UIUServiceHub - University Service System Security

1st Iram Ishika
011201153
dept. CSE of United International University
United City, Madani Ave, Dhaka 1212, Bangladesh
iishika201153@bsce.uiu.ac.bd

2nd Khan Mohammad Emon
011201394
dept. CSE of United International University
United City, Madani Ave, Dhaka 1212, Bangladesh
kemon201394@bsce.uiu.ac.bd

3rd Golam Kibria
011201369
dept. CSE of United International University
United City, Madani Ave, Dhaka 1212, Bangladesh
gkibria201369@bsce.uiu.ac.bd

4th Sayem Ahmed
011201379
dept. CSE of United International University
United City, Madani Ave, Dhaka 1212, Bangladesh
mahmed201379@bsce.uiu.ac.bd

***Abstract*—The UIU ServiceHub project aims to consolidate various university services, including Hostel Service, Shuttle Service Registration, Student Loans, Club Forum Discussion, and Posts, into a single, efficient platform for UIU students. Since the project didn't have any security features at first, it was vulnerable to security lapses. The purpose of this project is to solve important security issues while improving the usability and accessibility of university services. Implementing IP Control Access to prevent unauthorized access, password hashing, SQL Injection Prevention, Data Encryption & Decryption, and steps to stop unauthorized activities like taking screenshots or copying private data were among the main goals. Furthermore, we carried out a thorough examination and testing, which included vulnerability assessments and penetration testing, with the goal of raising stakeholders' knowledge of security issues. The development utilized tools such as Git, Visual Studio Code, and Sublime Text, and programming languages including HTML, CSS, JavaScript, MySQL, and PHP. Every stage of the project—from design to implementation—included security, and agile methods were used to ensure prompt and high-quality delivery.**

***Index Terms*—System Security, DoS/DDoS Attack, Hash Fuction, Encryption, Decryption**

I. Introduction

A. Project Background

Our project, UIU ServiceHub builds a one-stop platform for UIU students to access various university services which includes Hostel Service, Shuttle Service Registration, Student Loans, Club Forum Discussion, and Posts. However, accessing these services often involves navigating multiple platforms, leading to inefficiencies and inconvenience for students, therefore incorporating robust security measures. Our goal is to enhance efficiency, accessibility, and security in delivering essential university services to students.

B. Motivation

There were two reasons for choosing this project. First, improving and streamlining the way in which students get university services. We hope to reduce the difficulties involved in obtaining these services by combining all of the UIU offerings into one easily navigable platform. Second, the project gave us a chance to tackle actual security issues in a useful environment, which improved our comprehension of security best practices and concepts. Since the project didn't have any security features at first, it was open to possible security breaches. This emphasized how crucial it is to incorporate strong security measures in order to protect confidential student information and lessen the possibility of unwanted access. Our goal was to prepare for future cybersecurity attempts by bridging the gap between theoretical knowledge and actual implementation through the integration of security elements into an already-existing project.

C. Problem Statement

The project initially lacked robust security features, leaving it vulnerable to potential security breaches and unauthorized access attempts. This created serious threats to the confidentiality and integrity of student information. Furthermore, during the deployment of the DoS Attack Prevention utilizing Captcha, we discovered an issue in which the Captcha image did not display properly. Upon investigation, we discovered vulnerabilities in the code implementation and promptly addressed them, ensuring the effective display of the Captcha image.

1) Challenges Faced: A number of significant security flaws were found during the project's initial assessment, including:

• A deficiency in strong access control systems that allows for unauthorized access attempts.

• Vulnerability to common web-based attacks such as SQL injection, which could jeopardize the database's integrity.

• The lack of encryption techniques, leaving sensitive data vulnerable to interception and exfiltration.

• Inadequate precautions against illegal actions like taking screenshots or copying private data.

• Technical problems that arose during the installation of security features, such as the Captcha image display's initial malfunction.

D. Objectives

1) Integration of Security Features: Our primary goal was to identify and implement strong security measures to address the vulnerabilities discovered during the project. To accomplish this, we took a systematic approach and included the following security features:

• Access Control: IP Control Access is put into practice to stop illegal access attempts and infiltration.

• Attack Prevention: Implementing defenses against denial-of-service assaults, such as DoS Attack Prevention with Captcha. Fixing technical problems with code optimization and troubleshooting, including the broken Captcha picture display.

• Data Protection: Data Encryption & Decryption techniques are used to guarantee the confidentiality of critical data, and Password Hashing is implemented to prevent unwanted access and password propagation.

• Unauthorized Activity Prevention: Implementing methods to prevent unauthorized activity, such as taking screenshots or copying sensitive information.

2) Enhanced Security Awareness: In addition to providing security measures, we attempted to educate project stakeholders on the necessity of cybersecurity best practices. This included giving stakeholders training and resources to help them detect and mitigate security issues efficiently.

3) Evaluation and Testing: To confirm the effectiveness of the established security measures, we performed extensive evaluation and testing, including penetration testing and vulnerability assessments. This iterative procedure enabled us to successfully discover and address any lingering security vulnerabilities.

II. Literature Survey

In today's fast changing digital environment, the significance of robust software security measures cannot be emphasized. As cyber threats become more sophisticated and widespread, safeguarding the integrity and security of sensitive data has become a top priority for organizations across all industries. In response to this expanding concern, lots of research is being conducted to identify innovative and effective ways to secure software systems against malicious attacks.

Almogahed et al. [1] explores how refactoring approaches, particularly those involving information hiding, affect software security. It classifies various strategies according to their influence on security metrics to help developers improve software security. The paper examines the benefits of commonly used refactoring approaches across five case studies on security metrics such as COA, CIDA, and CCDA, hence contributing to a better safe software environment. The proposed method aids developers in selecting the appropriate strategies, saving time and effort in considering their security implications. Future study aims to widen the classification by investigating more common refactoring techniques, such as Extract Class and Move Method, to improve software security protections.

Shehab Farhan and Mostafa Mostafa [2] presents an approach for improving software security throughout the development process by incorporating security measures at each stage. It emphasizes the importance of examining security at all stages of the development life cycle, not just testing, in order to improve overall software security. This method ensures that applications are more secure, have higher quality, and perform better. Future aims include creating automated tools for security measurement throughout the software development life cycle.

Vaigandla et al. [3] discusses the enhancement of Intrusion Detection Systems (IDSs) in the context of IoT, emphasizing its critical role in improving security and data protection. The study claim for the use of advanced intrusion detection systems (IDS) in IoT contexts to effectively combat cyber threats. The study focuses on utilizing technologies such as databases, cloud computing, and big data processing to improve the efficiency of IDSs in IoT. Article looks at both signature-based and anomaly-based IDSs, explaining how they detect and prevent threats in IoT environments. Furthermore, it emphasizes the importance of host-based IDS (HIDS) in monitoring host environments to detect suspicious activity, hence improving the security of IoT devices and networks.

Lingham et al. [4] emphasize the importance of security in software development, averting against the risks of ignoring security features and the consequent vulnerabilities. They hope to contribute to the often-overlooked field of software security by advocating recognizing security implications as well as utilizing secure development methodologies and techniques. They emphasize the relevance of frameworks such as Microsoft SDL in driving security model development but also point out implementation obstacles. They use several publications to investigate security concerns, methods, and the efficacy of various security measures in software development. They emphasize the importance of continual progress in knowledge, skills, and the usage of strong security technologies to meet new cyber threats and ensure software security.

III. Tools and Validation

Concepts from websites like Uloop, Harvard University, My Campus UCAM, and the University of Oxford are incorporated into the software development process. To make better strategic decisions and improve their market position, businesses use benchmark product analysis to learn more about their competitors and find ways to improve. Table I is the comparison table between our project and the existing project . There are several programming languages used, such as PHP, Python, JavaScript, HTML, and CSS. Used application programming interfaces (APIs). Git is used as the version control system because of its ability to monitor changes in the code and facilitate teamwork. For effective code authoring, testing, and debugging, development environments like Visual Studio

TABLE I
COMPARISON TABLE

Feature	Uloop	Oxford	Harvard	My Campus	UCAM	UIUServiceHub
Virtual Hostel	No	No	No	No	No	Yes
Payment Method	Yes	Yes	Yes	Yes	Yes	Yes
Chat	No	Yes	Yes	Yes	No	Yes
Post	Yes	Yes	Yes	Yes	No	Yes
Shuttle Booking	No	No	Yes	No	Yes	Yes
Notification	Yes	Yes	Yes	Yes	Yes	Yes
Sorting	Yes	Yes	Yes	No	No	Yes
Profile	No	Yes	Yes	Yes	Yes	Yes
Loan	Yes	Yes	Yes	No	No	Yes
History	No	Yes	Yes	No	Yes	Yes

Code, Sublime Text, and Notepad are incorporated. This all-encompassing strategy guarantees the software development process's efficacy and dependability.

IV. PLAN, DESIGN, AND SOLUTION

In our Plan, Design, and Solution section, we carefully implemented security measures into each phase to protect the platform from any attacks and weaknesses.

A. Plan

We began with thorough studies to better understand student demands, then designed an intuitive platform architecture. We fine-tuned our strategy to meet consumer needs through iterative design and feedback. Agile techniques provided timely delivery while maintaining quality. Our security measures were confirmed through rigorous testing, which included unit testing, and system testing. During the planning phase, security considerations were prioritized alongside functional requirements. We conducted a thorough risk assessment to identify potential security threats and vulnerabilities.

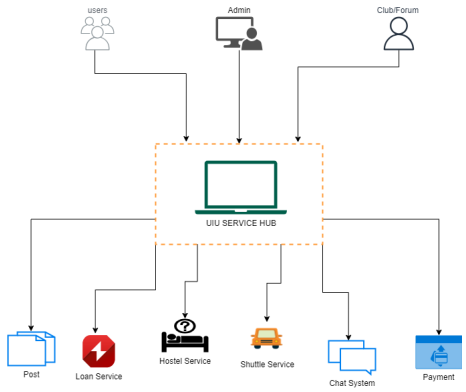

Fig. 1. DataFlow Diagram

B. Solution

Our solution involved integrating robust security measures to address every aspect of the breach Quadrilateral:

- **Infiltration:**
- Implemented an Email Alert System to notify administrators of suspicious activities.
- Integrated an Intrusion Detection System to monitor and respond to unauthorized access attempts in real-time.
- **Propagation:**
- Deployed mechanisms for DoS Attack Prevention to ensure uninterrupted service availability.
- Implemented Advanced Authentication methods, including multi-factor authentication, to enhance user security.
- **Aggregation:**
- Implemented measures to prevent SQL injection attacks through input validation and parameterized queries.
- Employed Data Masking techniques to protect sensitive information from unauthorized access.
- **Exfiltration:**
- Deployed a Firewall to monitor and control network traffic, preventing unauthorized data exfiltration.
- Implemented measures to Prevent Taking Screenshots and ensured Data Encryption for sensitive information.

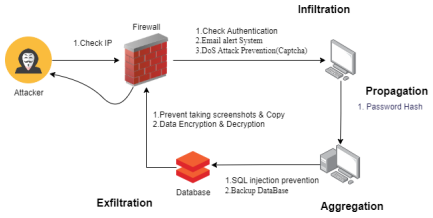

Fig. 2. Breach Attack

C. Implementation

We created a feature-rich platform using HTML, CSS, JavaScript, MySQL, and PHP, with an emphasis on security and usability. MySQL and PHP were utilized to protect the backend, and HTML, CSS, and JavaScript were employed for front-end development, resulting in a responsive and intuitive user experience. Throughout the deployment phase, we performed thorough testing and quality assurance to check functionality and security measures, resulting in a strong and dependable platform for UIU students.

Fig. 3 is our project's Schema diagram. It is a visually appealing depiction of a database system's structure and organization. The relationship between entities, properties, and connections in a database is modeled by it.

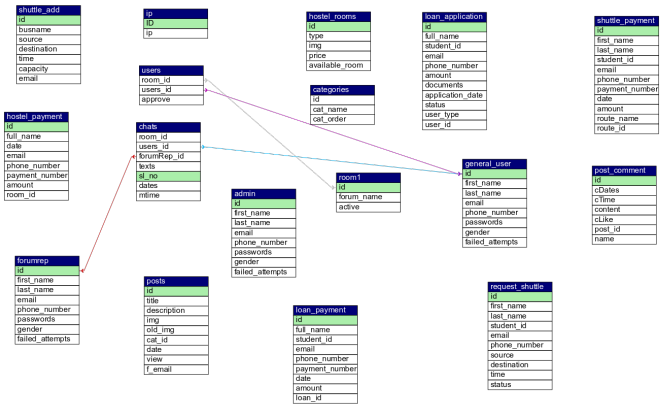

Fig. 3. Schema of our project

In fig. 4 of SWOT analysis for our project. Mainly SWOT analysis is a strategic planning tool that assists organizations in identifying their strengths, weaknesses, opportunities, and threats, thereby aiding decision-making and project planning.

Fig. 4. SWOT Analysis

D. Hash Function

Algorithm 1: Hash Function

Input: password
Output: hashed_password
Function `custom_hash(`*password*`)`**:**
  hash_value $\leftarrow 0$
  for $i \leftarrow 0$ **to** *length*(*password*) **do**
    hash_value $\leftarrow$ (hash_value $\ll$ 5) $\oplus$ ord(password$[i]$)
  hashed_password $\leftarrow$ dechex(hash_value)
  hashed_password $\leftarrow$ str_pad(hashed_password, 13,$'$ 0$'$, STR_PAD_RIGHT)
  return hashed_password

The design of the custom_hash function renders it non-reversible, resulting in the inherent loss of information pertaining to the initial password. The function initializes the hash_value to 0 and proceeds iteratively through each character of the password, executing an XOR operation with the ASCII value of the current character and a left shift by 5 bits. The mathematical representation of the transformation at each stage is as follows:

$$\text{hash\_value} = (\text{hash\_value} \ll 5) \oplus \text{ord}(\text{password}[i]),$$

where $\ll$ denotes the left shift operation and $\oplus$ denotes the XOR operation.

The hash_value $\ll 5$ operation shifts bits to the left, introducing zeros and potentially discarding higher bits, resulting in the loss of original information. The XOR operation entangles input data by combining the current hash value with the character's ASCII value. Each character has an impact on the entire hash value, making the process cumulative and complex. The left shift and XOR operations gradually remove the original character data since they are irreversible. Potential collisions are introduced by the final hexadecimal conversion and padding, which provide a fixed-length output. Therefore, mathematically, the function's transformations do not preserve enough information to allow inversion, proving the function's non-reversibility.

E. Encryption & Decryption

Algorithm 2: Encryption

Input: plaintext, key
Output: encryptedText
Function `simpleEncrypt(`*plaintext, key*`)`**:**
  keyLength $\leftarrow$ length(key)
  encryptedText $\leftarrow''$
  for $i \leftarrow 0$ **to** *length*(*plaintext*) **do**
    encryptedText $\leftarrow$ encryptedText + chr(ord(plaintext$[i]$) + ord(key$[i$ mod keyLength$]$))
  return base64_encode(encryptedText)

Algorithm 3: Decryption

Input: encryptedText, key
Output: decryptedText
Function `simpleDecrypt(`*encryptedText, key*`)`**:**
  encryptedText $\leftarrow$ base64_decode(encryptedText)
  keyLength $\leftarrow$ length(key)
  decryptedText $\leftarrow''$
  for $i \leftarrow 0$ **to** *length*(*encryptedText*) **do**
    decryptedText $\leftarrow$ decryptedText + chr(ord(encryptedText$[i]$) $-$ ord(key$[i$ mod keyLength$]$))
  return decryptedText

In this project, Symmetric encryption-decryption has been applied. The user's unique email address has been used as the encryption key.

V. Conclusion

The UIU ServiceHub initiative effectively satisfies the pressing need for a single platform that simplifies UIU students' access to a range of university services. We have greatly increased the effectiveness and convenience for students by combining services such as Club Forum Discussion, Posts, Student Loans, Shuttle Service Registration, and Hostel Service into a single, user-friendly interface. Furthermore, the project's focus on implementing strong security measures has changed it from a platform that could be vulnerable to a safe and dependable system. Providing the quadrilateral's four layers of solid security against breach is the primary goal. The system vulnerability is resolved by adopting an updated perspective on security assaults. For a more dependable and improved user experience, appropriate security precautions and actions have also been implemented. The first step in the project's journey was to uncover significant security flaws, such as slack access controls, a vulnerability to SQL injection attacks, a lack of data encryption, and inadequate safeguards against illegal activity. We have strengthened the platform against several cyber threats by methodically resolving these problems and putting in place robust security features including data encryption, IP Control Access, DoS Attack Prevention with Captcha, and unauthorized activity prevention measures. Furthermore, the project emphasized how crucial it is for stakeholders to be aware of security issues. We sought to foster a culture of cybersecurity vigilance by providing resources and training so that all users would be able to identify and counter possible security threats.

VI. Future Work

Despite the tremendous improvements, there are still a number of areas that the UIU ServiceHub can be improved:

Advanced Threat Identification and Response: Using machine learning techniques to detect anomalies can assist in the real-time identification and mitigation of new and sophisticated cyber threats.

Automated Security Testing: Early vulnerability detection and remediation in the software development lifecycle can be achieved by including continuous security testing in the development pipeline. Automated technologies for both static and dynamic analysis can help achieve this.

User Behavior Analytics: Putting user behavior analytics into practice can help provide more in-depth information about how students utilize the platform. This can enhance the user experience overall and assist in spotting questionable activity.

Improved Data Privacy Measures: You may make sure that student data is protected in compliance with the most recent laws and industry best practices by implementing GDPR-compliant data handling methods, among other enhanced data privacy measures.

Scalability and Performance Optimization: As the user base expands, it will become increasingly important to continuously optimize performance. Optimizing database queries and using scalable architectural solutions helps guarantee that the platform is responsive and effective even under high loads.

Integration of User Feedback: Collecting and incorporating user feedback on a regular basis can aid in the platform's ongoing improvement to better suit students' changing demands.

Mobile Application Development: By creating a specific mobile application for UIU ServiceHub, accessibility may be improved and students will be able to easily access services while they are on the go.

Advanced Encryption Techniques: Data security can be further improved by investigating and putting into practice more complex encryption methods, particularly for extremely sensitive data.

References

[1] A. Almogahed, M. Omar, and N. H. Zakaria, "Refactoring codes to improve software security requirements," *Procedia Computer Science*, vol. 204, pp. 108–115, 2022, international Conference on Industry Sciences and Computer Science Innovation. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922007517

[2] A. R. Shehab Farhan and G. M. Mostafa Mostafa, "A methodology for enhancing software security during development processes," in *2018 21st Saudi Computer Society National Computer Conference (NCC)*, 2018, pp. 1–6.

[3] K. Vaigandla, N. Azmi, and R. Karne, "Investigation on intrusion detection systems (idss) in iot," vol. 10, pp. 158–166, 03 2022.

[4] A. D. Lingham, N. T. K. Kin, C. W. Jing, C. H. Loong, and F. tuz Zahra, "Implementation of security features in software development phases," 2020.