UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

W O R K S H O P 1

R E P O R T

Name : SUFIANA ADLIN BINTI BAHAROM

Matric Number : B032410816

Program : BITD S1G1

Project Title : LESTARI DORMITORY SYSTEM (LDS)

Supervisor Name : PM. TS. DR. NURUL AKMAR BINTI EMRAN

Evaluator Name : GS. TS. DR. SAFIZA SUHANA BINTI KAMAL BAHARIN

EXECUTIVE SUMMARY

The unique Lestari Dormitory System (LDS) is a stand a stand-alone system that was created to transform maintenance reporting and administration in dormitory for students. The whole maintenance workflow is streamlined by this technology, which introduces a consolidated platform to solve the inefficiencies and difficulties associated with previous manual operations. By enabling students to safely report maintenance concerns using their student numbers, LDS guarantees a secure and convenient experience. After being submitted, these requests are methodically categorized and given a priority ranking by management, which makes the response process more structured and effective.

The capacity of LDS to deliver real-time information, which allows students to track the progress of their maintenance requests during the resolution process, is one of its primary advantages. This openness improves communication between students and dorm administration in addition to fostering trust. LDS also gives administrators the ability to access stored data and create comprehensive reports, providing insightful information for resource allocation and decision-making. LDS seeks to greatly enhance students' living circumstances by incorporating cutting-edge technology into maintenance procedures. The technology guarantees that maintenance issues are handled efficiently and quickly while also speeding up response times and cutting down on delays. Additionally, it streamlines communication channels, which facilitates students' expression of issues and management's effective response to them. In the end, LDS is a holistic platform that improves student happiness by fostering a more cozy, secure, and well-kept living environment. It is more than simply a maintenance tool. LDS is a big step toward increasing operational effectiveness, accountability, and the general student experience by utilizing technology to solve frequent issues with dorm upkeep.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

LIST OF FIGURES

PAGES

CHAPTER 1: INTRODUCTION

1.1. Introduction

Accommodations for students are essential to providing a safe and comfortable living environment, which has a big influence on their general academic experience and general well-being. However, maintaining these facilities to a high grade comes with several difficulties that may have an impact on how well students live their lives. Many student dormitory facilities now manage and report maintenance concerns using antiquated methods like spreadsheets and manual processes, which are becoming less and less effective. These approaches frequently lead to significant drops in student satisfaction, lengthy response times, and organizational bottlenecks. The Lestari Dormitory System (LDS), a customized software program intended to revolutionize the reporting, monitoring, and management of maintenance concerns in student dormitory, is suggested as a solution to these problems. LDS aims to offer a simplified, unified platform that fosters more accountability and transparency while also improving the effectiveness of maintenance operations. LDS seeks to improve students' living experiences by resolving the drawbacks of current manual procedures and creating a more encouraging and well-kept residential setting that promotes both academic achievement and personal development.

1.2. Problem Statement

- Weak and Ineffective Reporting Mechanisms.

The method of resolving dormitory issues is currently chaotic and ineffective due to the dependence on human reporting systems. Maintenance request submission and tracking become chaotic when techniques like paper forms, emails, or simple platforms like Google Forms are used. These inefficiencies are made worse by the absence of uniformity in reporting practices, which results in uneven data handling and collecting. Timely responses are hampered by these manual procedures, which frequently result in delays in the submission and processing of maintenance requests.

- Poor Accountability and Transparency

Poor accountability and transparency in maintenance operations are the outcome of a decentralized system. Lack of real-time updates on the progress of their maintenance requests frequently leaves students feeling frustrated and mistrustful. It

is challenging to manage the status of maintenance work and identify accountable parties in the absence of an efficient monitoring system. Furthermore, a poor feedback system restricts students' ability to offer suggestions for maintenance services, impeding attempts at ongoing development.

- Poor Data Management

Maintenance information might be difficult to assess and use for strategic planning since information is frequently dispersed across several platforms or manually kept. Administrators are unable to quickly spot maintenance patterns or allocate resources as efficiently as they could due to this chaos. Finding valuable insights and creating thorough reports become difficult and time-consuming tasks in the absence of a centralized data management system. Proactive maintenance planning and well-informed decision-making are further hampered by the absence of historical data and analytics tools.

1.3. Objective (s) of the project

- To develop a centralized maintenance reporting system.

To Implement a user-friendly platform for students to report maintenance issues and track the status of their requests.

- To enhance transparency and accountability.

To Provide real-time updates on maintenance request statuses, ensuring students are informed throughout the resolution process and fostering trust in the maintenance management process.

- To improve data management and analysis.

To Provide a strong data management system that gathers, arranges, and evaluates maintenance data so that managers can make well-informed decisions and allocate resources efficiently.

1.4. Scope

1.4.1. Module to be developed

i. Secure Login: Provides secure login functionality for students, staff, and administrators using student numbers or staff IDs, with role-based access control.

ii. User Authentication: Ensures verified access for authorized users, differentiating permissions between admins and staff.

iii. Account Management: Allows **administrators** to create, update, delete, and manage user accounts, including assigning roles and managing staff privileges.

iv. Issue Reporting: Enables students to submit maintenance requests with detailed information and urgency levels.

v. Categorization of requests: Organizes reported issues into predefined categories; admins can add or modify categories.

vi. Prioritization of requests: Allows **management (admins and authorized staff)** to prioritize requests based on urgency and impact.

vii. Track Maintenance Progress: Both **staff and students** can monitor maintenance request statuses in real-time; staff update progress, admins oversee overall workflow.

viii. Data Storage: Securely stores all maintenance data with access controls based on user roles.

ix. Penalty Management: Admins manage penalties issued to students, including creation, updates, and removal.

x. Report Generation: **Admins** generate detailed reports and analytics on maintenance activities, user performance, and system usage.

1.4.2. Target User

1. Administrators (Admins)

    a. Secure Login: Access using staff ID that are already stores in database admin table with administrator privilege.

    b. Manage Requests: Secure Login: Access using staff ID with administrator privileges (Create, update, delete user accounts, and assign roles (admin, staff)).

    c. View Reports: Generate and access detailed reports on maintenance activities, user performance, and penalties.

d. Manage Accounts: Issue, update, and track penalties for students.
e. Manage Penalties : Issue, update, track, and delete penalties for students, as well as review penalty policies and reports.

2. Staff

a. Secure Login: Access using staff ID with limited staff privileges.
b. Manage Assigned Requests: View and update status of maintenance requests assigned to them.
c. Track Requests: Update request progress and communicate with students.
d. View Requests: Access only requests relevant to their assigned tasks.
e. Monitor Penalties: View penalties related to their assigned students and assist in ensuring compliance.

3. Students

a. Secure Login: Access using student matric number.
b. Report Issues: Submit maintenance requests with detailed information.
c. Track Requests: Monitor the status and progress of their maintenance requests in real-time.
d. View Penalties: Access their own penalty history and any newly assigned penalties.
e. Pay Penalties: Make penalty payments through the system and receive payment receipts.

1.5. Project Significance

- To enhanced maintenance efficiency.

The efficiency of maintenance operations in student housing is expected to be greatly increased with the installation of the Lestari Dormitory System (LDS). LDS removes the inefficiencies and bottlenecks connected to manual procedures by centralizing

the reporting, monitoring, and administration of maintenance requests. Because of this efficiency, maintenance personnel will be able to address student concerns more rapidly and efficiently, guaranteeing that problems are fixed as soon as possible and with the least amount of disturbance to students' lives.

- To improved transparency and accountability.

By giving real-time information on the status of maintenance requests, LDS will promote an environment of accountability and openness. From the time their concerns are first submitted until they are finally resolved, students will be able to see exactly how far along they are. Since students are constantly updated at every stage, this greater openness fosters confidence between them and management. Furthermore, the tracking features of the system guarantee that accountable parties may be quickly located, improving accountability throughout the maintenance procedure.

- Data-driven decision making.

Data-driven decision-making is made possible by LDS, which unifies maintenance data into a single system. To guide resource allocation and strategic planning, administrators may quickly review historical data, spot reoccurring maintenance problems, and examine patterns. The capacity to provide thorough reports facilitates ongoing optimization and development by offering insightful information about how well maintenance activities are doing.

- To ensure it simplified communication.

By offering a clear and efficient route for reporting and monitoring maintenance concerns, LDS streamlines communication between management and students. By eliminating the need for back-and-forth emails or phone conversations, the system makes sure that all pertinent information is recorded and effectively shared.
Misunderstandings are reduced and maintenance requests are handled precisely and on time thanks to this streamlined communication.

- To increase student satisfaction.

By enhancing the general living conditions in student housing, LDS implementation ultimately seeks to raise student happiness. An atmosphere that is more pleasant,

secure, and well-maintained for students is produced by LDS by improving maintenance efficiency, transparency, accountability, and communication a greater.

1.6. Gantt Chart of Project Activities

Gantt chart of this project activities a shown as Table 1.1 below:

Table 1.1 : Gantt chart of Project Activities

MONTHS		*MARCH*	*MARCH*	*MARCH*	*APRIL*	*APRIL*	*MAY*	*MAY*	*MAY*	*MAY*	*MAY*	*JUN*	*JUN*	*JUN*	*JUN*
TOPIC	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**	**Week 11**	**Week 12**	**Week 13**	**Week 14**	**Week 15**
Perbincangan Tajuk dengan Penyelia															
Penyerahan dan Penilaian Kertas Cadangan															
Analisis dan Rekabentuk															
Kemajuan Projek 1															
Kemajuan Projek 2															
Kemajuan Projek 3															
Pembentangan Akhir dan Penyerahan Laporan															

Table 1.6.: Gantt Chart Of Project Activities

CHAPTER 2: SYSTEM ANALYSIS

This chapter presents an analysis of the issues identified in the current systems. Flowcharts are utilized to visually illustrate and explain these problems. Conducting this analysis is essential as it lays the foundation for proposing effective solutions in the subsequent design chapter.

2.1. Problem Decomposition Description

1. Weak and Ineffective Reporting Mechanism.
    i. Manual Reporting: The present dependence on manual procedures or basic technologies such as Google Forms results in the reporting of maintenance concerns being chaotic and inefficient.
    ii. Lack of Standardization: Different methods of reporting (e.g., paper forms, emails) result in inconsistent data collection and management.
    iii. Delays in Reporting: Manual processes often lead to delays in submitting and processing maintenance requests.

2. Poor Accountability and Transparency.
    i. Limited Visibility: Students lack real-time updates on the status of their maintenance requests, leading to frustration and mistrust.
    ii. Inadequate Feedback Mechanism: Students have limited opportunities to provide feedback on maintenance services, hindering continuous improvement.

3. Poor Data Management.
    i. Data Disorganization: Maintenance data is scattered across different platforms or stored manually, making it difficult to analyse and use for strategic planning.
    ii. Lack of Insights: Without a centralized data management system, administrators cannot easily identify maintenance trends or optimize resource allocation.

2.1.1. Flowchart

Each process and problem will be described below:

2.1.1.1. Flowchart for student Reporting (Current Manual Process – Face to Face)

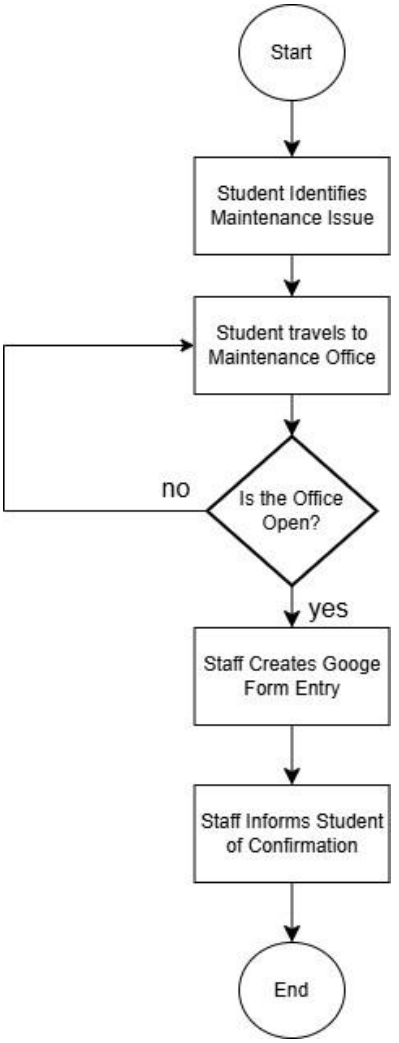


Figure 2.1.1.1.: Flowchart for student Reporting (Face to Face)

Figure 2.1.1.1 illustrates the process of students reporting maintenance issues by physically going to the maintenance office. The process begins when a student identifies a maintenance issue and travels to the office. The key step involves determining if the office is open; if not, the student must return during operating hours, adding significant inconvenience. If the office is open, the student describes the issue to a staff member, who then creates a Google Form entry based on the student's description. Finally, the staff member informs the student that their report has been submitted. However, this method presents several drawbacks. The entire process is

heavily dependent on the office's operating hours, rendering it inaccessible outside those times. The need to physically travel to the office introduces significant inconvenience and potential delays. Furthermore, the reliance on verbal communication between the student and staff member can lead to misinterpretations or the omission of crucial details. The staff member's accurate and consistent data entry into the Google Form is also a point of potential error. Finally, the student receives only a verbal confirmation, lacking any means to track the request or verify its accurate submission

2.1.1.2. Flowchart for student Reporting (Current Manual Process - WhatsApp)

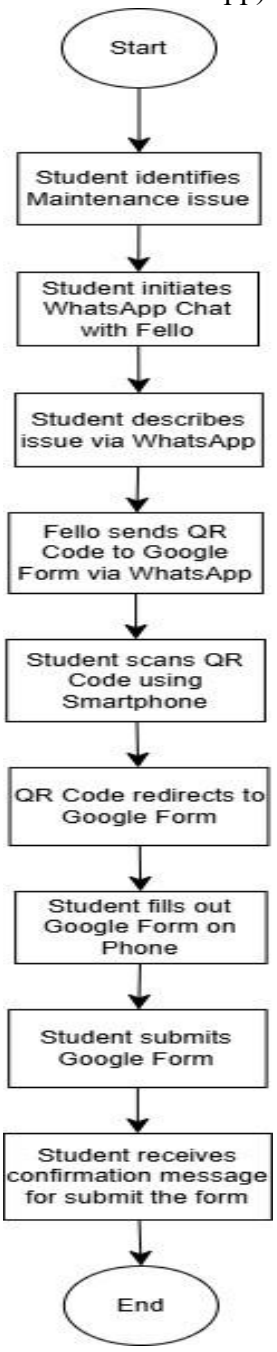


Figure 2.1.1.2.: Flowchart for student Reporting (WhatsApp)

Figure 2.1.1.2 illustrates students reporting issues via WhatsApp to a 'Fello', who provides a QR code for a Google Form. The student fills out this form - detailing their name, matric number, room, and the issue - and submits it. Key drawbacks include the dependency on 'Fello' availability, and potential delays related to Fello responsiveness. Furthermore, the initial WhatsApp exchange and QR code are vulnerable to data loss from phone/WhatsApp issues, requiring the student to restart. Describing the issue via unstructured WhatsApp messages can lead to misinterpretations requiring re-entry on the Google Form. The Google Form also introduces risks of data entry errors and lacks real-time tracking.

Based on the analysis of the issues present in the current system, the main problems of the Lestari Dormitory System (LDS) can be categorized into three key areas, as illustrated in Figure 2.1.2. These problems are elaborated as follows:

2.1.2. Problem Description

The main problems of Lestari Dormitory System (LDS) are:

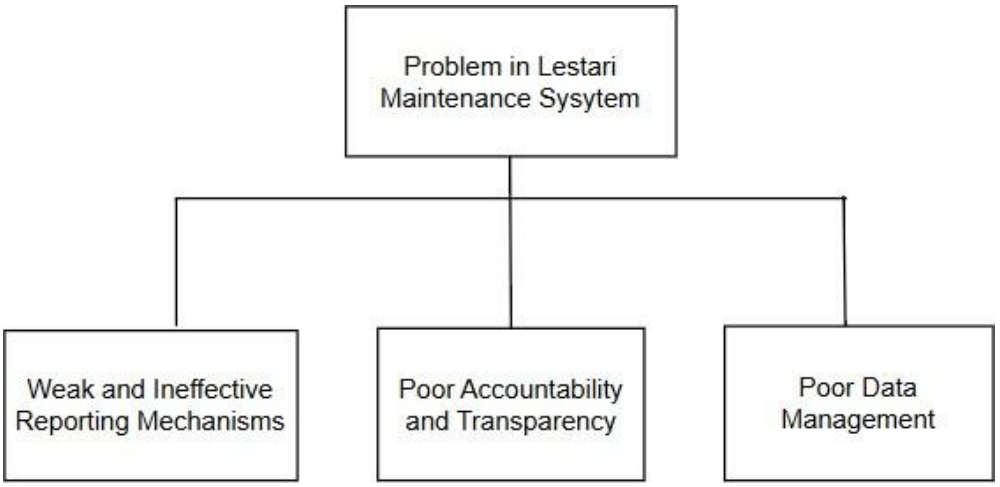


Figure 2.1.2.: Problem Description Structure Chart

a) Weak and ineffective reporting mechanisms.

The current reporting mechanisms for maintenance issues in the Lestari Dormitory System (LDS) are weak and ineffective. This is primarily due to the reliance on manual procedures and basic technologies like Google Forms, which results in chaotic and inefficient reporting processes. The lack of standardization in reporting methods, such as using paper forms and emails, leads to inconsistent data collection and management. Furthermore, these manual processes often cause delays in both submitting and processing maintenance requests. This problem highlights the challenge faced by the system in efficiently managing maintenance concerns due to the disorganized nature of reporting. It underscores the need for a solution that allows for standardized, efficient, and timely reporting of maintenance issues to ensure that all necessary information is accurately collected and processed, thereby enhancing the overall effectiveness of the maintenance system.

b) Poor accountability and transparency.

The Lestari Dormitory System (LDS) faces issues with poor accountability and transparency. Students lack real-time updates on the status of their maintenance requests, leading to frustration and mistrust. This limited visibility into the progress of their requests exacerbates the problem, as students are left without clear information on when issues will be resolved. Additionally, the system lacks an adequate feedback mechanism, limiting students' opportunities to provide input on maintenance services. This hinders continuous improvement, as valuable insights from users are not effectively captured or utilized. This problem highlights the challenge faced by the system in maintaining trust and ensuring that maintenance services meet student needs efficiently. It underscores the need for a solution that provides real-time updates and facilitates meaningful feedback to enhance accountability and transparency within the maintenance process.

c) Poor data management.

The Lestari Dormitory System (LDS) struggles with poor data management. Maintenance data is disorganized, scattered across various platforms or stored manually, which complicates analysis and strategic planning. This disorganization hinders the ability to leverage data for informed decision-making. Furthermore, the lack of a centralized data management system prevents administrators from easily identifying maintenance trends or optimizing resource allocation. This problem highlights the challenge faced by the system in effectively utilizing data to improve maintenance operations. It underscores the need for a solution that centralizes data management, providing clear insights and enabling administrators to make data-driven decisions to enhance efficiency and resource utilization within the maintenance process.

The Lestari Dormitory System (LDS) faces challenges in maintenance management due to inefficient manual reporting, lack of real-time updates and feedback, and disorganized data storage. These issues cause delays, reduce transparency, and hinder effective decision-making. A centralized, standardized, and data-driven solution is needed to improve reporting accuracy, accountability, and overall system efficiency.

CHAPTER 3: SYSTEM DESIGN

This chapter outlines the design of the proposed system, focusing on how the system components are structured and interact with one another. The design phase translates the requirements in solving the problems identified in the analysis phase into a blueprint for implementation.

The chapter begins with the structure chart, which illustrates the hierarchical organization of system modules and their interactions. This chart provides a clear overview of how the system is decomposed into manageable components. Following this, the program design section outlines the logic and flow of key processes within the system, ensuring that each module performs its intended function efficiently.

The database design is then presented, including the Entity-Relationship Diagram (ERD), which models the relationships between data entities, and the data dictionary, which defines the attributes, types, and constraints of each data element used in the system.

Finally, the user interface design section showcases the layout and functionality of the system's front-end, focusing on usability, accessibility, and user experience. Together, these components form a comprehensive design framework that supports the successful development and deployment of the proposed system.

3.1. Program Design

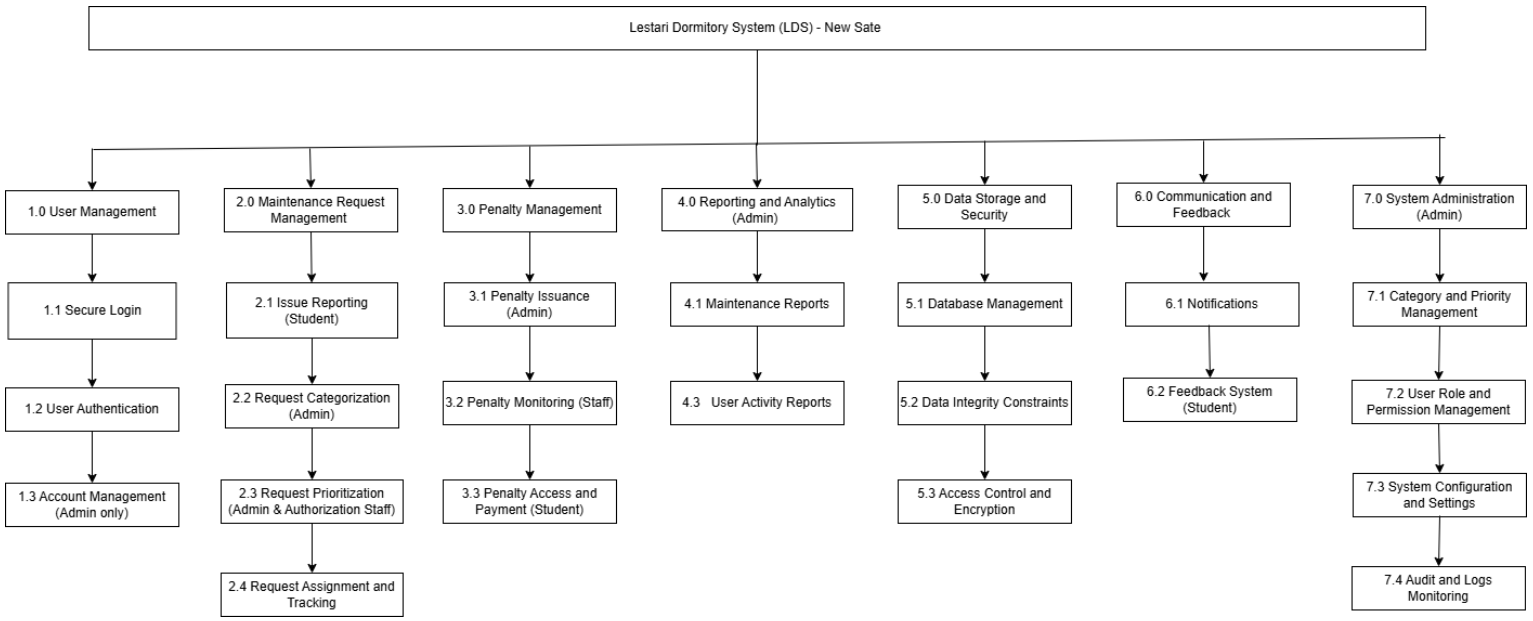


Figure 3.1.: New System Structure Chart

The design of the programs of the proposed system are illustrated by the flowcharts.

Flowchart.

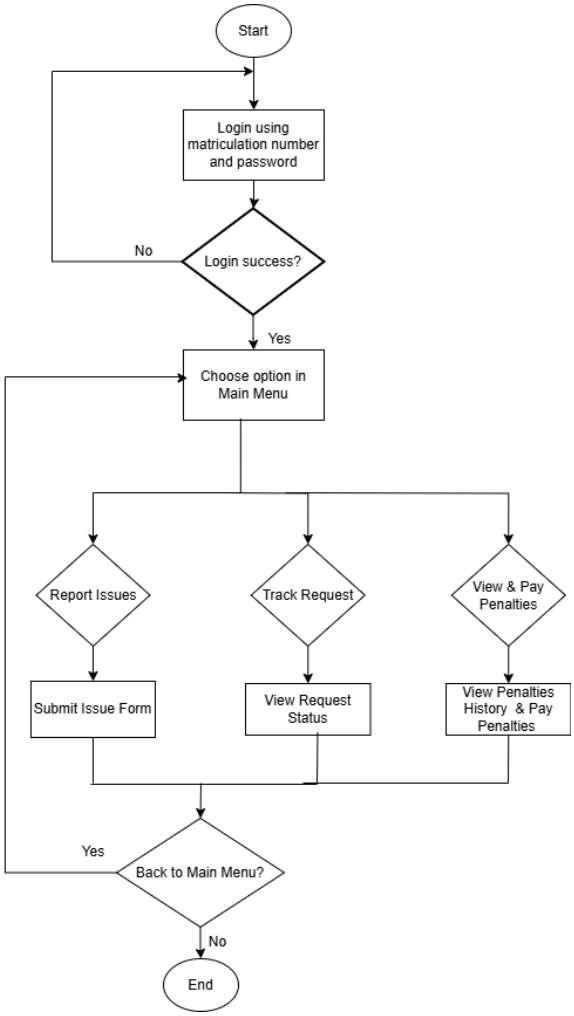


Figure 3.1.1.: Flowchart of Lestari Dormitory System Students.

Figure 3.1.1 illustrates the flowchart of the Lestari Dormitory System (LDS) for students. Initially, students are prompted to enter their matriculation number and password to log into the system. Upon successful login, they are directed to the main menu where they can choose between two primary options: "Report Issues", "Track Requests" or "View & Pay Penalties" If they select "Report Issues," they proceed to submit a maintenance request by filling out a form. If they choose "Track Requests" they view the status of their previously submitted requests and if they choose "View and Pay Penalties", they can view their penalties history and proceed to pay their penalties and assigned a staff to confirm their payment. After completing either action, students decide whether to return to the main menu or exit the system. This structured flow ensures that students can efficiently navigate and utilize the system's functionalities.

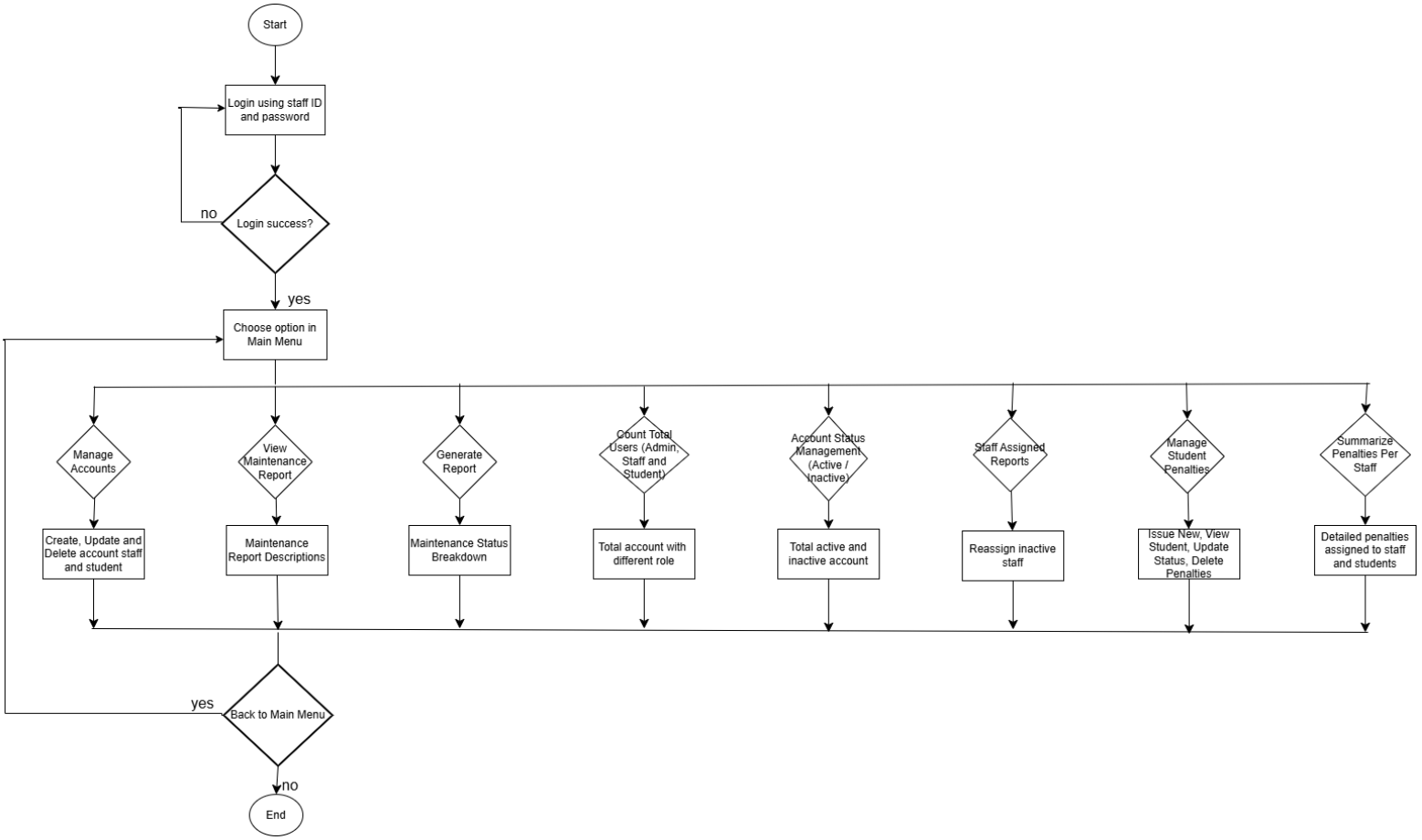


Figure 3.1.2.: Flowchart of Lestari Dormitory System Admin

Figure 3.1.2 illustrates the updated workflow for administrators within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access an enhanced main menu offering options to manage accounts, handle maintenance requests, view reports, and manage penalties. In “Manage Accounts,” administrators can create, update, delete user accounts, control account statuses (active/inactive), and view total users by role. The “Manage Requests” section allows them to view, categorize, prioritize, assign requests, and review staff-assigned reports. Through “View Reports,” they generate detailed reports on maintenance activities, user performance, penalties, and staff assignments. The “Penalty Management” module enables issuing, updating, tracking, and summarizing penalties for students, including penalty summaries per staff. After completing any task, administrators can choose to return to the main menu or end their session, ensuring a streamlined and comprehensive management process aligned with the system’s expanded functionalities.

3.1.3. Flowchart Of Lestari Dormitory System (LDS) For Admin Manage Account.

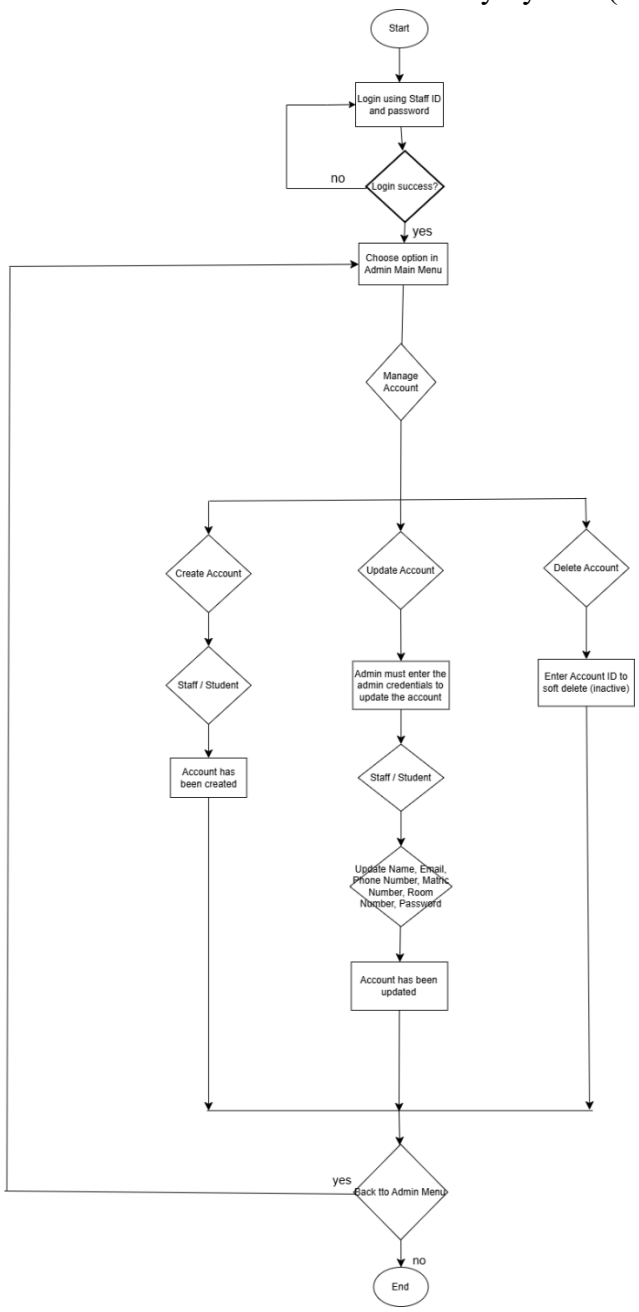


Figure 3.1.3.: Flowchart of Lestari Dormitory System Admin Manage Account

Figure 3.1.3 illustrates the detailed flowchart for managing maintenance requests within the Lestari Maintenance System (LMS) for admins. Initially, admins log in using their staff ID and password. Upon successful login, they proceed to the main menu and select "Manage Requests." This option allows them to view, categorize, prioritize, and update the status of maintenance requests. After completing these actions, admins decide whether to return to the main menu or exit the system. This structured flow ensures that admins can efficiently manage and monitor maintenance operations.

3.1.4. Flowchart Of Lestari Dormitory System (LDS) For Admin View Maintenance Report

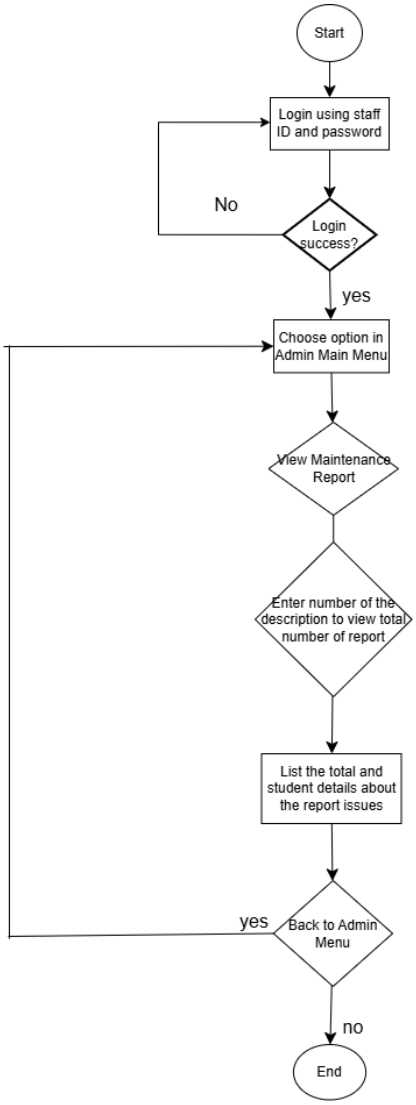


Figure 3.1.4.: Flowchart of Lestari Dormitory System Admin View Maintenance Report

Figure 3.1.4 illustrates the detailed flowchart for administrators viewing reports within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select "View Reports," which provides options to generate a variety of reports including maintenance history, current status, future schedules, penalty summaries, user statistics by role, and staff assignment reports. Upon selecting a report type, the system displays the relevant data, allowing administrators to analyze dormitory operations comprehensively. After reviewing the report, administrators can choose to return to the main menu or exit the system. This enhanced flow ensures efficient monitoring and management aligned with the expanded functionalities of the LDS.

3.1.5. Flowchart Of Lestari Dormitory System (LDS) For Admin Generate Report

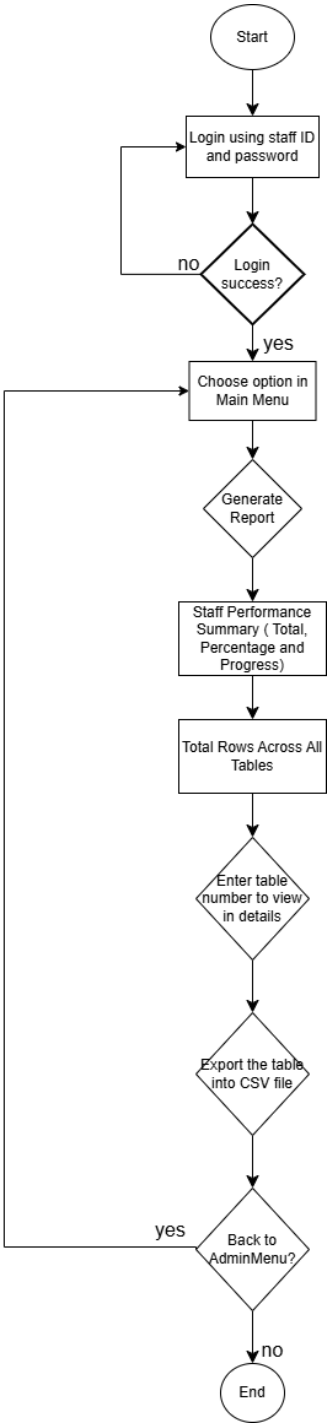
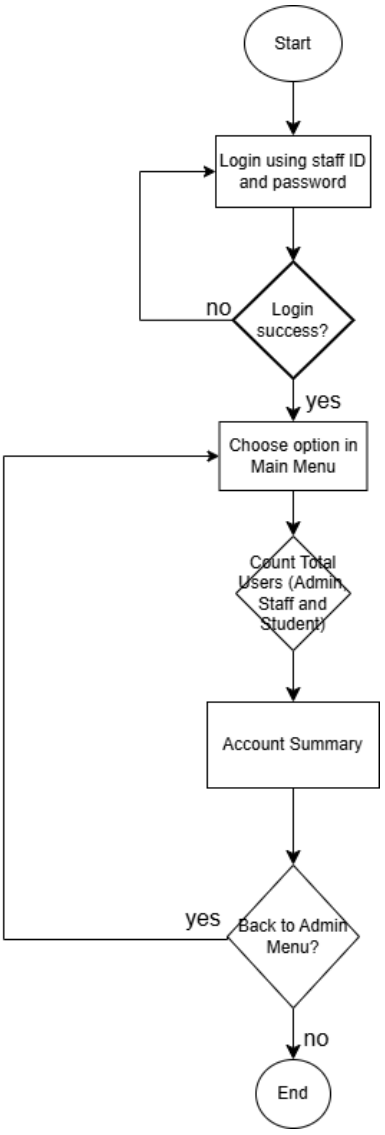


Figure 3.1.5.: Flowchart Of Lestari Dormitory System Admin Generate Report

Figure 3.1.5 illustrates the detailed flowchart for administrators generating reports within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select "Generate Reports," which provides options to view comprehensive data including staff performance summaries, maintenance status

breakdowns, monthly trends, and detailed database tables. Administrators can also export report data to CSV files for further analysis. Upon completing report generation and review, administrators may choose to return to the main menu or exit the system. This enhanced flow ensures efficient monitoring and management of dormitory operations, aligning with the system's expanded reporting and administrative capabilities.

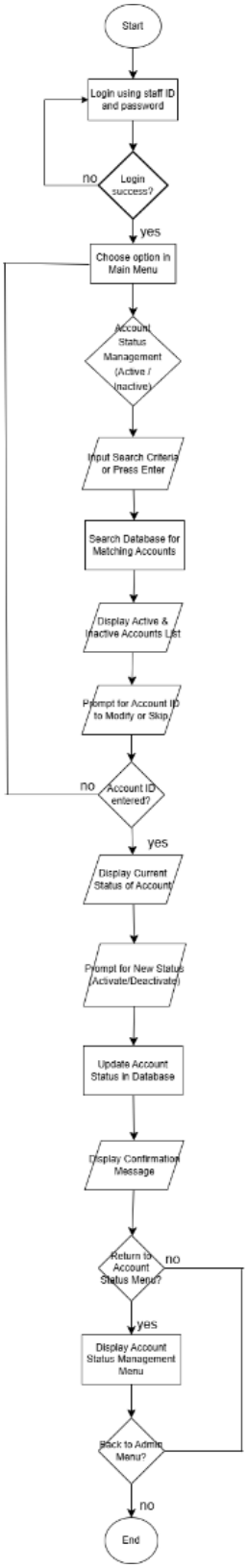
3.1.6. Flowchart Of Lestari Dormitory System (LDS) For Admin Count Total Users (Admin, Staff and Student)



3.1.6.: Flowchart Of Lestari Dormitory System Admin Count Total Users (Admin, Staff and Student)

Figure 3.1.6 illustrates the detailed flowchart for administrators to view user account summaries within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select the option to view the total number of users, categorized by roles including admins, staff, and students. The system displays a concise account summary showing the count of each user type, assisting administrators in monitoring system usage and managing resources effectively. After reviewing the summary, administrators may return to the main menu or exit the system. This flow supports efficient oversight of user accounts aligned with the system's comprehensive management capabilities.

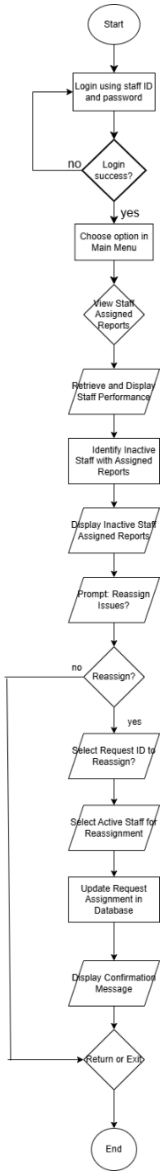
3.1.7. Flowchart Of Lestari Dormitory System (LDS) For Admin Account Status Management (Active / Inactive)



3.1.7.: Flowchart Of Lestari Dormitory System Admin Account Status Management (Active / Inactive)

Figure 3.1.7 illustrates the detailed flowchart for administrators managing account statuses within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the account status management menu, where they can enter a student or staff ID, name, or email to filter accounts, or press Enter to generate a full list of active and inactive accounts. The system then displays the accounts with details including account ID, type, status, name, and email, grouped by active and inactive status along with totals. Administrators are prompted to enter an account ID to modify its status or leave it blank to skip. If an account ID is entered, the status is shown, and the administrator can update it to active or inactive. The system saves the changes and confirms the update. After completing these actions, administrators may choose to return to the account status menu or exit the system. This structured flow ensures efficient and comprehensive management of user account statuses, supporting effective dormitory administration aligned with the system's role-based access and reporting capabilities.

3.1.8. Flowchart Of Lestari Dormitory System (LDS) For Admin Staff Assigned Report

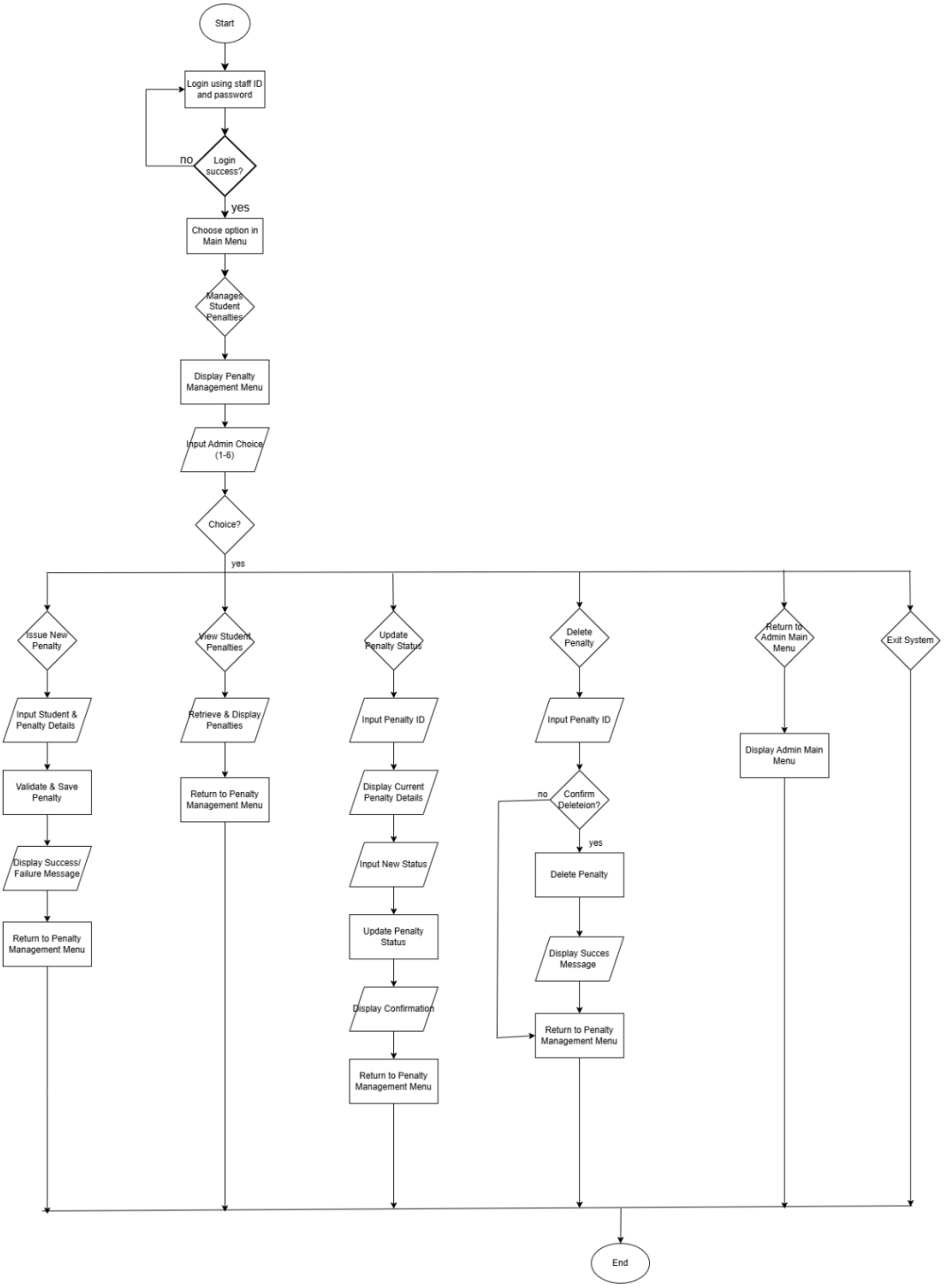


3.1.8.: Flowchart Of Lestari Dormitory System Admin Staff Assigned Report

Figure 3.1.8 illustrates the detailed flowchart for administrators managing staff assigned reports within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select the option to view staff assigned reports. The system displays a performance summary for each staff member, including their ID, name, phone number, status, and counts of resolved, rejected, on-hold, and reassigned requests. If any inactive staff members have outstanding assigned maintenance requests, these are listed with their request IDs, descriptions, statuses, and current assignments. Administrators are then prompted to decide whether to reassign these pending requests to active staff members. If reassignment is chosen, the administrator selects the requests and designates active staff to take over the assignments. The system updates the records and confirms the changes. After completing these actions, administrators may choose to return to the staff assigned report menu or exit the system. This

structured flow ensures ongoing maintenance task management and efficient staff workload distribution, supporting effective dormitory operations and service continuity.

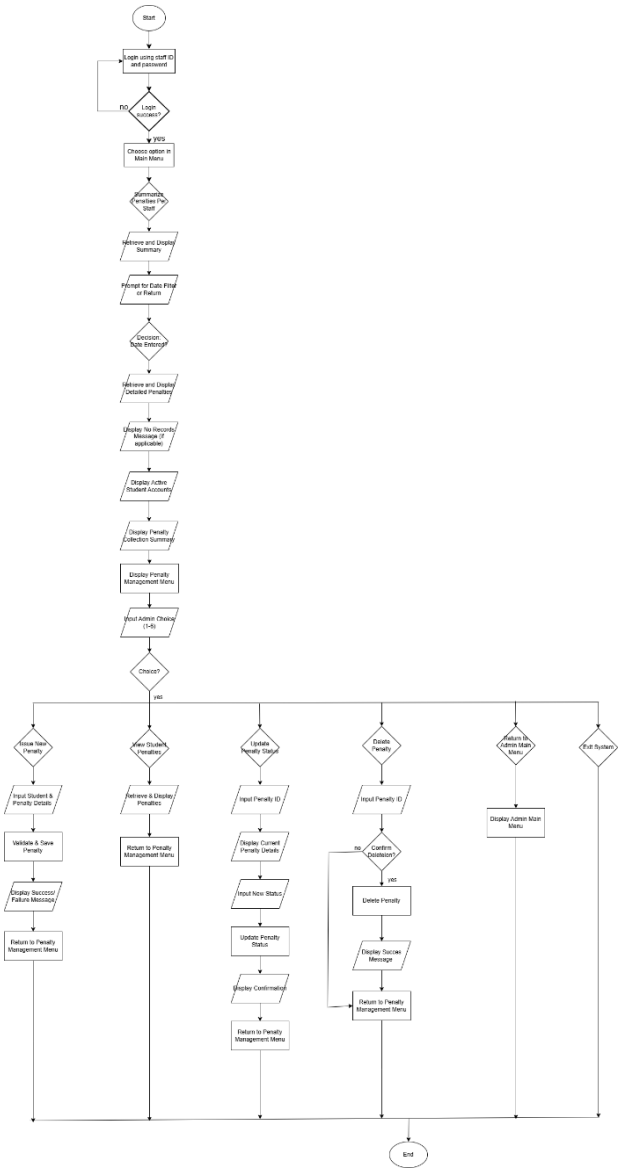
3.1.9. Flowchart Of Lestari Dormitory System (LDS) For Admin Manage Student Penalties



3.1.9.: Flowchart Of Lestari Dormitory System Admin Manage Student Penalties

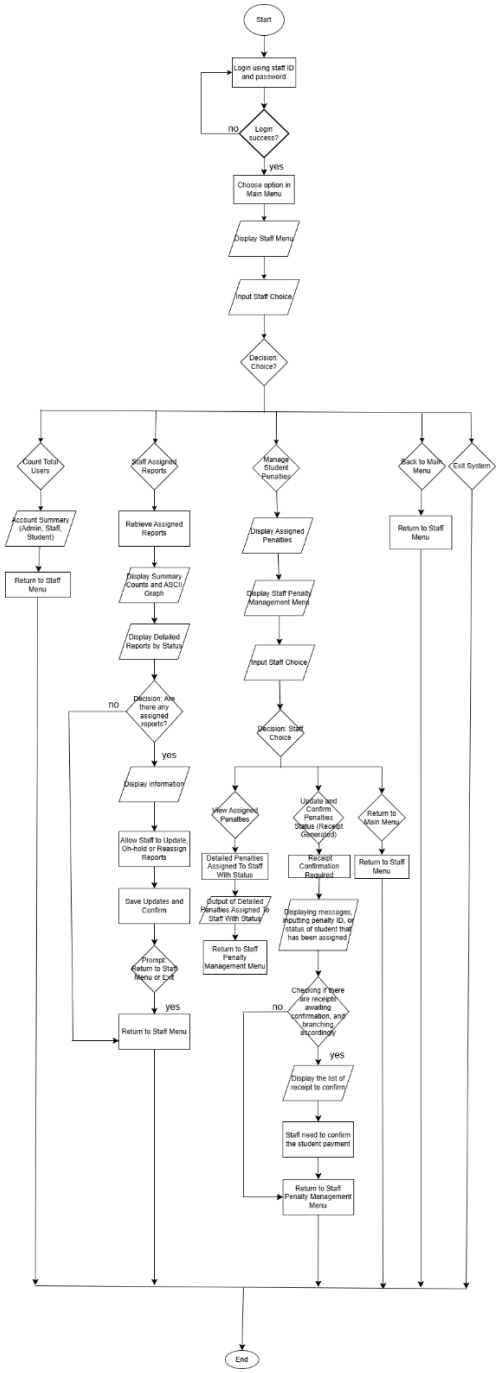
Figure 3.1.9 illustrates the detailed flowchart for administrators managing student penalties within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select the "Manage Student Penalties" option. The system then presents a penalty management menu where admins can choose to issue new penalties, view existing student penalties, update penalty statuses, delete penalties, return to the main menu, or exit the system. Depending on the selected option, administrators input relevant details such as student information, penalty descriptions, and status updates. The system processes these inputs by validating, saving, updating, or deleting records accordingly, and displays confirmation or error messages. Administrators can also export penalty data for reporting purposes. After completing their tasks, administrators may return to the penalty management menu or exit the system. This structured flow ensures effective monitoring and administration of student penalties, supporting fair enforcement and record-keeping within the dormitory.

3.2.0.: Flowchart Of Lestari Dormitory System (LDS) For Admin Summarize Penalties Per Staff



3.2.0.: Flowchart Of Lestari Dormitory System Admin Summarize Penalties Per Staff

Figure 3.2.0 illustrates the detailed flowchart for administrators summarizing penalties per staff within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, administrators access the main menu and select the option to summarize penalties assigned to each staff member. The system displays a summary list showing staff IDs, names, and total penalties assigned. Administrators can filter penalties by entering a specific date, upon which the system displays detailed penalty records for that date or a message if no records are found. The system also presents a list of active student accounts and a penalty collection summary showing total penalties collected and total amount collected. After reviewing the data, administrators may return to the main menu or exit the system. This structured flow enables efficient monitoring and reporting of penalty assignments, supporting transparent and effective dormitory management.



3.2.1. Flowchart of Lestari Dormitory System Staff

Figure 3.2.1. illustrates the flowchart for staff operations within the Lestari Dormitory System (LDS). After logging in with their staff ID and password, staff members access the staff menu, where they can choose from several options: count total users (admin, staff, student), view their assigned maintenance reports, or manage student penalties. For each function, the system guides the staff through relevant sub-menus and actions, such as viewing summaries, updating penalty statuses, or confirming receipts. The flowchart includes decision points for user choices and system checks (e.g., whether there are assigned reports or receipts to confirm), and allows staff to return to the main menu or exit the system at any point. This structured flow ensures staff can efficiently perform their roles in user management, maintenance tracking, and penalty administration, supporting effective dormitory operations.

3.3. Database Design

Business Rules for Lestari Dormitory System (LS):

a) Students log in using one unique matriculation number and password, which belongs to one student account.

b) Administrators log in using one unique staff ID and password, which belongs to one administrator account.

c) Each student is identified by one and only one matriculation number, and each matriculation number belongs to one student.

d) Each administrator is identified by one and only one staff ID, and each staff ID belongs to one administrator.

e) Students can submit zero or many maintenance requests, and each maintenance request is submitted by one and only one student.

f) Each maintenance request has one and only one status, and each status belongs to one maintenance request.

g) Each maintenance request is assigned to one category, and each category can be assigned to zero or many maintenance requests.

h) Each maintenance request has one and only one priority, and each priority belongs to one maintenance request.

i) Administrators can manage zero or many student accounts, and each student account is managed by one and only one administrator.

j) Administrators can create zero or many administrator accounts, and each administrator account is managed by one and only one administrator.

k) Administrators can view zero or many reports, and each report is viewed by one and only one administrator.

l) Each maintenance request is assigned to one and only one staff member, and each staff member can be assigned zero or many maintenance requests.

m) Staff members log in using one unique staff ID and password, which belongs to one staff account.

n) Staff members can view zero or many maintenance requests assigned to them, and each maintenance request is assigned to one and only one staff member.

o) Staff members can update the status of maintenance requests assigned to them, including marking requests as resolved, rejected, or on hold.

p) If a staff member becomes inactive, their assigned maintenance requests can be reassigned to active staff members by administrators.

q) Each penalty is linked to one and only one student, and each student can have zero or many penalties.

r) Each penalty is created and managed by one and only one staff member, and each staff member can manage zero or many penalties.

s) Each penalty has one and only one status (e.g., Pending, Confirmed, Paid), and each status belongs to one penalty.

t) Students can view their own penalties but cannot modify penalty details or status.

u) Staff members can update penalty statuses and confirm receipt of penalty payments.

v) Penalty payments generate receipts that require confirmation by staff members.

w) The system maintains a penalty collection summary, including total penalties collected and total amount collected.

x) Each student account is linked to one and only one room, and each room can be assigned to zero or many students.

y) Each student account has one and only one email address, which is unique within the system.

z) The system enforces unique constraints on all identifiers: matriculation numbers for students, staff IDs for administrators and staff, and penalty IDs.

3.3.1 Conceptual Design : Entity Relationship Diagram (ERD).

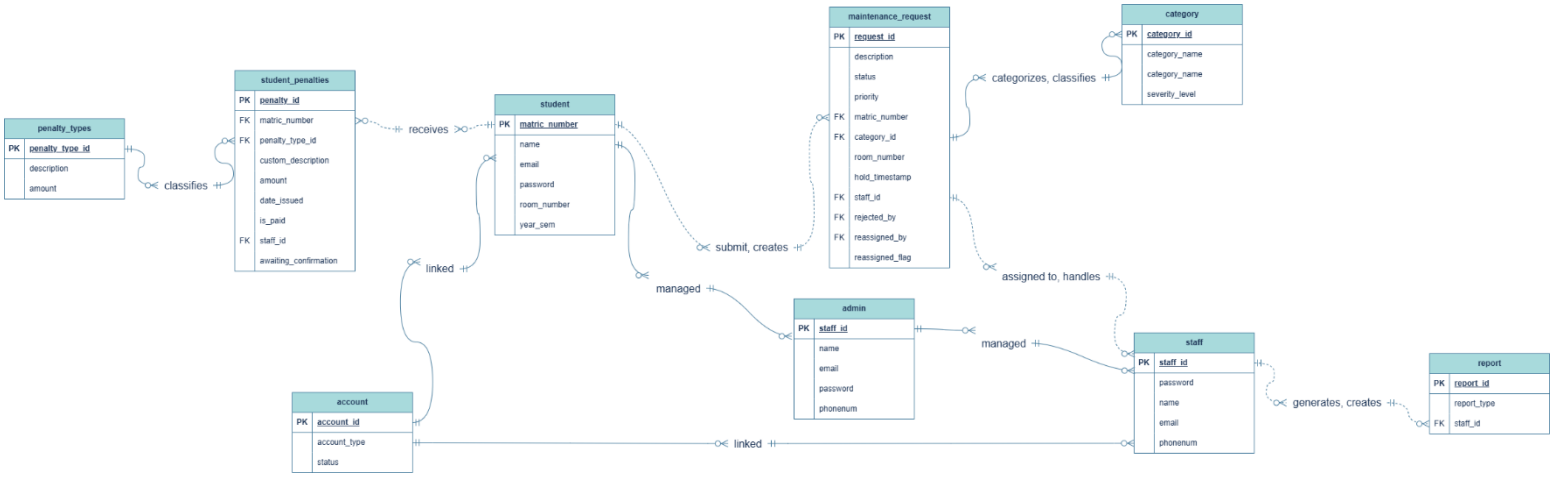


Figure 3.3.1.: Entity Relationship Diagram of Lestari Dormitory System

3.3.2 Logical Design: Data Dictionary

The ERD is translated into the logical design in form of data dictionary. The data types in the data dictionary are defined based on selected DBMS which is MySQL and the tables in this section represent data dictionaries of the entities.

3.3.2.1 account Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
account_id	varchar(30)	Yes	Primary Key	
account_type	varchar(20)	No		
status	varchar(30)	No		

Figure 3.3.2.1.: account Table

3.3.2.2 admin Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
staff_id	varchar(20)	Yes	Primary Key	
name	varchar(50)	Yes		

email	varchar(50)	Yes		
password	varchar(255)	Yes		
phonenum	varchar(20)	Yes		

Figure 3.3.2.2.: admin Table

3.3.2.3 category Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
category_id	varchar(20)	Yes	Primary Key	
category_name	varchar(30)	No		
severity_level	varchar(30)	No		

Figure 3.3.2.3.: category Table

3.3.2.4 maintenance_request Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
request_id	varchar(20)	Yes	Primary Key	
description	text	Yes		
status	varchar(20)	Yes		
priority	varchar(20)	Yes		
matric_number	varchar(20)	Yes	Foreign Key	Student
category_id	varchar(20)	Yes	Foreign Key	Category
room_number	varchar(20)	Yes		
hold_timestamp	datetime	Yes		
staff_id	varchar(20)	Yes	Foreign Key	Staff
rejected_by	varchar(20)	Yes	Foreign Key	Staff
reassigned_by	varchar(20)	No	Foreign Key	Staff
reassigned_flag	int(11)	No		Default: 0

Figure 3.3.2.4.: maintenance_request Table

3.3.2.5 penalty_types Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
penalty_type_id	int(11)	Yes	Primary Key	
description	varchar(100)	Yes		
amount	decimal(10,2)	Yes		

Figure 3.3.2.5.: penalty_types Table

3.3.2.6 report Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
report_id	varchar(50)	Yes	Primary Key	
report_type	varchar(50)	No		
staff_id	varchar(20)	No	Foreign Key	Staff

Figure 3.3.2.6.: report Table

3.3.2.7 staff Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
staff_id	varchar(20)	Yes	Primary Key	
name	varchar(50)	Yes		
email	varchar(50)	Yes		
password	varchar(255)	Yes		
phonenum	varchar(20)	Yes		

Figure 3.3.2.7.: staff Table

3.3.2.8 student Table

Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
matric_number	varchar(20)	Yes	Primary Key	
name	varchar(50)	Yes		
email	varchar(50)	Yes		
password	varchar(255)	Yes		
room_number	varchar(20)	Yes	Foreign Key	
year_sem	varchar(30)	Yes		

Figure 3.3.2.8.: student Table

3.3.2.9 student_penalties Table

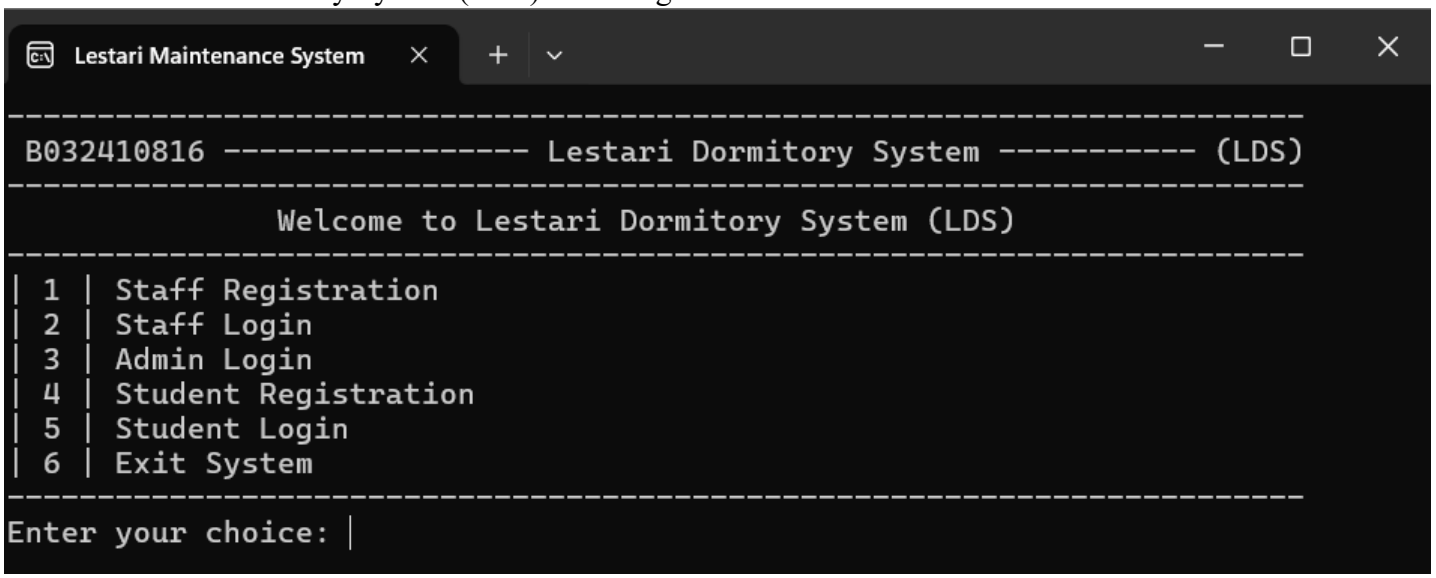
Attribute Name	Data Type & Size	Required?	PK/FK	Reference Table
penalty_id	int(11)	Yes	Primary Key	
matric_number	varchar(20)	Yes	Foreign Key	Student
penalty_type_id	int(11)	No	Foreign Key	PenaltyTypes
custom_description	varchar(255)	No		
amount	decimal(10,2)	Yes		
date_issued	datetime	No		Default: current_timestamp()
is_paid	tinyint(1)	Yes		Default: 0
staff_id	varchar(20)	No	Foreign Key	Staff
awaiting_confirmation	tinyint(1)	Yes		Default: 1

Figure 3.3.2.9.: student_penalties Table

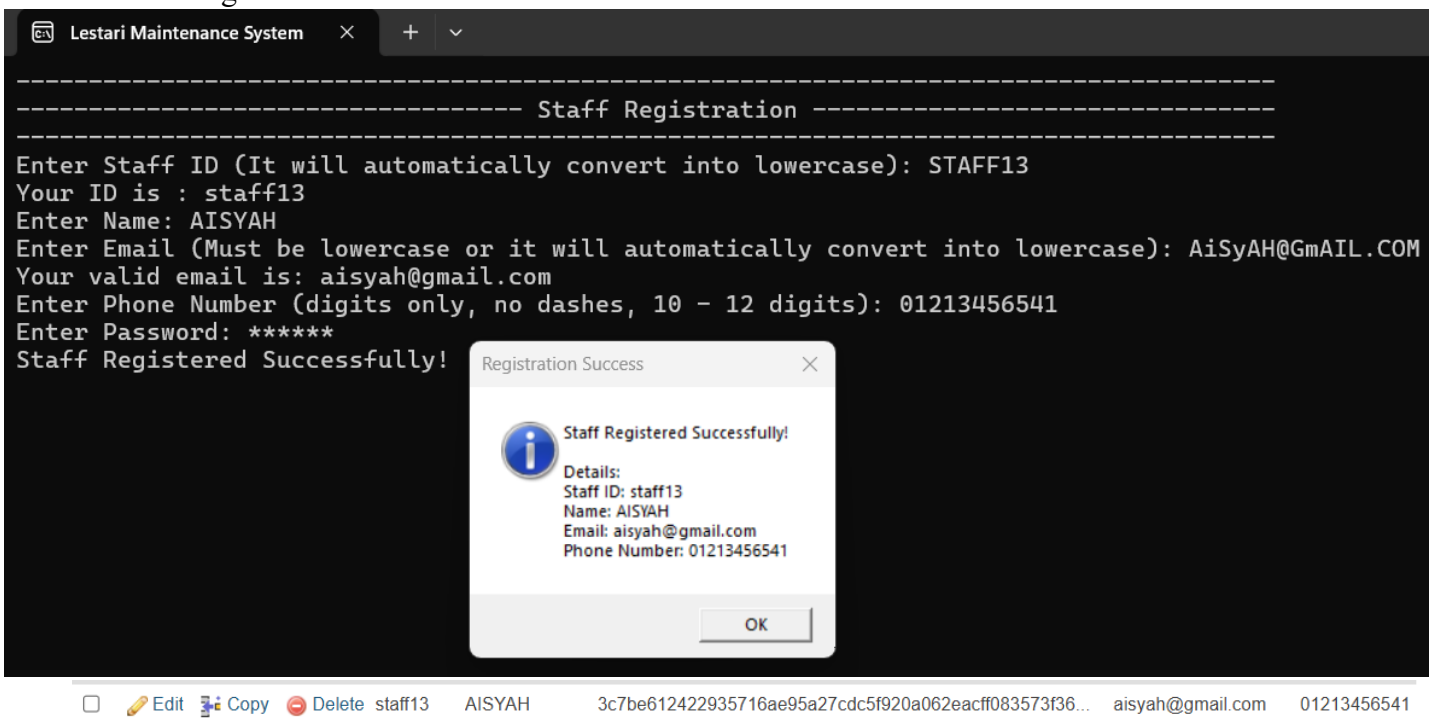
3.4. Interface Design

The interface design of the system

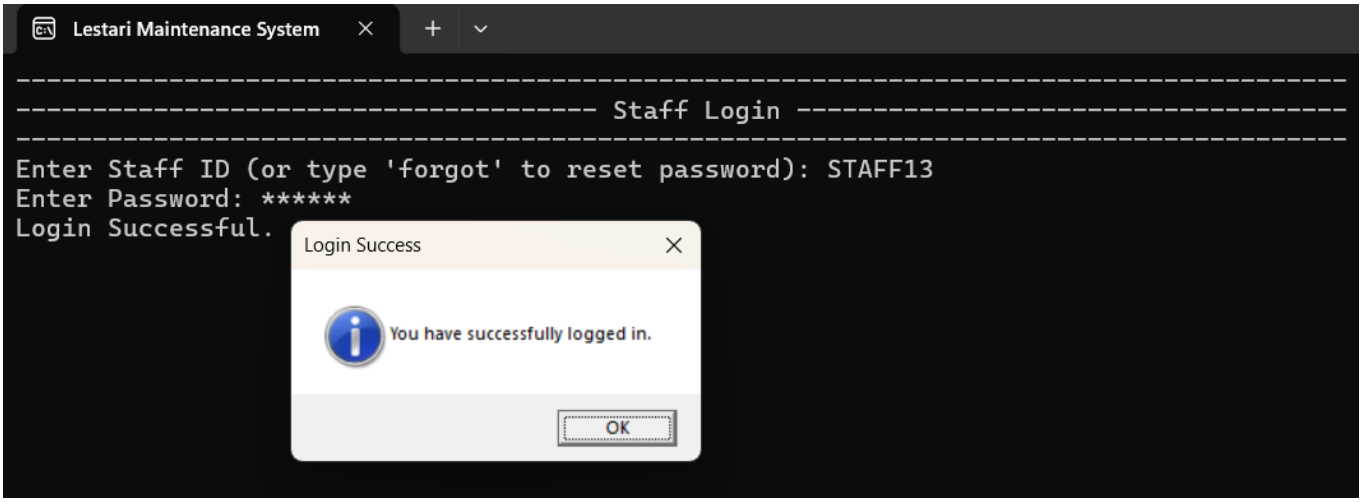
1. Lestari Dormitory System (LDS) Main Page



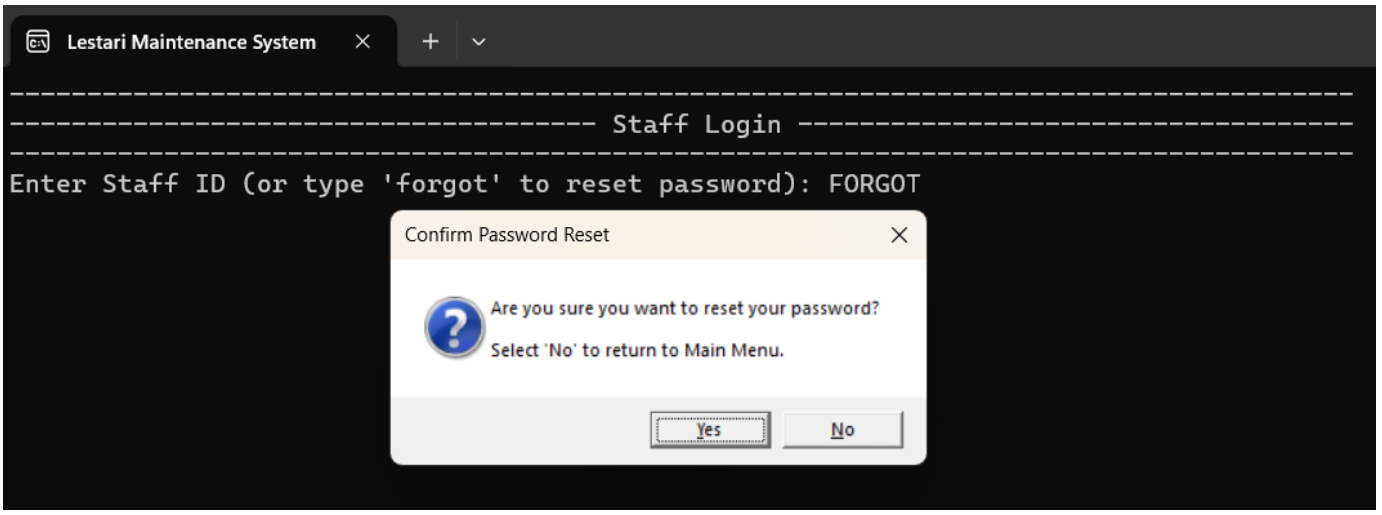
2. Staff Registration

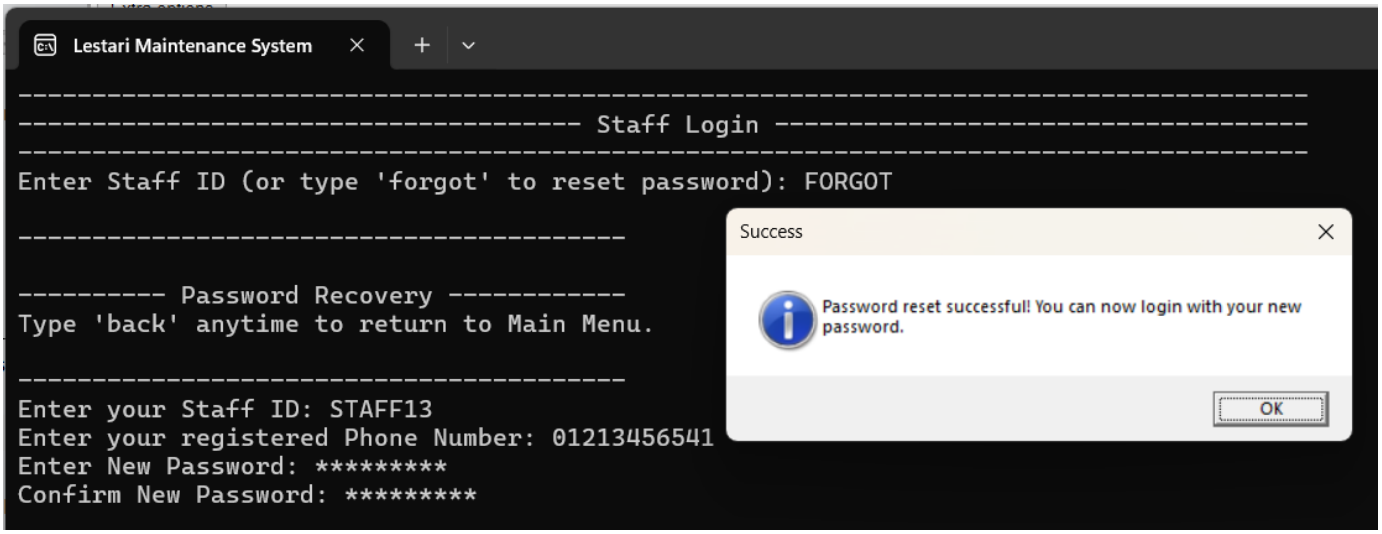


3. Staff Login

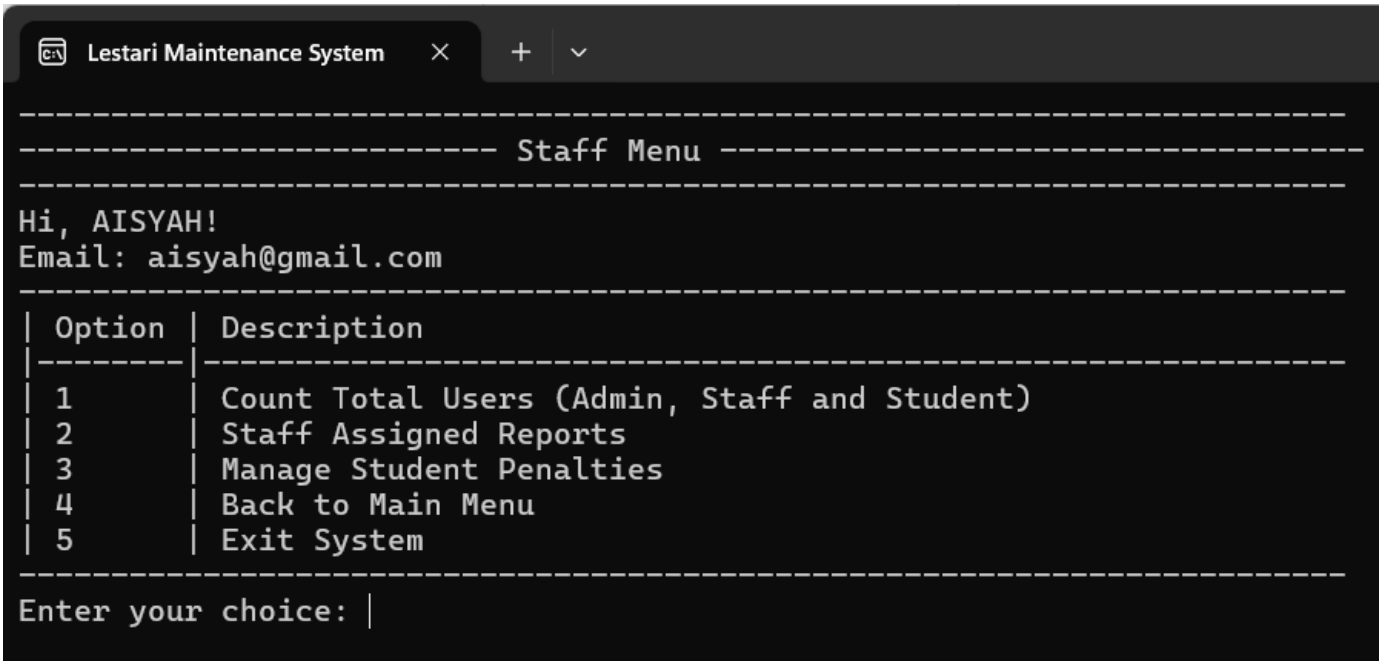


4. Staff Forget Password

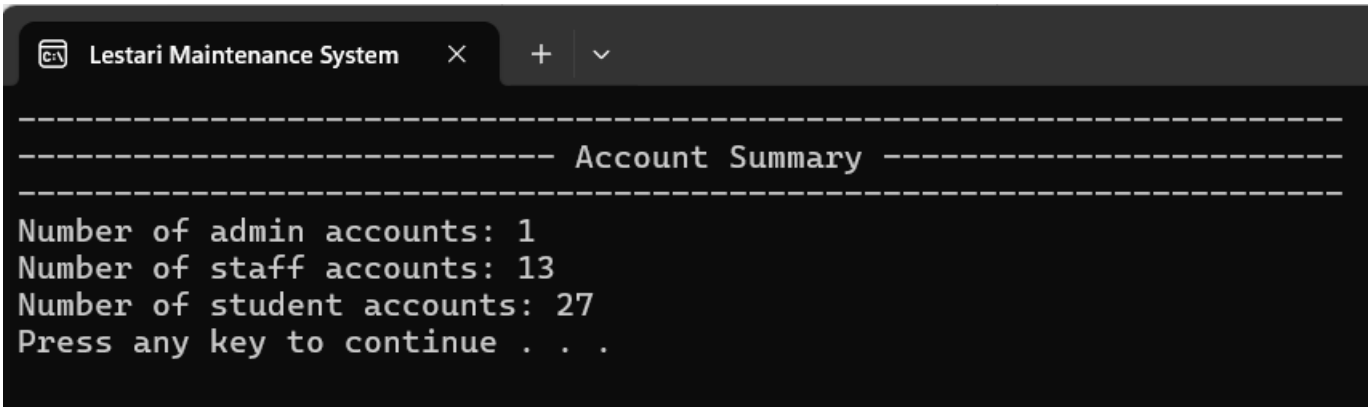




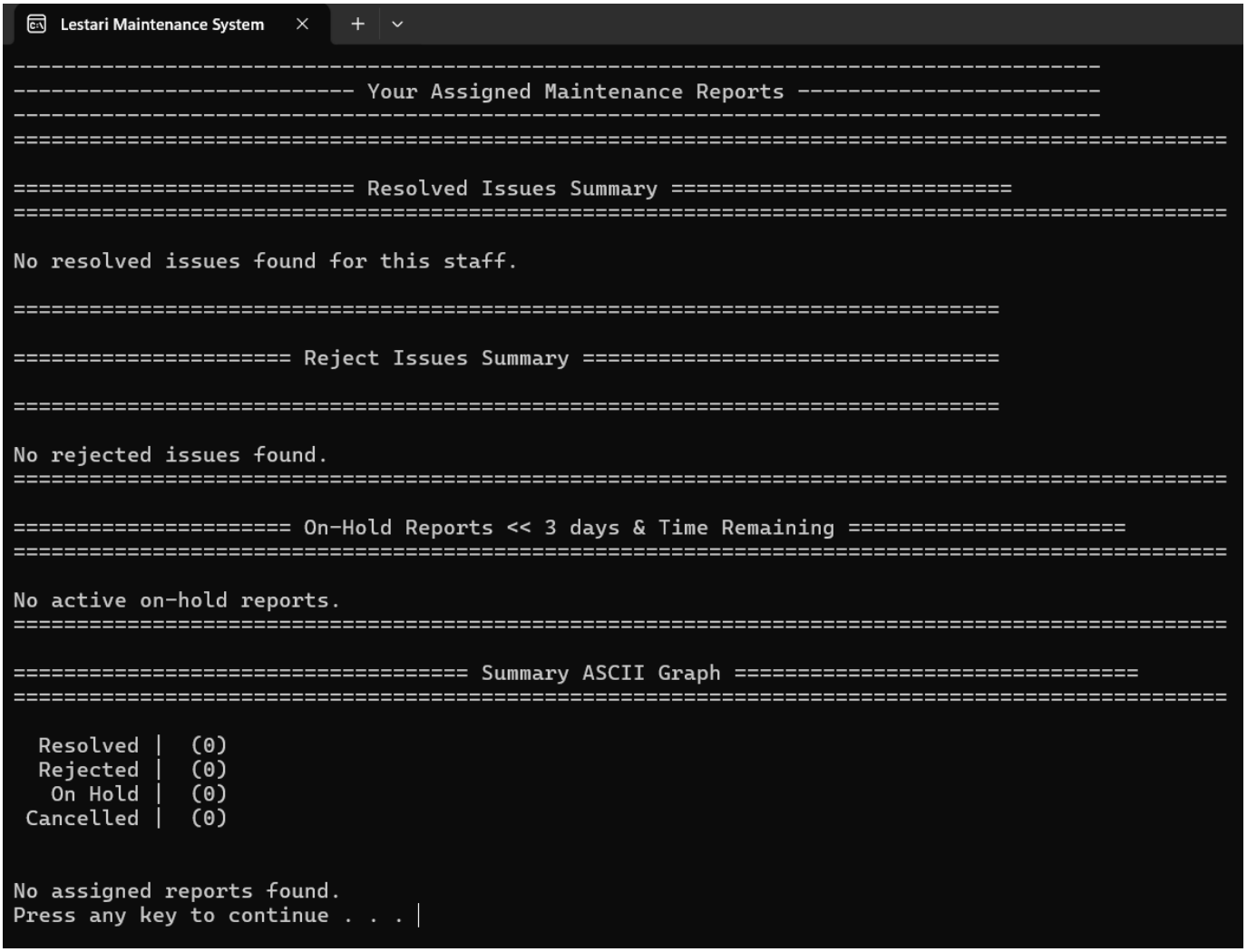
5. Staff Main Menu



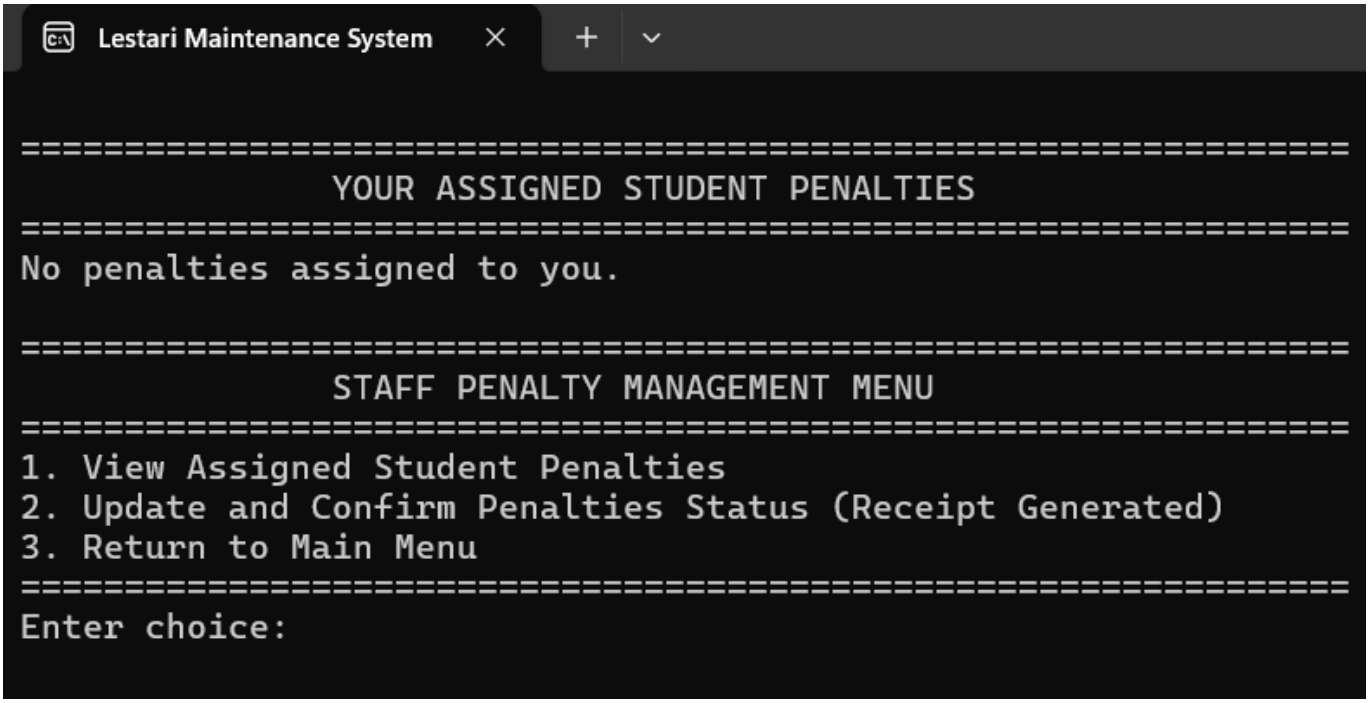
6. Staff – Count Total Users (Admin, Staff and Student)



7. Staff – Staff Assigned Reports

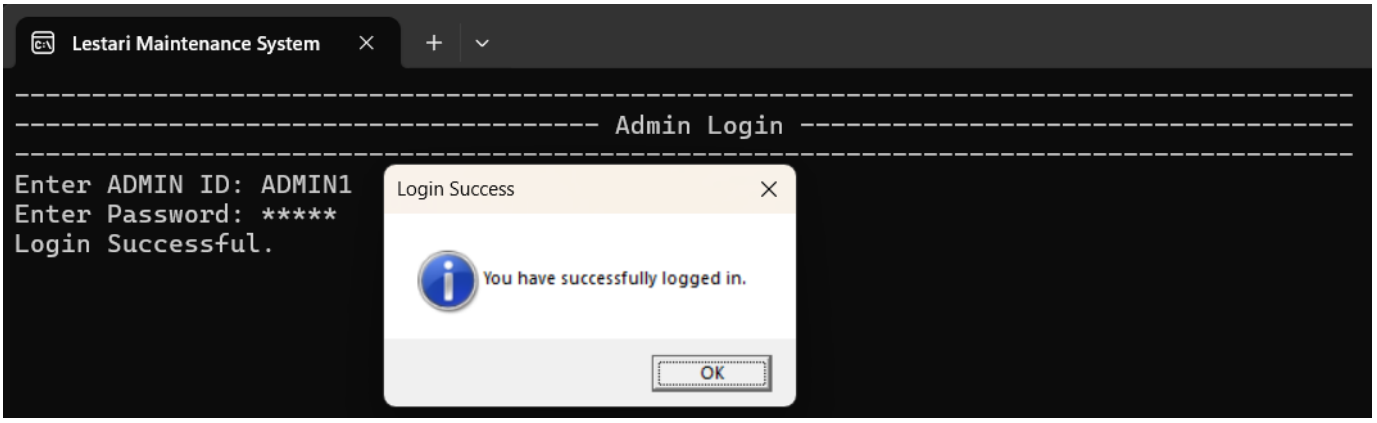


8. Staff – Manage Student Penalties

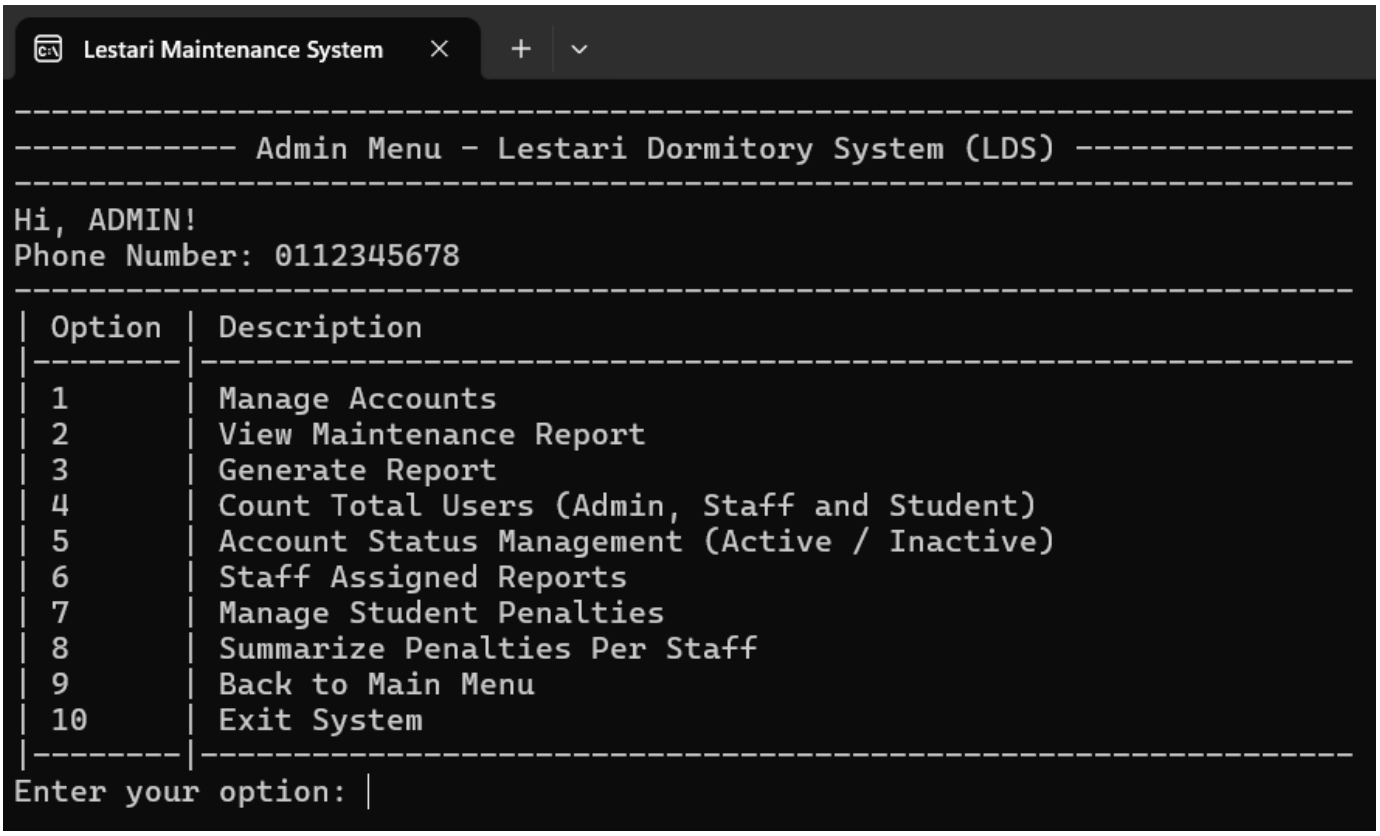


9. Admin Login

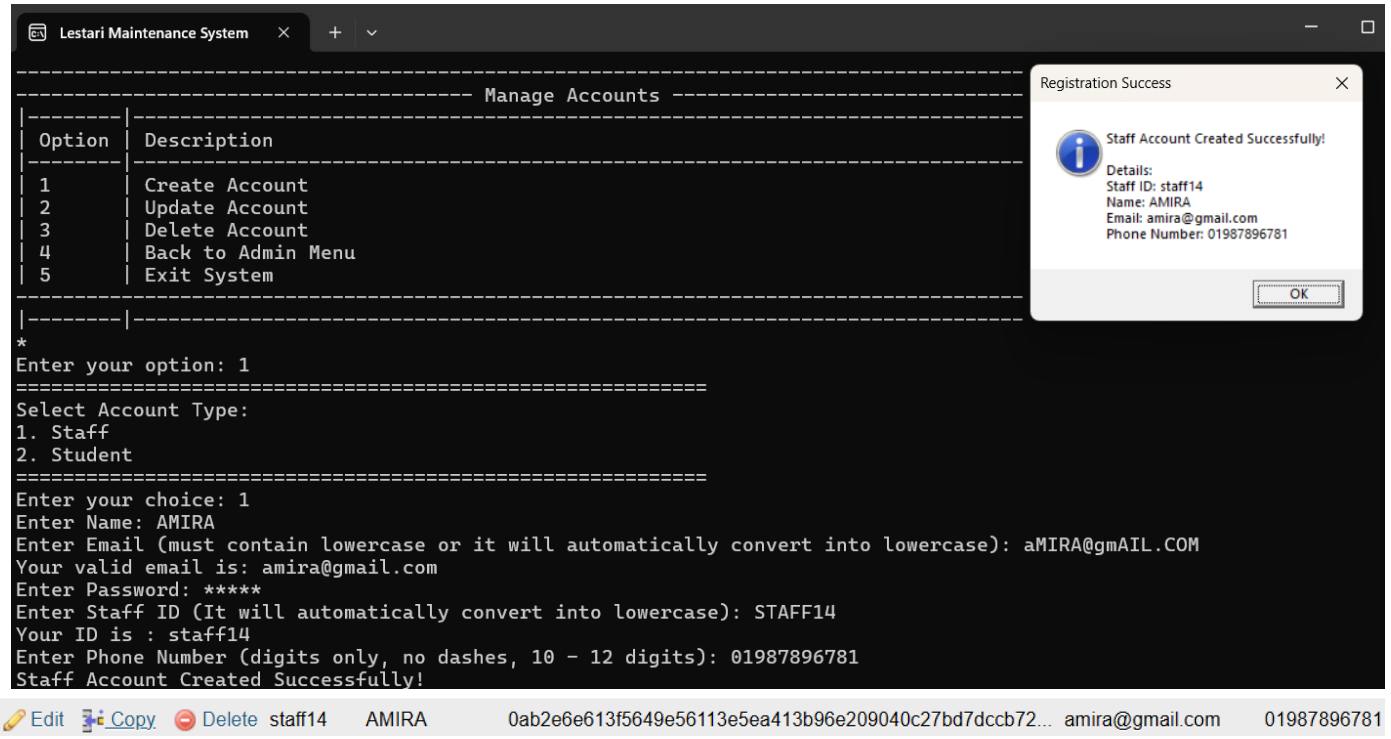
```
--------------------------------------------------------------------------------
------------------------------------ Admin Login -------------------------------
--------------------------------------------------------------------------------
Enter ADMIN ID: admin1
Enter Password: *****
Invalid Credentials! Please enter exact ADMIN ID and Password.
Press any key to continue . . .
```



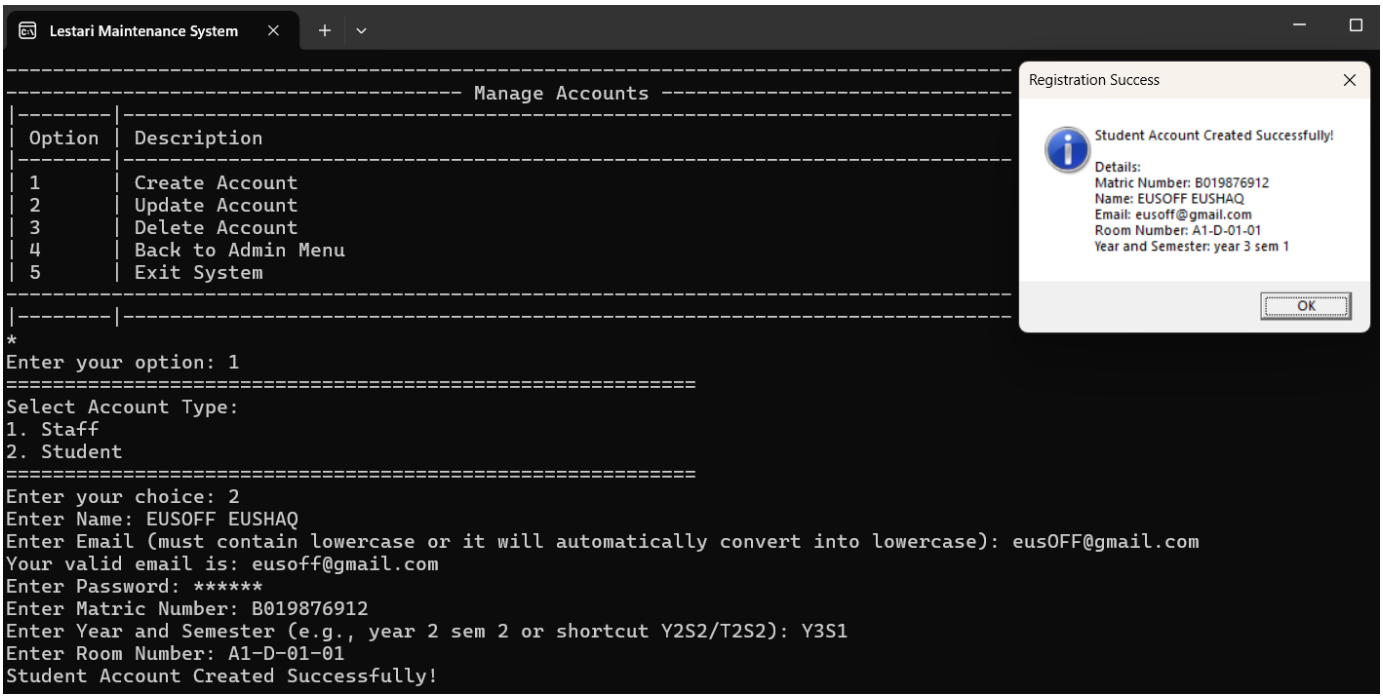
10. Admin Menu

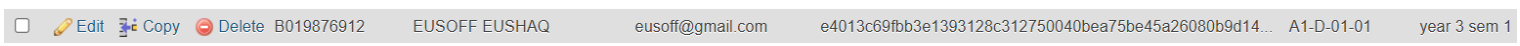


11. Admin – Create Staff Account

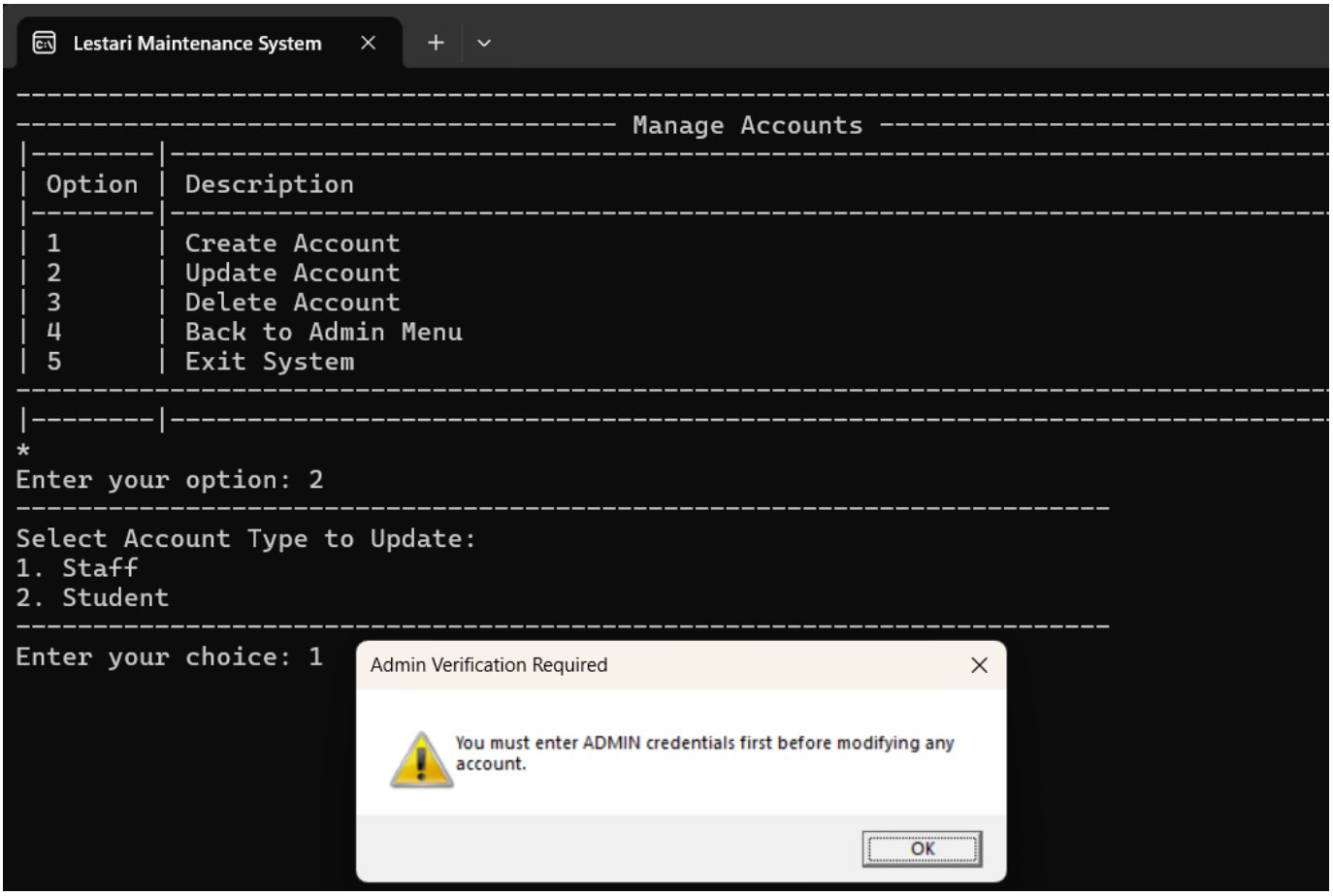


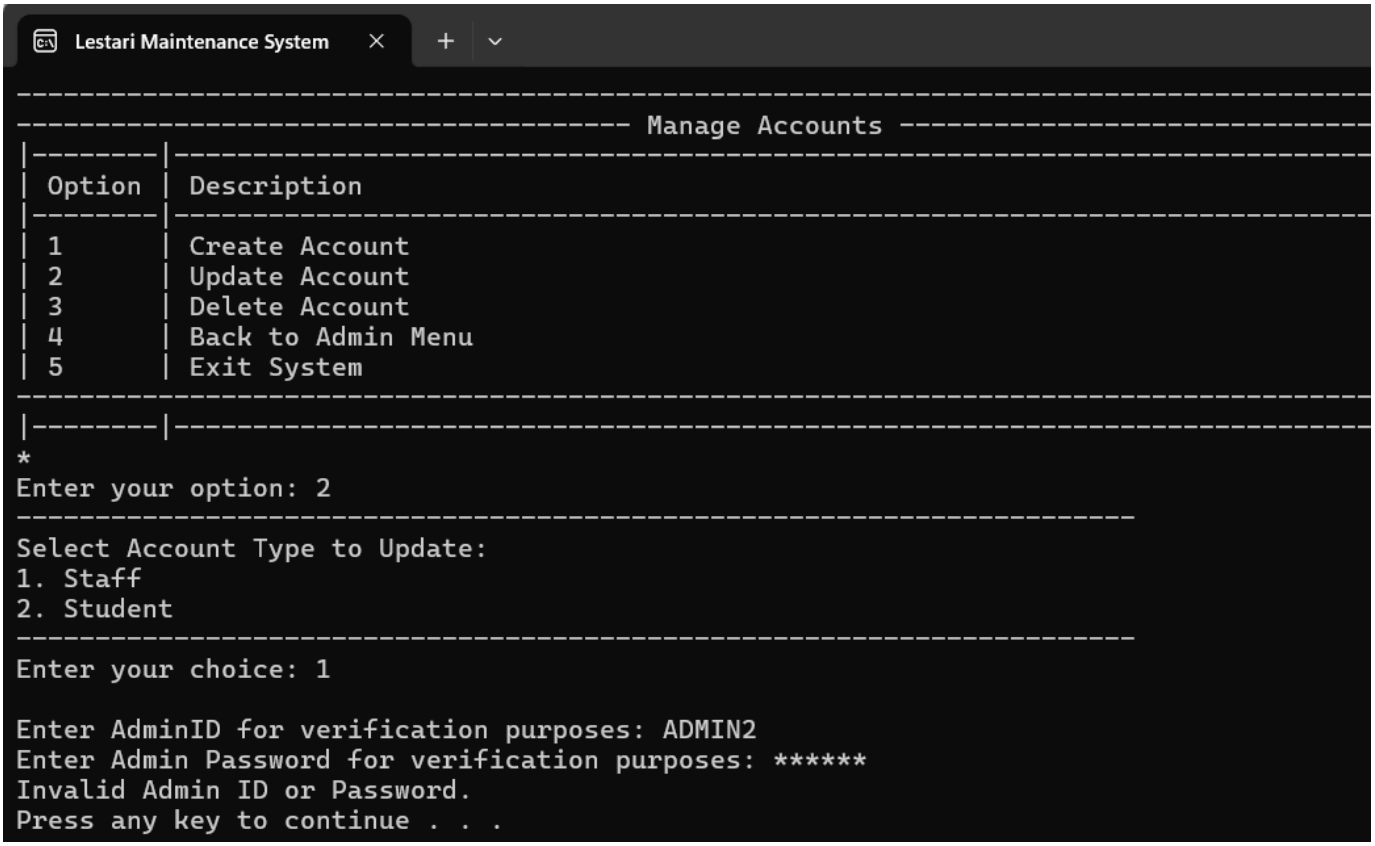
12. Admin – Create Student Account

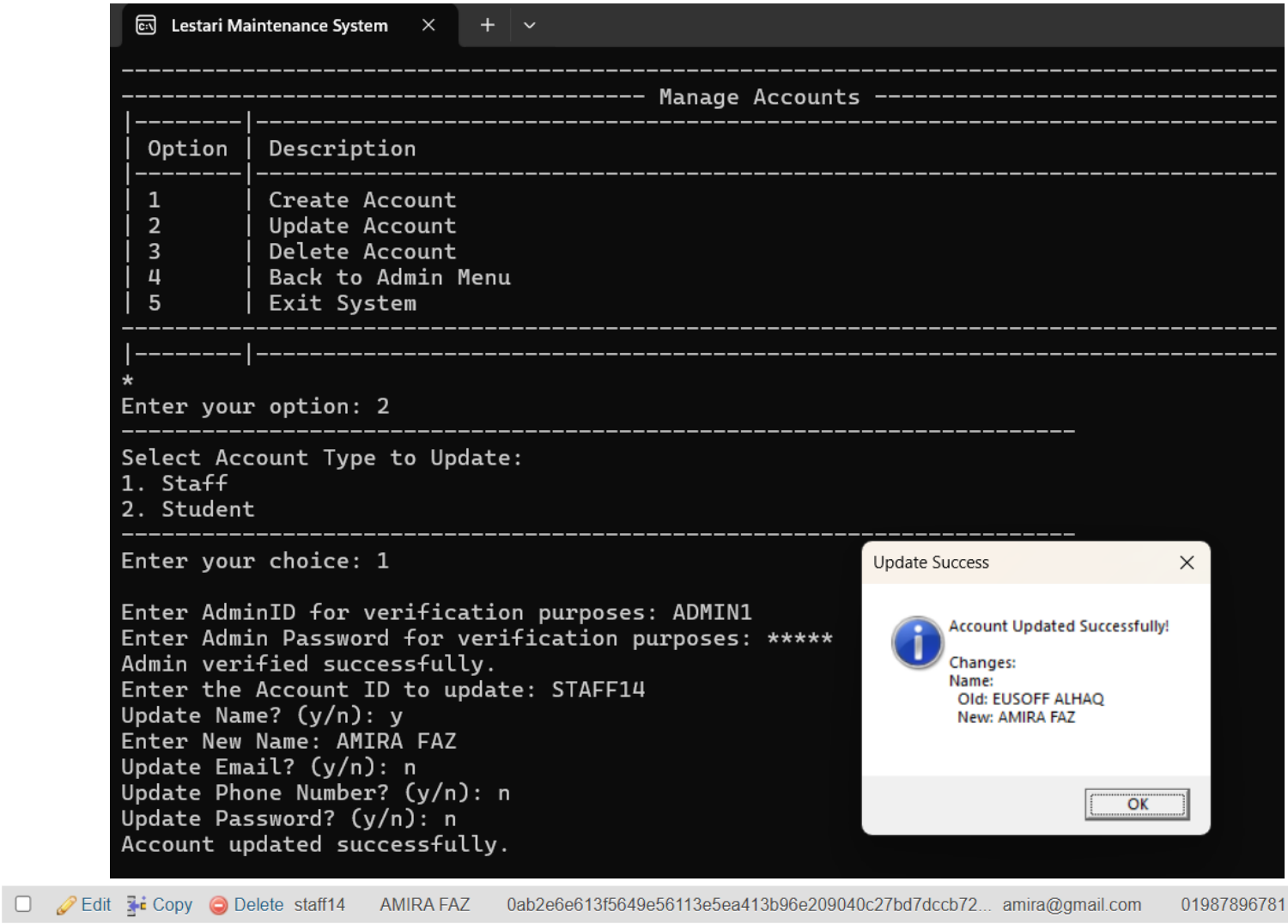




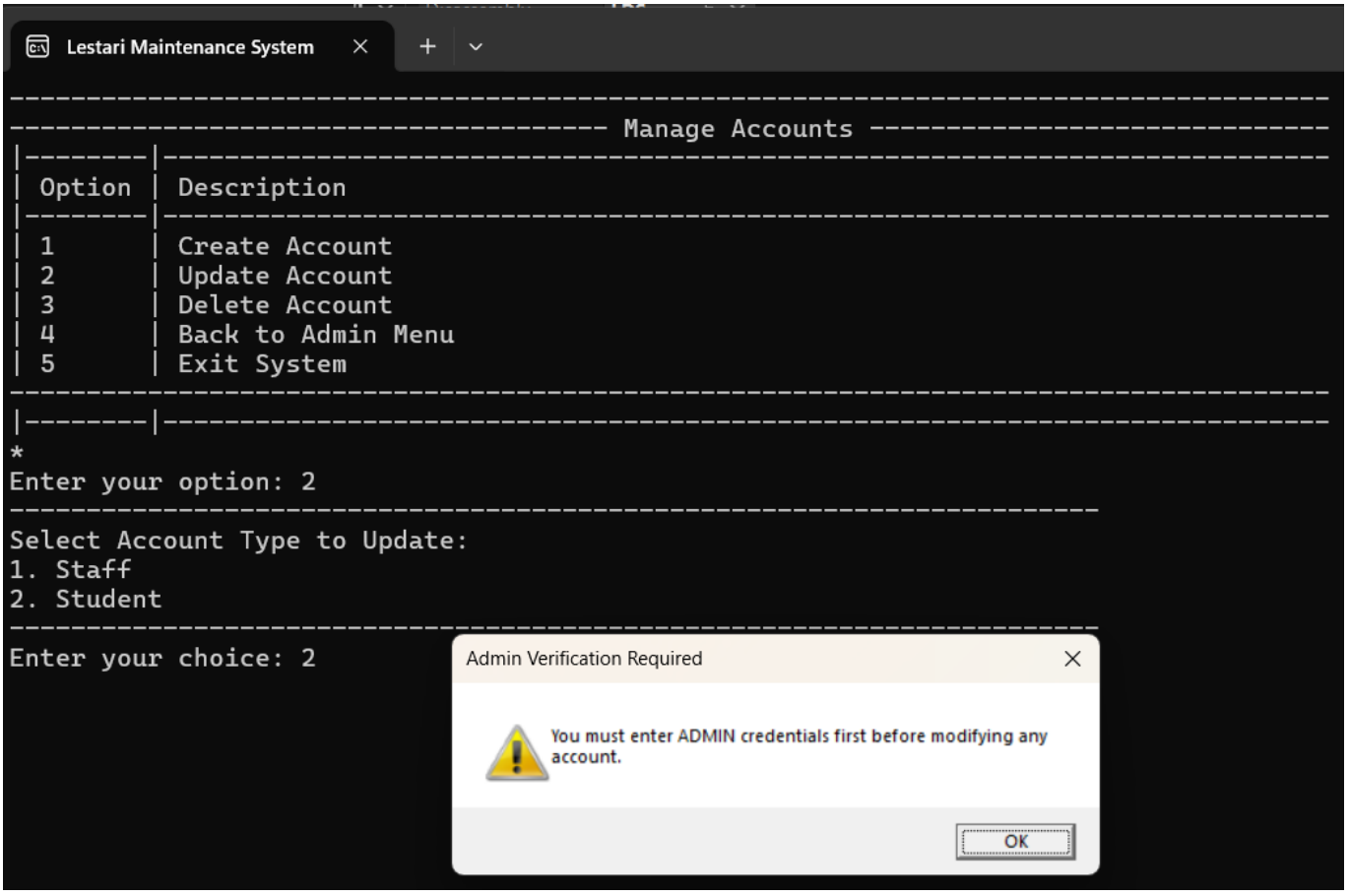
13. Admin – Update Staff Account

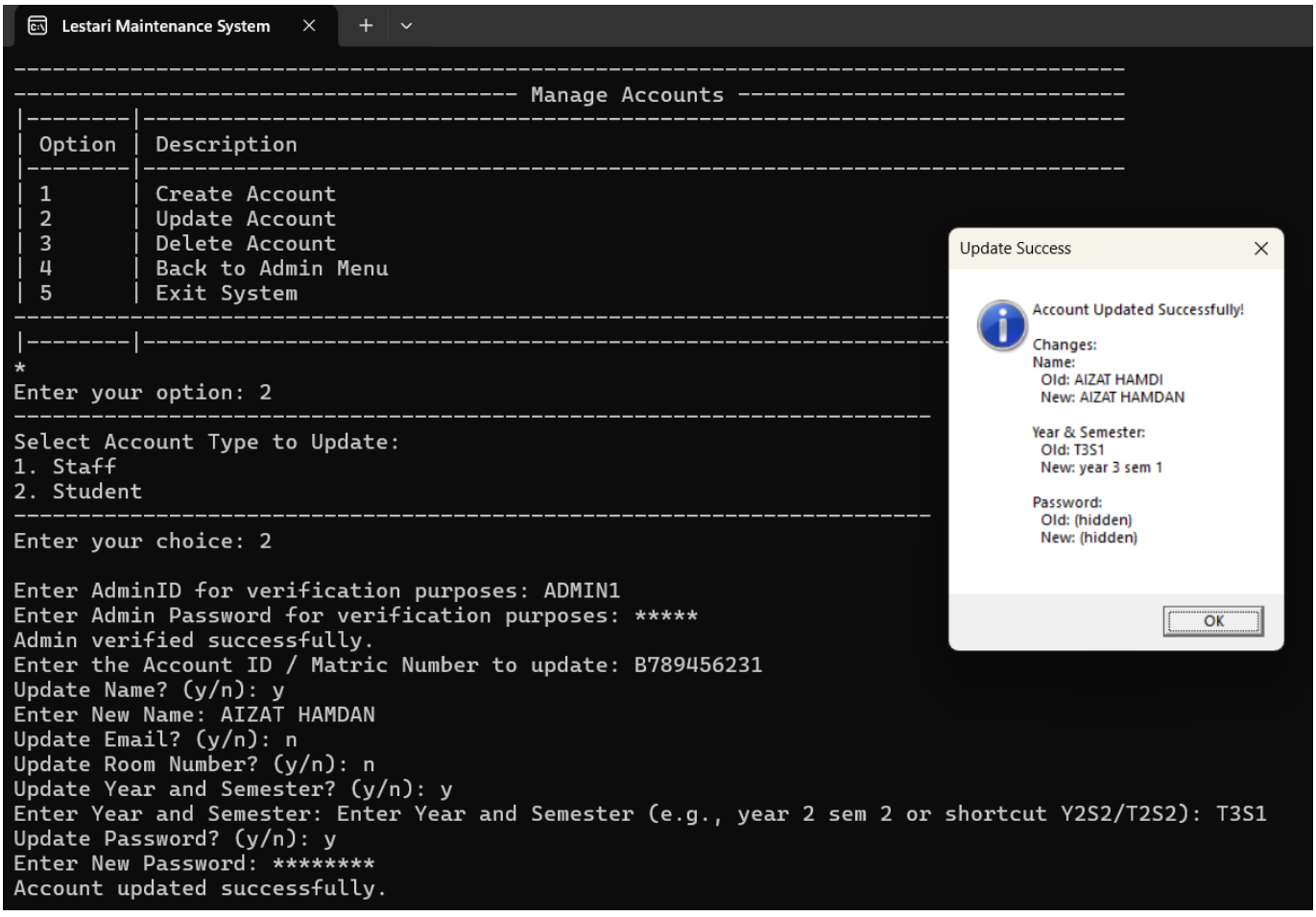

Lestari Maintenance System
Manage Accounts
Option | Description
1 | Create Account
2 | Update Account
3 | Delete Account
4 | Back to Admin Menu
5 | Exit System
Enter your option: 2
Select Account Type to Update:
1. Staff
2. Student
Enter your choice: 1
Admin Verification Required
You must enter ADMIN credentials first before modifying any account.
OK


Lestari Maintenance System
Manage Accounts
Option | Description
1 | Create Account
2 | Update Account
3 | Delete Account
4 | Back to Admin Menu
5 | Exit System
Enter your option: 2
Select Account Type to Update:
1. Staff
2. Student
Enter your choice: 1
Enter AdminID for verification purposes: ADMIN2
Enter Admin Password for verification purposes: ******
Invalid Admin ID or Password.
Press any key to continue . . .

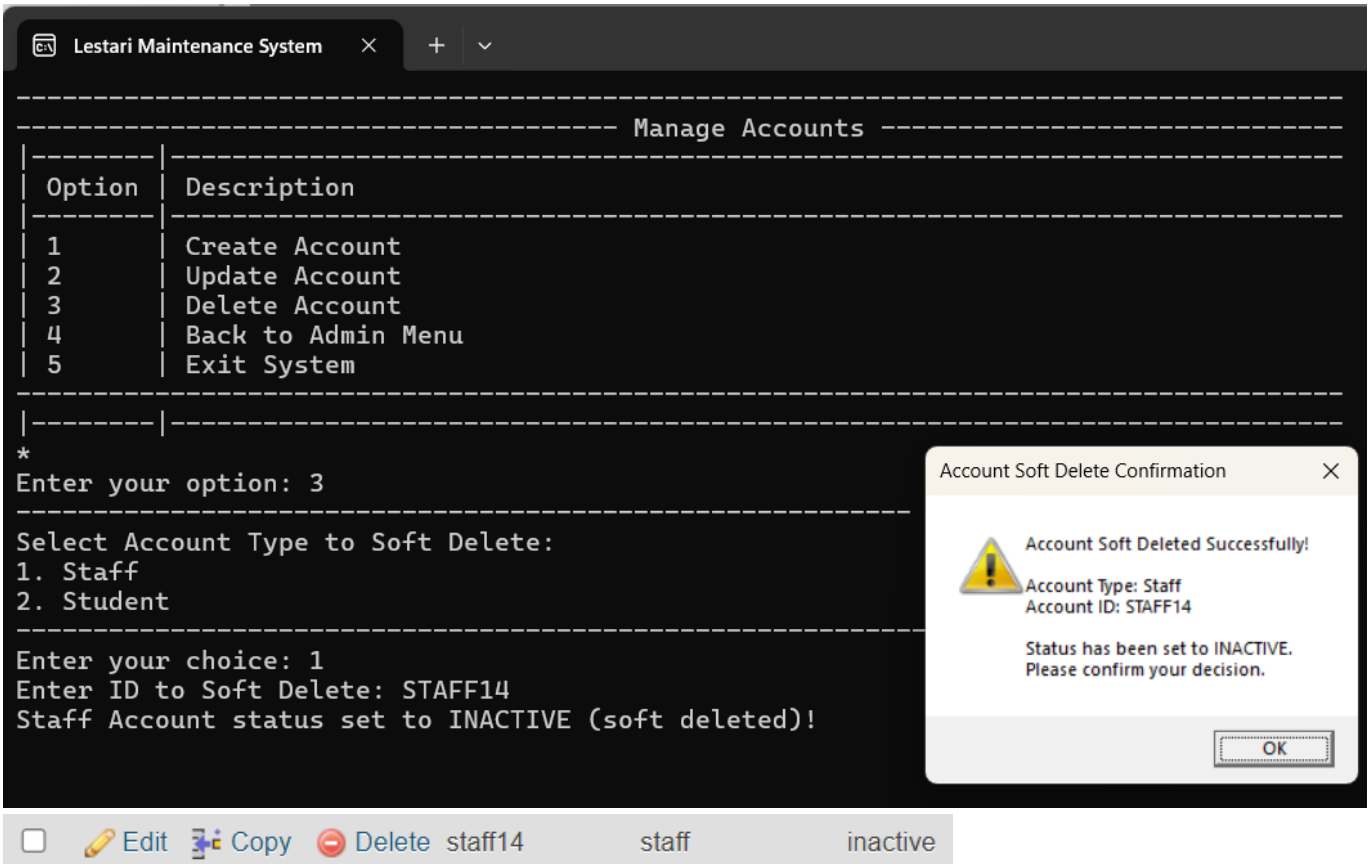


14. Admin – Update Student Account

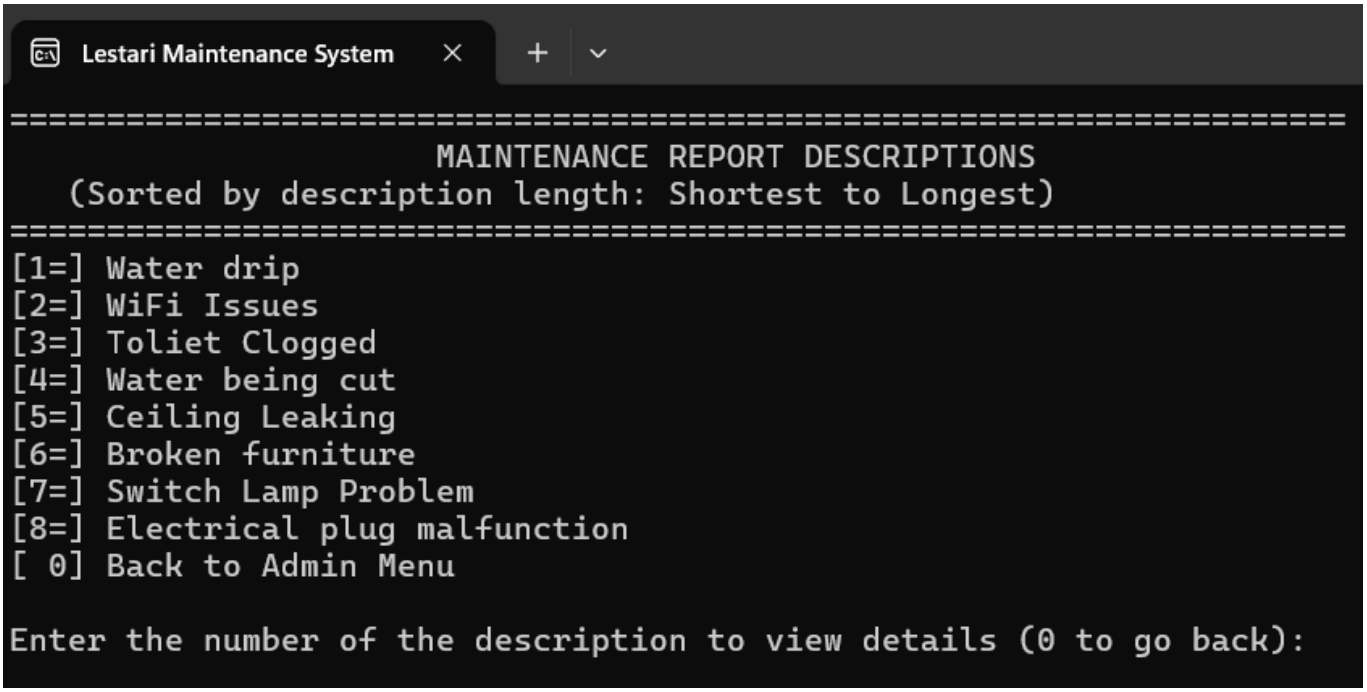

Lestari Maintenance System
Manage Accounts
Option | Description
1 | Create Account
2 | Update Account
3 | Delete Account
4 | Back to Admin Menu
5 | Exit System
Enter your option: 2
Select Account Type to Update:
1. Staff
2. Student
Enter your choice: 2
Admin Verification Required
You must enter ADMIN credentials first before modifying any account.
OK

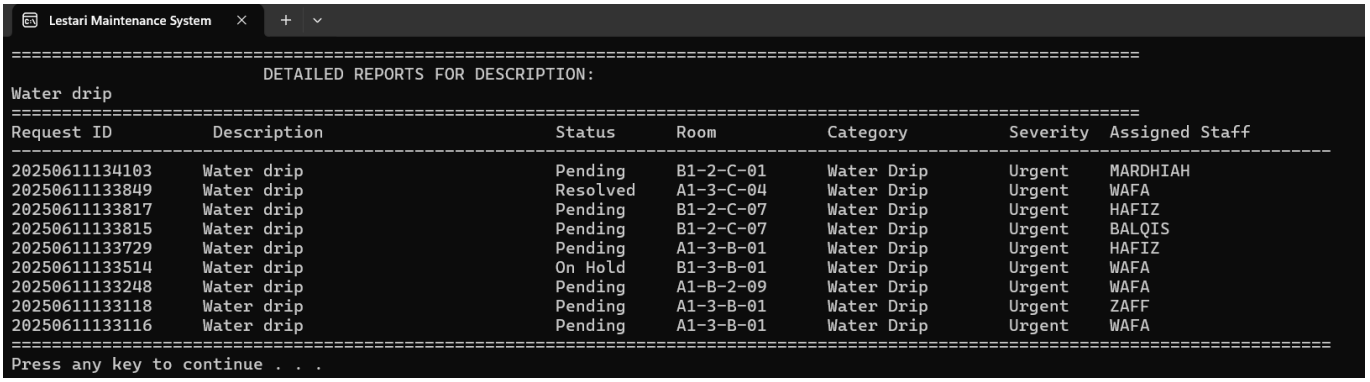

Lestari Maintenance System
Manage Accounts
Option | Description
1 | Create Account
2 | Update Account
3 | Delete Account
4 | Back to Admin Menu
5 | Exit System
Enter your option: 2
Select Account Type to Update:
1. Staff
2. Student
Enter your choice: 2
Enter AdminID for verification purposes: ADMIN1
Enter Admin Password for verification purposes: *****
Admin verified successfully.
Enter the Account ID / Matric Number to update: B789456231
Update Name? (y/n): y
Enter New Name: AIZAT HAMDAN
Update Email? (y/n): n
Update Room Number? (y/n): n
Update Year and Semester? (y/n): y
Enter Year and Semester: Enter Year and Semester (e.g., year 2 sem 2 or shortcut Y2S2/T2S2): T3S1
Update Password? (y/n): y
Enter New Password: ********
Account updated successfully.
Update Success
Account Updated Successfully!
Changes:
Name:
Old: AIZAT HAMDI
New: AIZAT HAMDAN
Year & Semester:
Old: T3S1
New: year 3 sem 1
Password:
Old: (hidden)
New: (hidden)
OK

15. Admin – Delete Account (Soft Delete)

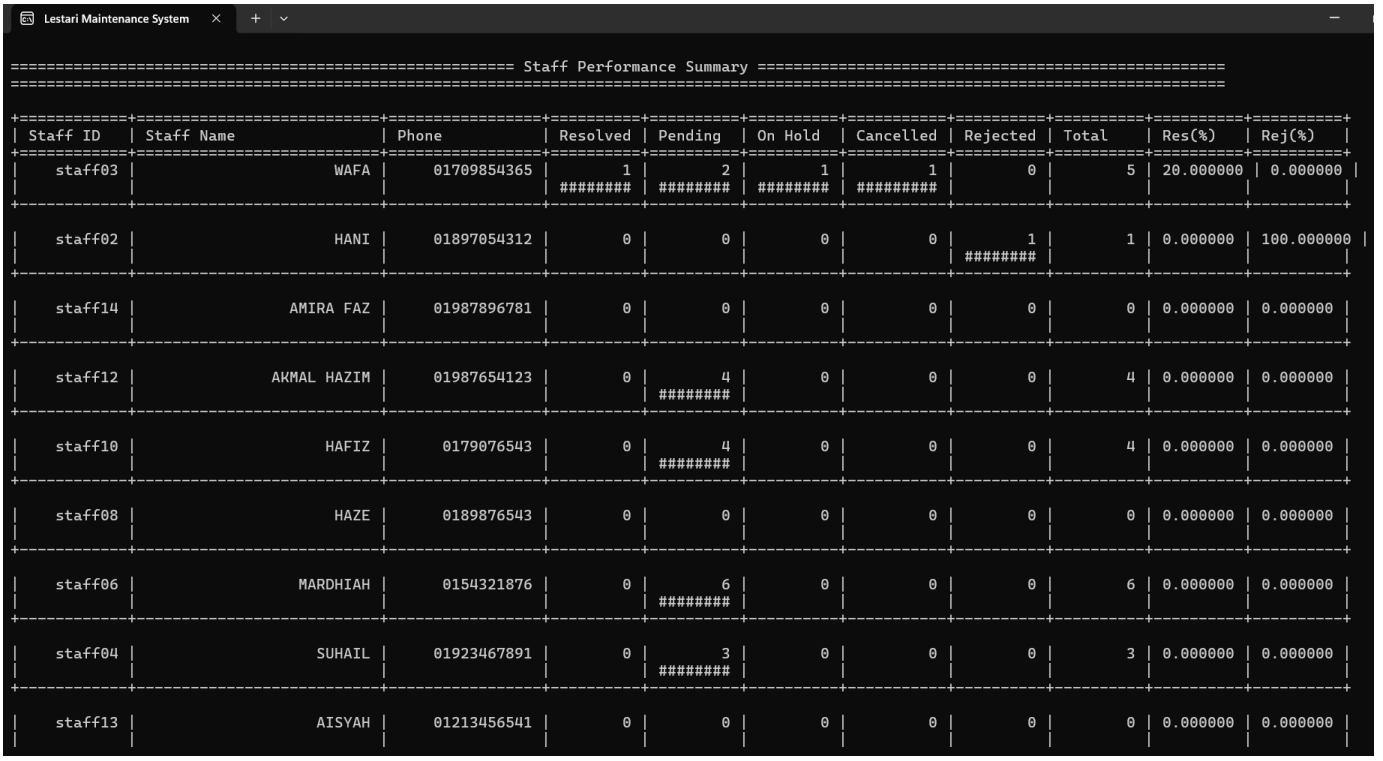


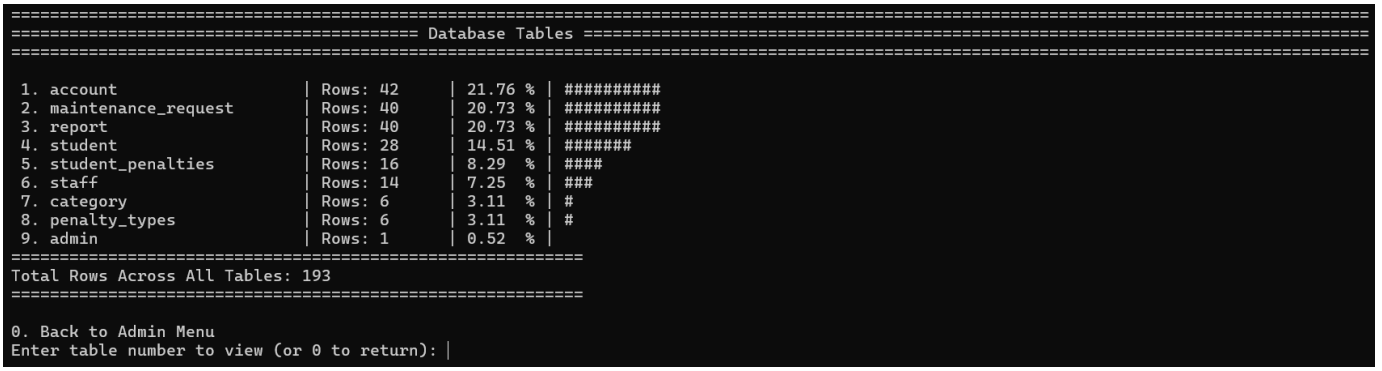
16. Admin – View Maintenance Report



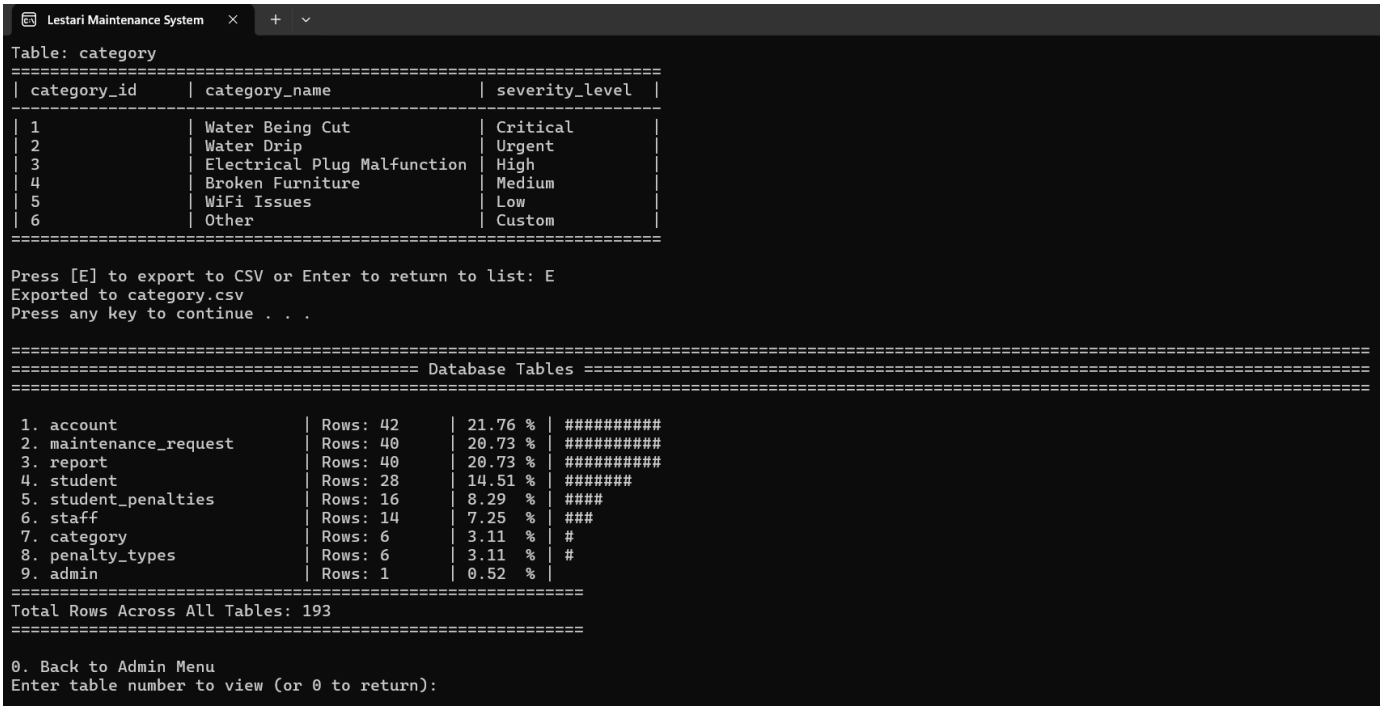


17. Admin – Generate Report

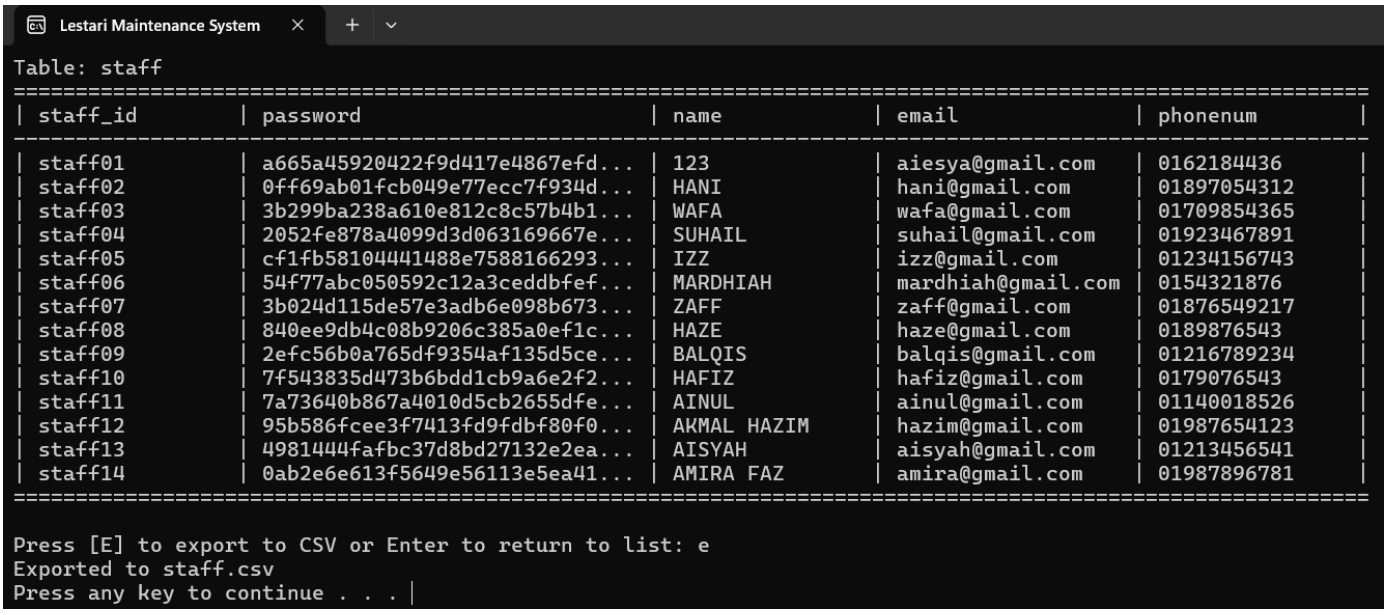




```
============================================================================================================================================
========================================== Database Tables ==========================================
============================================================================================================================================

 1. account                  | Rows: 42     | 21.76 % | ##########
 2. maintenance_request      | Rows: 40     | 20.73 % | ##########
 3. report                   | Rows: 40     | 20.73 % | ##########
 4. student                  | Rows: 28     | 14.51 % | #######
 5. student_penalties        | Rows: 16     | 8.29  % | ####
 6. staff                    | Rows: 14     | 7.25  % | ###
 7. category                 | Rows: 6      | 3.11  % | #
 8. penalty_types            | Rows: 6      | 3.11  % | #
 9. admin                    | Rows: 1      | 0.52  % |
============================================================
Total Rows Across All Tables: 193
============================================================

0. Back to Admin Menu
Enter table number to view (or 0 to return):
```

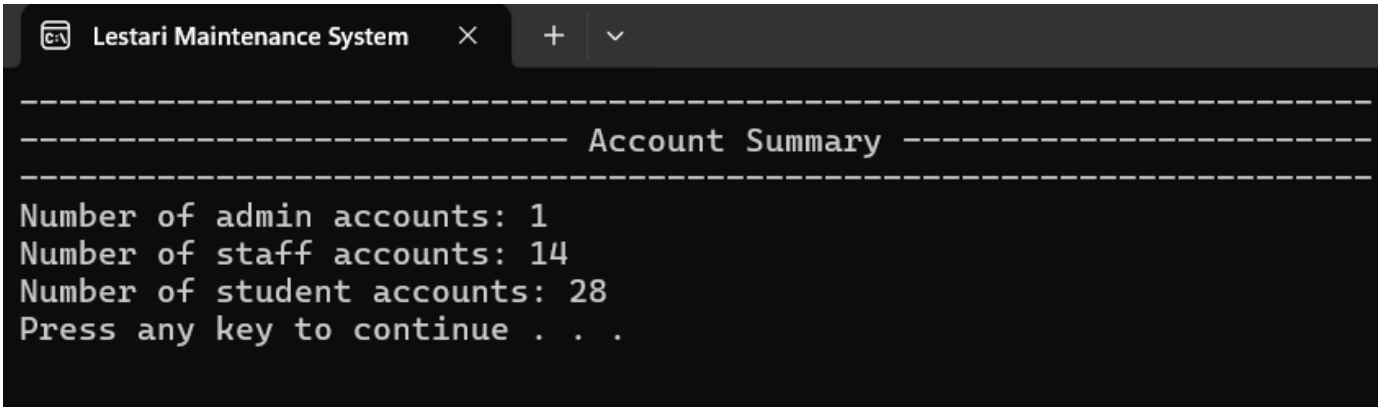


```
Lestari Maintenance System

Table: category
================================================================
| category_id     | category_name               | severity_level  |
----------------------------------------------------------------
| 1               | Water Being Cut             | Critical        |
| 2               | Water Drip                  | Urgent          |
| 3               | Electrical Plug Malfunction | High            |
| 4               | Broken Furniture            | Medium          |
| 5               | WiFi Issues                 | Low             |
| 6               | Other                       | Custom          |
================================================================

Press [E] to export to CSV or Enter to return to list: E
Exported to category.csv
Press any key to continue . . .

============================================================================================================================================
========================================== Database Tables ==========================================
============================================================================================================================================

 1. account                  | Rows: 42     | 21.76 % | ##########
 2. maintenance_request      | Rows: 40     | 20.73 % | ##########
 3. report                   | Rows: 40     | 20.73 % | ##########
 4. student                  | Rows: 28     | 14.51 % | #######
 5. student_penalties        | Rows: 16     | 8.29  % | ####
 6. staff                    | Rows: 14     | 7.25  % | ###
 7. category                 | Rows: 6      | 3.11  % | #
 8. penalty_types            | Rows: 6      | 3.11  % | #
 9. admin                    | Rows: 1      | 0.52  % |
============================================================
Total Rows Across All Tables: 193
============================================================

0. Back to Admin Menu
Enter table number to view (or 0 to return):
```

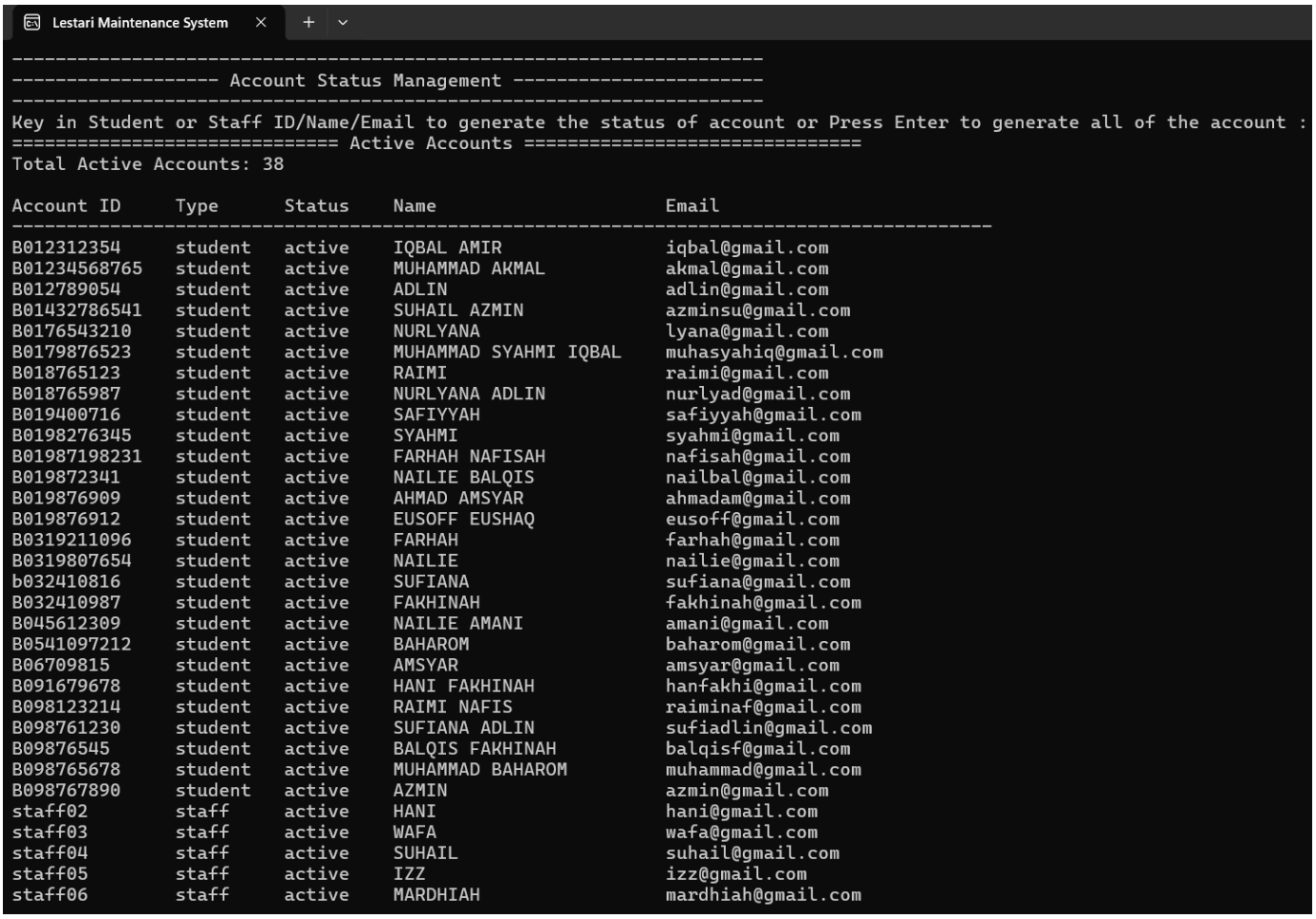


```
Lestari Maintenance System

Table: staff
================================================================================================================================
| staff_id        | password                          | name          | email               | phonenum       |
--------------------------------------------------------------------------------------------------------------------------------
| staff01         | a665a45920422f9d417e4867efd...    | 123           | aiesya@gmail.com    | 0162184436     |
| staff02         | 0ff69ab01fcb049e77ecc7f934d...    | HANI          | hani@gmail.com      | 01897054312    |
| staff03         | 3b299ba238a610e812c8c57b4b1...    | WAFA          | wafa@gmail.com      | 01709854365    |
| staff04         | 2052fe878a4099d3d063169667e...    | SUHAIL        | suhail@gmail.com    | 01923467891    |
| staff05         | cf1fb58104441488e7588166293...    | IZZ           | izz@gmail.com       | 01234156743    |
| staff06         | 54f77abc050592c12a3ceddbfef...    | MARDHIAH      | mardhiah@gmail.com  | 0154321876     |
| staff07         | 3b024d115de57e3adb6e098b673...    | ZAFF          | zaff@gmail.com      | 01876549217    |
| staff08         | 840ee9db4c08b9206c385a0ef1c...    | HAZE          | haze@gmail.com      | 0189876543     |
| staff09         | 2efc56b0a765df9354af135d5ce...    | BALQIS        | balqis@gmail.com    | 01216789234    |
| staff10         | 7f543835d473b6bdd1cb9a6e2f2...    | HAFIZ         | hafiz@gmail.com     | 0179076543     |
| staff11         | 7a73640b867a4010d5cb2655dfe...    | AINUL         | ainul@gmail.com     | 01140018526    |
| staff12         | 95b586fcee3f7413fd9fdbf80f0...    | AKMAL HAZIM   | hazim@gmail.com     | 01987654123    |
| staff13         | 4981444fafbc37d8bd27132e2ea...    | AISYAH        | aisyah@gmail.com    | 01213456541    |
| staff14         | 0ab2e6e613f5649e56113e5ea41...    | AMIRA FAZ     | amira@gmail.com     | 01987896781    |
================================================================================================================================

Press [E] to export to CSV or Enter to return to list: e
Exported to staff.csv
Press any key to continue . . .
```

18. Admin – Count Total Users (Admin, Staff and Student)

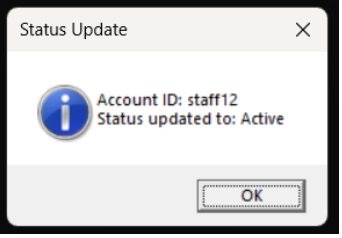


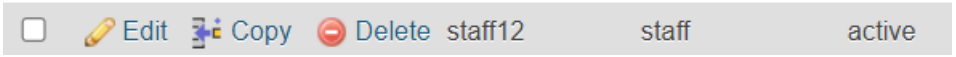
19. Admin – Account Status Management (Active / Inactive)



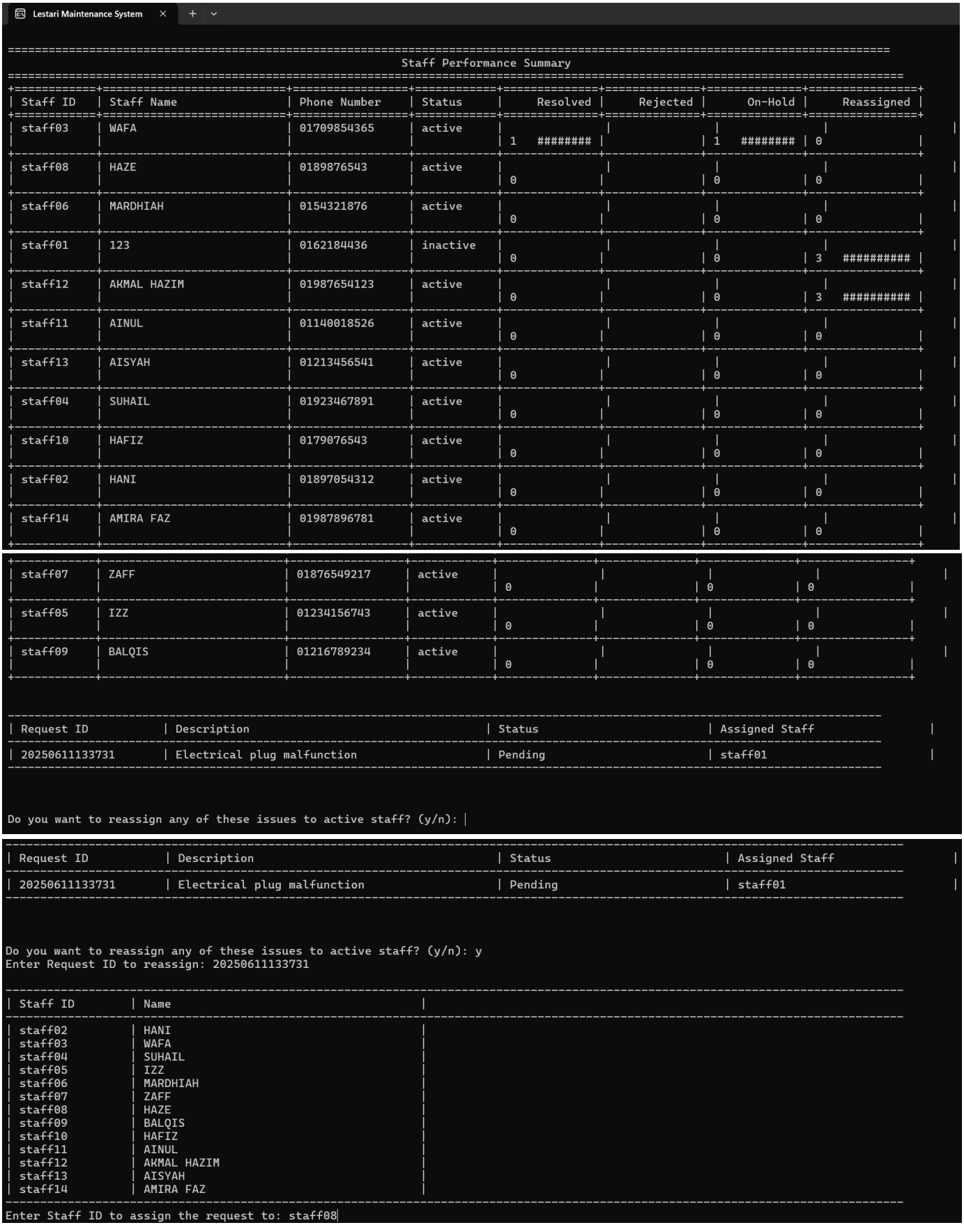
```
staff07         staff      active     ZAFF                    zaff@gmail.com
staff08         staff      active     HAZE                    haze@gmail.com
staff09         staff      active     BALQIS                  balqis@gmail.com
staff10         staff      active     HAFIZ                   hafiz@gmail.com
staff11         staff      active     AINUL                   ainul@gmail.com
staff13         staff      active     AISYAH                  aisyah@gmail.com
staff14         staff      active     AMIRA FAZ               amira@gmail.com

============================== Inactive Accounts ==============================
Total Inactive Accounts: 3

Account ID      Type       Status     Name                    Email
-------------------------------------------------------------------------------
B789456231      student    inactive   AIZAT HAMDAN            aizt@gmail.com
staff01         staff      inactive   123                     aiesya@gmail.com
staff12         staff      inactive   AKMAL HAZIM             hazim@gmail.com


Enter account ID to modify status (or leave empty to skip):
```

```
============================== Inactive Accounts ==============================
Total Inactive Accounts: 3

Account ID      Type       Status     Name                    Email
-------------------------------------------------------------------------------
B789456231      student    inactive   AIZAT HAMDAN            aizt@gmail.com
staff01         staff      inactive   123                     aiesya@gmail.com
staff12         staff      inactive   AKMAL HAZIM             hazim@gmail.com


Enter account ID to modify status (or leave empty to skip): staff12
--------------------------------------------------------------------------------
 1 for Active Account
 0 for Deactivate Account
--------------------------------------------------------------------------------
Set status (1=Active, 0=Deactivate): 1
Status updated successfully.
```





20. Admin – Staff Assigned Report


Lestari Maintenance System
Staff Performance Summary
Staff ID	Staff Name	Phone Number	Status	Resolved	Rejected	On-Hold	Reassigned
staff03	WAFA	01709854365	active	1 ########		1 ########	0
staff08	HAZE	0189876543	active	0		0	0
staff06	MARDHIAH	0154321876	active	0		0	0
staff01	123	0162184436	inactive	0		0	3 ##########
staff12	AKMAL HAZIM	01987654123	active	0		0	3 ##########
staff11	AINUL	01140018526	active	0		0	0
staff13	AISYAH	012134566541	active	0		0	0
staff04	SUHAIL	01923467891	active	0		0	0
staff10	HAFIZ	0179076543	active	0		0	0
staff02	HANI	01897054312	active	0		0	0
staff14	AMIRA FAZ	01987896781	active	0		0	0
staff07	ZAFF	01876549217	active	0		0	0
staff05	IZZ	01234156743	active	0		0	0
staff09	BALQIS	01216789234	active	0		0	0
Request ID	Description	Status	Assigned Staff
20250611133731	Electrical plug malfunction	Pending	staff01
Do you want to reassign any of these issues to active staff? (y/n):
Do you want to reassign any of these issues to active staff? (y/n): y
Enter Request ID to reassign: 20250611133731
Staff ID	Name
staff02	HANI
staff03	WAFA
staff04	SUHAIL
staff05	IZZ
staff06	MARDHIAH
staff07	ZAFF
staff08	HAZE
staff09	BALQIS
staff10	HAFIZ
staff11	AINUL
staff12	AKMAL HAZIM
staff13	AISYAH
staff14	AMIRA FAZ
Enter Staff ID to assign the request to: staff08

```
Enter Staff ID to assign the request to: staff08
Request 20250611133731 successfully reassigned to staff :  staff08.
Press any key to continue . . .
```

it	Copy	Delete	20250611133731	Electrical plug malfunction	Pending	Urgent	b01432786541	3	A1-3-B-01	2025-06-11 13:37:31	staff08	-	ADMIN1	1

21. Admin – Manage Student Penalties

Lestari Maintenance System

```
============================================================
                  PENALTY COLLECTION SUMMARY
============================================================
Total Penalties Collected: 3
Total Amount Collected: RM225.00

========================================================================
         STUDENTS WHO HAVE PAID THEIR PENALTIES SUCCESSFULLY
========================================================================

Student Name                    Matric No      Penalty Description          Amount         Date Issued             Handled By
-------------------------------------------------------------------------------------------------------------------------------
SUFIANA                         B032410816     Unauthorized Pets            RM75.00        2025-06-11 13:05:52     123
RAIMI                           B018765123     Smoking in Room              RM100.00       2025-06-11 13:05:21     WAFA
SUFIANA                         B032410816     Electric Cooker              RM50.00        2025-06-10 17:44:53     HANI

=============================================================================================================
                        DETAILED PENALTIES ASSIGNED TO STUDENTS (ALL STAFF)
=============================================================================================================

ID   Student Name        Matric No      Description             Amount    Date Issued             Paid      Confirmed   Staff
-------------------------------------------------------------------------------------------------------------------------------
74   SUFIANA             B032410816     Electric Cooker         RM50.00   2025-06-13 11:15:09     No        Pending     123
73   NAILIE BALQIS       B019872341     Electric Cooker         RM50.00   2025-06-13 11:03:46     No        Pending     123
72   SYAHMI              B0198276345    Noise Violation         RM25.00   2025-06-11 13:16:04     No        Pending     SUHAIL
71   NAILIE AMANI        B045612309     Smoking in Room         RM100.00  2025-06-11 13:15:43     No        Pending     WAFA
70   ADLIN               B012789054     Custom Penalty          RM40.00   2025-06-11 13:15:16     No        Pending     HANI
69   SYAHMI              B0198276345    Smoking in Room         RM100.00  2025-06-11 13:14:40     No        Pending     123
68   FARHAH NAFISAH      B01987198231   Noise Violation         RM25.00   2025-06-11 13:13:56     No        Pending     123
67   MUHAMMAD BAHAROM    B098765678     Unauthorized Pets       RM75.00   2025-06-11 13:08:22     No        Pending     WAFA
66   MUHAMMAD AKMAL      B01234568765   Smoking in Room         RM100.00  2025-06-11 13:08:12     No        Pending     HANI
65   NAILIE              B0319807654    Electric Cooker         RM50.00   2025-06-11 13:07:17     No        Pending     WAFA
64   NAILIE BALQIS       B019872341     Unauthorized Pets       RM75.00   2025-06-11 13:07:07     No        Pending     HANI
63   SUHAIL AZMIN        B01432786541   Electric Cooker         RM50.00   2025-06-11 13:06:27     No        Pending     HANI
62   NURLYANA ADLIN      B018765987     Electric Cooker         RM50.00   2025-06-11 13:06:04     No        Pending     WAFA
61   SUFIANA             B032410816     Unauthorized Pets       RM75.00   2025-06-11 13:05:52     Yes       Yes         123
60   RAIMI               B018765123     Smoking in Room         RM100.00  2025-06-11 13:05:21     Yes       Yes         WAFA
59   SUFIANA             B032410816     Electric Cooker         RM50.00   2025-06-10 17:44:53     Yes       Yes         HANI
```

```
No pending confirmations. All receipts are either confirmed or not yet paid.

============================================================
                  ACTIVE STUDENT ACCOUNTS
============================================================

-------------------------------------------------------------------------------------------------------------------------
 Matric Number           | Name                          | Email                    | Room Number
-------------------------------------------------------------------------------------------------------------------------
B012789054               | ADLIN                         | adlin@gmail.com          | B1-3-B-01
B019876909               | AHMAD AMSYAR                  | ahmadam@gmail.com        | A1-3-A-01
B06709815                | AMSYAR                        | amsyar@gmail.com         | A1-2-D-07
B098767890               | AZMIN                         | azmin@gmail.com          | A1-2-D-03
B0541097212              | BAHAROM                       | baharom@gmail.com        | A1-4-A-02
B09876545                | BALQIS FAKHINAH               | balqisf@gmail.com        | B1-D-G-01
B019876912               | EUSOFF EUSHAQ                 | eusoff@gmail.com         | A1-D-01-01
B032410987               | FAKHINAH                      | fakhinah@gmail.com       | B1-3-C-05
B0319211096              | FARHAH                        | farhah@gmail.com         | B1-2-C-01
B01987198231             | FARHAH NAFISAH                | nafisah@gmail.com        | B1-G-D-05
B091679678               | HANI FAKHINAH                 | hanfakhi@gmail.com       | B1-2-A-09
B012312354               | IQBAL AMIR                    | iqbal@gmail.com          | A1-3-B-01
B01234568765             | MUHAMMAD AKMAL                | akmal@gmail.com          | A1-B-2-09
B098765678               | MUHAMMAD BAHAROM              | muhammad@gmail.com       | A1-3-C-04
B0179876523              | MUHAMMAD SYAHMI IQBAL         | muhasyahiq@gmail.com     | A1-C-1-04
B0319807654              | NAILIE                        | nailie@gmail.com         | B1-3-D-05
B045612309               | NAILIE AMANI                  | amani@gmail.com          | B1-G-A-01
B019872341               | NAILIE BALQIS                 | nailbal@gmail.com        | B1-G-A-01
B0176543210              | NURLYANA                      | lyana@gmail.com          | B1-2-C-07
B018765987               | NURLYANA ADLIN                | nurlyad@gmail.com        | B1-2-C-07
B018765123               | RAIMI                         | raimi@gmail.com          | A1-1-B-03
B098123214               | RAIMI NAFIS                   | raiminaf@gmail.com       | A1-G-B-01
B019400716               | SAFIYYAH                      | safiyyah@gmail.com       | B1-3-C-06
B032410816               | SUFIANA                       | sufiana@gmail.com        | B1-1-C-04
B098761230               | SUFIANA ADLIN                 | sufiadlin@gmail.com      | B1-3-C-07
B01432786541             | SUHAIL AZMIN                  | azminsu@gmail.com        | A1-3-B-01
B0198276345              | SYAHMI                        | syahmi@gmail.com         | A1-4-A-02
```

```
============================================================
                    PENALTY MANAGEMENT
============================================================
1. Issue New Penalty
2. View Student Penalties
3. Update Penalty Status
4. Delete Penalty
5. Return to Admin Menu
6. Exit System
------------------------------------------------------------
Enter choice: 1

Enter Student Matric Number, Name, or Room Number to search: SUFIANA
------------------------------------------------------------

Matched Students:
------------------------------------------------------------
No.  | Matric Number | Name                          | Room
------------------------------------------------------------
1    | B032410816    | SUFIANA                       | B1-1-C-04
2    | B098761230    | SUFIANA ADLIN                 | B1-3-C-07

Enter the number of the student to select (0 to cancel):
```

```
Enter the number of the student to select (0 to cancel): 1

Selected Student: SUFIANA (Matric: B032410816)
=============================================================================================
Predefined Penalties:
=============================================================================================
1. Electric Cooker (RM50)
2. Extension Plug >3 (RM30)
3. Smoking in Room (RM100)
4. Unauthorized Pets (RM75)
5. Noise Violation (RM25)
6. Other (Custom Penalties)
-------------------------------------------------------
Select penalty type (1-6): 1
Penalty issued successfully!
Press any key to continue . . .
```

Edit Copy Delete 75 B032410816 1 Electric Cooker 50.00 2025-06-15 23:43:53 0 staff02 1

22. Admin – View Student Penalties

```
============================================================
                    PENALTY MANAGEMENT
============================================================
1. Issue New Penalty
2. View Student Penalties
3. Update Penalty Status
4. Delete Penalty
5. Return to Admin Menu
6. Exit System
------------------------------------------------------------
Enter choice: 2
-------------------------------------------------------------------------------------------------------------------
All Student Penalties:
===================================================================================================================
ID  | Matric Number | Description          | Amount   | Date Issued         | Paid? | Confirmed? | Staff ID
-------------------------------------------------------------------------------------------------------------------
75  | B032410816    | Electric Cooker      | RM50.00  | 2025-06-15 23:43:53 | No    | No         | staff02
74  | B032410816    | Electric Cooker      | RM50.00  | 2025-06-13 11:15:09 | No    | No         | staff01
73  | B019872341    | Electric Cooker      | RM50.00  | 2025-06-13 11:03:46 | No    | No         | staff01
72  | B0198276345   | Noise Violation      | RM25.00  | 2025-06-11 13:16:04 | No    | No         | staff04
71  | B045612309    | Smoking in Room      | RM100.00 | 2025-06-11 13:15:43 | No    | No         | staff03
70  | B012789054    | Custom Penalty       | RM40.00  | 2025-06-11 13:15:16 | No    | No         | staff02
69  | B0198276345   | Smoking in Room      | RM100.00 | 2025-06-11 13:14:40 | No    | No         | staff01
68  | B01987198231  | Noise Violation      | RM25.00  | 2025-06-11 13:13:56 | No    | No         | staff01
67  | B098765678    | Unauthorized Pets    | RM75.00  | 2025-06-11 13:08:22 | No    | No         | staff03
66  | B01234568765  | Smoking in Room      | RM100.00 | 2025-06-11 13:08:12 | No    | No         | staff02
65  | B0319807654   | Electric Cooker      | RM50.00  | 2025-06-11 13:07:17 | No    | No         | staff03
64  | B019872341    | Unauthorized Pets    | RM75.00  | 2025-06-11 13:07:07 | No    | No         | staff02
63  | B01432786541  | Electric Cooker      | RM50.00  | 2025-06-11 13:06:27 | No    | No         | staff02
62  | B018765987    | Electric Cooker      | RM50.00  | 2025-06-11 13:06:04 | No    | No         | staff03
61  | B032410816    | Unauthorized Pets    | RM75.00  | 2025-06-11 13:05:52 | Yes   | Yes        | staff01
60  | B018765123    | Smoking in Room      | RM100.00 | 2025-06-11 13:05:21 | Yes   | Yes        | staff03
59  | B032410816    | Electric Cooker      | RM50.00  | 2025-06-10 17:44:53 | Yes   | Yes        | staff02

Enter Student Matric Number to view specific penalties (or press Enter to skip):
```

```
Enter Student Matric Number to view specific penalties (or press Enter to skip): B032410816

-------------------------------------------------------------------------------------------------------------------
Penalty History for Matric Number: B032410816
===================================================================================================================
ID  | Description          | Amount   | Date Issued         | Paid? | Confirmed? | Staff ID
-------------------------------------------------------------------------------------------------------------------
75  | Electric Cooker      | RM50.00  | 2025-06-15 23:43:53 | No    | No         | staff02
74  | Electric Cooker      | RM50.00  | 2025-06-13 11:15:09 | No    | No         | staff01
61  | Unauthorized Pets    | RM75.00  | 2025-06-11 13:05:52 | Yes   | Yes        | staff01
59  | Electric Cooker      | RM50.00  | 2025-06-10 17:44:53 | Yes   | Yes        | staff02
Press any key to continue . . .
```

23. Admin – Update Penalty Status

```
============================================================
                    PENALTY MANAGEMENT
============================================================
1. Issue New Penalty
2. View Student Penalties
3. Update Penalty Status
4. Delete Penalty
5. Return to Admin Menu
6. Exit System
------------------------------------------------------------
Enter choice: 3
Enter Penalty ID to mark as paid: 75
Penalty marked as paid!
Press any key to continue . . .
```

☐	Edit Copy Delete	75	B032410816	1	Electric Cooker	50.00	2025-06-15 23:43:53	1	staff02	1

24. Admin – Delete Penalty

```
============================================================
                    PENALTY MANAGEMENT
============================================================
1. Issue New Penalty
2. View Student Penalties
3. Update Penalty Status
4. Delete Penalty
5. Return to Admin Menu
6. Exit System
------------------------------------------------------------
Enter choice: 4
Enter Penalty ID to delete: 75
Are you sure you want to delete penalty 75? (y/n): y
Penalty deleted successfully!
Press any key to continue . . .
```

☐	Edit Copy Delete	61	B032410816	4	Unauthorized Pets	75.00	2025-06-11 13:05:52	1	staff01	0
☐	Edit Copy Delete	62	B018765987	1	Electric Cooker	50.00	2025-06-11 13:06:04	0	staff03	1
☐	Edit Copy Delete	63	B01432786541	1	Electric Cooker	50.00	2025-06-11 13:06:27	0	staff02	1
☐	Edit Copy Delete	64	B019872341	4	Unauthorized Pets	75.00	2025-06-11 13:07:07	0	staff02	1
☐	Edit Copy Delete	65	B0319807654	1	Electric Cooker	50.00	2025-06-11 13:07:17	0	staff03	1
☐	Edit Copy Delete	66	B01234568765	3	Smoking in Room	100.00	2025-06-11 13:08:12	0	staff02	1
☐	Edit Copy Delete	67	B098765678	4	Unauthorized Pets	75.00	2025-06-11 13:08:22	0	staff03	1
☐	Edit Copy Delete	68	B01987198231	5	Noise Violation	25.00	2025-06-11 13:13:56	0	staff01	1
☐	Edit Copy Delete	69	B0198276345	3	Smoking in Room	100.00	2025-06-11 13:14:40	0	staff01	1
☐	Edit Copy Delete	70	B012789054	6	Draw Wall	40.00	2025-06-11 13:15:16	0	staff02	1
☐	Edit Copy Delete	71	B045612309	3	Smoking in Room	100.00	2025-06-11 13:15:43	0	staff03	1
☐	Edit Copy Delete	72	B0198276345	5	Noise Violation	25.00	2025-06-11 13:16:04	0	staff04	1
☐	Edit Copy Delete	73	B019872341	1	Electric Cooker	50.00	2025-06-13 11:03:46	0	staff01	1
☐	Edit Copy Delete	74	B032410816	1	Electric Cooker	50.00	2025-06-13 11:15:09	0	staff01	1

Check all *With selected:* Edit Copy Delete Export

25. Admin – Summarize Penalties Per Staff

Lestari Maintenance System

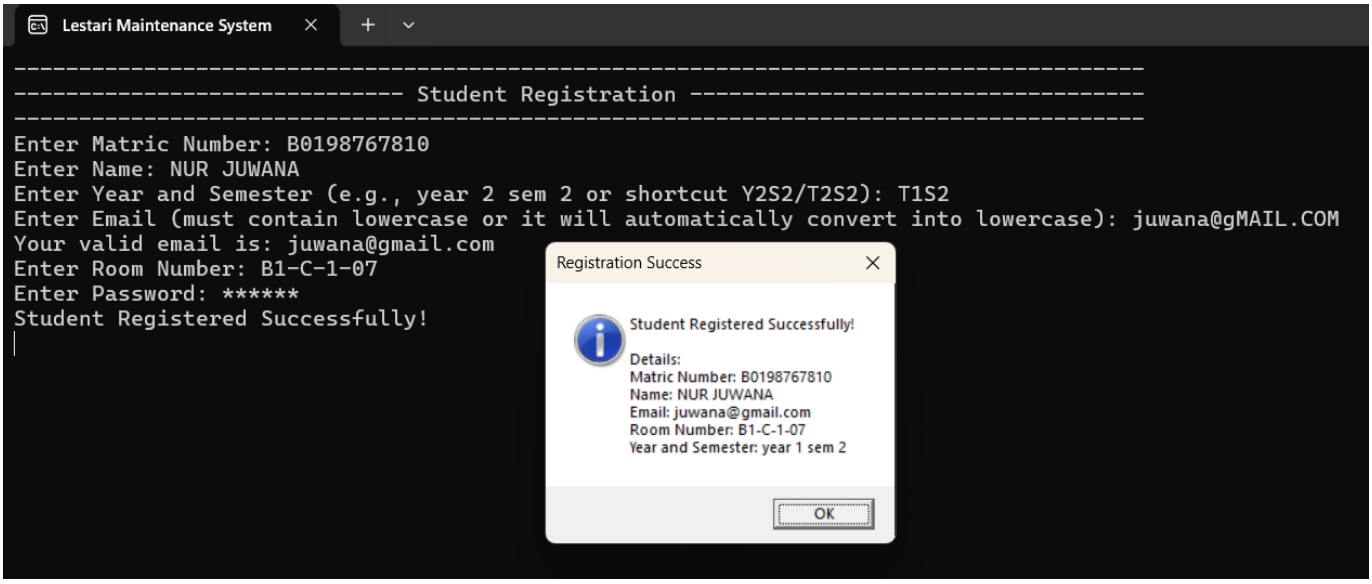
```
==================================================
     SUMMARY: PENALTIES ASSIGNED TO EACH STAFF
==================================================
Staff ID        Staff Name                      Total Penalties
------------------------------------------------------------
staff01         123                             5
staff03         WAFA                            5
staff02         HANI                            5
staff04         SUHAIL                          1

============================================================

============================================================
         FILTER PENALTIES ASSIGNED ON A SPECIFIC DATE
============================================================
Enter a date to filter (format YYYY-MM-DD)
Or enter 0 to return to the menu.
------------------------------------------------------------
Your input: 2025-06-15
No penalty records found for 2025-06-15.
```

```
======================================================================
          DETAILED PENALTIES ASSIGNED TO STUDENTS (ALL STAFF)
======================================================================
ID   Student Name        Matric No      Description             Amount    Date Issued    Paid    Confirmed   Staff
-------------------------------------------------------------------------------------------------------------------
74   SUFIANA             B032410816     Electric Cooker         RM50.00   2025-06-13 11:15:09No      Pending     123
73   NAILIE BALQIS       B019872341     Electric Cooker         RM50.00   2025-06-13 11:03:46No      Pending     123
72   SYAHMI              B0198276345    Noise Violation         RM25.00   2025-06-11 13:16:04No      Pending     SUHAIL
71   NAILIE AMANI        B045612309     Smoking in Room         RM100.00  2025-06-11 13:15:43No      Pending     WAFA
70   ADLIN               B012789054     Custom Penalty          RM40.00   2025-06-11 13:15:16No      Pending     HANI
69   SYAHMI              B0198276345    Smoking in Room         RM100.00  2025-06-11 13:14:40No      Pending     123
68   FARHAH NAFISAH      B01987198231   Noise Violation         RM25.00   2025-06-11 13:13:56No      Pending     123
67   MUHAMMAD BAHAROM    B098765678     Unauthorized Pets       RM75.00   2025-06-11 13:08:22No      Pending     WAFA
66   MUHAMMAD AKMAL      B01234568765   Smoking in Room         RM100.00  2025-06-11 13:08:12No      Pending     HANI
65   NAILIE              B0319807654    Electric Cooker         RM50.00   2025-06-11 13:07:17No      Pending     WAFA
64   NAILIE BALQIS       B019872341     Unauthorized Pets       RM75.00   2025-06-11 13:07:07No      Pending     HANI
63   SUHAIL AZMIN        B01432786541   Electric Cooker         RM50.00   2025-06-11 13:06:27No      Pending     HANI
62   NURLYANA ADLIN      B018765987     Electric Cooker         RM50.00   2025-06-11 13:06:04No      Pending     WAFA
61   SUFIANA             B032410816     Unauthorized Pets       RM75.00   2025-06-11 13:05:52Yes     Yes         123
60   RAIMI               B018765123     Smoking in Room         RM100.00  2025-06-11 13:05:21Yes     Yes         WAFA
59   SUFIANA             B032410816     Electric Cooker         RM50.00   2025-06-10 17:44:53Yes     Yes         HANI
```

```
No pending confirmations. All receipts are either confirmed or not yet paid.
============================================================
                  ACTIVE STUDENT ACCOUNTS
============================================================
------------------------------------------------------------------------------
Matric Number | Name                | Email                      | Room Number
------------------------------------------------------------------------------
B012789054    | ADLIN               | adlin@gmail.com            | B1-3-B-01
B019876909    | AHMAD AMSYAR        | ahmadam@gmail.com          | A1-3-A-01
B06709815     | AMSYAR              | amsyar@gmail.com           | A1-2-D-07
B098767890    | AZMIN               | azmin@gmail.com            | A1-2-D-03
B0541097212   | BAHAROM             | baharom@gmail.com          | A1-4-A-02
B09876545     | BALQIS FAKHINAH     | balqisf@gmail.com          | B1-D-G-01
B019876912    | EUSOFF EUSHAQ       | eusoff@gmail.com           | A1-D-01-01
B032410987    | FAKHINAH            | fakhinah@gmail.com         | B1-3-C-05
B0319211096   | FARHAH              | farhah@gmail.com           | B1-2-C-01
B01987198231  | FARHAH NAFISAH      | nafisah@gmail.com          | B1-G-D-05
B091679678    | HANI FAKHINAH       | hanfakhi@gmail.com         | B1-2-A-09
B012312354    | IQBAL AMIR          | iqbal@gmail.com            | A1-3-B-01
B01234568765  | MUHAMMAD AKMAL      | akmal@gmail.com            | A1-B-2-09
B098765678    | MUHAMMAD BAHAROM    | muhammad@gmail.com         | A1-3-C-04
B0179876523   | MUHAMMAD SYAHMI IQBAL | muhasyahiq@gmail.com     | A1-C-1-04
B0319807654   | NAILIE              | nailie@gmail.com           | B1-3-D-05
B045612309    | NAILIE AMANI        | amani@gmail.com            | B1-G-A-01
B019872341    | NAILIE BALQIS       | nailbal@gmail.com          | B1-G-A-01
B0176543210   | NURLYANA            | lyana@gmail.com            | B1-2-C-07
B018765987    | NURLYANA ADLIN      | nurlyad@gmail.com          | B1-2-C-07
B018765123    | RAIMI               | raimi@gmail.com            | A1-1-B-03
B098123214    | RAIMI NAFIS         | raiminaf@gmail.com         | A1-G-B-01
B019400716    | SAFIYYAH            | safiyyah@gmail.com         | B1-3-C-06
B032410816    | SUFIANA             | sufiana@gmail.com          | B1-1-C-04
B098761230    | SUFIANA ADLIN       | sufiadlin@gmail.com        | B1-3-C-07
B01432786541  | SUHAIL AZMIN        | azminsu@gmail.com          | A1-3-B-01
B0198276345   | SYAHMI              | syahmi@gmail.com           | A1-4-A-02
```

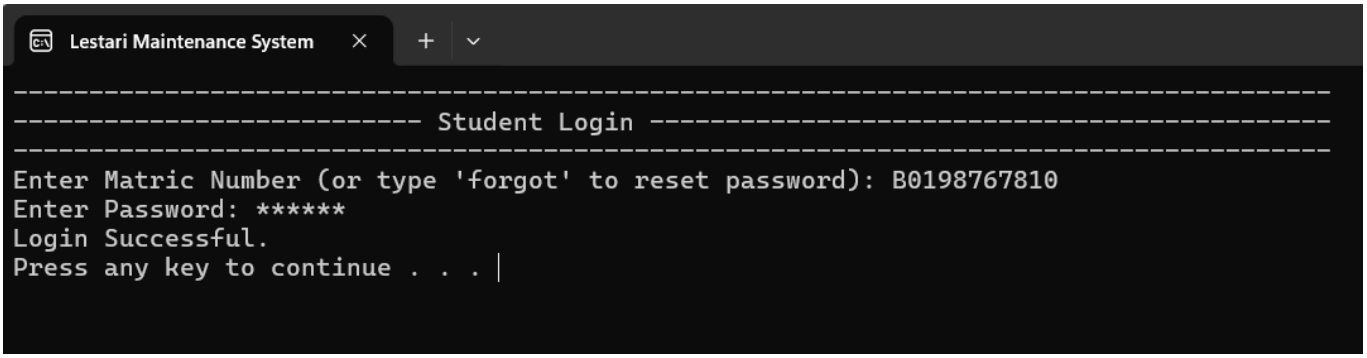
```
============================================================
                PENALTY COLLECTION SUMMARY
============================================================
Total Penalties Collected: 3
Total Amount Collected: RM225.00

============================================================
                   PENALTY MANAGEMENT
============================================================
1. Issue New Penalty
2. View Student Penalties
3. Update Penalty Status
4. Delete Penalty
5. Return to Admin Menu
6. Exit System
------------------------------------------------------------
Enter choice:
```

26. Student Registration

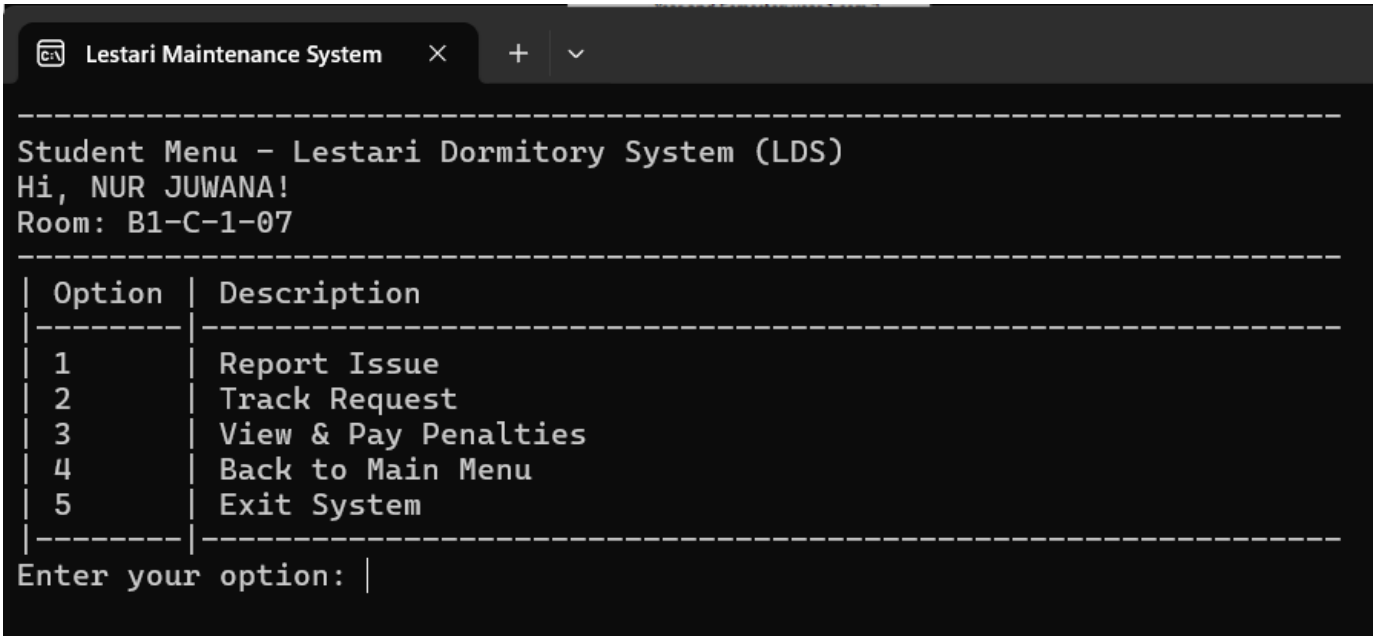


Edit Copy Delete B0198767810 NUR JUWANA juwana@gmail.com f9678de32dbc80b8318ada0b2b056c1a2638ee10d4ab44a56e... B1-C-1-07 year 1 sem 2

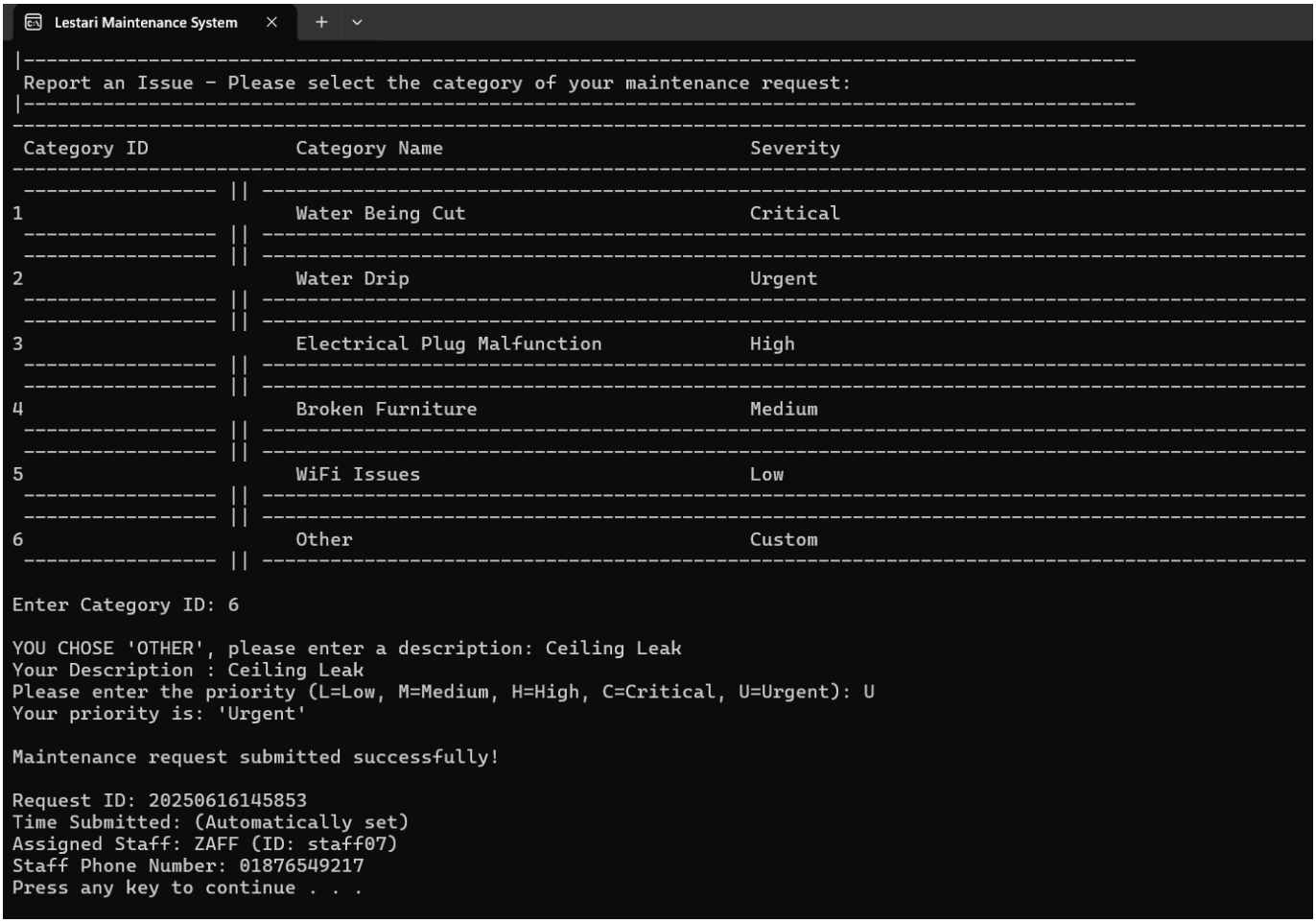
27. Student Login



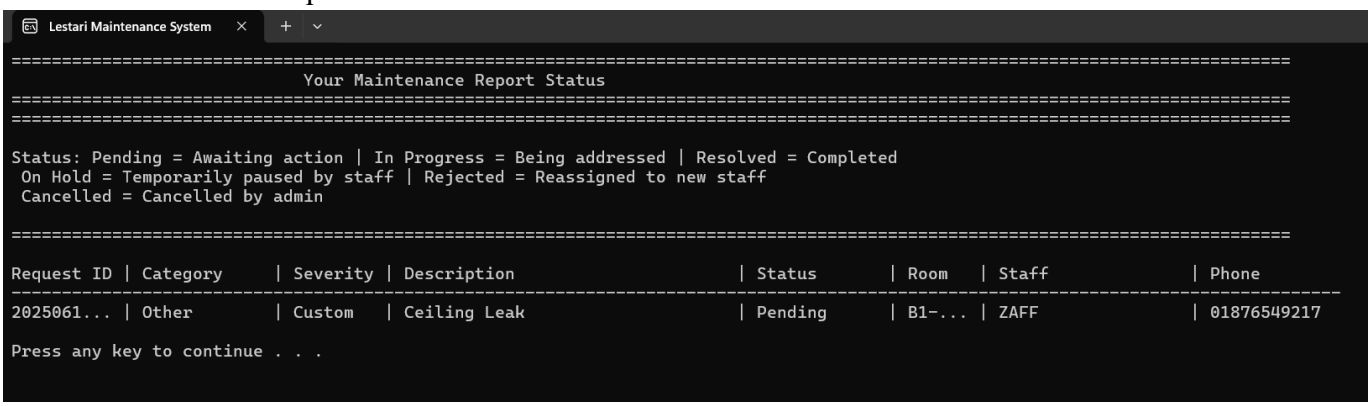
28. Student Menu



28. Student – Report Issue

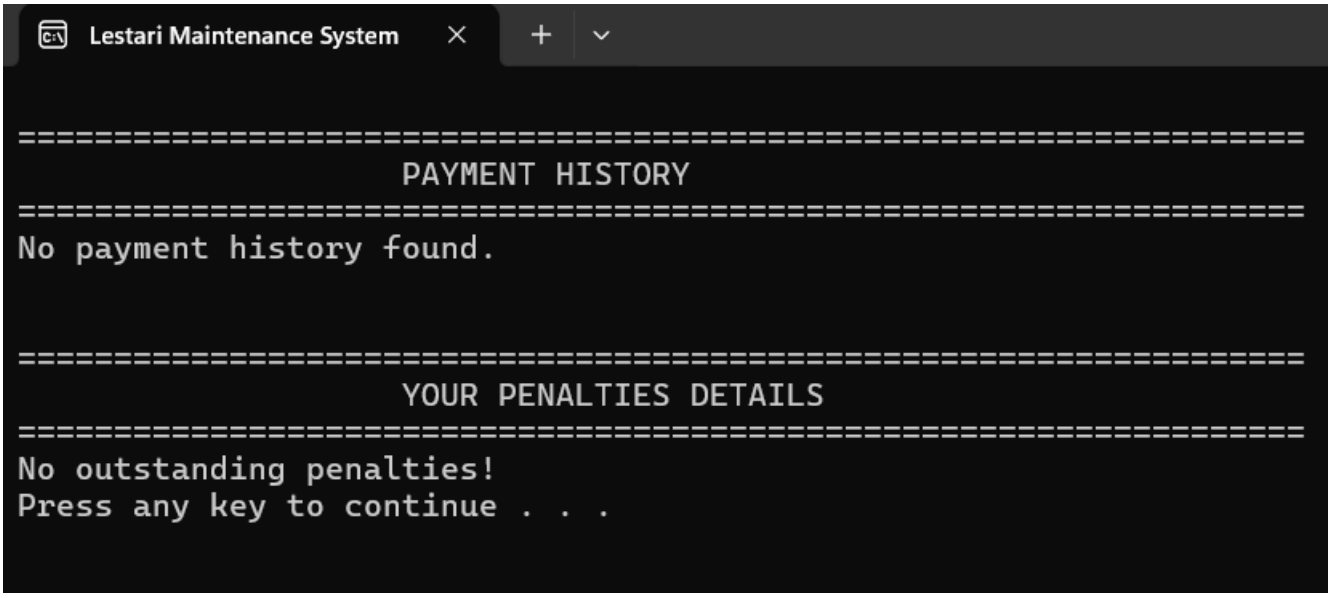


29. Student – Track Request



20250616145853	Ceiling Leak	Pending	Urgent	b0198767810	6	B1-C-1-07	2025-06-16 14:58:53	staff07	-	NULL

30. Student – View Penalties



CHAPTER 4 : SYSTEM IMPLEMENTATION

4.1 Naming Conversion

Naming conventions are essential in programming because they provide a structured and consistent way to name variables, functions, and classes. Adhering to naming conventions makes the code more readable and maintainable. In this system, I have followed a consistent naming convention for all variables and functions, which ensures clarity and ease of understanding.

Here are some of the rules I applied:

a) **Variable Names**: I used descriptive names for variables, starting with a lowercase letter and following camelCase style. For example, maintenanceRecord, lastUpdated, and userInput.

b) **Function Names**: For function names, I followed a convention where the name clearly describes its purpose, using verbs in lowercase. For example, updateMaintenance(), addNewRecord(), and deleteRecord().

c) **Class Names**: For class names, I used PascalCase style, capitalizing the first letter of each word. For example, MaintenanceSystem, UserManager.

4.2. Function

Functions are integral to breaking down complex tasks into manageable chunks of code. In the implementation of the Lestari Maintenance System (LMS), I created multiple functions to handle specific tasks, such as updating records, adding new entries, or managing the user interface.

For instance, one important function is addMaintenanceRecord(), which is responsible for inserting new maintenance records into the system. This function receives parameters such as the maintenance type, date, and assigned technician, and processes them to store the data in the appropriate data structure.

Here is an example of a function from the code:

```
// ======================
```

```
// MAIN MENU
// ======================
void displayMainMenu() {
int option; while (true) {
system("cls"); // Clear screen
cout << "------------------------------------------------------------------------
-\n";
cout << "B032410816 ---------------- Lestari Maintenance System ----------
(LMS)\n";
cout << "------------------------------------------------------------------------
-\n";
cout << "               Welcome to Lestari Maintenance System (LMS)\n"; cout <<
"-------------------------------------------------------------------
-\n";
cout << "| 1 | Admin Registration\n";
cout << "| 2 | Admin Login\n"; cout <<
"| 3 | Student Registration\n"; cout <<
"| 4 | Student Login\n"; cout << "| 5 |
Exit\n";
cout << "------------------------------------------------------------------------
-\n";
cout << "Enter your choice: "; if (!(cin
>> option)) { // Validate input
cin.clear();
cin.ignore(numeric_limits<streamsize>::max(), '\n');
cout << "Invalid input! Please enter a number.\n";
system("pause"); continue; }
cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear input buffer

// Handle menu selection switch
(option) {
case 1: adminRegister(); break; case 2:
adminLogin(); break; case 3: studentRegister();
break; case 4: studentLogin(); break; case 5:
exitSystem(); break; default: cout << "Invalid
option! Please try again.\n"; continue;  //
explicitly restart the loop system("pause");
// Loop will continue and redisplay the menu automatically continue;
// explicitly restart the loop
}
}
}
```

4.3. Selection

The selection structure controls the flow of the program based on conditions. It is used to make decisions by evaluating whether a certain condition is true or false.

For example, I used an if statement in the code to check whether a maintenance record exists before allowing the deletion of that record:

```
        // Check credentials
        string query = "SELECT name FROM admin WHERE staff_id='" + id + "' AND password='" + pw
    + "'";
        qstate = mysql_query(conn, query.c_str());
    if (!qstate) {
            res = mysql_store_result(conn);
    if (mysql_num_rows(res) > 0) {
    row = mysql_fetch_row(res);
    currentStaff = id;
    currentAdminName = row[0];              cout
    << "Login Successful.\n";
    mysql_free_result(res);
    displayAdminMenu();
            }
    else {
                cout << "Invalid Credentials!\n";
    mysql_free_result(res);
            }
    }     else
    {
            cout << "Database error.\n";
        }
        system("pause");

    }
```

This decision structure ensures that the program behaves as expected and prevents errors such as deleting non-existing records.

Additionally, the switch statement was used to handle different user options in the menu system, making the code more organized and easy to extend in the future.

```
        switch (choice) {
        case 1: manageAccounts(); break;
        case 2: viewReports(); system("pause"); break;
case 3: generateReports(); system("pause"); break;
        case 4: return;
case 5: {
system("cls");
            cout << "--------------------------------------------------------------------------\n";
cout << "--------------------------- Account Summary ----------------------\n";
cout << "--------------------------------------------------------------------\n";
string countQuery = "SELECT COUNT(*) FROM account";             qstate = mysql_query(conn,
countQuery.c_str());              if (!qstate) {
                res = mysql_store_result(conn);
row = mysql_fetch_row(res);
                cout << "Total number of registered users: " << row[0] << endl;
                mysql_free_result(res);
            }
else {
                cout << "Failed to retrieve total user count.\n";
            }
            string adminCountQuery = "SELECT COUNT(*) FROM account WHERE account_type = 'admin'";
qstate = mysql_query(conn, adminCountQuery.c_str());              if (!qstate) {
                res = mysql_store_result(conn);
row = mysql_fetch_row(res);
```

```
                    cout << "Number of admin accounts: " << row[0] << endl;
                    mysql_free_result(res);
                }
    else {
                    cout << "Failed to retrieve admin account count.\n";
                }
                string studentCountQuery = "SELECT COUNT(*) FROM account WHERE account_type =
    'student'";
                qstate = mysql_query(conn, studentCountQuery.c_str());
    if (!qstate) {
                    res = mysql_store_result(conn);
    row = mysql_fetch_row(res);
                    cout << "Number of student accounts: " << row[0] << endl;
    mysql_free_result(res);
                }
    else {
                    cout << "Failed to retrieve student account count.\n";
                }
                system("pause"); break;
            }
            case 6: exitSystem(); break;
    default:            exitSystem();
    system("pause");
                continue;  //explicity restart the loop
    }
```

4.4. Control

Control structures are used to manage the flow of execution in the program. Apart from selection (if, switch), iteration is another key aspect of control. In this system, I used for loops and while loops to iterate over collections of maintenance records.

For example, to display all maintenance records, I used a for loop to iterate over the list of records:

```
for (int i = 0; i < num_fields; i++) {
                cout << " " << left << setw(colWidths[i]) << fields[i].name << " |";
            }
```

4.5. Pointer

Pointers are variables that store memory addresses. In this system, I used pointers to manage dynamic memory allocation, especially when creating and deleting records in the maintenance system. Using pointers allowed the system to handle memory efficiently and avoid unnecessary copying of data.
For example, the following snippet uses a pointer to dynamically allocate memory for a new record:

```
// Return description for category 1-5, else empty string
string getDescriptionFromCategory(const string& category_id) {
if (category_id == "1") return "Water being cut";     else if
(category_id == "2") return "Water drip";
```

```
            else if (category_id == "3") return "Electrical plug malfunction";
        else if (category_id == "4") return "Broken furniture";     else if
        (category_id == "5") return "WiFi Issues";
            else return "";  // For '6' or others, return empty to prompt user input

                    }


        // Automatically assign description and priority based on category_id
        description = getDescriptionFromCategory(category_id);
priority = getPriorityFromCategory(category_id);

        if (description.empty() && priority.empty()) {
            // For '6' or others, prompt user for custom input description
cout << "YOU CHOOSE TO SELECT 'OTHER', Please enter a description: ";
cin >> description;
            getline(cin >> ws, description);

            // For '6' or others, prompt user for custom input priority
            cout << "Please enter the priority (e.g., Low, Medium, High, Critical, Urgent) : ";
cin >> priority;
            cout << "Your priority is: '" << priority << "'" << endl;
if (priority.empty()) {
                cout << "Warning: Priority is empty! Admin assignment may fail.\n";
}

        }
else {
            cout << "Description for this category: " << description << endl;
cout << "Priority for this category: " << priority << endl;

        }

        // Output the final values
        cout << "\nFinal Description: " << description << endl;

        cout << "Final Priority: " << priority << endl;
```

4.6. Error Handling

Error handling is essential in ensuring that the system can gracefully handle unexpected events or incorrect user input. In this system, I implemented basic error handling mechanisms to prevent the program from crashing when invalid operations are performed.

For instance, before attempting to delete a maintenance record, I check if the record exists:

```
        // ========================
        // STUDENT MENU (REPORT ISSUE + TRACK REQUEST)
        // ========================

        // Return description for category 1-5, else empty string
        string getDescriptionFromCategory(const string& category_id) {
        if (category_id == "1") return "Water being cut";     else if
        (category_id == "2") return "Water drip";
```

```
    else if (category_id == "3") return "Electrical plug malfunction";
else if (category_id == "4") return "Broken furniture";     else if
(category_id == "5") return "WiFi Issues";
    else return "";  // For '6' or others, return empty to prompt user input
}

// Return priority for category 1-5, else empty string
string getPriorityFromCategory(const string& category_id) {
if (category_id == "1") return "Critical";     else if
(category_id == "2") return "High";     else if (category_id
== "3") return "Urgent";     else if (category_id == "4")
return "Medium";     else if (category_id == "5") return
"Low";
    else return "";  // For '6' or others, return empty to prompt user input
}
void displayStudentMenu() {
int choice;     while (true)
{         system("cls");
        cout << "-------------------------------------------------------------------------
\n";
        cout << "Student Menu - Lestari Maintenance System (LMS)\n";
cout << "Hi, " << currentStudentName << "!\n";         cout << "Room:
" << currentStudentRoom << endl;
        cout << "-------------------------------------------------------------------------
\n";
        cout << "| Option | Description\n";
        cout << "|--------|----------------------------------------------------------------
\n";
        cout << "| 1      | Report Issue\n";
cout << "| 2      | Track Request\n";         cout
<< "| 3      | Back to Main Menu\n";         cout
<< "| 4      | Exit\n";
        cout << "|--------|----------------------------------------------------------------
\n";
        cout << "Enter your option: ";
cin >> choice;         cin.ignore();

        switch (choice) {
            // ========================
            // 1. REPORT ISSUE
// ========================         case
1: {

            system("cls");
            cout << "|-------------------------------------------------------------------
--------------------------\n";
            cout << " Report an Issue - Please select the category of your maintenance
request:\n";
            cout << "|-------------------------------------------------------------------
-------------------------\n";

            // Show all categories
            string cat_query = "SELECT category_id, category_name, severity_level FROM
category";
            qstate = mysql_query(conn, cat_query.c_str());
if (!qstate) {
                res = mysql_store_result(conn);
                cout << "|-----------------------------------------------------------------
--------------------------\n";
```

```
                cout << left << setw(25) << "Category ID" << setw(40) << "Category Name" <<
setw(30) << "Severity" << endl;
                cout << "|-------------------------------------------------------------------
--------------------------\n";
                while ((row = mysql_fetch_row(res))) {
                    cout << "|-------------------------------------------------------------------
--------------------------\n";
                    cout << left << setw(25) << row[0] << setw(40) << row[1] << setw(30) <<
row[2] << endl;
                    cout << "|-------------------------------------------------------------------
--------------------------\n";
                }
                mysql_free_result(res);
            }
else {
                cout << "Failed to retrieve categories.\n";
system("pause"); break;
            }

            // Gather issue details
            string category_id, description, priority;
cout << "Enter Category ID: ";             getline(cin,
category_id);



            // Automatically assign description and priority based on category_id
description = getDescriptionFromCategory(category_id);             priority =
getPriorityFromCategory(category_id);

            if (description.empty() && priority.empty()) {
                // For '6' or others, prompt user for custom input description
cout << "YOU CHOOSE TO SELECT 'OTHER', Please enter a description: ";
cin >> description;
                getline(cin >> ws, description);

                // For '6' or others, prompt user for custom input priority
                cout << "Please enter the priority (e.g., Low, Medium, High, Critical, Urgent)
: ";
                cin >> priority;
                cout << "Your priority is: '" << priority << "'" << endl;
                if (priority.empty()) {
                    cout << "Warning: Priority is empty! Admin assignment may fail.\n";
}

            }
else {
                cout << "Description for this category: " << description << endl;
cout << "Priority for this category: " << priority << endl;
            }

            // Output the final values
            cout << "\nFinal Description: " << description << endl;
cout << "Final Priority: " << priority << endl;
```

4.7. Calculation

Calculation is a crucial part of this system to ensure accurate financial reporting and transparency. In this system, I implemented a calculation mechanism to summarize the total number of penalties collected and the total amount of money received from paid penalties. For instance, when displaying the penalty collection summary, the system executes a SQL query that counts all paid penalties and sums their corresponding amounts. The use of aggregate functions like COUNT and SUM ensures that the data is accurately calculated directly from the database, minimizing the risk of manual errors. Additionally, the system handles cases where there might be no penalties by using default values, ensuring that the summary always displays meaningful information. This approach provides users with clear and reliable financial figures, supporting better decision-making and record-keeping.

```
//============================================================================================================
=
            // PENALTY COLLECTION SUMMARY (TOTAL AMOUNT)
            //
============================================================================================
            // Step 1.5: Display total penalties and total amount collected
            std::cout << "\n";
            std::cout << "=================================================================\n";
            std::cout << "                    PENALTY COLLECTION SUMMARY                   \n";
            std::cout << "=================================================================\n";

            std::string summaryQuery =
                "SELECT COUNT(*) AS total_penalties, "
                "IFNULL(SUM(amount), 0) AS total_amount_collected "
                "FROM student_penalties "
                "WHERE is_paid = TRUE";

            qstate = mysql_query(conn, summaryQuery.c_str());
            if (qstate == 0) {
                res = mysql_store_result(conn);
                if (res && mysql_num_rows(res) > 0) {
                    MYSQL_ROW row = mysql_fetch_row(res);
                    std::string totalPenalties = row[0] ? row[0] : "0";
                    std::string totalAmount = row[1] ? row[1] : "0.00";

                    std::cout << "Total Penalties Collected: " << totalPenalties << "\n";
                    std::cout << "Total Amount Collected: RM" << totalAmount << "\n";
                }
                else {
                    std::cout << "No penalty collection data found.\n";
                }
                mysql_free_result(res);
            }
            else {
                std::cout << "Failed to retrieve penalty summary: " << mysql_error(conn) << "\n";
            }
```

CHAPTER 5: CONCLUSION

This chapter discusses the constraints encountered during the development of the Lestari Dormitory System (LDS) and outlines future improvements to enhance its effectiveness and user experience. The LDS was created to transform maintenance reporting and administration in dorms for students. The system introduces a consolidated platform to solve the inefficiencies and difficulties associated with previous manual operations. By enabling students to report maintenance concerns securely, LDS guarantees a secure and convenient experience. After being submitted, these requests are categorized and prioritized by management, which makes the response process more structured and effective.

5.1 Constraints

During the development of the Lestari Maintenance System (LMS), several constraints were identified:

a) Technical Limitations:

The system's real-time tracking feature faced initial challenges in syncing updates across multiple user interfaces, requiring iterative testing to ensure consistency. Integration with existing university databases (e.g., student number records) posed compatibility issues, necessitating custom API development.

b) User Adoption:

Transitioning users from manual processes (e.g., WhatsApp/Google Forms) to LMS required phased training to mitigate resistance to change.

c) Time Constraints:

Limited development time resulted in prioritizing core functionalities (e.g., issue reporting, tracking) over advanced features like predictive analytics for maintenance trends.

5.2 Future Improvements

To enhance LMS further, the following improvements are proposed:

a) Mobile Application Integration:

Develop a dedicated mobile app to allow students and administrators to submit/view requests, receive push notifications, and track progress on-the-go.

b) Automated Notifications:

Implement SMS/email alerts for status updates (e.g., when a request is resolved), reducing reliance on manual communication.

c) Advanced Analytics Dashboard:

Introduce data visualization tools to help administrators identify recurring issues, allocate resources efficiently, and predict maintenance needs.

d) Feedback Mechanism:

Add a post-resolution feedback feature for students to rate service quality, enabling continuous improvement.

e) Multi-Campus Scalability:

Expand LMS to support other campuses by modularizing the system architecture and incorporating location-based categorization.

CHAPTER 6 : REFERENCES

1. JK Bengkel 1, “Buku Panduan Bengkel 1 BITU 2913”, FTMK, 2025.
2. McLaughlin, M., “Oracle Database 11g & MySQL 5.6 Developer Handbook”, McGraw Hill, 2011.
3. AL-KHAYYAT, A. T. K., “WEB-BASED MANAGEMENT SYSTEM FOR THE DORMITORY OF UNIVERSITY STUDENTS”, ResearchGate, 2023.