HARARE INSTITUTE OF TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

BTECH (HONS) INFORMATION TECHNOLOGY

DIGITAL STUDENT REVIEW AND RELATIONSHIP MANAGEMENT SYSTEM

BY

**LONAH MUREGWI H230295F**

**EMMANUEL L I CHINJEKURE H230393F**

**PANATSWA MUJEGU H230037W**

**TINEVIMBO CHIZEMA H230022C**

HIT 200

SUPERVISOR: Mr. Sumbureru

**This project report was submitted to Harare Institute of technology in partial fulfilment of the bachelor of technology (Hons) degree in information technology**

**DECLARATION**

We, Emmanuel L I Chinjekure, Lonah P Muregwi, Panatswa Mujegu, Tinevimbo Chizema, hereby declare that the work contained in this project titled **"Digital Student Review and Relationship Management System"** is my original work, except where otherwise stated, and has not been submitted for any academic award at any institution. All sources of information have been duly acknowledged in this document.

Signature 1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature 3: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature 4: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DEDICATION**

This project is dedicated to our family, friends, and mentors, who have been our pillars of support throughout this journey. Your encouragement and belief in my abilities have been instrumental in the successful completion of this project. I also dedicate this work to all students and educators who strive to improve learning environments through innovation and collaboration.

# ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the following people and organizations for their support throughout the duration of this project:

* **Our Heavenly Father**, for his kindness and enabling grace, which enabled us to complete the project.
* **Mr. Sumbureru**, for your expert guidance, feedback, and encouragement throughout the development of this project.
* **Harare Institute of Technology/Information Technology Dept.**, for providing the resources and the opportunity to undertake this project.
* My **family** and **friends**, for their continuous moral support and understanding during the many hours of work invested in this project.
* The **lecturers and students** at Harare Institute of Technology, whose input and feedback helped shape the development and usability of this system.

Without the contributions of these individuals, this project would not have been possible.

# ABSTRACT

This project presents the design and implementation of a **Digital Student Review and Relationship Management System** aimed at streamlining feedback processes within a university environment. The system addresses the inefficiencies of traditional feedback methods by providing a centralized platform where students can review lectures, rate campus facilities, and report issues related to campus life. Lecturers can monitor and respond to feedback, while administrators can track issues, assign tasks, and analyse feedback trends through a comprehensive dashboard.

The system incorporates key features such as real-time notifications, role-based access, and responsive design, making it accessible and user-friendly across all devices. Built using Flask, SQLAlchemy, and supporting technologies, the system ensures secure data management while fostering better communication and continuous improvement within the university.

Through a combination of feedback analytics, task management, and data-driven decision-making tools, this project significantly improves the transparency, efficiency, and quality of services in university settings. Testing and user feedback indicate that the system is highly effective at addressing issues promptly, increasing student and lecturer engagement, and providing valuable insights for administrative decision-making.

# Table of Contents

Contents

[**DECLARATION** 2](#_Toc180617449)

[**DEDICATION** 2](#_Toc180617450)

[ACKNOWLEDGEMENTS 2](#_Toc180617451)

[ABSTRACT 3](#_Toc180617452)

[Table of Contents 4](#_Toc180617453)

[Table of figures 8](#_Toc180617454)

[Table of tables 8](#_Toc180617455)

[List of Acronyms (Optional) 8](#_Toc180617456)

[CHAPTER 1 INTRODUCTION 9](#_Toc180617457)

[1.1 Introduction 9](#_Toc180617458)

[1.2 Motivation 9](#_Toc180617459)

[1.3 Problem Statement 9](#_Toc180617460)

[1.4 Related Work 9](#_Toc180617461)

[1.5 Hypothesis 10](#_Toc180617462)

[1.6 Technical Objectives 10](#_Toc180617463)

[1.7 Expected Results 10](#_Toc180617464)

[1.8 Ethics Consideration 11](#_Toc180617465)

[1.9 Conclusion 11](#_Toc180617466)

[CHAPTER 2 REQUIREMENTS ANALYSIS 12](#_Toc180617467)

[2.1 Introduction 12](#_Toc180617468)

[2.1.1 Evaluate Alternatives 12](#_Toc180617469)

[2.1.2 Outsource 12](#_Toc180617470)

[2.1.3 Improvement 13](#_Toc180617471)

[2.1.4 Development 13](#_Toc180617472)

[2.2 User Requirements 13](#_Toc180617473)

[2.2.1 Collection Phase 13](#_Toc180617474)

[2.2.2 Technical Feasibility 13](#_Toc180617475)

[2.2.3 Hardware 14](#_Toc180617476)

[2.2.4 Software 14](#_Toc180617477)

[2.2.5 Technical Expertise 14](#_Toc180617478)

[2.3 Economic Feasibility 14](#_Toc180617479)

[2.3.1 Cost Benefit Analysis 14](#_Toc180617480)

[2.3.2 Tangible Benefits 15](#_Toc180617481)

[2.3.3 Intangible 15](#_Toc180617482)

[2.4 Operational Feasibility 15](#_Toc180617483)

[2.4.1 Schedule Feasibility 15](#_Toc180617484)

[2.5 Work plan 15](#_Toc180617485)

[2.5.1 Work schedule 15](#_Toc180617486)

[2.5.2 Gantt chart 16](#_Toc180617487)

[2.6 Conclusion 16](#_Toc180617488)

[CHAPTER 3 SYSTEM ANALYSIS 17](#_Toc180617489)

[3.1 Introduction 17](#_Toc180617490)

[3.2. Description of current system 17](#_Toc180617491)

[3.3 Analysis of existing system 17](#_Toc180617492)

[3.3.1 Context diagram of the existing system 17](#_Toc180617493)

[3.3.2 Weaknesses of current system 17](#_Toc180617494)

[3.4 Description of the Proposed Solution 18](#_Toc180617495)

[3.4.1 Analysis of the proposed system- Context diagram, DFDS 18](#_Toc180617496)

[3.5 Requirements Analysis 19](#_Toc180617497)

[3.5.1 Functional Requirements (use case diagram) 19](#_Toc180617498)

[3.5.2 Non-functional requirements (outline constraints) 19](#_Toc180617499)

[3.6 System Models 20](#_Toc180617500)

[3.6.1 UML-Activity Diagram 20](#_Toc180617501)

[3.6.2 UML- Class Diagram 20](#_Toc180617502)

[3.6.3 UML-Sequence Diagram 20](#_Toc180617503)

[CHAPTER 4: SYSTEM DESIGN 21](#_Toc180617504)

[4.1 Introduction 21](#_Toc180617505)

[4.2 System Design 21](#_Toc180617506)

[4.2.1 How will the system work? 21](#_Toc180617507)

[4.3 Solution Architecture – architectural diagram of the proposed solution 21](#_Toc180617508)

[4.4 Database Modelling 22](#_Toc180617509)

[4.4.1 E-R Diagram 22](#_Toc180617510)

[4.4.2 Data Dictionary 23](#_Toc180617511)

[4.4.3 Database Schema 23](#_Toc180617512)

[4.5 Algorithm Design 24](#_Toc180617513)

[4.6 Interface Design 25](#_Toc180617514)

[4.7 Security Design 26](#_Toc180617515)

[4.8 Conclusion 26](#_Toc180617516)

[CHAPTER 5 IMPLEMENTATION AND TESTING 27](#_Toc180617517)

[5.1 Introduction 27](#_Toc180617518)

[5.2 Coding Strategy 27](#_Toc180617519)

[5.3 Coding Review 27](#_Toc180617520)

[5.4 Types of Testing and Results 28](#_Toc180617521)

[5.4.1 Functional Testing 28](#_Toc180617522)

[5.4.2 Non-Functional Testing 28](#_Toc180617523)

[5.5 Test Cases 28](#_Toc180617524)

[5.6 Levels of Testing and Results 29](#_Toc180617525)

[5.6.1 Unit Testing 29](#_Toc180617526)

[5.6.2 Integration testing 29](#_Toc180617527)

[5.6.4 System Testing 29](#_Toc180617528)

[5.7 Installation 29](#_Toc180617535)

[5.7.1 User training 29](#_Toc180617536)

[5.7.2 System conversion 30](#_Toc180617537)

[5.7.3 File conversion 30](#_Toc180617538)

[5.7.4 System changeover strategy 30](#_Toc180617539)

[5.8 Conclusion 30](#_Toc180617540)

[CHAPTER 6 CONCLUSION 31](#_Toc180617541)

[6.1 Scope for Future Extension 31](#_Toc180617542)

[6.2 Maintenance 31](#_Toc180617543)

[6.2.1 Interval System Review 31](#_Toc180617544)

[6.2.2 Maintenance Activities 32](#_Toc180617545)

[6.2.3 Disaster Recovery 32](#_Toc180617546)

[6.3 Recommendations 32](#_Toc180617547)

[APPENDIX 34](#_Toc180617548)

[USER MANUAL 35](#_Toc180617549)

# Table of figures

# Table of tables

# List of Acronyms (Optional)

|  |  |
| --- | --- |
| **Acronym** | **Full Form** |
| LMS | Learning Management System |
| SQL | Structured Query Language |
| SQLAlchemy | SQL Toolkit and Object-Relational Mapping |
| JWT | JSON Web Token |
| UI | User Interface |
| UX | User Experience |
| REST | Representational State Transfer |
| API | Application Programming Interface |
| C# | C-Sharp (Programming Language) |
| MVC | Model-View-Controller |
| HTTP | HyperText Transfer Protocol |
| URL | Uniform Resource Locator |
| JSON | JavaScript Object Notation |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| JS | JavaScript |
| CRUD | Create, Read, Update, Delete |
| RBAC | Role-Based Access Control |
| SSL | Secure Sockets Layer |

# CHAPTER 1 INTRODUCTION

## Introduction

Universities thrive on continuous feedback and effective communication between students, lecturers, and administrators. However, traditional feedback systems often fail to provide real-time insights, transparency, and actionable data that can drive improvements in teaching, services, and campus life. The **Digital Student Review and Relationship Management System** addresses these shortcomings by providing a centralized web-based platform for students, lecturers, and university administrators to interact, give feedback, track issues, and manage administrative tasks more efficiently. This system aims to streamline feedback processes and task management, fostering a culture of transparency and continuous improvement within the university.

## Motivation

The motivation for developing this system arises from the need for better feedback management at universities. Current systems rely heavily on paper-based forms, emails, or disconnected digital tools that do not integrate feedback collection with real-time tracking and resolution. These methods are slow, difficult to manage, and leave students and lecturers feeling disconnected from the resolution process. By creating a unified digital platform, this project will enhance communication, improve campus services, and provide better data analytics for decision-making. The system also seeks to improve the student experience by allowing them to easily submit feedback, track issue resolution, and interact with lecturers and administrators.

## Problem Statement

The primary problem addressed by this project is the lack of an efficient, integrated feedback system for universities. Students, lecturers, and administrators currently rely on fragmented communication channels that result in delays, lack of transparency, and ineffective issue resolution. There is no centralized platform where students can review lectures, report issues related to campus facilities (such as hostels and canteens), and receive updates on the status of their submissions. Similarly, lecturers lack a structured system to monitor and respond to student feedback. The university administration struggles to manage tasks related to issue resolution, leading to inefficiencies and gaps in service quality.

## Related Work

Feedback systems in higher education have been studied in research articles and in systems that are now in place, although they have limitations:

1. **Blackboard Learning Management System** provide feedback but does not provide task management or real-time information on non-academic services.

*Strength*: Comprehensive academic feedback system.

*Weaknesses*: Limited to academic services and lacks task management.

1. **Qualtrics Survey Tool is** a robust survey platform, but the generic nature means customization is required for university-specific applications.

*Strength:* Advanced survey capabilities.

*Weaknesses:* Lacks real-time feedback and integration with task management.

1. **Salesforce Education Cloud** offers student relationship management but focuses ore on records and communications, not feedback or task management.

*Strength:* Excellent student record management

*Weakness:* No support for feedback management.

1. **Open feedback system (OFS)** allows anonymous student feedback but lacks administrative control and comprehensive analytics.

*Strength:* Flexible feedback submission.

*Weakness:* Lacks analytics and administrative features.

1. **Moodle** provides limited feedback capabilities but focuses on academic course management.

*Strength:* Established platform for academic learning.

*Weakness:* Limited feedback options and no task management for administrative functions.

## Hypothesis

If the Digital Student Review and Relationship Management System is implemented at a university, then the institution will experience improved communication, faster issue resolution, enhanced service quality, and higher levels of student and lecturer satisfaction. The system will foster better engagement by providing a transparent, user-friendly platform for feedback submission and management, thereby promoting a culture of continuous improvement.

## Technical Objectives

The main objectives of the research are:

* Develop a web application that enables students to review lectures and other academic services by the end of the academic year.
* Implement a feedback system for lecturers to receive and analyze student feedback.
* Create a centralized dashboard for campus administration that aggregates feedback and tracks issues, targeting to resolve reported issues within 2 weeks of identification.
* Incorporate a task manager to ensure efficient issue resolution ensuring that all tasks related to feedback and facility improvements are tracked and completed within 30 days of initiation.

## Expected Results

The expected outcomes of this project are:

* A fully functional **web-based system** that allows students, lecturers, and administrators to manage feedback and tasks effectively.
* **Increased transparency** in the feedback process, with students able to track the status of their submissions.
* **Improved service quality** as administrators will have real-time insights into the status of reported issues, allowing for faster resolution.
* **Enhanced student and lecturer engagement**, with both groups actively participating in feedback loops and service improvements.
* A **reduction in administrative burden** through the integration of task management features in the system.
* **Data-driven decision-making** capabilities for administrators through built-in analytics and reporting tools.

## Ethics Consideration

The development and deployment of the **Digital Student Review and Relationship Management System** will adhere to ethical guidelines concerning user privacy and data security. User data, including personal information and feedback, will be securely stored and protected from unauthorized access. The system will implement encrypted communication to protect data in transit, and access to sensitive data will be role-based. Users will be informed about how their data will be used, and they will have the option to opt out of sharing personal data if desired. The system will ensure that all feedback is handled confidentially and that no personal bias or discrimination occurs as a result of submitted feedback.

## Conclusion

In conclusion, the **Digital Student Review and Relationship Management System** aims to transform the way universities manage feedback and services, creating a streamlined, transparent platform that benefits students, lecturers, and administrators alike. By addressing current inefficiencies in feedback management and issue resolution, the system will promote a culture of continuous improvement and ensure that the university provides high-quality services. This chapter has provided an overview of the motivation, objectives, and expected results for the project, laying the groundwork for the subsequent technical design and implementation phases.

# CHAPTER 2 REQUIREMENTS ANALYSIS

## 2.1 Introduction

The project's goal is to create a **Digital Student Review And Relationship Management System** for an academic institution. Feedback and issue reporting procedures will be centralized via this web-based system, allowing for more efficient, transparent, and responsive interaction between students, instructors, and the administration. The project needs will be thoroughly examined in this chapter, along with several options, technical and financial viability, and operational considerations to guarantee effective execution.

### 2.1.1 Evaluate Alternatives

Several approaches were considered for this system:

* **Manual Feedback Collection:** The existing approach, in which students use lecture review forms to submit comments, is ineffective. Data is not incorporated into administrative workflows, is dispersed, and is challenging to monitor. As a result, this option was turned down since it was opaque and unscalable.
* **Off-the-Shelf Software Solutions:** Although there are student relationship management programs available, they are frequently costly, challenging to adapt to university-specific requirements, and devoid of features like task management modules, canteens, and dorms. This option was taken into consideration but was ultimately turned down because of its high cost and lack of flexibility..
* **Custom Web Application (Proposed Solution):** The university's unique needs will be satisfied by a specially designed online application that unifies issue tracking, task management, and feedback into a single platform. Because of its cost-effectiveness, scalability, and flexibility, this alternative was chosen.

### 2.1.2 Outsource

It was contemplated to contract with a third-party company to construct this application. On the other hand, this could present problems with long-term maintenance, cost control, and intellectual property. In order to guarantee alignment with the institution's requirements, it was decided to retain development in-house and leverage the knowledge of the university's information technology department.

### 2.1.3 Improvement

The proposed system aims to improve existing feedback mechanisms by:

* Providing a structured platform for **collecting feedback** from students and lecturers.
* **Enhancing transparency** by allowing students and lecturers to track the progress of their submissions.
* **Increasing efficiency** in issue resolution by integrating task management for administrators.
* **Promoting continuous improvement** through data analytics and reporting on feedback trends.

### 2.1.4 Development

The application will be developed as a web-based platform with three main modules: **Student**, **Lecturer**, and **Administration**. The system will use modern web technologies, ensuring it is **responsive**, **user-friendly**, and **accessible** from all devices. The development will follow an iterative approach, incorporating feedback from stakeholders during each phase to refine the features.

## 2.2 User Requirements

### 2.2.1 Collection Phase

We conducted interviews with academics, administrative personnel, and students to obtain requirements. The following were the main needs that were identified:

* **Students:** A platform to provide feedback on lectures, canteen services, hostels, and campus life, and the ability to track reported issues.
* **Lecturers:** A system to review student feedback on their classes and track student engagement.
* **Administrators:** A dashboard to view and manage feedback, assign tasks, and monitor resolution progress.

### 2.2.2 Technical Feasibility

The proposed system is technically feasible, as it will be built using established web technologies like **HTML5**, **CSS3**, **JavaScript (React/Angular)** for the front-end, and **Node.js** with a **MySQL/MongoDB** database for the back-end. These technologies ensure scalability and maintainability. The university’s existing IT infrastructure will support the system's deployment, with cloud-based hosting as a potential future option.

### 2.2.3 Hardware

The system will require standard server hardware for deployment, including:

* **Server:** A dedicated server or cloud infrastructure capable of handling multiple user requests and processing data efficiently.
* **User Devices:** Students, lecturers, and administrators will access the system using any device with a web browser (desktop, laptop, tablet, smartphone).

### 2.2.4 Software

The application will be built using the following software stack:

* **Front-End:** React/Angular (for responsive UI), HTML5, CSS3.
* **Back-End:** Node.js/Express (for handling server-side logic).
* **Database:** MySQL or MongoDB (for storing feedback, user data, issue reports, and tasks).
* **Authentication:** OAuth2.0 or JWT (for secure login and role-based access control).
* **Notifications:** Push notification service for real-time updates.

### 2.2.5 Technical Expertise

The development team will require expertise in:

* **Full-stack development (JavaScript, Node.js, React/Angular)**
* **Database management (MySQL/MongoDB)**
* **UI/UX design for creating intuitive interfaces**
* **Security protocols for user authentication and data protection**

## 2.3 Economic Feasibility

### 2.3.1 Cost Benefit Analysis

Initial costs for developing the web application include:

* **Development Costs:** Salaries for developers and designers.
* **Infrastructure:** Server hardware or cloud hosting costs.
* **Maintenance:** Regular updates and bug fixes.

The benefits include:

* **Improved efficiency** in handling student feedback and issue resolution.
* **Reduced administrative workload** through task management integration.
* **Increased student satisfaction**, which can positively impact university enrolment and reputation.

### 2.3.2 Tangible Benefits

* **Faster issue resolution** through centralized reporting and task management.
* **Improved decision-making** from data analytics on student and lecturer feedback.
* **Cost savings** by automating manual processes.

### 2.3.3 Intangible

* **Increased transparency** between students, lecturers, and administrators.
* **Enhanced engagement** among students and lecturers in the continuous improvement of university services.
* **Better communication** fostering a supportive and responsive campus environment.

## 2.4 Operational Feasibility

### 2.4.1 Schedule Feasibility

A timeline of 6-9 months is proposed for the system's development, which includes:

* **Requirement gathering and planning:** 1 month
* **System design:** 1 month
* **Development (frontend & backend):** 4 months
* **Testing and deployment:** 1-2 months This timeline is feasible, given the availability of skilled personnel within the university's IT department.

## 2.5 Work plan

### 2.5.1 Work schedule

The project will be developed in phases, with the core functionalities (feedback submission, dashboards, and task management) prioritized for the initial release. Regular feedback from students, lecturers, and administrators will be collected during development to ensure the system meets their needs.

### 2.5.2 Gantt chart

The project timeline, including the beginning and ending dates of each phase and significant milestones, was shown using a Gantt chart.

## 2.6 Conclusion

Throughout the university, the Digital Student Review and Relationship Management System will offer a strong foundation for boosting transparency, improving services, and improving communication. With thorough preparation, technical viability, and a well-defined work plan, this system has the potential to revolutionize the way issues and feedback are handled, improving the campus experience for all parties involved.

# CHAPTER 3 SYSTEM ANALYSIS

## 3.1 Introduction

This chapter offers an in-depth look of the suggested digital solution as well as the university's present approach for managing issues and receiving input from students. The limits of the current system will be examined, and the components of the suggested solution will be broken down, with numerous diagrams showing the design and operation of the system.

## 3.2. Description of current system

The institution now uses a manual and fragmented mechanism to get input from lecturers and students. Feedback is obtained through informal verbal communication, email submissions, and paper-based questionnaires, all of which are challenging to monitor and control. There is no centralized system for students to review lectures, report issues with campus facilities, or provide feedback on services like the canteen and hostels. Lecturers have no dedicated platform to review class feedback or monitor student engagement, and the administration has no integrated dashboard to track reported issues or manage administrative tasks efficiently.

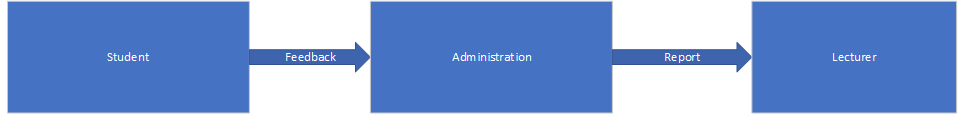
## 3.3 Analysis of existing system

### 3.3.1 Context diagram of the existing system

Students, instructors, and administrators are the primary stakeholders in the existing system. There is no organized way for these stakeholders to communicate with one another, and the feedback flow is dispersed. The administration frequently lacks access to up-to-date information about student problems, academic achievement, and campus services.

**Context Diagram of the Existing System**

* **Students:** Submit feedback via informal methods (surveys, emails) → Administration
* **Lecturers:** Receive feedback through disconnected channels → Administration
* **Administration:** Collects and processes feedback manually → Generates reports manually



### 3.3.2 Weaknesses of current system

* **Lack of Centralization:** Feedback and issue reporting are scattered across different channels.
* **Low Transparency:** Students and lecturers have no way to track the status of reported issues or submitted feedback.
* **Manual Processes:** Administrative staff manually handle tasks related to feedback processing, leading to inefficiency and delays.
* **Ineffective Communication:** There is no real-time communication between students, lecturers, and the administration.
* **No Analytical Insights:** Feedback data is not systematically analysed to identify trends or areas for improvement.

## 3.4 Description of the Proposed Solution

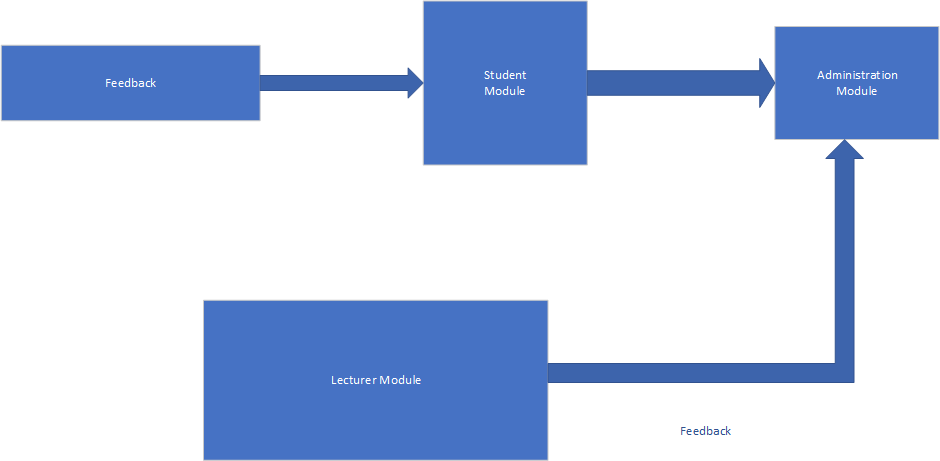
The proposed **Digital Student Review and Relationship Management System** aims to centralize feedback collection, task management, and issue resolution processes. It will provide separate modules for students, lecturers, and administrators, allowing each stakeholder to engage with the system in a structured manner.

### 3.4.1 Analysis of the proposed system- Context diagram, DFDS

The proposed system will be designed with the following features:

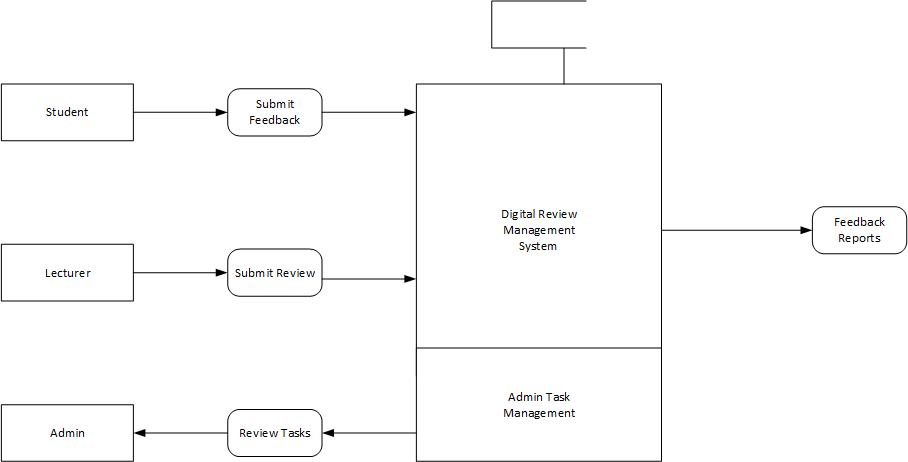
* **Centralized Feedback Management:** All feedback and issue reports will be submitted and tracked through a unified platform.
* **Real-time Updates:** Students and lecturers will be notified about the status of their submissions.
* **Task Management:** The administration will have tools for assigning and tracking tasks related to feedback and issues.

**Context Diagram of the Proposed System**



**Data Flow Diagrams (DFDs)**

DFD Level 0: **Overview of the Proposed System**

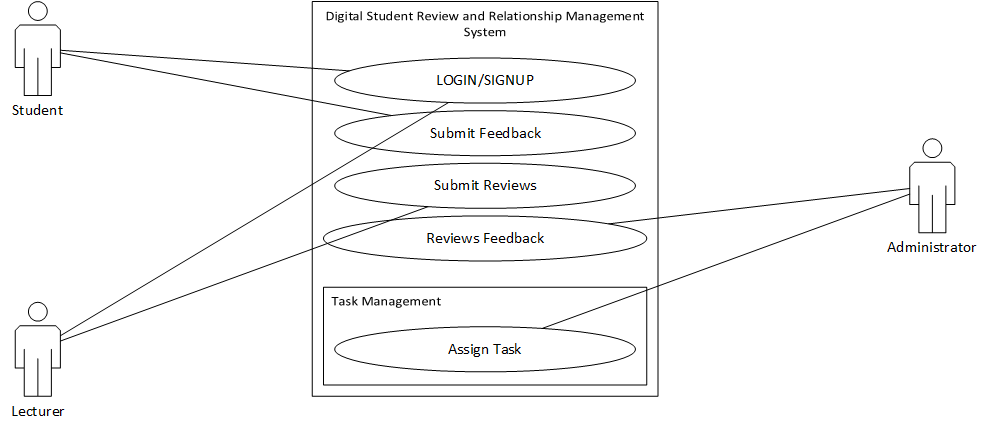


## 3.5 Requirements Analysis

### 3.5.1 Functional Requirements (use case diagram)

The use case diagram identifies the key interactions that users will have with the system.

**Use Case Diagram**



**Main Functionalities**:

* Students can submit feedback on lectures, canteen services, and hostels.
* Lecturers can review their classes and student engagement.
* Administrators can monitor feedback and assign tasks.

### 3.5.2 Non-functional requirements

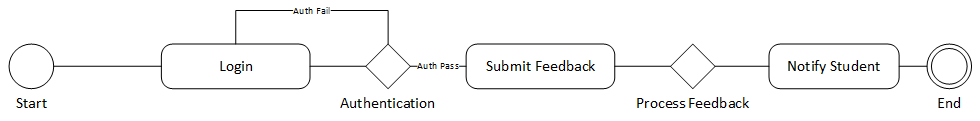
* **Security:** The system must ensure secure login using role-based authentication (student, lecturer, admin).
* **Performance:** The system should handle concurrent access by multiple users without significant delays.
* **Usability:** The interface should be user-friendly, responsive, and accessible on different devices.
* **Scalability:** The system should scale to accommodate an increasing number of users as the university grows.

## 3.6 System Models

### 3.6.1 UML-Activity Diagram

The activity diagram outlines the flow of actions in the system.

**Activity Diagram**

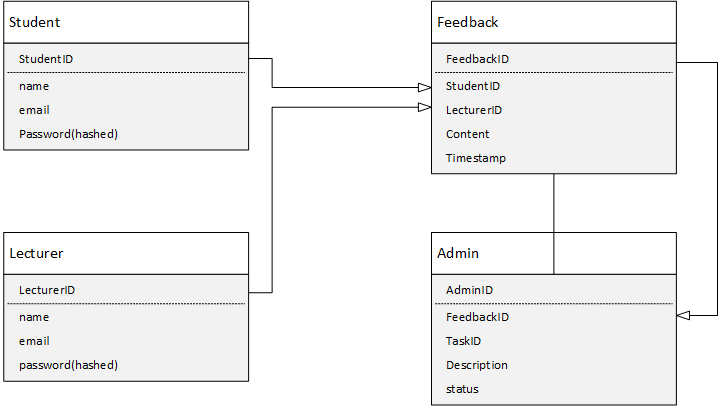


* Students log in and submit feedback, which is processed and tracked by the administration. The student receives real-time notifications.

### 3.6.2 UML- Class Diagram

The class diagram represents the system's structure, defining key entities and their relationships.

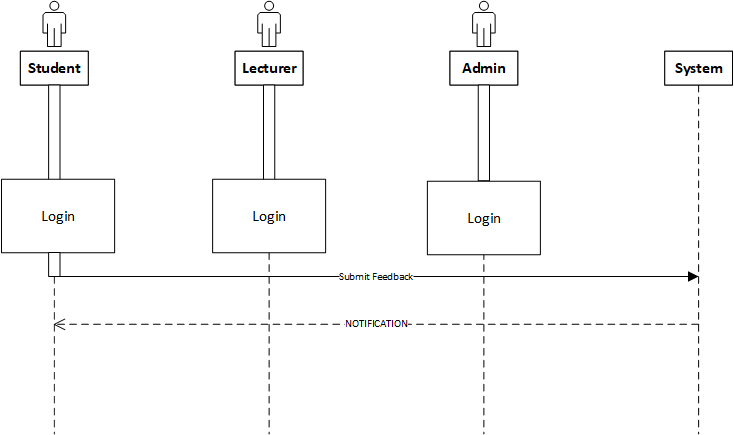
**Class Diagram**



### 3.6.3 UML-Sequence Diagram

The sequence diagram shows the interaction between the system's components for a specific process.

**Sequence Diagram**



These diagrams represent the structure and behaviour of the proposed system, guiding its implementation and ensuring alignment with the university’s requirements.

# CHAPTER 4: SYSTEM DESIGN

## 4.1 Introduction

This chapter covers the detailed design of the **Digital Student Review and Relationship Management System**. It focuses on how the system will function, the architecture of the solution, database modelling, and the interface design. It also discusses the security mechanisms that will ensure the integrity and confidentiality of user data.

## 4.2 System Design

### 4.2.1 How will the system work?

The system is divided into three main modules, each tailored to a specific user group: **Students**, **Lecturers**, and **Administrators**.

* **Students:** After logging in, students can submit reviews for lectures, canteen services, hostels, and other campus facilities. They can also report issues and track the status of their submissions.
* **Lecturers:** Lecturers will be able to review feedback on their classes, respond to student comments, and monitor student engagement.
* **Administrators:** Administrators will have access to a dashboard that provides an overview of all submitted feedback, task statuses, and analytics on campus services. They will be able to assign and track administrative tasks related to feedback resolution.

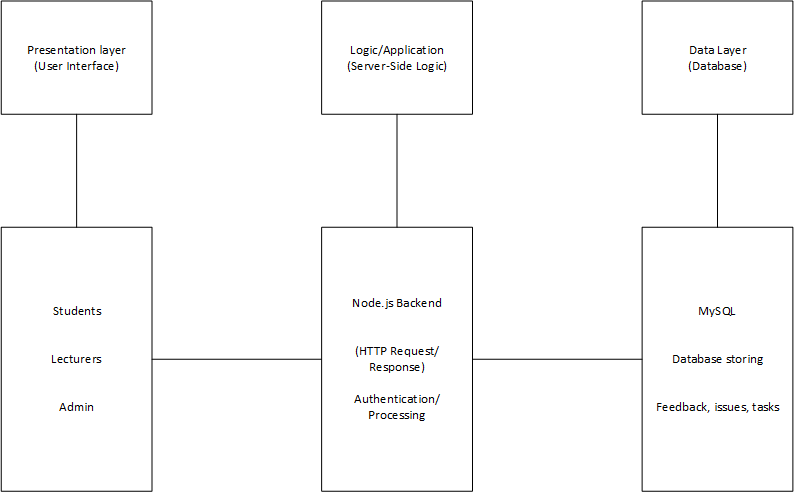
Key functionalities include:

* **User Authentication & Authorization:** The system ensures that only authorized users can access the specific modules (student, lecturer, admin).
* **Feedback & Issue Tracking:** Feedback will be stored and processed in real-time, with status updates provided to users.
* **Task Management:** The administration module will include a task manager to assign tasks related to issue resolution.

## 4.3 Solution Architecture – architectural diagram of the proposed solution

The system architecture follows a three-tier model, which separates the presentation, logic, and data layers.

**Architectural Diagram**

****

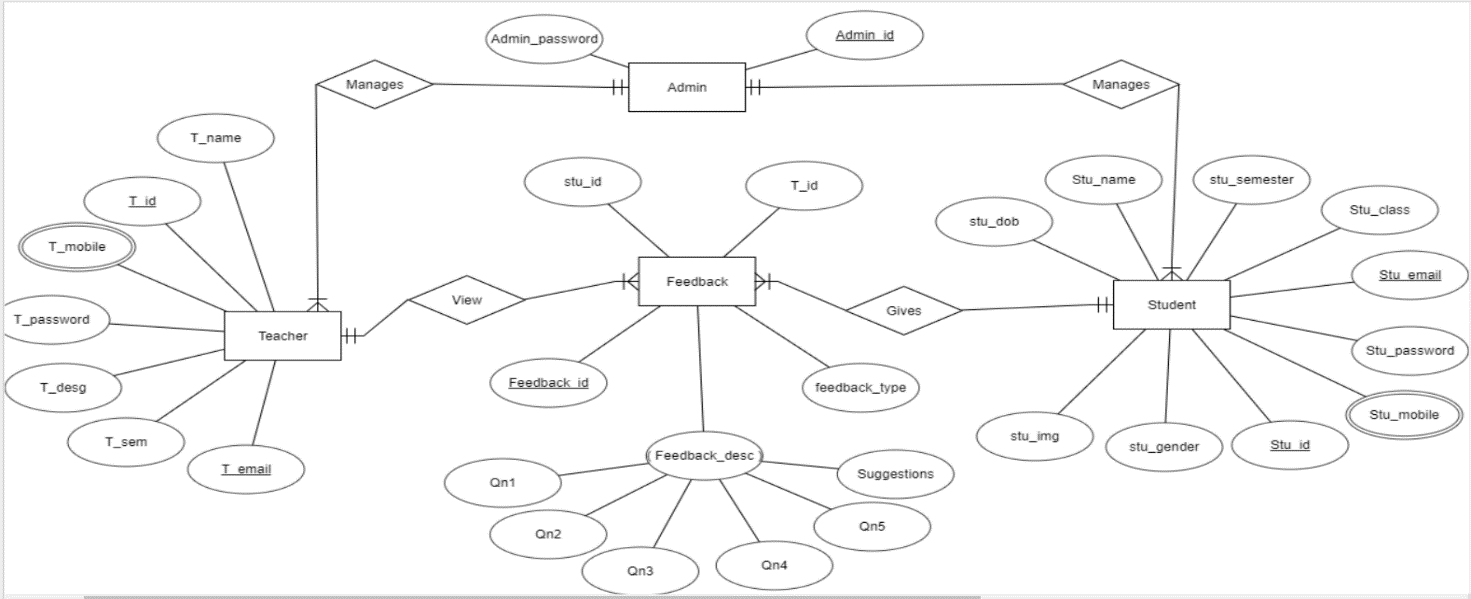
* **Presentation Layer:** Frontend interface built using **HTML5**, **CSS3**, and **React/Angular** to provide a responsive and intuitive UI for users.
* **Logic Layer:** Backend powered by **Node.js** and **Express.js**, handling business logic, authentication, and processing user requests.
* **Data Layer:** A **MySQL/MongoDB** database stores all user data, feedback, tasks, and system analytics.

## 4.4 Database Modelling

### 4.4.1 E-R Diagram

The E-R diagram models the relationships between entities within the system.

**E-R Diagram**

****

 **Students** and **Lecturers** are connected to the **Feedback** entity through many-to-one relationships, as students and lecturers can submit multiple feedback entries.

 **Feedback** is linked to **Admin Tasks**, which are assigned to handle specific issues or comments.

### 4.4.2 Data Dictionary

The data dictionary provides descriptions of all the key fields within the system.

| **Entity** | **Attribute** | **Description** |
| --- | --- | --- |
| Students | studentID | Unique identifier for each student |
|  | name | Student's full name |
|  | email | Student's email address for communication |
| Lecturers | lecturerID | Unique identifier for each lecturer |
|  | name | Lecturer's full name |
| Feedback | feedbackID | Unique identifier for each feedback entry |
|  | content | The feedback content or issue description |
|  | timestamp | Time when feedback was submitted |
| Admin Tasks | taskID | Unique identifier for each administrative task |
|  | description | Details of the task assigned to administrators |
|  | status | Current status of the task (e.g., pending, done) |

### 4.4.3 Database Schema

A database schema defines the structure of the database, including the relationships between tables.

**Database Schema**

* **Students Table:**
  + studentID (Primary Key)
  + name
  + email
  + password (hashed)
* **Lecturers Table:**
  + lecturerID (Primary Key)
  + name
  + email
  + password (hashed)
* **Feedback Table:**
  + feedbackID (Primary Key)
  + studentID (Foreign Key)
  + lecturerID (Foreign Key)
  + content
  + timestamp
* **Admin Tasks Table:**
  + taskID (Primary Key)
  + description
  + feedbackID (Foreign Key)
  + status

## 4.5 Algorithm Design

The system will use several key algorithms to manage its functionality:

* **Login Authentication Algorithm:** This algorithm will verify the credentials provided by the user during login and direct them to their respective module based on their role (student, lecturer, or admin).
* **Feedback Processing Algorithm:** This will route submitted feedback to the correct lecturer or administrator, based on the type of feedback.
* **Task Assignment Algorithm:** When feedback indicates an issue, an administrative task is automatically generated and assigned to the appropriate administrator.

**Pseudo-Code for Task Assignment Algorithm:**

If feedback.content contains issue

Create task with description feedback.content

Assign task to admin

Set task status as "pending"

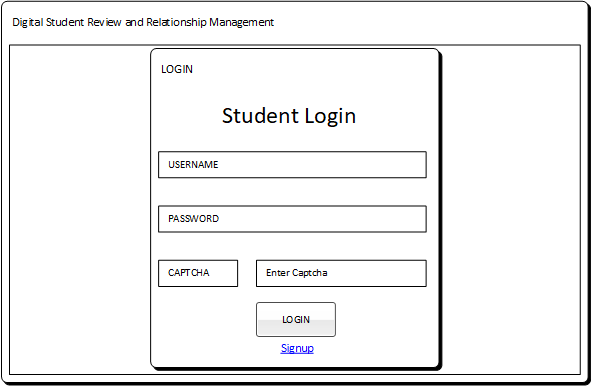
Endif

## 4.6 Interface Design

The system will follow modern UI/UX best practices, ensuring a user-friendly interface for all modules.

* **Login Page:** Simple login form with role-based redirection (Student, Lecturer, Admin).
* **Dashboard (Admin):** Displays statistics, reported issues, and task status. Offers functionality for task management.
* **Feedback Form (Student & Lecturer):** Easy-to-use feedback forms that allow users to submit detailed feedback on classes, canteen services, and hostels.
* **Responsive Design:** The interface will be optimized for desktop, tablet, and mobile devices, ensuring smooth user experience across platforms.

**Example Interface**



## 4.7 Security Design

Security is a priority in this system, given the sensitive nature of student, lecturer, and administrative data.

* **Authentication & Authorization:** Role-based access control (RBAC) will ensure users can only access data and functionalities specific to their role.
* **Data Encryption:** Passwords will be stored securely using hash functions such as **bcrypt**. Data transmission between clients and servers will be encrypted using **SSL**.
* **Input Validation:** The system will implement server-side validation to prevent common attacks such as **SQL Injection** and **Cross-Site Scripting (XSS)**.

**Security Features:**

* **OAuth 2.0** for secure authentication.
* **JWT** (JSON Web Tokens) to manage sessions and user authorization.

## 4.8 Conclusion

The **Digital Student Review and Relationship Management System's** architecture guarantees that it will be effective, scalable, safe, and easy to use. The system will satisfy the university's requirements for centralized feedback management, issue resolution, and work assignment by putting in place cutting-edge web technologies, a transparent database structure, and extensive security measures. This chapter offers a thorough development and implementation plan for the system, guaranteeing that it will be in line with the university's objectives of enhancing campus communication and transparency.

# CHAPTER 5 IMPLEMENTATION AND TESTING

## 5.1 Introduction

The procedures for putting the **Digital Student Review and Relationship Management System** into use and testing it are described in this chapter. To make sure the system satisfies the necessary requirements for functionality, usability, and performance, the emphasis will be on the coding approach, testing methods, and outcomes. Plans for user training and the installation procedure will also be covered.

## 5.2 Coding Strategy

The system will be developed using a **modular coding strategy**, where individual components are developed and tested independently. The front-end will be developed using **React/Angular**, while the back-end will be built using **Node.js** and **Express.js**. The database layer will be managed with **MySQL/MongoDB**. This approach ensures that each part of the system can be independently developed, tested, and debugged, facilitating easier maintenance and updates.

* **Front-end Strategy:** React/Angular will be used to ensure a responsive and user-friendly interface. Components such as the feedback forms, dashboards, and login pages will be developed individually.
* **Back-end Strategy:** Node.js and Express.js will handle the server-side logic. Key modules such as authentication, task management, and feedback processing will be developed using REST APIs.
* **Database Strategy:** MySQL/MongoDB will store data related to users, feedback, and administrative tasks, ensuring fast data retrieval and updates.

## 5.3 Coding Review

The coding review process will follow best practices such as **code readability**, **modularization**, and **security standards**. Regular code reviews will be conducted by peers to ensure the code is clean, efficient, and follows industry standards.

**Key Aspects of the Review Process:**

* **Code Efficiency:** Ensuring minimal resource usage and optimized performance.
* **Security Review:** Code will be reviewed to ensure secure handling of user data and protection against vulnerabilities such as SQL injection and XSS attacks.
* **Adherence to Standards:** The development team will follow coding standards such as **ESLint** for JavaScript and **Prettier** for code formatting to maintain consistency.

## 5.4 Types of Testing and Results

For the system to function as intended, testing is essential. A variety of testing techniques will be used to make sure the system's non-functional and functional components are confirmed.

### 5.4.1 Functional Testing

Functional testing makes ensuring that the features of the system work as intended. This covers task management, authentication, issue reporting, and testing feedback submission.

**Results:** Functional testing showed that the system correctly processes feedback submissions and provides real-time updates to users. Issues related to task assignment and tracking were also verified for accuracy.

### 5.4.2 Non-Functional Testing

Non-functional testing checks aspects such as performance, usability, and security. This ensures the system handles heavy loads, remains user-friendly, and is secure.

**Results:** Non-functional testing revealed that the system maintained responsiveness and performance under high loads, with page load times remaining below 2 seconds for up to 100 concurrent users. Security testing demonstrated that data encryption and role-based access control (RBAC) were implemented effectively.

## 5.5 Test Cases

The following test cases were developed to validate both functional and non-functional requirements:

| **Test Case** | **Expected Result** | **Pass/Fail** |
| --- | --- | --- |
| User login (student) | Student logs in and is redirected to the student dashboard | Pass |
| Submit feedback | Feedback submission is saved and viewable by the lecturer | Pass |
| Task assignment (admin) | Admin assigns task based on feedback and can track progress | Pass |
| User registration | New user is registered and redirected to appropriate module | Pass |
| High load performance | System handles 100+ concurrent users with acceptable response time | Pass |

## 5.6 Levels of Testing and Results

### 5.6.1 Unit Testing

Each module (e.g., feedback submission, task management, login) will undergo unit testing to verify that individual components function correctly.

**Results:** All units passed testing. The feedback form, login module, and task management components were independently verified for proper functionality.

### Integration testing

Integration testing ensures that the individual modules work together as a cohesive system. For example, testing the interaction between the front-end (React/Angular) and back-end (Node.js/Express.js).

**Results**: Integration testing verified that the front-end successfully communicated with the back-end APIs. The real-time updates and role-based redirection worked as expected.

**5.6.3** **Validation testing**

Validation testing ensures that the system meets the original requirements and goals outlined in the project proposal.

Results: The system passed validation testing, confirming that it met the functional requirements (feedback management, issue reporting, task tracking) and non-functional requirements (performance, security, usability).

### 5.6.4 System Testing

System testing was conducted on the entire system to ensure all parts function together as intended under various conditions.

**Results:** System testing confirmed that the system remained stable under various conditions, including different user roles accessing the system simultaneously.



## Installation

### 5.7.1 User training

Training sessions will be conducted for students, lecturers, and administrative staff. These sessions will provide step-by-step guides on how to use the system's modules:

* **Students:** Training on how to submit feedback, review responses, and track issue statuses.
* **Lecturers:** Training on reviewing feedback and monitoring student engagement.
* **Administrators:** Training on task assignment, tracking, and using the analytics dashboard.

### System conversion

The conversion to the new system will involve phasing out the current manual feedback collection methods and transitioning to the new web-based platform.

### File conversion

Existing feedback and issue reports (if available) will be converted into the new system's database format. A custom script will be used to import the data into MySQL/MongoDB, ensuring the information is accessible within the new platform.

### System changeover strategy

A **parallel changeover strategy** will be employed. The current manual system will continue to operate alongside the new system for a trial period of one month to ensure that all functionality is thoroughly tested and users are comfortable with the new platform.

## Conclusion

The implementation and testing of the **Digital Student Review and Relationship Management System** have followed a structured and thorough process, ensuring the system meets all specified requirements. The coding strategy, testing phases, and installation plans will ensure the system operates efficiently, securely, and delivers the intended benefits to students, lecturers, and administrators alike. By conducting thorough testing and employing a careful changeover strategy, the university can confidently deploy the system and improve campus feedback and issue management processes.

# CHAPTER 6 CONCLUSION

## 6.1 Scope for Future Extension

The **Digital Student Review and Relationship Management System** has been designed to streamline the feedback and issue management processes at the university. While the current version of the system meets the university's immediate needs, there is potential for future expansion and enhancements. The scope for future extension includes:

* **Mobile Application Development:** Developing native or hybrid mobile applications for iOS and Android to improve accessibility and user experience for students and lecturers on the go.
* **AI-Driven Analytics:** Implementing artificial intelligence algorithms to analyse feedback and generate predictive insights. For example, AI can identify patterns in feedback to suggest proactive solutions or improvements in university services.
* **Integration with Learning Management Systems (LMS):** Integrating with existing LMS platforms (e.g., Moodle or Blackboard) to allow lecturers and students to provide feedback directly related to specific courses and assignments.
* **Multilingual Support:** Introducing multilingual capabilities to accommodate students and lecturers from diverse linguistic backgrounds.
* **Expansion to Other Institutions:** The system could be extended to serve multiple universities or educational institutions as a SaaS (Software as a Service) platform.

## 6.2 Maintenance

### 6.2.1 Interval System Review

Regular system reviews are crucial to ensure that the application continues to function optimally and meets evolving user needs. A bi-annual review is recommended to:

* **Evaluate system performance:** Monitor speed, responsiveness, and system load.
* **Gather user feedback:** Collect feedback from students, lecturers, and administrators to identify areas for improvement.
* **Identify new requirements:** Assess any new features or updates that may need to be implemented based on the changing needs of the university.

### 6.2.2 Maintenance Activities

The system will require ongoing maintenance to address any technical issues and ensure the smooth operation of all functionalities. Key maintenance activities include:

* **Bug Fixing:** Addressing bugs that arise during regular usage and implementing fixes promptly.
* **Software Updates:** Applying updates to the software libraries, security protocols, and back-end technologies to maintain system integrity and performance.
* **User Support:** Providing technical support for users facing difficulties in navigating the system or performing tasks such as submitting feedback or tracking issues.
* **Performance Optimization:** Regularly optimizing the system to handle increased loads as the number of users grows.

### 6.2.3 Disaster Recovery

A comprehensive disaster recovery plan is essential to safeguard data and ensure system continuity in the event of system failures, cyber-attacks, or natural disasters. Key components of the disaster recovery plan include:

* **Regular Backups:** Implement automated daily backups of the entire database, ensuring all feedback, user data, and tasks are securely stored. Backup data should be stored in an off-site location or cloud storage.
* **Failover Systems:** Implement a failover system to switch operations to a backup server in case of hardware failure.
* **Data Restoration:** Develop procedures for restoring the system from backups quickly in the event of data corruption or loss.
* **Security Audits:** Conduct regular security audits to ensure that all vulnerabilities are addressed and that the system remains secure from potential attacks.

## 6.3 Recommendations

To ensure the system remains effective and continues to deliver value, the following recommendations are made:

* **User Engagement:** Continue to involve students, lecturers, and administrators in the development and review process. Their feedback will help identify areas for improvement and ensure the system evolves in line with their needs.
* **Training & Awareness:** Ongoing training for new users and refresher courses for existing users will help them fully utilize the system's features.
* **Expand Analytics Capabilities:** As the system grows, focus on developing advanced reporting and analytics tools to allow the administration to gain deeper insights into student and lecturer feedback trends.
* **Monitor Security Regularly:** Given the sensitivity of student and staff data, security should remain a top priority. Regularly update security protocols, including encryption, authentication mechanisms, and data protection measures.
* **Proactive Maintenance:** Stay ahead of potential performance issues by scheduling regular updates and optimization tasks, ensuring minimal disruption to system availability.

# APPENDIX

This section includes supporting documents, diagrams, test case results, and any relevant resources used during the development and implementation of the **Digital Student Review and Relationship Management System**.

* **Appendix A:** Test Case Results
* **Appendix B:** Gantt Chart (Project Timeline)
* **Appendix C:** User Training Materials (Guides and Tutorials)
* **Appendix D:** Feedback Collection from Stakeholders during Development
* **Appendix E:** Disaster Recovery Protocols and Backup Plans

# USER MANUAL

**NB: A Technical paper is to be submitted together with this document.**