UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SECJ3303 INTERNET PROGRAMMING

GROUP PROJECT - SECTION 16

Final Project Report

Project Title: AchieveSync (School data management system)

Lecturer : Dr Faris Syahmi Samidi

Prepared by:

Team 5: NetNinjas

	Team Members	
NO.	**NAME**	**MATRICS NUMBER**
1.	Ahmad Ameerul Rasyid bin Hamidun	A22MJ5001
2.	RASHADUL NAFIS RIYAD	A22MJ3010

3.	Abdullah Al Nahid	A22MJ3066
4.	Asha	A22MJ8019

Table of contents Pages

1. Introduction (Background)

Educational institutions today face growing challenges in managing and organizing vast amounts of data efficiently, especially when relying on manual systems prone to errors and inefficiencies. At SK Saujana Utama, the current approach to managing school competition events and achievement records heavily depends on disparate tools like Telegram and Excel sheets. This fragmented system makes it difficult to retrieve specific information quickly, often resulting in redundant efforts and missed details. Recognizing these challenges, this report aims to introduce a comprehensive web-based platform designed to digitize and modernize the management of school-related data at SK Saujana Utama.The platform seeks to modernize data collection, storage, and reporting by offering a centralized system accessible to teachers and administrative staff.

1.1 System Objectives

1. **Centralized Data Management**:
    - To provide a single platform for storing and managing information about school competition events and school achievements (teachers and students) records in an organized and systematic manner.
2. **Role-Based Access Control**:
    - To ensure secure access by implementing user roles, such as admin and teachers, with permissions tailored to their responsibilities.
3. **Streamlined Data Entry and Retrieval**:
    - To simplify the process of adding, editing, and retrieving event/achievements information, including details such as event or achievements (names, date/time, participants, and related details).
4. **Efficient Reporting**:
    - To enable administrators to generate user activity logs, with an option to download them in pdf format for easy sharing and documentation.

5. **Responsive and Accessible Interface**:
    - To provide a user-friendly platform accessible from multiple devices, including desktops, tablets, and mobile phones, ensuring ease of use for all stakeholders.
6. **Enhanced Security**:
    - To safeguard sensitive school information through secure login, password recovery options, and data encryption.
7. **Scalability for Future Needs**:
    - To design a flexible system that can accommodate additional features or user roles as the school's requirements evolve.

8. **Real-Time Insights and Analytics**:

- To deliver actionable insights to stakeholders (e.g., admin, teacher) through an analytics dashboard and report pages.

1.2 Scope:

- The system targets **teachers (staff)** and **administrators** at SK Saujana Utama.
- It will manage data related to competition events and achievement records.
- Features include data input forms, search tools, reporting, and an analytics dashboard.
- The platform excludes parental access and focuses solely on internal staff usage.

2. Problem Statement

1. The data collection of the school is manually arranged, making it difficult to get information by categories such as school achievement records which includes teachers and student accomplishments.

2. Identify shortcomings and areas for improvement, including manual data entry, data duplication, and lack of integration.

2.1 Issues in current process

1. This process is time consuming and inefficient.

2. The form filling gets tedious and is repeated in the first proposal document and the final storage step.

3. Approval requires a new document to be drafted, which becomes tedious for different processes.

4. Data is scattered in many different forms making it extremely difficult to find and retrieve, as well as allowing for duplications.

5. Data is decentralized which means it can be easily lost and not found.

2.2 Possible solutions

1. Improve efficiency by using digital data entry formats which can provide shortcuts to document preparation.

2. Instead of preparing two documents for proposal and storage steps. The system can use information already entered in the proposal document to fill up the stored document relatively

3. Store data on the digital system eliminating duplications and making it easier to search for and retrieve.

4. Centralize data in the system instead of having 3 storage formats and locations.

3. Overview of Project

The NetNinjas Data Management System is a web-based application designed to streamline the management of school events and achievement records. The system addresses the challenges of manual data handling, providing a centralized platform for teachers and administrative staff to add, edit, delete, and view records efficiently.

Key features of the system include:

- **Event Records Management**: Enables users to store detailed information about school events, such as names, dates, venues, participant numbers, levels, and organizers where they can view,edit and delete the records if they want to.

- **Achievement Records Management**: Allows users to add,edit,view and delete student and teacher achievement records, including event names,achiever names, dates, positions, and certificates.

- **Role-Based Access**: Ensures secure access where teachers(staff) can input data while administrators oversee and manage the system (approve or reject the data by teachers).

- **Search Functionality**: Facilitates quick retrieval of specific records through keyword-based searching options.

- **Reporting and Analytics**: Administrators can generate comprehensive reports on user activity/logs, helping in strategic planning and evaluation. Also , through the dashboard both users(Admin and Teacher) have analytics tools such as graphs for better analysing the data.

This system leverages modern web technologies and the Spring MVC framework to offer a user-friendly interface, secure data handling, and robust functionality tailored to the needs of educational institutions. It transforms traditional methods into a more organized, efficient, and scalable solution.

3.3. Architecture Overview

The system will follow the Spring **Model-View-Controller (MVC)** design pattern, ensuring a clear separation of concerns:

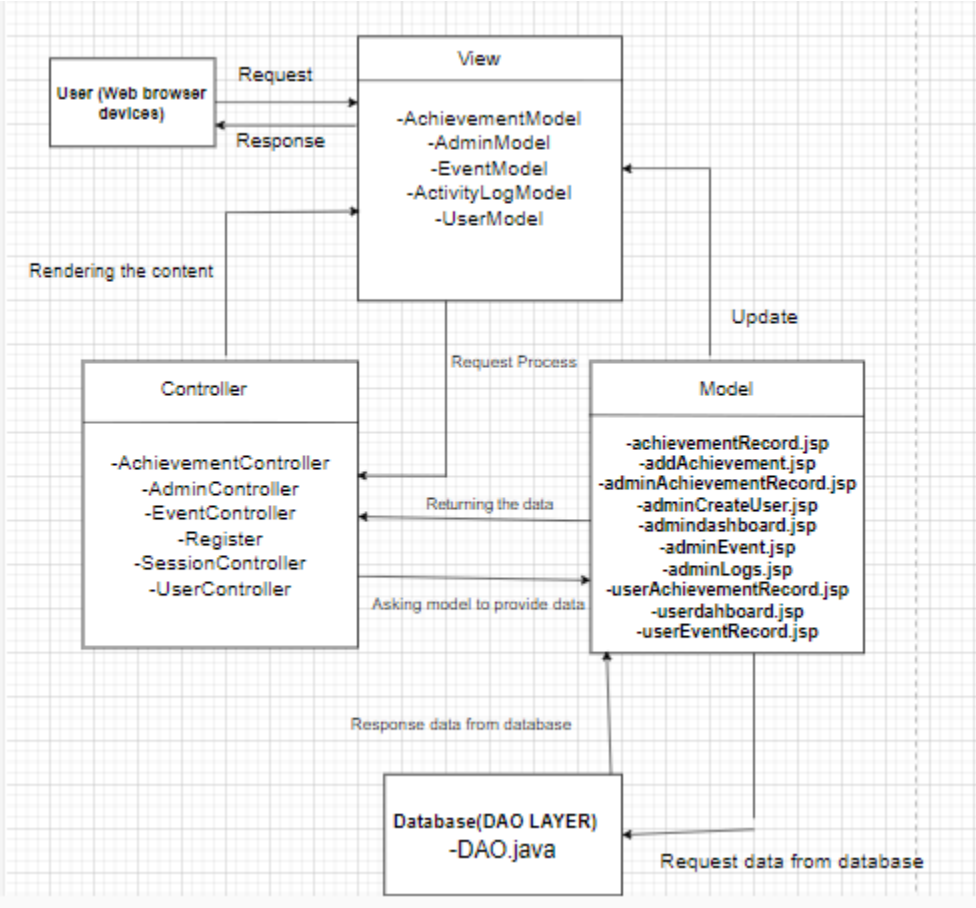
- **Model**: Represents the data and business logic (e.g., competition event details, user roles, achievement records).
- **View**: Handles the presentation layer using JSP templates.
- **Controller**: Manages user requests and determines the appropriate view or model to respond.

Technology Stack:

- **Backend**: Spring MVC.
- **Frontend**: HTML, CSS, JavaScript, and Bootstrap for responsive UI.
- **Database**: MySQL for relational data management.
- **Server**: Apache Tomcat embedded in SpringTool Suite.
- **Tools**: Maven dependency management and build automation.

4. System Design and Architecture

MVC Architecture Diagram



Data Flow Explanation

1. The User initiates a request by interacting with the View (e.g., submitting a form).
2. The Controller receives the request and processes it by invoking the Model.
3. The Model performs business logic, interacts with the Database if necessary, and sends the processed data back to the Controller.
4. The Controller updates the View, which renders the output to the User.

Description of System Components

1. User (Client-Side)

- **Description**: Represents the users of the system, including teachers and administrators, who interact with the application using web browsers on devices like laptops, tablets, or smartphones.
- **Role**:
    - Sends requests to the system (e.g., adding an event, viewing achievements).
    - Receives and views the processed responses from the server (e.g., event details, search results).

2. Controller

- **Description**: The controller acts as the central component that manages the flow of data and processes user input.
- **Responsibilities**:
    - Handles incoming HTTP requests (e.g., form submissions or URL requests).
    - Processes the requests by invoking the appropriate methods in the Service Layer of the Model.
    - Updates the View with the data processed by the Model.
- **Example**:
    - When a teacher submits an event creation form, the controller processes the data and updates the `events` table in the database.

3. View

- **Description**: The presentation layer of the system responsible for displaying data to the user.
- **Responsibilities**:
    - Receives data from the Controller and presents it to the user in a readable format.
    - Consists of JSP pages styled with HTML and CSS.
- **Example**:
    - The user sees a list of all events on the event records page (user`EventRecord.jsp`), complete with options to edit or delete records.

4. Model

- **Description**: The data and business logic layer of the application.
- **Subcomponents**:
    - **Service Layer**:
        - Implements business rules and processes the logic before interacting with the database.
    - **DAO (Data Access Object) Layer**:
        - Handles all interactions with the database.
        - Example: Executes SQL queries to retrieve event or achievement data.
    - **Business Logic**:
        - Core logic specific to the application (e.g., calculating the number of events/achievements in a month).
- **Role**: Acts as a bridge between the Controller and the Database.

5. Database

- **Description**: The backend data storage system, implemented using MySQL, where all system information is stored.
- **Tables**:
    - `users`: Stores login credentials and roles for teachers and admins.
    - `events`: Contains information about school events such as name, date, and venue.
    - `achievements`: Holds records of student and teacher achievements.
    - activitylogs: contains the timestamp,actor, action and object.
- **Role**:
    - Ensures data persistence.
    - Responds to CRUD operations (Create, Read, Update, Delete) initiated by the DAO.

Database Schema

1. **users Table**

Column Name	**Data type**	**Description**
id	Int (primary key, auto increment)	Unique identifier for each user
username	text	Username for login
password	text	password
role	text	User role defining access level(Admin,Teacher)

2. **achievements Table**

Column Name	**Data type**	**Description**
id	iINT (Primary Key)	Unique identifier for each achievement
achieverName	text	Name of the achiever
eventName	text	Name of the associated event
date	date	Date of the event
position	text	Position achieved (e.g., Winner, Runner-up
represents	text	Level of representation (e.g., School
pic	text	Person in charge of the event
certificates	text	Pdf of certificate given
status	text	Pending/Approved status of the records.

submittedby	text	References `users.id` (Who created the achievement)

3. **Events Table**

Column Name	**Data type**	**Description**
id	INT (Primary Key)	Unique identifier for each event
eventName	text	Name of event.
eventDate	date	Date of the event
venue	text	Location of the event
level	text	Event level (e.g., School, District)
noOfParticipant	int	Total number of participants
represents	text	School
organizer	text	Organizer of the event
status	text	Pending/Approved status of the records.
submittedby	text	References `users.id` (Who created the event)

4. **activitylogs Table**

Column Name	Data type	Description
timestamp	timestamp	When the action occurred
actor	text	User role (Admin/Teacher)
action	text	Description of the action (e.g., "Added Event"
object	text	Description of the activity

5. Module Functionality and Actor

Example:

Member	Module	Model: DAO, Database	View: JSP, CSS, Bootstrap	Controller: CRUD	Personalization & Authorization (session/filter)	Reporting (Transaction, History)
		Technical implementation of MVC			Project features	
Member A	Module 1 – User Authentication	User DAO, Login Database	(Login, Create User Pages)	User Login, Session Controller	Session Management, Role-based Authorization	N/A
Member B	Module 2- Data Storage & Management	Data DAO, Storage Database	(Upload, Manage Pages)	Data CRUD Controller	Admin,user Control for Access Levels	N/A
Member C	Module 3 – Data Retrieval	Retrieval DAO, sQuery Database	(Search Results)	Search Controller	User/Admin specific Search	N/A
Member D	Module 4- Analytics & Reporting	Analytics DAO, Reporting Database	(Dashboard, Reports Pages)	Analytics Controller	Admin control , User control	User activity reports(logs), Export reports to various formats (PDF, Excel)

Module 1 – User Authentication

Functionality:

- Handles secure login and registration processes for system users.
- Supports role-based authorization to ensure only authorized users can access specific system functionalities.

Actor(s):

- **Admin**: Oversees user roles and authorizations. Can create and manage user accounts.
- **Teachers**: Register and login to access the system. Their access level is determined by assigned roles.

Module 2 – Data Storage & Management

Functionality:

- Allows users manage (add,edit,delete) , and store various types of data related to school competition events and achievement details.
- Provides admin controls to monitor and regulate data submissions and updates.

Actor(s):

- **Admin**: Supervises data submissions, approves/reject records, and manages access levels for different users.
- **Teachers**: Upload, update, manage and organize data, such as competition event information and achievements records (teachers and students).

Module 3 – Data Retrieval

Functionality:

- Enables users to search and retrieve stored data effectively.
- Implements robust search system by (event name, type,date,competition level etc) tools to enhance the data discovery experience based on user-specific needs.

Actor(s):

- **Teachers**: Search and retrieve data relevant such as event details, pdf files, previous documents and achievements info.
- **Admin:** Search and retrieve pending event and achievement records that need to be approve/reject.

Module 4 – Analytics & Reporting

Functionality:

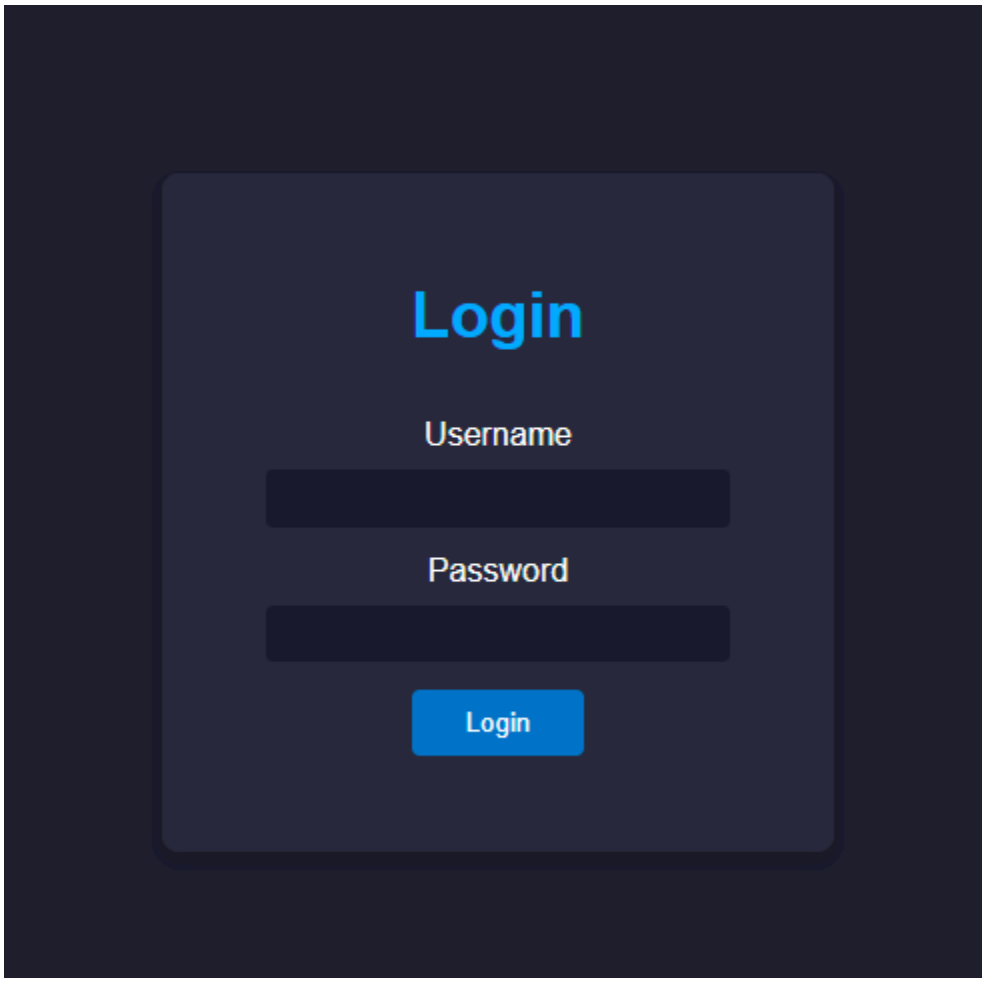
- Generates analytics dashboards to present insights from the data.
- Creates reports on user activities, daily logins, or other metrics.
- Supports exporting reports in multiple formats, such as PDF or Excel, for external use.

Actor(s):

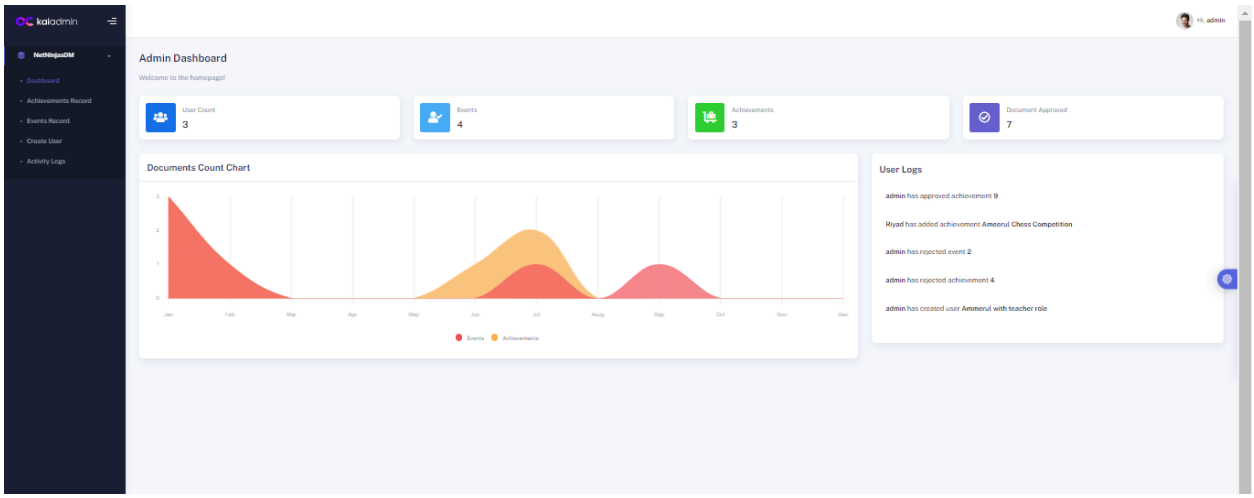
- **Admin**: Accesses and analyzes system-wide reports. Download and export files in pdf/excel format like user activity reports.
- **Teachers:** Users can view an analytics dashboard that represents insights from the data.

6. Final Interface

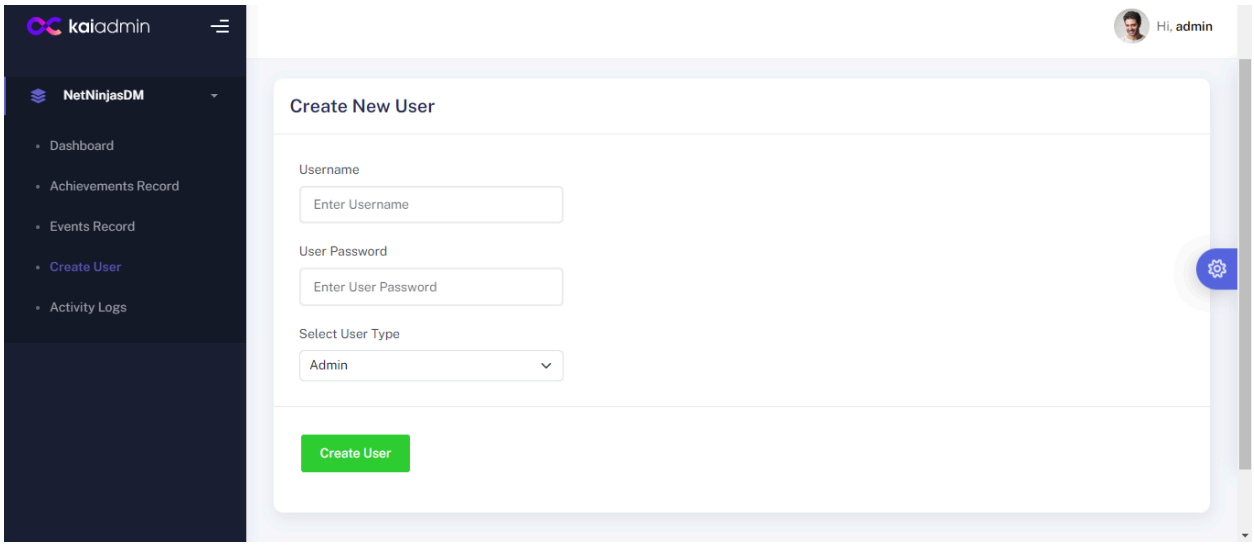
User login for both(Admin and teacher)



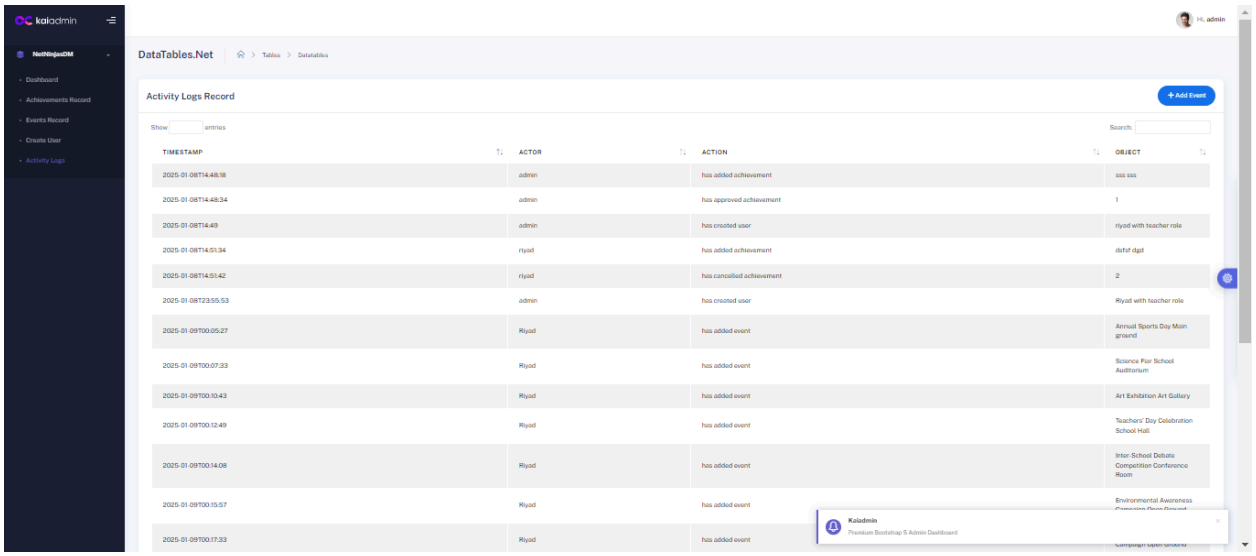
Admin Dashboard



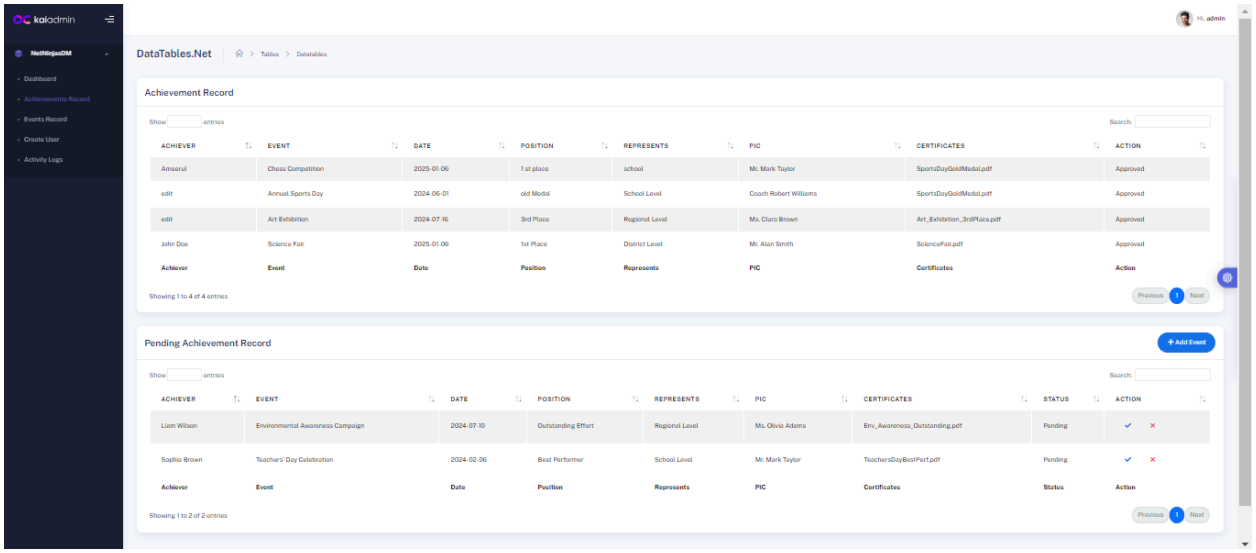
Admin Create User



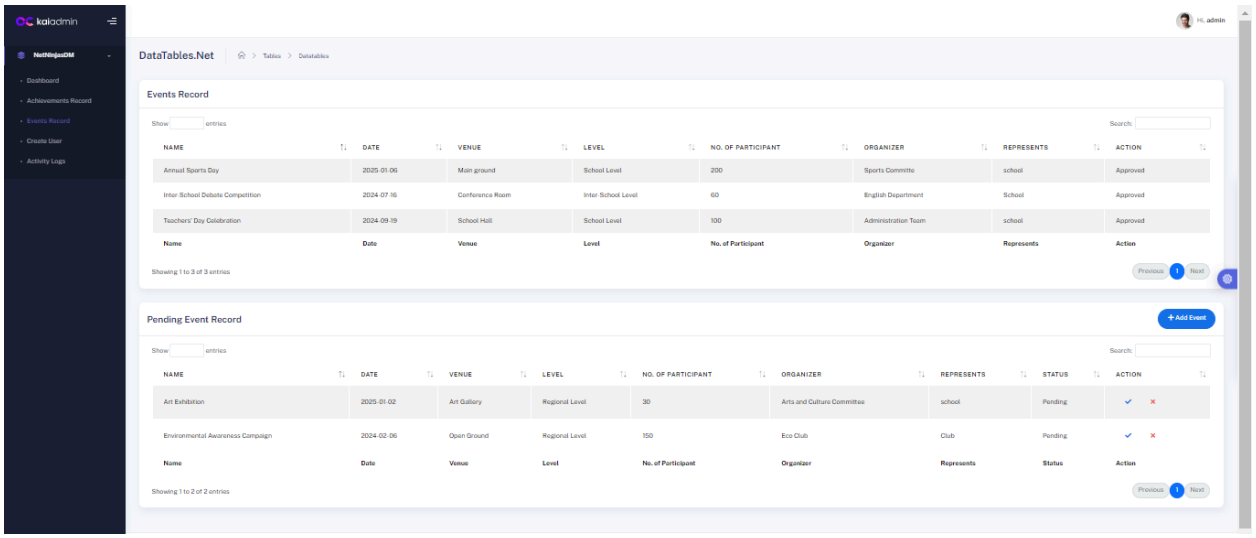
Admin Activity logs



Admin Achievement Record

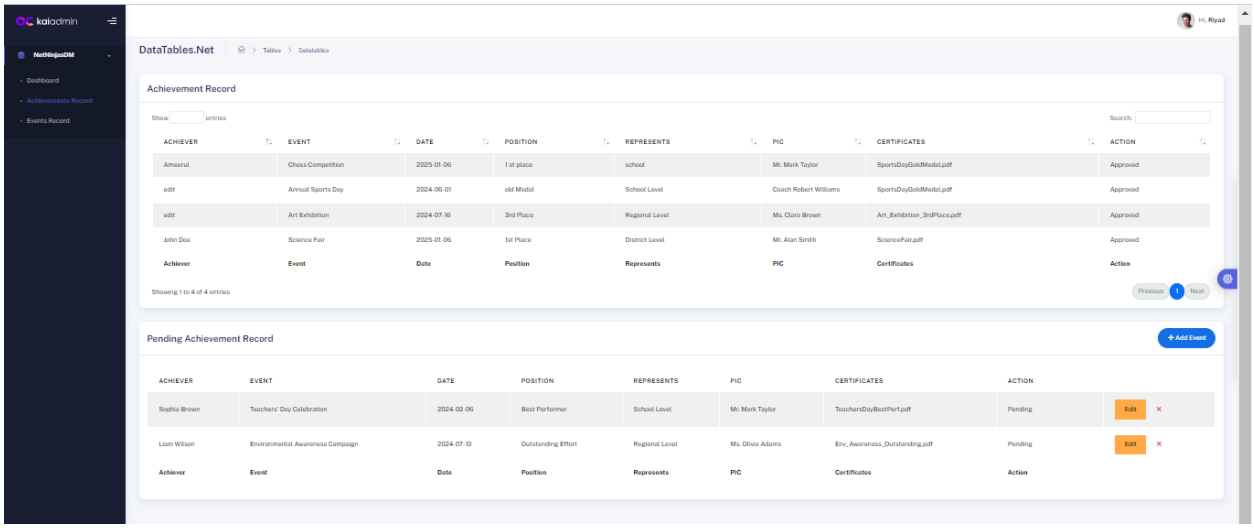


Admin Event Record

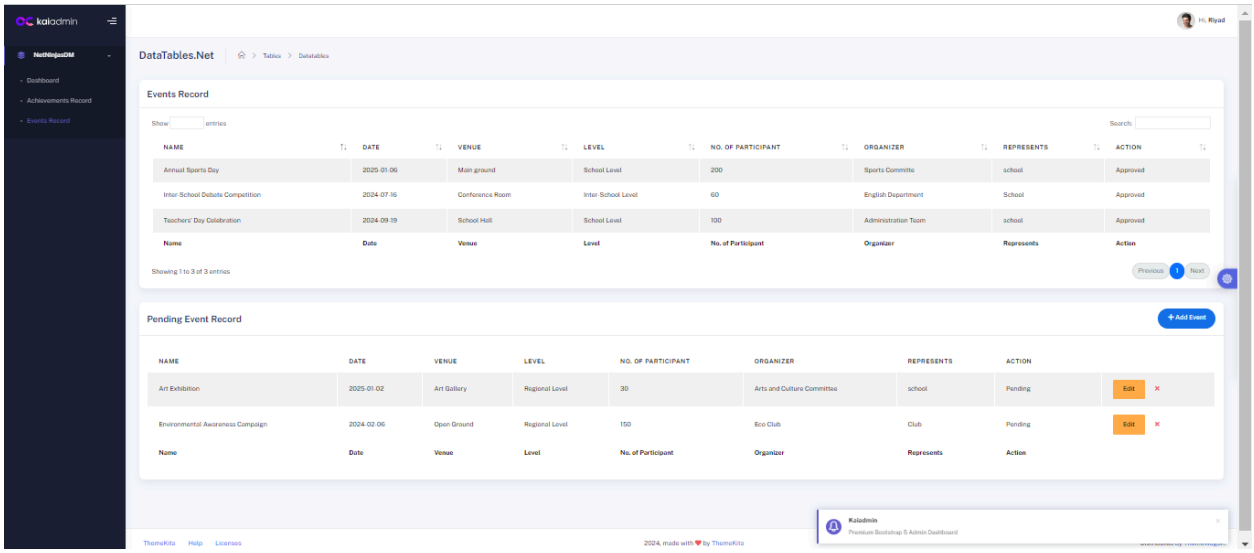


User Dashboard

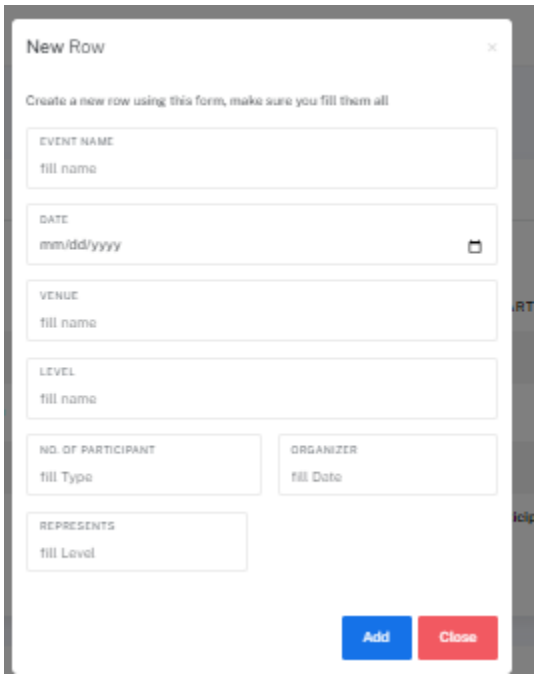
User Achievement Record



User Event Record



Form


New Row
Create a new row using this form, make sure you fill them all
EVENT NAME
fill name
DATE
mm/dd/yyyy
VENUE
fill name
LEVEL
fill name
NO. OF PARTICIPANT
fill Type
ORGANIZER
fill Date
REPRESENTS
fill Level
Add
Close

7. Implementation Details

SessionController

```java
    @RequestMapping("/")
    public String checkSession(HttpSession session) {
        // Check if a session attribute for the logged-in user exists
        String username = (String) session.getAttribute("loggedinusername");
        String role = (String) session.getAttribute("loggedinrole");


        if (username != null) {
            System.out.println("username not null" + "logedinusername is " + username);



            System.out.println("username not null" + "loggedinrole is " + role);
            if ("admin".equalsIgnoreCase(role)) {
                return "redirect:/admindashboard"; // Redirect to admin home

            } else {
            //if ("teacher".equalsIgnoreCase(role))
                return "redirect:/userdashboard"; // Redirect to teacher home
            }
            // User is logged in, redirect to home page
           // return "redirect:/dashboard";
        } else {
            System.out.println("username is null");
            // User is not logged in, redirect to login page
            return "/login";
        }
    }
```

This code snippet is a Spring MVC controller method that handles user redirection based on session attributes. When the root URL (`/`) is accessed, it checks the `HttpSession` for the attributes `loggedinusername` and `loggedinrole`. If the username exists, indicating the user is logged in, the method redirects to either the admin dashboard (`/admindashboard`) or the user dashboard (`/userdashboard`) based on the user's role. If no session exists, it redirects the user to the login page (`/login`). The flow ensures secure session management and role-based access, providing a seamless user experience. Debugging statements log the current session details for verification.

AdminController(Approve achievement)

```
@RequestMapping(value = "/adminApproveAchievement", method = RequestMethod.POST)
public String approveAchievement(@RequestParam("id") int id, HttpSession session) {
    System.out.println("approvee: ");
    DAO.approveAchievementById(id);

    DAO.savelog((String) session.getAttribute("loggedinusername"), " has approved achievement", " " + id );
    return "redirect:/adminAchievementRecord"; // Redirect to the list page
}
```

This code snippet defines a Spring MVC controller method that handles the approval of achievements by an admin. When the endpoint `/adminApproveAchievement` is accessed via a POST request, the method retrieves the achievement ID from the request parameters and calls the `DAO` method to approve the achievement in the database. It also logs the action, associating it with the currently logged-in admin by retrieving their username from the session. Finally, the method redirects the user to the admin achievement records page (`/adminAchievementRecord`) to display the updated list of achievements. This ensures efficient handling and tracking of achievement approvals by administrators.

Admin Logs records

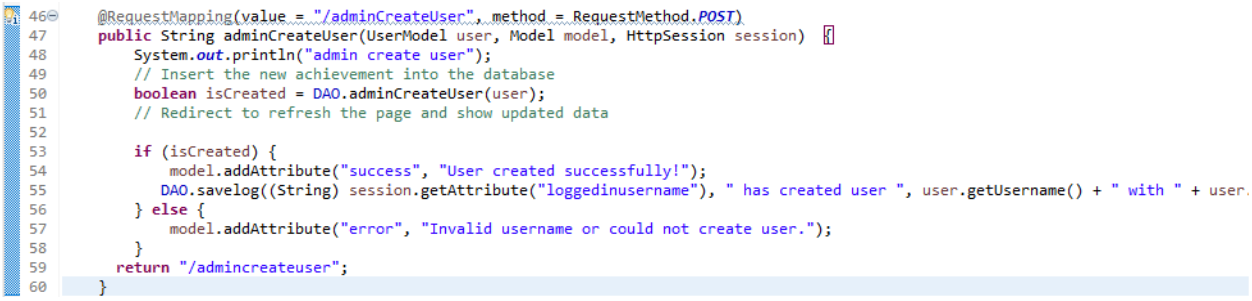
```
@RequestMapping(value = "/adminLogsRecord", method = RequestMethod.GET)
   public String adminLogsRecord(Model model)  {

       model.addAttribute("activitylogs", DAO.getAllLogs()); // approved items

     return "adminLogsRecord";
   }
```

This code snippet defines a Spring MVC controller method that retrieves and displays activity logs for the admin. When the endpoint `/adminLogsRecord` is accessed via a GET request, the method fetches all logs from the database by invoking the `DAO.getAllLogs()` method. The retrieved logs are added to the `Model` as an attribute named `activitylogs`. The method then returns the name of the view (`adminLogsRecord`), which renders the logs on the corresponding page, providing the admin with a detailed record of all system activities.

Admin Create User



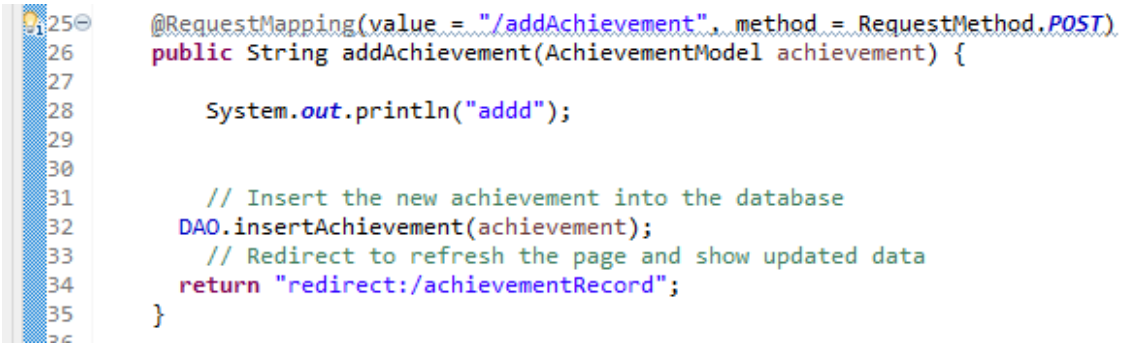
```
@RequestMapping(value = "/adminCreateUser", method = RequestMethod.POST)
public String adminCreateUser(UserModel user, Model model, HttpSession session) {
    System.out.println("admin create user");
    // Insert the new achievement into the database
    boolean isCreated = DAO.adminCreateUser(user);
    // Redirect to refresh the page and show updated data

    if (isCreated) {
        model.addAttribute("success", "User created successfully!");
       DAO.savelog((String) session.getAttribute("loggedinusername"), " has created user ", user.getUsername() + " with " + user.
    } else {
        model.addAttribute("error", "Invalid username or could not create user.");
    }
  return "/admincreateuser";
}
```

This code snippet defines a Spring MVC controller method that allows an admin to create a new user. When the `/adminCreateUser` endpoint is accessed via a POST request, it processes the data submitted through the form, represented by the `UserModel` object. The method calls `DAO.adminCreateUser(user)` to attempt the user creation in the database.

If the operation is successful, a success message is added to the `Model` and a log entry is saved, indicating which admin created the user and their assigned role. If the operation fails, an error message is added to the `Model`. Finally, the method returns the view name `/admincreateuser`, ensuring the admin receives feedback on the operation while staying on the user creation page.

UserController(add achievement)



```
@RequestMapping(value = "/addAchievement", method = RequestMethod.POST)
public String addAchievement(AchievementModel achievement) {

    System.out.println("addd");


    // Insert the new achievement into the database
  DAO.insertAchievement(achievement);
    // Redirect to refresh the page and show updated data
  return "redirect:/achievementRecord";
}
```

This code snippet is a Spring MVC controller method that handles adding a new achievement to the database. When the `/addAchievement` endpoint is accessed via a POST request, the method receives an `AchievementModel` object populated with the form data submitted by the user. It calls the `DAO.insertAchievement()` method to save the achievement details in the database. After successful insertion, the method redirects the user to the `/achievementRecord` page to display the updated list of achievements. This approach ensures efficient data handling and a seamless user experience.

UserController(user dashboard)

```
@RequestMapping("/userdashboard")
public String admindashboard( HttpSession session, Model model) {


 model.addAttribute("userapprovedachievementcount", DAO.getuserApprovedAchievementsCount((String) session.getAttribute("loggedinuser
 model.addAttribute("userapprovedeventcount", DAO.getuserApprovedEventsCount( (String) session.getAttribute("loggedinusername") ));
 model.addAttribute("achievementbymonths", DAO.getAchievementsByMonth());
 model.addAttribute("eventbymonths", DAO.getEventsByMonth());

    System.out.println("Logged in as: " + session.getAttribute("loggedinusername"));
    // Return home view if the user is logged in
    return "userdashboard"; // JSP or HTML page for the home
}
```

The `admindashboard()` method handles the request to the user dashboard (`/userdashboard`), fetching user-specific data such as the count of approved achievements and events, as well as monthly breakdowns of achievements and events, and adds them to the model for display. The method also prints the logged-in username for debugging purposes. The `adminAchievementRecord()` method retrieves and displays a list of approved and pending achievements for the logged-in user, adding these lists to the model for rendering the `userAchievementRecord` view. Both methods use `HttpSession` to access the logged-in user's data.

8. Future Enhancements

Possible Improvements in Design:

1. **Advanced User Interface**: Incorporate more interactive and visually appealing designs using modern UI frameworks.
2. **Mobile Application**: Develop a dedicated mobile app to increase accessibility.

Possible Functional Enhancements:

1. **Real-Time Notifications**: Integrate push notifications for updates on approvals and upcoming events.
2. **AI-Powered Insights**: Use AI to provide predictive analytics and trends based on historical data.
3. **Integration with External Systems**: Allow synchronization with third-party platforms for broader functionality (e.g., Google Calendar integration).
4. **Gamification**: Introduce gamified elements to boost engagement among teachers and admins.

9. Conclusion

The AchieveSync project was an enriching experience that addressed real-world challenges in data management for SK Saujana Utama. Through the development of this centralized web-based platform, we enhanced the efficiency of handling competition and achievement records. The project emphasized the importance of teamwork, planning, and user-centered design. It demonstrated how technology can transform traditional manual processes into streamlined, scalable solutions, paving the way for future enhancements and broader applications in educational institutions.