FINAL PROJECT

SERVER SIDE INTERNET PROGRAMMING

Topic:

SIAKAD

"Sistem Informasi Akademik"

LECTURER:

Williem, M.Sc

Conducted by:

STUDENT OF INFORMATICS

Abdillah Algifary - 001202400076

PRESIDENT OF UNIVERSITY

Jl. Ki Hajar Dewantara, Kota Jababeka, Cikarang Baru, Bekasi

17550-Indonesia Phone (021) 8910 9762-6. Fax (021) 8910 9768

www.presuniv.com | Email: presuniv@gmail.com

TABLE OF CONTENT

CHAPTER I

INTRODUCTION

The rapid advancement of information technology has profoundly transformed various sectors, including education. Educational institutions, from schools to universities, are increasingly leveraging technology to enhance administrative efficiency, improve learning experiences, and facilitate better communication among stakeholders. In this context, a robust **Sistem Informasi Akademik (SIAKAD)** becomes an indispensable tool. A well-designed SIAKAD centralizes crucial academic data, streamlines administrative processes, and provides accessible information to students, lecturers, and staff. This final project report details the development of a server-side internet programming solution for a SIAKAD, aiming to address the growing need for a comprehensive, efficient, and user-friendly academic management platform.

1.1 Project Background

Traditionally, academic administration in many institutions has relied on manual processes, leading to inefficiencies, data inconsistencies, and delays. Managing student registrations, course enrollments, grade submissions, attendance records, and faculty assignments manually can be time-consuming, prone to errors, and labor-intensive. This often results in frustration for students seeking information, lecturers attempting to manage their courses, and administrative staff struggling with extensive paperwork.

The primary motivation behind this project is to overcome these challenges by developing a modern, web-based SIAKAD using server-side internet programming. This system will provide a centralized digital platform for managing all academic-related activities. By automating routine tasks and offering real-time access to information, the proposed SIAKAD aims to significantly improve the overall efficiency and effectiveness of academic operations. Furthermore, the project seeks to enhance transparency, reduce operational costs, and foster a more connected and informed academic community. The development of this server-side application will focus on creating a secure, scalable, and user-friendly system that can adapt to the evolving needs of an educational institution.

1.2 Objective

The primary objectives guiding the development of this SIAKAD prototype were meticulously defined to ensure a functional, secure, and user-centric system. Each objective contributes to the overarching goal of creating an effective academic data management solution:

- **To develop a secure user login system using PHP sessions:** This objective is crucial for ensuring that only authorized personnel can access and manipulate sensitive academic data. PHP sessions provide a robust mechanism for maintaining user state across multiple page requests, thereby securing access to different modules of the system after successful authentication.
- **To enable CRUD (Create, Read, Update, Delete) operations on academic data:** This forms the backbone of any data management system. Implementing CRUD functionalities allows administrators to fully control the lifecycle of academic records,

including adding new students or courses (Create), viewing existing data (Read), modifying incorrect or outdated information (Update), and removing obsolete entries (Delete). This ensures data accuracy and currency.

- **To calculate and display student GPA and course summaries:** Beyond mere data storage, the system aims to provide valuable insights. This objective focuses on implementing the logic required to automatically compute students' Grade Point Averages (GPA) based on their academic performance. Additionally, it involves generating summaries of course data, which can provide an overview of course enrollment, performance trends, or other relevant statistics.
- **To provide a secure and easy-to-use academic data management system:** This overarching objective emphasizes both the security and usability aspects. The system must not only protect sensitive information through authentication but also offer an intuitive and straightforward interface that minimizes the learning curve for users, allowing them to perform their tasks efficiently and without unnecessary complications

1.3 System Design and Architecture

The SIAKAD prototype adopts a straightforward, modular architecture, typical of many web applications built with PHP and MySQL. This design prioritizes simplicity and direct communication between components, making it manageable for a foundational project.

- **Frontend: HTML and Basic Inline CSS within PHP Files:** The user interface (UI) is primarily constructed using standard HTML5 markup, providing the structural foundation for all web pages. Styling is applied using basic inline CSS directly within the PHP files. While this approach offers immediate visual feedback during development, it is acknowledged that for larger, more complex applications, external CSS stylesheets and potentially CSS frameworks would be more maintainable and scalable. The inline CSS serves to provide fundamental visual separation and readability for the prototype.
- **Backend: PHP (Procedural Programming):** PHP serves as the core server-side scripting language, responsible for processing user requests, interacting with the database, and generating dynamic HTML content. The project utilizes a procedural

programming paradigm, where functions are called in a sequential manner to perform specific tasks. Each PHP file typically handles a distinct module or functionality, such as user authentication, data insertion, or report generation. This approach, while effective for smaller applications, can be refactored into an object-oriented structure for enhanced maintainability and scalability in future iterations.

- **Database: MySQL (Assumed Usage Based on Typical Architecture):** MySQL is the chosen relational database management system (RDBMS) for storing all academic data. It is a robust and widely used open-source database, well-suited for web applications. The database schema would typically include tables such as `users` (for login credentials), `students` (for student personal information), `courses` (for course details), and `grades` (linking students, courses, and their respective scores). PHP scripts interact with MySQL using the `mysqli` extension to execute SQL queries for data manipulation and retrieval.
- **Session Management: PHP Sessions using `$_SESSION`:** PHP's built-in session management mechanism, accessed via the `$_SESSION` superglobal array, is employed to maintain user state across different pages. Upon successful login, user-specific data (e.g., username, user ID) is stored in the session. This allows the system to verify user authentication on subsequent page requests without requiring re-authentication, ensuring a persistent and secure user experience throughout their interaction with the system.
- **Navigation Structure: A Centralized `index.php` Provides Links to System Modules:** The `index.php` file acts as the main entry point and dashboard for authenticated users. It serves as a central hub, providing clear navigation links to various modules of the SIAKAD, such as adding new records (`tambah.php`), editing existing data (`edit.php`), or accessing reports like student GPA (`rata_mahasiswa.php`) and course summaries (`rekap_mk.php`). This structure ensures intuitive access to all system functionalities.

- **Overall Architecture:** The system follows a simple client-server model. The client (web browser) sends requests to the PHP server. The PHP scripts on the server process these requests, interact with the MySQL database as needed, and then generate HTML responses that are sent back to the client for display. This direct communication model, while effective for the prototype, lays the groundwork for more complex architectures like Model-View-Controller (MVC) in future enhancements.

.

CHAPTER II

IMPLEMENTATION OVERVIEWS

The SIAKAD prototype is composed of several distinct PHP files, each dedicated to a specific functionality within the system. This modular approach enhances code organization and simplifies debugging.

- **`login.php`:** This file is the gateway to the system. It presents an HTML form for users to input their username and password. Upon submission, it processes these credentials, validates them against records stored in the database, and if successful, initiates a PHP session to register the user's authenticated state before redirecting them to the main dashboard (`index.php`).
- **`logout.php`:** Designed for secure session termination, this script explicitly destroys the active PHP session. This action effectively logs the user out of the system, clearing all session-related data, and subsequently redirects them back to the login page (`login.php`), ensuring that unauthorized access after a user session ends is prevented.
- **`index.php`:** Serving as the main dashboard and authenticated entry point, `index.php` first verifies the user's session status to ensure they are logged in. It then acts as the central navigation hub, providing a user-friendly interface with links to all other accessible modules of the SIAKAD, such as data entry forms, editing pages, and various academic reports.
- **`tambah.php`:** This module facilitates the creation of new academic records. It displays an HTML form where users can input details for new students, courses, or other relevant data types. Upon form submission, the script processes the `POST` data, performs necessary validation (though basic in this prototype), and then securely inserts the new record into the appropriate table in the MySQL database.

- **`edit.php`:** This file enables the modification of existing academic data. It typically receives a unique identifier (e.g., student ID, course ID) via the URL's `GET` parameters. Using this ID, it fetches the corresponding record from the database and pre-populates an HTML form with the existing data. Users can then make changes, and upon form submission, the script updates the relevant record in the MySQL database.
- **`rata_mahasiswa.php`:** This report generation module is responsible for calculating and displaying the average Grade Point Average (GPA) for students. It queries the database to retrieve student grade information, applies the necessary computational logic to determine the GPA for each student, and then presents these results in a clear, formatted HTML table for easy readability and analysis.
- **`rekap_mk.php`:** This script provides a summary of course-related data and performance statistics. It queries the database to gather information about courses, such as enrollment numbers, average grades for specific subjects, or other relevant metrics. The results are then displayed, often grouped by subject or course name, offering a concise overview of academic performance and resource utilization related to courses.

2.1 Testing and Evaluation

The SIAKAD prototype underwent a series of manual test cases to validate its functional correctness, stability, and adherence to the defined objectives. The testing process involved simulating typical user interactions and observing the system's responses.

- **Login Test:**
    - **Valid Credentials:** Inputting a pre-registered username and password successfully redirected the user to index.php, confirming authentication. Session variables were correctly set.
    - **Invalid Credentials:** Entering incorrect usernames or passwords resulted in an "Invalid username or password" error message, preventing access to the dashboard. This confirmed the basic security mechanism.

- **Add Data (e.g., Student Record):**
    - **Valid Input:** Filling out the "Add New Record" form (tambah.php) with complete and valid student details (NIM, Name, Major) successfully inserted the data into the students table in the database. A "New student record added successfully!" message was displayed.
    - **Incomplete Input:** Attempting to submit the form with empty required fields triggered a "All fields are required!" message, demonstrating basic client-side (HTML required attribute) and server-side validation.
- **Edit Functionality (e.g., Student Record):**
    - **Data Retrieval:** Navigating to edit.php?id=X (where X is an existing student ID) successfully pre-populated the edit form with the student's current data, fetched from the database.
    - **Update Accuracy:** Modifying existing student details in the form and submitting it correctly updated the corresponding record in the database. The system displayed a "Student record updated successfully!" message, and re-fetching the data confirmed the changes.
    - **Invalid ID:** Providing a non-existent ID in the URL resulted in a "Student not found" message, preventing attempts to edit non-existent records.
- **GPA Calculation (rata_mahasiswa.php):**
    - **Data Accuracy:** After populating the database with sample student and grade data, the rata_mahasiswa.phpreport correctly calculated and displayed the average scores for each student. These results were cross-checked against manual GPA calculations, confirming the accuracy of the aggregation logic.
    - **Edge Cases:** Tested with students having no grades (they would not appear in the report due to the JOINclause, which is expected behavior for this specific query).
- **Course Summary (rekap_mk.php):**
    - **Data Aggregation:** The rekap_mk.php report accurately displayed the total number of unique students enrolled in each course and the average score for each course, based on the sample data.
    - **Empty Courses:** Courses with no enrolled students or grades were correctly listed with '0' for student count and 'NULL' or '0' for average score (depending on database AVG behavior with no rows), due to the LEFT JOIN.

- **Security (Session Management):**
    - **Unauthorized Access:** Attempting to access index.php or other protected modules directly without logging in successfully redirected the user to login.php, confirming the session-based authentication mechanism.
    - **Logout Functionality:** Clicking the "Logout" button successfully destroyed the session and redirected to the login page, preventing further access without re-authentication.

The system met the functional expectations outlined in the objectives and demonstrated stability under the conducted manual test conditions. While the testing was not exhaustive (e.g., no automated testing, load testing), it confirmed the core functionalities of the SIAKAD prototype.

CHAPTER III

CONCLUSION

This final project successfully demonstrates the fundamental principles and practical application of server-side internet programming using PHP and MySQL for developing an Academic Information System (SIAKAD). The prototype effectively addresses the initial challenges associated with manual academic data management by providing essential functionalities for secure user authentication, comprehensive CRUD operations on student and course data, and valuable reporting capabilities such as GPA calculation and course summaries.

The modular structure of the system, with dedicated PHP files for each core function, contributes to its maintainability and extensibility, even within a procedural programming paradigm. While developed as a basic version, the SIAKAD prototype serves as a robust proof-of-concept, validating the feasibility of building efficient academic management solutions using readily available web technologies. Its successful implementation lays a solid foundation for future enhancements and expansion into a more feature-rich and production-ready system.

CHAPTER IV

FUTURE WORK

To evolve the SIAKAD prototype into a more robust, secure, and user-friendly production-ready system, several key enhancements are recommended for future development:

- **Implement Hashed Password Storage using `password_hash()`:** Currently, passwords are (assumed to be) stored in plain text or simple escape strings, which is a significant security vulnerability. Future iterations must utilize PHP's `password_hash()` function for storing passwords securely and `password_verify()` for authentication. This salting and hashing mechanism significantly protects user credentials against brute-force attacks and database breaches.
- **Enhance Front-end using CSS Frameworks like Bootstrap:** The current UI relies on basic inline CSS, which limits responsiveness, aesthetic appeal, and maintainability. Integrating a modern CSS framework like Bootstrap would provide a consistent, responsive, and visually appealing design. This would significantly improve the user experience across various devices (desktop, tablet, mobile) and simplify future UI development.
- **Add Database-Driven Dropdowns and AJAX for Smoother Interactions:** Many data entry forms (e.g., assigning grades to students, associating students with courses) could benefit from dynamic, database-driven dropdowns. Furthermore, incorporating AJAX (Asynchronous JavaScript and XML) would allow for partial page updates without full page reloads, leading to a much smoother and more interactive user experience, particularly for operations like search, filtering, or live validation.
- **Include Role-Based Access Control (RBAC) for Admins, Students, and Lecturers:** The current system assumes a single administrative user role. A critical enhancement would be to implement RBAC, differentiating access levels for various user types (e.g., administrators, students, lecturers). This would ensure that each user group can only access and modify data relevant to their roles, thereby enhancing security and tailoring the user experience.

- **Improve Input Validation and Error Handling:** While basic validation is present, a more comprehensive input validation strategy is needed to prevent invalid data from entering the database (e.g., ensuring NIMs are unique, grades are within a valid range). Robust error handling, including user-friendly error messages and logging, would make the system more resilient and easier to debug.
- **Implement a Deletion Functionality:** While CRUD implies Delete, it was not explicitly detailed. A dedicated module for safely deleting records (e.g., students, courses) with appropriate confirmation mechanisms would be essential.
- **Modularize Database Connection:** Instead of repeating database connection code in every file, create a separate `config.php` or `db_connection.php` file that can be included, promoting reusability and easier credential management.
- **Adopt an MVC (Model-View-Controller) Architecture:** For long-term maintainability and scalability, refactoring the procedural code into an MVC pattern would separate concerns (data logic, presentation, control flow), making the codebase more organized, testable, and easier for multiple developers to work on.