2024

CDMS

FINAL DOCUMENTATION

Prepared for :

custody database management system

cdms.vercel.app

SUEZ CANAL UNIVERSITY
FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS
COMPUTER SCIENCE PROGRAM

CDMS

CUSTODY DATABASE MANAGEMENT SYSTEM

2024-2025

In Partial Fulfilment Of The Requirements For the Degree Of Bachelor Of Science In Computer Science

BY STUDENTS

YOUSSEF MOHAMED **AHMED SALAH**

BASEM MOHAMED **ESLAM EID**

SUPERVISED BY

DR.RIHAM MOHARAM

DR. DINA ELMANAKHLY

ACKNOWLEDGEMENT

We express our gratitude to the Almighty Allah for granting us this opportunity to benefit humanity through our project. With His blessings, we aim to spread knowledge that will make a remarkable impact.

We extend our heartfelt appreciation to our supervisors, Dr. Riham Moharam and Dr. Dina Elmanakhly, for their invaluable guidance, wisdom, and unwavering support throughout this journey.

We also want to thank the faculty members and students who generously shared their insights and requirements, helping us understand the needs and challenges of the existing system. Your participation and collaboration have been instrumental in creating a user-centric solution.

To our exceptional team members, thank you for your hard work, creativity, and commitment to excellence. Each of you brought unique skills and perspectives, contributing to a harmonious and productive work environment. Together, we have achieved something remarkable.

To everyone mentioned, we sincerely appreciate your contributions, support, and belief in us. This project would not have been possible without each and every one of you. We are honoured to have had the opportunity to work with such incredible individuals.

With deep gratitude and warm regards,

The Custody Database Management System (CDMS) Team

TABLE OF CONTENTS

01

02

03

04

05

LIST OF TABLES

01

LIST OF FIGURES

FIGURES

02

We express our gratitude to the Almighty Allah for granting us this opportunity to benefit humanity through our project. With His blessings, we aim to spread knowledge that will make a remarkable impact.

We extend our heartfelt appreciation to our supervisors, Dr. Riham Moharam and Dr. Dina Elmanakhly, for their invaluable guidance, wisdom, and unwavering support throughout this journey.

We also want to thank the faculty members and students who generously shared their insights and requirements, helping us understand the needs and challenges of the existing system. Your participation and collaboration have been instrumental in creating a user-centric solution.

To our exceptional team members, thank you for your hard work, creativity, and commitment to excellence. Each of you brought unique skills and perspectives, contributing to a harmonious and productive work environment. Together, we have achieved something remarkable.

To everyone mentioned, we sincerely appreciate your contributions, support, and belief in us. This project would not have been possible without each and every one of you. We are honoured to have had the opportunity to work with such incredible individuals.

With deep gratitude and warm regards,

The Custody Database Management System (**CDMS**) Team

The Custody Database Management System (**CDMS**) is a sophisticated software solution tailored to meet the custody management needs of universities, facilitating transparent and efficient handling of various custody items, including machines, laptops, and equipment allocated to employees. With each university entrusted with the responsibility of safeguarding and managing its assigned custody, **CDMS** emerges as a comprehensive tool to ensure accountability, streamline custody tracking, and mitigate the risks associated with mismanagement and theft.

By incorporating innovative features and a user-friendly interface, **CDMS** simplifies the process of tracking university custody, ensuring accountability and minimizing the risk of mismanagement. It provides comprehensive data analysis, empowering stakeholders to make informed decisions and administrators to allocate resources efficiently.

What sets **CDMS** apart is its ability to transform custody management practices through advanced technologies and design philosophy. Leveraging cutting-edge tools and methodologies, **CDMS** streamlines custody tracking and enhances communication channels, ensuring accuracy, efficiency, and transparency.

The development of **CDMS** involved thorough analysis of university custody management practices. It embraces the latest technological advancements to deliver a solution that fosters a more transparent and accountable environment for managing university custody. The resulting solution offers a user-friendly interface, robust security features, and scalability to meet the evolving needs of universities.

In conclusion, **CDMS** revolutionizes university custody management by enhancing efficiency, accuracy, and transparency. Its innovative features, advanced technologies, and user-centred design position it as a transformative solution for higher education administration. With **CDMS**, universities can streamline custody tracking, optimize resource allocation, and empower stakeholders to make informed decisions, marking a significant advancement in custody management practices.

Welcome to the documentation for the **CDMS** project. This chapter provides an overview of the project, including its objectives, problem statement, background, scope, methodologies, literature review, and document structure.

1.1 Problem Statement

The current custody management system in our university faces challenges like device loss, difficulty accessing information, and tracking received dates and device locations. These issues stem from manual processes and scattered data storage, leading to delays and mistakes. Stakeholders struggle to manage custody effectively and ensure device security. To address these challenges, we urgently need a modern custody database management system. This system should simplify processes, improve accountability, and make accessing information easier. Its implementation aims to enhance custody management practices and prevent further device loss or mismanagement.

1.2 What is CDMS?

The Custody Database Management System (**CDMS**) is a proposed solution tailored to revolutionize custody management within our university. **CDMS** aims to provide a streamlined and user-friendly platform for both stakeholders and administrative staff. Leveraging modern technologies such as web-based interfaces and real-time data integration, **CDMS** seeks to simplify custody management processes. The system will automate manual tasks, facilitate accurate custody information retrieval, and streamline custody allocation procedures.

1.3 Project Background

The realm of custody management systems has witnessed notable advancements, with universities increasingly adopting digital solutions to

streamline custody management processes. Our project seeks to capitalize on modern technologies and industry best practices to develop a user-friendly and efficient custody database management system. By addressing the shortcomings of the current system and incorporating feedback from stakeholders, our objective is to improve the custody management experience for both users and administrative staff.

1.4 Objectives and Goals

The primary goal of the Custody Database Management System (**CDMS**) project is to develop a modern and efficient custody management system that enhances the custody management experience for all stakeholders involved. The system aims to automate and streamline custody processes, reducing errors, delays, and administrative burdens.

To achieve this goal, the objectives of the project are:

- **Improve User Experience:** Enhance the custody management process for stakeholders, including employees, administrative staff, and supervisors, by providing a user-friendly interface and eliminating manual steps.
- **Automate Custody Processes:** Streamline and automate various tasks, such as custody allocation, tracking, and reporting, to save time and increase efficiency.
- **Enhance Accessibility and Availability:** Ensure easy access to up-to-date custody information, enabling stakeholders to make informed decisions.
- **Improve Data Accuracy and Integrity:** Implement robust data validation mechanisms to ensure accurate and reliable custody records and information.
- **Support Scalability and Future Growth:** Design the system to accommodate future growth, increased custody demands, and integration with other university systems.

By achieving these objectives, **CDMS** aims to transform the custody management process, making it more streamlined and user-friendly for all stakeholders involved.

1.5 Scope

The scope of the Custody Database Management System (**CDMS**) project encompasses the development of a comprehensive custody management system. The system will cover the entire custody management process, from custody allocation to tracking and reporting, for various custody items within our university.

The key elements within the scope of the project include:

Custody Allocation: The system will provide a user-friendly interface for custodians and administrative staff to allocate custody items to employees, assign custody locations, and track custody movements.

Custody Tracking: The system will enable stakeholders to track the custody history of items, including received dates, locations, and custody supervisors, ensuring transparency and accountability in custody management practices.

Schedule Generation: The system will generate class schedules based on course availability and student preferences, aiming to minimize conflicts and optimize course selections.

Reporting and Analysis: The system will generate reports and analytics to provide insights into custody utilization, trends, and potential areas for improvement, empowering administrators to make informed decisions regarding custody allocation and management.

Data Management: The system will ensure the accuracy and integrity of custody data, including item details, custody histories, and user permissions, through robust data validation .

It is important to note that the scope of the project is limited to the development of the CDMS software system and its implementation, and it does not include hardware or networking infrastructure.

Any future enhancements, system integrations, or additional features beyond the defined scope will be considered for future project phases or separate initiatives.

1.6 Methodologies

The Custody Database Management System (**CDMS**) project will adopt an iterative and collaborative development approach. It will involve requirement elicitation through stakeholder consultations, analysis of existing custody management systems, and feedback from end users. The system design will be guided by industry-standard practices and principles of user-cantered design, ensuring a user-friendly interface and efficient workflow. During the implementation phase, modern technologies and programming languages will be utilized to develop a robust and scalable system. Rigorous testing, including functional and usability testing, will be conducted to ensure the reliability and usability of the **CDMS**.

1.7 Literature Review: Custody Management Systems

Custody management systems play a pivotal role in educational institutions, facilitating the efficient tracking and management of various custody items. The adoption of technology-based custody management systems has become increasingly prevalent, aiming to streamline processes, enhance efficiency, and improve overall management practices. This literature review delves into existing research on custody management systems, focusing on their features, implementation, and effectiveness in educational settings. By examining this body of knowledge, our aim is to identify best practices and inform the development of an efficient custody management system for our project.

Custody management systems offer a range of features designed to improve custody tracking and streamline management processes. These features may include a user-friendly interface, centralized data storage, real-time tracking capabilities, and comprehensive reporting functionalities. Additionally, the implementation of a custody management system requires universities to invest in new technologies, provide training for users, and allocate resources for system development and maintenance. Despite the potential benefits, universities may encounter challenges such as technical expertise limitations, budget constraints, and resistance to change during system implementation.

Usability is a critical factor influencing the effectiveness of custody management systems and user satisfaction. A system's usability depends on various elements such as interface design, navigation, functionality, content,

and user support. Studies suggest that improving system usability can enhance the user experience for stakeholders involved in custody management.

The cost of implementing a custody management system can be significant for universities, especially those with limited resources. Factors influencing the cost include system features, size, complexity, and ongoing maintenance requirements. Universities should carefully evaluate the costs against potential benefits and long-term return on investment when considering system implementation.

Effectiveness metrics for custody management systems include user activity, user satisfaction, system performance, custody tracking accuracy, and overall management efficiency. An effective custody management system can streamline processes, reduce errors, and enhance accountability in custody management practices.

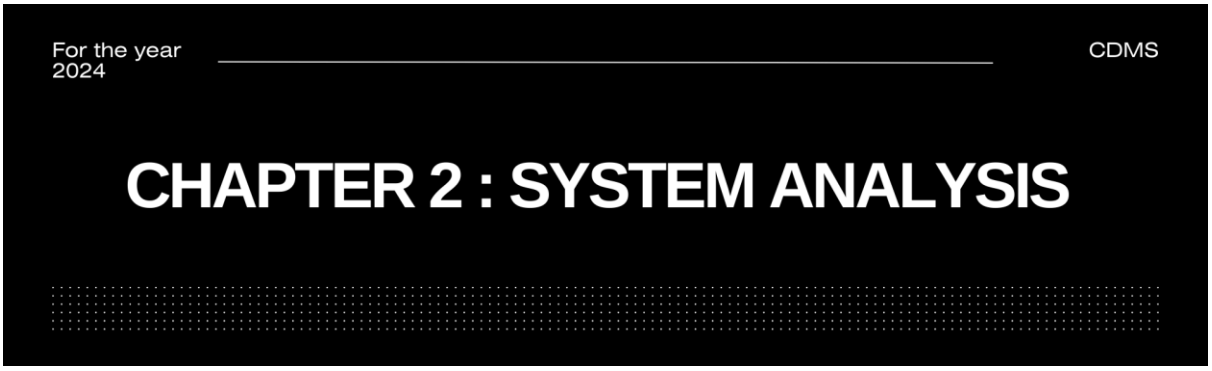
In conclusion, custody management systems are essential tools for universities to streamline custody management processes, improve efficiency, and enhance overall management practices. Key factors to consider when selecting and implementing a custody management system include system features, implementation considerations, usability, cost, and effectiveness metrics. Integration with other academic systems, adoption of open-source solutions, and collaboration with other institutions can further optimize system efficiency and effectiveness.

1.8 Document Structure

The documentation is organized into the following chapters:

- Chapter 1: Introduction: Provides a comprehensive overview, setting the stage by outlining the project's objectives, problem statement, background, scope, methodologies, literature review, and document structure.
- Chapter 2: System Analysis: Provides an in-depth analysis of the existing student registration system, including process analysis, technology assessment, user feedback, and data analysis.
- Chapter 3: System Design: Presents the system design, architecture, database schema, and user interface design.

- Chapter 4: Implementation & Development: Describes the implementation details, development environment, and technologies used.
- Chapter 5: Conclusion & Future Works: Summarizes the project and suggests future enhancements.
- References: contains a list of the work citations referenced in the project. It includes academic papers that have been used to support the information presented in the earlier chapters.



The System Analysis chapter focuses on analyzing **CUSTODY DATABASE MANAGEMENT SYSTEM** to gather requirements, understand user needs, evaluate the current system, and identify assumptions and constraints. The key sections covered in this chapter are:

1. Requirement Elicitation
2. Requirement Documentation
3. Current System Analysis
4. Assumptions and Constraints

2.1 Requirement Elicitation

In this section we will explore the requirement elicitation process for developing the Custody Database Management System tailored for the Faculty of Science Department of Computer. Requirement elicitation involves gathering, analyzing, and documenting the functional and non-functional requirements of the system. These requirements serve as the foundation for designing and developing the Custody Database Management System.

During the requirement elicitation phase, it is essential to identify the stakeholders who will be impacted by or have a vested interest in the Custody Database Management System. The stakeholders for this system within the Faculty of Computer Science include:

- ***Faculty Members:*** Academic staff who will utilize the system for managing inmate data, monitoring security incidents, and accessing relevant information.

- ***Administrative Staff:*** Personnel from administrative departments within the Faculty responsible for maintaining the custody database, generating reports, and ensuring compliance with regulations and policies.

- ***IT Personnel:*** The technical experts responsible for system development, implementation, and maintenance. They ensure the Custody Database Management System aligns with the university's IT infrastructure, security measures, and technical requirements
- ***Security personnel:*** Custody officers and security staff tasked with overseeing the custody facility and utilizing the system for inmate management, visitor tracking, and incident reporting

2.1.2 Elicitation Techniques

To gather comprehensive requirements for the Custody Database Management System, various elicitation techniques can be employed, including:

- ***Interviews :*** Conducting one-on-one or group interviews with stakeholders to understand their needs, preferences, and challenges related to custody management.

- ***Surveys:*** Distributing surveys to faculty, staff, and security personnel to gather feedback on their requirements and expectations from the custody database system.
- **Workshops:** Facilitating workshops or focus groups to brainstorm ideas, identify key functionalities, and prioritize requirements for the system.

- ***Observation:*** Observing current custody management practices, interactions between staff and inmates, and existing data management procedures to identify pain points and areas for improvement.

- ***Prototyping:*** Developing prototypes or mock-ups of the Custody Database Management System to gather feedback from stakeholders and refine requirements based on user input.

Regarding the current usability, functionality, and user experience of the current custody database management system, the following are the results of the survey:

Usability:

- 62% of the respondents were dissatisfied.
- 9.4% were neither satisfied nor dissatisfied.
- only 28.6% expressed satisfaction.
- The primary areas of dissatisfaction were related to the layout, navigation, and design of the system.
- Respondents frequently cited issues with finding information, cumbersome navigation, and technical difficulties.

Functionality:

- 50.6% of respondents were dissatisfied with the functionality of the current custody database management system.
- 49.4% were satisfied.
- Primary areas of dissatisfaction included the speed of the system, difficulties with managing inmate records, and the lack of automation in certain processes.
- Respondents highlighted challenges with data retrieval, reporting, and integration with other systems.

User Experience:

- 50.9% of respondents were dissatisfied with their overall user experience with the current custody database management system.
- 4.9% were neither satisfied nor dissatisfied
- 39.4% expressed satisfaction.
- Primary areas of dissatisfaction mirrored those in the usability and functionality sections, including the system's layout and navigation, inefficiencies in the custody management processes, and the lack of automation.

Analysis :

Based on the survey results, it is evident that the current custody database management system requires significant improvements in usability, functionality, and user experience. Users face challenges in navigating the system, accessing information efficiently, and completing tasks effectively. The

new system design should prioritize addressing these issues and focus on the following key areas:

- Enhancing the user interface to improve navigation and streamline workflows.
- Improving system responsiveness and speed to increase efficiency in managing custody-related tasks.
- Introducing additional functionality to automate processes such as inmate record management, incident reporting, and data analysis.
- Incorporating user feedback and best practices in system design to ensure a more intuitive and user-friendly experience.

2.1.3 Documentation Process

To ensure a clear and organized representation of the elicited requirements, thorough documentation was maintained. The documentation includes both *functional and non-functional requirements* and some examples *of use cases* and *scenarios* of custody database management system.

Functional Requirements (FRs): define the specific functionalities and features that custody database management system should possess.

Non-functional Requirements (NFRs)*:* Non-functional requirements focus on the qualities and characteristics of the custody database management system. These requirements ensure that the system meets desired standards and provides an optimal user experience

The FRs & NFRs for our system will be provided in detail next section: “2.3Requirements Documentation.”

Use Cases and Scenarios are powerful tools for capturing and analyzing system requirements. Use cases and scenarios provide detailed insights into how users interact with the custody database management system and help identify essential functionalities needed to achieve desired outcomes. Examples of *use cases & scenarios* include :

Use Cases:

1. **Device Assignment:**

- **Description:** his use case involves assigning a device (computer or laptop) to a staff member within the computer science department.
- Actors : Administrative Staff.
- steps:

- The administrative staff member logs into the custody database management system.
- They navigate to the device management module.
- The staff member selects an available device from the inventory.
- They assign the device to a specific staff member by entering their information or scanning their ID.
- The system updates the device record to indicate the assigned user and generates a unique QR code for the device.

2.Device Location Management:

- Description: This use case involves managing the location of devices within the computer science department.
- Actors: Administrative Staff
- Steps:
- The administrative staff member logs into the custody database management system.
- They select a device whose location needs to be updated.
- The staff member scans the QR code of the device or searches for it within the system.
- They update the device's location by selecting the appropriate department, lab, or room from a predefined list.
- The system records the updated location information and updates the device's status accordingly.

3. Device Maintenance and Repair:

- **Description:** This use case involves reporting and resolving maintenance issues for devices.
- Actors: Administrative Staff.
- steps:
- The technician or administrative staff member identifies a malfunctioning device.
- They log into the custody database management system and navigate to the maintenance module.

- The staff member submits a maintenance request, providing details about the issue and the affected device.
- The system generates a maintenance ticket and assigns it to an available technician for resolution.
- The technician receives the maintenance ticket, inspects the device, performs necessary repairs, and updates the ticket status in the system.
- Once the issue is resolved, the system notifies the requester and updates the device's status to indicate that it is operational.

4.Device Decommissioning:

- Description: This use case involves decommissioning devices that are no longer in use.
- Actors: Administrative Staff
- Steps:
- The administrative staff member identifies a device that is no longer needed or has reached the end of its lifecycle.
- hey log into the custody database management system and select the device for decommissioning.
- The staff member updates the device status to indicate that it is decommissioned or ready for disposal.
- The system removes the device from active inventory and archives its record for historical purposes.

4.Device Transfer or Reassignment:

- Description: This use case involves transferring devices from one user to another or reassigning them to different departments.
- Actors: Administrative Staff
- Steps:
- The administrative staff member logs into the custody database management system.
- They select the device to be transferred or reassigned.
- The staff member updates the device record to indicate the new assigned user or department.
- The system generates a notification to inform both the previous and new users of the device transfer.

- If necessary, the staff member physically relocates the device to its new location and updates the system accordingly.

Scenarios:

- Description:

Scenarios for Custody Database Management System:

- Steps:

1. **Successful Device Assignment:**

- Description: This scenario illustrates the successful assignment of a device to a staff member within the computer science department using the custody database management system.
- Actions:
- An administrative staff member logs into the custody database management system using their credentials.
- They navigate to the device management module to view available devices.
- The staff member selects an available computer or laptop from the inventory.
- They assign the device to a specific staff member by scanning their ID or entering their information into the system.
- The system updates the device record to indicate the assigned user and generates a unique QR code for the device

2. **Device Location Update:**

- Description**:** This scenario involves updating the location of a device within the computer science department.
- Actions:
- An administrative staff member logs into the custody database management system.
- They select a device whose location needs to be updated.
- The staff member scans the QR code of the device or searches for it within the system.

- They update the device's location by selecting the appropriate department, lab, or room from a predefined list.
- The system records the updated location information and updates the device's status accordingly.

3. **Device Maintenance and Repair**

- Description**:** This scenario depicts the process of reporting and resolving maintenance issues for a device
- Actions:
- A lab technician or administrative staff member identifies a malfunctioning device within the computer science department.
- They log into the custody database management system and navigate to the maintenance module.
- The staff member submits a maintenance request, providing details about the issue and the affected device.
- The system generates a maintenance ticket and assigns it to an available technician for resolution.
- The technician receives the maintenance ticket, inspects the device, performs necessary repairs, and updates the ticket status in the system.
- Once the issue is resolved, the system notifies the requester and updates the device's status to indicate that it is operational.

4. **Device Decommissioning**

- Description**:**

This scenario involves removing a device from active use within the computer science department.

- Actions:
- An administrative staff member identifies a device that is no longer in use or has reached the end of its lifecycle.
- They log into the custody database management system and select the device for decommissioning.
- The staff member updates the device status to indicate that it is decommissioned or ready for disposal.
- The system removes the device from active inventory and archives its record for historical purposes.

- If necessary, the staff member initiates the disposal process or transfers the device to a designated storage area.

5.Device Transfer or Reassignment:

- Description**:** This scenario involves transferring a device from one staff member to another or reassigning it to a different department.
- Actions:
- An administrative staff member logs into the custody database management system and selects the device to be transferred.
- They update the device record to indicate the new assigned user or department.
- The system generates a notification to inform both the previous and new users of the device transfer.

- If necessary, the staff member physically relocates the device to its new location and updates the system accordingly.
- The system records the transfer details, including the date, time, and reason for the transfer, for audit and tracking purposes.

2.2 Requirements Documentation

This section provides a structured overview of the functional, and non-functional requirements and acceptance criteria that define a system's expected behaviours and characteristics. It serves as a central reference for stakeholders, facilitating effective communication, analysis, and validation of the system's functionalities.

2.2.1 Functional Requirements

The functional requirements for custody database management system include, but are not limited to, the following:

1. **Device Management:**

- The system shall allow administrative staff to add new devices to the database, specifying details such as device type, model, serial number, and initial location.
- The system shall support the assignment of devices to staff members within the computer science department, recording user information and generating unique QR codes for each device.

- The system shall enable administrative staff to update device information, including status, location, and user assignment.
- The system shall provide functionality for marking devices as decommissioned or ready for disposal, archiving their records, and removing them from active inventory.

2. **Location Management:**

- The system shall maintain a database of locations within the computer science department, including departments, labs, rooms, and storage areas.
- The system shall allow administrative staff to assign devices to specific locations within the department, updating their records accordingly.
- The system shall support the tracking of device movements between locations, recording historical location data for each device.

3. **Maintenance and Repair:**

- The system shall enable staff members to report maintenance issues for devices, providing details such as the nature of the problem and affected device.
- The system shall assign maintenance tickets to available technicians for resolution, tracking the status of each ticket until completion.
- The system shall allow technicians to update ticket statuses, record actions taken, and close tickets upon successful resolution.
- The system shall provide notifications to relevant stakeholders regarding the status of maintenance tickets and device repairs.

4. **User Management:**

- The system shall support user authentication and access control, allowing only authorized personnel to log in and perform specific actions.
- The system shall maintain user profiles for administrative staff, technicians, and other relevant stakeholders, storing information such as name, contact details, and role.

- The system shall provide administrative staff with the ability to create, modify, and deactivate user accounts as needed.

2.2.2 Non-Functional Requirements

The non-functional requirements for custody database management system include, but are not limited to, the following:

1.Performance:

- The system shall be capable of handling a large volume of concurrent users and device transactions without significant degradation in performance.
- The system shall have response times for common operations (e.g., device assignment, maintenance ticket creation) that meet or exceed predefined benchmarks.

2.Usability:

- The system shall have an intuitive user interface that is easy to navigate and use, minimizing the need for extensive training.
- The system shall provide clear and concise error messages to users in the event of invalid inputs or system failures.
- The system shall support multiple languages and accessibility features to accommodate diverse user needs.

3.Security Requirements:

- The system shall implement robust security measures to protect sensitive data and prevent unauthorized access or tampering.
- The system shall encrypt sensitive information such as user credentials, device details, and maintenance records to ensure confidentiality.
- The system shall enforce role-based access control, granting different levels of privileges to users based on their roles and responsibilities.

4.Maintainability Requirements:

- The system should be modular and well-structured to facilitate future enhancements or modifications.
- The system's code should adhere to coding standards and best practices for maintainability.
- The system should include proper documentation, such as user manuals and developer guides, to aid in system maintenance and troubleshooting.

5.Compatibility Requirements:

- The system should be compatible with various devices, browsers, and operating systems to accommodate a diverse user base. It should be responsive and adaptable to different screen sizes.

6.Reliability:

- The system shall be highly available, with uptime exceeding 99% to ensure continuous access for users.
- The system shall regularly back up database records and maintain redundancy to minimize the risk of data loss in the event of hardware or software failures.

2.2.3 Acceptance Criteria

Acceptance criteria are essential guidelines used to evaluate whether the system meets its requirements. They provide measurable benchmarks for assessing the system's compliance and alignment with stakeholder expectations.

1.Device Management:

- Administrative staff can successfully add a new device to the database, and the device details are accurately recorded.
- Devices can be assigned to staff members, and the system generates a unique QR code for each device.
- Device information can be updated, including status, location, and user assignment, with changes reflected in the database.

- Decommissioned devices are archived, and their records are removed from active inventory.

2.Location Management:

- Locations within the department can be added to the database, and device assignments accurately reflect their current locations.
- Devices can be assigned to specific locations, and historical location data is maintained for tracking purposes.
- Movement of devices between locations is recorded, and the system updates device records accordingly.

3.Maintenance and Repair:

- Staff members can report maintenance issues for devices, and tickets are successfully created in the system.
- Maintenance tickets are assigned to available technicians, and their status is updated throughout the resolution process.
- Technicians can record actions taken to resolve maintenance issues, and tickets are closed upon successful resolution.
- Relevant stakeholders receive notifications regarding the status of maintenance tickets and device repairs.

4.User Management:

- User authentication and access control are implemented, allowing only authorized personnel to log in and perform actions.
- Administrative staff can create, modify, and deactivate user accounts as needed, and user profiles are accurately maintained.

5.Usability:

- The user interface is intuitive and easy to navigate, with minimal training required for users to perform common tasks.
- Error messages are clear and concise, providing users with helpful guidance in the event of invalid inputs or system failures.

6.Performance:

- The system can handle a large volume of concurrent users and device transactions without significant degradation in performance.
- Response times for common operations meet or exceed predefined benchmarks, ensuring efficient use of the system.

7.Security:

- Robust security measures are implemented to protect sensitive data and prevent unauthorized access or tampering.
- Sensitive information is encrypted to ensure confidentiality, and role-based access control is enforced to manage user privileges.

8.Reliability:

- The system is highly available, with uptime exceeding 99%, ensuring continuous access for users.
- Database records are regularly backed up, and redundancy measures are in place to minimize the risk of data loss in case of failures.

2.3 Current System Analysis

The current system analysis in custody database management system involves a thorough assessment of the existing student registration processes, technologies, and user experiences. Its goal is to identify areas for improvement and guide the design of a more efficient and user-friendly registration system. The following is what we obtained in each step in the current system analysis process:

1. Manual Data Entry: The system heavily relies on manual data entry methods, where administrative staff maintain records of devices, their assignments, and locations using spreadsheets or paper-based forms. This manual approach is time-consuming, error-prone, and lacks efficiency.

2. Lack of Centralized Database: There is no centralized database or dedicated system for managing custody-related information. Instead, data is dispersed across multiple spreadsheets or documents, leading to inconsistency and difficulty in tracking devices and their movements.

3. Limited Accessibility: Since data is not stored in a centralized system, accessibility to custody-related information is limited. Administrative staff may face challenges in accessing up-to-date information about device assignments, locations, maintenance history, and user details.

4. Inefficient Tracking: Due to the lack of a centralized system, tracking the movement of devices within the department is inefficient. Administrative staff may struggle to identify the current location of devices, leading to delays in device allocation and retrieval.

5. Maintenance Management: Maintenance and repair requests are often managed through informal channels, such as email or verbal communication.

6. Security Concerns: With data dispersed across various documents and spreadsheets, maintaining data security and integrity becomes challenging. There is a risk of unauthorized access to sensitive information, as well as potential data loss or corruption due to manual handling.

7. Scalability Issues: The current system lacks scalability, making it difficult to accommodate the growing number of devices and users within the department. As the faculty expands, the manual processes become increasingly unsustainable and prone to errors.

8. User Experience: Administrative staff may find it challenging to navigate through multiple documents or spreadsheets to retrieve the information they need. This can lead to frustration and inefficiency in performing custody-related tasks

In summary, the current custody management system suffers from inefficiencies, lack of centralized data management, limited accessibility, and scalability issues. There is a clear need for a comprehensive Custody Database Management System to streamline processes, improve data accuracy, enhance accessibility, and ensure better tracking and management of devices within the Faculty of Science Department of computer.

2.4 Assumptions and Constraints

This section outlines the key assumptions and constraints guiding the Custody database management system analysis and design. By identifying these factors, we establish the boundaries, limitations, and expectations that influence the system's development. Let's explore the *assumptions* and *constraints* shaping **CDMS**.

Assumptions:

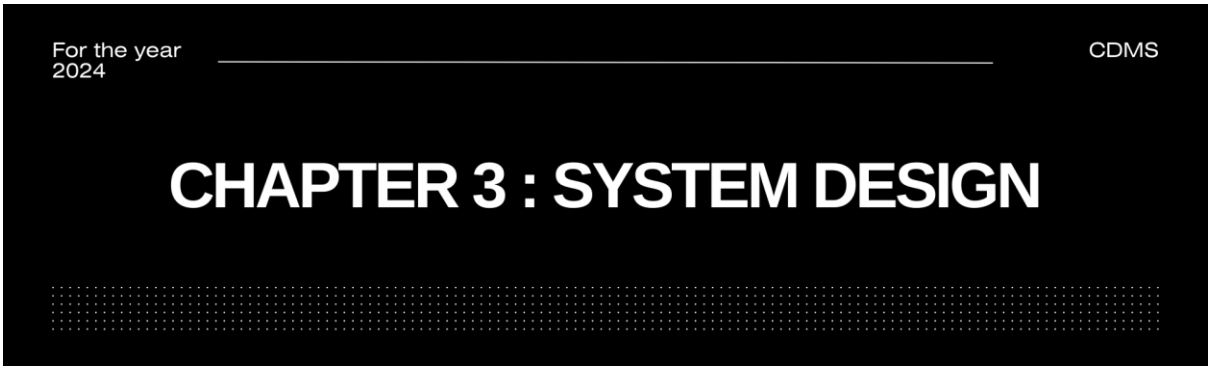
- Users have basic computer literacy skills and internet access.
- Relevant and up-to-date data is available for system analysis.
- Industry best practices and guidelines are followed.
- The system is scalable and flexible to accommodate future growth and changes.

Constraints:

- Technical compatibility with existing infrastructure.
- Budget and resource limitations.
- Regulatory compliance requirements.
- Adherence to the project timeline.
- Interoperability and integration with other systems.

Summary:

System Analysis delves into the analysis phase of the project, aiming to gather requirements, evaluate the existing system, and identify any assumptions and constraints. It comprises four key sections: Requirement Elicitation, Requirement Documentation, Current System Analysis, and Assumptions and Constraints. Through these sections, we gain insights into user needs, define system requirements, assess the strengths and weaknesses of the current system, and identify any limitations or dependencies. This analysis sets the foundation for designing an improved and efficient registration system.



The System Design chapter in the documentation of **CDMS** focuses on the design aspects of the system. It builds upon the analysis conducted in the previous chapter and translates the requirements into a well-defined and efficient system design. The key sections covered in this chapter are:

1. System Architecture Design
2. Data Design
3. User Interface Design
4. Database Design

These sections provide a detailed analysis and design of the system's architecture, data storage and management, user interface, and database structure. The chapter aims to provide a comprehensive understanding of how the various components of the system work together to achieve the desired functionality and user experience.

3.1 System Architecture Design

The system architecture of **CDMS** is designed to ensure scalability, modularity, and maintainability. It follows an API architecture consisting of server side and client side.

3.1.1 System Overview

CDMS comprises the following components:

In the context of .NET, API architecture typically refers to the design principles and patterns used to develop Application Programming Interfaces (APIs) using the .NET framework. Here's a brief overview:

1. RESTful APIs:

- Representational State Transfer (REST) is a popular architectural style for designing networked applications. RESTful APIs in .NET adhere to the principles of REST, which include using standard HTTP methods (GET, POST, PUT, DELETE) for performing operations on resources.

- ASP.NET Web API is a framework for building RESTful APIs using .NET. It provides features for routing, content negotiation, model binding, and serialization, making it easy to create HTTP services that can be consumed by a variety of clients.

2. ASP.NET Core Web API:

- ASP.NET Core is a cross-platform, open-source framework for building modern web applications and services. ASP.NET Core Web API allows developers to build lightweight, high-performance APIs that can run on Windows, Linux, and macOS.

- It provides built-in support for features like dependency injection, middleware, routing, and model binding, enabling developers to create scalable and maintainable APIs.

3. gRPC:

- gRPC is a high-performance RPC (Remote Procedure Call) framework developed by Google. It uses HTTP/2 for transport and Protocol Buffers (protobuf) for serialization, offering efficient communication between services.

- .NET Core supports building gRPC services and clients, allowing developers to create APIs with strong typing, bi-directional streaming, and support for multiple programming languages.

4. Swagger/OpenAPI:

- Swagger/OpenAPI is a specification for describing RESTful APIs using a JSON or YAML format. It provides a standardized way to document APIs, including endpoints, parameters, responses, and authentication methods.

- .NET frameworks like ASP.NET Core provide built-in support for generating Swagger/OpenAPI documentation for APIs, making it easier for developers to document and consume APIs.

5. Authentication and Authorization:

- .NET provides various mechanisms for securing APIs, including JSON Web Tokens (JWT), OAuth, and IdentityServer. These mechanisms allow developers to authenticate and authorize users accessing the APIs, ensuring data security and privacy.

- ASP.NET Core Identity provides features for managing user authentication, roles, and claims, which can be integrated with APIs to enforce access control policies.

6. Versioning and Versioning Strategies:

- APIs often require versioning to support backward compatibility and evolution over time. .NET provides various versioning strategies, including URI versioning, query string versioning, and header versioning.

- Libraries like Microsoft.AspNetCore.Mvc.Versioning can be used with ASP.NET Core Web API to implement versioning strategies and manage multiple versions of APIs.

Overall, .NET provides a robust platform for building APIs, offering flexibility, performance, and scalability to meet the requirements of modern applications and services.

3.1.2 Class Diagram

The Class Diagram is a graphical representation used in object-oriented programming to illustrate the structure and behaviour of a system or application. It showcases the classes, objects, and their relationships, providing a visual depiction of how they interact with each other.

Class diagrams are widely utilized in software engineering as a communication tool to convey system design to developers, stakeholders, and team members. They include class names, attributes, methods, and relationships such as association, inheritance, composition, and aggregation. Class diagrams play a crucial role in documenting and understanding the design of a system concisely and intuitively.

In Figure 1, the class diagram for **CDMS** has the following main classes and their respective **subclasses** with their attributes, methods, and each entry data type.

1. Authentication & User Management System

- **User Roles**: The application defines different user roles:
  - **Admin**: Responsible for managing user accounts, granting permissions, and overseeing the entire system.
  - **Regular Users**: Access the application with limited privileges.
- **Registration & Login**:
  - Only the admin can register new users.
  - Users log in using their credentials (username and password).
  - Upon successful login, the system generates an **access token** valid for **14 days**.
- **Password Constraints**:
  - The application enforces password policies (e.g., minimum length, complexity) to enhance security.

2. Main View & Navigation System

- **Search Functionality**:
  - Users can search for stock items based on various parameters:
    - **Name**: Search by item name.
    - **Username**: Filter by the user who manages the stock.
    - **Location Name**: Find items stored in specific locations.
    - **Category Name**: Categorize items (e.g., electronics, office supplies).
    - **Device ID**: Locate items by their unique identifiers.
- **User-Friendly Interface**:
  - The main view provides an intuitive interface for users to navigate through the application.
  - Clear menus and organized sections enhance usability.

3. Backup & Logging System

- **Database Backup**:
  - Regular backups ensure data integrity and disaster recovery.
  - Admins can schedule automated backups or trigger them manually.
- **Logging**:
  - The system logs all interactions:
    - **Client-Side Requests**: User queries, searches, and actions.
    - **Server-Side Responses**: System behaviour, errors, and updates.
  - Logs are essential for auditing, troubleshooting, and monitoring system performance.
  - Admins can access the log files.

4. Device Specifications

- This system manages additional information related to stock items:
  - **Device Type**: Categorization (e.g., computers, peripherals, lab equipment).
  - **Model**: Specific model or version.
  - **Serial Number**: Unique identifier for each item.
  - **Specifications**: Technical details (e.g., RAM, storage capacity).

5. Other Systems Mentioned

- **Namefigation System**:
  - Unfortunately, the document does not provide detailed information about this system.
  - It might relate to naming conventions, labelling, or categorization within the stock.
- **Cloud Domain**:
  - Mentioned briefly but not elaborated upon.
  - Could refer to cloud-based storage or hosting.
- **Back-End Layer Documentation**:
  - While mentioned, specifics about the back-end layer are not explicitly described

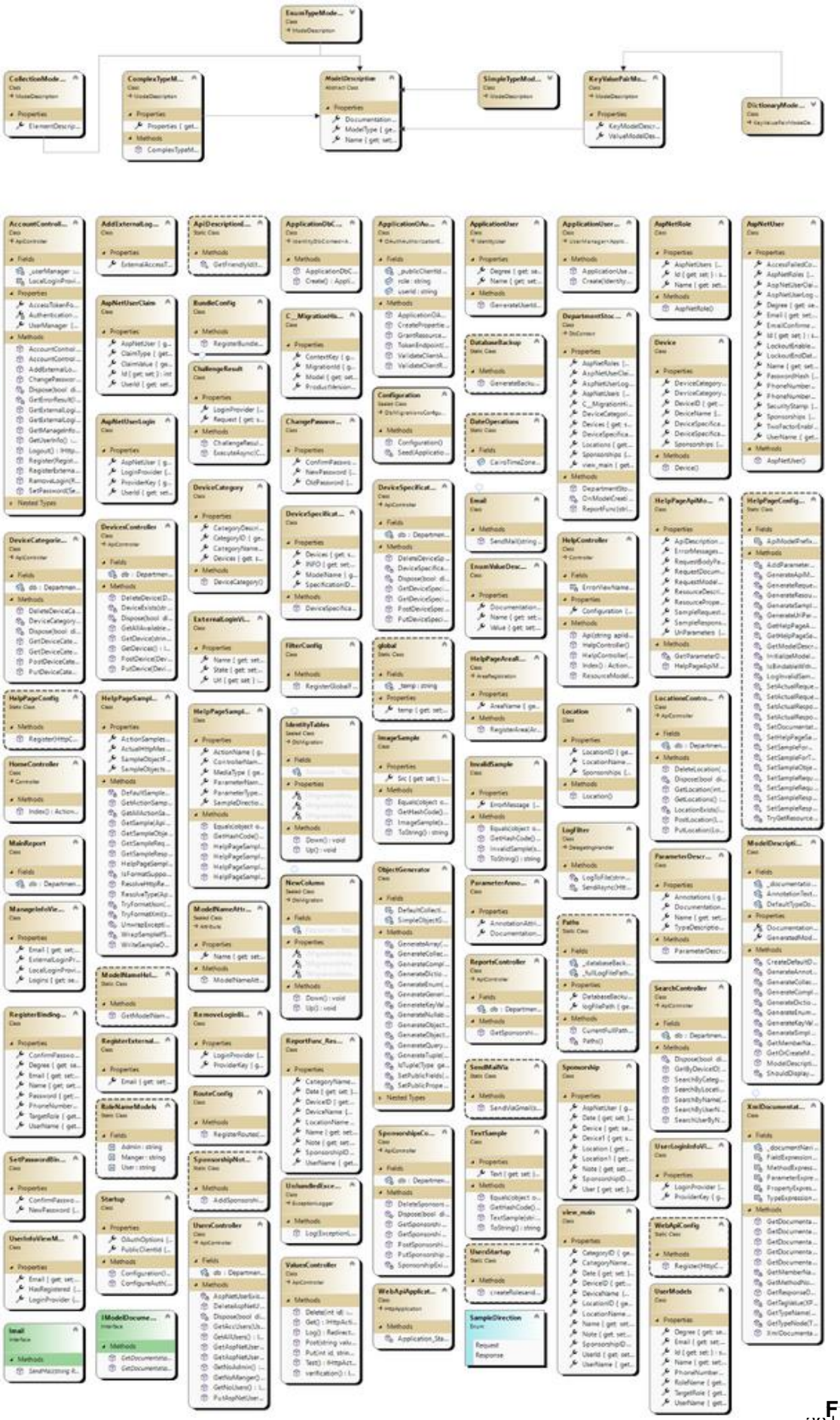


Fig 1

3.2 API Endpoints and app flow

1. Register

- Method: POST
- Link: api/Account/Register
- Parameters: UserName, Name, Email, PhoneNumber, TargetRole, Degree, Password, ConfirmPassword
- Body Method: x-www-form-urlencoded
- Authentication: None
- Default Response Code: Ok - 200
- Notes: Allows a user to register an account.

2. Login (get access token)

- Method: GET
- Link: /token
- Parameters: UserName, Password, grant_type
- Body Method: x-www-form-urlencoded
- Authentication: None
- Default Response Code: Ok - 200
- Notes: Retrieves an access token for authentication purposes.

3. Change Password

- Method: POST
- Link: api/Account/ChangePassword
- Parameters: OldPassword, NewPassword, ConfirmPassword
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Allows a user to change their account password.

4. Get No All Users

- Method: GET
- Link: api/Users/NoOfAllUsers
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves the total number of all users.

5. Get No of Users

- Method: GET
- Link: api/Users/NoUsers
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves the total number of users.

6. Get No of Managers

- Method: GET
- Link: api/Users/NoManger
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves the total number of managers.

7. Get No of Admins

- Method: GET
- Link: api/Users/NoAdmin
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves the total number of admins.

8. Get Full info Account of Users By Role Name

- Method: POST
- Link: api/Users/FullAccountUsers
- Parameters: RoleName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves the full account information of users based on the role name.

9. Get All Users

- Method: GET
- Link: api/Users/AllAccountUsers
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves all user accounts.

10. Get UserById

- Method: GET
- Link: api/Users/SingleAccountUser
- Parameters: Id
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a user account based on the user ID.

11. Edit User

- Method: PUT
- Link: api/Users/EditAccountUsers
- Parameters: UserName, Name, Email, PhoneNumber, Degree
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: NoContent - 204
- Notes: Edits a user account.

12. Delete User

- Method: DELETE
- Link: api/Users/DeleteAccountUsers
- Parameters: Id
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Accepted - 202
- Notes: Deletes a user account.

13. Get All Locations

- Method: GET
- Link: api/AllLocations
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves all locations.

14. Get Location by Id

- Method: GET
- Link: api/SingleLocation
- Parameters: Id
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a location based on the location ID.

15. Add New Location

- Method: POST
- Link: api/AddLocation
- Parameters: LocationName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Created - 201
- Notes: Adds a new location.

16. Edit Location

- Method: PUT
- Link: api/EditLocation
- Parameters: LocationID, LocationName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: NoContent - 204
- Notes: Edits a location.

17. Delete Location

- Method: DELETE
- Link: api/DeleteLocation
- Parameters: LocationID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Accepted - 202
- Notes: Deletes a location.

18. Get All Categories

- Method: GET
- Link: api/AllDeviceCategories
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves all device categories.

19. Get Category by Id

- Method: GET
- Link: api/SingleDeviceCategory
- Parameters: CategoryID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a device category based on the category ID.

20. Add New Category

- Method: POST
- Link: api/AddDeviceCategory
- Parameters: CategoryName, CategoryDescription
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Created - 201
- Notes: Adds a new device category.

21. Edit Category

- Method: PUT
- Link: api/EditDeviceCategory
- Parameters: CategoryID, CategoryName, CategoryDescription
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: NoContent - 204
- Notes: Edits a device category.

22. Delete Category

- Method: DELETE
- Link: api/DeleteDeviceCategory
- Parameters: CategoryID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Accepted - 202
- Notes: Deletes a device category.

23. Get All Device Specifications

- Method: GET
- Link: api/AllDeviceSpecifications
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves all device specifications.

24. Get Device Specification by Id

- Method: GET
- Link: api/SingleDeviceSpecification
- Parameters: SpecificationID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a device specification based on the specification ID.

25. Add New Device Specification

- Method: POST
- Link: api/AddDeviceSpecification
- Parameters: ModelName, INFO
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Created - 201
- Notes: Adds a new device specification.

26. Edit Device Specification

- Method: PUT
- Link: api/EditDeviceSpecification
- Parameters: SpecificationID, ModelName, INFO
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: NoContent - 204
- Notes: Edits a device specification.

27. Delete Device Specification

- Method: DELETE
- Link: api/DeleteDeviceSpecification
- Parameters: SpecificationID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Accepted - 202
- Notes: Deletes a device specification.

28. Get All Devices

- Method: GET
- Link: api/AllDevice
- Parameters: None
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves all devices.

29. Get Device by Id

- Method: GET
- Link: api/SingleDevice
- Parameters: DeviceID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a device based on the device ID.

30. Add New Device

- Method: POST
- Link: api/AddDevice
- Parameters: DeviceName, SpecificationID, CategoryID, LocationID
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Created - 201
- Notes: Adds a new device.

31. Edit Device

- Method: PUT

- Link: api/EditDevice

- Parameters: DeviceID, DeviceName, SpecificationID, CategoryID, LocationID

- Body Method: x-www-form-urlencoded

- Authentication: Bearer token

- Default Response Code: NoContent - 204

- Notes: Edits a device.

32. Delete Device

- Method: DELETE

- Link: api/DeleteDevice

- Parameters: DeviceID

- Body Method: None

- Authentication: Bearer token

- Default Response Code: Accepted - 202

- Notes: Deletes a device.

33. Get All Sponsorships

- Method: GET

- Link: api/AllSponsorships

- Parameters: None

- Body Method: None

- Authentication: Bearer token

- Default Response Code: Ok - 200

- Notes: Retrieves all sponsorships.

34. Get Single Sponsorship

- Method: GET
- Link: api/SingleSponsorship
- Parameters: SponsorshipID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Retrieves a sponsorship based on the sponsorship ID.

35. Add New Sponsorship

- Method: POST
- Link: api/AddSponsorship
- Parameters: SponsorName, SponsorDescription
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Created - 201
- Notes: Adds a new sponsorship.

36. Edit Sponsorship

- Method: PUT
- Link: api/EditSponsorship
- Parameters: SponsorshipID, SponsorName, SponsorDescription
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: NoContent - 204
- Notes: Edits a sponsorship.

37. Delete Sponsorship

- Method: DELETE
- Link: api/DeleteSponsorship
- Parameters: SponsorshipID
- Body Method: None
- Authentication: Bearer token
- Default Response Code: Accepted - 202
- Notes: Deletes a sponsorship.

38. Search Main By Name

- Method: POST
- Link: api/SearchMainByName
- Parameters: Name
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Searches for a main item by name.

39. Search Main By UserName

- Method: POST
- Link: api/SearchMainByUserName
- Parameters: UserName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Searches for a main item by user name.

40. Search Main By LocationName

- Method: POST
- Link: api/SearchMainByLocationName
- Parameters: LocationName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Searches for a main item by location name.

41. Search Main By CategoryName

- Method: POST
- Link: api/SearchMainByCategoryName
- Parameters: CategoryName
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token
- Default Response Code: Ok - 200
- Notes: Searches for a main item by category name.

42. Get Main By DeviceID

- Method: POST
- Link: api/GetMainByDeviceID
- Parameters: DeviceID
- Body Method: x-www-form-urlencoded
- Authentication: Bearer token- Default Response Code: Ok - 200
- Notes: Retrieves a main item based on the device ID.

3.3 Data Design

The data design of **CDMS** focuses on defining the entities, attributes, and relationships necessary for storing and managing users and their devices information.

3.3.1 Data Requirements

The main *entities* identified for **CDMS** include:

1. *DEVICE CATEGORIES*
2. *DEVICES*
3. *DEVICE SPECIFICATION*
4. *USERS*
5. *Admin*
6. *Professor*

These entities represent various aspects of the **CDMS** system, its primary purpose is to streamline stock management, enhance efficiency, and provide accurate information to authorized users.

3.3.2 Entity-Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) is a crucial tool in database design and system analysis. It provides a clear and concise representation of the entities (such as tables or objects), attributes, and relationships within a system or database. By visually illustrating how these components are connected, the ERD helps stakeholders gain a comprehensive understanding of the system's data structure. This diagram serves as a blueprint for database developers, enabling them to design efficient and effective database schemas.

Moreover, the ERD aids in system analysis, allowing stakeholders to identify key entities, their attributes, and the relationships between them. It plays a vital role in ensuring data integrity, facilitating data modelling, and supporting decision-making processes related to system design, optimization, and maintenance.

which represents the ER Diagram for **CDMS** , the mentioned entities are connected through 10 ***relationships***, represented by diamond-shaped symbols, which define how the entities interact and collaborate to achieve specific functionalities. Below are some, not all, examples of the relationships:

a. Device Category - Device: The Device Category table contains categories for different devices, while the Device table stores information about individual devices. The relationship between them is established by the Device Category foreign key in the Device table,

linking each device to its respective category. This relationship allows for organizing devices into categories.

b. Device - Device Specifications: The Device table contains devices, and the Device Specifications table stores specifications for these devices. The relationship between them is established by the Device Specification foreign key in the Device table, connecting each device to its specifications. This relationship enables storing detailed information about each device.

c. Sponsorship - AspNetUsers: The Sponsorship table represents sponsorships where users sponsor devices. The AspNetUsers table stores information about users. The relationship between them is established by the User foreign key in the Sponsorship table, referencing the Id primary key in the AspNetUsers table. This relationship allows associating sponsorships with the users who sponsor them.

d. Sponsorship - Device: The Sponsorship table also has a relationship with the Device table. This relationship is established by the Device foreign key in the Sponsorship table, referencing the Device ID primary key in the Device table. It indicates which device is being sponsored in each sponsorship entry.

e. Sponsorship - Location: Additionally, the Sponsorship table relates to the Location table. The relationship is established by the Location foreign key in the Sponsorship table, referencing the Location ID primary key in the Location table. This relationship signifies where each sponsorship occurred.

These are just a few examples of the relationships present in the ERD of **CDMS** . The complete ERD will provide a comprehensive overview of all the entities and their relationships, facilitating a better understanding of the system's structure and data flow.

3.4 Database Design

The database design of **CDMS** includes the logical schema and tables required for storing and managing data.

3.4.1 Database Schema Design

The logical schema design for the **CDMS** database includes the following ***tables***:

a. **Device Category:**
   i. Attributes: CategoryID (Primary Key), CategoryName, CategoryDescription
   ii. Description: Stores information about different categories of devices.

b. **DeviceSpecifications:**
   i. Attributes: SpecificationID (Primary Key), ModelName, INFO
   ii. Description: Stores specifications for various device models.

c. **Device:**
   i. Attributes: DeviceID (Primary Key), DeviceCategory (Foreign Key), DeviceName, DeviceSpecification (Foreign Key)
   ii. Description: Stores information about devices, including their category, name, and specification.

d. **Sponsorship:**
   i. Attributes: SponsorshipID (Primary Key), User (Foreign Key), Device (Foreign Key), Date, Location (Foreign Key), Note
   ii. Description: Stores sponsorship information, including the user who sponsors, the sponsored device, date, location, and notes.

e. **Location:**
   i. Attributes: LocationID (Primary Key), LocationName
   ii. Description: Stores information about different locations.

f. **AspNetUsers:**
   i. Attributes: Id (Primary Key), Email, EmailConfirmed, PasswordHash, SecurityStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEndDateUtc, LockoutEnabled, AccessFailedCount, UserName, Name, Degree
   ii. Description: Stores user information, including authentication details and additional profile information.

g. **AspNetRoles:**
   i. Attributes: Id (Primary Key), Name
   ii. Description: Stores user roles for authorization purposes.

h. **AspNetUserClaims:**
   i. Attributes: Id (Primary Key), UserId (Foreign Key), ClaimType, ClaimValue
   ii. Description: Stores user claims for additional authentication data.
i. **AspNetUserLogins:**
   i. Attributes: LoginProvider, ProviderKey, UserId (Foreign Key)
   ii. Description: Stores external logins associated with users.
j. **AspNetUserRoles:**
   i. Attributes: UserId (Foreign Key), RoleId (Foreign Key)
   ii. Description: Maps users to roles for authorization.

Each table plays a crucial role in managing different aspects of the **CDMS** system, capturing relevant data, and facilitating efficient data retrieval and manipulation.

3.4.2 Schema Diagram

The Schema Diagram, also known as the Database Schema or Database Design, is a visual representation of a database system's structure, including tables, columns, keys (primary & foreign), and relationships. It is built upon the foundation established in the Entity-Relationship Diagram (ERD) and translates it into a more concrete database implementation.

The Schema Diagram is crucial for developers and administrators to understand and manage the database effectively. It helps in designing a well-organized and efficient database, ensuring data integrity and optimal performance. The schema diagram also serves as a communication tool, facilitating collaboration among stakeholders and informing decision-making in data modelling and application development.

Figure 2, which represents the Schema diagram for **CDMS**

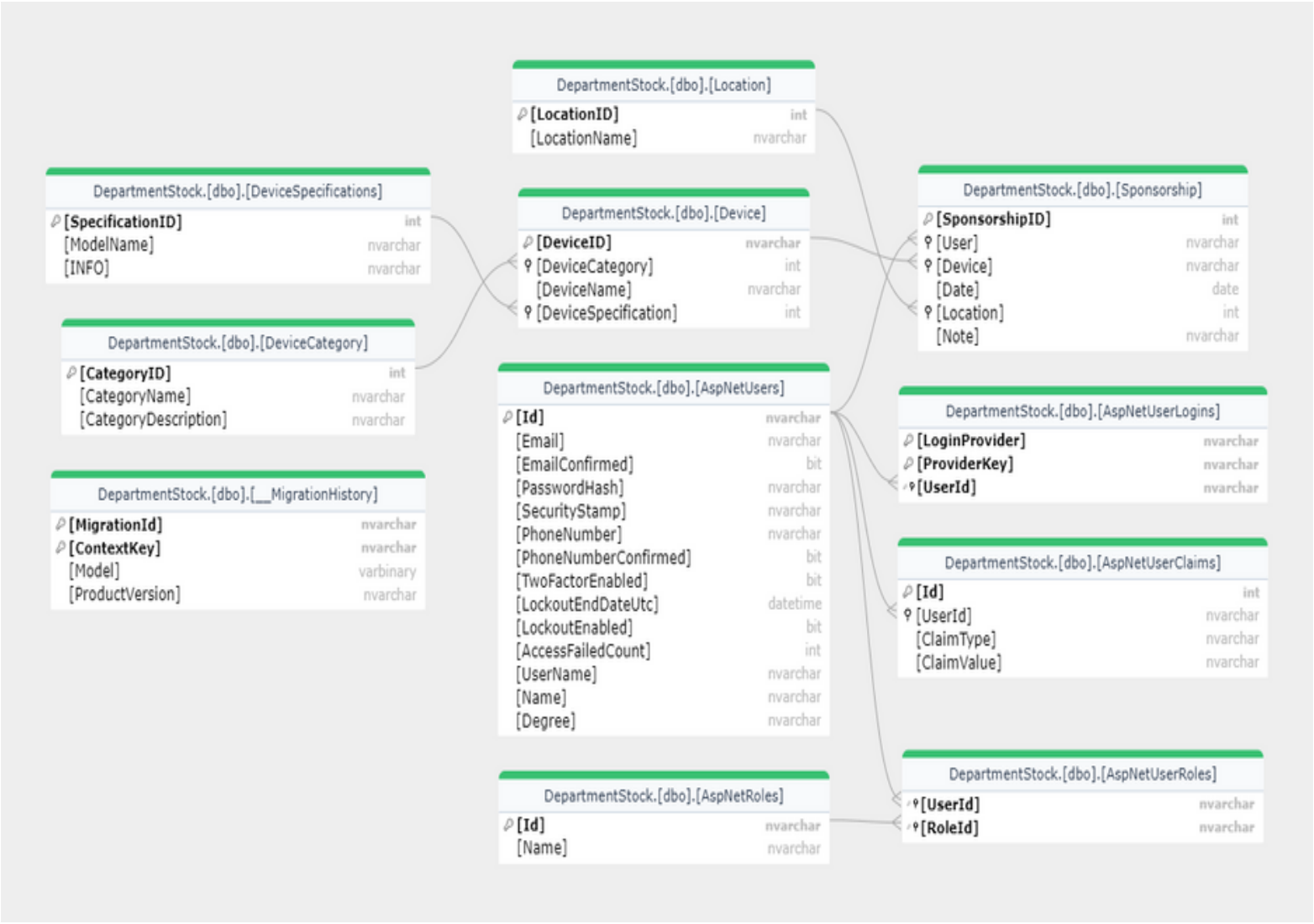


Fig 2

3.5 User Interface Design

The user interface design of **CDMS** aims to provide an intuitive and user-friendly experience for both users and administrative staff.

3.5.1 User Interface Requirements

The user interface should include the following features:

- **Login Page:**
  - The login page serves as the entry point for authorized users.
  - It should include fields for entering credentials such as username and password.

    - Additionally, there could be an option for password recovery or account registration (for new admins)
- **Dashboard:**
    - After successful login, users are directed to the dashboard, which provides an overview of the system.
    - The dashboard displays key metrics and summary information related to custody management, such as the total number of devices in custody, recent activity, and alerts.
    - It may include quick access links or shortcuts to frequently used features.
- **Devices Management:**
    - This section allows admins to manage information related to devices (e.g., computers, laptops).
    - Users can view a list of all devices in custody, along with details such as device ID, model, location, custodian, etc.
    - Admins can add new devices, edit existing device information, or remove devices from custody.
- **Custodians Management:**
    - Here, admins can manage custodian information, including staff members responsible for overseeing custody of devices.
    - Users can view a list of custodians, along with their contact details, assigned devices, etc.
    - Admins can add new custodians, update their information, or remove custodians from the system.
- **Reports:**
    - This section allows users to generate reports on custody-related activities, such as device inventory, custodian assignments, transaction history, etc.
    - Users can customize report parameters, select date ranges, and choose specific criteria for generating reports.
    - Reports can be exported in various formats (e.g., PDF, CSV) for further analysis or sharing.
- **Notifications and Alerts:**
    - The system should provide real-time notifications and alerts to users regarding important events or actions, such as device allocations, custody transfers, low inventory, etc.

- Notifications can be displayed within the interface or sent via email for immediate attention.

- **Accessibility and Responsiveness:**
  - The user interface should be designed to be accessible to users with disabilities, following accessibility standards and guidelines.
  - It should also be responsive, adapting to different screen sizes and devices (e.g., desktops, tablets, mobile phones) for optimal user experience.

3.5.2 Wireframe Screens

The wireframe screens serve as a visual representation of the user interface, illustrating the layout, structure, and functionality of the system's screens. It guided us in creating the actual screen design and obtaining a cohesive system While not exhaustive, the selected wireframes represent the most critical screens that significantly impact the user experience. They offer a glimpse into the overall design direction, user interactions, and information presentation within the system. The wireframes are carefully crafted to align with the system's requirements, ensuring intuitive navigation, clear information hierarchy, and a visually appealing interface.

a. Users Page Wireframe

Creating a wireframe for the Users page of the Custody Database Management System (**CDMS**) that includes features such as Home, Add, and Edit offers several benefits:

1. Visual Representation: Wireframes provide a visual representation of the Users page layout and functionality, allowing stakeholders to understand how different features are organized and accessed.
2. User Flow Planning: By mapping out the flow between the Home, Add, and Edit features, wireframes help designers plan the user journey and ensure a seamless navigation experience for users.
3. Feature Prioritization: Wireframes enable designers to prioritize features based on their importance and frequency of use. For example, the Home feature may be given more prominence, while the Add and Edit features may be accessible through secondary navigation or contextual menus.

Fig 3. Users Page Wireframe Screen

b. **Location**

Designing a wireframe for the Location of Devices web page with features such as Home (showing all IDs with their locations) and options to Add or Edit locations offers several benefits:

- **Clear Visualization:** Wireframes provide a clear visual representation of the layout and functionality of the Location of Devices web page. Stakeholders can easily understand how devices are organized and how they can add, edit, or viedevice locations.

- **Efficient Navigation:** The Home feature, displaying all device IDs with their respective locations, allows users to quickly navigate through the list and find specific devices. This feature helps streamline the process of locating devices within the system.
- **Accessibility**: Providing a centralized Home page listing all device IDs and their locations enhances accessibility for users, as they can access relevant information from a single location without the need for extensive searching.
- **Data Management:** The ability to Add or Edit locations directly from the web page allows administrators or authorized users to efficiently manage

device locations. This feature simplifies the process of updating location information and ensures the accuracy of the data stored in the system.

Fig 4. Location Wireframe Screen

c.**Report**

Designing a wireframe for the Report page in the Custody Database Management System offers several benefits:

- **Customizable Reporting:** The wireframe provides options for customizing reports based on specific criteria such as device type, location, custodian, or time period.
- **Comprehensive Information:** The Report page wireframe includes sections for various types of information, such as device status, custody history, maintenance records, and inventory summaries.

Fig 5. Report Wireframe Screen

3.5.3 Use Case Diagram

A use case diagram is a visual representation that illustrates the interactions between system users (actors) and the **CDMS** system. It provides an overview of the system's functionality and the roles of different actors involved.

From Figure 7, the **CDMS** Use Case Diagram, we can identify the following **actors** and their respective *actions*:

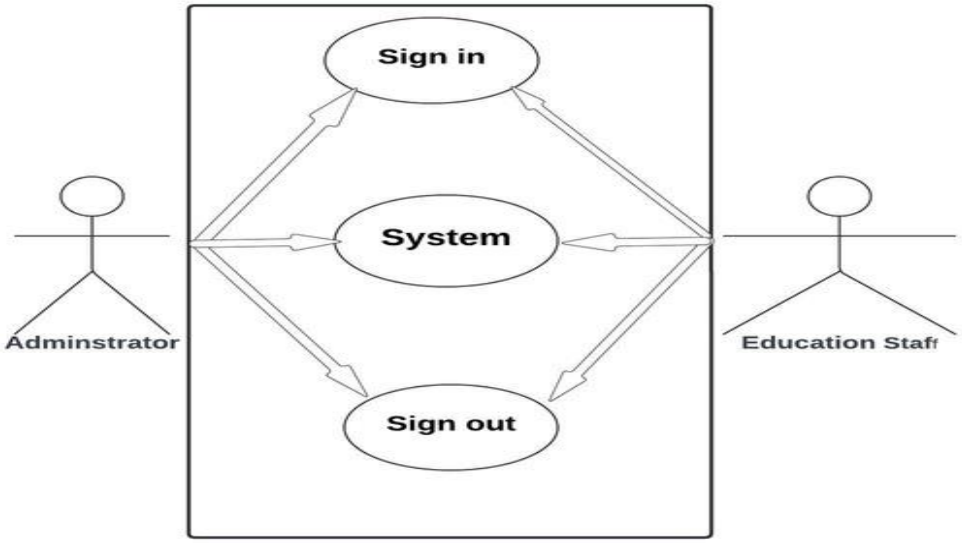


Fig 6. CDMS Use Case

1.Administrator:

The primary user of the **CDMS** system. Responsible for managing user accounts, system settings, and overall system administration.

Actions:

- **Manage User Accounts:** Create, update, and delete user accounts.
- **Configure System Settings:** Adjust system configurations, such as permissions and access controls.
- **Monitor System Activity:** View logs and reports to monitor system usage and performance.

2.Education Staff:

Users responsible for managing custody records, assigning devices, and tracking device locations.

Actions:

- **Add Device:** Enter details for new devices into the system, including device type, serial number, and location.
- **Edit Device Information:** Modify existing device information, such as updating device status or assigning a new custodian.
- **Search for Device:** Locate devices within the database based on specific criteria, such as device type or custodian.
- **Assign Device:** Assign devices to users or locations, updating custody records accordingly.
- **Generate Reports:** Generate reports on device inventory, custody history, maintenance schedules, etc.

3.System:

Represents the **CDMS** itself, which facilitates interactions between users and the database.

Actions:

- **Authenticate User**: Verify user credentials to grant access to the system.
- **Record Actions:** Log user interactions and system events for auditing and tracking purposes.
- **Manage Database:** Handle data storage, retrieval, and manipulation operations to support user actions.

Summary :

The system design chapter focuses on the design aspects of the system, including System Architecture, Data Design, User Interface Design, and Database Design. It provides a comprehensive analysis and design of these components to ensure efficient functionality and a seamless user experience.

The Implementation and Development chapter focuses on the practical aspects of building the **CDMS** system. This chapter covers the following key sections:

1. Development Technologies and Tools
2. Code Implementation
3. Deployment Plan
4. Future Deployment Considerations
5. Gantt Chart

This chapter provides an overview of the technologies and tools used, the code implementation details, a plan for deploying the system, considerations for future deployments, and a visual representation of the project timeline.

4.1 Development Technologies and Tools

The implementation of the **CDMS** involves the utilization of various technologies, frameworks, and tools to create a robust and efficient system. This section provides an overview of the development technologies and tools employed in the project.

4.1.1 Front-end Technologies

The user interface of the **CDMS** system is developed using the following technologies:

Technology	Description
HTML	Structure web pages and define content elements.
CSS	Enhance the visual appearance and layout of web pages.

JavaScript	A programming language that adds interactivity and dynamic functionality to the system's user interface.

Table 1. Front-end Technologies

4.1.2 Back-end Technologies

The back end of the **CDMS** system is powered by the following technologies:

Technology	Description
ASP.NET	ASP.NET is a web development framework for building dynamic web applications and services using the C# programming language. It allows developers to create web pages and web services by leveraging reusable components and libraries provided by Microsoft. With ASP.NET, developers can build interactive websites, web APIs, and web applications that run on the Microsoft .NET platform.
C#	A server-side programming language is used to handle server-side processing, database connectivity, and system logic.
EF	Entity Framework (EF) is an object-relational mapping (ORM) framework in C# that simplifies data access by enabling developers to work with relational databases using object-oriented programming concepts. It provides a set of tools and libraries for creating, querying, updating, and deleting data from a database without needing to write low-level SQL code explicitly
LINQ	LINQ (Language Integrated Query) in C# is a powerful feature that provides a uniform way to query data from different types of data sources such as collections, arrays, databases, XML, and more, directly from within the C# language syntax. LINQ allows developers to write queries in a declarative manner, similar to SQL, which makes it easier to manipulate and retrieve data
SQL Server	A popular relational database management system is utilized for efficient data storage and retrieval for the system.

Table 2. Back-end Technologies

4.1.3 Frameworks

To expedite development and enhance functionality, the following frameworks are incorporated into the system:

Technology	Description
Bootstrap	A popular CSS framework that simplifies responsive web design, making the system adaptable to different devices and screen sizes.
jQuery	Lightweight JavaScript framework for interactive components of the system. It simplifies HTML document traversal, event handling, and animation in the system.

Table 3. Frameworks

4.1.4 Web Server

The system runs on the IIS server, a robust and reliable web server known for its performance, security, and scalability. Therefore, the system is compatible with Windows only

4.1.5 Development Tools

The development of the **CDMS** system involves the use of the following tools:

Tools	Description
IIS	Internet Information Services (IIS) is a web server software developed by Microsoft for hosting websites and web applications on Windows servers. It provides a platform for serving HTTP, HTTPS, FTP, FTPS, SMTP, and NNTP protocols. IIS supports various features including website hosting, application hosting, security configurations, and performance optimization. It is widely used by businesses and organizations for hosting dynamic web content and services on the Windows operating system platform.
Visual Studio	Visual Studio Community is a free, fully-featured integrated development environment (IDE) from Microsoft, popular among developers for building a wide range of applications including web, desktop, mobile, and cloud-based solutions.
Web Browser	Google Chrome, Mozilla Firefox, or Safari are recommended for testing and running the web application

Microsoft Word 365	A word processing application used for creating and editing the system's documentation.
Visual Studio Code	A lightweight and versatile code editor for writing and editing HTML, CSS, JavaScript
React.js	React.js , Node.js و npm (Node Package Manager) , Create React App (CRA), Git ,GitHub .
Node.js	Is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It is built on the Chrome V8 JavaScript engine, which powers Google Chrome, and allows developers to use JavaScript to write server-side scripts for building scalable network applications
npm (Node Package Manager)	is a package manager for Node.js, the popular JavaScript runtime environment. It is used to discover, install, manage, and distribute JavaScript libraries and tools.
Create React App (CRA)	a popular toolchain provided by Facebook for quickly setting up React.js applications with a predefined project structure and build configuration.
Git ,GitHub	Git is a distributed version control system (VCS) designed to track changes in source code during software development. It allows multiple developers to collaborate on projects efficiently by managing revisions, facilitating code review, and enabling the merging of changes from different contributors.

Table 4. Development Tools

These development tools provide the necessary infrastructure and support for the implementation and documentation of the **CDMS**.

4.2 Code Implementation

The codebase of **CDMS** follows a modular and maintainable approach, with a clear separation of concerns and well-structured code files. The main components of the codebase are as follows:

1. **HTML Templates:** The system utilizes HTML templates to define the layout and structure of the various pages, such as the registration form, course listing, and user profile. These templates provide a consistent and

responsive design across different devices. Below are examples of some key templates along with their descriptions:

2. ***Log-in Page***

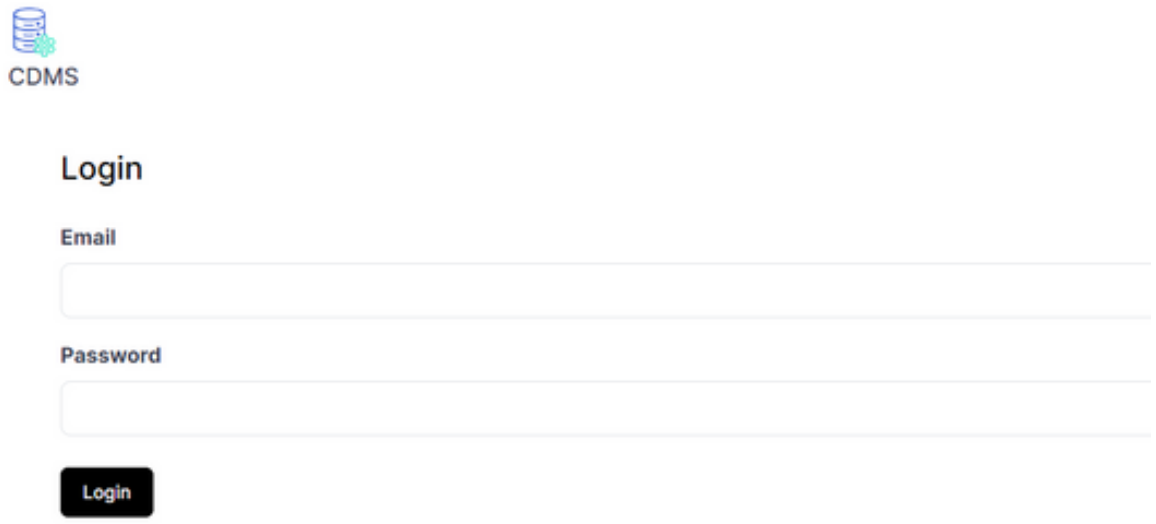


Fig 7. Log-in Page

This layout is designed to provide a straightforward and intuitive interface for users to access a system. It typically consists of a concise form where users can enter their login credentials, username, and password. The layout is minimalistic, focusing on essential elements to ensure a seamless login process. The layout aims to be visually appealing while maintaining a clean and uncluttered design. Its purpose is to facilitate easy access to the system while prioritizing user convenience and security.

3. ***Registration Form Page :***

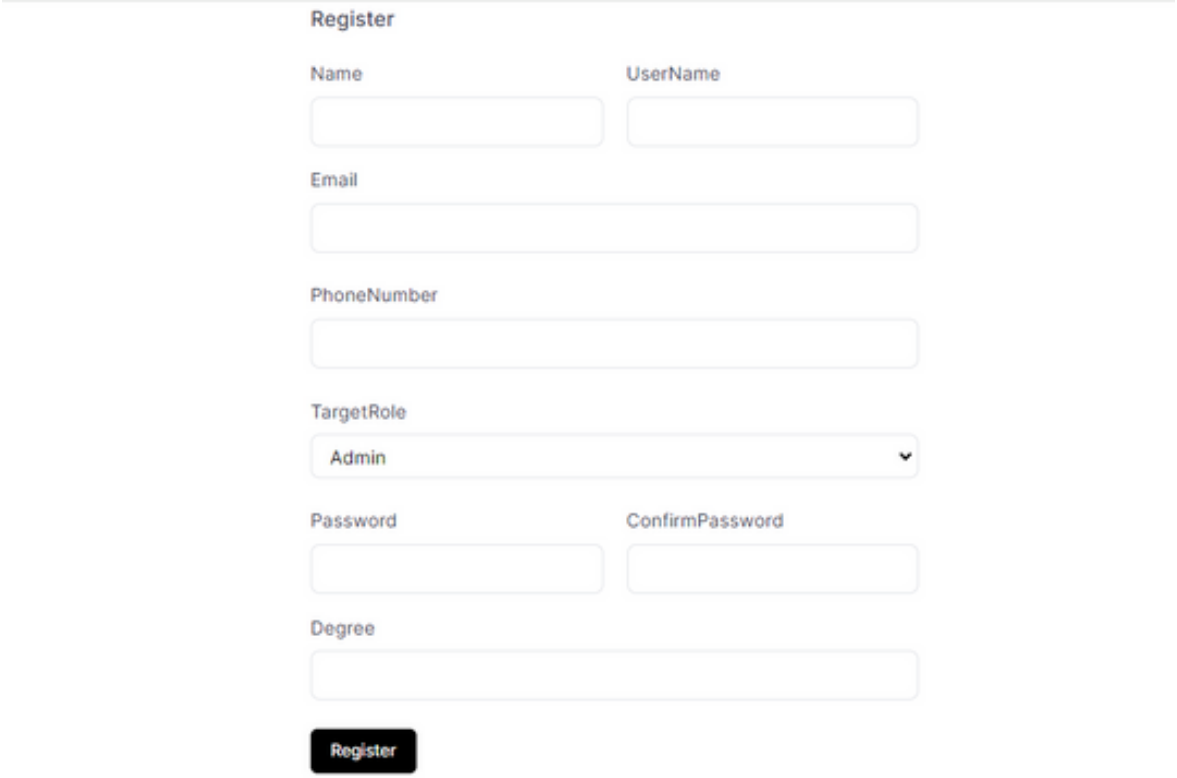


Fig 8. Registration Form Page

This layout is responsible for displaying the form where the admin can enter the registration details of each user. It includes fields such as name, email, password and other relevant information. The screenshot shows the design and structure of the registration form, highlighting the input fields and any additional components such as buttons. This design is responsible only for those who can create an account for those who have the right to access the system, no one can create an account for themselves..

4.*Dashboard Page :*

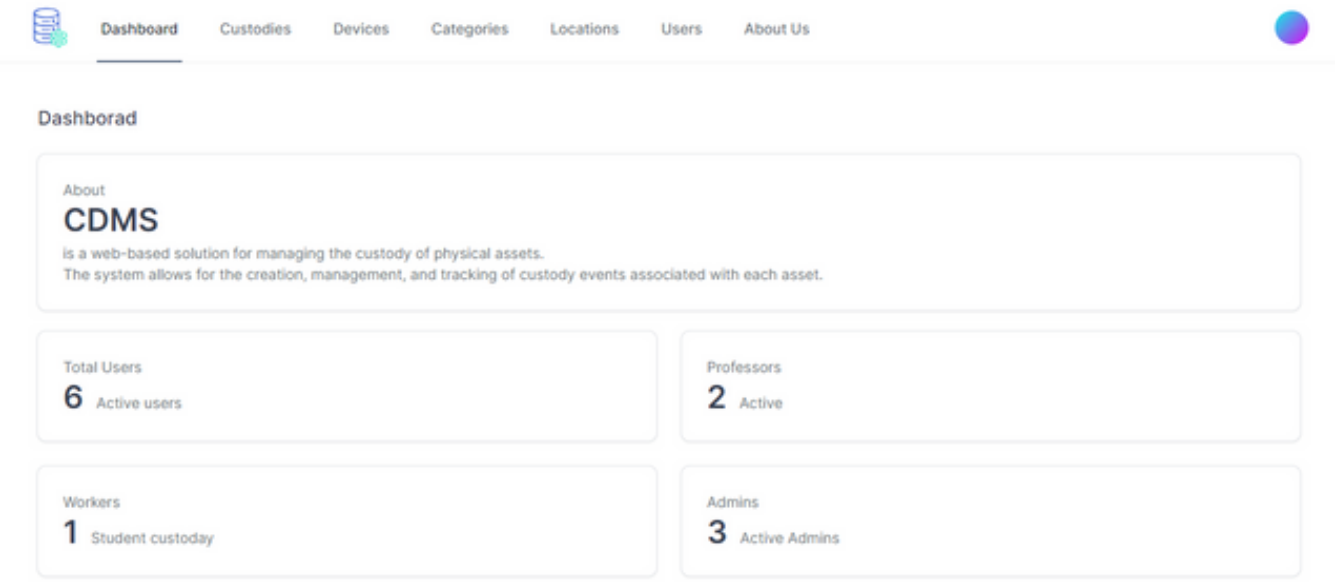


Fig 9. Dashboard Page

This design provides information about what the site actually does, along with the number of users categorized as Admin, Professor, and Worker.

5. *Custodies*

Here in this table, the people responsible for the devices are displayed, when they receive them, the locations of these devices, and the classification of the devices under their responsibility. Also, this data can be modified, deleted, and new ones can be added. Also, any column in the table can be searched for:

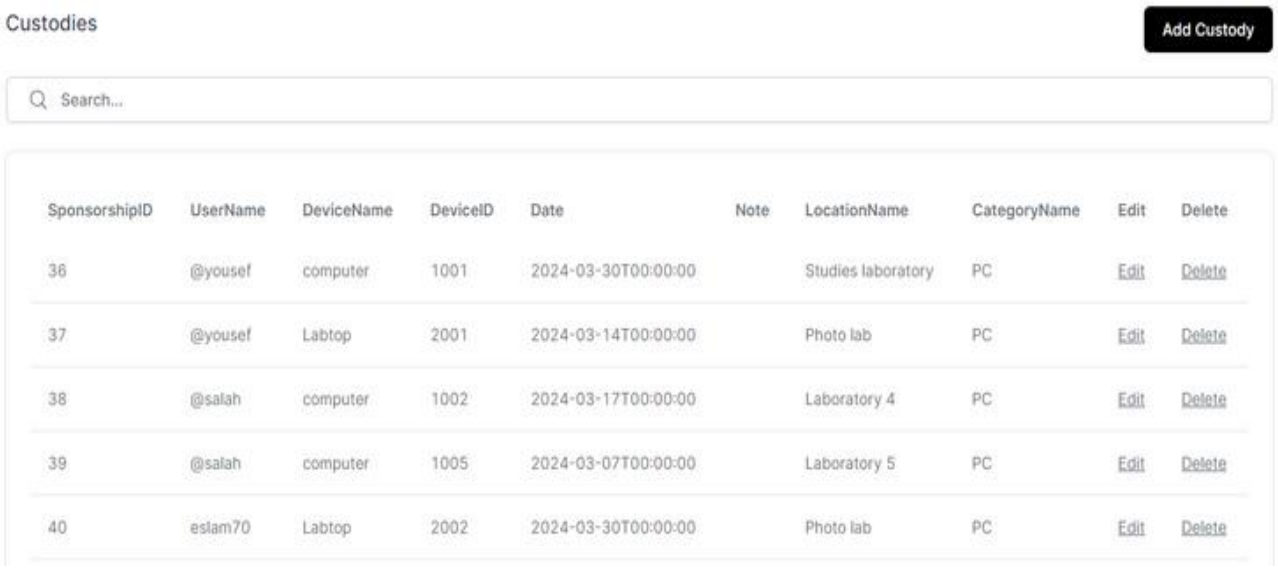


Fig 10.Custodies Page

6. ***Devices:***

This design was designed to upload all the data related to the devices and their types, and each device has an ID. There are also categories of devices, and it is possible to modify, delete, add, and also search:

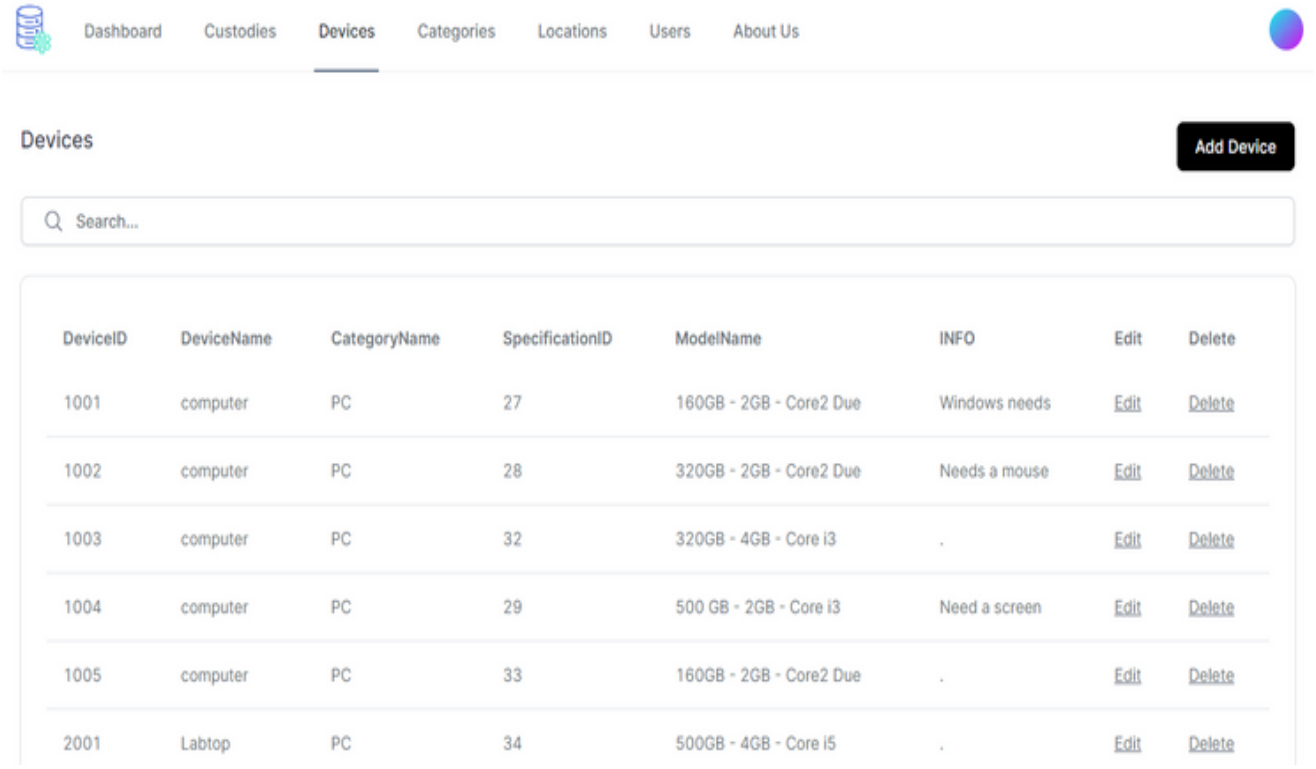


Fig 11.Devices Page

1.6 categories:

This design was designed to classify devices in order to divide them and quickly search for them. Search, modification, and addition are available:

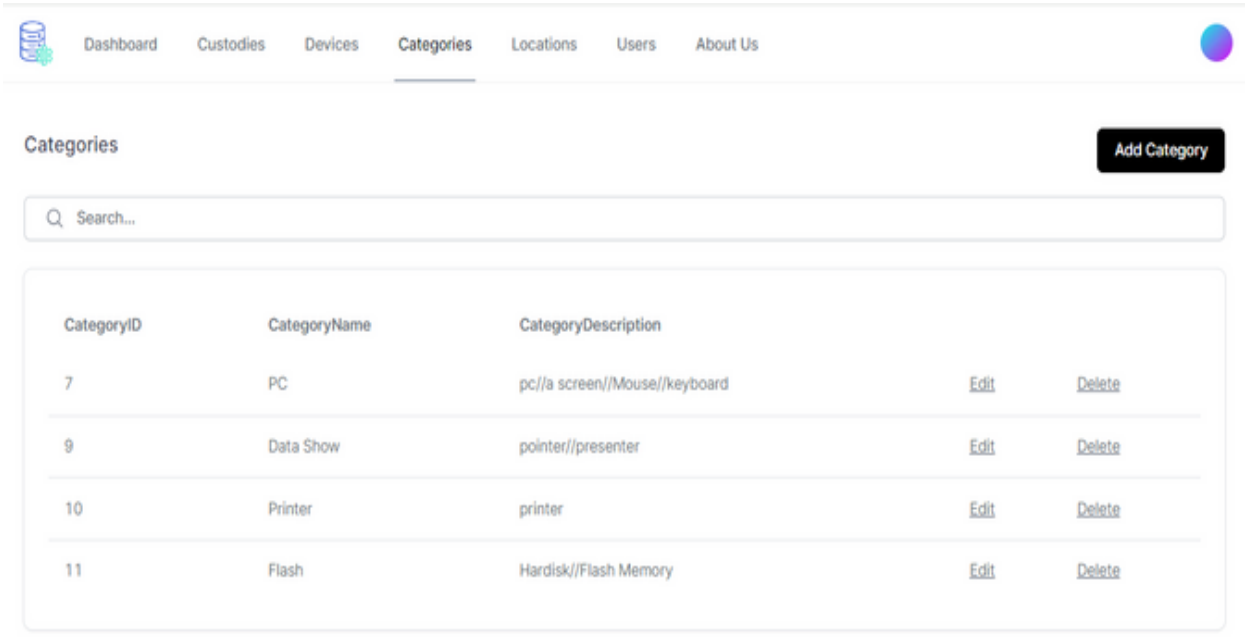


Fig 12.Categories Page

1.7 Locations:

This is designed to display all the places where the devices are located, with the ability to delete, modify, and add:

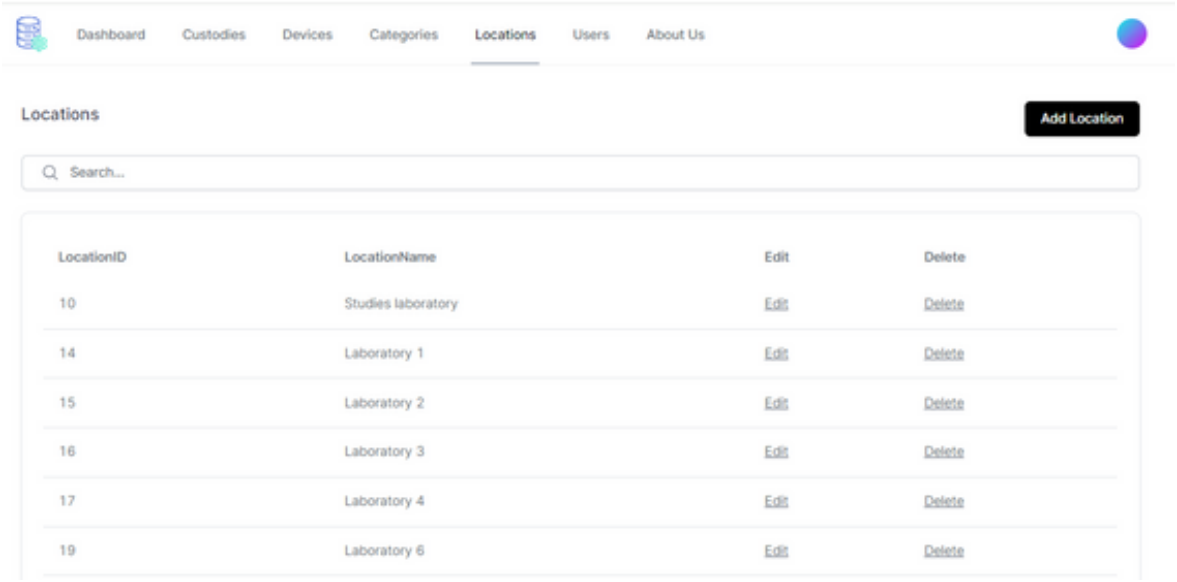


Fig 13.Location Page

1.7 Users:

In this design, there are users who are allowed to interact with the system with all the information about them, with the ability to edit, delete, and also add a new user:

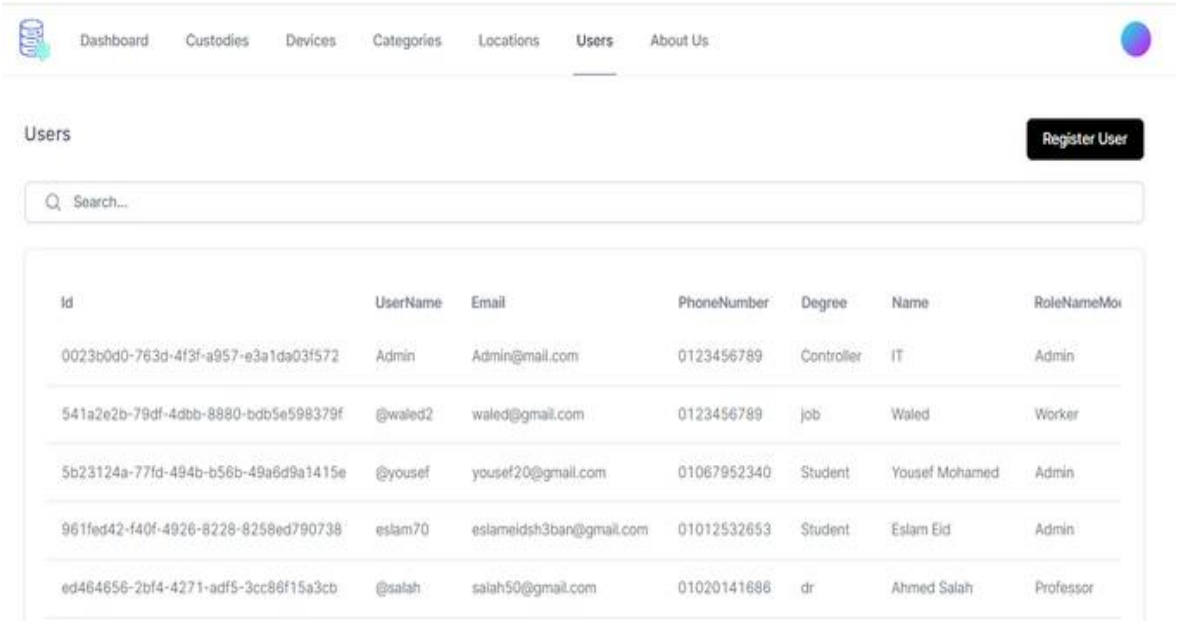


Fig 14. Users Page

1**. CSS Stylesheets:** Custom CSS stylesheets are employed to customize the appearance of the web pages, including colours, fonts, spacing, and responsive layouts. CSS classes and selectors are used to target specific elements and apply styling rules. Here are some implemented snippets of CSS codes:

2. ***Responsive Styles***

This stylesheet contains media queries that define specific styles for different screen sizes. It ensures that the layout and appearance of elements on the web page adapt and respond effectively to various devices. It includes rules

for adjusting the margin-bottom property for certain elements at different breakpoints.

3. ***Table Tools Styles***

The following stylesheet defines styles for a table and its various states, particularly when interacted with or selected. It includes rules for setting the background colour, text colour, and hover effects for active rows and links within the table.

4. ***Style Tweaks***

The following stylesheet contains various style adjustments and modifications. It includes rules for setting the padding of the body element and the footer, as well as defining off-canvas behaviour for smaller screens using media queries. It also includes custom styles for a specific navigation bar with the class **`.navbar-magbanua`**.

5. ***Global Styles***

The following stylesheet defines various global styles for different elements on the web page. It includes rules for setting the margin-bottom property for elements with the class **`.img-portfolio`**, creating a hover effect with reduced opacity for elements with the class **`.img-hover`**, and adjusting the height and dimensions of elements within the **header.carousel** section.

6.**JavaScript Functions:** The JavaScript code enhances the user experience by adding dynamic functionality to the system. This includes form validation, real-time updates, and interactive elements. Event handlers and functions are implemented to handle user interactions and perform necessary actions. Here are some examples of JavaScript functions implemented in CDMS:

7. ***validateForm():***

This function is used for form validation. It retrieves the values of the name and email fields from an HTML form and checks if they are empty. If any of the fields are empty, it displays an alert message and returns `false` to prevent form submission. You can customize the validation logic to suit your specific requirements.

```
const handleLogin = async () => {
    const id = toast.loading("Please wait...", {
      closeButton: true,
    });
    const response = await fetch("/api/login", {
      method: "POST",
      body: new URLSearchParams({
        UserName: email,
        Password: password,
      }),
    });

    const result = await response.json();
    if (response.ok) {
      const data = JSON.parse(result);
      Cookies.set("username", data.userName, { expires: 14 });
      Cookies.set("Role", data.Role, { expires: 14 });
      Cookies.set("UserId", data.UserId, { expires: 14 });
      Cookies.set("access_token", data.access_token, { expires: 14 });
      await router.push("/");
    } else {
      toast.update(id, {
        render: "Wrong email or password",
        type: "error",
        isLoading: false,
        closeButton: true,
      });
    }
```

8. ***updateUser:***

This function is only used for the admin to change information for one user, and no one who uses the system can change information for another

```
const data = {
      Id: id,
      UserName: UserName,
      Name: Name,
      Email: Email,
      PhoneNumber: PhoneNumber,
      TargetRole: TargetRole,
      Degree: Degree,
    };

    const url = "/api/editUser";

    const options = {
      method: "POST",
      body: new URLSearchParams(data),
    };

    const response = await fetch(url, options);
    if (response.status == 400) {
      toast.update(idd, {
        render: "Somthing went wrong",
        type: "error",
        isLoading: false,
        closeButton: true,
      });
    } else {
      toast.update(idd, {
        render: "Updated",
        type: "success",
        isLoading: false,
        closeButton: true,
      });
```

1. **Server-Side Integration:** The **CDMS** system interacts with the server to fetch and update data. Although the server-side implementation details are beyond the scope of this documentation, it is worth noting that the system communicates with the server via APIs to retrieve course information, and user data, and perform registration-related operations.

4.3 Deployment Plan

The deployment plan outlines the step-by-step process to successfully deploy the **CDMS** student registration system. Each step focuses on setting up the server environment, configuring the application, and ensuring proper security measures.

Step	Description
Server Setup	Set up a production server with the necessary hardware and software requirements, including the operating system, web server, and database server.
Configuration	Configure the server environment to meet the system requirements, including installing and configuring the necessary dependencies and libraries.
Database Setup	Set up the MSSQL database and import the necessary database schema. Configure the database connection settings in the application.
Application Deployment	Transfer the application code to the production server. Set up the necessary file permissions and directory structure. Install any required dependencies and libraries.
Security Considerations	Implement security measures, such as using HTTPS for secure communication, applying necessary firewall rules, and ensuring secure database access.
Testing and Quality Assurance	Perform thorough testing to ensure the system functions as expected in the production environment. Conduct load testing to assess system performance under typical usage scenarios.
Data Migration	If applicable, migrate data from the development or testing environment to the production database. Ensure the data is accurately transferred and validated.
User Training	Provide training and documentation to system administrators and end-users to familiarize them with the deployed system and its functionalities.

Monitoring and Maintenance	Set up monitoring tools to track system performance, detect errors, and ensure high availability. Establish a maintenance plan for regular updates, bug fixes, and system enhancements.

Table 5. Deployment Plan

4.4 Future Deployment Considerations

The future deployment considerations focus on scalability, cloud deployment options, continuous integration and deployment, containerization, high availability, backup and disaster recovery, and security enhancements.

Consideration	Description
Scalability	Evaluate the system's ability to handle increased user traffic and data volume. Implement scaling strategies, such as horizontal scaling, to accommodate growing user demands.
Cloud Deployment	Explore the possibility of deploying the system on cloud platforms like AWS, Google Cloud, or Azure. Leverage cloud infrastructure and services to enhance scalability, reliability, and flexibility.
Continuous Integration and Deployment	Implement continuous integration and deployment (CI/CD) pipelines to automate the deployment process. This streamlines development, testing, and deployment cycles, ensuring efficient and error-free releases. *(Note: CI/CD is a software development practice that involves automatically building, testing, and deploying code changes to production environments.)
Containerization	Adopt containerization technologies like Docker and Kubernetes for improved portability, scalability, and resource utilization. Containerization simplifies deployment and facilitates **microservices-based** architectures. *(Note: Docker packages the application and its dependencies into containers. These containers are portable and can be deployed on any machine reducing compatibility issues. Kubernetes for container orchestration, managing and scaling containers in distributed environments.)
High Availability	Implement redundancy and failover mechanisms to ensure the high availability of the system. Explore load-balancing solutions and distribute application instances across multiple servers or regions.
Backup and Disaster Recovery	Establish robust backup and disaster recovery procedures to protect critical data and ensure business continuity. Regularly backup the system data and test the recovery process.

Security Enhancements	Continuously assess and enhance system security to mitigate potential risks and vulnerabilities. Stay updated with the latest security practices and technologies to safeguard sensitive user data.

Table 6. Future Deployment Considerations

By following the Deployment Plan and considering Future Deployment Considerations, the **CDMS** system can be successfully deployed and prepared for future growth and advancements.

4.5 Gantt Chart

A Gantt chart is a visual representation of project tasks and their timelines. It provides a comprehensive overview of the project schedule, including task dependencies, milestones, and resource allocation. The Gantt chart helps in planning, monitoring, and coordinating project activities to ensure timely completion.

Figure 15 represents the Gantt chart for the implementation of **CDMS**.

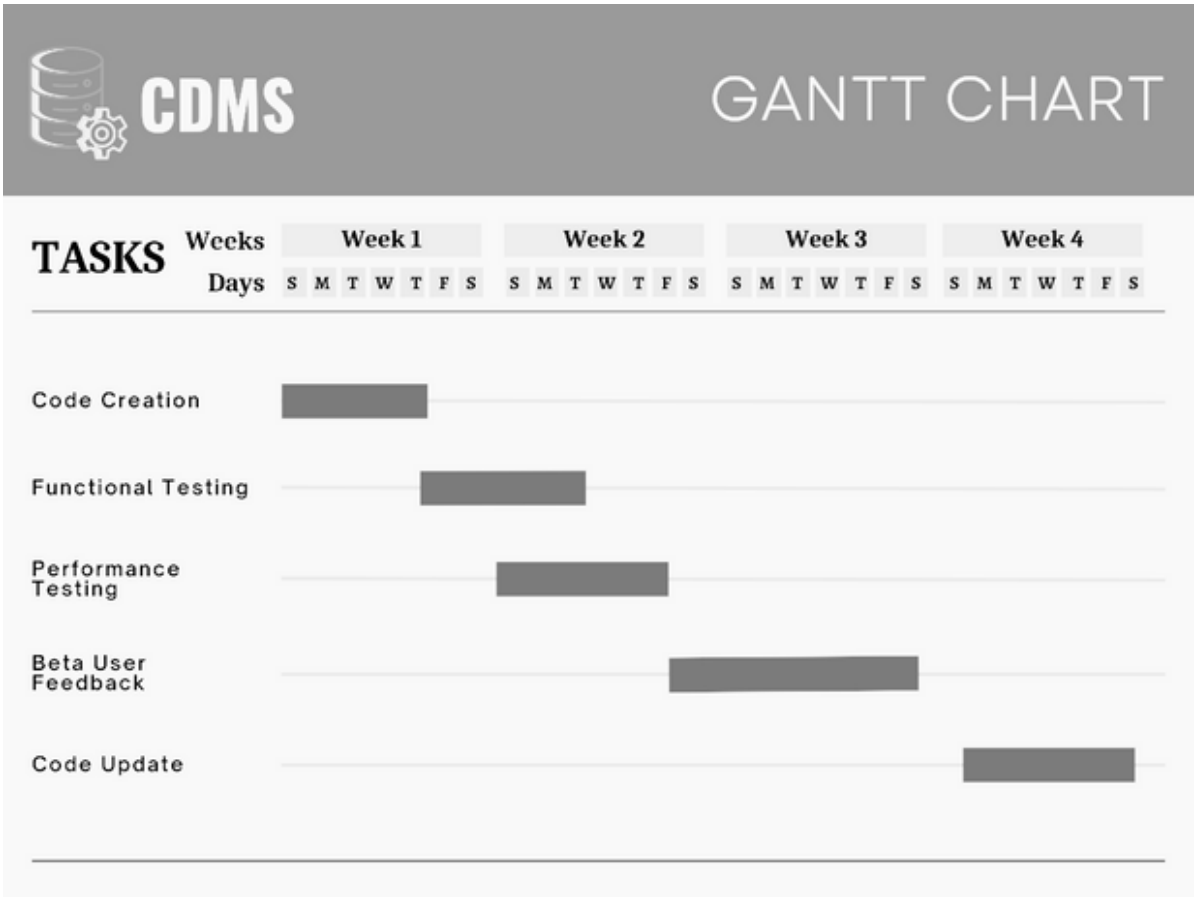


FIG 15

Summary

The "Implementation and Development" chapter covers the practical aspects of building the **CDMS** system. It includes sections on development technologies and tools, code implementation, deployment plan, future deployment considerations, and a Gantt chart. This chapter provides an overview of the implementation process and serves as a guide for understanding the development of the **CDMS** system.

The Conclusion and Future Work chapter serves as the culmination of the **CDMS** project. The chapter provides a summary of the key findings, outcomes, and accomplishments of the **CDMS** project. It reflects upon the insights gained throughout the documentation and presents recommendations for future enhancements and developments. It encompasses the following sections:

1. Conclusion
2. Future Work Recommendations

5.1 Conclusion

This section highlights the main results and achievements of the project. It shows how a **CDMS** successfully organizes and facilitates a data storage and custody system. While summarizing the results of the project, it also acknowledges any limitations or areas for improvement.

CDMS has proven to be a valuable solution in enhancing custody and instrumentation processes. By performing a comprehensive system analysis and design, we successfully identified and documented requirements, evaluated the existing system, and designed a more efficient and user-friendly recording system.

Throughout the project, we achieved several important milestones. A comprehensive requirements elicitation process ensured that the system met the diverse needs of administrators, system users, and IT users. The

implementation and development phase involved advanced techniques and tools, creating a robust and scalable system.

A **CDMS** has proven effective in simplifying the enrolment process, improving data accuracy, and enhancing the overall user experience. It has facilitated communication and collaboration between departments, administrators and IT users, resulting in efficient course selection and timely warnings.

However, it is important to acknowledge that a **CDMS** may have some limitations and areas for improvement. Further improvements can be made to improve system scalability, integrate additional functionality and improve performance. Additionally, ongoing maintenance and support will be crucial to ensuring the system's long-term success and adaptability.

5.2 Future Work Recommendations:

To further enhance the capabilities and effectiveness of **CDMS** , the following future action recommendations have been identified. These recommendations are intended to expand functionality and improve the user experience of the system. By implementing these improvements, the system can better meet the evolving needs of departments, users, and IT users. Here are the main areas for future development:

- **QR CODE SCAN:** With QR Code Scanner, you can easily embed a QR code into your website. When a person visits the site, they can use their smartphone to scan the QR code and get their personal information and other details. You can also use this tool to show off your product or service.

- **Mobile Application:** This mobile application is an extension of the existing web application. The purpose of this application is to make it easy for users to access information and resources and products, services and programs.

- **MICROSERVICE:** Emigrate monolith application to microservices application.

These future business recommendations aim to further improve the efficiency, automation, and user experience of the **CDMS**. By implementing these improvements, the system will continue to evolve and meet the changing needs of departments, administrators, administrative staff, and IT users. It is

essential to prioritize continuous development, testing and user feedback to ensure this.

Now that you have all the information on the website

We're done publishing, and it's time for that

Start playing it.

- SQL Notes for Professionals book is compiled from Stack Overflow Documentation
- Thinking Architecturally - Nathaniel Schutta
- Database Systems - Ramez Elmasri Department of Computer Science and Engineering The University of Texas at Arlington Shamkant B. Navathe College of Computing Georgia Institute of Technology

CDMS
Custody Database management system

CDMS

CUSTODYDATABASE MANGMENT SYSTEM

2024