A red square with white text

Description automatically generated

**Final Assessment – Project Work**

**Fundamentals of Data Science**

**Group Project**

Submitted By: Aditya, Resham, Aaradhya, Devyani

The British College, Kathmandu

Kathmandu, Nepal

Date:

**Final Assessment – Project Work**

Student management system – Report

**Title: Comprehensive Report on Student Management System**

* **Table of Contents**

**1. Introduction**

- Background

- Purpose of the System

- Scope

**2. System Overview**

- Description of the System

- Components and Features

- User Roles

**3. Design and Implementation**

- Architecture

- Data Flow Diagram

- Class Diagram

- Sequence Diagrams

**4. User Guide**

- How to Use the System

- Step-by-Step Instructions for Each Feature

**5. Testing**

- Test Cases and Scenarios

- Results and Analysis

**6. Security Measures**

- Password Management

- User Authentication

- Data Encryption

**7. Future Enhancements**

- Proposed Features

- Potential Improvements

**8. System Performance and Scalability**

- Performance Evaluation

- Scalability Analysis

**9. User Feedback and Usability Study**

- Feedback Collection Process

- Usability Study Methodology

- Analysis of User Feedback

**10. Legal and Ethical Considerations**

- Compliance with Data Protection Regulations

- Ethical Use of Student Data

**11. Maintenance and Support**

- Bug Tracking and Resolution

- Software Update Process

**12. Conclusion**

- Summary of Findings

- Future Outlook

**13. References**

1. Introduction

* Background:

The educational landscape has witnessed a significant shift towards digitalization, necessitating the development of robust systems for managing student data effectively. Traditional methods of record-keeping have become obsolete in the face of evolving technological advancements. Hence, the need arises for a modern Student Management System (SMS) that can streamline administrative tasks and enhance operational efficiency.

* Purpose of the System:

The primary purpose of this system is to provide a comprehensive platform for managing various aspects of student information, including academic records, extracurricular activities, and administrative tasks. By offering a user-friendly interface and incorporating advanced features, the system aims to revolutionize the way educational institutions handle student-related data.

* Scope:

This SMS encompasses a wide range of functionalities, including user registration, authentication, grade recording, extracurricular activity management, and administrative tasks. It caters to the diverse needs of students, faculty members, and administrative staff, providing tailored solutions for each user category.

2. System Overview

* Description of the System:

The Student Management System is a sophisticated software application developed using Python programming language. It operates through a command-line interface (CLI), providing users with an intuitive platform for accessing and managing student data. The system leverages text files for storing user credentials, grades, and extracurricular activity details, ensuring seamless data management and accessibility.

* Components and Features:

- User Registration and Authentication: Allows users to register and authenticate their credentials securely.

- Grade Management: Facilitates the recording and viewing of student grades for various academic subjects.

- Extracurricular Activity Management: Enables users to add, view, and manage extracurricular activities undertaken by students.

- Administrative Tasks: Provides administrators with tools for managing user accounts, including user registration and deletion.

* User Roles:

- Student: Can access their academic records, view grades, and manage extracurricular activities.

- Faculty Member: Has privileges to view student grades and extracurricular activities.

- Administrator: Possesses administrative rights to manage user accounts and perform system maintenance tasks.

Functions used in the Program:

register\_user(self)

* + Purpose: This method allows users to register as students.
  + Functionality: It prompts the user to input a username and password and confirms the password. It then checks if the passwords match and meet the length requirement before appending the username and password to the 'password.txt' file.

register\_admin(self)

* + Purpose: This method allows users to register as administrators.
  + Functionality: Similar to register\_user, it prompts the user to input a username and password, confirms the password, and checks for password match and length before appending the credentials to the 'adminpassword.txt' file.

add\_marks(self)

* Purpose: This method allows a\_\_init\_\_(self)
  + Purpose: This is the constructor method of the Project class. It initializes the attributes display\_marks and view\_eca with default values.
  + Functionality: Sets the initial values for the display\_marks and view\_eca attributes when a new Project object is created.

validate\_credentials(file\_path, username, password)

* + Purpose: This is a static method used for validating user credentials by checking them against records stored in a file.
  + Functionality: It takes a file path, username, and password as input parameters. It reads the file line by line, compares the username and password with stored credentials, and returns True if a match is found, indicating successful validation. Otherwise, it returns False.

login(self)

* + Purpose: This method handles the login process for users.
  + Functionality: It prompts the user to input their user type (student or admin), username, and password. Then it calls the validate\_credentials method to verify the provided administrators to add marks for students in different subjects.
  + Functionality: It prompts the administrator to input the student's name and marks for each subject (Data Science, IT, FOM, Academic English, ITF). Then it appends the marks to the 'grades.txt' file.

add\_eca(self)

* + Purpose: This method allows administrators to add extracurricular activity details for students.
  + Functionality: It prompts the administrator to input the student's username, sports, clubs, services, and additional activities. Then it appends these details to the 'eca.txt' file.

view\_eca(self)

* + Purpose: This method allows users to view extracurricular activity details for a specific student.
  + Functionality: It prompts the user to input the student's username and then reads the 'eca.txt' file to find and display the extracurricular activities associated with the given username.

delete\_user(self)

* + Purpose: This method allows administrators to delete a user's account.
  + Functionality: It prompts the administrator to input the username of the user they want to delete. It then reads the 'password.txt' file, removes the line containing the specified username, and writes the remaining lines back to the file.

display\_marks(self)

* + Purpose: This method allows users to view marks for all students.
  + Functionality: It reads the 'grades.txt' file and prints out the marks for each student in the format specified.

These functions collectively provide the functionalities needed for user authentication, registration, data manipulation, and administrative tasks within the Student Management System.

3. Design and Implementation

* Architecture:

The system follows a client-server architecture, where the client interacts with the server through the CLI interface. It adopts a modular design approach, with distinct modules responsible for handling different aspects of the system functionality. The modular design facilitates scalability, maintainability, and extensibility.

* Data Flow Diagram:

The data flow diagram illustrates the flow of information within the system, depicting how data moves between various components and external entities. It provides a visual representation of the data processing and interaction flow, aiding in system comprehension and analysis.

* Class Diagram:

The class diagram offers a visual representation of the system's structure, depicting the relationships between different classes and their attributes. It serves as a blueprint for the system design, guiding developers in implementing the system components and functionalities effectively.

* Sequence Diagrams:

Sequence diagrams capture the interaction between system components or objects during runtime, illustrating the flow of control and messages exchanged between them. They provide insights into the system's behavior and communication patterns, aiding in understanding and debugging system interactions.

4. User Guide

* How to Use the System:

- Users can initiate the system by selecting the appropriate action from the main menu, including login, registration, or administrative tasks.

- The system guides users through the various functionalities using intuitive prompts and instructions, ensuring a seamless user experience.

* Step-by-Step Instructions for Each Feature:

**1. Login:** Users must enter their username and password to authenticate their credentials and gain access to the system.

**2. Registration**: New users can register by providing their username and password, followed by a confirmation step to verify their identity.

**3. Viewing Grades**: Students and faculty members can view student grades by entering the student's name or username.

**4. Adding Grades:** Administrators can add grades by specifying the student's name or username and entering the respective marks for each academic subject.

**5. Managing ECAs:** Users can manage extracurricular activities by providing details such as sports, clubs, services, etc., associated with the student's profile.

**6. Administrative Tasks:** Administrators can perform administrative tasks such as user registration, deletion, and system maintenance through dedicated menu options.

5. Testing

* Test Cases and Scenarios:

- A comprehensive set of test cases covers various scenarios, including valid and invalid inputs, boundary cases, and error handling scenarios.

- Test scenarios simulate real-world usage scenarios to validate the system's functionality, performance, and reliability.

* Results and Analysis:

- Test results indicate the system's compliance with functional requirements, highlighting any discrepancies or issues encountered during testing.

- Analysis of test results provide insights into the system's robustness, scalability, and usability, guiding developers in refining the system's design and implementation.

6. Security Measures

* Password Management:

- Passwords are stored securely using industry-standard encryption techniques to prevent unauthorized access and ensure data confidentiality.

- Password policies enforce stringent requirements, including minimum length, complexity, and expiration, to enhance password security.

* User Authentication:

- The system employs robust authentication mechanisms, including username/password-based authentication and multi-factor authentication, to verify user identity and prevent unauthorized access.

- Authentication processes incorporate advanced encryption and hashing algorithms to safeguard user credentials against brute-force attacks and unauthorized access attempts.

* Data Encryption:

- Sensitive data such as user credentials, grades, and extracurricular activity details are encrypted using strong encryption algorithms to protect against data breaches and unauthorized access.

- Data encryption ensures data confidentiality and integrity, mitigating the risk of data leakage and unauthorized data manipulation.

7. Future Enhancements

* Proposed Features:

- Integration with relational databases for centralized data storage and enhanced data management capabilities.

- Implementation of role-based access control (RBAC) to enforce fine-grained access control policies and ensure data privacy and security.

- Development of a web-based user interface (UI) for improved accessibility and usability across different devices and platforms.

* Potential Improvements:

- Performance optimization measures, including database indexing, query optimization, and caching mechanisms, to enhance system responsiveness and scalability.

- Integration of advanced analytics and reporting

capabilities to enable data-driven decision-making and performance monitoring.

- Implementation of automated testing frameworks and continuous integration (CI) pipelines to streamline the software development lifecycle (SDLC) and ensure code quality and reliability.

8. System Performance and Scalability

* Performance Evaluation:

- Performance testing assesses the system's responsiveness, throughput, and resource utilization under varying load conditions.

- Key performance metrics, including response time, throughput, and resource consumption, are measured and analyzed to identify performance bottlenecks and optimize system performance.

* Scalability Analysis:

- Scalability testing evaluates the system's ability to handle increasing user load and data volume without compromising performance or stability.

- Scalability benchmarks assess the system's scalability characteristics, including horizontal and vertical scalability, elasticity, and resilience to spikes in user demand.

9. User Feedback and Usability Study

* Feedback Collection Process:

- User feedback is collected through surveys, interviews, and usability testing sessions to gather insights into user perceptions, preferences, and pain points.

- Feedback forms and questionnaires are distributed to users to solicit feedback on the system's usability, functionality, and overall user experience.

* Usability Study Methodology:

- Usability testing sessions involve real users interacting with the system while performing predefined tasks and scenarios.

- Observations, feedback, and user behavior are captured and analyzed to identify usability issues, interface design flaws, and areas for improvement.

* Analysis of User Feedback:

- User feedback and usability study findings are analyzed to identify common themes, patterns, and areas for enhancement.

- Usability issues and user pain points are prioritized based on severity and impact, guiding developers in addressing critical issues and improving overall system usability and user satisfaction.

10. Legal and Ethical Considerations

* Compliance with Data Protection Regulations:

- The system adheres to data protection regulations, including the General Data Protection Regulation (GDPR) and the Family Educational Rights and Privacy Act (FERPA), to ensure compliance with data privacy and security requirements.

- Data protection measures, including data anonymization, pseudonymization, and access controls, are implemented to safeguard sensitive student information and prevent unauthorized access or disclosure.

* Ethical Use of Student Data:

- Ethical considerations, including consent, transparency, and data minimization, are integral to the system's design and operation.

- User consent is obtained for data collection, processing, and sharing, and transparent data handling practices are followed to ensure user trust and confidence in the system.

11. Maintenance and Support

* Bug Tracking and Resolution:

- A systematic bug tracking and resolution process is implemented to identify, prioritize, and address software defects and issues.

- Bug reports are logged, triaged, and assigned to development teams for resolution, and bug fixes are verified through rigorous testing and quality assurance processes before deployment.

* Software Update Process:

- The system follows a structured software update process, including version control, release management, and change management, to ensure seamless deployment of software updates and patches.

- Software updates are tested in a staging environment before production deployment to mitigate the risk of introducing regressions or disruptions to system functionality.

12. Conclusion

* Summary of Findings:

- The Student Management System represents a significant advancement in educational technology, providing a comprehensive platform for managing student information effectively.

- Through its user-friendly interface, robust security measures, and advanced features, the system addresses the diverse needs of educational institutions and stakeholders, enhancing operational efficiency and data management capabilities.

* Future Outlook:

- With ongoing development and enhancements, the Student Management System is poised to evolve further, incorporating new features, technologies, and best practices to meet the evolving needs of the education sector.

- Continuous feedback, user engagement, and collaboration with stakeholders are essential for driving innovation and ensuring the system's continued relevance and effectiveness in a dynamic educational landscape.

13. References

- List of sources consulted during system development, including documentation, research papers, industry standards, and best practices.

---

This comprehensive report provides an in-depth analysis of the Student Management System, covering various aspects such as system design, implementation, security measures, scalability, user feedback, legal considerations, maintenance, and support. It serves as a valuable resource for stakeholders, developers, and policymakers involved in educational technology initiatives.

**Output:**

**Firstly, registering as a user/student**

**A black background with blue dots

Description automatically generated**

**Then, registering as an adminA black background with white text

Description automatically generated**

**Adding grades to the user/studentA black rectangle with white dots

Description automatically generatedA black rectangle with white dots

Description automatically generated**

**A black rectangle with white dots

Description automatically generated**

**Displaying the grade obtained by the user/student**

**A computer screen shot of a black screen

Description automatically generated**

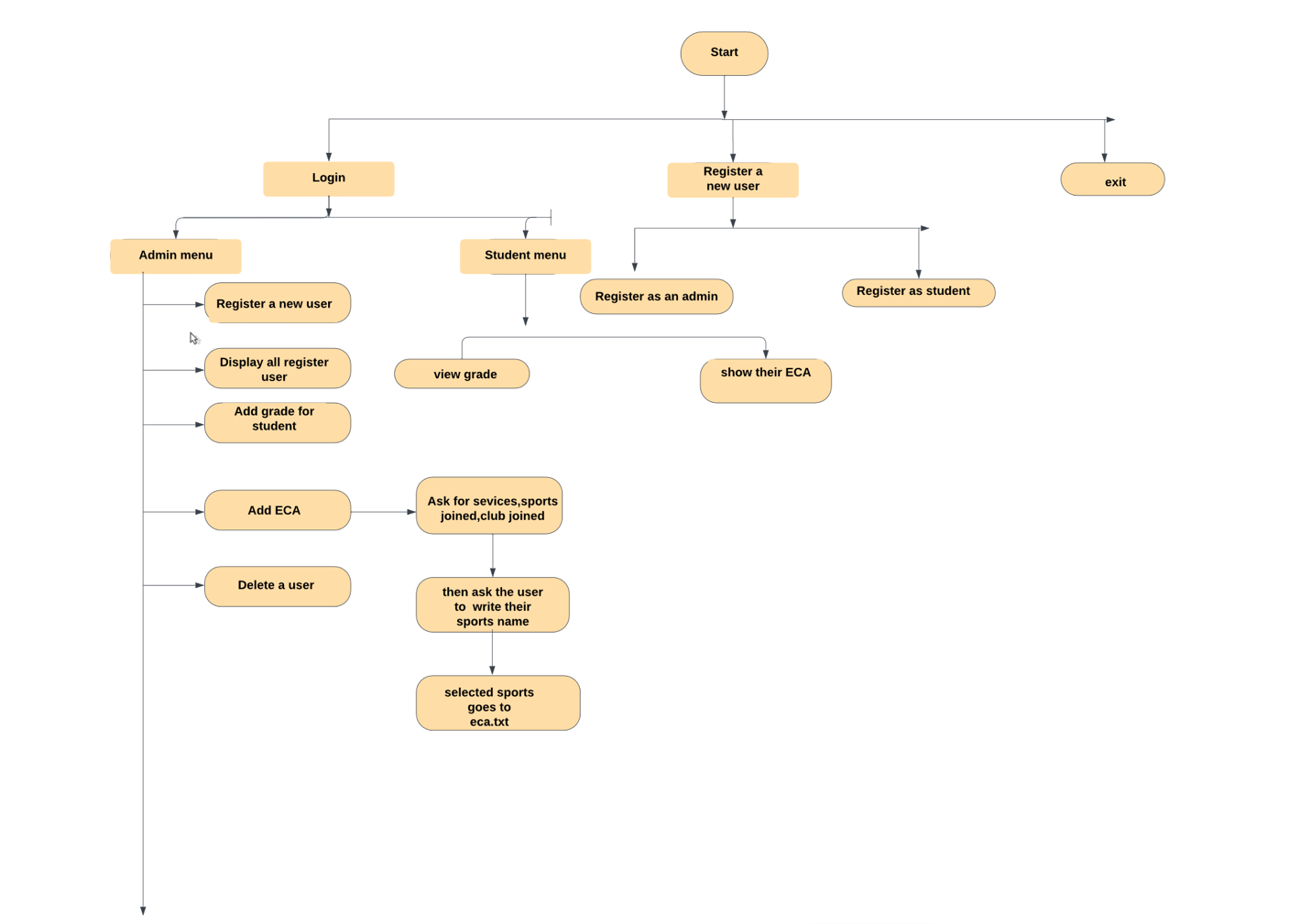
**Here, We haven’t added ECA details so that it shows “NO ECA details found”. A black screen with white text

Description automatically generated**

**If We have added ECA details, then it shows like thisA black rectangle with white dots

Description automatically generated**

**FLOWCHART HERE:**

****