xcode.
4) Build the app for distribution by selecting a physical iOS device or a simulator, and then choose "Product" ¿ "Archive".
5) Once the archive is successfully created, it can be distributed via the App Store Connect platform for testing or submission to the App Store.

REFERENCES

[1] Apple Support, "How iMessage sends and receives messages securely". https://support.apple.com/fr-fr/guide/security/sec70e68c949/web
[2] Arora, M. (2017, January 17). Does mobile app work without internet: Native mobile apps. Open Source For You. Retrieved December 14, 2022, from https://www.opensourceforu.com/2017/01/mobile-app-without-internet/
[3] Crossman, A. (2020, March 19). What you need to understand about purposive sampling. ThoughtCo. Retrieved December 23, 2022, from https://www.thoughtco.com/purposive-sampling-3026727
[4] December 2022 Mobile User Statistics: Discover the Number of Phones in The World Smartphone Penetration by Country or Region (2022, November 30). BankMyCell. Retrieved December 16, 2022, from https://www.bankmycell.com/blog/how-many-phones-are-in-the-world
[5] Federal Information Processing Standards –FIPS, "Advanced Encryption Standard (AES)". Federal Information Processing Standards Publications. http://csrc.nist.gov/publications/fips/fips197/fips197.pdf, 2001
[6] Friedlein, A. (2022, March 31). Smart notifications: the next evolution of messaging. Ably. Retrieved December 9, 2022, from https://ably.com/blog/smart-notifications-evolution-of-messaging

[7] Hamilton, T. (2022, November 5). What is user acceptance testing (UAT)? Guru99. Retrieved December 23, 2022, from https://www.guru99.com/user-acceptance-testing.html
[8] Hamilton, T. (2022, November 26). What is security testing? types with example. Guru99. Retrieved December 23, 2022, from https://www.guru99.com/what-is-security-testing.html
[9] Kemp, S. (2022, February 15). Digital 2022: The Philippines - datareportal – global digital insights. DataReportal. Retrieved December 16, 2022, from https://datareportal.com/reports/digital-2022-philippines
[10] Leonard, J (2017, May 15). Push Notifications: It is Time to Move on From Using SMS For Service Messages. Business 2 Community. Retrieved December 9, 2022, from https://www.business2community.com/mobile-apps/push-notifications-time-move-using-sms-service-messages-01843056
[11] Mercado, N. (2022, September 8). Smishing: 'Personalized' text scams in the Philippines. INQUIRER.net. Retrieved December 9, 2022, from https://newsinfo.inquirer.net/1660190/watch-smishing-personalized-text-scams-in-the-philippines
[12] Morrison, R. (2021, May 10). People are now so reliant on their smartphones, the devices are 'becoming our homes', experts warn. Daily Mail Online. Retrieved December 21, 2022, from https://www.dailymail.co.uk/sciencetech/article-9561421/People-reliant-smartphones-devices-homes-experts-warn.html
[13] Mousa, H. (2022, January 10). 15 open-source push notification projects, alternative to Apple and google (firebase) services. MEDevel.com: Open-source Guide to Healthcare and Medical Software. Retrieved December 11, 2022, from https://medevel.com/15-os-push-notification/
[14] Philippines: Smartphone users 2017-2026. Statista. (2022, July 27). Retrieved December 11, 2022, from https://www.statista.com/statistics/467186/forecast-of-smartphone-users-in-the-philippines/
[15] Sauro, J. (2011, February 3). Measuring usability with the system usability scale (SUS). MeasuringU. Retrieved December 23, 2022, from https://measuringu.com/sus/
[16] Tasiopoulos, J. (2022, May 4). Using SMS Alerts for Mass Notification. Rave Mobile Safety. Retrieved December 9, 2022, from https://www.ravemobilesafety.com/blog/using-sms-alerts-mass-notification/
[17] Tutorialpoint, "Advanced Encryption Standard. Simply easy learning". All Rights Reserved. http://www.tutorialspoint.com/cryptography/ advanced_encryption_standard.htm, 2016
[18] What is the Advanced Encryption Standard (AES)? Sunny Valley Networks. https://www.sunnyvalley.io/docs/network-security-tutorials/what-is-advanced-encryption-standard-aes
[19] What is SQL injection? tutorial examples: Web security academy. PortSwigger. https://portswigger.net/web-security/sql-injection
[20] Automatic SQL injection and database takeover tool. SQLMap. https://sqlmap.org/
[21] What is web vulnerability scanning?. PortSwigger. https://portswigger.net/burp/vulnerability-scanner/guide-to-vulnerability-scanning
[22] Web application security scanner. Acunetix. (2021, June 22). https://www.acunetix.com/